NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF CHEMISTRY

SIMULATION, OPTIMIZATION AND EXPERIMENTAL VALIDATION OF AN IMPROVED VERSION OF THE GRADIENT STEADY STATE RECYCLE (GSSR) PROCESS

TIAGO PEREIRA DOS SANTOS

Master/BSc in Chemical and Biochemical Engineering

DOCTORATE IN SUSTAINABLE CHEMISTRY

NOVA University Lisbon
April, 2024

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
CHEMISTRY

SIMULATION, OPTIMIZATION AND EXPERIMENTAL VALIDATION OF AN IMPROVED VERSION OF THE GRADIENT STEADY STATE RECYCLE (GSSR) PROCESS

TIAGO PEREIRA DOS SANTOS

Master/BSc in Chemical and Biochemical Engineering

Adviser: José Paulo Barbosa Mota
Full Professor, NOVA School of Science and Technology, NOVA University Lisbon

Examination Committee

Chair: Doctor José Luis Capelo Martinez
Full Professor, NOVA School of Science and Technology, NOVA University Lisbon

Rapporteurs: Doctor Carlos Manuel Santos da Silva
Associate Professor, University of Aveiro

Doctor Ricardo Jorge Sousa da Silva
Researcher, Institute of Experimental and Technological Biology (IBET)

Adviser: Doctor José Paulo Barbosa Mota
Full Professor, NOVA School of Science and Technology, NOVA University Lisbon

Members: Doctor Ana Mafalda Almeida Peixoto Ribeiro
Auxiliary Professor, Faculty of Engineering, University of Porto

Doctor Rute de Almeida Ferreira Castro
Researcher, Institute of Experimental and Technological Biology (IBET)

Doctor José Luis Capelo Martinez
Full Professor, NOVA School of Science and Technology, NOVA University of Lisbon

Doctor Ana Belén Pereiro Estévez
Principal Researcher, NOVA School of Science and Technology, NOVA University Lisbon

DOCTORATE IN SUSTAINABLE CHEMISTRY

NOVA University Lisbon
April, 2024

Simulation, Optimization and Experimental Validation of an Improved Version of the Gradient Steady State Recycle (GSSR) Process

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.27) [1].

Acknowledgements

First and foremost, I would like to thank my supervisor, Professor José Paulo Mota, for his support, technical help, good energy and words of encouragement throughout these 6 years. His efforts to provide me good working conditions, networking opportunities in conferences and internships will always be remembered. Also, he gave me free reigns so I can develop my autonomy and was always there to lift me when I failed, immensely speeding up my growth.

Secondly, I want to thank the NOVA School of Science and Technology for finantial support and for giving me a place to work during this time. I also want to thank the Foundation for Science and Technology for finantial support though the PhD grant PD/BD/142951/2018, it was essential for this work.

Thirdly, I want to express my gratitude to the Lund University in Sweden, especially the Department of Chemical Engineering, which harbored me for 5 months, 2 of those were during the pandemic. They let me stay in their facilities, gave me a place to work and with amazing conditions, even though they did not benefit at all from my stay there. Their help in those troubled times will never be forgotten.

Fourthly, I want to thank my parents for unending support, both emotional and finantial. Without them, I would not be able to endure these last 6 years and deliver this work in its completion. Also want to thank my grandparents, who already passed away, for all the finantial support given, specially paying my tuition fees.

Last but not least, I want to thank all my close friends, with all the co-working sessions, deep conversations, emotional support, words of encouragement, it was priceless the help they gave me to be able to support all my PhD, specially the last 2 years.

Thank you all, I will never forget you.

”

‘Tudo é ousado a quem a nada se atreve’ (Fernando Pessoa)

“

Abstract

Chromatography plays a major role in the downstream processing of biopharmaceuticals. Due to its high specificity, is also one of the major expenses in the production train, that is why it is a process in constant development. Continuous chromatography, compared to batch, achieves the same degrees of purity and has much higher yields, with the trade-off of having a higher investment cost and an increase in process complexity. The Gradient Steady State Recycle (GSSR), developed by our research group, is a 3-column semi-continuous process for central-cut separation of ternary and pseudo-ternary mixtures, comprised of 8 steps. This work aims to use state-of-the-art computational tools to improve the GSSR process previously reported, to develop an experimental set-up that can mimic any 3-column or less chromatographic process and to develop a control and automation software in the *Julia* language. The numerical optimizations performed showed that the GSSR can be simplified from 8 to 6 steps, and in certain conditions in can be further simplified to only 4 steps. Furthermore, comparing to the original GSSR problem, our optimizations predict a productivity increase of about 4-fold, a concentration ratio increase of about 5-fold and a solvent consumption decrease of about 10-fold. Lastly, we used a ternary mixture of Uridine, Guanosine and Tryptophan and tested our experimental set-up using the 4-step GSSR process and show that the simulations predict the experimental results with a good degree of accuracy, which further proofs the robustness of the mathematical tools created in-house.

Keywords: chromatography, continuous chromatography, gradient steady state recycle, GSSR, mathematical optimization, system modeling

Resumo

A cromatografia tem um papel crucial no processo a jusante de biofármacos. Devido à sua alta especificidade, é um dos processos mais custosos na linha de produção, o que faz com que esteja em constante desenvolvimento. Cromatografia contínua, comparada com descontínua, atinge os mesmo graus de pureza com graus de recuperação muito maiores, com a desvantagem do investimento inicial ser muito grande e a complexidade do processo ser maior. O Reciclo com Gradiente em Estado Estacionário (RGEE), desenvolvido pelo nosso grupo de investigação, é um processo de 3 colunas semi-contínuo para separação de corte central de misturas ternárias ou pseudo-ternárias, composto por 8 passos. Este trabalho tem o objetivo de utilizar avançadas ferramentas computacionais para melhorar o RGEE, desenvolver um protótipo experimental que consiga mimetizar qualquer processo cromatográfico composto por 3 ou menos colunas e desenvolver um software de controlo e automação de todos os equipamentos instalados no protótipo na linguagem *Julia*. As otimizações numéricas que foram feitas mostram que o RGEE pode ser reduzido de 8 para 6 passos, e em determinadas condições pode ser ainda reduzido para apenas 4 passos. Mais ainda, comparando com o problema original do RGEE, as nossas otimizações prevêem uma produtividade 4 vezes superior, um rácio de concentração 5 vezes superior e um consumo de solvente 10 vezes inferior. Por fim, utilizámos uma mistura ternária de Uridina, Guanosina e Triptofano e usámo-la para testar o nosso protótipo experimental usando a versão de 4 passos do RGEE e mostramos que as simulações desenvolvidas conseguem prever os resultados experimentais com um bom grau de precisão, o que valida a robustez das ferramentas matemáticas criadas no nosso grupo.

Palavras-chave: cromatografia, cromatografia contínua, reciclo com gradiente em estado estacionário, RGEE, otimização matemática, modelação de sistemas

Contents

List of Figures

List of Tables

Acronyms

BD Backwards Differentiation *(p. 51)*
BDex Blue Dextran *(p. 105)*

CD Central Differentiation *(p. 52)*
CIP Cleaning In Place *(p. 25)*
CSS Cyclic Steady State *(p. 77)*
CST Continuously Stirred Tank *(p. 40)*

DOF Degrees of Freedom *(p. 49)*
DSP Downstream Process *(p. 14)*

EtOH Ethanol *(p. 75)*

FA Frontal Analysis *(p. 109)*
FD Forward Differentiation *(p. 51)*

GA Genetic Algorithm *(pp. 72, 79)*
GP Genetic Programming *(p. 72)*
GSSR Gradient Steady State Recycle *(p. 26)*

HETP Height Equivalent to a Theoretical Plate *(p. 38)*
HPLC High Pressure Liquid Chromatography *(p. 7)*

IgG Immunoglobulin G *(p. 23)*

KPIs Key Performance Indicators *(p. 86)*

LDF Linear Driving Force *(p. 39)*

MCSGP Multi-column Counter-current Solvent Gradient Purification *(p. 21)*
MetOH Methanol *(p. 120)*

OC Orthogonal Collocation *(p. 52)*

PCC Periodic Counter-current Chromatography *(p. 24)*
POI Product of Interest *(p. 16)*

SMB Simulated Moving Bed *(p. 18)*

TMB True Moving Bed *(p. 17)*

USP Upstream Process *(p. 14)*

1 Introduction

This Chapter will cover all the themes necessary for the reader to understand the novelty of this work.

It will explain the fundamentals of chromatography, a little bit of its scientific history and the principles of adsorption that rule the chromatographic process. It will also talk about the ways to run a chromatographic process, either in batch or continuous mode and the evolution of the continuous operations.

1.1 Adsorption

Adsorption is the main phenomenon that rules this entire work. This phenomenon is in the middle of everything achieved in this project, so it is very important to explain the basics behind this process.

1.1.1 Main Principles

In its fundamental, adsorption happens when a component, that can be a gas or a liquid, interacts with the surface of another component, either a liquid or a solid, and adheres to it. The component that adheres is called adsorbate and the component that "offers" the surface to be adhered to is called adsorbent. This phenomenon, called adsorbent-adsorbate interaction, can be made in several ways, some strong and some weak. The most common adsorbent-adsorbate interactions are electrostatic or Van der Waals interactions, which are reversible. There are irreversible ones too, which usually involve electron transfer between them, such as chemical adsorption [2].

Figure 1.1 graphically explains the adsorption and absorption phenomena. While the absorption phenomenon implies that a phase change occurs (i.e. a gas touches a liquid and becomes a liquid), the adsorption phenomenon is just at the surface, where the component adheres to it but does not change their state.

The adsorption phenomena can be manipulated in ingenious ways, depending entirely on the adsorbent-adsorbate interaction. For instance, a very popular adsorbent is the Zeolite 13X, a solid adsorbent used for removing nitrogen from the air [4]. It has a very

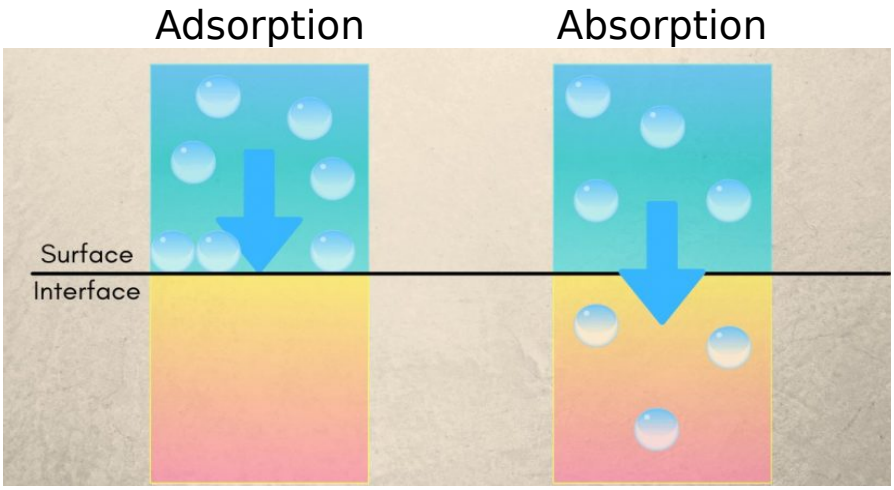

Figure 1.1: Scheme of the adsorption and absorption phenomena. Adapted from [3].

high surface area[1] and a very high selectivity towards nitrogen[2]. The result is a gas with a very high oxygen percentage, due to the fact that the nitrogen got stuck in the Zeolite 13X.

But what actually drives the adsorption to take place? Continuing with the Zeolite 13X example, even though it has high affinity towards nitrogen, there is still a need to force the gas to contact with the solid in some way. That "way" is called a driving force. This driving force can be any physical property, such as pressure, temperature, electric current, etc. The trick is to find out which driving force is best for the adsorbent-adsorbate combination that you are working with.

There are extensive literature reviews regarding adsorption processes in all kinds of industry [5–9]. This shows that, despite being such a simple process, is still widely used in the today's industrial world.

The example above is about gas adsorption, when a gaseous component adsorbs on top of an adsorbent, in that case a solid. The next topic will be about liquid adsorption, specifically when a liquid component interacts with a solid adsorbent.

1.1.2 Liquid Adsorption

The main difference between liquid and gas adsorption is that liquids are incompressible. To stimulate the adsorption of a gas you usually use pressure as a driving force. By increasing the pressure, meaning that you increase the number of molecules in a certain fixed volume, the probability that a molecule adsorbs increases, since you are decreasing the distance between the gas and the solid surface. However, in a liquid, increasing the pressure does not yield such outcome. In this case, the best driving force that you can use with liquids is concentration. Increasing the concentration of a substance in a solution

[1] A high surface area means that the adsorbent has a very high area available for the adsorbate to adhere.
[2] high selectivity towards nitrogen means that the adsorbent is much more inclined to interact with the nitrogen than with the oxygen.

means introducing more molecules of said substance in solution, increasing the chance that those molecules arrive at and interact with the solid surface.

1.1.2.1 An Example

Until now it was covered several fundamentals of adsorption, so now let us introduce an example. Let us imagine that a solid adsorbent called SALTY exists and was developed exclusively to adsorb sodium chloride, NaCl, the adsorbate, and remove it from liquid solutions. Separately we have, in a jar, 1L of water where 0.5g of NaCl was added and stirred it until the salt is completely dissolved and the solution is homogeneous.

After the preparation of the salt solution with a concentration of 0.5g/L of salt, 1g of SALTY is added to the jar and continuously stirred so the adsorbent is homogeneously spread inside the jar for a long period of time, say 1 day. After that time period, if the SALTY actually did its job, then the concentration of NaCl in the water has lowered substantially (if NaCl adheres to the adsorbent, then the concentration in the liquid decreases). That is when we call Mr. John, which is an expert in determining salt concentration in water just by tasting it. According to his palate, he determines the salt concentration in the end of the experiment to be 0.25g/L. This means that SALTY removed 0.25g of NaCl from the water.

This is the most common experimental protocol for running an adsorption experiment, simply prepare a solution with the adsorbate inside, add the adsorbent in the solution and wait for it to remove the adsorbate.

1.1.3 Adsorption Isotherm

Let us pick again on the example described in Section 1.1.2.1. It is known that, if the concentration of NaCl in the liquid is 0.5g/L, SALTY can remove 0.25g from solution. But what if the concentration in the liquid was different, for instance 2.0g/L? Assuming the exact same experiment was run but with the concentration stated before. Again Mr. John was called to taste the water in the end of the experiment and he determined a concentration of 1.5g/L. This means that the adsorbent removed 0.5g of NaCl from the water, which was more than before. Why did this happen? Simply because the driving force was modified. In the first experiment, SALTY had 0g of salt adsorbed in it and concentration of the mother liquor was 0.5g/L, while in the second experiment SALTY had 0g of salt adsorbed in it and the concentration was 2.0g/L. As explained in Section 1.1.2, the most common driving force used for liquid adsorption is concentration, and by increasing concentration we increase the probability of salt molecules to interact with SALTY.

In the end, after several experiments were made, only changing the initial concentration, a plot like Figure 1.2 can be created. But now a question arises, why does the curve reach a maximum value? As shown in the plot, the increase in adsorbed mass is not linear, but tends to a maximum value, in this case ~1.8. This happens because of the surface

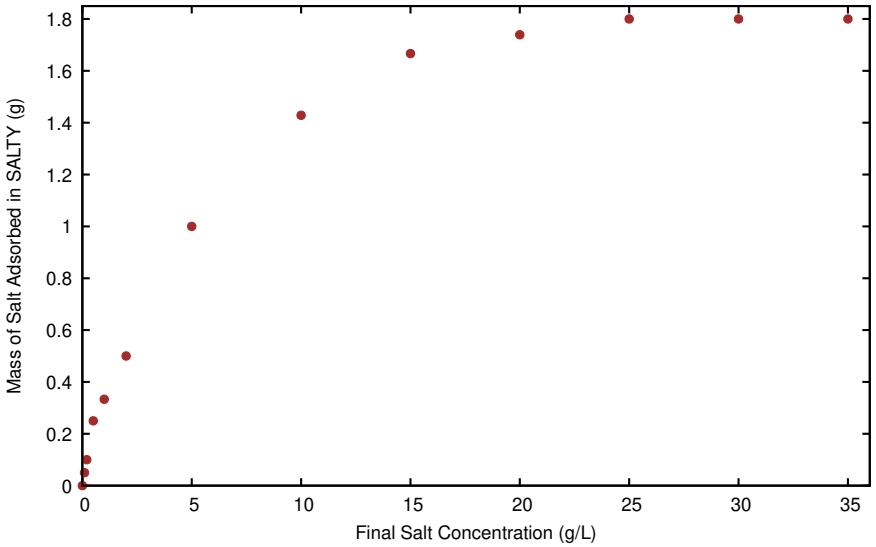

Figure 1.2: Results after several adsorption experiments. The higher the initial concentration, the higher the amount of adsorbed mass.

area available in SALTY. Every adsorbent has a fixed surface area and can adsorb a finite amount of a substance. In this case, 1g of SALTY can only adsorb roughly 1.8g of NaCl. That value is called the saturation point. If such a value had to be overcome, an increase in the amount of SALTY added in the experiments had to be made.

The curve in Figure 1.2 is usually called an adsorption isotherm. Isotherm is named because of the fact that these experiments are run at the same temperature. After performing such experiments with specific concentrations, a function can be estimated that would fit the experimental data. Such a function is called an isotherm model.

1.1.4 Concluding Remarks

In the end, the adsorption experiments in batch mode have limited applications, since this involves having the adsorbent in suspension and in the end it has to be removed from the liquid by precipitation. But what if the adsorbent could be put inside a container, for instance a column, trap it and let the solution pass through it? That will be covered in the next section.

1.2 Batch Chromatography

Chromatography is a more advanced way of performing adsorption experiments as the ones explained in Section 1.1.2.1. This section will cover the basic principles, the main advantages and disadvantages and some examples.

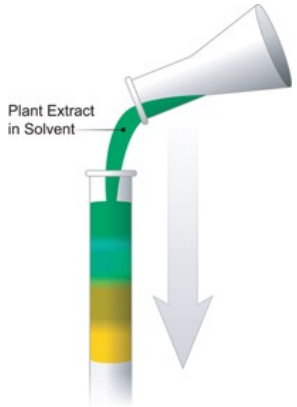

Figure 1.3: Tswett Experiment [12].

1.2.1 Main Principles

Chromatography follows the same principles as adsorption, but instead of having the adsorbent in suspension inside the liquid, it is trapped inside a column and instead the liquid is forced to go through the column. In this way, since the adsorbent is fixed, there is no need to remove it from the solution by precipitation. Furthermore, the physical proximity of the adsorbate and adsorbent is significantly reduced since the solution is forced to go through a tight volume.

1.2.1.1 The Name "Chromatography"

The name chromatography was given by Mikhail Tswett, a Russian botanist, when in 1902 he separated plant pigments by letting them pass through a column filled with calcium carbonate [10, 11]. He used an organic solvent to extract the pigments from said plants and obtained a mixture. The mixture was introduced inside the column and, by gravity, the solution passed through it. The result was the separation of the pigments, resulting in a column divided in sections of different colors, since the pigments were traveling at different speeds. Figure 1.3 shows a schematic of the experiment performed by Tswett. In the image we can see the pigments being separated, since each pigment has a different affinity towards the column, making them leave the column at different timings. The name chromatography came from this experiment, a remark of the colors observed in the column.

1.2.1.2 The Stationary Phase

Now another question arises. How does the liquid pass through a tightly packed column of a solid adsorbent? Because the adsorbent is porous. It was previously mentioned what the surface area was, but not explained how can this property be manipulated. Usually, when an adsorbent is manufactured, almost always comes as spheres of very small dimensions (diameter values are usually in the micrometer scale). The most important

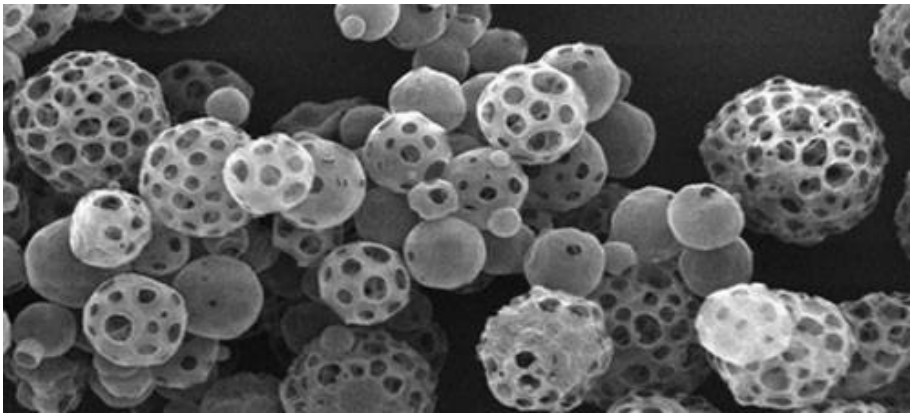

Figure 1.4: Porous particles [13].

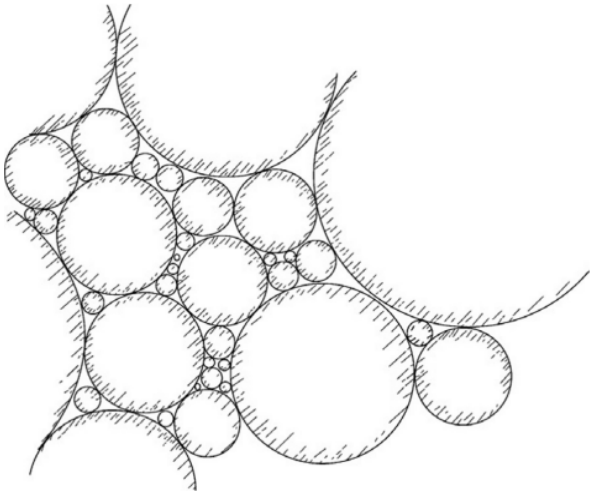

Figure 1.5: Typical pore size distribution. Adapted from [16].

part of those spheres is that they have an internal volume that represents a significant percentage of the total particle volume. Figure 1.4 shows a microscopic picture of what an adsorbent looks like. It is clearly seen the void inside the beads, which means a bigger surface area in total.

Another point that is important to talk about is the pore size. Theoretically, the bigger the surface area the better, but if the size of the pores are too small the molecules will not enter them. In the other hand, if they are too big then the surface area decreases (bigger pores mean less pores, which logically means a smaller surface area). Before any experiment, the properties of the molecules to be separated have to be carefully studied, so we can choose the best adsorbent available for our experimental system. Ideally we would want that every pore of every bead would be the same. However, that is impossible in practice. In the current days, manufacturing companies can create beads with a very low pore size distribution. This means that, although the pores will be different, they can reduce the size range to a very small one [14, 15].

Figure 1.5 is just a graphical representation of what actually lies inside the packed

column. It is practically impossible to have all particles with the exact same size, which means that, the higher the discrepancy, the harder it is to predict what is going to happen inside the column. That is why technologies that reduce this discrepancy are in high demand.

There is also a threshold regarding the bead size. As you can imagine, the smaller the particles, the bigger the number of particles we can pack inside a column, so the surface area will be bigger [17]. However, if the beads are too small, the force needed to push the fluid through the column increases significantly, and can reach a point that the beads or the column itself are damaged by such pressure. Regardless, using small beads was very advantageous because the process efficiency had massive increases with reducing the bead size [17]. In this way, the technology used to create more resistant beads and columns evolved and that created a new field of chromatography, the High Pressure Liquid Chromatography (HPLC), which is a topic that will be discussed later in further detail.

1.2.1.3 The Process of Separation

Now, one might ask how can such a process separate components based on adsorption and selectivities. If the liquid is forced to go through the column, either by pressure or simply by gravity, components with bigger selectivity will have more affinity for the adsorbent, hence moving slower through the column. In this way, we can collect the outlet of the column in the time spans correspondent to each component.

Let us pick on the example given in Section 1.2.1.1. After the entire volume is injected, the column has to be cleaned because the adsorbent is saturated with the pigments. In the end there is a need to inject pigment-free solution, so the concentration difference makes the pigments leave the adsorbent and go to the solution, regenerating it. The regeneration is also affected by the selectivities, so we can also collect pure pigments even during desorption. In the end, in both injection and desorption steps there are some time spans in which we can collect the components with the desired purity.

Another thing worth discussing is the solvent itself. For us to be able to separate the components by adsorption we must make sure that the solvent does not adsorb as well. This is important because the mass of solvent is much higher compared to the mass of pigments in solution, and if the solvent is also adsorbed, it would occupy the entirety of the surface area available for adsorption, hence preventing the pigments to adsorb.

With this example one can understand the biggest advantage in regards to adsorption versus chromatography: while adsorption experiments can remove one or more components from a solution, chromatographic experiments can remove and separate more than one component, based on the different selectivities of each component.

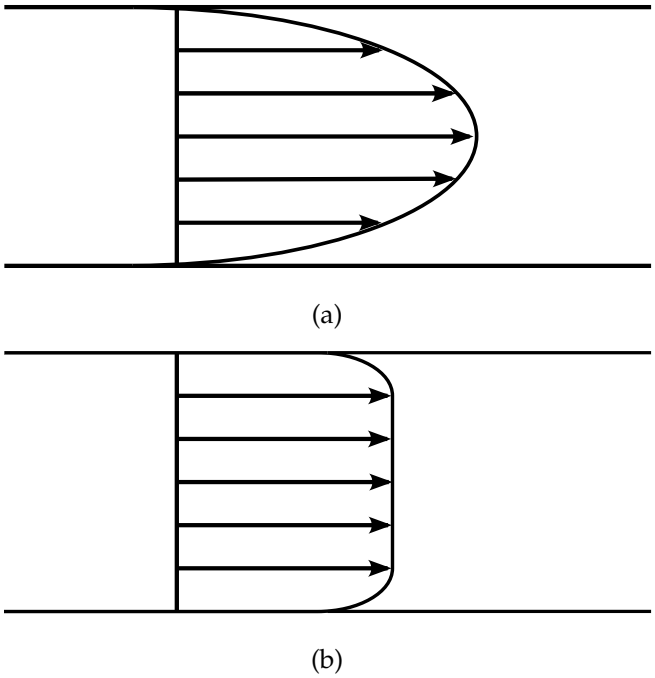

Figure 1.6: Velocity profiles of a fluid inside a column. Profile (a) happens when the fluid is traveling at low speeds and (b) happens when the fluid travels at high speeds.

1.2.2 Behavior of a Fluid Inside a Packed Column

In an ideal situation, the liquid particles injected inside a column will all travel at the same speed, but in practice that is not the case. The closer the particles are from the column wall, the slower they actually move. This is due to friction between the particles and the wall, that is why the further away they are from the wall, the faster they move, with the maximum speed being in the middle of the column [18, 19].

Figure 1.6 has a scheme of two typical velocity profiles. Figure 1.6a represents the typical velocity profile of a fluid that is flowing at slow speeds. In this case, the profile is parabolic, with the particles touching the wall being at null speed and the ones in the center of the column traveling at the highest speed. This is normal when gravity is the force that is making the liquid move. Figure 1.6b shows how the velocity profile of a fluid that is traveling at high speeds is presented. The faster the fluid moves, the more the profile is flattened, comparing to Figure 1.6a. This is usually achieved when we put pressure in the fluid being injected, much like the piston in a syringe.

This information is quite relevant because we want the fluid transport to be as homogeneous as possible in order for the chromatographic process to be as efficient as possible. If the adsorption phenomenon is occurring in the same conditions in every point in the column, whether the particles are close or far away from the wall, the process efficiency increases and our ability to predict the outcomes increases as well.

1.2.3 Why Batch?

When we say that a chromatographic process runs in batch mode, it means that the solution we want to pass through the column has to be processed in a discontinuous manner. Using Tswett experiment as an example, if we prepared 10L of solution and wanted to pass the entire volume through the column in a single step, we would need a much bigger column than Tswett had. This is explained, again, by the fact that the adsorbent has a maximum capacity for the adsorbates, and if the main solution possesses more than that limit, the process does not work. It is worth remembering that a chromatographic process is comprised by three steps - 1) inject the solution into the column, 2) collect the products that were separated according to their selectivities and 3) clean the column with adsorbate-free solution in order to remove all the adsorbate remaining in the adsorbent.

However, making a column with enough adsorbent to process 10L of solution in a row is not practical, since the most expensive part of a chromatographic process is the actual adsorbent. The natural workaround is to divide the mother liquor in smaller volumes that our column can process. Considering this information, the logical conclusion is that, since the adsorbent is the most expensive material, we make a column as small as possible and divide the main solution in volumes that our column can process. However, this is not as practical as one might expect, since dividing the main solution in very small parts takes a long time and the chromatographic process in itself (feeding the column with that amount of volume, let the volume pass through the column and then clean it) takes its time. In the end, by saving in adsorbent we are spending more in labor time. Considering everything mentioned before, there is a need to find a compromise between the column volume and the total solution volume to be processed.

1.2.4 Analysis Methods

In Section 1.1.2.1 we used Mr. John to analyze the outcome of our adsorption experiment, but it only worked because we were evaluating the salt content of an aqueous solution. This was of course a fictional method, since the human palate does not have such accuracy, but the salt content can actually be determined by conductivity. The salt molecule, NaCl, when dissolved in a liquid, in this case water, separates into Na^+ and Cl^-. These two ions, when separated, move freely inside the solution and can conduct electricity. By passing an electric current through the fluid, we can measure the resistance these ions make to that current, hence determining the concentration of salt in said solution.

In Section 1.2.1 the experiment performed by Tswett could be analyzed because the pigments had different colors. How would we analyze such experiment if the components to be separated were not able to be distinguished by taste nor by color? We use an analytical technique called absorbance. In this technique, light passes through the fluid and hits the molecules of the components at study. As the molecule amount increases, the light has a harder time to pass through the fluid, increasing the light absorbance by the components.

With this technique, having colorless solutions, which are the most common, is irrelevant, and the degree of precision is much bigger.

The conductivity is an absolute type of measure. This means that, no matter how many types of salt you have in solution, the conductivity measure will always be the sum of all of them. However, absorbance does not work the same way. When it comes to light absorption, each type of molecule absorbs in a different way, and so each component has a different absorbance signal. This means that the absorbance of each component is relative to some reference value and in that way we can compare the absorbance of each component in a multicomponent solution.

1.2.5 Types of Chromatography

Until now we talked about chromatography in general, as a process that manipulates traveling speeds in order to separate component mixtures. However, those speed differences can be achieved in several ways, as explained in Section 1.1.1, and the type phenomenon used will highly influence how separation is achieved.

Before we dive in the particular types, we need to understand something about how molecules are formed. A molecule is composed by a combination of atoms, which are connected to each other by several types of bonds. There is a significant number of different types of bonds, but they all share one phenomenon: electron transfer. It is this transfer that will determine which type of chromatography we are dealing with.

1.2.5.1 Reversed-Phase Chromatography

This type is mostly used with molecules in which some atoms are bonded through covalent bonds. A covalent bond is when two atoms simultaneously share one, two or three electrons. Each atom has a specific number of electrons ready to interact with other atoms, and when they do interact, the remaining atoms form what we call an electronic cloud. It is easy to deduce that, in a molecule with several atoms, all the electrons that are not part of the bonds form different clouds in different zones of the molecule and some clouds are bigger than others (the cloud depends on the number of electrons that are part of it). The higher the number of electrons in a cloud, the more likely it is for that zone of the molecule to interact with another molecule. This is what we call a pole. If a molecule has several clouds and are of different electronic densities, then we say a molecule is polar. If the clouds of a molecule have all the same electronic density, then the molecule is non-polar.

Figure 1.7 really helps understanding this concept. The atoms that influence the electronic cloud are the ones that are in the extremities of the molecule. If the atoms in the extremities are quite different from each other in terms of atomic number, the bigger the disparity between the number of electrons, hence a bigger electronic cloud. On the left we have polar molecules, and as you can see the atoms in the extremities are different

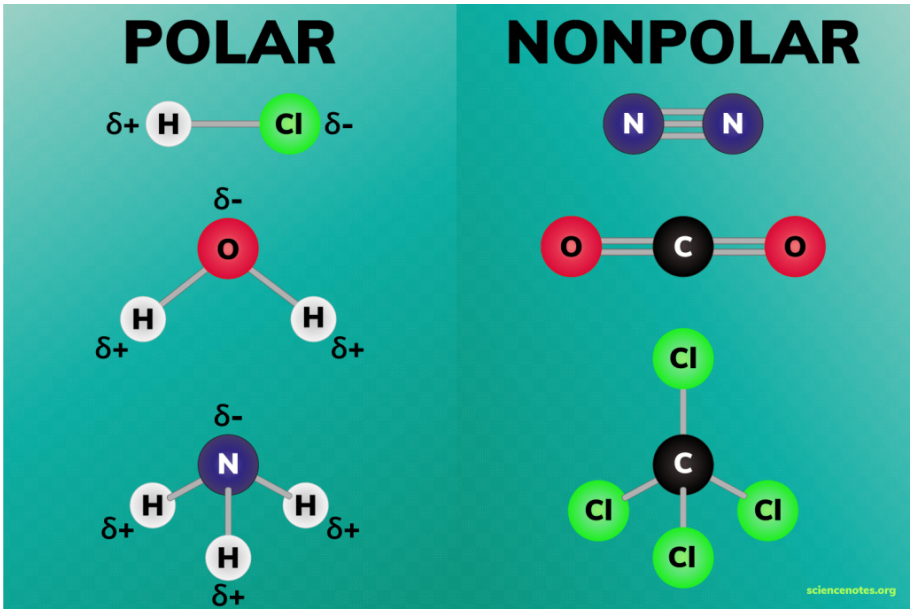

Figure 1.7: Schematics of the electron cloud of polar and non-polar molecules. Adapted from [20].

from each other (i.e. hydrogen has less electrons than chloride, hence bigger cloud). The same happens with the water molecule and the NH_3 molecule.

Oppositely, when the atoms in the extremities are the same, it means there is no electronic disparity between them, hence no electronic cloud. The three examples shown on the left of the picture represent such molecules.

An interesting thing about polarity is that like attracts like. Non-polar molecules have more affinity to other non-polar molecules and the same with polar ones. In reversed-phase chromatography, the adsorbent is non-polar, meaning that it will have more affinity for non-polar components in solution. The higher the non-polarity of said components, the bigger the delay caused by the adsorbent to the adsorbate [21, 22].

Another aspect of this separation technique is the solvent choice. The solvent must be polar, so it does not adsorb as well, as explained in Section 1.2.1. Generally, it is common practice to use aqueous solvents or organic solvents that are polar, such as alcohols.

1.2.5.2 Hydrophobic Interaction Chromatography

This technique uses the same principles as in reversed-phase, with the major difference being the mobile phase [22–24]. Reversed-phase uses a mobile phase that is polar, and most of the times that mobile phase is comprised of alcohols such as ethanol, methanol, isopropanol, etc. These compounds are quite harsh in some components, specially biological components such as proteins, peptides, antibodies, etc. In order for those components to keep their integrity and not lose their biological properties, you can add a small amount of salt to the mobile phase to increase the ionic strength, in order for it to be milder for biological species. You can also add a specific tampon solution in order

to manipulate the pH of the mobile phase. In this way, hydrophobic interaction is a better choice when you are working in the pharmaceutical industry, for example, while reversed-phase is a better choice for chemical processes.

1.2.5.3 Normal-Phase Chromatography

Normal-phase chromatography also uses polarity as the main separation principle, but is the total opposite of both techniques presented before [25]. With this chromatography type, the adsorbent used is of polar nature, so the components that are mixed in solution have to be polar for them to be adsorbed. Regarding the mobile phase, since it cannot be adsorbed it has to be highly non-polar.

Actually this was one of the first chromatographic methods to be invented. For example, the experiment that Tswett made was performed in normal-phase, since calcium carbonate is a polar compound, the pigments are also polar and the mobile phase used to dissolve the pigments was a non-polar organic solvent.

1.2.5.4 Ion Exchange Chromatography

This technique is based on the ionic bonds some molecules can make [22, 25]. An ionic bond is formed when one of the atoms misses an electron and the other one has one to spare. Separated they are unstable, but when they interact they become very stable, since the share of that single electron makes the molecule electronically neutral. This is a strong bond, much stronger than a covalent bond. In order to adsorb such molecules, the adsorbent must also be electrically charged, either positive or negatively (the opposite charge of the molecules to be adsorbed).

One very important property for ion exchange is the pH. The pH is a way of measuring the concentration of H^+ and OH^- ions in a solution. The pH goes from 0 to 14, with 0 being the most acidic (higher concentration of H^+) and 14 the most basic (higher concentration of OH^-).

The mobile phase needs to be well chosen. Since the components to be separated are electrically charged, the mobile phase needs to be prepared for such. The best way is to use an aqueous solvent with diluted salt and a weak acid/base to equilibrate the solution pH. This equilibration can be tricky, because the solution pH must be chosen depending on the components that we are dissolving. For instance, biological components are significantly complex, and can transform themselves into anions or cations depending on the pH of the solution. The pH value that makes such a molecule neutral is called pI (isoelectric point). A pH smaller than the pI leads to a positively charged molecule and vice-versa. Now that the pI is known, we can manipulate the charge of the molecule for it to adapt to our stationary phase, whether it is negative or positively charged.

Another thing to bear in mind regarding ion exchange is the elution step. Comparing to reversed-phase, the only thing needed for elution is to wash the column with component-free solvent, but in ion exchange that is not sufficient, since the ionic bonds are too strong.

Instead, a different mobile phase has to be prepared, usually with a high concentration of salt. A salt is very good for this because is a molecule that, when dissolved in water, decomposes in an anion and a cation, and with enough concentration it basically replaces the component that is tied to the surface of the adsorbent, leaving it free in solution. After that just wash the column again with the main mobile phase to remove the salt that is in the adsorbent and you are ready for another experiment.

1.2.5.5 Affinity Chromatography

Chromatography by affinity manipulates one of the most specific types of adsorption. It takes advantage of the fact that, in nature, there are some molecules that, almost exclusively, have a big tendency to bind to molecules of a very specific structure, which we call the ligand. This is extremely beneficial if we are talking about solutions that have dozens of components and only one of them is of importance. With this chromatographic strategy we can bind such a molecule to the adsorbent surface, knowing that only the molecule of interest inside the solution will bind to our stationary phase [25, 26].

These bonds are usually weak, such as electrostatic or Van der Waals interactions. For elution, it is common to use a suitable mobile phase that changes the pH of the column, promoting the desorption [25, 26].

1.2.5.6 Size Exclusion Chromatography

Size exclusion chromatography is very special. It is the only type that does not use the adsorption phenomenon as the separation method but the actual size of the molecules. As explained in Section 1.2.1, the beads used as adsorbent are porous to increase the surface area for adsorption. In adsorption processes, it is ideal for the particle pore to always have the same size. However, it is impossible to build such an ideal world, as explained in Section 1.2.1.2. In size exclusion it is the complete opposite. The stationary phase is created to be as diverse as possible in terms of pore size. Of course this diversity has to be controlled, otherwise we would never achieve the results we are aiming at. The pore size distribution has to be manipulated according to the molecule size of the components to be tested [22, 25].

As an example, let us consider a mixture with three components, A, B and C. The size of the molecules are in the order $A < B < C$. We can see that the pore size distribution will have two major thresholds: 1) both A and B can enter the pores but C cannot and 2) only A can enter the pores. In the first case, if all pores had that size, both A and B would travel at the same speed through the column and they would travel much slower than C, purifying C. In the second case, if all pores had that size, B and C would travel at the same speed through the column and A would travel much slower than both, purifying A. However, if both thresholds can be combined in the same stationary phase, all three components would travel at different speeds, which allows us to purify all three in the same run.

Regarding specificity, this type has the lowest. There are molecules which size is very similar and create a stationary phase that emphasizes such small thresholds is extremely difficult. This is only used when the molecules are very different in size, which makes the stationary phase easier to be developed. However, since only the pore size matters, the material used to create the stationary phase is not that relevant, so the costs in that department are quite reduced. Furthermore, there are no interactions between the stationary phase and the components, so any mobile phase can be used, as long as the components and the solid are not negatively affected by it.

1.2.6 Concluding Remarks

In this entire section we learned about how we can separate and purify components using chromatography. We discussed how the actual separation process occurs. We also talked about the different types of chromatographic processes, how to determine the type of chromatography that we should use in our process by analyzing the components in question and the mobile phase and how to choose the most adequate stationary phase.

After this introduction to the theory of chromatography, we want to understand how these principles can be applied in the real world, more specifically in the pharmaceutical industry, which will be covered in the next section.

1.3 Chromatography in the Pharmaceutical Industry

Producing a biological molecule and purifying it can be divided in two processes: the upstream (USP) and downstream (DSP) processes. Each are comprised by several unit operations, but as the name of this section indicates, we are going to focus on the chromatography role in this process, more specifically in the DSP.

1.3.1 What is a Biopharmaceutical?

A biopharmaceutical is a component that is extracted from living sources, such as microorganisms, humans, animals or plants [27]. Such products can be proteins (including antibodies), nucleic acids (DNA/RNA), stem cells, tissue, among others. Such components are still widely used in vaccines, gene and cell therapies, production of monoclonal antibodies, etc.

In the early days of biopharmaceutics only natural components were possible to be obtained. Now, with the high demand of biomolecules and the advance of technology, biopharmaceuticals are now synthetically made, such as virus-like particles, viral vectors and recombinant DNA [28]. Either way, after production they all need to be purified, which will be explored in the next section.

1.3.2 The Production of a Biopharmaceutical

Molecules used for vaccines and gene therapy have to be introduced inside the human body, so the regulatory agencies (FDA, EMA, etc.) are very strict when it comes to the levels of purity of such components [29]. Regardless, biomolecules are still the best way to prevent and cure several diseases, such as smallpox, rabies, pertussis, hepatitis A and B, etc. [28].

Some of them are naturally produced and only need to be purified, but the production of a lot of them must be engineered, which is the case with recombinant proteins, for instance. To obtain such molecule we have to "manipulate" in order for it to excrete what we want instead of what it usually excretes.

It all starts inside a cell. Whether we want to produce an aminoacid, a protein or an antibody, we need to manipulate cells in order for them to produce what we desire [29]. This step is part of the USP, which originates not only our POI but many other byproducts that have to be removed from the final product. This is usually performed using a bioreactor filled with cell medium in order to give cells the optimal conditions to grow, producing the compound we want. However, cells do not produce only the POI. They excrete a lot of different other components such as nucleic acids, enzymes, sugars, vitamins, etc. On top of that, the final supernatant has all of these components plus whole and dead cells too. You can see that the final cell culture supernatant is a very complex mixture and we only want one product among all of these, that needs to be purified. This is why it is estimated that the purification step takes around 60 to 80% of the total production cost [29].

After the production, the DSP starts, and usually is comprised of several operation units, such as [28–30]:

- **Clarification:** This step aims to remove whole cells from the supernatant, since the size difference between whole cells and the remaining components is very big. This is usually accomplished by filtration or centrifugation, for example.

- **Initial Purification:** After removing the whole cells, this step can remove several other components, such as enzymes and other contaminants, hence reducing the feedstock volume. Chromatography is usually used to perform this step.

- **Intermediate Processing:** Very similar with the previous step, it aims to remove more contaminant, such as viruses, DNA and RNA, endotoxins and proteins. Chromatography is mainly used in this step, but now with a higher resolution than the previous step.

- **Polishing:** Last step of the DSP. In the end of this step our POI is highly pure, with a very high recovery as well. High resolution chromatography is the preferred method for this step.

This work is focused on the polishing step, mostly using ion-exchange and/or affinity chromatography, explained in Section 1.2.5, so we will focus on those operation units.

1.3.3 Chromatography in the Polishing Phase

Biopharmaceuticals are high valued products, so its recovery must be as high as possible in order to not waste it. Also, since they are to be used in therapeutics and vaccines, their purity must be as high as possible too.

The polishing phase still has several components, but their selectivities are all close to the POI selectivity, that is why they were not separated in the previous steps. To complete this task we need very specific chromatographic media, which can be a resin, a membrane or a monolith and that is actually able to separate our POI from all other contaminants with high degrees of purity. These types of high resolution media are certainly capable of that, but at the cost of recovery. It is known that, in batch chromatography, recovery and purity are inversely proportional. There is a way to circumvent that, which will be talked about in the next section.

1.4 Continuous Chromatography: Is It Possible?

In Section 1.2.3 we said that chromatography is performed in batch mode. Even for mixtures with more than two components, it is quite easy to obtain high purity values for the target component. However, if the retention times of both the product of interest (POI) and the impurities are similar, to obtain high purity you would have to discard a lot of the POI that is going out along with the impurities. Such as purity, recovery is a percentage and compares the amount of POI recovered in the end of the chromatographic process and the amount of POI fed to the system. It becomes clear that, in batch chromatography, the purity is inversely proportional to the recovery: the higher the purity requirements, the more amount of POI has to be discarded, hence lower recovery.

Figure 1.8 shows a usual chromatogram of a binary mixture. We can see that both components overlap in a certain time interval, in which both are impure. For now let us assume that P1 is the POI and P2 is just an impurity we want to remove from solution. In order to collect highly pure P1, we can choose a lower t_1 value. However, by doing so, we are increasing the amount of P1 in the P1+P2 fraction, which means we are throwing away a bigger amount of P1, reducing recovery. Inversely, by choosing a bigger t_1 value, we are significantly increasing recovery, but at the same time we are collecting P1 together with P2, reducing P1 purity. The same train of thought can be used if P2 is the POI and P1 the waste. And what if both components were of high value? Now we have two time points we can manipulate, but in the end the outcome is the same: by increasing the purity of any of the components we are decreasing their recovery and vice-versa. This inverse proportionality comes from the fact that, in batch mode, the only variable we can effectively manipulate is the interval between t_1 and t_2.

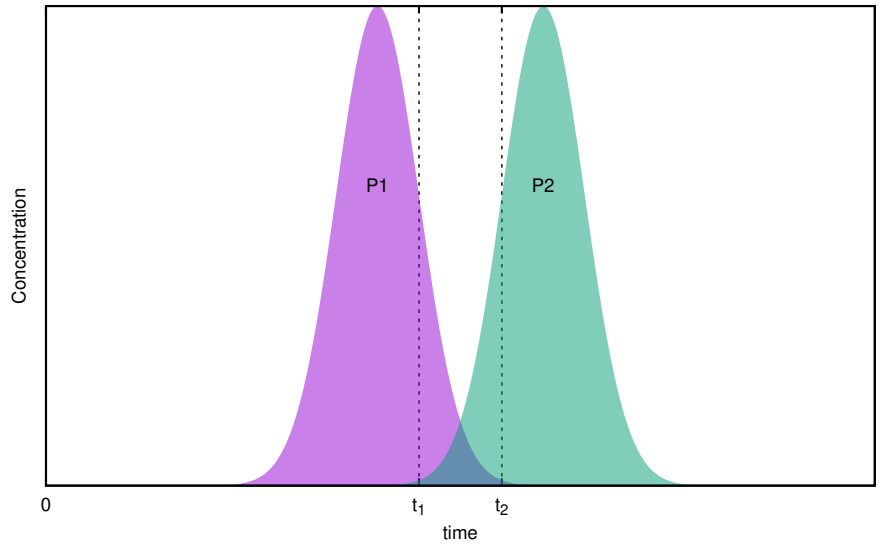

Figure 1.8: Chromatogram of the elution of a binary mixture. P1 and P2 represent the two products inside the mixture.

What we can do in order to maintain the same levels of purity and increase recovery is to collect the P1+P2 fraction and reinject them in the column after the first run is finished. We can actually reinject said fraction together with more fresh feed in order to increase the process productivity. This process significantly increases the recovery because, instead of discarding the impure fraction, we are collecting and reprocessing it. However, we are using only one column, so the re-chromatography step is time consuming because it is not automated, it has to be manually reinjected inside the column.

With all of this in mind, various concepts were formulated in order to tackle this problem, but all of them share one common thing: they are all continuous.

1.4.1 True Moving Bed: the First Concept for Continuous Chromatography

The True Moving Bed (TMB) chromatography is an entirely theoretical concept and it explains how, by making the stationary phase move in the opposite direction of the fluid, one can make the continuous separation of a binary mixture. When the solid and liquid phases move in a counter-current fashion, the driving force is maximized, promoting mass transfer and consequently the adsorption phenomenon. The "magic" of this concept is the speed of both the liquid and the solid. Since chromatography is all about speed, if we fix the speed of one of the phases and adjust the speed of the other, we can make one component travel with the solid and another component travel with the liquid.

Figure 1.9 shows a simple diagram of how the TMB process works. After choosing the appropriate column size for our system, as explained in previous sections, we can divide it in four zones, as shown in the diagram. The fresh solvent flows from left to right and the solid travels from right to left. Assuming that we determined the right flowrates for this process, the most retained component, which we call A, will travel with the solid and

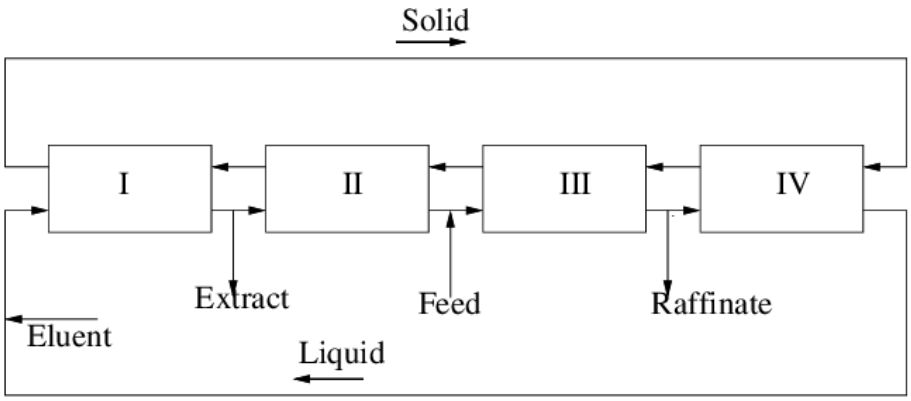

Figure 1.9: Schematics of the True Moving Bed Process. Adapted from [31].

the least retained component, which we call B, will travel with the liquid, simply due to their affinities. This is very advantageous, since with this method you can continuously inject solution into the system without stopping to clean. But why four zones? Because this system has four different instances: 1) mixture of A with solvent, called the extract, 2) mixture of B with solvent, called the raffinate, 3) pure eluent and 4) solid completely cleaned. As we can see in the figure, feed is continuously injected in the middle of the column, which is between Zones II and III, component A travels with the solid and is collected in the end of Zone I as the extract and component B travels with the liquid and is collected in the end of Zone III as the raffinate.

On paper this technique seems very efficient, since it reduces the experimental labor, reduces solvent consumption and can actually achieve the same purity values achieved in batch processes, with the major advantage of a massive increase in yield [32]. However, in practical terms, this process is not viable at all. To understand the reason, we have to discuss how it could be built. The simplest way would be to have a single circular tube with four ports (feed, raffinate, extract and solvent). We would also need two liquid pumps (feed and solvent) and one solid pump (resin recirculation). Right from the start there is a difficulty, which is creating the outlets for the extract and raffinate, because the equipment to accurately divide one liquid stream into two is very expensive. Furthermore, the adsorbent is made of very small particles, smaller than grains of sand, which corrodes the tubing and other equipment that comes in contact with, no matter the material used to make it [33].

Regardless, the TMB concept gives too many advantages to be put aside simply by practical reasons. There has to be a way to use the advantages of the TMB while avoiding its impracticality. With this mindset the SMB was created.

1.4.2 Simulated Moving Bed: born from the TMB infeasibility

The Simulated Moving Bed (SMB) concept was created to overcome the impracticalities of the TMB process. The major issue with the TMB is the mobility of the adsorbent, which causes corrosion in the tubing and inefficiencies when it comes to solid homogeneity

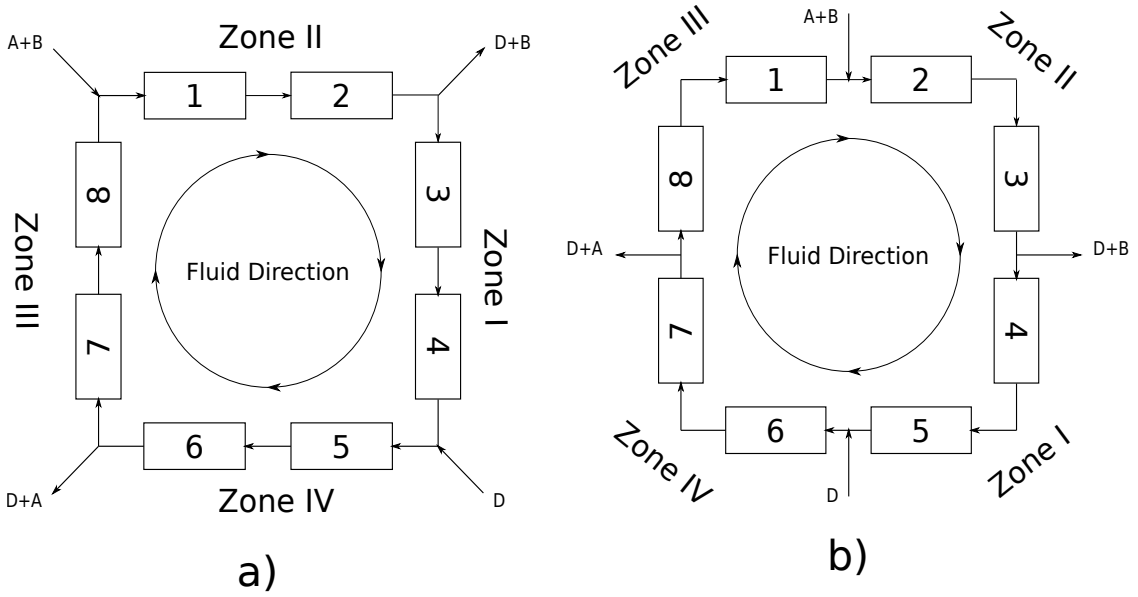

Figure 1.10: Schematics of the Simulated Moving Bed process. a) represents the SMB system at $t = 0$ and b) at $t = \tau$. τ is the switching interval.

inside the column [33, 34]. To solve this problem, an idea arose: instead of making the solid move, why not making the entry and/or exit points of all streams move? The idea was first implemented by Broughton et al in 1961, which was called the Sorbex process [35]. Another major use of the SMB in the petrochemical industry was the Parex process [36–38], to purify p-xylene. It had 24 columns divided by four zones, which is a massive investment.

The implementation is actually quite simple. Using the TMB as example, we can use four columns packed with adsorbent representing each zone. Now, in order to emulate the counter-current flow of the solid, we move the entry and exit points of the system, called ports, at specific time intervals. This switch must match the flow direction of the liquid. In this way we can have the benefits of the solid flowing in counter-current without the downsides of the TMB practical implementation.

Let us use Figure 1.10 to further explain the process in more detail. We can establish a relationship between subfigure a) and Figure 1.9, with the difference that the former has two columns per zone, instead of just one. After a certain amount of time τ, which we call the switching interval, all ports are moved one column in the direction of the fluid, as shown in subfigure b). It is worth emphasizing that the zones are entirely dependent on the port positions, so if the ports switch by one column, then the zones also shift positions by one column. For instance, in a) Zone II is comprised by columns 1 and 2 and in b) Zone II is now columns 2 and 3. Regarding the switching interval, this is totally dependent on the column size, meaning the smaller the column, the smaller the switching interval.

Now we have to talk about how well the SMB emulates the TMB. In the TMB process the solid is continuously flowing, in the SMB that is not true. As explained before, the port switching is done after the switching interval is reached, meaning that the port switching

is a discrete process, not continuous. In order for the SMB to perfectly emulate the TMB the port switching had to be continuously moved. Such a concept is infeasible in reality. The best way to do it is to reduce the column size and increase the number of columns, so the total adsorbent amount is constant. However, using a very high number of columns is also impractical, because if you increase the number of columns, you increase the number of port switches and consequently the amount of equipment needed to establish an SMB system. In the end, it is practically impossible to fully recreate the TMB concept by means of port switching, so a compromise has to be reached regarding the number of columns used.

The SMB applications mentioned, although highly efficient, were very complex, had huge investment costs as well as high maintenance costs, not to mention the time spent just training workers to use such system. Nevertheless, this was easily paid off by the increase in recovery and productivity and the significant decrease in solvent consumption.

The processes mentioned above were comprised of a significant amount of columns, which is the major cost when it comes to chromatographic processes. In these industries, the investment could be easily made due to the fact that they knew it would pay off in the long run, but the thought of reducing the number of columns is always a recurring one.

After the first process, SMB units with 24 columns started to get obsolete. Now companies already sell systems with 16, 12 and even 8 columns [39]. There also works with 4 columns [40]. Currently, 2 column SMB units are very popular [41].

However, the SMB process is not suited for all types of chromatographic separations. The processes coming up next fill the gaps that the SMB leaves regarding other types of separations.

1.4.3 6-column Multi-column Counter-current Solvent Gradient Purification: for mixtures with more than two components

As explained before, the SMB separates the feed mixture into two fractions. In a binary mixture, we can achieve high degrees of purity and recovery for both components, but what if we have a mixture with three? We can obtain high purity and recovery for one component, but the others have to be collected all together in the same fraction. Of course we could implement another SMB train to purify the remaining two. However, this strategy would double the amount of columns needed, and column packing is, usually, the most expensive material in a chromatographic process.

Figure 1.11 shows a diagram of a normal ternary mixture. In this case, from 0 to t_1 we can collect W with high purity, between t_1 to t_2 the P is mixed with W, between t_2 and t_3 we can collect P with high purity, between t_3 and t_4 P is mixed with S and starting from t_4 we can collect S with high purity.

With this in mind we can conclude that increasing t_3 and decreasing t_4 we can increase the purity of P, since we are reducing the time window in which we collect it. However, by doing so, we are increasing the P+W and P+S fractions, reducing the recovery. But

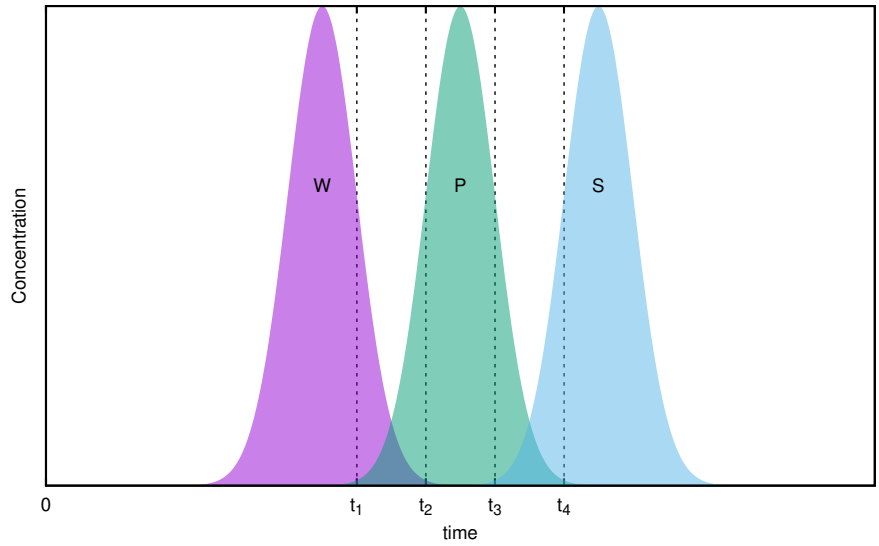

Figure 1.11: Chromatogram of the elution of a ternary mixture. W, P and S symbolize weak impurities, POI and strong impurities, respectively.

what if we decrease t_3 and increase t_4? In that way we are increasing our recovery, while decreasing the purity.

We explained in Section 1.3 that chromatography plays a major role as the last technique used in the process train of creation of a biopharmaceutical. The result of everything before this step is never a binary or a ternary mixture, is always a dozen of components, and the product of interest almost always elutes between all the impurities, making it even harder to purify. However, the POI is almost never the first or the last component to elute, meaning that we can treat this mixture as a "pseudo-ternary" mixture. We call it pseudo-ternary because it has more than three components, but in the separation process it can be treated as if only three components exist: all the components that elute before the POI are treated as a single product W, the POI is treated as another component and all the components that elute after the POI are treated as a single product S. This separation technique is called central-cut separation.

The 6-column MCSGP was first introduced in 2007 by a research group led by Massimo Morbidelli [42]. The first assumption was that all three components of the mixture were of high value and needed to be collected with high purity and recovery, so they developed a system with six columns and two processing trains that run in parallel: three of the columns are interconnected, called the interconnected lane, and the other three are in batch model, called the batch lane.

Figure 1.12 shows the concept in more detail. During the first switching interval, columns 1, 3 and 5 are separating highly pure S, P and W, respectively. In parallel, columns 2, 4 and 6 are being eluted with solvent to remove impurities that adsorb even weaker than W. In the second switching interval, columns 2, 4 and 6 are collecting S, P and W, respectively, while columns 3, 5 and 1 are being eluted in an interconnected fashion.

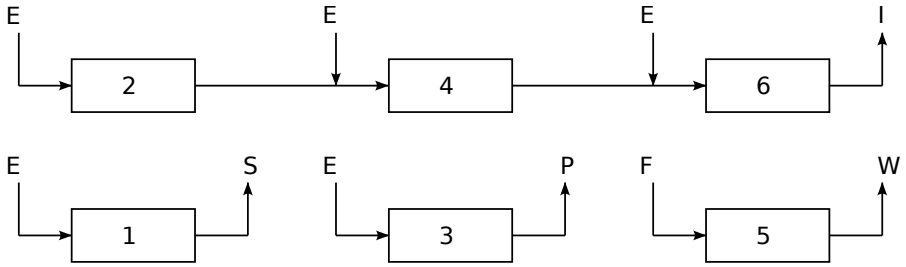

Figure 1.12: Diagram of the 6-column MCSGP. The letters E, I, S, P and W represent the solvent, waste inerts, strong impurities, POI and weak impurities, respectively.

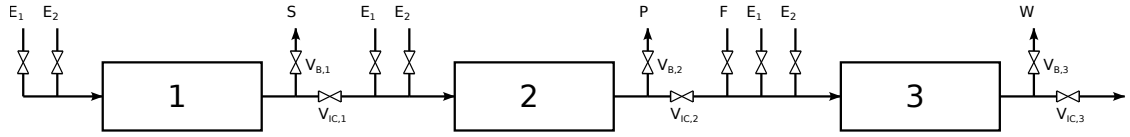

Figure 1.13: Diagram of the 3-column MCSGP. V_B are the ON/OFF valves for the batch lane and V_{IC} the valves for the interconnected lane, one per each column. E_1 and E_2 are solvent streams. S, P and W are the strong impurities, POI and weak impurities, respectively.

The switching interval can be chosen by analyzing the batch chromatogram of the mixture to be separated, like Figure 1.11. With this strategy we can achieve very high purities and recoveries for all three components.

1.4.4 3-column MCSGP

As explained in Section 1.4.3, the first MCSGP was built with six columns separated by two lanes, the interconnected and the batch lanes. However, these lanes are run in parallel and are totally independent of each other, then why not reducing the system from six to three columns? The research group from Massimo Morbidelli thought the same, put that idea into practice [43] and designed the 3-column MCSGP.

As we can see in Figure 1.13, there are two ways we can operate this train: 1) all three V_B valves are opened and V_{IC} valves are closed, activating the batch lane and 2) all three V_B valves are closed and V_{IC} valves are opened, activating the interconnected lane. In this process, another solvent stream was added to add more flexibility in the process. In the six column version, only one switching interval exists, since both lanes are run in parallel and all columns must switch at the same time. However, in the three column version, there is no need for the switching interval to be the same for both lanes. Since they are run sequentially, each lane can have its own switching interval. Mathematically speaking,

$$\tau = \tau_B + \tau_{IC} \tag{1.1}$$

With τ_B and τ_{IC} being the switching intervals of the batch and interconnected lanes, respectively, while τ is the total switching interval. Only after τ do the columns switch positions.

By comparing the 6-column with the 3-column, at first sight it seems that $\tau_{6col} < \tau_{3col}$ always holds up, but actually it is not entirely true. Since τ_{3col} can be split into two smaller intervals, those intervals can be fully optimized by using different flowrates for each lane, being able to reduce those time intervals. In this way, hypothetically, we can have $\tau_{6col} \geq \tau_{3col}$.

1.4.5 2-column MCSGP

We already know that the biggest cost of any chromatographic process is the chromatographic media, that is why the concern of decreasing the number of columns is more and more imminent in the industry.

By studying the two previous versions of the MCSGP, the biggest strength of this process is that it recycles the impure fractions and purifies it later in the cycle to massively increase yield. However, if it is true that the major advantage is the recycling of the impure fractions, it is also true that the column outlets exist in only two states: 1) W, P or S exit with high purity and 2) P is mixed with W or S. Either way, P is either impure or pure. That being said, there is a good chance that a system with only two columns would do the trick as good as their predecessors. Again, Morbidelli and his peers had the same thought and the 2-column MCSGP was born [44].

In Figure 1.14 is represented the flowsheet of the real system. This system is also governed by a switching interval, the time when both columns switch positions. In this system there are four steps during a switching interval. Step τ_1 is an interconnected step, where we are directing the impure fraction W+P from column1 to column 2. Step τ_2 is a batch step, we inject fresh feed in column 2 while collecting pure P in column 1. Step τ_3 is an interconnected step where we are recycling the P+S fraction exiting column 1 into column 2. Finally, step τ_4 is a batch step, where we collect pure W from column 2 while collecting pure S from column 1. This process still maintains the same level of flexibility and performance of the six and three column MCSGP.

1.4.6 Periodic Counter-current Chromatography

The upstream process of a biopharmaceutical produces the POI together with several other components, all of them byproducts of cell culture. In a specific case, such as the separation of Immunoglobulin G (IgG) from a cell culture supernatant, the most efficient way for this example is to use protein A as the chromatography media. It was explained in Section 1.3 all the pros and cons of using protein A in such a process, nevertheless is still widely used in the biopharmaceutical industry, since there are still no other alternatives available that are as good.

For this type of separation, the high protein A selectivity makes the IgG and nothing else to bind to it, leaving all the other impurities to flow through. However, when the IgG starts leaving the column, the injection has to stop and the column elution starts. This

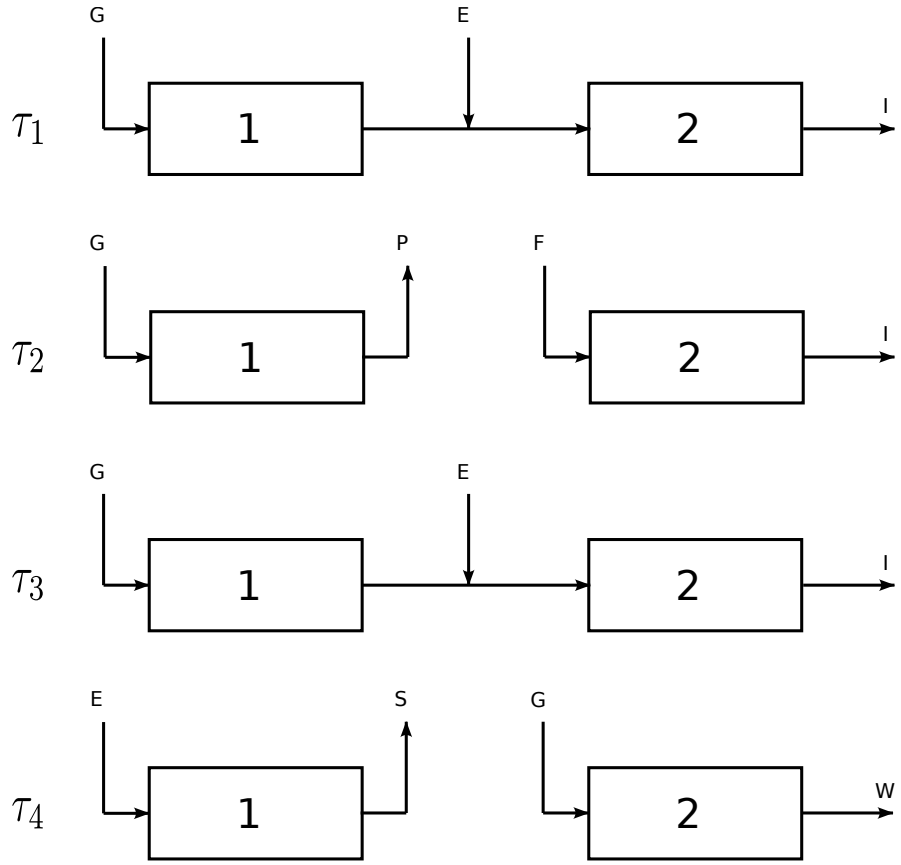

Figure 1.14: Design of the 2-column MCSGP. G, S and F are the gradient, solvent and feed streams while W, S, P and I are weak impurities, strong impurities POI and very weak impurities, respectively.

process achieves high purity and recovery, but a very poor resin utilization, since the IgG breaks through way before the entire column is saturated.

To overcome this the Periodic Counter-current Chromatography (PCC) was invented [45]. One entire cycle is comprised of three switching intervals (the same as the number of columns) and each is subdivided into two steps. Those two steps are presented in Figure 1.15. During step τ_1 we collect POI from column 3 while redirecting unbound impurities to waste. Column 2 stays idle because it is not needed in that step. During step τ_2, when the product breaks through, the outlet of column 1 is redirected to column 2, making it possible to inject feed in column 1 until full saturation occurs. In the meantime, column 3 goes through the process of regeneration, making it ready for another feed loading. After those two steps, one switching interval is finished and all columns switch positions.

1.4.7 CaptureSMB

Most of the continuous chromatographic processes that exist are valid for all types of chromatography types (reversed-phase, affinity, ion-exchange, etc.), but the CaptureSMB (CSMB) was designed to be used with affinity chromatography. In 2015, the first time this

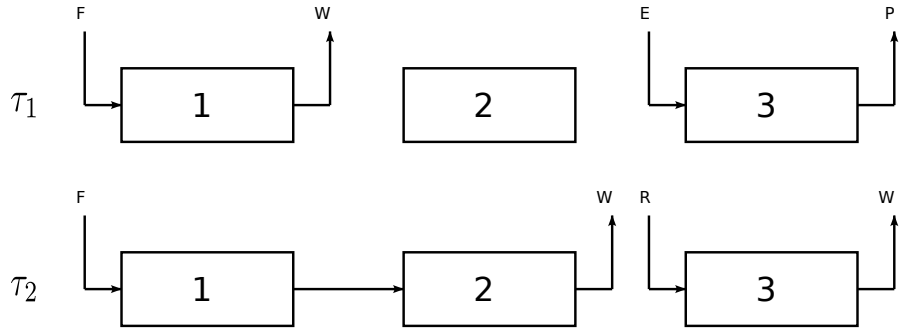

Figure 1.15: Design of the 3-column PCC. F, W, E, P and R represent the feed, waste, eluent, POI and regeneration streams, respectively.

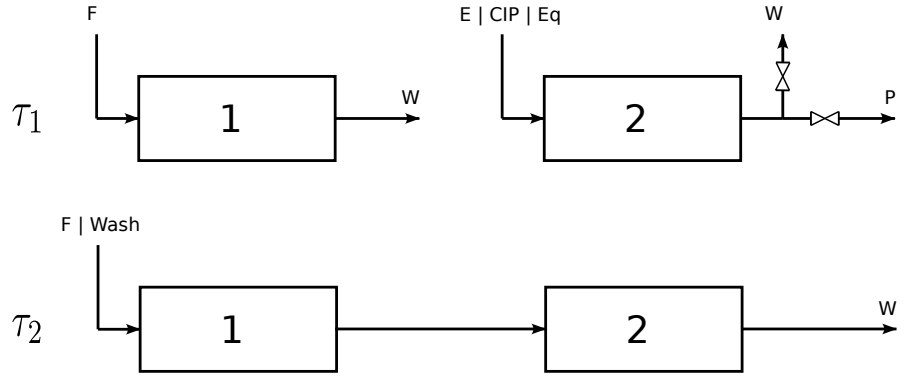

Figure 1.16: Design of the CaptureSMB process. F is for feed stream, E for elution, CIP for cleaning in place, Eq for equilibration, Wash for washing step, W for waste and P for product.

system was published [46], the authors used specifically protein A affinity chromatography as a case study to develop it.

Protein A, as explained in Section 1.3, is extremely efficient in separating mAbs and similar types of molecules from cell culture supernatants, due to the fact that its selectivity towards such components is much higher than the rest of the components inside the supernatant. However, even though the purity and recovery is high, using one column in batch mode makes it impossible for the adsorbent to be fully saturated with our POI, since we start losing it before that point.

Figure 1.16 explains how this problem is circumvented. This system has two switching intervals, again corresponding to the number of columns in the process, and each interval is comprised of only two steps. Step τ_1 is a batch step, where feed is injected in column 1 while column 2 is being eluted, washed and cleaned in place (CIP). It is worth noting that the flowrate of the feed stream should be lower than the flowrate used for elution and wash, to prevent any product loss. During τ_1, column 2 is subdivided into two steps, one for collecting waste and other for collecting product. Step τ_2 is an interconnected step, in which we are recycling the adsorbate to increase our recovery and maximize the capacity of column 1.

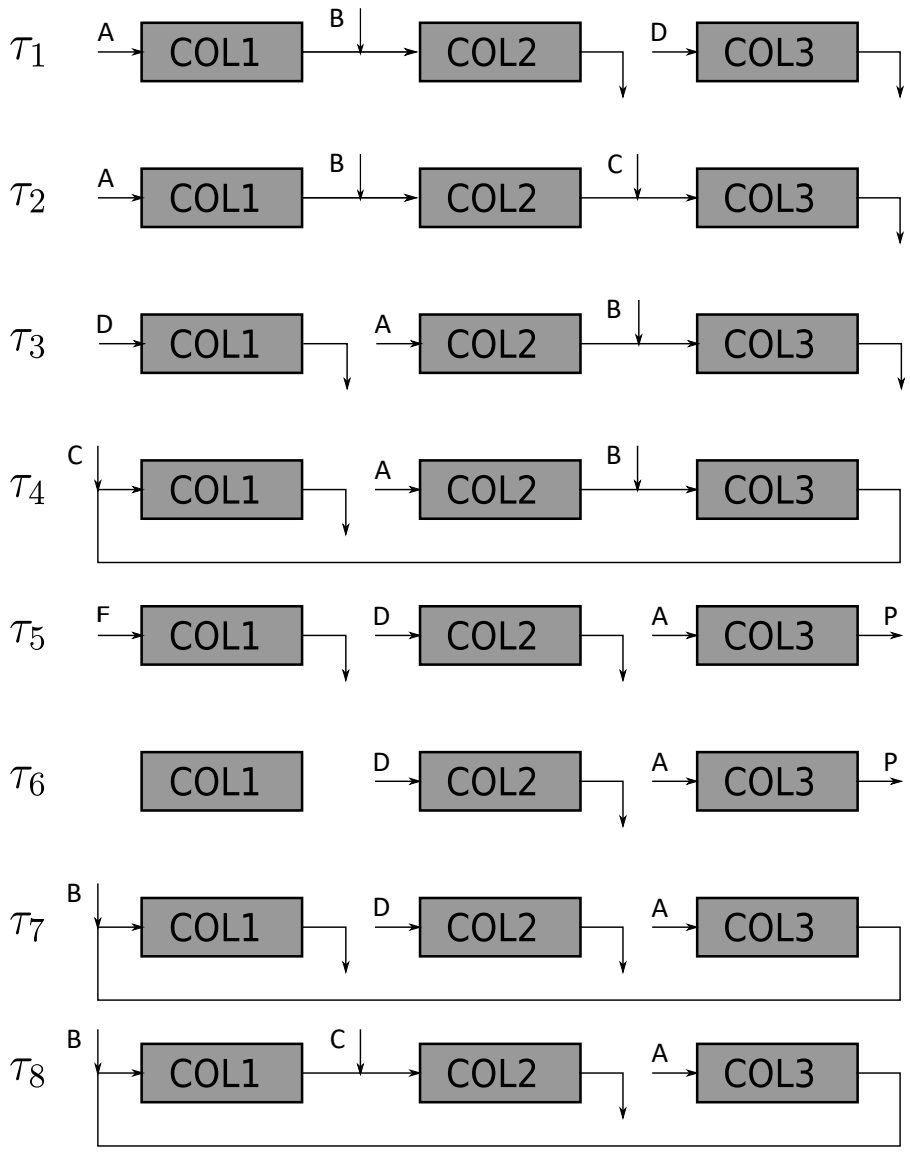

Figure 1.17: Design of the GSSR process. A is the gradient stream, B, C and D are different solvent streams, F is the feed stream and P is the product stream.

1.4.8 Gradient Steady State Recycle: not only for purification

The Gradient Steady State Recycle (GSSR) process is a peculiar one. This system, developed by NOVASep in 2010 [47], had major influences from the 3-column MCSGP. It is also a three column process, but with a small but significant difference. In the MCSGP processes, when a switching interval ends, all inlets and outlets move forward one column. However, in the GSSR, only the solvent streams move after a switching interval, the feed and product streams are always fixed in columns 1 and 3, respectively.

Figure 1.17 shows us exactly that. The process has four major process intervals, τ_I, τ_D, τ_F and τ_P, which are the switching interval, desorption, feed and production times. The three switching intervals are as follows, 1) comprised by τ_1 and τ_2, which are just elution and equilibration, 2) comprised by τ_3 and τ_4, they are the same as the previous but all

solvent streams moved forward one column and 3) comprised by the four remaining steps, τ_5 to τ_8. In this one we introduce the feeding and production step, as well as an extra step (τ_6) to ensure maximum flexibility, in case the product collection might take longer than the feeding step. Nevertheless, we see that all four solvent streams moved forward one column, both inlets and outlets.

Even though we have eight steps per cycle, the time length of all of them can be expressed as a linear combination of the four process times mentioned above as follows,

$$\tau_1 = \tau_D \tag{1.2a}$$
$$\tau_2 = \tau_I - \tau_D \tag{1.2b}$$
$$\tau_3 = \tau_D \tag{1.2c}$$
$$\tau_4 = \tau_I - \tau_D \tag{1.2d}$$
$$\tau_5 = \tau_F \tag{1.2e}$$
$$\tau_6 = \tau_P - \tau_F \tag{1.2f}$$
$$\tau_7 = \tau_D - \tau_P \tag{1.2g}$$
$$\tau_8 = \tau_I - \tau_D \tag{1.2h}$$

This process is also designed for central-cut separation. However, different from the MCSGP, since F and P are always injected and collected from the same columns, this means that column 1 does not need to be the same as columns 2 and 3. This allows for additional flexibility, since we can put a fixed bed reactor[3] instead of a chromatographic column in position 1, for example.

It is still not widely popular but has a lot of potential, it just needs to be fine-tuned in order to be able to compete with the most used processes. This work is motivated by this purpose: how can we make the GSSR a more competitive process for it to be implemented by the industry?

1.4.9 Concluding Remarks

Apart from all the processes stated above, there are many more continuous and semi-continuous processes with more or less columns [48–55]. Recently our research group released two papers regarding semi-continuous chromatography using only one column and a plugflow device for storing the impure fractions [56, 57]. It becomes clear how the industry wants to shift towards automated processes, due to their advantages regarding recovery, resin utilization, product concentration and reduction in solvent consumption. Such processes are also highly automated, meaning we can also reduce man labor and also human errors.

[3] a fixed bed reactor is a column packed with a catalyst that performs a chemical reaction instead of adsorption.

In the current times, with technology evolving in such a rapid manner, it is also very convenient to use them to reduce even more experimental work, hence raw material costs. All the processes mentioned above can be mathematically described, which means that we can perform several experiments in a purely theoretical way, without spending any raw materials. This allows us to improve our processes much faster and with less costs. In recent times this has become an integral part in the Process Engineering field [58–61].

As explained in Section 1.4.8, this work is about a novel approach into simulating and optimizing the GSSR process. It tries to prove that a well made mathematical description of such a process can significantly reduce the amount of experiments required to optimize our chromatographic processes. Furthermore, by optimizing it using several numerical methods, we can obtain the optimal process parameters to minimize raw material consumption, reduce process time and increase productivity. Next chapter will cover such topics.

2

System Design and Mathematical Description

The main purpose of this work was to simulate and optimize the GSSR, but for that we needed an experimental set-up for us to experimentally validate our assumptions and calculations, and so we did. In this chapter we will talk about the design we came up with, its mathematical description and simulation and optimization of said process. For that we used a case-study, which was the one NOVASep published when the GSSR came out.

2.1 Mathematical Background

This section aims at introducing all the mathematical knowledge needed to proper understand this work. We are going to talk about how the components travel through a packed column, considering all the phenomena mentioned in the previous chapter (fluid velocity, adsorption, mass transfer, etc.). The mathematical description of such phenomena is crucial for us to understand what is actually happening inside the column, enabling us to predict, with a good degree of accuracy, the outcome of our experiments. Let us dive into it.

2.1.1 Law of Mass Conservation

The theory of mass conservation is the most important one. Not just for chromatographic processes, but for all chemical processes. It basically follows the principle of Lavoisier, who discovered, in the 18th century, that mass is never destroyed nor created, only transformed. This principle holds true for every type of reaction, either chemical or physical. For example, a very common chemical reaction is the production of ammonia, which needs nitrogen (N_2) and hydrogen (H_2) as the only reagents. If we wanted to write it as a chemical equation, it is as follows:

$$H_2 + N_2 \longrightarrow NH_3 \tag{2.1}$$

However, this is not complete. According to the conservation of mass, the mass (i.e. the number of atoms of each species) of the sum of the reagents must be equal to the mass of the sum of the products. According to this equation, we have two H atoms on the left side but three on the right side. Regarding N, we have two on the left side and one on the right side. Since this is physically impossible, we need to rearrange this equation in order for the mass conservation law to hold. If we write it like this,

$$3\,H_2 + N_2 \longrightarrow 2\,NH_3 \tag{2.2}$$

We are stating that, to create two molecules of ammonia, we need three molecules of hydrogen and one molecule of nitrogen. In this way, the mass conservation law is held (six atoms of hydrogen and two atoms of nitrogen on both sides, respectively). The method of introducing coefficients behind the molecules in order for the atom number to be the same in both sides of the equation is called stoichiometry and the constants are called stoichiometric coefficients.

Counting atoms is the way to go when we are dealing with actual chemical reactions, where the reagents are turned into other molecules. Adsorption is a physical reaction, there is no atom change in the molecules, they are either stuck to the adsorbent or not. In this case, we only need to keep track of the molecules themselves instead of their atoms, which makes it a bit simpler.

Let us assume that we packed our adsorbent SALTY into a column in order to remove salt from an aqueous solution by adsorption. For us to be able to accurately predict the amount of salt that leaves the column (the less amount the better) without experiment it in the laboratory, we need to find a way to know where our salt molecules are. Are they adsorbed in the solid? Are they inside the particles but not adsorbed? Are they outside the particles? Are they still entering the column or already exiting it? We have to keep track on them wherever they might be.

Before we start said quantification, we need to talk about the porosity of the column. As shown in Figure 1.4, the bead is porous to increase surface area, hence promoting adsorption. However, when you pack said adsorbent in a column, the beads themselves have empty spaces among them, which is a porosity of a different magnitude. We have to take them both into account when applying the mass conservation law. Let us call ϵ_b to the porosity outside the particles, ϵ_p to the porosity inside the particles and ϵ_t for the sum of the two, representing the total porosity inside the column. All porosities represent volume fractions, which means their value range is between 0 and 1. Since it represents the volume outside the particles, we can describe it as follows,

$$V_b = \epsilon_b V_c \tag{2.3}$$

Where V_b is the interparticular dead volume and V_c is the total column volume.

The intraparticular dead volume is different, because it is usually given by the manufacturer and depends entirely on the beads themselves. We need to quantify first the volume the solid occupies in the column, and that can be represented by,

$$V_s = (1 - \epsilon_b)V_c \tag{2.4}$$

Where V_s is the volume of packing. This one is intuitive. If ϵ_b is the volume fraction occupied by everything but the beads, then $1 - \epsilon_b$ is the volume fraction the beads occupy. If ϵ_p is the empty volume inside the particles, then,

$$V_p = \epsilon_p V_s = \epsilon_p(1 - \epsilon_b)V_c \tag{2.5}$$

Where V_p is the intraparticular dead volume. Now that we know both dead volume fractions, we can express the total dead volume as the sum of those fractions,

$$\begin{aligned} V_0 &= V_b + V_p \\ \epsilon_t V_c &= \epsilon_b V_c + \epsilon_p(1 - \epsilon_b)V_c \\ \epsilon_t &= \epsilon_b + \epsilon_p(1 - \epsilon_b) \end{aligned} \tag{2.6}$$

Where V_0 is the total porous volume of the column. Now that we understood the concept of porosity, it is time to proceed to our mass balance. Figure 2.1 is here to help us visualize what is going on inside a column. In here we have represented a cross-sectional cut of a column, it is basically a fictitious slice of thickness Δl of a real column. That being said, c_l is the concentration of salt entering our slice, $c_{l+\Delta l}$ is the concentration of salt exiting the slice, A is the cross-sectional area and ϵ_t is the total porosity of the column. Since the amount of salt injected in the column does not disappear, we can quantify the amount that gets retained inside the column by comparing the mass injected and the mass that exits the column. That mass difference represents the mass that remained in the bulk plus the mass that got adsorbed in the solvent.

Now that we have established where the salt can be, we can define a general expression:

$$m_{in} = m_{out} + m_{acc} \tag{2.7}$$

Where m represents the mass of salt and indices in, out and acc simbolize the inlet, outlet and accumulated masses, respectively. Now, as said before, this accumulated mass can be divided in two terms, the mass that stays in the bulk and the mass that is adsorbed in the packing. Hence,

$$m_{acc} = m_{bulk} + m_{ads} \tag{2.8}$$

Combining the last two equations we obtain,

$$m_{in} = m_{out} + m_{bulk} + m_{ads} \tag{2.9}$$

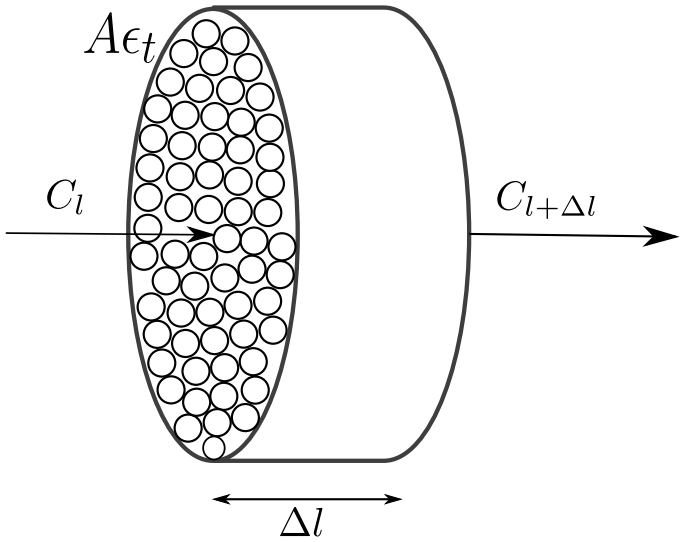

Figure 2.1: Cross-sectional cut of thickness Δl of a packed column of length L_c.

Before we go any further, we have to address the fact that the mass is dependent of the length of the column, Δl, but also dependent of time. As time goes by, as more volume is injected, the accumulated masses, as well as the outlet, change and so we have to take that into account as well.

As we said in Section 1.1, the driving force of adsorption processes is the concentration, so it is useful to transform these masses in concentrations. From now on, volume will be expressed in cm^3, mass in g, time in min, area in cm^2 and length in cm. That being said, let us apply the mass conservation principles to Figure 2.1:

$$m_{in} = V_{inj}\ c_l \tag{2.10a}$$

$$m_{out} = V_{inj}\ c_{l+\Delta l} \tag{2.10b}$$

$$m_{bulk} = A\Delta l\ \epsilon_t\ c_{bulk} \tag{2.10c}$$

$$m_{ads} = A\Delta l\ (1-\epsilon_t) c_{ads} \tag{2.10d}$$

In this example, $A\Delta l\epsilon_t = V_0$ and $A\Delta l\ (1-\epsilon_t) = (1-\epsilon_t)V_c$. In here we take into account the amount of non-porous solid that exists in the column ($(1-\epsilon_t)V_c$). By introducing these expressions into the mass balance,

$$V_{inj}\ c_l = V_{inj}\ c_{l+\Delta l} + A\Delta l\ (\epsilon_t\ c_{bulk} + (1-\epsilon_t) c_{ads}) \tag{2.11}$$

It is also convenient to work with flowrates instead of volumes. Since a flowrate is volume injected per unit of time, in this case cm^3/min, we can include a variable Δt that represents a time interval. We can add it in the above equation in the following way:

$$\begin{aligned} \frac{V_{inj}}{\Delta t}\, c_l &= \frac{V_{inj}}{\Delta t}\, c_{l+\Delta l} + \frac{A\Delta l}{\Delta t}\, (\epsilon_t\, c_{bulk} + (1-\epsilon_t) c_{ads}) \\ Q\, c_l &= Q\, c_{l+\Delta l} + \frac{A\Delta l}{\Delta t}\, (\epsilon_t\, c_{bulk} + (1-\epsilon_t) c_{ads}) \end{aligned} \tag{2.12}$$

Where Q is the volumetric flowrate injected. We have to further define c_{bulk} and c_{ads}. From now on, $c_{ads} = q$, since q is the conventional letter to represent adsorbed mass in a solid.

As previously stated, the accumulated masses are time dependent. In order to describe them, we use the same time interval Δt, and define them as follows,

$$c_{bulk} = c_{t+\Delta t} - c_t \tag{2.13a}$$

$$c_{ads} = q_{t+\Delta t} - q_t \tag{2.13b}$$

By combining them with Equation 2.12 we get:

$$\begin{aligned} Q\, c_l &= Q\, c_{l+\Delta l} + \frac{A\Delta l}{\Delta t}\, (\epsilon_t\, (c_{t+\Delta t} - c_t) + (1-\epsilon_t)(q_{t+\Delta t} - q_t)) \\ Q\, c_l &= Q\, c_{l+\Delta l} + A\Delta l \left(\epsilon_t\, \frac{c_{t+\Delta t} - c_t}{\Delta t} + (1-\epsilon_t) \frac{q_{t+\Delta t} - q_t}{\Delta t}\right) \end{aligned} \tag{2.14}$$

The choice of expanding Δt has a reason. As you may start to understand, we are now able to express the right-hand side of the equation in the form of derivatives. The balance assumes the form,

$$Q\, c_l = Q\, c_{l+\Delta l} + A\Delta l \left(\epsilon_t\, \frac{\partial c}{\partial t} + (1-\epsilon_t) \frac{\partial q}{\partial t}\right) \tag{2.15}$$

We can apply the exact same reasoning for Δl. By rearranging the left-hand side we obtain,

$$\begin{aligned} Q \frac{c_{l+\Delta l} - c_l}{\Delta l} &= -A \left(\epsilon_t\, \frac{\partial c}{\partial t} + (1-\epsilon_t) \frac{\partial q}{\partial t}\right) \\ Q \frac{\partial c}{\partial l} &= -A \left(\epsilon_t\, \frac{\partial c}{\partial t} + (1-\epsilon_t) \frac{\partial q}{\partial t}\right) \end{aligned} \tag{2.16}$$

This type of mathematical model cannot be analytically solved. The only way is to use numerical methods and solve it computationally. In order to facilitate that, instead of solving it from 0 to L_c, if we can transform it so we only solve from 0 to 1 it would be a very good computational improvement. With this in mind, let us introduce variable $x = l/L_c$ that ranges between 0 and 1, since l ranges between 0 and L_c. Since $l = xL_c$,

$$
\begin{aligned}
Q\frac{\partial c}{\partial (xL_c)} &= -A\left(\epsilon_t\ \frac{\partial c}{\partial t} + (1-\epsilon_t)\frac{\partial q}{\partial t}\right) \\
\frac{Q}{L_c}\frac{\partial c}{\partial x} &= -A\left(\epsilon_t\ \frac{\partial c}{\partial t} + (1-\epsilon_t)\frac{\partial q}{\partial t}\right)
\end{aligned}
\tag{2.17}
$$

This transformation is actually better because the range of those derivatives are always fixed. If for some reason we need to switch our column to another of a different size we just need to change L_c and/or A.

We can also make the same transformation for the time derivatives. By knowing the time τ an experiment takes, we can take that value as the upper range of integration and introduce the variable $\theta = t/\tau$ and make the exact same as before,

$$
\begin{aligned}
\frac{Q}{L_c}\frac{\partial c}{\partial x} &= -A\left(\epsilon_t\ \frac{\partial c}{\partial (\theta\tau)} + (1-\epsilon_t)\frac{\partial q}{\partial (\theta\tau)}\right) \\
\frac{Q}{L_c}\frac{\partial c}{\partial x} &= -\frac{A}{\tau}\left(\epsilon_t\ \frac{\partial c}{\partial \theta} + (1-\epsilon_t)\frac{\partial q}{\partial \theta}\right)
\end{aligned}
\tag{2.18}
$$

Since $AL_c = V_c$,

$$
\frac{\tau Q}{V_c}\frac{\partial c}{\partial x} + \epsilon_t\ \frac{\partial c}{\partial \theta} + (1-\epsilon_t)\frac{\partial q}{\partial \theta} = 0 \tag{2.19}
$$

This is the known De Vault equation [33, 62], developed by Don De Vault in 1943.

Another thing that this model considers is that the adsorbed mass is always in equilibrium with the bulk concentration, as explained in Section 1.1.3. This means that the process of adsorption is so fast that the equilibrium between bulk and solid is instantly reached. This assumption is valid for a lot of processes, specially when the components are in very diluted conditions. However, for some processes this does not work and such cases will be discussed further on.

2.1.2 Axial Dispersion of a Fluid Inside a Column

The previous section showed the mass balance of a packed column. However, the model assumed that the liquid was traveling in plug flow mode, much like Figure 1.6b. When we are working with packed beds, this is almost never the case. When the fluid is traveling, it usually starts to exit before the predicted due to the dispersion of the fluid through the particles.

Figure 2.2 gives us an example of both behaviors. If you inject a concentration of salt of 1g/L inside a column that is completely clean, in reality the fluid you injected is traveling in plugflow mode. If the fluid inside the column also travels in plugflow mode, then the outlet will look pretty much as the inlet, represented in the purple curve. However, if the

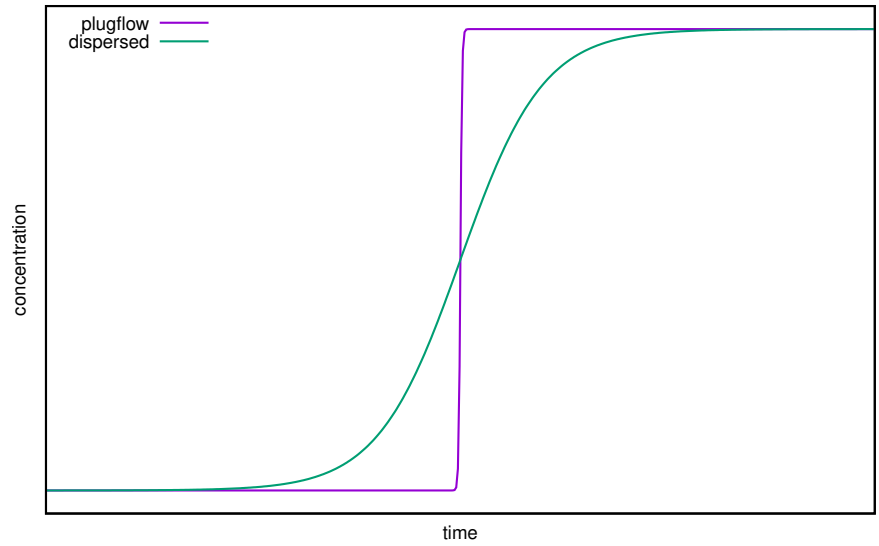

Figure 2.2: Example of a plugflow and an axially dispersed behaviors.

fluid is dispersed throughout the column, then the outlet will be much less steep, much like the green curve.

To mathematically take into account such dispersion, we need to use the Fick Law of Diffusion [63, 64]. This law states that a component travels from a place of high concentration to a place of low concentration to promote homogeneity in a solution. In a column, this means that the solute tends to move faster through the length of the column due to the difference in concentration between the injected solution and the solute-free liquid inside the column. This explains why the components, when dispersion occurs, start exiting the column sooner than when there is no dispersion. In general, Fick diffusion law has the form,

$$J = -D_{ax}\frac{\partial c}{\partial l} \tag{2.20}$$

Where J is the diffusive flux of concentration with units $g/cm^2/min$ and D_{ax} is the axial diffusion coefficient with units cm^2/min. Since it is a spatial flux, it will only influence the inlet and outlet concentrations. Rewriting m_{in} and m_{out} we have:

$$\begin{aligned} m_{in} &= V_{inj}c_l - A\epsilon_t \Delta t D_{ax}\frac{\partial c}{\partial x}\Big|_l \\ m_{out} &= V_{inj}c_{l+\Delta l} - A\epsilon_t \Delta t D_{ax}\frac{\partial c}{\partial x}\Big|_{l+\Delta l} \end{aligned} \tag{2.21}$$

Since the flux term is $g/cm^2/min$, in order for the result to be just g, we need to multiply by $A\Delta t$. The ϵ_t introduced is simply because the fluid traveling cannot pass through the solid, so the effective sectional area used by the fluid is $A\epsilon_t$. Again, in order to use the volumetric flowrate we divide by Δt:

$$\begin{aligned} m_{in} &= Qc_l - A\epsilon_t D_{ax}\frac{\partial c}{\partial x}\Big|_l \\ m_{out} &= Qc_{l+\Delta l} - A\epsilon_t D_{ax}\frac{\partial c}{\partial x}\Big|_{l+\Delta l} \end{aligned} \tag{2.22}$$

We can go further and add the flowrate to the diffusion term. Since $Q = V_{inj}/\Delta t = A\epsilon_t v$, with v being the linear velocity of the fluid with units cm/min, we can multiply v/v to the derivative term. With this we get:

$$\begin{aligned} m_{in} &= Q\left(c_l - \frac{D_{ax}}{v}\frac{\partial c}{\partial x}\Big|_l\right) \\ m_{out} &= Q\left(c_{l+\Delta l} - \frac{D_{ax}}{v}\frac{\partial c}{\partial x}\Big|_{l+\Delta l}\right) \end{aligned} \tag{2.23}$$

Now we have to add this term to our mass balance previously shown. Instead of starting right from the start, let us start with Equation 2.14 and add the diffusion term:

$$\begin{aligned} Q\left(c_l - \frac{D_{ax}}{v}\frac{\partial c}{\partial l}\Big|_l\right) = Q\left(c_{l+\Delta l} - \frac{D_{ax}}{v}\frac{\partial c}{\partial l}\Big|_{l+\Delta l}\right) \\ + A\Delta l\left(\epsilon_t\,\frac{c_{t+\Delta t} - c_t}{\Delta t} + (1-\epsilon_t)\frac{q_{t+\Delta t} - q_t}{\Delta t}\right) \end{aligned} \tag{2.24}$$

Here, we divide again the entire equation by Δl and we obtain a second derivative in the diffusion term, by applying the same logic as before. In this case we get:

$$Q\left(-\frac{\partial c}{\partial l} + \frac{D_{ax}}{v}\frac{\partial^2 c}{\partial l^2}\right) = A\left(\epsilon_t\,\frac{\partial c}{\partial t} + (1-\epsilon_t)\frac{\partial q}{\partial t}\right) \tag{2.25}$$

Again we will add $x = l/L_c$, $\theta = t/\tau$ and $AL_c = V_c$:

$$\frac{\tau Q}{V_c}\left(\frac{D_{ax}}{vL_c}\frac{\partial^2 c}{\partial x^2} - \frac{\partial c}{\partial x}\right) = \epsilon_t\,\frac{\partial c}{\partial \theta} + (1-\epsilon_t)\frac{\partial q}{\partial \theta} \tag{2.26}$$

This is called the equilibrium-dispersive model. The next transformation that we are about to apply is not just for convenience but also to be able to obtain more information from this model. For that reason we want to introduce the Péclet number [65]. The Péclet number is a dimensionless correlation that compares advective with diffusive transport. Advective transport is when a component is transported by a moving fluid (i.e. debris being flushed by a river stream) and diffusive transport is when a component travels from the zone of higher concentration to the zone of lower concentration. The correlation has the form:

$$Pe = \frac{vL_c}{D_{ax}} \tag{2.27}$$

For high Péclet number values, the weight that the diffusive transport has in the mass balance is negligible, meaning that the mass transport is made entirely through advection. The opposite is also true, for very low Péclet number values, the diffusive transport is the only one that has impact in the global mass transport.

Introducing the Péclet number in our equation is quite trivial:

$$\epsilon_t \frac{\partial c}{\partial \theta} + (1 - \epsilon_t)\frac{\partial q}{\partial \theta} = \frac{\tau Q}{V_c}\left(\frac{1}{Pe}\frac{\partial^2 c}{\partial x^2} - \frac{\partial c}{\partial x}\right) \tag{2.28}$$

There is one more thing missing in this model. For us to formulate this equation, we used the cross-sectional area to help us visualize what happens inside the column. However, the logic employed is not valid to the extremities of the column, where $x = 0$ and $x = 1$. This happens because to solve a derivative in a specific point we need at least two values, one defined before said point and one after, that is why for the extremities we need two more equations. The boundary conditions are as follows,

$$c(0,t) - \frac{1}{Pe}\frac{\partial c(0,t)}{\partial x} = c^{in} \tag{2.29}$$

$$\frac{\partial c(1,t)}{\partial x} = 0 \tag{2.30}$$

$$c(1,t) = c^{out} \tag{2.31}$$

In here, c^{in} is the concentration injected inside the column. Without axial dispersion $c(0,t) = c^{in}$, but with axial dispersion the condition in the inlet is the one above. The third equation is introduced just for simplicity reasons, it will be useful later on in this chapter.

At last, we need to introduce the initial condition, at $\theta = 0$, which is the state of the system at the start of a simulation. In this process, the concentrations of every component are null, since the columns are clean from solutes, hence:

$$c(x,0) = 0 \tag{2.32}$$

And we reached the equation that describes the transport of our components throughout a packed column with adsorption.

2.1.3 Linear Driving Force Model

When it comes to the adsorption kinetics, for most of the cases is so fast that the equilibrium-dispersive model suffices. However, for large columns and multi-component systems, the adsorption process has more impediments and so the equilibrium is not achieved instantaneously [66]. When this happens, it means that the molecules have more difficulties to reach the active sites, making the adsorption process much slower than normal. In this case, we say that the mass transfer is the governing phenomenon and we need to be able to quantify it.

To recap, Equation 2.28 represents the mass balance when instant adsorption equilibrium and axial dispersion occur. Now, if we want to take into account mass transfer limitations, let us make some transformations. Let q^* be the adsorbed mass in equilibrium and q the adsorbed mass when equilibrium is not reached. In this case,

$$q^* = f(c) \tag{2.33}$$

Where $f(c)$ can be any isotherm model mentioned before. Considering this, the mass balance previously should have been written as:

$$\epsilon_t \frac{\partial c}{\partial \theta} + (1 - \epsilon_t)\frac{\partial q^*}{\partial \theta} = \frac{\tau Q}{V_c}\left(\frac{1}{Pe}\frac{\partial^2 c}{\partial x^2} - \frac{\partial c}{\partial x}\right) \tag{2.34}$$

However, with mass transfer limitations, we have to change q^* to q because the equilibrium is not achieved. But now we have no way to quantify q. To circumvent this problem, the linear driving force model is used to relate q and q^* in the following way:

$$\frac{\partial q}{\partial \theta} = \tau K_m (q^* - q) \tag{2.35}$$

This model assumes that the driving force, which is the difference between the adsorbed mass in equilibrium and the actual adsorbed mass, is directly proportional to the adsorption rate. The bigger the difference, the faster the adsorption occurs. In this model, K_m is called the mass transfer coefficient. In the end, the mass balance of a chromatographic column assuming axial dispersion and mass transfer limitations is the following:

$$\epsilon_t \frac{\partial c}{\partial \theta} + (1 - \epsilon_t)\frac{\partial q}{\partial \theta} = \frac{\tau Q}{V_c}\left(\frac{1}{Pe}\frac{\partial^2 c}{\partial x^2} - \frac{\partial c}{\partial x}\right) \tag{2.36a}$$

$$\frac{\partial q}{\partial \theta} = \tau K_m (q^* - q) \tag{2.36b}$$

2.1.4 Column Efficiency

The separation efficiency of a column is a very important subject. In a distillation column, the efficiency is measured by the number of plates that we introduce in the column. The higher the number of plates, the higher the efficiency. In a packed column, the length is not discretely measured. In order to apply the same principle, we can calculate the Height Equivalent to a Theoretical Plate (HETP). As the name suggests, it is the column height that corresponds to a distillation column plate. In this case, the lower the height, the bigger the number of theoretical plates, hence higher efficiency. It can be determined as:

$$HETP = \frac{L}{N} \tag{2.37}$$

Where L is the column length and N is the number of theoretical plates. The tricky part is determine N, since it depends entirely on the interaction between the fluid and the solid phase.

According to [67], we can relate the axial dispersion coefficient to the number of theoretical plates by:

$$D_{ax} = \frac{Lv}{2N} \tag{2.38}$$

With some algebraic transformation, we can relate the Péclet number with the number of theoretical plates:

$$N = \frac{Pe}{2} \tag{2.39}$$

It is important to note that these correlations can only be applied with the assumption that the adsorption kinetics is infinitely fast, as described in Section 2.1.2.

2.1.5 Diffusion in a Packed Column

The Linear Driving Force (LDF) model comes into rescue when the equilibrium-dispersive model fails to describe the system in study. The diffusive model is another way, similar to the LDF model, to improve the accuracy of the equilibrium-dispersive model.

The diffusive model states that that the diffusion of the molecules in the packing bed take place in several ways, so the apparent dispersion coefficient D_{ax} is not constant and is the contribution of different terms. The most well-known correlation is the Van Deemter Equation [68]. The correlation is as follows:

$$HETP = A + \frac{B}{v} + Cv \tag{2.40}$$

A is the eddy diffusion. When the packing is not well packed, it creates some preferential paths, and this parameters takes that into account. For columns with no packing, this term is null. B is the molecular diffusion in the longitudinal direction and C is the resistance to mass transfer between the compound and the resin.

We can rearrange the previous equation to have it depending on the Péclet number and the volumetric flowrate [69]:

$$\frac{1}{Pe} = \frac{1}{Pe_h} + \alpha\frac{Q}{V_c} + \beta\frac{V_c}{Q} \tag{2.41}$$

In this case, Pe_h is the hydrodynamic Péclet number, α is a constant representing the resistances of mass transfer and β represents the molecular diffusion in relation to the column length. In this model, Pe is not constant, but instead depends on the solute and the volumetric flowrate. For high flowrates, the impact of molecular diffusion becomes negligible and $\beta \approx 0$.

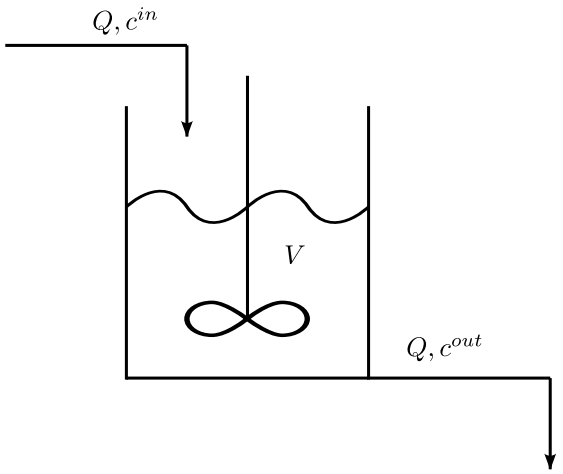

Figure 2.3: Graphical representation of a Constinuously Stirred Tank (CST).

2.1.6 Hydrodynamic Behavior in a Continuously Stirred Tank

A Continuously Stirred Tank (CST), as the name suggests, is a fluid container with continuous stirring. It is normally used to homogenize solutions and also to keep solids in suspension, such as in batch adsorption experiments. The CST model is also widely used to mathematically model dead volumes of experimental systems.

Diagram 2.3 shows what a CST looks like. A flowrate Q with a concentration c^{in} is introduced inside a tank. We assume perfect stirring, which means that the liquid that is being introduced is instantaneously mixed with the fluid already inside the tank. The tank has a volume V and a concentration c^{out}. In this case we assume that the fluid volume is constant, meaning that the inlet flowrate is the same as the outlet flowrate. Also, since we are assuming perfect stirring, the outlet concentration is the same as the tank concentration. To perform the mass balance of a CST we already know that:

$$IN = OUT + ACCUMULATED \tag{2.42}$$

In this case:

$$Qc^{in} = Qc^{out} + V\frac{dc^{out}}{dt} \tag{2.43}$$

The third term is the accumulation term. Through time the concentration will change as we recycle the fluid inside the tank. We also have to establish the initial condition of the CST model:

$$\frac{dc^{out}}{dt} = \frac{Q}{V}\left(c^{in} - c^{out}\right) \qquad \text{for} \quad t > 0 \tag{2.44}$$

$$c^{out} = c_0 \qquad \text{for} \quad t = 0 \tag{2.45}$$

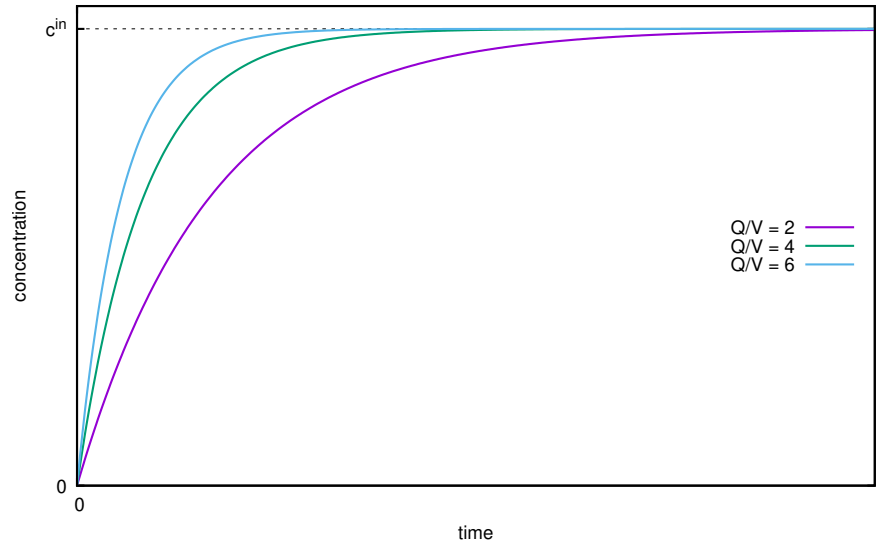

Figure 2.4: Hydrodynamic behavior of a CST.

Figure 2.4 shows the concentration profile of a CST for different Q/V. This ratio controls the steepness of the curve. In other words, it dictates how fast the concentration inside the CST reaches the inlet concentration.

Even though is quite a simple model, it is very important for system modeling, specially when modeling system dead volumes, such as UV sensors, tubing, etc.

2.1.7 Isotherm Models

As mentioned in Section 1.1.3, an isotherm is a mathematical equation that is able to quantify the amount of adsorbed mass in a given resin by knowing the concentration of said component in the bulk. This relation is not always the same for every component, since a lot of components have different structures, different atom numbers, different electronic clouds, different charges, etc. These are some of the factors, from the molecule side, that influence the adsorption behavior. There are also factors from the adsorbent side that influence the adsorption, such as the components that are attached to the surface, the homogeneity of the surface, the material from which the adsorbent is made, etc. Since there are so many combinations of these factors, a lot of isotherm behaviors exist too. Figure 2.5 shows the five types of isotherm behaviors that each adsorbent-adsorbate combination can have. Type I is the most common one, which is also called a favorable isotherm. When the concentration in the bulk continues to increase, so as the adsorbed amount, a saturation point is reached and adsorption is stopped. Type II starts as a favorable isotherm, but when the bulk concentration reaches a critical point, it starts growing exponentially again, just like when the resin as totally clean. This shows that the solute molecules are adsorbing on top of the stationary phase but also on top of each other, creating a second layer of adsorption. It is a special of the type IV isotherm, in which the solubility of the solute in solution is the limiting factor for adsorption. Type III

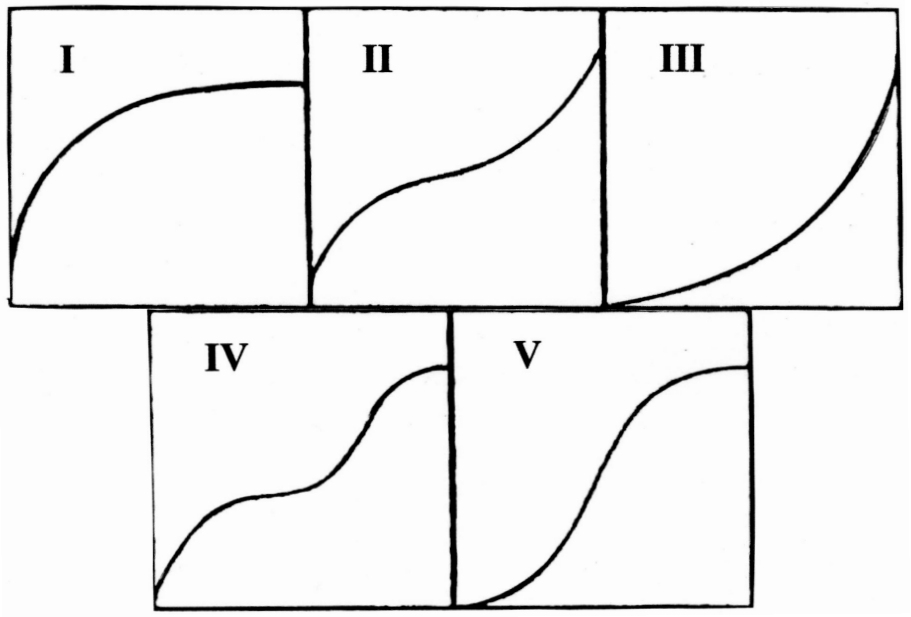

Figure 2.5: The five types of adsorption behavior [70].

is called an unfavorable isotherm, since the adsorbed amount increases with the increase in bulk concentration. It is a special case of the type V isotherm, in which the solubility of the solid in the bulk is the limiting factor when it comes to adsorption. Type IV is a typical multi layered adsorption, where there is an inflexion point in the adsorption curve. This means that, after a certain bulk concentration, the solutes starts adsorbing on top of other solute molecules, since the entire surface is already occupied. Type V appears commonly in resins where the pore distribution is quite wide. In lower bulk concentration the adsorption is quite fast, but after a point it starts to stabilize. In the beginning the adsorption is made in the large-pore beads and when they are saturated, the small-pore particles start to be occupied and eventually the entire resin saturates.

2.1.7.1 Linear Isotherm

The linear isotherm is the most common because it can be applied to all combinations of adsorbent-adsorbate, as long as the component concentration is in diluted conditions. However, for each combination you have to find out what diluted means. Is 0.005g/L too concentrated already? Or is 5g/L still diluted?. For every combination, there is always a range of concentrations in which the adsorbed mass linearly increases, there is not a fixed range. The model is as follows:

$$q = Hc \tag{2.46}$$

Where q is the adsorbed mass and c is the bulk concentration. H is the line slope and is called the Henry constant [71].

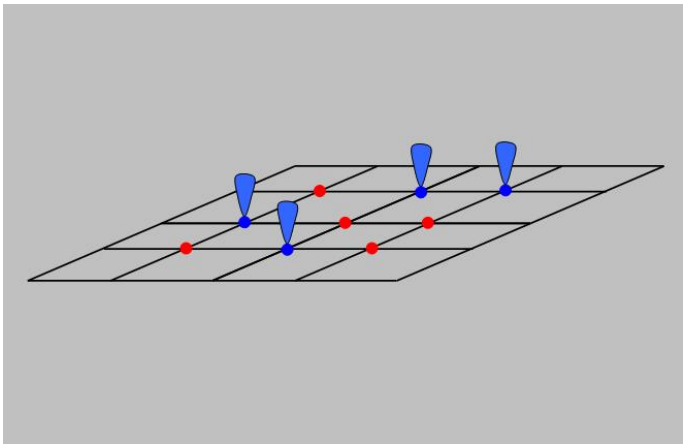

Figure 2.6: Picture of an homogeneous adsorbent surface. In red we have unoccupied sites ready for adsorption and in blue we have already occupied adsorption sites.

2.1.7.2 Langmuir Isotherm

The langmuir isotherm is one of the most popular. It was first described by Irving Langmuir in 1916 and stated that, in a surface, there is a finite number of sites where the molecules can adhere [72]. He also stated that the molecules can only adsorb in the surface but not on top of themselves, which means that the number of active sites existent is equal to the number of molecules that are allowed to adsorb. After the surface is saturated, remaining molecules will stay in the bulk.

Figure 2.6 depicts how a solid surface area would be occupied. Another assumption of the langmuir model is that the surface is homogeneous, meaning that all the sites are identical and they can only adsorb one molecule at the same time. The blue spot are sites that are occupied by a molecule and the red sites are available for the molecules to adhere to. With all of these assumptions, the model takes the form:

$$q = q_m \frac{Kc}{1 + Kc} \tag{2.47}$$

In this model q_m represents the maximum amount of molecules that can be adsorbed in a single layer and K is an equilibrium that shows how prone is a molecule to actually adsorb in the stationary phase.

There are two interesting cases with this model. The first is when c is extremely small. If the concentration is close to 0, then $1 >> Kc$, meaning that the denominator is simply 1. In that case, the model is as follows,

$$q = q_m Kc \tag{2.48}$$

This is very similar to the linear isotherm because it is. The langmuir isotherm, in very diluted conditions, behaves as the linear isotherm because the adsorbed mass is very far from the maximum adsorbed mass allowed from the resin.

The other case is when the bulk concentration is very high. In that case $Kc >> 1$, which makes the model look like:

$$q = q_m \tag{2.49}$$

Indeed this is the case, as explained before. For very high concentrations, the surface area becomes saturated, making it impossible to adsorb more molecules. In those conditions, no matter how much we increase the bulk concentration, the adsorbed mass will remain constant.

2.1.7.3 Freundlich Isotherm

The Freundlich isotherm is an empirical correlation to determine the adsorbed mass [73]. It is called empirical because it was not mathematically derived based on physical assumptions, just like the Langmuir model. It was based on a lot of experiments made and Freundlich came up with the idea of fitting a mathematical equation that did not have any physical meaning but could fit the data pretty well. However, this model is particularly good at fitting adsorption behaviors in heterogeneous surfaces (surfaces which active sites are not all identical to each other). The model is as follows,

$$q = Kc^{1/n} \tag{2.50}$$

Where K and n are just fitting parameters. In general n values different to 1 makes it likely, but not certain, that the surface is heterogeneous in nature.

2.1.7.4 Sips Isotherm

The Sips isotherm directly derives from the langmuir isotherm, but with an added parameter that counts for surface heterogeneity [74]. The model is:

$$q = q_m \frac{Kc^{\beta}}{1 - Kc^{\beta}} \tag{2.51}$$

Where q_m is the maximum adsorbed amount, K is the equilibrium constant and β is a fitting parameter to account for heterogeneity. The farther β is from 1, the more heterogeneous the surface is.

2.1.7.5 Toth isotherm

The Toth isotherm is another modification of the langmuir isotherm to account for heterogeneous surfaces [74]. However, this model can also take into account some multilayer systems [75, 76]. The model is:

$$q = q_m \frac{Kc}{\left[1 + (Kc)^n\right]^{1/n}} \tag{2.52}$$

Where K is the equilibrium constant and n is a fitting factor to account for heterogeneity.

2.1.7.6 Extended Langmuir Isotherm

The extended langmuir isotherm model [77] is an adaptation of the langmuir model for multi-component mixtures. The main premise is that, in a multi-component system, every component compete with each other for the active sites, so the isotherm of each component is dependent on the concentration of all the other components. The model takes this form:

$$q_i = q_m \frac{K_i c_i}{1 + \sum_{i=1}^{NC} K_i c_i} \tag{2.53}$$

In this model, index i represents each component and NC is the total number of components in the mixture. However, if the concentrations of each component are very low, this model is not necessary, since in diluted conditions the available space for adsorption is so abundant that there is no need for the molecules to compete for the active sites.

2.2 System Design

The previous section was meant to introduce the mathematical fundamentals used to simulate physical chromatographic processes. All these fundamentals were used in this work, in order to simulate the physical process we have built in our laboratory. In order to actually built it, we first designed the flowsheet, presented in Figure 2.7. We wanted to build a prototype that would be able to emulate all 3-column or less chromatographic processes. In order to accomplish that, all columns must be linked with each other, so the outlet of every column can be directed to the inlet of every column, excluding itself. Furthermore, the column series is circular, meaning that the previous column of column 1 is column 3. Since we want to implement switching intervals as all continuous processes, every column has a UV signal after the outlet so we can track the concentration profiles in every column. We have three pumps, the feed pump F, the solvent pump E1 and the gradient pump E2. Also, the outlet of all three columns can lead to the waste and product streams, denominated by W and P, respectively. For every column we have five inlets, the F stream, the E1 stream, the E2 stream, and the outlets of the other two columns. Those five inlets can be simultaneously injected inside a column. We added a bypass stream in the connector for the column 1 inlet for test purposes. In the other hand, the outlet of every column can be directed to collect product, discard waste or go to the inlet of one of the other two columns. All of the streams have an ON/OFF pneumatic valve associated, in order to close or open said path.

After the experimental set-up was assembled, there is the need to mathematically describe it, which will be explained in the next section.

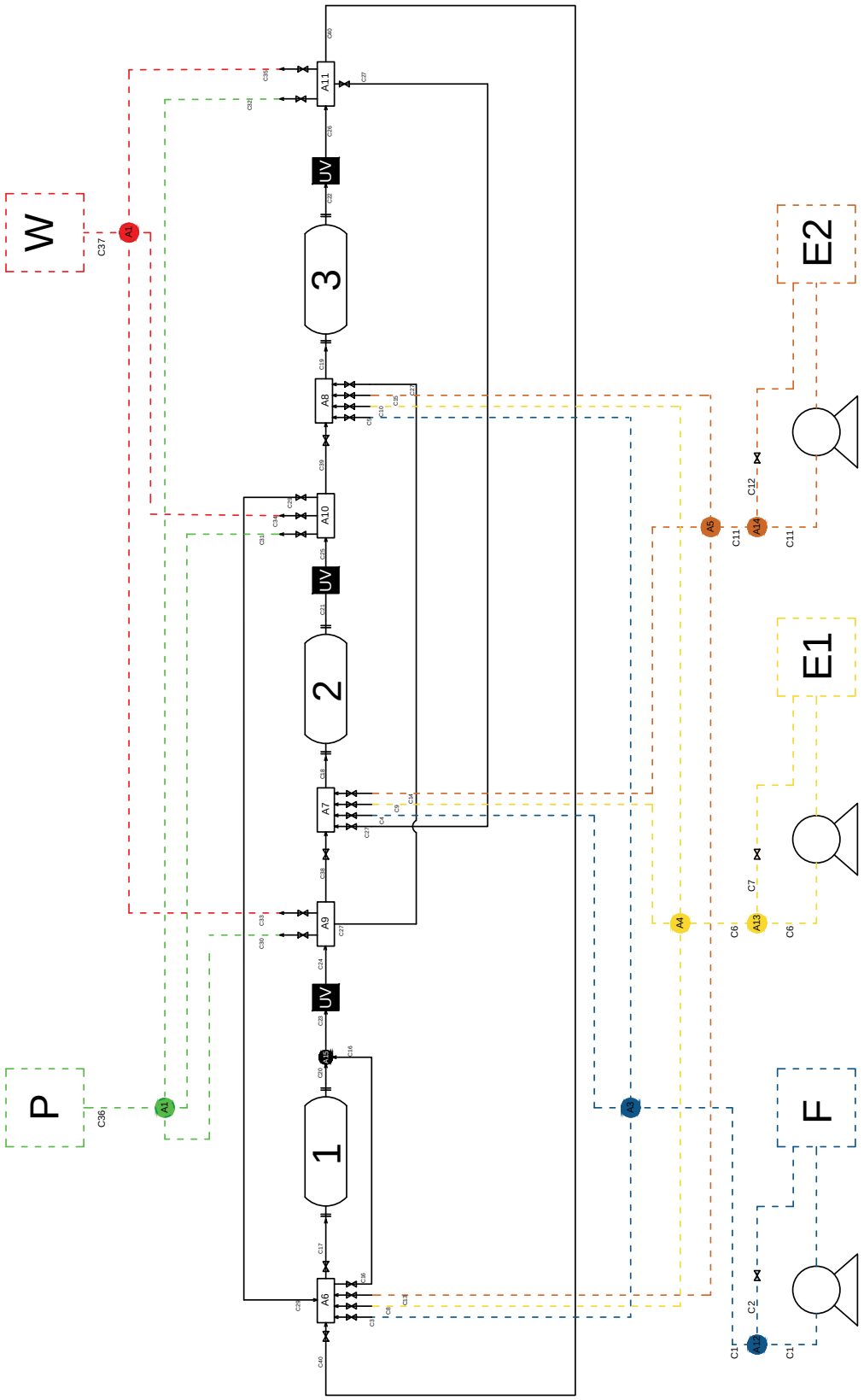

Figure 2.7: Flowsheet for the laboratory set-up. The outlet of every column is connected to the inlet of every column, excluding itself.

2.2.1 Mathematical Description

The prototype assembled was meant to have maximum flexibility in terms of operation. This means that we can use 1, 2 or 3 columns in the same operation step. It also means that the order of the columns used can be any order, so the total number of possible pathways for the fluid to go through is very high. Our mathematical model must be as simple as possible, but at the same time as general as possible, in order to cover any possible event.

Let us start by introducing some nomenclature. $R = \{F, E1, E2\}$ is the set of the raw inlet streams, $I = \{F, E1, E2, PR, N\}$ is the set of inlet streams of each column and $O = \{PR, N, P, W\}$ is the set of outlet streams of each column. PR and N are abbreviations for the previous and next columns, respectively.

Each column has two connectors associated, represented by sets I and O. The mass balance for the inlet connector is:

$$Q_{ic} = \sum_{i \in I} V^{in}_{ic,i} \; Q^{in}_{ic,i} \tag{2.54a}$$

$$c^{in}_{ic,is} \; Q_{ic} = \sum_{i \in I} V^{in}_{ic,i} \; Q^{in}_{ic,i} \; \bar{c}^{in}_{ic,is,i} \tag{2.54b}$$

Where index ic represents the number of columns, index is represents the number of components, V^{in} denotes the ON/OFF valve of the inlet streams, Q^{in} the flowrate of the inlet streams and $\bar{c}^{in}$ the concentration of the inlet streams. Since V^{in} are ON/OFF, or open/close, valves, the only valid values are 0 or 1.

Equation 2.54a is the volumetric balance of the inlet connector. Considering that there are no leakages in the system, the sum of the flowrates of every inlet stream is the flowrate that enters the column. Equation 2.54b follows the same logic but for the mass. Again, if there are no leakages, then the sum of the component mass that is injected by the inlet streams is the mass that enters the column.

Now we can follow the same logic for the outlet connector:

$$Q_{ic} = \sum_{i \in O} V^{out}_{ic,i} \; Q^{out}_{ic,i} \tag{2.55a}$$

$$c^{out}_{ic,is} \; Q_{ic} = \sum_{i \in O} V^{out}_{ic,i} \; Q^{out}_{ic,i} \; \bar{c}^{out}_{ic,is,i} \tag{2.55b}$$

However, for the outlet streams there is an extra restriction. The flowrate that exits the column and goes towards the outlet connector cannot be divided into several streams, opposite to the inlet connector. This means that, at any point in time, there can only be one open outlet valve:

$$\sum_{i \in O} V^{out}_{ic,i} = 1 \tag{2.56}$$

With this consideration, we do not need a sum, just simply compare the outlet streams with the column outlet:

$$Q_{ic,i}^{out} = V_{ic,i}^{out}\ Q_{ic} \tag{2.57a}$$
$$\bar{c}_{ic,is,i}^{out} = V_{ic,i}^{out}\ c_{ic,is}^{out} \tag{2.57b}$$

Even though every inlet and outlet has an associated valve, some of them always share the same value. For instance, the outlet stream N of column 2 is the same as the inlet stream PR of column 3. The same can be said for the outlet stream PR of column 2 being the same as the inlet stream N of column 1. The total combinations are as follows:

$$V_{PR,1,is}^{in} = V_{N,3,is}^{out} \tag{2.58a}$$
$$V_{PR,2,is}^{in} = V_{N,1,is}^{out} \tag{2.58b}$$
$$V_{PR,3,is}^{in} = V_{N,2,is}^{out} \tag{2.58c}$$
$$V_{N,1,is}^{in} = V_{PR,2,is}^{out} \tag{2.58d}$$
$$V_{N,2,is}^{in} = V_{PR,3,is}^{out} \tag{2.58e}$$
$$V_{N,3,is}^{in} = V_{PR,1,is}^{out} \tag{2.58f}$$

Where is is the index representing the solutes in solution, and the second index represents the column number. The same relationships can be applied for the flowrates:

$$Q_{PR,1,is}^{in} = Q_{N,3,is}^{out} \tag{2.59a}$$
$$Q_{PR,2,is}^{in} = Q_{N,1,is}^{out} \tag{2.59b}$$
$$Q_{PR,3,is}^{in} = Q_{N,2,is}^{out} \tag{2.59c}$$
$$Q_{N,1,is}^{in} = Q_{PR,2,is}^{out} \tag{2.59d}$$
$$Q_{N,2,is}^{in} = Q_{PR,3,is}^{out} \tag{2.59e}$$
$$Q_{N,3,is}^{in} = Q_{PR,1,is}^{out} \tag{2.59f}$$

And the concentrations:

$$c_{PR,1,is}^{in} = c_{N,3,is}^{out} \tag{2.60a}$$
$$c_{PR,2,is}^{in} = c_{N,1,is}^{out} \tag{2.60b}$$
$$c_{PR,3,is}^{in} = c_{N,2,is}^{out} \tag{2.60c}$$
$$c_{N,1,is}^{in} = c_{PR,2,is}^{out} \tag{2.60d}$$
$$c_{N,2,is}^{in} = c_{PR,3,is}^{out} \tag{2.60e}$$
$$c_{N,3,is}^{in} = c_{PR,1,is}^{out} \tag{2.60f}$$

The system design expressed in this section was fully generalized in order to accommodate every scenario and possible changes to the system. If we want to increase the number of inlet streams, as well as increase or decrease the outlet streams, it is very easy to transform the problem to suit the new streams. In this way we ensure maximum flexibility to changes in our mathematical model.

2.3 Simulation

In chromatographic processes, either in batch or continuous mode, the associated mathematical models are often too complex to be analytically solved. Instead, we need to use numerical methods in order to solve the system of equations. In this way the problem becomes mathematically solvable, and the implementation of these numerical methods makes it easier for us to implement our problem in a suitable programming language.

2.3.1 Evaluation of the Degrees of Freedom

Before starting implementing the model using a certain software, we have to first evaluate if it is well-posed, and we can see it by checking the degrees of freedom. The degrees of freedom (DOF) of a mathematical model is the difference between the number of equations used to describe our system and the number of variables used by those equations. A system is considered mathematically well described when the number of variables equals the number of equations, meaning that $DOF = NV - NE = 0$. NV and NE are the number of variables and number of equations, respectively. When $DOF = 0$, it means that the NV variables that we have can be calculated and have one and only one valid solution.

An equation system, to be properly solved, always needs a DOF of 0, otherwise we need another approaches to solve it. If the $DOF < 0$, then the problem is poorly described, meaning that, it either lacks equations or there are unnecessary variables in the system, hence a negative DOF is proof that the mathematical model is not well described. However, there are times when $DOF > 0$ and the problem is still well described. Actually, most of the times this is what happens. Sometimes we have variables in our system that do not have any equation associated, but are still very important to our physical process. But even though the problem might be well described, the set of variables will have an infinite set of solutions, because no equation is forcing such variables to be a specific value. Most likely, the excess freedom of such variables will remove the physical sense of the other well described ones, since they usually depend on each other. In this case, we need to add restrictions to our problem. In chromatographic processes, often $DOF > 0$, so we normally set restrictions, such as minimum values of purity and recovery of our POI for instance. We also need to set an objective for our problem, in order for the extra variables to reach a valid solution that satisfy our process.

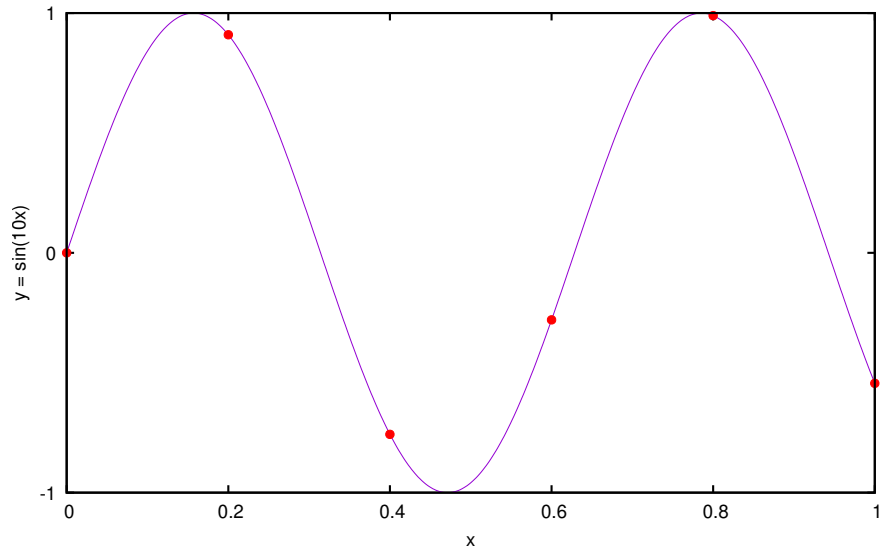

Figure 2.8: Discretization example.

2.3.2 Discretization Methods for Solving Differential Equations

The equations described in the previous sections are considered continuous, meaning that the function can take any value inside a certain interval. However, its complexity makes it impossible for us to reach a solution that is also continuous. A way to overcome that is to transform them from continuous to discrete. If we can make this change, instead of trying to find a solution that would fit the entire continuous domain, we only need to find numerical solutions for each discrete point that we choose.

Let us use the x domain in our mathematical model as an example, which is between 0 and 1. If we used the continuous domain, we would need to find a solution that would be valid in the entire domain, but in this case that is not possible. What we can do is divide $x \in [0, 1]$ domain into a series of discrete points, such as $x \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. Now we only need to solve the equation for six different points. Now we have one issue: a derivative is the slope of the tangent line in a certain point, but to calculate the slope of a tangent we need to use, at least, two points of the equation. If we transform the infinite domain $[0, 1]$ into six points, as we did earlier, we would be able to determine the derivative of each point by using the previous one.

Figure 2.8 shows a graphical example of $y = \sin(10x)$. The points in red mark the discretized domain chosen. Imagine we want to determine the derivative at point $x = 0.2$. To determine the slope in said point we need two values of x, 0.2 and 0 and two values of y, $y(0.2)$ and $y(0)$. The slope is determined as follows,

$$m = \frac{dy}{dx}(0.2) = \frac{\Delta y}{\Delta x}(0.2) = \frac{y(0.2) - y(0)}{0.2 - 0} = \frac{0.909 - 0}{0.2 - 0} = 4.545 \tag{2.61}$$

With m representing the slope and Δ is a way of representing an interval.

Now, if you look at the plot it is obvious that the slope at 0.2 is negative, not positive, which means that our calculation is highly inaccurate. With this example we reach the

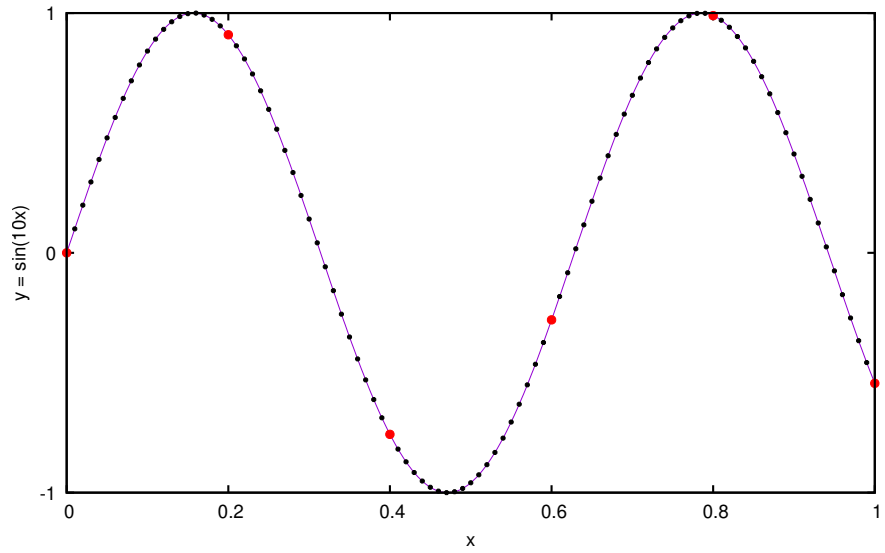

Figure 2.9: Discretization example with good accuracy.

main point to take into account when discretizing a continuous domain: the discretization accuracy. In our example, the accuracy is extremely low, since we divided a continuous domain into six points only. In this case, the approximation calculations of a derivative will be highly inaccurate, specially for functions sensitive to changes, such as the sin function. In the beginning, the best way is to increase the discretized domain. Instead of having only six points, let us have 101 points. Now, our domain will be $x \in \{0 : 0.01 : 1\}$. This nomenclature reads as follows: starting at 0, the next point is the previous point plus the increment value 0.01, ending at 1. The first five values would be $\{0, 0.01, 0.02, 0.03, 0.04\}$. Figure 2.9 shows the 101 discretization points, in black, and the original six points, in red. Let us make the slope calculation at point 0.2 but with the new discretized domain:

$$m = \frac{dy}{dx}(0.2) = \frac{\Delta y}{\Delta x}(0.2) = \frac{y(0.2) - y(0.19)}{0.2 - 0.19} = \frac{0.909 - 0.946}{0.2 - 0.19} = -3.7 \tag{2.62}$$

Now, by looking at the figure, -3.7 is much closer to reality than 4.545. We can actually compare this value with the real derivative value. The derivative function of $\sin(10x)$ is $10\cos(10x)$, so the real slope is:

$$m = \frac{dy}{dx}(0.2) = 10\cos(10 \times 0.2) = -4.161 \tag{2.63}$$

As you can see, even with 101 discretization points, it is still not accurate enough. Again, that is because the sin function is very sensitive, specially close to $x = 0.2$. If we were to calculate it at 0.4, the 6 and the 101 discretizations would give close approximations to reality, simply because between 0.2 and 0.4 the sin function is almost a straight line.

The technique previously used is called Backwards Differentiation (BD). The word backwards comes from the fact that, to determine the derivative of a point, we use the value right before. It is also possible to determine an approximated derivative using the value right after (Forward Differentiation, FD) and by using the one before and the one

after (Central Differentiation, CD). However, none of the three previously mentioned techniques were used to discretize the differential equations in this work.

2.3.2.1 Orthogonal Collocation

The discretization methods mentioned in the previous section used the slope definition to approximate the derivative value of a function in a certain point. Orthogonal Collocation (OC) does something different: it assumes that, for each discretization interval, the function in question can be approximated by a polynomial. This is quite advantageous because deriving a polynomial is quite trivial and its calculation is exact.

Let us look at Figure 2.8 again and assume the discretization domain of six points only, where $x \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. We still want to determine the derivative at $x = 0.2$, but now we will try to approximate a polynomial in the interval $[0, 0.2]$. A polynomial of degree n has the form,

$$P_n(x) = \sum_{i=0}^{n} \alpha_i x^i \tag{2.64}$$

We also need the derivative of $P_n(x)$, $P'_n(x)$:

$$P'_n(x) = \sum_{i=0}^{n} i\alpha_i x^{i-1} \tag{2.65}$$

Now, what we want is to choose a polynomial degree that would suit our purpose. In most cases it is impossible to know that beforehand, so we try several degrees to see what is best. Generally, it is common practice to start with second or third order polynomials, so let us start with a second order polynomial which has the form,

$$P_2(x) = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \tag{2.66}$$

By applying Equation 2.65 we can obtain $P'_2(x)$:

$$P'_2(x) = 0\alpha_0 x^{-1} + 1\alpha_1 x^0 + 2\alpha_2 x^1 = \alpha_1 + 2\alpha_2 x \tag{2.67}$$

The next step is to find the values of the coefficients α_0, α_1 and α_2. For that we will take advantage of the rules of vector multiplication. We can rewrite both $P_2(x)$ and $P'_2(x)$ as a vector multiplication in the following way:

$$P_2(x) = \begin{bmatrix} 1 & x & x^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{2.68}$$

The same for the derivative:

$$P_2'(x) = \begin{bmatrix} 0 & 1 & 2x \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{2.69}$$

As a normal equation system, since we have three unknowns (α coefficients), we need three different equations that involve them to find their values. One way is to give three different values to x and then solve the system. For this example, we will replace x by 0, 0.1 and 0.2. The system of equations would be:

$$P_2(0) = \alpha_0 + 0\alpha_1 + 0\alpha_2 \tag{2.70a}$$
$$P_2(0.1) = \alpha_0 + 0.1\alpha_1 + 0.01\alpha_2 \tag{2.70b}$$
$$P_2(0.2) = \alpha_0 + 0.2\alpha_1 + 0.04\alpha_2 \tag{2.70c}$$

Rewriting in matrix form:

$$\begin{bmatrix} P_2(0) \\ P_2(0.1) \\ P_2(0.2) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0.1 & 0.01 \\ 1 & 0.2 & 0.04 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{2.71}$$

Actually we already know how to calculate the left hand side of the previous equation. This polynomial is being used to approximate the function $\sin(10x)$ in the interval $[0, 0.2]$, so the values on the left hand side can be determined by evaluating $\sin(10x)$ in the domain points chosen. In this way,

$$\begin{bmatrix} P_2(0) \\ P_2(0.1) \\ P_2(0.2) \end{bmatrix} = \begin{bmatrix} \sin(0) \\ \sin(1) \\ \sin(2) \end{bmatrix} = \begin{bmatrix} 0 \\ 0.841 \\ 0.909 \end{bmatrix} \tag{2.72}$$

Adding this equation to Equation 2.71:

$$\begin{bmatrix} 0 \\ 0.841 \\ 0.909 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0.1 & 0.01 \\ 1 & 0.2 & 0.04 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \tag{2.73}$$

In order to determine the α coefficients, we need to determine the inverse of the matrix representing the x values. In matrix algebra, when we want to change a matrix from the right to the left hand side and vice-versa, the matrix has to turn into its inverse. Furthermore, only square matrices can be inverted. The explanation is beyond the scope of this work. With this in mind, we can make the rearrangement:

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0.1 & 0.01 \\ 1 & 0.2 & 0.04 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0.841 \\ 0.909 \end{bmatrix} \tag{2.74}$$

We can easily calculate the inverse matrix by using a software, in this case *Julia* :

$$\begin{bmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ -15 & 20 & -5\\ 50 & -100 & 50\end{bmatrix}\begin{bmatrix}0\\ 0.841\\ 0.909\end{bmatrix} \tag{2.75}$$

With the result being:

$$\begin{bmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\end{bmatrix} = \begin{bmatrix}0\\ 12.275\\ -38.65\end{bmatrix} \tag{2.76}$$

Replacing these values in Equation 2.67:

$$P_2'(0.2) = 12.275 - 2 \times 38.65 \times 0.2 = -3.185 \tag{2.77}$$

As you can see, we could approximate a polynomial with a good degree of accuracy. The derivative value is somewhat close to the real value -4.161. By comparing the BD first approximation with this one, the OC was much better.

Now let us try a third order degree polynomial. We will go straight to the matrix form:

$$P_3(x) = \begin{bmatrix}1 & x & x^2 & x^3\end{bmatrix}\begin{bmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} \tag{2.78}$$

And the derivative:

$$P_3'(x) = \begin{bmatrix}0 & 1 & 2x & 3x^2\end{bmatrix}\begin{bmatrix}\alpha_0\\ \alpha_1\\ \alpha_2\\ \alpha_3\end{bmatrix} \tag{2.79}$$

For a third order polynomial we have four unknowns, so we need four equations to create a well described equation system. We need to choose four x values. For this case 0 and 0.2 will always be used, but what about the other two? For choosing those we will use a mathematical technique called the Gaussian Quadrature **??**. This technique allows us to calculate the optimal x values in order for the polynomial to give the best approximations. However, the expansion and calculation procedure of this technique is beyond the scope of this thesis. The quadrature is only valid in the interval $[-1, 1]$, and the collocation points for a polynomial of degree three are $\{-0.577, 0.577\}$, which converted to the interval $[0, 0.2]$ they become $\{0.042, 0.158\}$. With this we can create our equation system in matrix form:

$$\begin{bmatrix} P_3(0) \\ P_3(0.042) \\ P_3(0.158) \\ P_3(0.2) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0.042 & 0.042^2 & 0.042^3 \\ 1 & 0.158 & 0.158^2 & 0.158^3 \\ 1 & 0.2 & 0.2^2 & 0.2^3 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{2.80}$$

$$\begin{bmatrix} 0 \\ 0.408 \\ 1 \\ 0.909 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0.042 & 0.0018 & 7.409 \times 10^{-5} \\ 1 & 0.158 & 0.025 & 0.0039 \\ 1 & 0.2 & 0.04 & 0.008 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{2.81}$$

And calculate the coefficients:

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0.042 & 0.0018 & 7.409 \times 10^{-5} \\ 1 & 0.158 & 0.025 & 0.0039 \\ 1 & 0.2 & 0.04 & 0.008 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0.408 \\ 1 \\ 0.909 \end{bmatrix}$$
$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 10.435 \\ -13.534 \\ -79.574 \end{bmatrix} \tag{2.82}$$

By calculating the derivative straight from the original expression:

$$P_3'(0.2) = \begin{bmatrix} 0 & 1 & 2 \times 0.2 & 3 \times 0.2^2 \end{bmatrix} \begin{bmatrix} 0 \\ 10.435 \\ -13.534 \\ -79.574 \end{bmatrix} = -4.528 \tag{2.83}$$

Figure 2.10 shows us both approximations, using second and third order degrees polynomials. We see that there is a clear improvement from the BD method.

In these examples we tried to use polynomials to approximate an already known function, but in our mathematical model we do not know the concentration function. There is a way to find it using the OC technique. In our model, the mass balance is comprised by first and second derivatives, so we need to find a way to describe the derivatives as a function of the original function. Using the third degree polynomial for the approximations, the equation system would be the same as presented in Equation 2.80 but now we have no way of calculating the left hand side. First let us develop the same reasoning but for the derivative:

$$\begin{bmatrix} P_3'(0) \\ P_3'(0.042) \\ P_3'(0.158) \\ P_3'(0.2) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 2 \times 0.042 & 3 \times 0.042^2 \\ 0 & 1 & 2 \times 0.158 & 3 \times 0.158^2 \\ 0 & 1 & 2 \times 0.2 & 3 \times 0.2^2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{2.84}$$

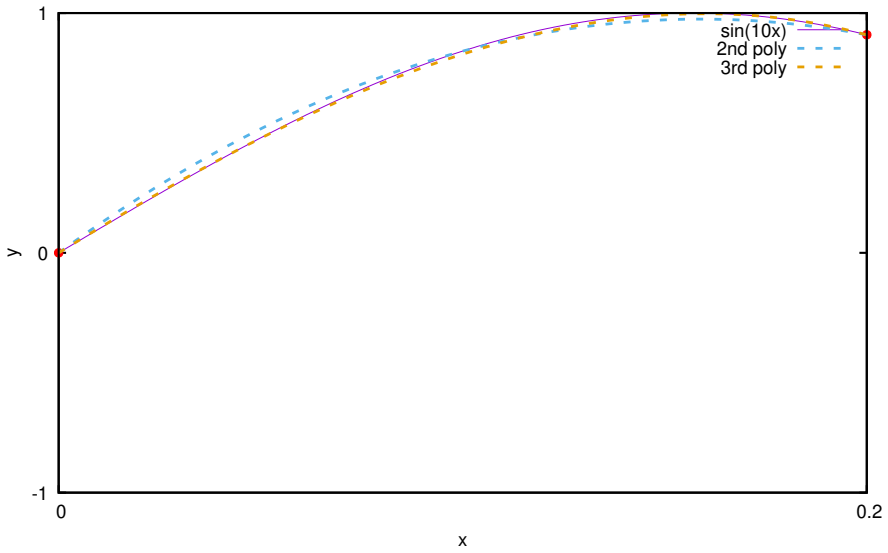

Figure 2.10: Approximate polynomials using OC method.

Before we go any further, we will establish some nomenclature for easiness:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0.042 & 0.042^2 & 0.042^3 \\ 1 & 0.158 & 0.158^2 & 0.158^3 \\ 1 & 0.2 & 0.2^2 & 0.2^3 \end{bmatrix} \tag{2.85}$$

$$B = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 2 \times 0.042 & 3 \times 0.042^2 \\ 0 & 1 & 2 \times 0.158 & 3 \times 0.158^2 \\ 0 & 1 & 2 \times 0.2 & 3 \times 0.2^2 \end{bmatrix} \tag{2.86}$$

The expression for the coefficient matrix has the form:

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = A^{-1} \begin{bmatrix} P_3(0) \\ P_3(0.042) \\ P_3(0.158) \\ P_3(0.2) \end{bmatrix} \tag{2.87}$$

Now we can substitute in the derivative system of equations:

$$\begin{bmatrix} P_3'(0) \\ P_3'(0.042) \\ P_3'(0.158) \\ P_3'(0.2) \end{bmatrix} = BA^{-1} \begin{bmatrix} P_3(0) \\ P_3(0.042) \\ P_3(0.158) \\ P_3(0.2) \end{bmatrix} \tag{2.88}$$

We can also do it for the second derivative, which will come in handy later on:

$$P_3''(x) = \begin{bmatrix} 0 & 0 & 2 & 6x \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{2.89}$$

The equation system will become:

$$\begin{bmatrix} P_3''(0) \\ P_3''(0.042) \\ P_3''(0.158) \\ P_3''(0.2) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 6 \times 0.042 \\ 0 & 0 & 2 & 6 \times 0.158 \\ 0 & 0 & 2 & 6 \times 0.2 \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} \tag{2.90}$$

Adding some nomenclature:

$$C = \begin{bmatrix} 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 6 \times 0.042 \\ 0 & 0 & 2 & 6 \times 0.158 \\ 0 & 0 & 2 & 6 \times 0.2 \end{bmatrix} \tag{2.91}$$

The system becomes:

$$\begin{bmatrix} P_3''(0) \\ P_3''(0.042) \\ P_3''(0.158) \\ P_3''(0.2) \end{bmatrix} = CA^{-1} \begin{bmatrix} P_3(0) \\ P_3(0.042) \\ P_3(0.158) \\ P_3(0.2) \end{bmatrix} \tag{2.92}$$

The square matrices BA^{-1} and CA^{-1} are the associated matrices used for approximating first and second derivatives, respectively. Now we can try and generalize this method. As seen before, a polynomial of degree n has $n+1$ coefficients, so we also need $n+1$ x points to create the equation system. The general form is as follows,

$$\begin{bmatrix} P_n(x_1) \\ P_n(x_2) \\ \vdots \\ P_n(x_{n+1}) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & \dots & x_1^n \\ 1 & x_2 & \dots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n+1} & \dots & x_{n+1}^n \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \tag{2.93a}$$

$$\begin{bmatrix} P_n'(x_1) \\ P_n'(x_2) \\ \vdots \\ P_n'(x_{n+1}) \end{bmatrix} = \begin{bmatrix} 0 & 1 & \dots & nx_1^{n-1} \\ 0 & 1 & \dots & nx_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \dots & nx_{n+1}^{n-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \tag{2.93b}$$

$$\begin{bmatrix} P_n''(x_1) \\ P_n''(x_2) \\ \vdots \\ P_n''(x_{n+1}) \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & n(n-1)x_1^{n-2} \\ 0 & 0 & \dots & n(n-1)x_2^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & n(n-1)x_{n+1}^{n-2} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \tag{2.93c}$$

By adding three more variables:

$$A = \begin{bmatrix} 1 & x_1 & \dots & x_1^n \\ 1 & x_2 & \dots & x_2^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n+1} & \dots & x_{n+1}^n \end{bmatrix} \tag{2.94a}$$

$$B = \begin{bmatrix} 0 & 1 & \dots & nx_1^{n-1} \\ 0 & 1 & \dots & nx_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \dots & nx_{n+1}^{n-1} \end{bmatrix} \tag{2.94b}$$

$$C = \begin{bmatrix} 0 & 0 & \dots & n(n-1)x_1^{n-2} \\ 0 & 0 & \dots & n(n-1)x_2^{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & n(n-1)x_{n+1}^{n-2} \end{bmatrix} \tag{2.94c}$$

And rearranging the general equations:

$$\begin{bmatrix} P_n(x_1) \\ P_n(x_2) \\ \vdots \\ P_n(x_{n+1}) \end{bmatrix} = A \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \tag{2.95a}$$

$$\begin{bmatrix} P_n'(x_1) \\ P_n'(x_2) \\ \vdots \\ P_n'(x_{n+1}) \end{bmatrix} = BA^{-1} \begin{bmatrix} P_n(x_1) \\ P_n(x_2) \\ \vdots \\ P_n(x_{n+1}) \end{bmatrix} \tag{2.95b}$$

$$\begin{bmatrix} P_n''(x_1) \\ P_n''(x_2) \\ \vdots \\ P_n''(x_{n+1}) \end{bmatrix} = CA^{-1} \begin{bmatrix} P_n(x_1) \\ P_n(x_2) \\ \vdots \\ P_n(x_{n+1}) \end{bmatrix} \tag{2.95c}$$

These three matrix-form equations are the basis for the implementation of the Orthogonal Collocation method, which we will apply next.

2.3.3 Discretization of the equilibrium-dispersive model using OC

The equilibrium-dispersive model is an equation that describes the concentration profile of a solute through a column and through time. Since it is a differential equation, we still need to find the solution that solves it, hence we can use the OC method to numerically solve this equation. By looking back at Equation 2.28, we know that the

concentration profile is a function that depends on time and space, but the equation itself only contains derivative terms, meaning that we can implement the OC method presented in Equations 2.95, but first we need to discretize the spatial and temporal domains. As a start, we will divide the domains into 10 discretization elements, so $x \in \{0 : 0.1 : 1\}$ and $\theta \in \{0 : 0.1 : 1\}$. We call it elements because between them we still have to introduce the actual collocation points, according to the gauss quadrature.

First we will have to convert the collocation points for each element in our domain. For any $y \in [-1, 1]$, we can determine $x \in [a, b]$ using the following formula:

$$x = a + (y + 1)\frac{b - a}{2} \tag{2.96}$$

We already know that the collocation points for a third degree polynomial in $[-1, 1]$ are $\{-0.577, +0.577\}$, so converting to a $[a, b]$ interval:

$$x_2 = a + (-0.577 + 1)\frac{b - a}{2} \tag{2.97a}$$

$$x_3 = a + (+0.577 + 1)\frac{b - a}{2} \tag{2.97b}$$

The usage of 2 and 3 as indices is due to the fact that, for a third order polynomial, each element possesses 4 points (two collocation points plus the two borders). We need to apply it to every element, such as $[0, 0.1]$, $[0.1, 0.2]$, up to $[0.9, 1]$. Let us assume that NE is the number of elements (10 in this example). Now, since the first point of the second element is the same as the last point of the first element and so on until the tenth element is reached, the actual differential equation is not applied in the borders of the elements, only in the collocation points, in order to avoid calculating the same point in two different ways. Hence, the discretized mass balance, directly applying Equations 2.95 in the spatial domain, is as follows,

$$\begin{aligned} \epsilon_t \frac{\partial c(\theta, x_{i,j})}{\partial \theta} + (1 - \epsilon_t)\frac{\partial q(\theta, x_{i,j})}{\partial \theta} = \\ \frac{\tau Q}{V_c}\left(\frac{1}{Pe}\sum_{k=1}^{n+1}\left(DD_{j,k}\, c(\theta, x_{i,k})\right) + \sum_{k=1}^{n+1}\left(D_{j,k}\, c(\theta, x_{i,k})\right)\right) \quad , \\ \text{for} \quad i = 1, .., NE \, , \quad j = 2, .., n \end{aligned} \tag{2.98}$$

Where i is the element index, j is the collocation point, $DD = CA^{-1}$ is the second derivative matrix and $D = BA^{-1}$ is the first derivative matrix. Furthermore, $D_{k,l}$ represents the matrix value in row k and column l. The same applies for DD. We still need to find a way to determine the concentration values for the element borders. Since each element has a point in common with the neighbor elements, we will introduce a continuity condition, which states that the derivative in the intersection of said elements must be the same, hence:

$$\sum_{k=1}^{n+1} \left(D_{n+1,k}\ c(\theta, x_{i-1,k}) = \sum_{k=1}^{n+1} \left(D_{1,k}\ c(\theta, x_{i,k})\right)\right), \quad \text{for} \quad i = 2, .., NE \tag{2.99}$$

Regarding the first and the last elements, the continuity conditions are not valid for the first and last values, respectively, and that is where the boundary conditions enter:

$$c(\theta, x_{1,1}) - \sum_{k=1}^{n+1} \left(D_{1,k}\ c(\theta, x_{1,k})\right) = c^{in} \qquad \text{for} \quad x = 0 \tag{2.100}$$

$$\sum_{k=1}^{n+1} \left(D_{n+1,k}\ c(\theta, x_{n+1,k})\right) = 0 \qquad \text{for} \quad x = 1 \tag{2.101}$$

Now we can apply the same logic for the temporal domain. However, there is a small difference comparing to the spatial domain. The spatial domain has boundary conditions at $x = 0$ and $x = 1$, while the temporal domain only has the initial condition, at $\theta = 0$. When applying the gauss quadrature, the collocation points will be located in the interval $[-1, 1)$, but for the temporal domain, since we have only one boundary condition, we need for one of the ends to be included in the quadrature. Luckily, this happens in the Radau Quadrature [78]. The radau quadrature follows the exact same principles as the Gauss one, with the exception that the collocation points include -1. In this case, for a polynomial of order three, the collocation points according to radau quadrature are $\{-1, 0.333\}$. However, what we actually need is for the collocation points to include 1 and not -1, so we need to invert them. Again let $y \in [-1, 1)$ and $x \in (a, b]$. The conversion rule is:

$$x = b - (y + 1)\frac{b - a}{2} \tag{2.102}$$

Now we can discretize the time derivatives, both at c and q:

$$\begin{aligned} &\epsilon_t \sum_{k=1}^{n_\theta+1} \left(DT_{j_\theta,k}\ c(\theta_{i_\theta,k}, x_{i_x,j_x})\right) + (1 - \epsilon_t) \sum_{k=1}^{n_\theta+1} \left(DT_{j_\theta,k}\ q(\theta_{i_\theta,k}, x_{i_x,j_x})\right) = \\ &\frac{\tau Q}{V_c}\left(\frac{1}{Pe} \sum_{k=1}^{n_x+1} \left(DD_{j_x,k}\ c(\theta_{i_\theta,j_\theta}, x_{i_x,k})\right) + \sum_{k=1}^{n_x+1} \left(D_{j_x,k}\ c(\theta_{i_\theta,j_\theta}, x_{i_x,k})\right)\right), \\ &\text{for} \quad i_\theta = 1, .., NE_\theta\ ,\ j_\theta = 2, .., n_\theta\ ,\ i_x = 1, .., NE_x\ ,\ j_x = 2, .., n_x \end{aligned} \tag{2.103}$$

Where NE_t is the number of temporal elements, i_t the index for temporal elements, j_t the temporal collocation point, NE_x the number of spatial elements, i_x the index for spatial elements, j_x the spatial collocation point and $DT = BA^{-1}$ but with different collocation points than D, since the used quadrature is not the same.

Since the temporal collocation points include the right-hand side border, there is no need to include a continuity condition, only the initial condition, at $\theta = 0$:

$$c(0, x) = 0 \tag{2.104}$$

We finally finished discretizing the equilibrium-dispersive model, now let us evaluate the degrees of freedom. First we need to calculate how many points we have for each domain. For the spatial domain we have NE_x elements and each element has n collocation points, so the total number of points in the spatial domain is:

$$N_x = n_x \times NE_x + NE_x + 1 \tag{2.105}$$

$NE_x + 1$ is the number of element extremities. Each element has two extremities, but one of them is shared with the neighbor element, except the first one.

For the temporal domain, the collocation points include one of the extremities, hence:

$$N_\theta = n_\theta \times NE_\theta + 1 \tag{2.106}$$

Regarding the variable number, we have only two, c and q (the rest are parameters known beforehand), both spread in the entire spatial and temporal domains:

$$N_{vars} = 2 \times N_x N_\theta \tag{2.107}$$

Now, the model is valid in the collocation points only, so we have $n_\theta NE_\theta \times n_x NE_x$ differential equations. The continuity conditions are valid between elements and excluding the spatial extremities and $\theta = 0$, so $(N_\theta - 1) \times (NE_x - 1)$. In the end, we have 2 boundary conditions in the entire temporal domain, $2N_\theta$. The initial condition is valid for the entire spatial domain but the boundaries, so there are $N_x - 2$ initial conditions. We cannot forget also the isotherm models to represent the adsorbed mass in the resin, which are $N_\theta N_x$ equations:

$$q(\theta, x) = f(\theta, x) \tag{2.108}$$

Where f is either an isotherm or a mass transfer model.

The total number of equations is:

$$N_{eqs} = n_\theta NE_\theta \times n_x NE_x + (N_\theta - 1)(NE_x - 1) + 2N_\theta + N_x - 2 + N_\theta N_x \tag{2.109}$$

Expanding:

$$
\begin{aligned}
N_{eqs} &= n_\theta NE_\theta \times n_x NE_x + n_\theta NE_\theta (NE_x - 1) + 2N_\theta + N_x - 2 + N_\theta N_x \\
N_{eqs} &= n_\theta NE_\theta (n_x NE_x + NE_x - 1) + 2n_\theta NE_\theta + 2 + N_x - 2 + N_\theta N_x \\
N_{eqs} &= n_\theta NE_\theta (n_x NE_x + NE_x + 1) + N_x + N_\theta N_x \\
N_{eqs} &= n_\theta NE_\theta N_x + N_x + N_\theta N_x \\
N_{eqs} &= (n_\theta NE_\theta + 1) N_x + N_\theta N_x \\
N_{eqs} &= N_\theta N_x + N_\theta N_x \\
N_{eqs} &= 2N_\theta N_x
\end{aligned}
\tag{2.110}
$$

As we can see, $N_{eqs} = N_{vars}$, which means that $DOF = 0$ and our problem is well described.

2.4 Optimization

When performing a simulation, we need a $DOF = 0$. To do that, we need to set values to all the parameters in the problem. But what if we do not want to? What if we actually want to determine said parameters according to some restrictions? For that we have to convert those parameters into variables, which makes $DOF > 0$. Now we have an optimization problem, in which we want to determine the best set of values for all the variables that we "unfixed".

2.4.1 The Concept of Optimum

It is known that to analytically determine the minimum value of a function we need to describe the derivative of that function and find where its value is 0. In other words, for a function $f(x)$, its minimum value is found when:

$$\frac{df}{dx} = 0 \tag{2.111}$$

However, when finding the roots of the derivative function, we do not know if it is either a minimum or a maximum. For that we have to evaluate the derivative function right before and right after the root.

Using our function $f(x)$, assume that it has one, and only one, minimum/maximum and x_1 is the root of the derivative. Mathematically speaking:

$$\frac{df}{dx}(x_1) = 0 \tag{2.112}$$

To evaluate the function before and after x_1, let us create a positive variable $\delta > 0$. This will give us two possible scenarios. The first one:

$$\begin{aligned} \frac{df}{dx}(x_1 - \delta) > 0 \\ \frac{df}{dx}(x_1 + \delta) < 0 \end{aligned} \tag{2.113}$$

And the second one:

$$\begin{aligned} \frac{df}{dx}(x_1 - \delta) < 0 \\ \frac{df}{dx}(x_1 + \delta) > 0 \end{aligned} \tag{2.114}$$

The first case tells us there is a maximum at x_1 and the second case tells us there is a minimum. This evaluation protocol is quite standard because it is simple and accurate, as long as you are sure there is only one extreme point between $x_1 - \delta$ and $x_1 + \delta$. In order to avoid including more than one extreme in that interval we can establish a small δ value and we are good to go.

This technique requires a detailed evaluation of the derivative function inside the interval we are evaluating. For simple functions this is quite intuitive and straightforward, but for more complex functions it is not at all trivial. For instance, let us state that $f(x) = x\sin(x)$ and we want to find the minimum of this function in the interval $[0, 15]$. First we need to obtain the derivative function:

$$\frac{df}{dx} = \frac{d}{dx}(x\sin(x)) = \sin(x) + x\cos(x) \tag{2.115}$$

And determine where it is null:

$$\begin{aligned} \sin(x) + x\cos(x) &= 0 \\ \sin(x) &= -x\cos(x) \\ \tan(x) &= -x \end{aligned} \tag{2.116}$$

Both functions are shown in Figure 2.11. This equation is actually impossible to solve analytically, so we have to use numerical tools to obtain the roots of the derivative. The Newton-Raphson method [79] is usually a very good tool for that since it is easy to implement and is very accurate, as long as you provide a decent initial guess. Furthermore, since this is a periodic function, there is more than one root, and we do not know if said root belongs to a maximum or a minimum, so we need to find all the roots in the domain and evaluate if it is a minimum or a maximum, as explained before. On top of that, if a function has several minima, we can categorize them as being a local or a global minimum. The global minimum is the best of all the minima, while the others are considered local minima. Obviously, the one we desire the most is the global minimum. To summarize, to discover the global minimum of a function inside a specific domain we can follow these four steps:

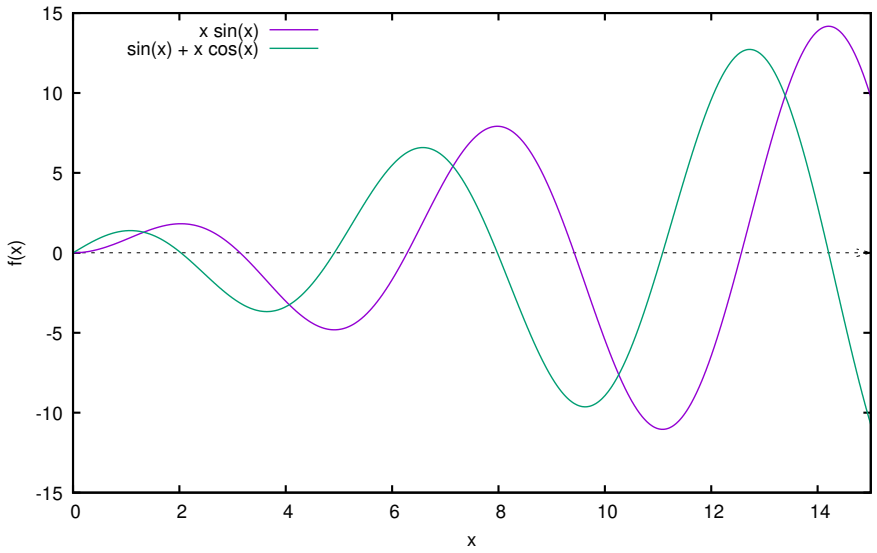

Figure 2.11: Example of functions with multiple extreme points.

1. Find the derivative function of $f(x)$, $\frac{df}{dx}$.
2. Find the roots of $\frac{df}{dx}$.
3. Evaluate if it is a maximum or a minimum and discard the maxima.
4. Evaluate all the minima and verify which one is the global.

This is a specific example to find the global minimum of a function, but these steps also work perfectly if you want to find the global maximum.

In the proposed example, we have $f(x)$, we could easily derive and obtain $\frac{df}{dx}$, we can graphically represent both functions and verify which roots represent minima and which of the minima is the best. By looking at Figure 2.11, we can comfortably state that the global minimum is inside the interval $[10, 12]$. By applying the Newton-Raphson method in $\frac{df}{dx}$ using an initial guess of 11, the mean of the interval chosen, we determined that the root is ~ 11.09, where $f(11.09) \approx -11.041$.

Now, not every problem is as simple as this one, since we know $f(x)$ and the derivative is quite easy to analytically determine. However, even if we know $f(x)$, it might be too complicated to analytically derive, so instead of finding the derivative, we have to use numerical methods to determine it. Orthogonal Collocation is a very good method to derive a function in a specific interval by approximating it with a polynomial in said interval. After the polynomial approximation, finding the derivative is simple. However, to apply such a technique we first need to graphically examine $f(x)$ and establish a good interval where the global minimum is located and then apply the OC method.

Unfortunately real-life problems are much more complex. Most of the times we do not even know $f(x)$, let alone the derivative. One example is the one described in Sections 2.1.1 and 2.1.2, where the mass balance in a packed column is governed by a differential

equation. In this case, even $f(x)$ has to be determined by numerical methods, since the equation in question cannot be analytically solved. In Section 2.3.2 we explained how to discretize a domain, but we did not talk about how we would actually solve the system of equations. That will be further discussed in Section 2.5.

2.4.2 Linear Optimization

A problem is defined as linear when all the equations and restrictions of the problem can be formulated as a linear combination of all the variables. For example, let us consider the following function:

$$f(x, y, z) = \alpha x + \beta y + \gamma z \tag{2.117}$$

Where x, y and z are variables and α, β and γ are parameters. In this case, f is considered a linear function, because it can be expressed as an addition of all variables.

As an example consider two products, $P_1 items/day$ and $P_2 items/day$, which can be sold at a profit of $S_1\$/item$ and $S_2\$/item$, respectively. The factory that produces them can only produce a maximum of $M_1 items/day$ and $M_2 items/day$, due to the amount of raw materials in stock. There is also another constraint, which is the time the machines that are available take to produce the products, hence, the factory is only able to produce a total amount of $TP items/day$ of both products combined. Since there are no more restrictions to our problem, we can describe it. We know that P_1 needs to be less than or equal to M_1 and bigger than or equal to zero (the product amount cannot be negative):

$$0 \leq P_1 \leq M_1 \tag{2.118}$$

The same logic is applied for P_2, hence:

$$\begin{aligned} 0 \leq P_1 \leq M_1 \\ 0 \leq P_2 \leq M_2 \end{aligned} \tag{2.119}$$

The sum of both products cannot exceed TP:

$$P_1 + P_2 \leq TP \tag{2.120}$$

Our main goal is to maximize our daily profit, which is given by the following expression:

$$profit = max\big(S_1 P_1 + S_2 P_2\big) \tag{2.121}$$

Now we can give values to the parameters so we can actually solve the problem. The parameters used for this example are displayed in Table 2.1.

Since we only have two optimization variables, P_1 and P_2, we can graphically create the problem, as shown in Figure 2.12. The depicted lines represent the restrictions of the

Table 2.1: Parameter values for the linear optimization example.

Parameter	Value	Units
M_1	8	items/day
M_2	10	items/day
S_1	10	\$/item
S_2	5	\$/item
TP	13	items/day

problem, while the painted region is the area where every pair (P_1, P_2) is valid and fulfills all restrictions. But how will we determine which of those infinite valid pairs gives the best profit? There is a method, called the Simplex Method [80, 81], which states that the maximum value of the objective function lies in one of the vertices of the feasible region. In this example, such vertices are $(0, 0)$, $(0, 10)$, $(3, 10)$, $(8, 5)$ and $(8, 0)$. In order to evaluate them, we just need to replace each value pair in our objective function:

$$\begin{aligned} profit &= 10 \times 0 + 5 \times 0 = 0 \\ profit &= 10 \times 0 + 5 \times 10 = 50 \\ profit &= 10 \times 3 + 5 \times 10 = 80 \\ profit &= 10 \times 8 + 5 \times 5 = 105 \\ profit &= 10 \times 8 + 5 \times 0 = 80 \end{aligned} \tag{2.122}$$

It is now clear that the best profit of 105 $\$/day$ comes when $P_1 = 8\ items/day$ and $P_2 = 5\ items/day$ [1].

2.4.2.1 The Knapsack Problem

The knapsack problem is a widely known linear optimization problem. To help us understand the topic let us assume we wanted to steal a shop. During the planning stage, we figured that there are N valuable items in that store and each will give us a profit P_i, with $i = 1, .., N$ representing each item. We know that the bag we have available has a maximum volume V_m and each item has a volume V_i. Hence, during the planning stage, based on the maximum volume of our bag, we want to determine the items that we can bring that give us the best profit. With that in mind we now can mathematically describe our problem:

$$V_t = \sum_{i=1}^{N} y_i V_i \tag{2.123}$$

V_t is the total volume of the items chosen. y_i is a binary variable, meaning that can only be 0 or 1. This variable is to help us choose the items that can be introduced in our bag in order to maximize our profit. We also know that:

[1] This problem was adapted from [82]

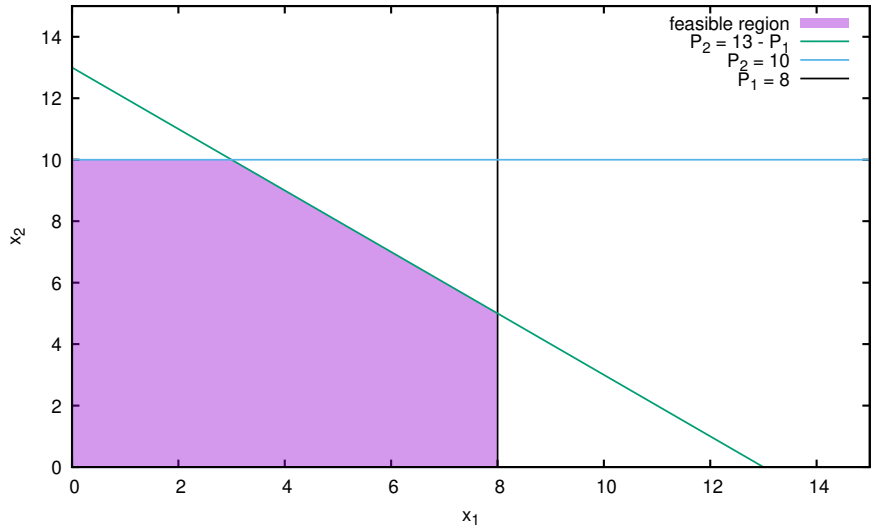

Figure 2.12: Example of a linear optimization problem. In this figure, $M_1 = 8$, $M_2 = 10$, $S_1 = 10$, $S_2 = 10$, $TP = 13$

Table 2.2: Set of values for solving the knapsack problem.

Items	Profit ($/item)	Volume (L/item)
1	3	5
2	2	3
3	6	5
4	2	1
5	13	12

$$\begin{aligned} V_t &\leq V_m \\ \sum_{i=1}^{N} y_i V_i &\leq V_m \end{aligned} \tag{2.124}$$

Since we want to maximize profit, the objective function becomes:

$$OF = max\left(\sum_{i=1}^{N} y_i P_i\right) \tag{2.125}$$

We can see that all equations are linear, since there are only additions between variables.

We can build a more concrete example. Table 2.2 contains a set of parameters that constitute a knapsack problem. Furthermore, is was assumed $V_m = 10L$.

To be noted that in this example $N = 5$, which is greater than 2. This means that the problem cannot be graphically solved. This specific example was solved using the *AMPL* programming language [83]. The algorithm that *AMPL* used is called *cplex* [84], which uses the Simplex algorithm to determine the best value for the function for the objective

function. The being said, with all the parameters stated before, the maximum profit of 10$ is obtained if we steal items 2, 3 and 4. This means that binary variable $y = \{0, 1, 1, 1, 0\}$.

2.4.3 Non-linear optimization

A problem is considered non-linear when there are multiplications between variables or powers. If $f(x, y) = 2x + 3y$ is linear, $f(x, y) = 2x^2 + 3y$ and $f(x, y) = 2xy$ are not, for example. This is extremely important, since the loss of linearity increases the problem complexity and forces us to use different numerical algorithms.

Again, consider the example in Section 1.1.2.1. We have adsorption experimental data that relates the amount of adsorbed mass in the resin with the concentration in the liquid. The actual values presented in Figure 1.2 are presented in Table 2.3. We have a good amount of experimental data, but now we desire to find a model that describes the adsorbed mass at any bulk concentration, not just those 13 points shown. By comparing Figures 1.2 and 2.5, our example is clearly a Type I isotherm, which is normally well described by the Langmuir isotherm model. Now, the Langmuir isotherm has two parameters, q_m and K, that need to be determined. The Langmuir isotherm has the form:

$$q = q_m \frac{Kc}{1 + Kc} \tag{2.126}$$

We want to determine the adsorbed amount predicted by our model in the experimental points tested:

$$q = q_m \frac{Kc_e}{1 + Kc_e} \tag{2.127}$$

With c_e representing the experimental bulk concentration points. It is clearly that the langmuir model is not linear, since there is a division between variables, which makes this a non-linear optimization problem. To solve it, we need an objective function. In this case, we want to minimize the distance between the experimental and the calculated values. Hence, our objective function becomes something like,

$$OF = min\left(\sum_{i=1}^{13}(q_i - q_{e,i})^2\right) \tag{2.128}$$

Basically, we want to determine the values of q_m and K so that the calculated values be as close as possible to the experimental ones. The square on the objective function is required because we are calculating distances between values, so we want them to be as close to zero as possible.

This specific example was solved using the Levenberg-Marquardt algorithm [85, 86], which will be discussed further in more detail. The objective function was minimized by manipulated both Langmuir parameters, which gave us the values of $q_m = 2.167 g/L$ and $K = 0.183 L/g$. The graphical representation is shown in Figure 2.13. q_m represents the maximum adsorbed amount that the resin can endure, and we know, by looking at

Table 2.3: Experimental points obtained by the adsorption experiment using 1g of SALTY.

c_e	q_e
0.0	0.000
0.1	0.050
0.2	0.100
0.5	0.250
1.0	0.333
2.0	0.500
5.0	1.000
10.0	1.429
15.0	1.667
20.0	1.739
25.0	1.800
30.0	1.800
35.0	1.800

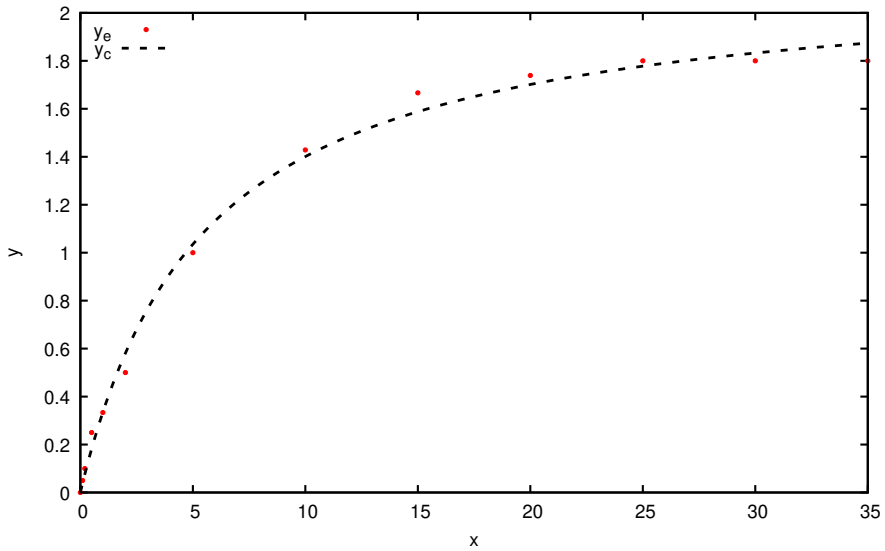

Figure 2.13: Solution of the optimization problem.

Table 2.3, that this value is $\sim 1.8 g/L$. There is a difference between the calculated and the experimental q_m, and it is up to the person optimizing this problem that has to decide if the difference is acceptable or too big. If the fitting is not satisfying, you can simply change the isotherm model to a different that might fit better your experimental data.

2.5 Numerical Algorithms

In the real-world we seldom come into mathematical models simple enough to be analytically solved. For that reason, several ways to numerically solve such models were developed and improved over the years [87–90].

This section will talk about the algorithms used in this work, specially the ones that

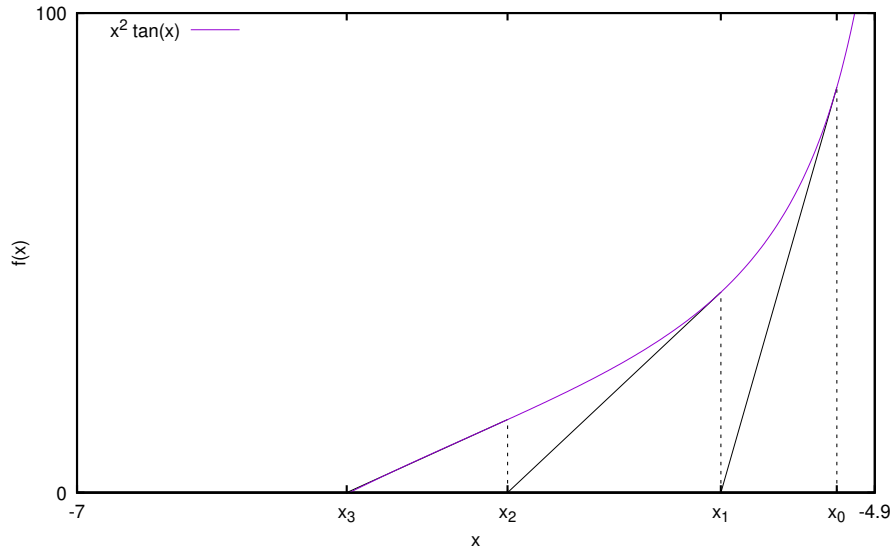

Figure 2.14: Example of the Newton-Raphson method.

were applied in-house for learning purposes.

2.5.1 Newton-Raphson Method

The Newton-Raphson method is a very simple, yet very accurate, algorithm to find the roots of a function [79, 91]. Some functions are too complex for us to determine the roots analytically, so this method is usually the first to be tried out.

The Newton-Raphson is an iterative method, which means the algorithm needs to be repeated several times until some condition is satisfied. For a function $f(x)$, this method uses the definition of derivative in order to find the next iteration point:

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}} \tag{2.129}$$

Where f' is the derivative of f and the index i represents the iteration number. Solving it for x_{i+1}:

$$x_{i+1} = x_i + \frac{f(x_i)}{f'(x_i)} \tag{2.130}$$

Figure 2.14 shows how the algorithm is applied. For the first iteration, x_i will be the initial guess, x_0. By definition, $f'(x_i)$ is the slope of the line parallel to $f(x_i)$. The main purpose is to determine where said line intersects the x axis, which is x_{i+1}. After discovering x_{i+1}, we evaluate $f(x_{i+1})$ and check if it meets the stopping condition. If it does not, we proceed for the next iteration. If it does, the algorithm stops and we found our root. The stopping condition is usually a tolerance, which states that the result cannot be above a certain error, such as $f(x_{i+1}) \leq 10^{-6}$. In this example, our tolerance is 10^{-6} and any value smaller than that is considered a root of the function.

It is never enough to stress how important the initial guess is. The function used for this example, $f(x) = x^2 tan(x)$, was truncated to the interval $[-7, -4.9]$. If the interval would be increased, periodic roots would be verified, as well as several non-smooth intervals. If the initial guess would fall in one of those areas, it is very likely that no root would be found by this method.

For this particular example, $f'(x) = \frac{x^2}{cos^2(x)} + 2xtan(x)$, but for other functions which derivative cannot be analytically determined, you can use numerical methods, such as orthogonal collocation or any finite difference method.

2.5.2 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm [85, 86] is widely used for parameter estimation. For a function $f(x, \beta)$ where x is the independent variable and $\beta = [\beta_1, ..., \beta_N]$ the vector of N parameters to be fitted, the algorithm tries to minimize the difference between the calculated and the experimental data by optimizing the values of β. In this way, for the experimental pairs (x_i, y_i):

$$LSQ(\beta) = min\left(\sum_{i=1}^{M} \big(f(x_i, \beta) - y_i\big)^2\right) \tag{2.131}$$

Where M is the number of experimental data pairs.

Like the Newton-Raphson method, it is an iterative process. The point is to iteratively change β to gradually minimize LSQ. For that, at every iteration, to $beta$ is added a change vector δ so that $\beta_{j+1} = \beta_j + \delta$, where j is the iteration number. The vector δ is determined by solving the equation:

$$LSQ(\beta + \delta) = \left(\sum_{i=1}^{M} \big(f(x_i, \beta + \delta) - y_i\big)^2\right) \tag{2.132}$$

Where,

$$f(x_i, \beta + \delta) \approx f(x_i, \beta) + \frac{\partial f(x_i, \beta)}{\partial \beta}\delta \tag{2.133}$$

This is a common approximation method called linearization. To get the minimum of the LSQ function, we have to get the derivative in respect to δ and find the roots:

$$\frac{\partial}{\partial \delta}\big(LSQ(\beta + \delta)\big) = 0 \tag{2.134}$$

This equation is solved for every iteration. The process ends when some stopping criteria is achieved, such as a tolerance level, for example.

2.5.3 Genetic Algorithm

Contrary to the previous two algorithms, which are deterministic, the genetic algorithm (GA) is stochastic. This means that, instead of relying on exact calculations, which deterministic models do, they add randomness to them.

In the case of the GA, the changes in the manipulated variables are made using an evolution approach. Just like all living beings adapt to their environment by going through thousand of generations and millions of years, our manipulated variables work in the same way. By mimicking the simultaneous cruelty and beauty of mother nature, we can create a calculation protocol that will optimize our variables in order to solve our problem [92, 93].

As an example, let us assume a population of N individuals. The GA can be explained with these steps:

1. At each generation, create an offspring of $N/2$ individuals, which leaves us with $3N/2$ individuals in total.

2. Of these $3N/2$ individuals, choose the N best individuals.

3. Repeat previous two steps until stopping criteria is achieved.

These steps were the general concept of the GA, so now we will talk about how these concepts are applied in Genetic Programming (GP).

Let $X = [x_1, ..., x_M]$ be the vector, of size M, of manipulated variables. Genetically speaking, X represents the genetic code of an individual. The being said, each individual has a unique genetic code, so in reality $X_i = [x_{i,1}, ..., x_{i,M}]$ where $i = 1, ..., N$.

When two individuals mate, crossover happens. In a crossover, the genetic code of the offspring becomes a mix of the genes of both parents. In general, each gene will be either from the parent 1 or parent 2 with a probability of 50%. That means that, if individual 1 mates with individual 2, they create an offspring $N + 1$, with a crossover probability α_c, with the following genetic code:

$$x_{N+1,j} = \begin{cases} x_{1,j} & \text{if } p < \alpha_c \\ x_{2,j} & \text{if } p \geq \alpha_c \end{cases} \qquad \text{for} \quad j = 1, ..., M \tag{2.135}$$

Where p is a random value between 0 and 1. In this case, by evaluating the goodness of adaptation (i.e. the objective function), we have the option of increasing or decreasing α_c in order to favor the genes of the best adapted parent (i.e. the one who has a lower objective function value).

After crossover, each offspring gene has a probability α_m of suffering a mutation. This mutation depends on the type of the manipulated variable (real, integer or binary variable). In this case, the genetic code would become:

$$x_{N+1,j} = \begin{cases} x^m_{N+1,j} & \text{if } p < \alpha_m \\ x_{N+1,j} & \text{if } p \geq \alpha_m \end{cases} \qquad \text{for} \quad j = 1, ..., M \tag{2.136}$$

Where $x^m_{N+1,j}$ is the gene after mutation.

2.5.3.1 Binary Variable Mutation

A binary variable, known also as logical variable, is a variable that can only take two variables, 0 or 1. It is mainly use for decision-making, such as choosing equipment for a factory, choosing which projects we should invest in an investment portfolio, etc.

Since a binary variable can only take two values, we use the flip mutation technique [94]:

$$x = \begin{cases} x^m & \text{if } p < \alpha_m \\ x & \text{if } p \geq \alpha_m \end{cases} \tag{2.137}$$

If $x = 0$ then $x^m = 1$ and vice-versa.

2.5.3.2 Integer Variable Mutation

An integer variable, as the name indicates, is a variable that belongs to the set of integer numbers $\mathbb{Z}$. It is widely used when we are optimizing integer quantities, such as the number of items that fit in a certain volume, how many equipments can we buy considering the budget available, etc.

It is not easy to generate random integer numbers. However, instead of trying to mutate an integer number directly, we can use binary arithmetic [94]. According to binary arithmetic, we can express any integer number as the sum of base-2 binary numbers. For instance, let B be the binary vector representing the integer value 10. Their representation takes the form:

$$\begin{gathered} B = [1, 0, 0, 1] \\ \sum_{i=1}^{4} B_i \times 2^{i-1} = 10 \end{gathered} \tag{2.138}$$

The superscript $i - 1$ is due to the fact that binary arithmetic starts at index 0, opposite to vectors which start at index 1.

With this notation we can represent any integer number. Furthermore, since the representing vector has only binary values, the binary mutation works perfectly. Using the above example, a mutation of B could be:

$$B_i = \begin{cases} B^m_i & \text{if } p < \alpha_m \\ B_i & \text{if } p \geq \alpha_m \end{cases} \qquad \text{for} \quad i = 1, 2, 3, 4 \tag{2.139}$$

2.5.3.3 Real Variable Mutation

A real variable is a variable that belongs to the set of real numbers $\mathbb{R}$. This variable type is commonly used to optimize real quantities, such as maximizing the amount of mass produced by a chemical process, minimize the time length of a process, etc.

Since we are no longer mutating binary values, besides the mutation probability α_m, we have one more parameter to take into account, which is the degree of mutation R_d and the maximum degree of mutation R_m. Suppose we have a real variable $x_l \leq x \leq x_u$ where x_l and x_u are the lower and upper bounds, respectively. The mutation to x can never be greater than R_m, so $x - R_m \leq x^m \leq x + R_m$. We can express it in the following way:

$$x = \begin{cases} x^m & \text{if } p < \alpha_m \\ x & \text{if } p \geq \alpha_m \end{cases} \tag{2.140}$$

The mutation rule is similar as the previous ones. However, the mutation protocol is completely different. As described by [95], we can create a real-value mutation by using a geometric series. For this mutation algorithm, $R_d \geq 1$ must be an integer number and the mutation increment x_d takes the form:

$$x_{d,i} = \begin{cases} \frac{1}{2^i} & \text{if } p < \frac{1}{R_d} \\ 0 & \text{if } p \geq \frac{1}{R_d} \end{cases} \qquad \text{for} \quad i = 1, ..., R_d \tag{2.141}$$

Finally we can obtain x^m:

$$x^m = \begin{cases} x + R_m \sum_{i=1}^{R_d} x_{d,i} & \text{if } p < 0.5 \\ x - R_m \sum_{i=1}^{R_d} x_{d,i} & \text{if } p \geq 0.5 \end{cases} \tag{2.142}$$

This works very well because the sum of the geometric series $\frac{1}{2^n}$ is always smaller than 1, no matter how big R_d is. Furthermore, the bigger R_d, the smaller the mutation size, due to the lower probability of mutating x_d.

2.6 Concluding Remarks

This Chapter covered all the mathematical foundations that lay base to several chemical processes, including chromatography. From simple mathematical models to complex discretization methods, process simulations and real-life optimization problems, as well as numerical algorithms when the analytical mathematical tools available are not enough, we are now ready to move on to the next chapter, which will cover several optimization approaches of the GSSR, as well as some new proposed configurations for the final central-cut process.

3

Optimization of the GSSR Process for Central-cut Separation

After introducing the theoretical fundamentals, as well as the mathematical tools necessary for this chapter, we will focus now on the simulation and optimization of the GSSR process. We used two main optimizations, 1) an in-house customized version of the *Evolutionary.jl* package [94] written in *Julia* , using genetic programming, and 2) the *IpOpt* software [96], an implementation of an interior-point method [97], mainly used for nonlinear optimization but can also be used for linear optimization.

3.1 First Application of the GSSR

In Section 1.4.8 we talked about the first publication of the GSSR process. We mentioned the logic behind it, as well as the reason to create a competitor of the MCSGP, a process widely known and used. In [47], they tested a 5-component peptide mixture. The target product is component 3, the closest impurities are components 2 and 4, component 1 is the weakest adsorbing and component 5 is the strongest adsorbing impurities.

3.1.1 Mathematical Model

After studying the adsorption behavior of the components, they verified that the components, under the concentration range studied, followed a linear isotherm model:

$$q_i = (\epsilon_p + H_i)c_i \qquad \text{for} \quad i = 0, ..., NC \tag{3.1}$$

Where i is the solute index (0 is the solvent), NC the number of components in solution, q is the adsorbed concentration per unit volume of resin, c the bulk concentration, ϵ_p the intraparticular porosity and H is the Henry constant.

In this case, the solvent was comprised by a mixture of water and ethanol (EtOH) at various volume fractions. Since it is known that the adsorption behavior of biological compounds, such as peptides, are strongly affected by the amount of organic solvent present in solution, we need to model the dependency of the Henry constant on c_0:

$$H_i = \frac{H_i^r}{c_0^{n_i}} \tag{3.2}$$

For $i = 0$, we establish that both H_i^r and n_0 are null, since EtOH does not adsorb. In this work, the equilibrium-dispersive model was used:

$$\epsilon_b \frac{\partial c_i}{\partial \theta} + (1 - \epsilon_b)\frac{\partial q_i}{\partial \theta} = \tau \frac{Q}{V}\left(\frac{1}{Pe_i}\frac{\partial^2 c_i}{\partial x^2} - \frac{\partial c_i}{\partial x}\right) \qquad \text{for} \quad 0 < x < 1 \tag{3.3}$$

The general model can be expanded in order to be more specific to this problem:

$$\left(\epsilon + \frac{V_x}{V_c}\right)\frac{\partial c_i}{\partial \theta} + (1 - \epsilon_b)H_i\left(\frac{\partial c_i}{\partial \theta} - \frac{n_i c_i}{c_0}\frac{\partial c_0}{\partial \theta}\right) = \tau \frac{Q}{V}\left(\frac{1}{Pe_i}\frac{\partial^2 c_i}{\partial x^2} - \frac{\partial c_i}{\partial x}\right) \qquad \text{for} \quad 0 < x < 1 \tag{3.4}$$

With boundary conditions,

$$c_i - \frac{1}{Pe_i}\frac{\partial c_i}{\partial x} = c_i^{in} \qquad \text{for} \quad x = 0 \tag{3.5}$$

$$\frac{\partial c_i}{\partial x} = 0 \qquad \text{for} \quad x = 1 \tag{3.6}$$

As discussed in Section 2.1.5, the Péclet number can be expressed as a function of the flowrate:

$$\frac{1}{Pe_i} = \frac{1}{Pe_h} + \alpha_i \frac{Q}{V_c} + \beta_i \frac{V_c}{Q} \tag{3.7}$$

With

$$\begin{aligned} \alpha_i &= \frac{\epsilon - \epsilon_b + (1 - \epsilon_b)H_i}{k_i(\epsilon + (1 - \epsilon_b)H_i)^2} \\ \beta_i &= \frac{\epsilon_b D_{m,i}}{\alpha_b L^2} \end{aligned} \tag{3.8}$$

Where ϵ is the total bed porosity, ϵ_b the external bed porosity, k_i the mass transfer coefficient, D_m the molecular diffusion coefficient, $\alpha_b \approx 3$ the tortuosity factor and L the column length.

Considering the range of flowrates used, the values of β are negligible and the second term on the right side of Equation 3.1.1 disappears.

The GSSR, as well as all other multi-column continuous chromatography processes, is cyclic, which is the property that actually makes it a continuous process. In the first cycle, the three columns are completely free from solutes, and as we run more cycles, they gradually start to accumulate solutes inside them. After a certain number of cycles, we

will start to verify that the concentration profile in the end of each cycle remains constant. In that moment we say that the system reached its steady state, or for cyclic processes, the cyclic steady state (CSS). The initial cycles before the CSS is reached are said to be in dynamic state. For the mathematical model, we can simulate the GSSR process directly in CSS conditions by implementing the following relation:

$$c_{ij}(3\tau_I, x) = c_{ij}(0, x) \qquad \text{for} \quad i = 0, ..., 5\ , \quad j = 1, ..., 3 \tag{3.9}$$

With the indices i and j representing the solutes and the columns, respectively. As we already know, $3\tau_I$ is the total length of a GSSR cycle. This relation says that the concentration inside the column in the end of the cycle is equal to the concentration inside the column in the beginning of the cycle. With this equality we enforce the CSS condition in the model.

The mass balances in the inlets and outlets of the columns is described in Section 2.2.1. Different from the other inlet streams, the gradient A stream is linearly time dependent, which means that it changes linearly over a certain period of time. In the GSSR the gradient slope is reset every switching interval, meaning that:

$$c_A(t) = c_A(0) + \frac{c_A(\tau_I) - c_A(0)}{\tau_I}(t - \tau_I \text{int}(t/\tau_I)) \qquad \text{for} \quad 0 \leq t \leq 3\tau_I \tag{3.10}$$

Where c_A is the EtOH concentration in the A stream. The term $\text{int}(t/\tau_I)$ represents the integer part of the fraction. This term varies between the number of columns. For instance, if $t \geq 2\tau_I$, then $\text{int}(t/\tau_I) = 2$ which means we are in the third switching interval.

The model discretization, both in time and space, was made using the OC method. These presented equations are the basis of the mathematical model used for simulation and optimization. This originates a big system of nonlinear equations that can be solved by a suitable numerical solver.

3.1.2 Design of the Process

The first thing to take into consideration is that we are dealing with peptides. These biological compounds have complex physicochemical properties, such as hydrophobic and hydrophilic sections, which makes their separation challenging [98]. Nevertheless, reversed phase chromatography has very high resolution and efficiency, even with these types of compounds, which is why it was chosen for this work. The results of the adsorption equilibria of the multi-component mixture can be seen in Table 3.1.

The target component is component 3. It is clear that the separation is very challenging, since the fitting parameters of all components, specially components 2, 3 and 4, are very similar. This implies that the operating parameters must be chosen very carefully and with high precision, otherwise the purified product will be outside specification.

The physical parameters of the experimental set-up are presented in Table 3.2. Using the system parameters and the retention time information of all 5 components, they

Table 3.1: Adsorption equilibria for the multi-component mixture. The target component to be purified is component 3.

Component	H^r	n
1	8×10^{-7}	8.36
2	0.016	3.83
3	0.016	4.02
4	0.018	4.08
5	0.017	4.47

Table 3.2: System and column parameters of the GSSR set-up.

Parameter	Value	Units
V_x	4.5	ml
L_c	10.0	cm
D_c	1.0	cm
ϵ_t	0.6	-
Pe	300.0	-

Table 3.3: Summary of the operating parameters of a GSSR cycle.

Stream	% EtOH (v/v)	ml/min	time	min
F	25	2.0	τ_I	3.5
A	0.324-0.354	5.7	τ_D	1.5
B	0.25	1.0	τ_F	0.5
C	0.25	1.25	τ_P	1.0
D	0.50	5.0		

implemented a design procedure to determine the operation parameters of the GSSR, which can be found in Table 3.3.

The major evaluation factors of general continuous chromatographic processes are the productivity, the purity and the recovery. To determine the purity we have to evaluate the amount of each component that exits through the product stream per cycle:

$$Pur = \frac{c_{P,3}}{\sum_{i=1}^{NS} c_{P,i}} \tag{3.11}$$

Where index $c_{P,i}$ is the concentration of component i in the product stream and NS the number of components in solution. In this work, the minimum purity specification was established as $Pur_{min} \geq 0.98$.

The recovery compares the amount of product collected in the product stream with the amount of product injected in the feed stream per cycle:

$$Rec = \frac{c_{P,3} Q_P \tau_P}{c_{F,3} Q_F \tau_F} \tag{3.12}$$

Table 3.4: Results from the GSSR simulations.

Parameter	Value
Purity	0.982
Recovery	0.975
Productivity	0.095

Where index F represents the feed stream. The minimum recovery specification was established as $Rec_{min} \geq 0.95$.

The productivity compares the weight the time length of the feed step has in an entire cycle:

$$Prod = Q_F \frac{\tau_F}{3\tau_I} \tag{3.13}$$

The results of the experimental runs in the pilot unit are shown in Table 3.4. The system design has proven to be a success, since both purity and recovery are within specification.

Even though the results were successful, there is still much room for improvement. With the rapid development in technology, both in processing power and software efficiency/speed, it is now easier than ever to run computationally expensive optimization problems in record times. These advantages of the modern world technology will be used to our advantage in the next sections.

3.2 Optimization using Genetic Programming

The way the GSSR process was designed in [47] did not violate the minimum purity and recovery, which made it a success. However, this work is aimed to improve the process design with the aid of more powerful computers and more efficient software and algorithms. In this case, we decided to try to use a Genetic Algorithm (GA).

3.2.1 Algorithm Implementation in *Julia*

The GA used [99] started as a fork of the *Evolutionary.jl* package built for *Julia* [94]. However, at the time we started to use it it was still in its early ages and some functionalities were missing. Moreover, the interface was not that user-friendly and the algorithm still did not support multi-threading. With that in mind, since the package was open-source, we decided to fork the project on *GitHub* and built on top of it everything we needed.

As described in Section 2.5.3, there are three types of variables, which we also call genes: binary, integer and real. The first thing we implemented was a more user-friendly way to create the variables:

```
gene1 = BinaryGene()
gene2 = IntegerGene(length)
gene3 = FloatGene(value, range; m = Rd)
gene2_value = bin(gene2)
```

Let us break them all down:

- `BinaryGene()` : creates a binary variable with random value 1 or 0.
- `IntegerGene(length)` : creates a vector of binary values of length `length` with random values 1 or 0. For instance, if `length = 3`, then the vector would be `[1,0,0]`. Again, the vector values are randomly generated by *Julia* .
- `FloatGene(value, range; m = Rd)` : creates a real variable of initial value `value` and a maximum mutation range R_m of `range` (the word "Float" comes from floating-point number, which in programming represents a real number). `m` represents the mutation degree R_d and is an optional argument of default value 20.
- `bin(gene2)` converts the binary vector in the corresponding integer value by binary arithmetic.

In any optimization problem, there is usually more than one manipulated variable. The combination of those manipulated variables can be interpreted as a chromosome, which is a combination of genes that represent our genetic code. Simply put, a chromosome is a vector of different genes that represent one individual of a population:

```
chromosome = [gene1, gene2, gene3]
```

The next thing to do is to choose a crossover and selection types:

```
Crossover(:SPX)
Selection(:RWS)
```

The symbol `:SPX` is a shorthand for single-point crossover, the rule explained in Section 2.5.3, while `:RWS` is shorthand for roulette wheel selection. The latter is simply to make mating selection random instead of sequential (individual 1 mates with 2, 3 with 4, etc.).

We also need an objective function, which will be the evaluation criteria for goodness of adaptation of an individual. It is required for this function to be a minimization function. If you want to maximize a certain quantity, let us say a profit P, you can minimize the negative of your quantity:

$$max(P) = min(-P) \tag{3.14}$$

The objective function must be a *Julia* function and has to accept as only argument a chromosome:

```
function objfun(chromosome)
    # function body
    return expression_to_be_minimized
end
```

Finally, for us to run our genetic algorithm we just need to set the population size and we are done:

```
chromosome = [gene1, gene2, gene3]
N = 100
results = ga(objfun, chromosome, N)
```

The function `ga` accepts as arguments the objective function `objfun`, the chromosome `chromosome` and the population size `N`. This function accepts several optional arguments which will not be covered here but can be consulted in [99].

Let us run a quick example. Imagine we have the problem:

$$(x-1)^2 = 0 \tag{3.15}$$

We can see right away that x is a real variable, so we can start building the GA problem. First let us create our variable:

```
x = FloatGene(2.0, 0.5)
```

In this case we give an initial guess of 2 with a maximum mutation range of 0.5. Now we need to choose the crossover and selection algorithms, which we will maintain:

```
Crossover(:SPX)
Selection(:RWS)
```

In this variable we have only one variable. Nevertheless, we need to create a single-gene chromosome, since the `ga` function only accepts chromosomes:

```
chromosome = [x]
```

We still need the objective function. In this case, we want to find the root of $(x-1)^2$. Since this is a quadratic function it does not have negative numbers, so the objective function is the function itself:

```
function objfun(chromosome)
    x = chromosome[1].value
    return (x - 1)^2
end
```

Finally we can run the algorithm:

```
N = 100
results = ga(objfun, chromosome, N)
```

We randomly chose 100 as the population size but it can be any integer number greater than 1 (we need a minimum of 2 individuals for the mating process). Increasing the

population size decreases the number of iterations to reach the optimum value, with the trade-off that each iteration takes a longer time.

`results` is a *Julia* structure that contains all the important information after the GA finishes. It has several fields, but the two most important are `bestFit`, which contains the objective function value of the best individual, and `bestInd`, which contains the winner chromosome. They can be accessed in the following way[1]:

```
results.bestFit      # best objective function value
results.bestInd      # best chromosome
```

The package created has many more functionalities. Each gene type has several optional arguments that can be tweaked to suit some specific needs. It is also capable of running in a multi-threaded, as well as in a distributed environment. Moreover, it is also capable of running in a cluster environment (several computers connected to each other running the same program). All of those functionalities were developed in-house for both convenience and learning purposes. However, the greatest functionality was the ability to run the algorithm with external softwares that could efficiently solve the large system of equations and obtain the objective function value, which sends it back to *Julia* for the GA to evaluate it and send back to the external software the genes of the new iteration. With this approach we can take advantage of the best of both worlds. This will be explained a bit in more detail further on.

3.2.2 Optimization Procedure

The GA developed evaluates the objective function, selects the best individuals and discards the rest to proceed to the next iteration. In this case, however, the objective function is not evaluated inside *Julia* because the tools available, such as *JuMP.jl* [100], are still in their early ages and cannot efficiently deal with very large problems such as the GSSR, and that is why we decided to use *AMPL* . Our research group uses this software for several years and we developed a good expertise in that programming language.

AMPL [83] stands for Algebraic Mathematical Programming Language. The syntax itself is very intuitive and it aims to be as similar as possible to mathematical notation, which makes it very easy to describe optimization problems and solve them. The solving step is actually made using softwares external to *AMPL* , with one of them being *IpOpt* [96], which stands for Interior-Point Optimization. *IpOpt* itself uses external linear solvers, such as the *WSMP* [101], which is shorthand for Watson Sparse Matrix Package.

That being said, the entire GSSR model was implemented in *AMPL* . Both $\theta \in [0, 1]$ and $x \in [0, 1]$ were fully discretized using the OC method. The solver chosen was *IpOpt* and the linear solver used was the *WSMP* .

The key is in the communication between *Julia* and *AMPL* . A rudimentary, yet easy approach is for each language to read from and write to the same file. The file works as an

[1] the text after the "#" sign in the code snippet represents a comment.

independent middleman and both programs have very good ways to access text files. The actual read/write of the file must be made inside the objective function, which is where both programs connect. An example:

```
function objfun(chromosome)
    var = chromosome[1].value
    open("commfile.txt", "w") do f
        write(f, var)
    end
    # wait for AMPL to calculate objective function
    objfun_value = read("commfile.txt")
    return objfun_value
end
```

By using "commfile.txt" as the intermediary, both programs can easily communicate. However, there is no way for *Julia* to know how long should it wait for *AMPL* to return the objective function. There is no problem if *Julia* waits longer than *AMPL* needs, but if *Julia* tries to read "commfile.txt" before *AMPL* writes the objective function value, the results from then on will be corrupted. Furthermore, this waiting time has to be provided by the user, and if the user does not know how much time it takes for *AMPL* to calculate he might provide a waiting time big enough, which makes the entire algorithm inefficient due to unnecessary times.

The strategy we came up with was to use pipes. In programming, pipes are exclusive communication channels between two programs. It is more efficient than temporary files because it uses virtual memory (RAM) instead of physical memory to store the information and since they are exclusive, the waiting steps are automatically handled by it. Even though their properties are completely different, when it comes to reading and writing they are treated just like normal files. Pipes are unidirectional, which means there can be only one information flow. That being said, we need to create two pipes, one with the *Julia* $\rightarrow$*AMPL* flow and another with the *AMPL* $\rightarrow$*Julia* flow

```
function objfun(chromosome)
    var = chromosome[1].value
    open("pipe_julia-ampl", "w") do f
        write(f, var)
    end
    objfun_value = read("pipe_ampl-julia")
    return objfun_value
end
```

In this way the user-provided waiting step is eliminated because `read("pipe_ampl-julia")` can only be executed after *AMPL* finishes calculation and writes the result to that pipe. This improves efficiency and makes it easier to use due to the fact that there is no need to "guess" the time spent for the objective function calculation.

Table 3.5: Nomenclature of all optimization approaches.

Shorthand	Optimization Type	Optimization Variables
OG-PT	original process design	-
GA-PT	GA optimization	τ_I , τ_D , τ_P , τ_F
GA-PT+I	GA optimization	τ_I , τ_D , τ_P , τ_F , $c_A(\tau_I)$ where $c_A(\tau_I) = c_A(0)$
GA-PT+G	GA optimization	τ_I , τ_D , τ_P , τ_F , $c_A(\tau_I)$, $c_A(0)$

Table 3.6: Results from the GA optimization GA-PT.

Parameter	Value
Purity	0.981
Recovery	0.999
Productivity	0.212

Table 3.7: Results from the GA optimization GA-PT+I.

Parameter	Value
Purity	0.980
Recovery	0.991
Productivity	0.344

3.2.3 Results

After developing the GA in *Julia* and the GSSR mathematical model in *AMPL* , we start to run our optimizations. We performed three different approaches, both with different manipulated variables but all with the same purpose, which was to maximize the productivity. The three approaches plus the original one are summarized in Table 3.5. For convenience, this nomenclature will be used throughout the document.

When running the GA algorithm, our first approach was obviously the simplest one, which was the optimization of the process times, GA-PT. After the optimizations, τ_I = 3.54min, τ_D = 1.13min, τ_F = 1.13min and τ_P = 1.13min. These variables provided us with a major increase in productivity, as shown in Table 3.6.

The second approach was a way to find if we could replace the gradient elution to isocratic and still have better results, since it could simplify the GSSR process in practical terms because an isocratic elution is much simpler and cheaper to implement than a gradient elution. In the end, we obtained the values 4.84min, 2.5min, 2.5min, 2.5min and 0.289v/v for τ_I , τ_D , τ_F , τ_P and $c_A(\tau_I)$, respectively. The end results are presented in Table 3.7.

The third approach was to verify if gradient elution was still worth it, despite the fact isocratic is always favorable in practice. After optimization, we obtained for τ_I , τ_D , τ_F , τ_P , $c_A(\tau_I)$ and $c_A(0)$ the values of 5.26min, 2.95min, 2.95min, 2.95min, 0.289v/v and 0.264v/v, respectively. Again, the end results are shown in Table 3.8.

Table 3.8: Results from the GA optimization GA-PT+G.

Parameter	Value
Purity	0.980
Recovery	0.994
Productivity	0.374

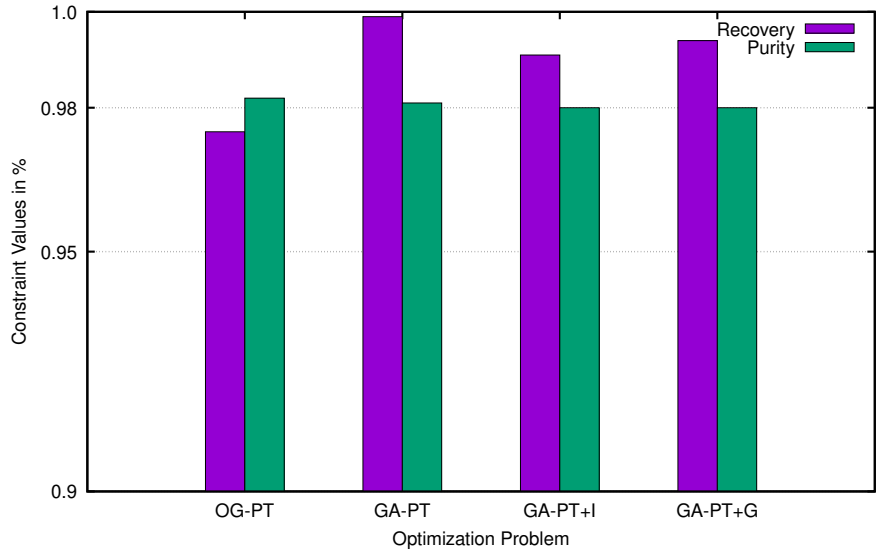

Figure 3.1: Purity and Recovery values for each optimization strategy.

Already at first glance, every approach implemented had improvements in comparison with the previous one. However, the major result was the fact that, for all three approaches, $\tau_F = \tau_D = \tau_F$. This is a major breakthrough because this means that, in a GSSR cycle, the time lengths of steps 6 and 7 are null, which means we can eliminate them. This reduces the complexity of the process, while at the same time increases productivity.

Figure 4.26 shows the evolution of the purity and recovery in each optimization. The purity remained rather constant, while the recovery improved greatly compared to the original design parameters.

Figure 3.2 shows the productivity value of each optimization approach. In here we can see the great improvements in productivity with every approach implemented. All GA optimizations offer better results.

Table 3.9 presents a summary of the manipulated variables at every optimization.

In Figure 3.3 is presented a graphical representation of the axial profiles of each column at the end of each step. The concentrations were normalized with the feed concentration. This is useful to see the concentration profiles of the impurities, since their concentration in the feed is already very low. This is another representation of the product coming out of column 3 at step 5 within purity and recovery specifications. A, B, C and D represent OG-PT, GA-PT, GA-PT+I and GA-PT+G, respectively. The axial profiles of B, C and D from steps 5, 6 and 7 are equivalent because $\tau_6 = \tau_7 = 0$.

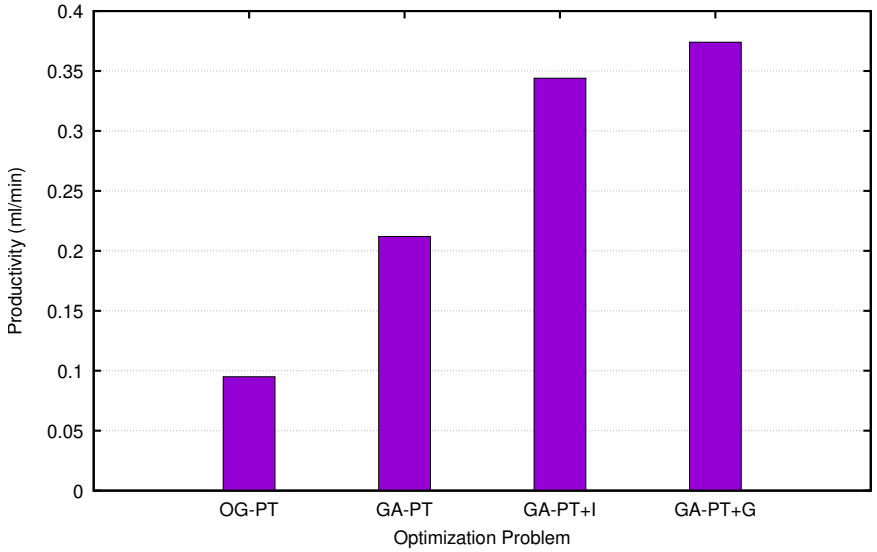

Figure 3.2: Productivity values for each optimization strategy.

Table 3.9: Summary of all the manipulated variables.

Variables	OG-PT	GA-PT	GA-PT+I	GA-PT+G
τ_I (min)	3.500	3.540	4.840	5.260
τ_D (min)	1.500	1.130	2.500	2.950
τ_P (min)	1.000	1.130	2.500	2.950
τ_F (min)	0.500	1.130	2.500	2.950
$c_A(0)$ (v/v)	0.324*	0.324*	0.289	0.289
$c_A(\tau_I)$ (v/v)	0.354*	0.354*	0.289**	0.264
Productivity	0.095	0.212	0.344	0.374

* fixed in that optimization step
** $c_A(\tau_I) = c_A(0)$

3.2.3.1 Evaluation of Key Performance Indicators

The increase in productivity, while obeying product specifications, is a major indicator that the process has improved. However, in order to evaluate if these optimizations are robust and actually improve the overall process we compared two more key performance indicators (KPIs), the dilution ratio DR and the solvent consumption ratio SCR.

The dilution ratio compares the concentration of the target product in the product stream with the concentration of the same in the feed tank. This is important because we might be having good productivity, recovery and purity, but in the end our product is highly diluted, which is undesirable. The expression is as follows,

$$DR = \frac{c_{3,P}}{c_{3,F}} \tag{3.16}$$

Where $c_{3,P}$ is the concentration of component 3 in the product stream and $c_{3,F}$ is the concentration in the feed stream. The higher the value of DR, the higher the concentration

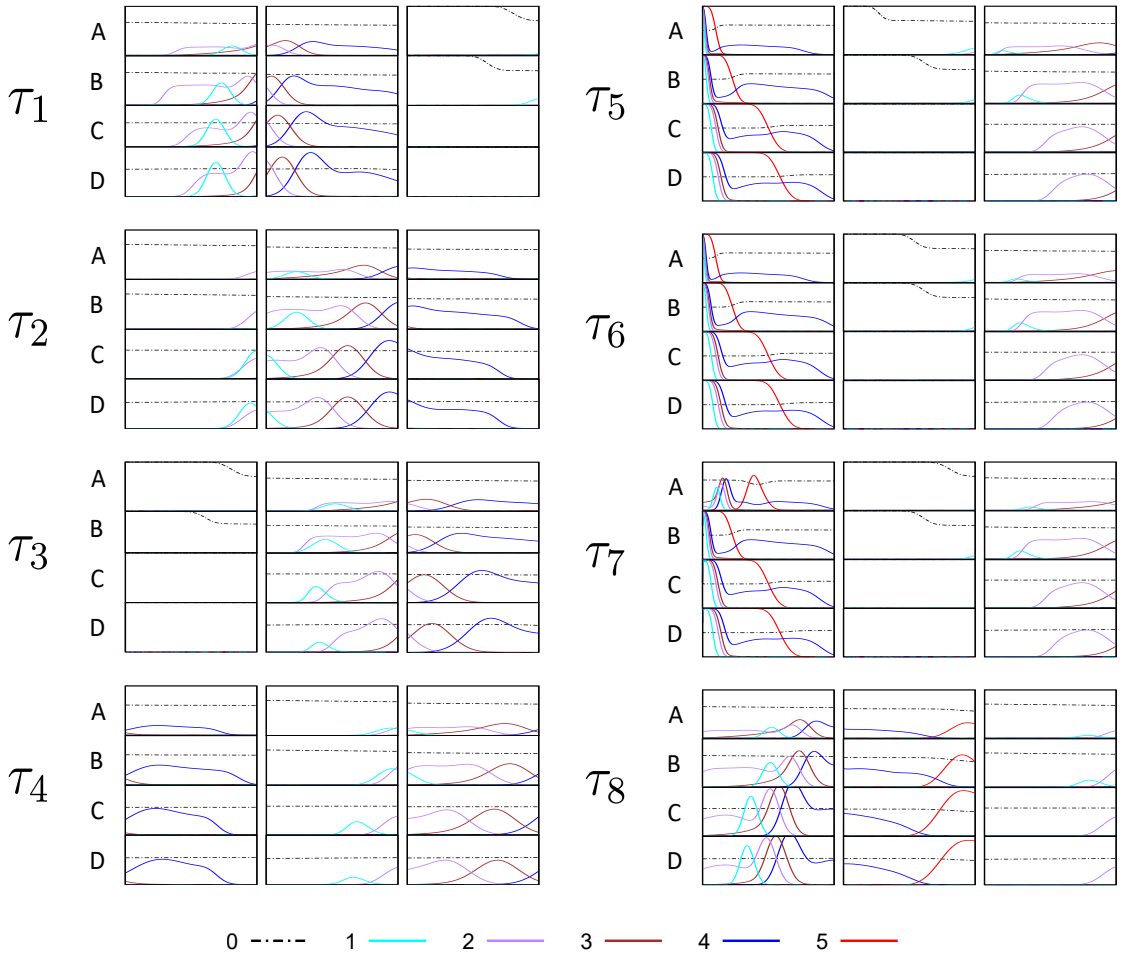

Figure 3.3: Axial concentration profiles of each column in each step for each optimization type. The letters A, B, C and D represent the optimization types OG-PT, GA-PT, GA-PT+I and GA-PT+G, respectively.

in the product stream.

The solvent consumption ratio compares the volume of solvent needed to produce one volume unit of product in each cycle. In here we are evaluating how much solvent we have to use to produce our target product, so the smaller the better. We can evaluate it in the following way:

$$SCR_s = \frac{V_s^T}{V_F^T} \qquad \text{for} \quad s = A, B, C, D \tag{3.17}$$

V_s^T is the total volume injected inside the system per cycle of stream s and V_F^T is the total volume of feed processed per cycle.

Figures 3.4 and 3.5 show the values of both KPIs for each optimization method.

Regarding the dilution ratio, the value almost doubled when comparing the original parameters and the GA parameters. This means that we were able to concentrate the target product in the product stream almost 2-fold, which is a major improvement. When comparing the three GA optimizations, the value remains practically constant. This might

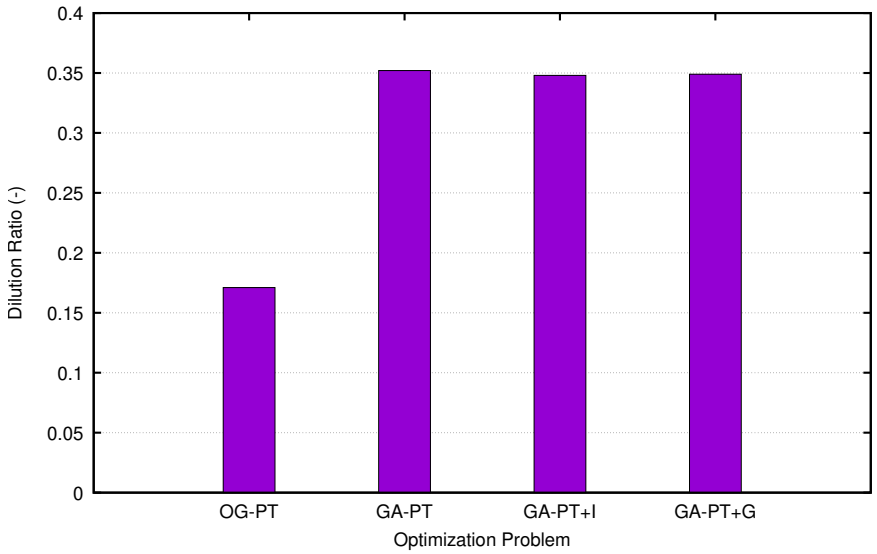

Figure 3.4: Dilution ratio values for each optimization strategy.

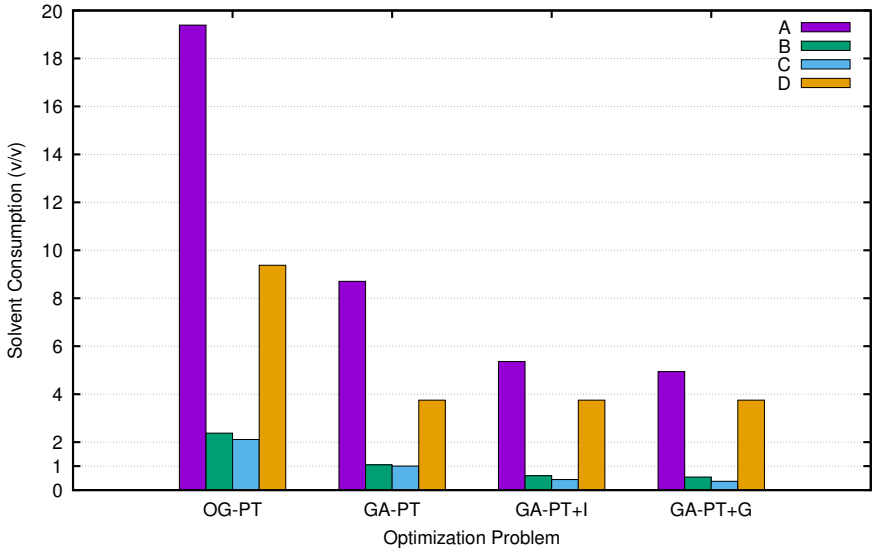

Figure 3.5: Solvent Consumption values for each optimization strategy.

be due to the fact that all three reach the equality $\tau_D = \tau_F = \tau_P$ and the feed volume is unchanged, so the dilution ratio cannot improve more than that.

When it comes to the solvent consumption ratio, we notice major improvements in all solvent streams when comparing with the original optimization. In the A stream there is a reduction of around 2-fold from the original optimization to the GA-PT optimization. In the GA-PT+G the A stream has a value of ~5, which is a significant decrease comparing with ~19.5 from the OG-PT. The D stream decreased more than 2-fold from the OG-PT to the GA-PT but maintained constant in all GA optimizations. Solvent streams B and D had an almost 4-fold decrease comparing the OG-PT and the GA-PT+G.

The GA optimizations managed to improve the original GSSR process design in all aspects. However, since the GA is a stochastic algorithm, it relies on probabilities to

improve itself and achieve better results. This means that the algorithm takes much longer since it is simply evaluating the objective function and randomly changing the manipulated variables and hope for the best in the next iteration. It is not said, but each optimization strategy took days, some of them even weeks to achieve such results and there is no way of knowing if those results are in fact local optima of the problem, since no derivatives are calculated in this algorithm. However, the next section will further improve on top of that.

3.3 Optimization using the Interior-point Solver *IpOpt*

Before using the GA, we attempted to use the *AMPL*/*IpOpt* combination, but we were having several issues of convergence, in which the solver could not reach an optimum. We thought that the problem was the isotherm models, since the derivatives of power-law models are unstable when they are close to 0. For that reason we switched to the GA because no derivatives were calculated. However, we never forgot this approach, because if we could get it to work the benefits would be immediate, both in terms of goodness of results and time spent to obtain them. Also, as said in the previous section, despite the good results that came from the GA optimizations, such stochastic algorithms take a much longer time to obtain good results. We are talking about days, even weeks, to get satisfying improvements. Furthermore, the optimization time exponentially increases with the increase of decision variables. The next step was to also use the flowrates as decision variables and we decided that the time spent using the GA was too great for the problem to be feasible. In that way we decided to come back to deterministic solvers, such as *IpOpt* [96], which we already know and have a good degree of experience using it, specially when combined with *AMPL* .

This section is a summary of the paper that we already published regarding deterministic optimization of the GSSR process [102].

3.3.1 Problem Formulation

The GSSR, as stated before, follows the same premise as the MCSGP, with the difference that the feed and product streams are fixed in columns 1 and 3, respectively. Fixing the feed and product streams in the extremes of the process forces the components to travel the longest distance possible, making it easier for the separation to occur.

Before the process description, it is convenient to perform some variable changes. The three time variables previously described, τ_D, τ_F and τ_P, can be described as a function of τ_I in the following way,

$$\begin{aligned} \tau_D &= \alpha_D \tau_I \qquad \text{for} \quad 0 \leq \alpha_D \leq 1 \\ \tau_F &= \alpha_F \tau_I \qquad \text{for} \quad 0 \leq \alpha_F \leq 1 \\ \tau_P &= \alpha_P \tau_I \qquad \text{for} \quad 0 \leq \alpha_P \leq 1 \\ \alpha_P &\geq \alpha_F \end{aligned} \tag{3.18}$$

This will help us formulate the problem in terms of injected volume instead of flowrates, which will be explained later on. The equilibrium-dispersive model is used for this strategy:

$$\left(\epsilon + \frac{V_x}{V_c}\right)\frac{\partial c_i}{\partial \theta} + (1-\epsilon_b)H_i\left(\frac{\partial c_i}{\partial \theta} - \frac{n_i c_i}{c_0}\frac{\partial c_0}{\partial \theta}\right) = \tau\frac{Q}{V}\left(\frac{1}{Pe_i}\frac{\partial^2 c_i}{\partial x^2} - \frac{\partial c_i}{\partial x}\right) \qquad \text{for} \quad 0 < x < 1 \tag{3.19}$$

With boundary conditions,

$$c_i - \frac{1}{Pe_i}\frac{\partial c_i}{\partial x} = c_i^{in} \qquad \text{for} \quad x = 0 \tag{3.20}$$

$$\frac{\partial c_i}{\partial x} = 0 \qquad \text{for} \quad x = 1 \tag{3.21}$$

The equilibrium-dispersive model assumes that the adsorption kinetics is infinitely fast. On top of all the assumptions explained before that come with the model, one other advantage is the fact that, instead of describing the model in time and space, we can change the time domain to volume domain. In practical terms, we can add a variable $w = \tau Q/V_c$ that represents the injected volume per column volume at step τ. If we make the variable change in Equation 3.3.1 we get:

$$\left(\epsilon + \frac{V_x}{V_c}\right)\frac{\partial c_i}{\partial \theta} + (1-\epsilon_b)H_i\left(\frac{\partial c_i}{\partial \theta} - \frac{n_i c_i}{c_0}\frac{\partial c_0}{\partial \theta}\right) = w\left(\frac{1}{Pe_i}\frac{\partial^2 c_i}{\partial x^2} - \frac{\partial c_i}{\partial x}\right) \qquad \text{for} \quad 0 < x < 1 \tag{3.22}$$

We can combine the Péclet constant expression:

$$\left(\epsilon + \frac{V_x}{V_c}\right)\frac{\partial c_i}{\partial \theta} + (1-\epsilon_b)H_i\left(\frac{\partial c_i}{\partial \theta} - \frac{n_i c_i}{c_0}\frac{\partial c_0}{\partial \theta}\right) = w\left[\left(\frac{1}{Pe_h} + \frac{\alpha_i w}{\tau}\right)\frac{\partial^2 c_i}{\partial x^2} - \frac{\partial c_i}{\partial x}\right] + \tau\beta_i\frac{\partial^2 c_i}{\partial x^2} \qquad \text{for} \quad 0 < x < 1 \tag{3.23}$$

The same can be done to the boundary conditions:

$$w\left[c_i - \left(\frac{1}{Pe_h} + \frac{\alpha_i w}{\tau}\right)\frac{\partial c_i}{\partial x}\right] - \tau\beta_i \frac{\partial c_i}{\partial x} = (wc_i)^{in} \qquad \text{for} \quad x = 0 \tag{3.24}$$

$$\frac{\partial c_i}{\partial x} = 0 \qquad \text{for} \quad x = 1 \tag{3.25}$$

With this formulation the diffusion term $\tau\beta_i \partial c_i/\partial x$ cannot be neglected. The reason is purely mathematical. There are steps in the GSSR where the flowrate through column 1 is stopped, which means that $w = 0$. In that way, if we neglect the diffusion term, the boundary condition where $x = 0$ becomes $0 = 0$, an indetermination, which can cause some problems to the solver. In this way, we simply need to let $\beta \neq 0$ and give it a very small value to prevent any numerical issues.

The GSSR system has four elution streams, A for gradient elution, B and C for isocratic elution with different solvent concentrations and D with the highest solvent concentration for cleaning in-place.

The four solvent streams can be described in function of the volume amount injected:

$$\begin{aligned} w_A &= \tau_I \frac{Q_A}{V_c} \\ w_B &= \tau_I \frac{Q_B}{V_c} \\ w_c &= (1 - \alpha_D)\tau_I \frac{Q_C}{V_c} \\ w_D &= \alpha_D \tau_I \frac{Q_D}{V_c} \end{aligned} \tag{3.26}$$

Regarding the feed stream, it is more convenient not to depend on the column volume:

$$w_F = \alpha_F \tau_I Q_F \tag{3.27}$$

Writing the problem in terms of volume amount injected has a main advantage. Consider the steps 5 and 6 in Figure 1.17. The collection step is usually greater than or equal to the feeding step, that is why step 6 exists. By using volume instead of time, we can merge both steps into step $\tau_5' = \tau_5 + \tau_6$. In this case, the number of steps to be considered for optimization is reduced, as well as the number of decision variables, since we trade α_F and Q_F with w_F. It cannot be underestimated the impact that one step and one decision variable less has on how long an optimization or simulation takes, specially the decrease in decision variables.

In a semi-continuous system like the GSSR, it is convenient to maximize the average feed flowrate per cycle, $\overline{Q_F}$, that is the amount of feed volume injected per cycle length:

$$\overline{Q_F} = \frac{w_F}{3\tau_I} \tag{3.28}$$

The mass of component i that is collected per cycle from column 3 is calculated as below:

$$y_i = \sum_{k \in P} w_{3,k} \int_0^1 c_{i,3,k}^{out} d\theta \tag{3.29}$$

Where P is the set of steps where product is collected. With this quantity is easy to calculate both purity and recovery from the target component:

$$\begin{aligned} Pur &= \frac{y_p}{\sum_{i=1}^{NC} y_i} \\ Rec &= \frac{y_p}{c_p^F w_F} \end{aligned} \tag{3.30}$$

Where p is the target component index and NC is the number of components. Both quantities above will be our restrictions, which in this case

$$Pur \geq Pur_{min}\ , \quad Rec \geq Rec_{min} \tag{3.31}$$

The concentration and solvent consumption ratios are also evaluated in this approach, but with slightly different formulations. The concentration ratio, which compares the target component concentration in the collection stream with the concentration in the feed stream, is written as

$$CR = \frac{y_p}{c_p^F \sum_{k \in P} w_{3,k}} \tag{3.32}$$

Again, p and P are the target component index and the set of steps where product is collected, respectively. On the previous approaches, the solvent consumption ratio compared the amount of volume of solvent used to produce a volume unit of purified product. With the current approach, it is more convenient to compare with the amount of volume injected per cycle:

$$SCR = 3 \times \frac{w_A + w_B + w_C + w_D}{w_F} \tag{3.33}$$

3.3.2 Results

The initial estimate of the operating conditions reported by [47] are: $\tau_I = 3.5$min, $\alpha_F = 0.5/\tau_I$, $\alpha_D = 1.5/\tau_I$, $\alpha_P = 1/\tau_I$, $Q_A = 5.7$ml/min, $Q_B = 1$ml/min, $Q_C = 1.25$ml/min, $Q_D = 5$ml/min, $Q_F = 2$ml/min, $c_A(0) = 0.324$v/v and $c_A(\tau_I) = 0.354$v/v. The concentrations of the solvent streams are fixed at $c_B = 0.25$v/v, $c_C = 0.25$v/v and $c_D = 0.5$v/v. The average feed flowrate per cycle in these conditions is $\overline{Q_F} = 0.095$ml/min.

Our first approach, called approach I, just like with the GA, lets us maximize $\overline{Q_F}$ by varying the cycle times only and with requirements of purity and recovery of 0.98

for both. The result was $\tau_I = 3.56$min and $\alpha_F = \alpha_P = \alpha_D = 0.33$. In this optimization $\overline{Q_F} = 0.22$ml/min and purity and recovery were 0.98 and 0.995 respectively. The fact that the values of α are equal means that steps 6 and 7 of Figure 1.17 disappear entirely, which shows the original values reported were conservative. This is understandable, since the original values were determined by using heuristic rules instead of mathematical optimizations.

In this case, the purity requirement is met by an infinitesimal margin, however the recovery is significantly greater than the required value, which means that using the cycle times only as decision variables is not enough to achieve the best results. With this approach, called approach II, both Q_F and α_F are removed from the problem and replaced by w_F, which makes it now five decision variables. The results of this optimization were $\tau_I = 3.5$min, $\alpha_P = \alpha_D = 0.331$ and $w_F = 11.39$ml, hence $Q_F = 9.83$ml/min. The achieved throughput was $\overline{Q_F} = 1.08$ml/min, around 11 times greater than the original parameters.

In this next case, called approach III, we added as decision variables both concentration extremes of the gradient stream A, $c_A(0)$ and $c_A(\tau_I)$. The concentrations of streams B, C and D will remain fixed, which gives us the minimum and maximum values of the gradient concentration of 0.25 and 0.5, respectively. This approach gave us the operating conditions of $\tau_I = 3.53$min, $\alpha_P = \alpha_D = 0.33$, $c_A(0) = 0.293$v/v, $c_A(\tau_I) = 0.428$v/v and $w_F = 11.65$ml, hence $Q_F = 10$ml/min. Purity and recovery are satisfied in their lower limits.

In approach III the feed flowrate was stuck on the maximum value of 10ml/min, so we decided to relax such restriction to verify if further improvement is possible. Moreover, we wanted to minimize the solvent consumption. Since streams B and C are isocratic, it is theoretically possible to replace them with stream A using a different gradient slope, so flowrates Q_B and Q_C were set to 0ml/min. Also, since stream D cannot be emulated using stream A, we decided to introduce a penalty for using it in the objective function:

$$max(\overline{Q_F} - \mu w_D) \tag{3.34}$$

Here, $\mu = 10^{-3}$ is a small positive constant to account for this penalty and w_D was added to the set of decision variables. For this set, called approach IV, the optimized values were $\tau_I = 4.2$min, $\alpha_P = \alpha_D = 0.281$, $w_D = 0$ml/min, $c_A(0) = 0.25$v/v, $c_A(\tau_I) = 0.424$v/v and $w_F = 15.32$ml, hence $Q_F = 12.98$ml/min. The productivity value was $\overline{Q_F} = 1.21$ml/min. This variable set is very satisfying, since streams B, C and D were eliminated and the productivity actually increased.

Table 3.10 summarizes the results previously shown, as well as the values of the KPIs for each approach. It is worthy of note the improvements of each KPI in each approach. The solvent consumption ratio decreased around 13 times the value of the original operating point, while the concentration had an increase of around 22-fold comparing to the original reports. Furthermore, the concentration ratio of approaches II, III and IV are greater than

Table 3.10: Summary of the optimized decision variables in each approach. The first row represents the results originally reported by [47].

	τ_I	α_D	α_F	α_P	Q_A	Q_B	Q_C	w_F	w_D	$c_A(0)$	$c_A(\tau_I)$	$\overline{Q_F}$	RC	SCR
	3.50	0.429	0.143	0.286	5.7	1.0	1.25	1.00	0.955	0.324	0.354	0.095	0.110	59.90
I	3.56	0.330	-	0.330	5.7	1.0	1.25	2.35	0.749	0.324	0.354	0.220	0.349	41.20
II	3.50	0.331	-	0.331	5.7	1.0	1.25	11.39	0.736	0.324	0.354	1.080	1.690	8.38
III	3.53	0.330	-	0.330	5.7	1.0	1.25	11.65	0.741	0.293	0.428	1.100	1.720	8.25
IV	4.20	0.281	-	0.281	5.7	0.0	0.00	15.32	0.000	0.250	0.424	1.210	2.230	4.70

1, which means that the target compound is more concentrated in the product stream than in the feed, which is a substantial improvement.

It is interesting to analyze how the internal recycling works when the concentration of the target component changes. For that, we fixed $c_A(0)$ and changed c_p^F while maintaining the impurities concentration fixed. Figure 3.6 shows how the decision variables change with the increase/decrease in c_P^F. The results are to be expected: the greater the target concentration compared to the impurities, the easier it is to separate them with high degrees of purity and recovery.

Mathematical optimizations are very accurate and are able to predict experimental results, but in laboratory work there are several points where error can occur, such as the pump flowrates, the time lengths of each step, the gradient precision, etc. With that in mind, we studied the impact that small deviations of τ_I, $c_A(\tau_I)$, w_F and α_D from their optimal values have in the purity and recovery. The results can be seen in Figure 3.7. A positive percentage value represents the increase amount in regards to the optimum value, a negative percentage value represents a decrease in regards to the optimum value. In all of the variables, the recovery is way more sensitive than the purity, specially in deviations in τ_I and $c_A(\tau_I)$. About w_F and α_D, it has some impact but not as much as the previous two.

As said before, experimental error is important to take into account. The applied flowrate from HPLC pumps are pretty accurate. The ON/OFF pneumatic valves in our lab are quite quick in changing pathways, as well as our software when it applies the time lengths of each step. However, the most accurate approach in applying a modifier gradient is by using two HPLC pumps, each with a fixed concentration of 0.25v/v and 0.5v/v, respectively. In this case, mixing the streams coming from two HPLC pumps have a degree of uncertainty that we will take into account for the next optimization approach. This gradient uncertainty can be introduced in our mathematical problem in the following way:

$$c_A(\tau_I) = (1 + \sigma_A)\overline{c_A}(\tau_I) \tag{3.35}$$

Where $\overline{c_A}(\tau_I)$ is the optimal value without uncertainty and σ_A is the deviation percentage, which will be set at 2%.

The strategy for solving this problem has already been reported [103, 104] and is as

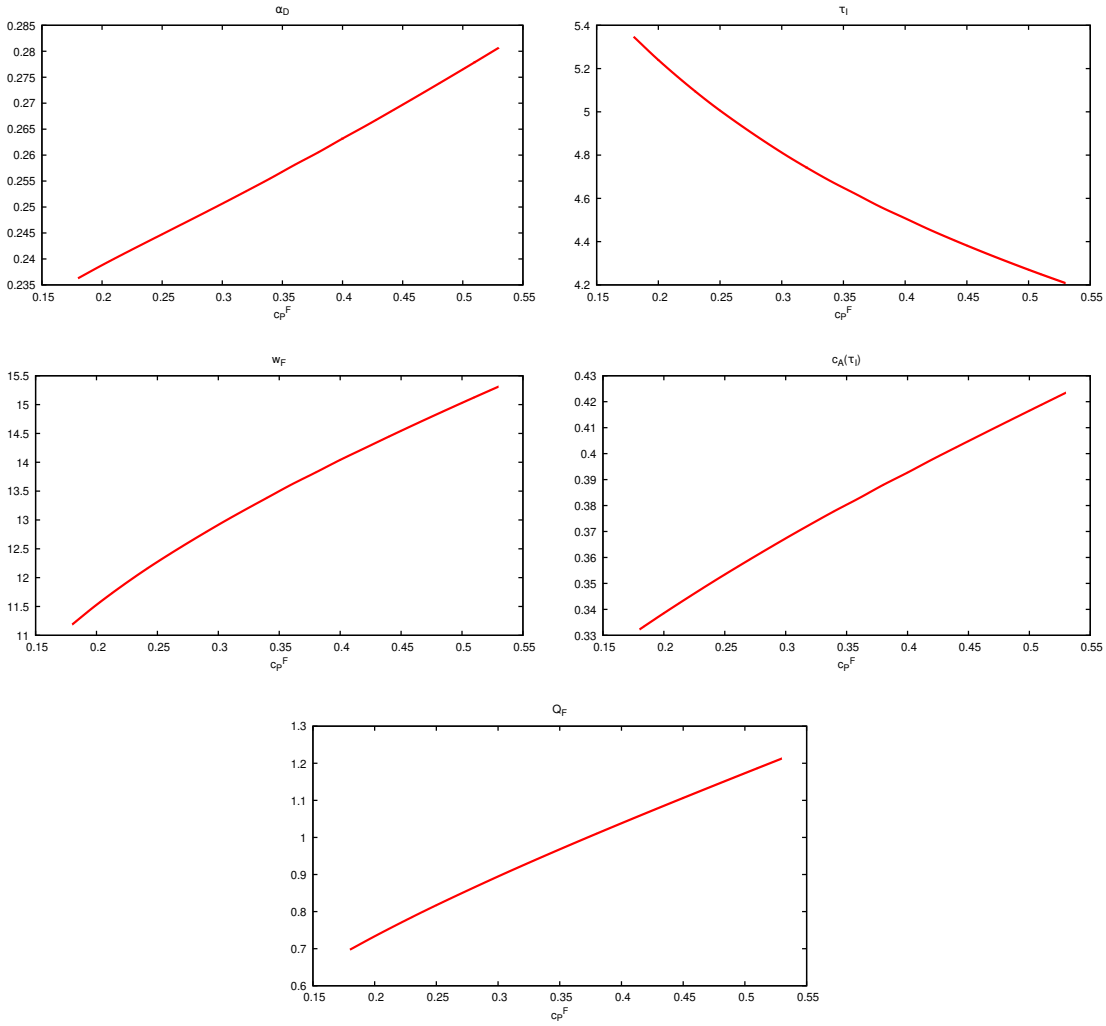

Figure 3.6: Impact of the target concentration in the feed stream. The higher the target concentration comparing to the impurities, the easier and less costly the separation is.

follows. We create two instances of the full mathematical problem, to one is assigned $\sigma_A = 0.02$ and to the other is assigned $\sigma_A = -0.02$. However, despite being two independent equation systems, they share the same decision variables, since the objective is to find one operating point that can satisfy the extremes of the gradient uncertainty. The results are $\tau_I = 5.29$min, $\alpha_P = \alpha_D = 0.267$, $w_D = 0$, $\overline{c_A}(\tau_I) = 0.343$v/v and $w_F = 8.06$ml, hence $Q_F = 5.7$ml/min. Also, $\overline{Q_F} = 0.507$ml/min, $CR = 0.981$ and $SCR = 11.2$. The instance with $\sigma_A = -0.02$ had a purity and recovery of 0.992 and 0.98 respectively, while the instance with $\sigma_A = 0.02$ had a purity and recovery of 0.98 and 0.98 respectively.

When comparing to the results of approach IV, $\overline{Q_F}$ went from 1.21 to 0.507ml/min, SCR from 4.7 to 11.2 and CR from 2.23 to 0.98. It is to be expected that the productivity and the KPIs worsen when we account for uncertainties. However, a reduction of around 58% in the productivity, an increase of around 138% in solvent consumption and a decrease of around 127% in the concentration ratio are way larger than anticipated for

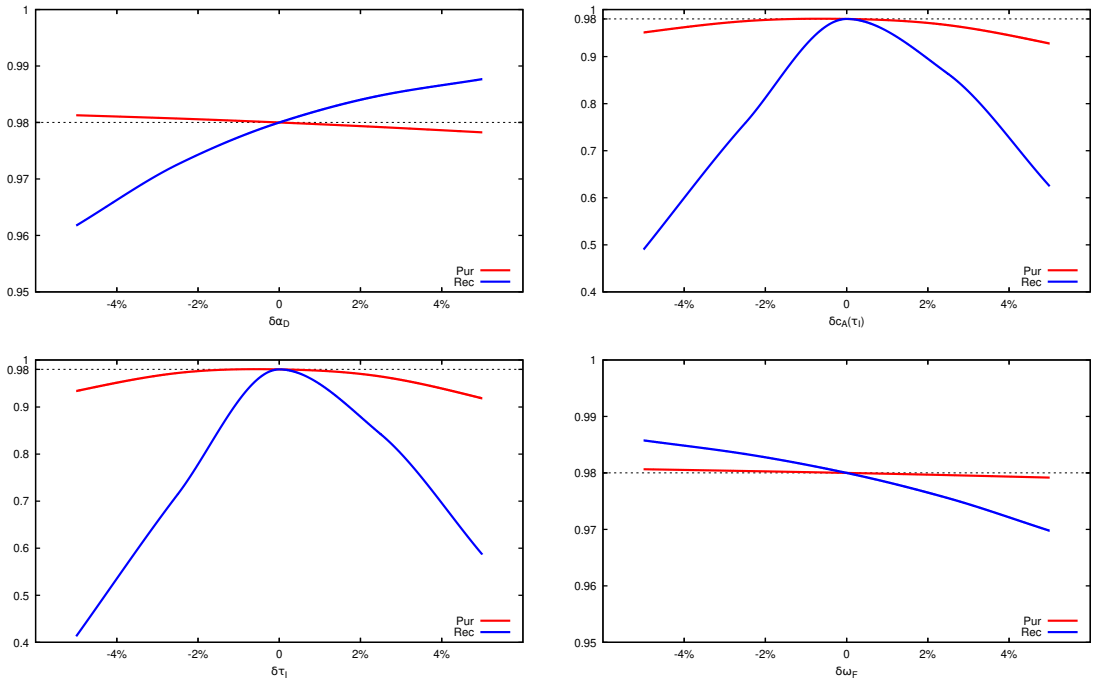

Figure 3.7: Impact of the decision variables in the recovery and purity. In all cases, the recovery is way more affected than the purity, even for small deviations of 2%

a maximum deviation of 2% from the optimal value. This shows that the optimal set of decision variables is not robust. Even though the results seem discouraging, it is worth nothing that a variability of 2% is actually quite high, since a regular HPLC pump can ensure flowrate accuracies of around 0.1% at maximum. Nevertheless, the optimal values presented are robust, since we can say that, due to the curves behavior in Figure 3.6, any value inside of the confidence interval of 2% is considered optimal.

All of the previous optimizations showed that for this specific separation process streams B, C and D are not needed. In that case, we can simplify the GSSR cycle structure.

Let us take another look at Figure 1.17. By removing all isocratic solvent streams, steps 1 and 2 can be merged to a single step of length τ_I. The exact same can be done for steps 3 and 4. Since all optimization approaches gave $\alpha_F = \alpha_P = \alpha_D$, steps 5 to 7 become one step of length $\alpha_D \tau_I$ and step 8 remains the same. With these changes, we can simplify the cycle to four steps, as shown in Figure 3.8.

Optimization of this simplified version gave the optimal values of $\tau_I = 4.57$min, $\alpha_P = 0.37$, $c_A(0) = 0.312$v/v, $c_A(\tau_I) = 0.263$v/v and $w_F = 16.86$ml, hence $Q_F = 9.97$ml/min. Regarding the throughput and the KPIs, $\overline{Q_F} = 1.23$ml/min, $CR = 1.72$ and $SCR = 4.19$. These values are as good as the values of approach IV, with the added advantage of this cycle being easier to experimentally implement. For modeling purposes, this cycle originates a smaller system of equations, which also speeds up the optimization process. One more interesting detail is the fact that the gradient slope is negative, something that is very hard to predict. This tells us that using modeling and programming tools to simulate

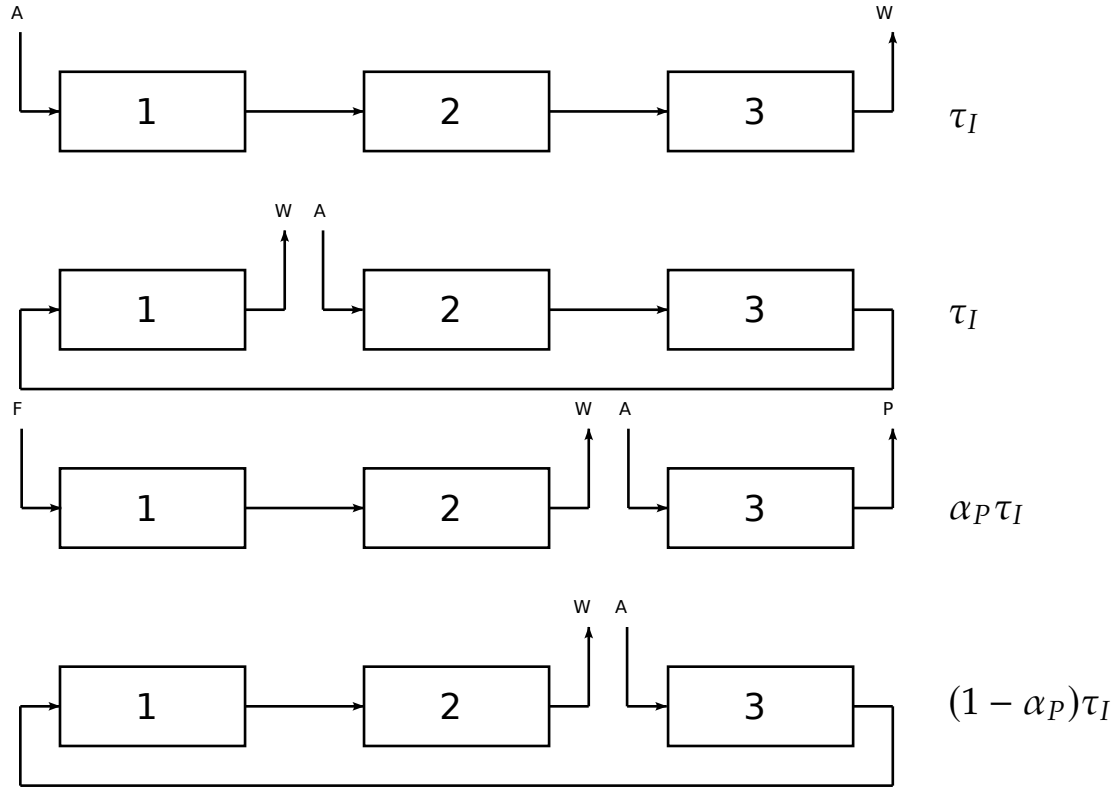

Figure 3.8: Simplified version of the GSSR process. Since the isocratic streams can be removed for this separation, the 8-step process can be simplified to 4 steps only.

and optimize these types of complex systems has great potential, since it is very difficult to deduce enough heuristic rules to encounter the optimal way of running such processes.

4

Experimental Work

A lot of work was done in order to implement a robust GSSR optimization using state-of-the-art computational hardware and software, with generic models and efficient numerical approaches. However, this is an innovative work, in which we have little to no literature to base on. In that case, the only way to verify is to make our own experiments and check if our models can accurately predict the outcome of such experiments. That is why experimental validation is so important, because it verifies if our theories are correct or not.

4.1 Laboratory Implementation

Simulation and optimization of any chromatographic process is definitely the way to go when it comes to reducing costs in raw materials and increasing productivity and recovery. However, for us to confirm if our optimizations are accurate, we need to test them by running some experiments. Furthermore, several parameters that are part of the optimization problem are specific to the experimental set-up, such as dead volume, bed porosity and column volume.

As explained in Section 2.2, we built our own experimental prototype in order to perform some experimental validations of our optimizations results. This section is meant to depict in detail about our set-up, such as the hardware used and the experiments performed to determine all the properties specific to it.

4.1.1 Materials and Methods

For the assembly of our set-up, we used three HPLC pumps, two Knauer K501 isocratic pumps with 10ml/min heads and one Knauer S1050 gradient pump with a 10ml/min head. The pathways are controlled by twenty five ON/OFF pneumatic valves from Valco, which are controlled by 5 Paralab,SA solid-state ver.2 relay boards. For columns, we have three 3ml SOURCE15RP C columns from Cytiva, with a bead size of 15μm. The capillaries used are stainless steel 1/16" ID. To measure the real flowrate we have installed also a

Table 4.1: Available macros

Macro	Description
`@uvc`	controls every aspect of a spectrometer
`@pump`	controls every aspect of an HPLC pump
`@valve` or `@valves`	controls every aspect of a pneumatic valve
`@step`	defines a specific pathway for the fluid
`@csinfo`	detailed information of every capillary, connector and valve in the system
`@monitor`	creates a file for monitoring and records every event in that file

Sartorius TE3102S balance. As a light source we used a Mikropack DH-2000. For UV detection we have a USB2000 and a USB4000 spectrometers, both from OceanOptics.

4.1.2 Software Implementation

Since we assembled the set-up from scratch, we also needed to create software to control all the equipment and automate the experimental protocols. For that purpose we chose *Julia* [105] to create the *ChromatographyStudio* package.

Julia is a programming language that was created mainly for numerical computing. However, due to its popularity, it started to grow to be able to do much more than just numerical computing. With this in mind, we gave it a try and created our control software entirely in *Julia* .

In order to do it, we used external software for the actual control and monitoring of our equipment, such as a C library for communication through RS-232 ports [106], a C library for reading the spectrometer data called *SeaBreeze* [107] and a C library for communication with the solid-state relays called *Comedi* [108].

4.1.2.1 Interaction with the software

Until now, our software has no graphical interface, everything has to be made in the command line. The way we interact with the software is through the use of macros, which work similarly as a usual function of any programming language, but the writing is more user-friendly and makes it possible for anyone to use it with a low learning curve.

The indicator of a *Julia* macro is the `@` sign and the name of the macro comes right after. The syntax is:

```
@macro_name command1 command2 command3 ...
```

Each macro argument is separated by a white space. Table 4.1 represents all the macros implemented in *ChromatographyStudio* .

These macros are the cornerstone of *ChromatographyStudio* , with them we can do anything with our system. We are able to combine them efficiently to run any type of

experiment, from simple pulse injections to full GSSR operation cycles in an intuitive manner.

4.1.2.2 Performing an Experiment

Let us simulate a standard pulse injection using *ChromatographyStudio* . Firstly, we have to import our package, which in *Julia* is done as follows:

```
using ChromatographyStudio
```

Now we need to prepare our software to read and analyze UV data. Since the USB2000 and USB4000 can read more than 2000 wavelengths, from 180 to 700nm, we need to set the wavelengths we want to plot in real-time:

```
@uvc A wavelengths=[220.0,230.0,250.0]
```

In this example we sent two parameters to the `@uvc` macro. the argument `A` represents the ID of the spectrometer used. This ID is useful because we can use both simultaneously, so each one needs a different ID so we can distinguish both. The spectrometer ID is chosen before importing the package, which by default is `A` for the USB2000 and `B` for the USB4000. For the sake of this example we will assume only the USB2000 spectrometer is connected.

Now we need to set the dark and reference spectra:

```
@uvc A set=dark
@uvc A set=ref
```

These arguments are used to establish the spectrum baselines. The argument `set=dark` is needed only once after *ChromatographyStudio* is imported, while the argument `set=ref` is used to set the baseline to zero, hence can be used at any time, and is always recommended to be used before the start of any experiment.

Finally, for real-time plotting of the UV data we run:

```
@uvc A plot=all
```

This argument opens a window with the UV data in real-time of the wavelengths previously chosen.

The next step is to set the flowrate of an HPLC pump. This can be achieved in the following way:

```
@pump F Q=1.0
```

The first argument `F` represents the pump ID. In this case, `F` stands for feed. The ID of the other two pumps are `E` and `G`, representing the elution and gradient pumps, respectively. With the second argument we are setting the pump flowrate to 1ml/min. `Q` is the nomenclature for volumetric flowrate. The `@pump` macro has several other arguments, but setting the flowrate is, by far, the most used argument, so we decided to create a simpler version just for the flowrate:

```
@pump F = 1.0
```

If you only want to set the flowrate, you can drop the `Q` and it still works.

The next step is to decide which pathway to use in our experiment. In this case, if we want to inject feed, send it through column 2 (we have three columns connected) and send it to waste we would use the `@step` macro in the following way:

```
@step F => COL2 => W
```

The translation is simple: we opened the path that starts in the feed pump, goes through column 2 and ends in the waste tank. We can join two pumps in the same path, as well as combining more than one column:

```
@step F+G => COL21 => P
```

Now we are injecting both pumps `F` and `E`, the fluid is entering column 2 and the outlet of column 2 goes to column 1, which instead ends in the product tank. The number order in `COL21` is important, it defines the order in which the columns are being used.

In general, the `@step` macro opens and closes the necessary valves for the specified pathway. However, if you want to open or close specific valves, for testing purposes or to specify a pathway not supported by `@step`, the `@valves` macro can help:

```
@valves 1,2,3,-4,-5,-6
@valves -all
```

Each valve number must be separated by commas. A positive value is for opening while the negative value is for closing. The first case, for common experiments, will not be needed, but the second case is used very often to signal the end of an experiment or simply close all the valves to prepare the system for shutdown.

The only thing that is still needed is to learn how to record the data in a file for later treatment. This can be accomplished with the `@monitor` macro:

```
@monitor file = "filename.mon"
@monitor stop
```

The `.mon` extension is important, *ChromatographyStudio* only writes the data in files with said extension. The second version is used to stop recording. Furthermore, the file written is in binary format to be lighter and more efficient during the real-time data recording, so it must be converted before it can be analyzed by common software. This conversion can be made with the `@mon_to_tsv` macro:

```
@mon_to_tsv "filename"
```

In this macro the extension can be dropped. It will find a file called `filename.mon` and convert it to `filename.tsv`, which then can be read by any data treatment software.

Finally, as an example, let us simulate a pulse injection experiment. We will use the feed pump to inject our target component to column 1 during 5 seconds at a flowrate of 1ml/min. Then, we will use the eluent pump for 2 minutes at a flowrate of 2ml/min to elute the component from the column. The experimental protocol would look like this:

```
using ChromatographyStudio
@uvc A wavelengths=[220.0,230.0,250.0]
@uvc A set=dark
@uvc A set=ref
@uvc A plot=all

@monitor file="monitorfile.mon"

@step F => COL1 => W
@pump F = 1.0
sleep(5.0)
@pump F = 0.0

@step E => COL1 => W
@pump E = 2.0
sleep(120.0)
@pump E = 0.0

@valves -all
@monitor stop
```

Before changing pathways with `@step`, we must turn off the flowrate of the pumps that are affected by the new pathway. In this example, before setting `@step E => COL1 => W`, we must set the flowrate of the feed pump to zero, otherwise the pump will shut down due to over pressure. The `sleep` function is an internal *Julia* function that works as a timer. The argument of the function must be in seconds and until the timer is finished, no other action will take place.

The *ChromatographyStudio* software has many more functionalities, but they will not be explored in this section. A documentation guide with the software explained in more detail is presented as an annex in this work.

4.1.3 Determination of System Properties

The set-up created comes with inherent parameters that must be determined, such as the column porosity, internal dead volume and hydrodynamic behavior of the fluid in the dead volume. For that, we performed several experiments, as well as some simulations, to achieve this objective.

4.1.3.1 Column Porosity Experiments

As said before, we have three SOURCE15RPC 3ml columns with a bead size of 15μm. The column itself has an internal dead volume, which is due to the geometry of the beads and their internal porosity. This porosity is very important to measure, since it contributes to the total dead volume in the system, as well as to the separation efficiency.

The total porosity of a packed column can be divided in two terms, the internal and external porosities. The internal porosity represents the void volume inside the particles, while the external porosity represents the void volume outside the beads due to their geometry. The total void volume of the column is represented by:

$$V_0^T = V_0^b + V_0^s \tag{4.1}$$

Where V_0^T is the total column dead volume, V_0^b the total void volume outside the particles and V_0^s the dead volume inside the particles. These three parameters can be written as a dependence of the column volume:

$$\begin{aligned} \epsilon_T V_c &= \epsilon_b V_c + (1-\epsilon_b)\epsilon_p V_c \\ \epsilon_T &= \epsilon_b + (1-\epsilon_b)\epsilon_p \end{aligned} \tag{4.2}$$

Where ϵ_T is the total dead volume fraction, ϵ_b the external dead volume fraction and ϵ_p the dead volume fraction inside the particles only. There are several experimental techniques to determine these parameters. For instance, the internal particle porosity can be determined by mercury porosimetry [109], the total porosity can be determined by pulse injections of salt and by using conductivity sensors instead of UV sensors we can calculate the retention time, since salt molecules are smaller than any pore size and the external porosity can be determined by pulse injections of BDex, since it has a big molecule size and is not able to enter the particle pores. Furthermore, you only need to experimentally determine two of them, since the other one can be determined by their physical relations.

An isotherm model represents the adsorbed mass in equilibrium with the mass in the surrounding fluid. This means that, in this case, the total amount of fluid is $\epsilon_T V_c$, while the amount of solid is $(1-\epsilon_T)V_c$. However, we can determine an apparent isotherm model by assuming the apparent surrounding liquid $\epsilon_b V_c$ is in equilibrium with the apparent solid $(1-\epsilon_b)V_c$. In this way we are accounting the component mass that is inside the particles, both adsorbed in the solid and floating inside the pores. The end result is the same and experimentally determining ϵ_b is more cost-effective than ϵ_T, since we can use UV data instead of conductivity sensors.

To determine the external porosity, we used BDex as our tracer performed pulse injections using an Äkta Avant 150 from Cytiva. For that we need to perform two distinct injections: one without column, to determine the system dead volume, and another with the column, to determine the external void volume. The results of said experiments were gathered in Figure 4.1.

With these experiments we can easily determine the external porosity:

$$V_0^b = \epsilon_b V_c \Leftrightarrow \epsilon_b = \frac{V_0^b}{V_c} \Leftrightarrow \epsilon_b = \frac{(1.935-0.641)ml}{3.0ml} \Leftrightarrow \epsilon_b = 0.431 \tag{4.3}$$

This porosity will be the one used for optimization of our central-cut experiments.

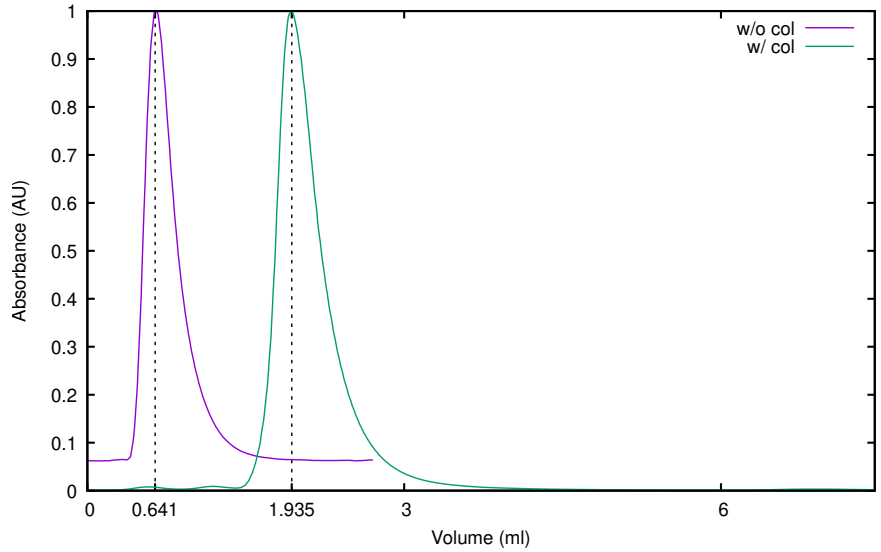

Figure 4.1: Determination of the external porosity ϵ_b. 0.641 is the system dead volume and 1.935 is the system dead volume plus the volume outside the particles.

4.1.3.2 Dead Volume Experiments

In any experimental set-up, the combination of tubing, UV sensors, electric and/or pneumatic valves, etc, create an amount of volume that is inherent to the system in question. Depending on the complexity of said system, that volume depends on the pathway chosen for the fluid to travel. Our system is not different. Considering all the material and hardware used to build it, that extra volume has to be taken into account in order for our models to be as close to reality as possible.

The system has three columns, all connected to each other to ensure maximum operational flexibility. However, this increased flexibility comes with an increase in available pathways, which leads to an increase in dead volume.

Diagram 4.2 shows some examples of configurations. The nomenclature used in the Figure is the same throughout the rest of this document. As you can see in the figure, the number order is important, it represents the column sequence in the pathway. The first number represents the column that is directly connected to the pumps.

The system was built to minimize the dead volume discrepancies between pathways, i.e. the dead volume for all 1-column pathways is identical, as well as for 2-column and 3-column pathways.

The simplest way to test the dead volume is with pulse injections. Since we have to remove the columns for these experiments, the injections can be made with any component that can be detected by UV. In order to not use our raw materials, we used Blue Dextran (BDex) because it is cheaper and produces very good UV signals. In that matter we dissolved BDex in water, with unknown concentration (the concentration is irrelevant, since we only want to evaluate the retention times). We used our feed pump for the injections and our gradient pump for the elution. The feed pump flowrate was 0.1ml/min

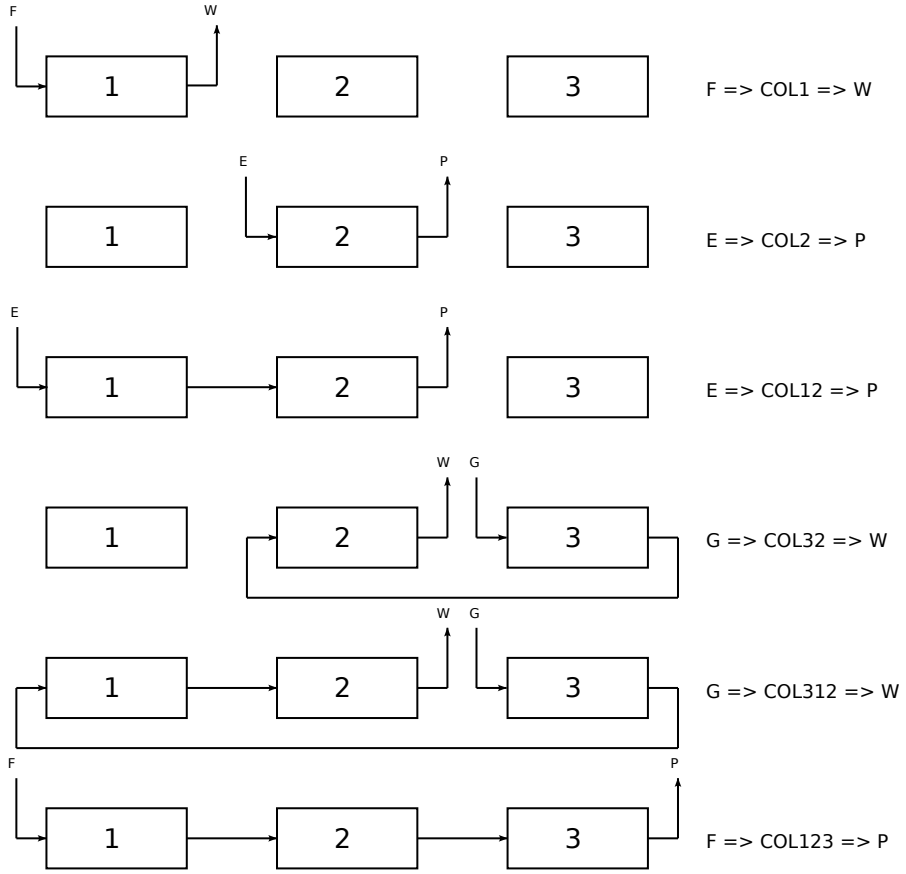

Figure 4.2: Pathway examples in the Lab Set-up. The notation in the right of the diagrams is the notation used in our *ChromatographyStudio* software. The number order is important, it represents the sequential order of the columns used.

and the injection time was 10 seconds, which injects a volume amount of 0.017ml. The elution volume was variable, it depended on how many columns the pathway had. Furthermore, since we want to test the GSSR process, so we only need to determine the dead volume of the pathways used by the GSSR. By taking another look at Figure 1.17, we can reduce the number of pathways to be evaluated to the ones presented in Table 4.2.

For every experiment three replicas were produced in order to check for reproducibility. Examples of 1-column, 2-column and 3-column dead volumes can be seen in Figures 4.3, 4.4 and 4.5, respectively. The dead volumes of all pathways are summarized in Table 4.3. If we calculate the relative standard deviation of each number of columns, we come with values of 10.9%, 1.6% and 3.5% for 1-column, 2-column and 3-column configurations. Experience tells us that values below 5% are considered statistically viable. That being said, the value for 1-column configurations is too high to be considered viable. However, the GSSR has the feed stream fixed in column 1 and the product stream fixed in column 3. In that way, the product will always go through the 3 columns before being collected, so as long as the 3-column configurations are viable, it is not important for the others to also be.

Table 4.2: Every pathway used by the GSSR process.

nº columns	pathways
1	COL1
	COL2
	COL3
2	COL12
	COL23
	COL31
3	COL123
	COL231
	COL312

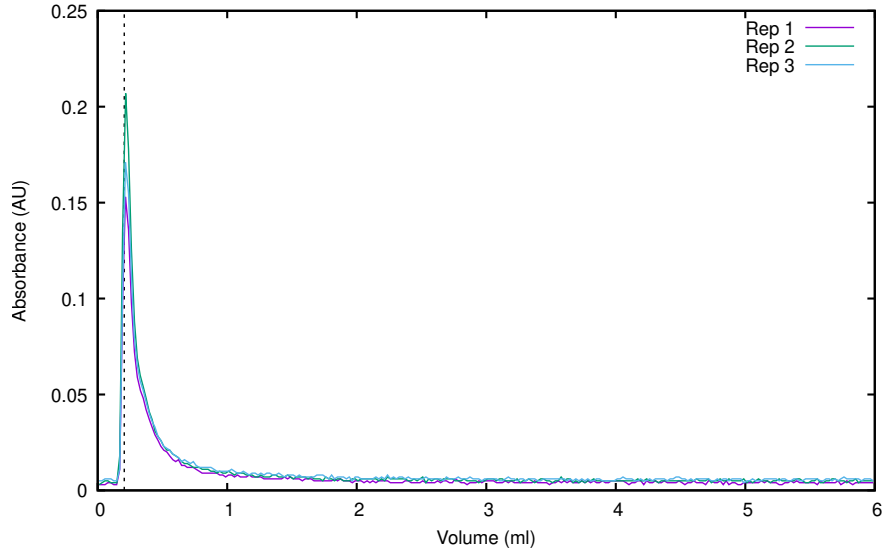

Figure 4.3: Determination of the dead volume for the COL3 configuration.

Finally, we want to determine the extra dead volume per column V_x, which is needed for our mathematical models. The set COL_3 represents the 3-column configurations previously mentioned, NCONF the number of configurations and NCOL the number of columns. By calculating the average dead volume and divide it by the number of columns, we can determine V_x. It takes the form:

$$V_x = \frac{\sum_{i \in \text{COL}_3} V_{x,i}}{NCONF \times NCOL} = \frac{1.899 + 1.777 + 1.808}{3 \times 3} = 0.609ml \tag{4.4}$$

This is the value used in the future optimizations for central-cut separation.

4.2 Isotherm Determination

The main purpose of the experimental work is to perform a central-cut separation, which involves a mixture of three or more compounds, as explained in Section 1.4.3.

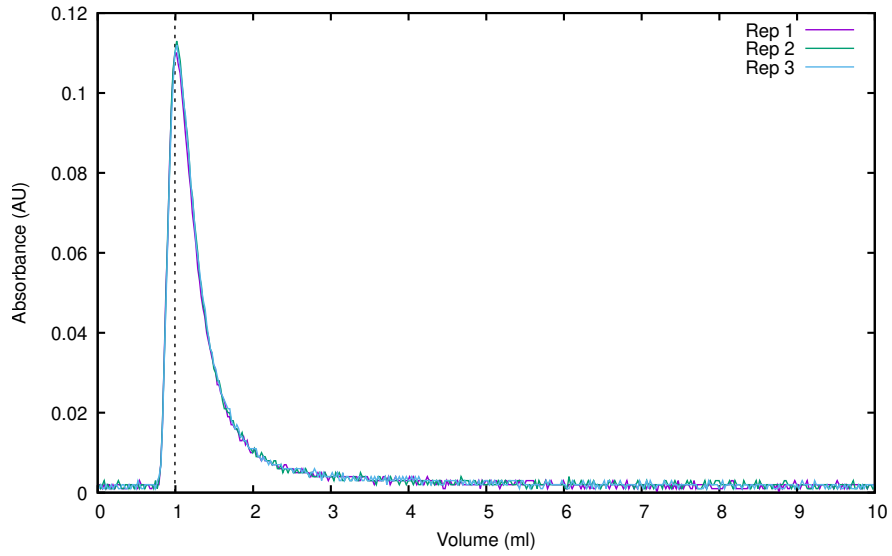

Figure 4.4: Determination of the dead volume for the `COL12` configuration.

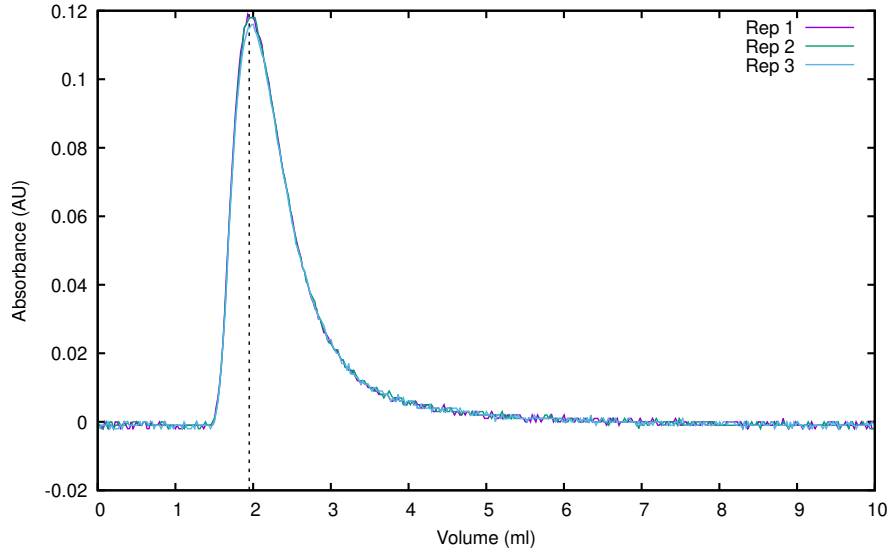

Figure 4.5: Determination of the dead volume for the `COL123` configuration.

Furthermore, we wanted to test our models with a hard separation problem, in which the product has a high overlay with the impurities. We also wanted a simple experimental protocol, so we decided to create a ternary mixture of aminoacids. Figure 4.6 groups the aminoacids due to their polarity. Since we have three SOURCE15RPC columns, which are for reversed-phase chromatography, we chose the most apolar aminoacids and went along with them to try and find three that could satisfy the conditions we imposed. In that case, using the chart presented, all aminoacids in the "D" area plus Glycine were chosen. After searching on PubChem, a well-known website to find information about chemical compounds [110], we found that they had the best solubilities when using methanol as the organic solvent.

The experiments were conducted in an Äkta Avant 150 HPLC from Cytiva. All 9

Table 4.3: Every pathway used by the GSSR process.

nº columns	pathways	Dead volume (ml)
1	COL1	0.215
	COL2	0.173
	COL3	0.192
2	COL12	0.968
	COL23	0.960
	COL31	0.938
3	COL123	1.899
	COL231	1.777
	COL312	1.808

aminoacids were dissolved in a mixture of water with 25% methanol with a concentration of ~2.0 g/L. The exact concentration was not important, since we only wanted to determine the retention volume. For each aminoacid, three injections of 100μL were performed in order to ensure reproducibility. In the end, we gathered all the data and came up with Figure 4.7. By taking a closer look, we can see that the retention volumes for Alanine, Proline and Tryptophan are very similar, which makes it a promising mixture to continue further.

After choosing the three components to use from now on, we have to test their adsorption behavior by determining the isotherms of all three components. We have to take into account that the adsorption of biological components usually depend on the amount of organic solvent in solution, so in this case the isotherms have two independent variables, the concentration in the bulk and the concentration of organic solvent, in this case methanol. In this case, we decided Frontal Analysis was the way to go.

4.2.1 Frontal Analysis

Frontal Analysis (FA) is an experimental technique that allows us to determine the mass that is adsorbed in the resin of a chromatographic column. As explained in Section 1.1.3, the amount of adsorbed mass is directly correlated with the concentration that is injected inside the column, due to the concentration driving force. By following this principle, we can determine the amount adsorbed by comparing the mass injected in the column and the mass that leaves the column.

To further explain this, let us create an example. We have a column packed with 1ml of resin and the total porosity of the bed is 0.6. Now we inject a solution of concentration 1g/ml for 10 minutes at a flowrate of 1ml/min. Furthermore, the system that we are using for these experiments, without the column, has an inherent dead volume of 0.5ml. The result of the experiment is represented in Figure 4.8. In this image, c_{out} is the concentration in the column outlet, V_r is the retention volume and V_0 is the system dead volume. The colored areas A_u and A_l are used to find V_r. Since c_{out} is an experimental curve, we cannot

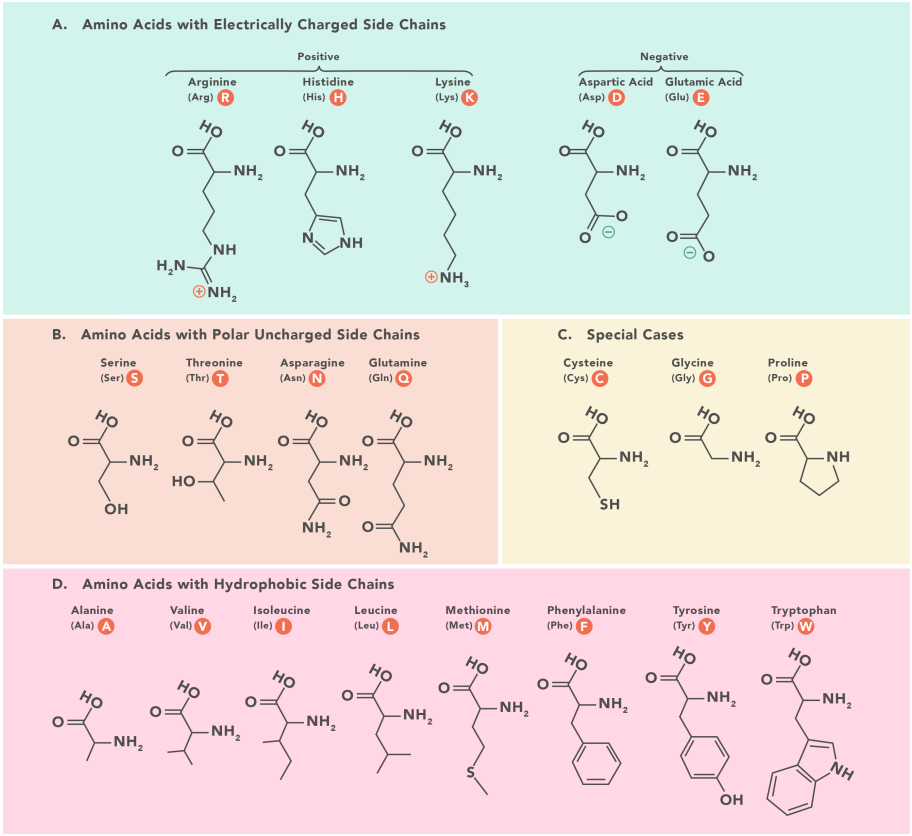

Figure 4.6: Aminoacid Polarity Chart. The aminoacids used for screening were all the ones in the "D" area plus Glycine and Proline in the "C" area [111].

find the second derivative and get the inflexion point. However, what we can do is to numerically determine the area above the curve - A_u - and below the curve - A_l - and find the point where $A_u = A_l$. In this way you can find the inflexion point, which represents the retention volume of the component.

Now, since we know the dead volume and determined the retention volume, we can determine the mass accumulated in the solid. c_{out} represents the mass that is leaving the column, so the total mass of product that left the column is represented by,

$$M_{out} = \int_{V_0}^{V_f} c_{out} dV \tag{4.5}$$

M_{out} is the total mass that left column in a given time interval t_f, $V_f = t_f Q_F$ is the total amount of volume injected, which in this case is the time length of the experiment multiplied by the injection flowrate. In the other hand, the mass that stayed in the column is expressed as,

$$M_{acc} = M_{in} - M_{out} = \int_{V_0}^{V_f} c_{in} dV - \int_{V_0}^{V_f} c_{out} dV \tag{4.6}$$

Where M_{acc} is the accumulated mass in the column and M_{in} the total mass that entered the column. To obtain these two terms we need to perform two experiments, one without

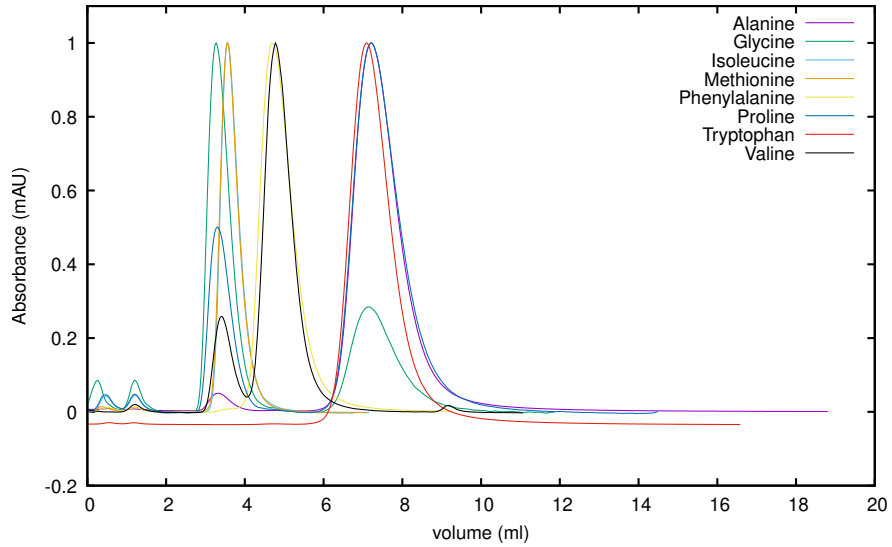

Figure 4.7: Aminoacid Screening. Due to their proximity in retention volume, the aminoacids chosen were Alanine, Proline and Tryptophan.

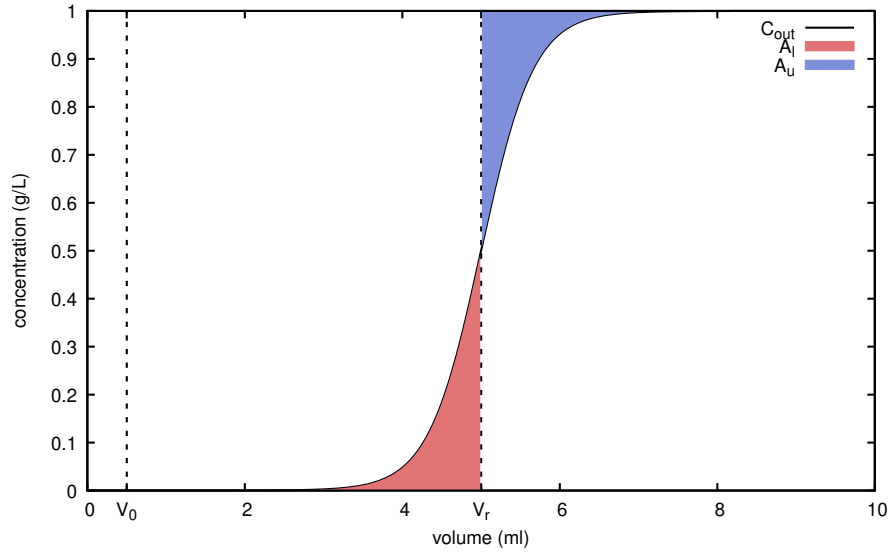

Figure 4.8: Frontal Analysis experiment. c_{out} is the concentration in the column outlet, A_u is the area above the curve and A_l the area below the curve. V_0 and V_r represent the dead and retention volumes, respectively.

the column to get the first term and another with the column to get the second term. In order to reduce the number of experiments, instead of determining M_{in} and M_{out} we only need to determine M_{out} and determine the retention volume V_r. In that way, we can determine the accumulated mass as follows:

$$M_{acc} = (V_r - V_0)(c_F - c_i) \tag{4.7}$$

Where c_F is the concentration of the feed and c_i is the concentration already inside the column. Furthermore, the accumulated mass can be divided in two terms: the mass in

the void volume of the column and the mass adsorbed in the resin:

$$M_{acc} = c_F V_c \epsilon_t + q V_c (1 - \epsilon_t) \tag{4.8}$$

$V_c = 1$ is the column volume, $\epsilon_t = 0.6$ is the total bed porosity and q is the adsorbed mass per volume of resin. We can join both expressions to form the following:

$$c_F V_c \epsilon_t + q V_c (1 - \epsilon_t) = (V_r - V_0)(c_F - c_i) \tag{4.9}$$

Which by algebraic manipulation leads to:

$$q = \frac{(V_r - V_0)(c_F - c_i) - c_F V_c \epsilon_t}{V_c (1 - \epsilon_t)} \tag{4.10}$$

This calculation protocol needs to be applied to different feed concentrations, in order to create a plot similar to Figure 1.2. To obtain a good isotherm prediction, we need several data points, so we need to create several solutions with different concentrations in order to determine a good isotherm approximation. Another way is to perform a stair experiment, which will be discussed in the next section.

4.2.2 Stair Experiments

A stair experiment is an experimental protocol that reduces the amount of experiments needed to determine an isotherm. In order to determine an isotherm model we need several equilibrium points. If we want to have 10 equilibrium points using FA, we would need to create 10 solutions at different concentrations and repeat the calculation procedure in order to determine the adsorbed mass in equilibrium. A stair experiment can reduce these 10 experiments to 1.

The way is quite simple. Let us assume we want to study the adsorption behavior from 0 to 2g/ml of a component. We will stick to the same operation conditions as before, an injection time of t_F of 10min and a flowrate Q_T 1ml/min. The feed solution has a concentration c_F of 2g/ml and the eluent stream has a concentration c_E of 0g/ml. If we want to obtain 10 equilibrium points, we can play with the flowrates of both pumps in order to inject the concentration desired. For instance, if we want to test the equilibrium at $c_F = 0.2 g/ml$, then the feed flowrate would be $Q_F = 0.1 ml/min$ and the eluent flowrate would be $Q_E = Q_T - Q_F = 0.9 ml/min$. Furthermore, we do not need to clean the column first before the next injection, after the equilibrium is reached we can inject a bigger concentration, such as 0.4g/ml. In this way, $Q_F = 0.2 ml/min$ and $Q_E = 0.8 ml/min$. We can repeat the process until $Q_F = Q_T$. Figure 4.9 shows what the outcome of a stair experiment usually looks like. The image example depicts three frontal analysis curves, in which stair i reaches a plateau $c_{f,i}$, which represents the injected concentration in the beginning of the stair.

The calculation protocol is similar to the FA protocol, we just need to adjust the parameters for every stair.

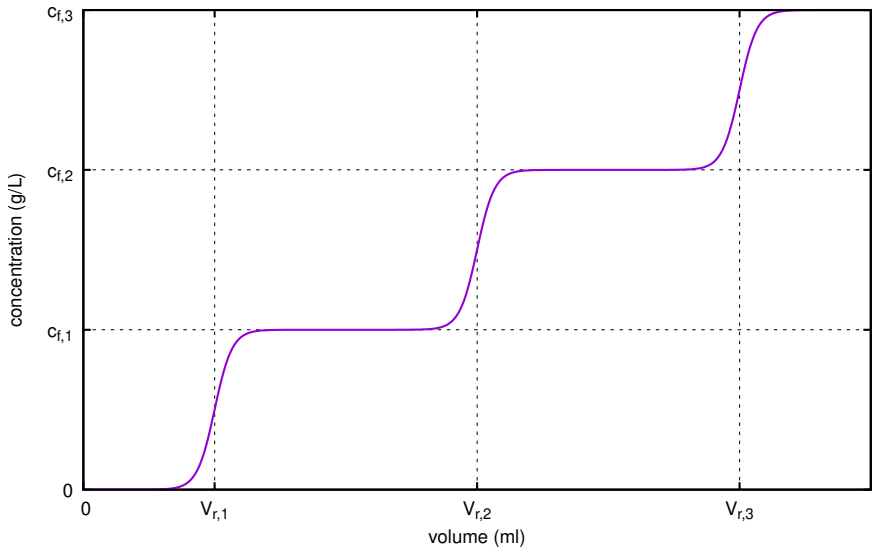

Figure 4.9: Stair experiment. Each stair i reaches a plateau $c_{f,i}$ and has a retention volume $V_{r,i}$.

As a general example, we will assume we want to observe the adsorption behavior in NS concentration points. The total flowrate Q_T is equal to the sum of both Q_F and Q_E, the feed and eluent flowrates, respectively. As we can see, the initial concentration inside the column of one step is equal to the injected concentration in the previous step. Furthermore, the initial instant t_0 when each step starts is equal to the instant that the previous step ended. In other words:

$$t_{0,i} = (i-1)t_f \qquad \text{for} \quad i = 1, ..., NS \tag{4.11}$$

The beginning of the first step is $t_0 = 0$, the second step $t_0 = 10$, etc. Regarding the accumulated mass of each step, we can express it in the following way:

$$\begin{aligned} M_{acc,1} &= (V_{r,1} - V_0 - t_{0,1}\, Q_T)(c_{f,1} - 0) \\ M_{acc,i} &= (V_{r,i} - V_0 - t_{0,i}\, Q_T)(c_{f,i} - c_{f,i-1}) \qquad \text{for} \quad i = 2, ..., NS \end{aligned} \tag{4.12}$$

For the first step the column is completely clean from product. If the concentration steps are evenly spaced, we can simplify the previous equation:

$$M_{acc,i} = (V_{r,i} - V_0 - t_{0,i}\, Q_T)\frac{c_F}{NS} \qquad \text{for} \quad i = 1, ..., NS \tag{4.13}$$

The accumulated mass calculated is simply the amount of mass that stayed in the column in the specific step. However, in order to determine the real accumulated mass inside the column in equilibrium with a certain concentration, we have to sum all the accumulated masses calculated in the previous steps. Mathematically speaking:

$$\bar{M}_{acc,i} = \sum_{j=1}^{i} M_{acc,j} \tag{4.14}$$

Table 4.4: Operation parameters for the stair experiments.

Parameter	Units	Value
c_F	g/L	2.0
Q_T	ml/min	2.0
NS	-	10
t_f	min	4.0
c_0*	v/v	[20,30,40]

* index 0 represents methanol fraction

Where $\bar{M}_{acc,i}$ is the real accumulated mass in equilibrium with the concentration injected in step i. We also know that:

$$\bar{M}_{acc,i} = c_{f,i} V_c \epsilon_t + q_i V_c (1 - \epsilon_t) \qquad \text{for} \quad i = 1, ..., NS \tag{4.15}$$

Now we can determine the adsorbed mass in equilibrium with each concentration:

$$q_i = \frac{\bar{M}_{acc,i} - c_{f,i} V_c \epsilon_t}{V_c (1 - \epsilon_t)} \qquad \text{for} \quad i = 1, ..., NS \tag{4.16}$$

The calculation protocol is a bit more extensive and complex, but we significantly reduce experimental time and resources, which is a very positive trade-off.

4.2.3 Results

To obtain adsorption data for all three components, we chose to perform stair experiments. As previously mentioned, adsorption behavior of biological compounds tend to be influenced by the solvent composition, so we had to verify that by performing the stair experiments for different methanol compositions also. Table 4.4 shows the operation parameters chosen for the experiments.

Figures 4.10, 4.11 and 4.12 show the stair experiments performed for Alanine, Proline and Tryptophan, respectively. The curves for Tryptophan were the first ones to be performed. In that time we were still calibrating the minimum amount of time necessary for equilibrium to be reached at every stair, that is why the volume injected for each methanol concentration is different. The other two components were performed after calibration was finished, so the amount of volume is the same for both. As a primary analysis, we can verify that the stairs of both Alanine and Proline barely change with the change in solvent concentration. The same cannot be deduced for Tryptophan.

By applying the algorithm explained in Section 4.2.2, the isotherms were calculated and can be seen in Figures 4.13, 4.14 and 4.15.

According to Alanine and Proline isotherms, they confirm what we could see in the stair experiments: they have a weak dependence, almost negligible, on the methanol concentration. The same cannot be said about Tryptophan; it is clear now that Tryptophan isotherm is significantly dependent on the methanol content.

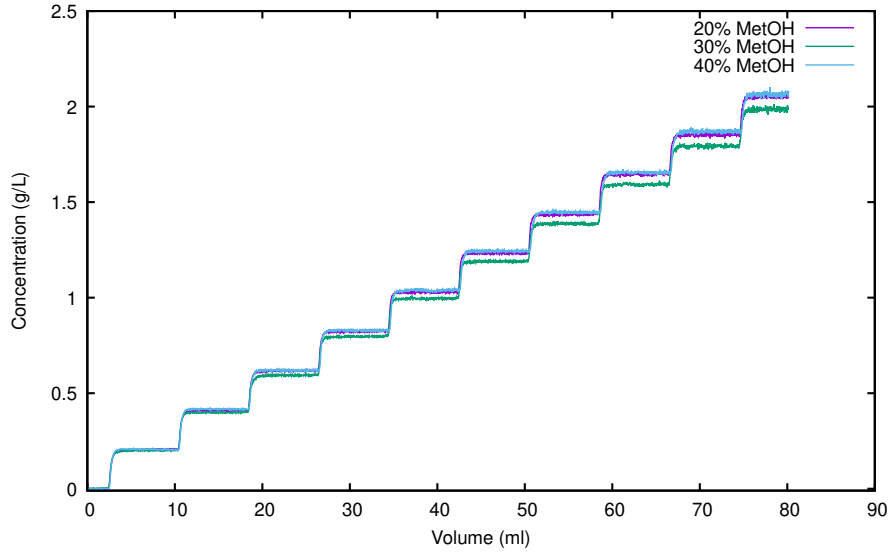

Figure 4.10: Stair experiment for Alanine.

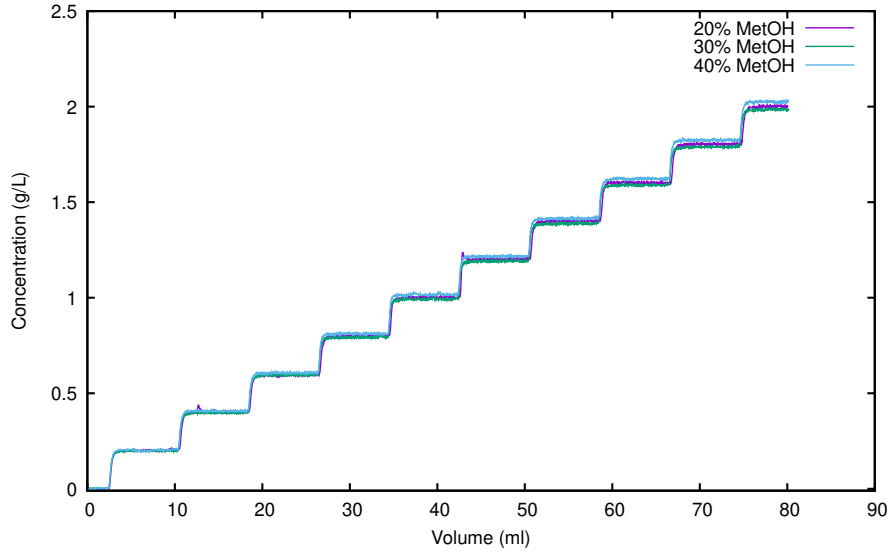

Figure 4.11: Stair experiment for Proline.

There is one major information that we can get from the data: the isotherms of Alanine and Proline are very similar, which was expected due to the aminoacid screening performed. However, the values for Tryptophan are way above the values of the other two. This is very different from the information we gathered from the screening. We are not completely sure how this discrepancy appeared, but after some investigation we came up with a reasonable explanation. The aminoacids used for screening were very old, they were first opened around 3 years before these experiments took place. After the screening, we freshly ordered the three aminoacids to have a good quantity. We are suspicious that the tryptophan used for the screening was already highly impure and the injections were stopped too soon, before Tryptophan could actually desorb. Nevertheless, even though the retention times for Tryptophan are way superior than the other two, the retention

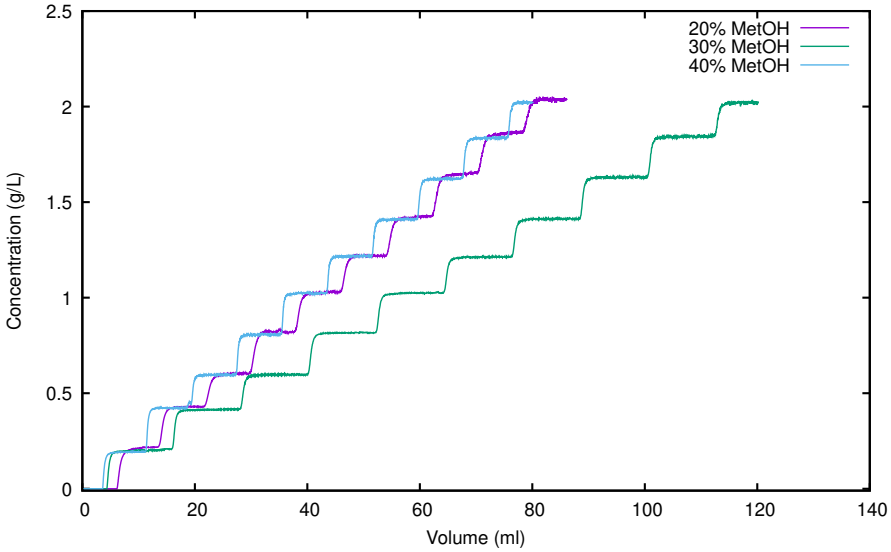

Figure 4.12: Stair experiment for Tryptophan.

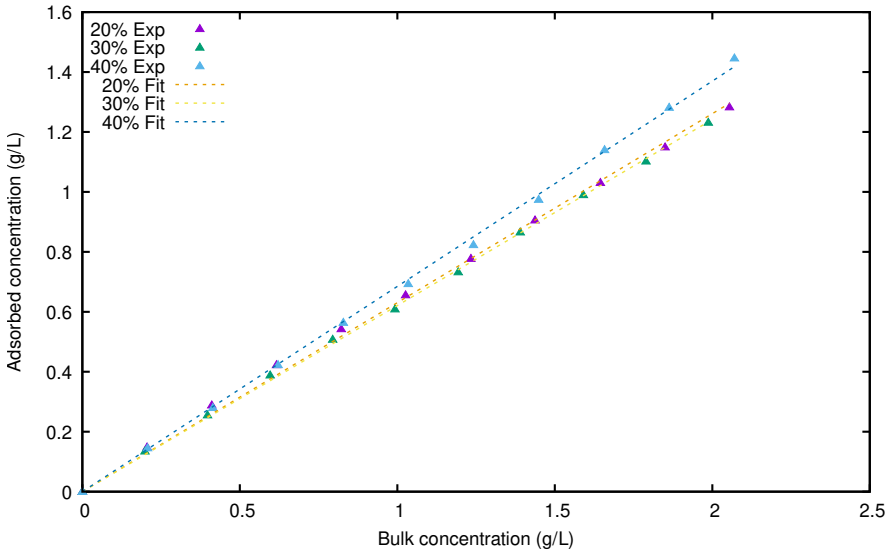

Figure 4.13: Alanine isotherm at different methanol concentrations.

times for Alanine and Proline are very similar, which still is a challenging separation process by chromatography. After this conclusion, we still think this ternary mixture is a viable choice for further central-cut separations.

We used a linear isotherm model to fit all adsorption data. Figure 4.16 shows the Henry constant dependence on the solvent concentration. The Henry values determined are summarized in Table 4.5.

The models used for fitting the Henry data were the following:

$$H_i = p_1 + p_2 c_0 + p_3 c_0^2 \qquad \text{for} \quad i = A, P, T \tag{4.17}$$

Where A, P and T represent Alanine, Proline and Tryptophan, respectively, and p_1, p_2

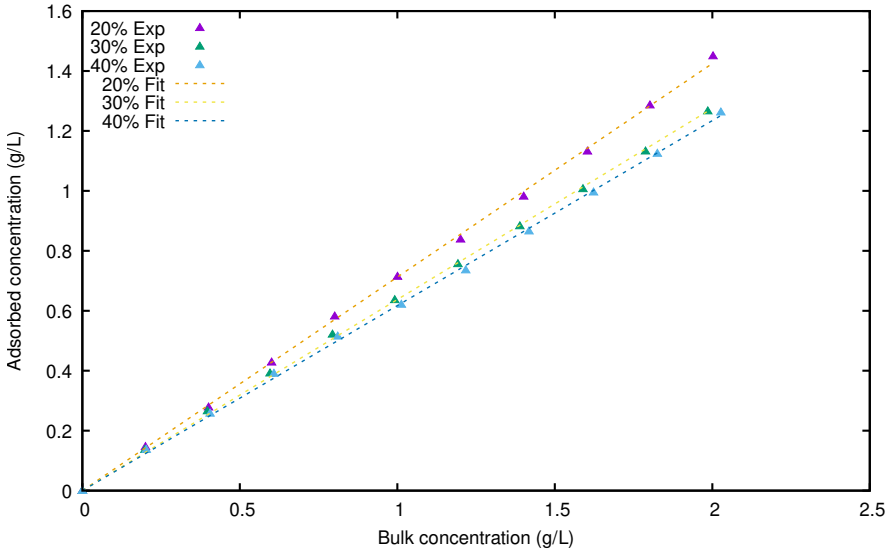

Figure 4.14: Proline isotherm at different methanol concentrations.

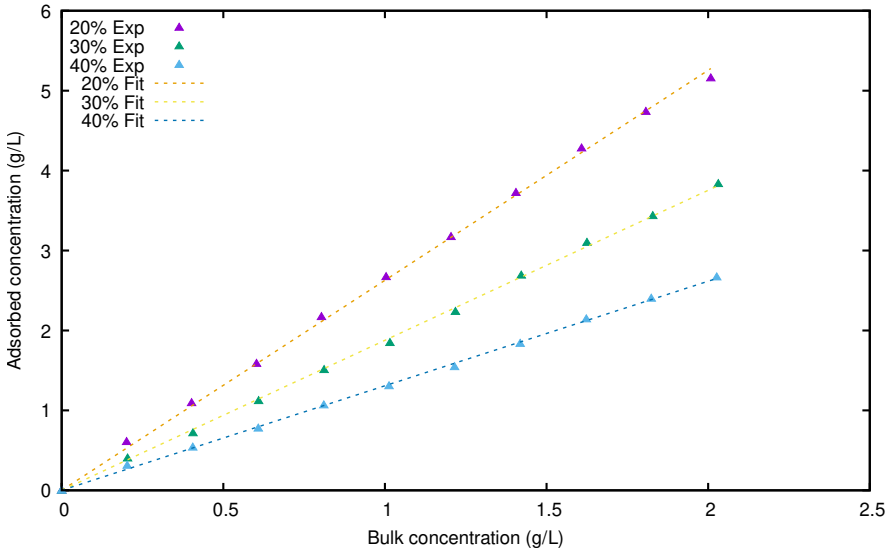

Figure 4.15: Tryptophan isotherm at different methanol concentrations.

and p_3 are fitting parameters with no physical meaning. Such parameters are presented in Table 4.6.

A very important detail about Table 4.5 are the values for Alanine. Taking a closer look, the Henry value at 30% MetOH is smaller than at 20% MetOH and 40% MetOH. This raises a warning flag, due to the fact that the adsorption behavior either increases or decreases with the increase in solvent concentration. The dependence of the Henry constant on the solvent concentration is not supposed to be convex. This information led us to perform new isotherm experiments for different solvent concentrations. However, before proceeding with more stair experiments, we could retrieve one more piece of information from the data: it is clear that the Henry values are decreasing with the increase in solvent concentration. This means that the adsorption capacity with 40%

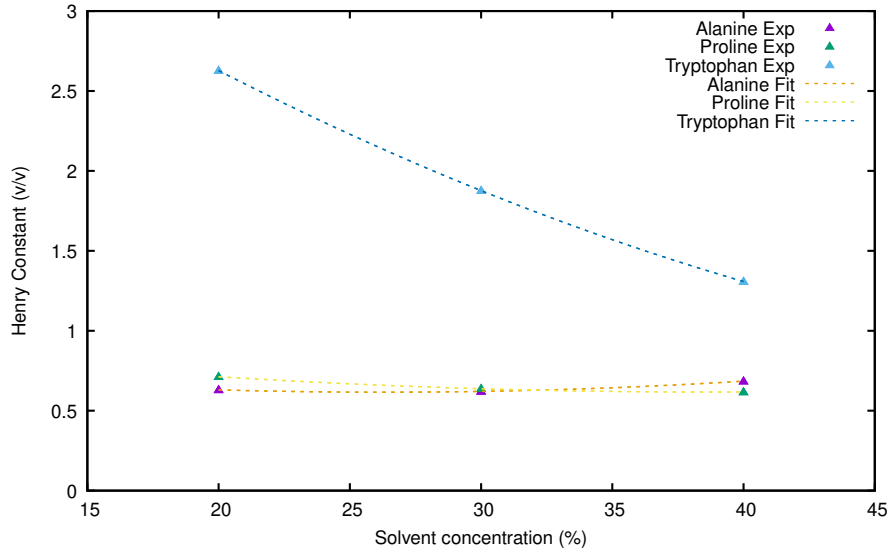

Figure 4.16: Henry constant dependence on the solvent concentration for all components.

Table 4.5: Henry constants determined by the stair experiments.

Component	% MetOH (v/v)	Henry (-)
Alanine	20	0.631
	30	0.621
	40	0.685
Proline	20	0.712
	30	0.637
	40	0.617
Tryptophan	20	2.627
	30	1.877
	40	1.308

MetOH is smaller but it is still linear, which means that is still far from reaching the total capacity of the resin. Since it is linear at 40% MetOH we can deduce that for lower values of MetOH the isotherm remains linear. For that reason there is no need to perform more stair experiments, by performing one pulse injection for each MetOH concentration and obtain the Henry constants. Again, this approach only works because we already know that the isotherms are linear.

In Figure 4.17 we show one of the pulse experiments performed as an example. For each experiment three different injections were made in order to verify the reproducibility of the results. The operation parameters for the experiments are summarized in Table 4.7. The injected volume V_{inj} was achieved by pumping the raw solution for 5 seconds at a flowrate of 1ml/min. Since the HPLC pump as a small inertia, this explains why the absorbance values of the peaks do not match. However, the retention time of each peak matches with a relative standard deviation below 1%.

Table 4.6: Fitting parameters for the Henry data.

Component	% MetOH (v/v)	Henry (-)
Alanine	p_1	0.871
	p_2	-0.019
	p_3	3.7×10^{-4}
Proline	p_1	1.026
	p_2	-0.021
	p_3	2.7×10^{-4}
Tryptophan	p_1	4.674
	p_2	-0.121
	p_3	9.1×10^{-4}

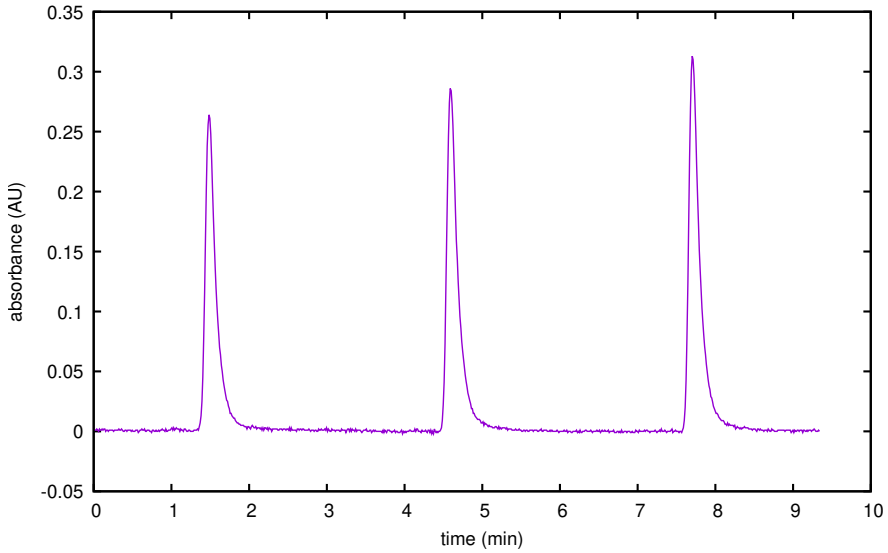

Figure 4.17: Pulse Injection of Proline dissolved in 10% MetOH. The injection volume was 0.083ml, elution time was 3min at a flowrate of 2ml/min. 3 replicas were performed in order to ensure reproducibility.

Table 4.7: Operation parameters for the pulse experiments.

Parameter	Units	Value
Q_E	ml/min	2.000
t_E	min	3.000
V_{inj}	ml	0.083

Since we know the isotherms are linear for the entire range of methanol concentrations that we will use, the pulse experiments will enable us to determine the Henry constant in the following way:

Table 4.8: New Henry constants determined by the pulse experiments.

Component	% MetOH (v/v)	Henry (-)
Alanine	05	0.495
	50	0.465
Proline	10	0.485
	50	0.464
Tryptophan	10	4.199
	50	0.782

$$\begin{aligned} t_{r,i} &= \frac{\epsilon_b + H_i}{Q_E/V_c} \\ V_{r,i} &= (\epsilon_b + H_i)V_c \end{aligned} \tag{4.18}$$

Where $t_{r,i}$ is the time when the peak is obtained, while $V_{r,i}$ is the volume amount when the peak is obtained. However, that is not the real retention time, because the system has an associated dead volume, so:

$$V_{r,i} - V_x = (\epsilon_b + H_i)V_c \tag{4.19}$$

And in terms of the Henry constant:

$$H_i = \frac{V_{r,i} - V_x}{V_c} - \epsilon_b \tag{4.20}$$

We first performed the experiments for 10% (5% for Alanine) and 50% MetOH, with the results presented in Table 4.8.

Right from the start, we can see the big discrepancy regarding the values determined with the stair experiments and the new ones. After investigating why this happened, we found out that the pumps used for the stair experiments were badly calibrated, pumping flowrates lower than requested. This led to an overestimation of the adsorption behavior of all components. With this in mind, we decided to repeat all the MetOH concentrations and determine new Henry constants using pulse experiments only. To make sure we will not overestimate them again, we monitored the fluid weight in the outlet of the system and used that data to determine the real flowrate. In order to convert the mass data into volume, we used the help of a calculator to determine the density of a mixture of water and methanol, depending on the methanol fraction [112]. The temperature used was 15^{o}C. The fractions given by the calculator are in weight and the experiments are in volume fractions, so conversion is necessary. The conversion law used was:

$$W_f = \frac{\rho V_f}{\rho V_f + (1 - V_f)} \tag{4.21}$$

Table 4.9: Converted methanol volume to weight fractions at 15°C. At this temperature, $\rho = 0.7964$.

V_f	W_f
0.05	0.040
0.10	0.081
0.20	0.166
0.25	0.210
0.30	0.254
0.40	0.347
0.50	0.443

Table 4.10: Real Henry constants determined by the pulse experiments.

Component	% MetOH (v/v)	Henry (-)
Alanine	05	0.495
	20	0.477
	30	0.468
	40	0.467
	50	0.465
Proline	10	0.485
	25	0.478
	30	0.473
	50	0.464
Tryptophan	10	4.199
	20	2.159
	30	1.294
	40	0.880
	50	0.782

W_f is the MetOH weight fraction, V_f the MetOH volume fraction and ρ the MetOH density. Table 4.9 reveals the outcome of those calculations. The values in the second column were the ones introduced in the calculator previously mentioned.

Finally, the real Henry constants are presented in Table 4.10.

Again, the difference in results between the pulse and the stair experiments is quite significant. However, the Henry constants taken from the pulse injections present a monotonic behavior, which was the expected behavior.

Taking a closer look at the Henry constants for Alanine and Proline, they are highly similar, almost equal. With this adsorption behavior of both components, it is very unlikely that both components can be separated. With that information we decided to check for other components and try to come up with a different ternary mixture. Our research group has a good experience with Guanosine and Uridine, both aminoacids that adsorb in hydrophobic resins [56, 57, 113]. For that reason, we decided to keep Alanine and test the other two.

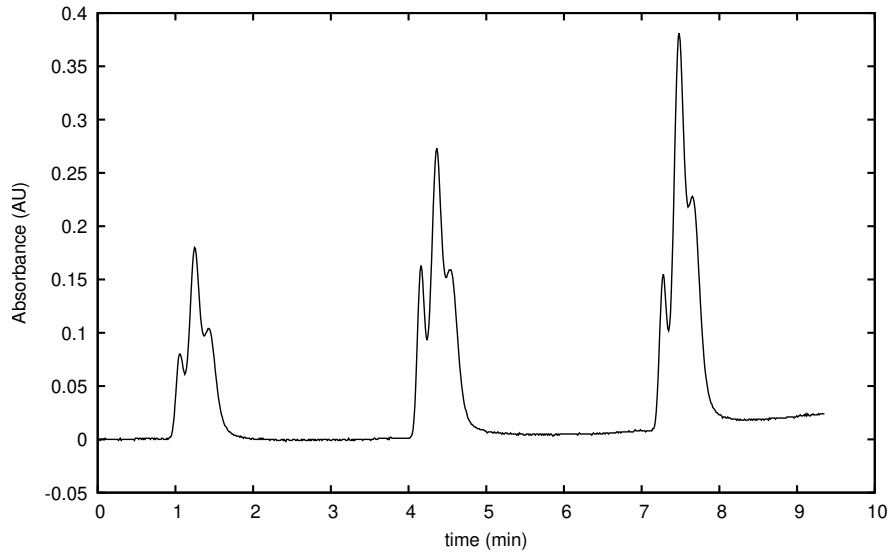

Figure 4.18: Screening of a mix of Alanine, Guanosine and Uridine. The mixture has 0.5g/L of each component, dissolved in 20% MetOH v/v.

We performed a rapid screening by mixing the three components and making one pulse injection to verify if the retention times differ enough to be able to have some separation. Figure 4.18 shows three injections of the ternary mixture dissolved in 20% MetOH, where we can see a partial but clear separation of the three compounds. This is a very attractive mixture, since the overlap makes the separation rather challenging, which makes it a perfect candidate to test our GSSR system. The calculated Henry constants for peaks 1, 2 and 3 are 0.542, 0.742 and 0.916, respectively.

Another note is the fact that none of the previous Henry constants check out with the Henry constant previously determined for Alanine, which is 0.477. With this new information, besides determining the Henry constant dependence on c_0 for Guanosine and Uridine, we also repeated the protocol for Alanine to ensure they are correct. After all the pulse experiments were made, the Henry constants can be seen in Table 4.11.

The fitted model for the data is the following:

$$H_i = \frac{H_i^0}{c_0^{n_i}} \qquad \text{for} \quad i = A, G, U \tag{4.22}$$

A, G and U are shorthand for Alanine, Guanosine and Uridine, H^0 is a reference Henry constant and n is a fitting parameter with no physical meaning. The fitting results are shown in Figures 4.19, 4.20 and 4.21. Figure 4.22 shows all three together, which is seen that the adsorption behavior of Guanosine and Uridine depend on the solvent concentration, while Alanine has no dependence on c_0.

The results of the parameter estimation are shown in Table 4.12.

Table 4.11: Real Henry constants determined by the pulse experiments.

Component	% MetOH (v/v)	Henry (-)
Alanine	10	0.534
	20	0.537
	30	0.534
	40	0.534
	50	0.535
Uridine	10	0.969
	20	0.743
	30	0.638
	40	0.594
	50	0.564
Guanosine	10	1.602
	20	0.959
	30	0.701
	40	0.611
	50	0.574

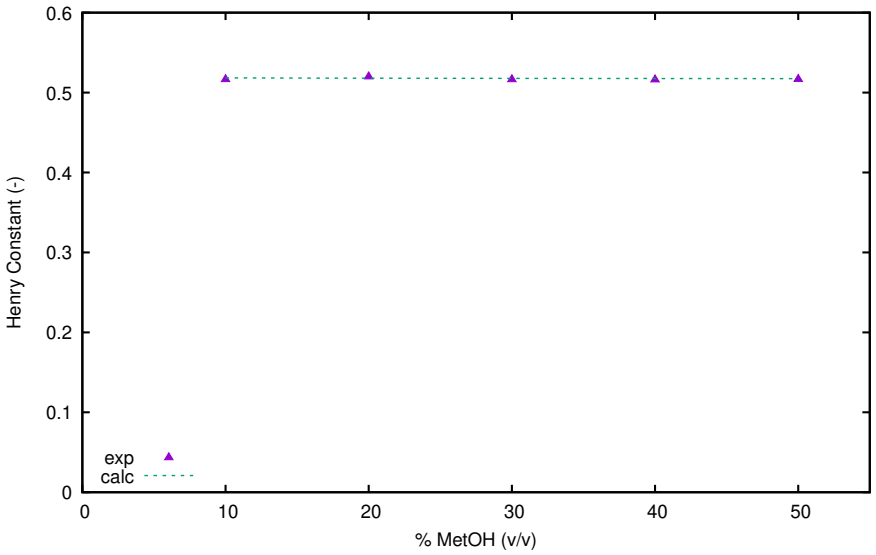

Figure 4.19: Real Henry constant data for Alanine.

4.3 Case Study: the GSSR process

The GSSR process, as described in Section 1.4.8, is a 3-column chromatographic process for central-cut separation of ternary or pseudo-ternary solutions. For that we used the three RESOURCE15RPC columns with a resin volume of 3ml and a bead size of 15μm. They are pre-packed columns and every column piece is made of PEEK, which made the maximum pressure of each column to be around 40bar. We made several pressure drop tests with the three columns in series and a flowrate of 2ml/min would give a pressure of around 25bar. A flowrate of 2.5ml/min would still be below the maximum level, but

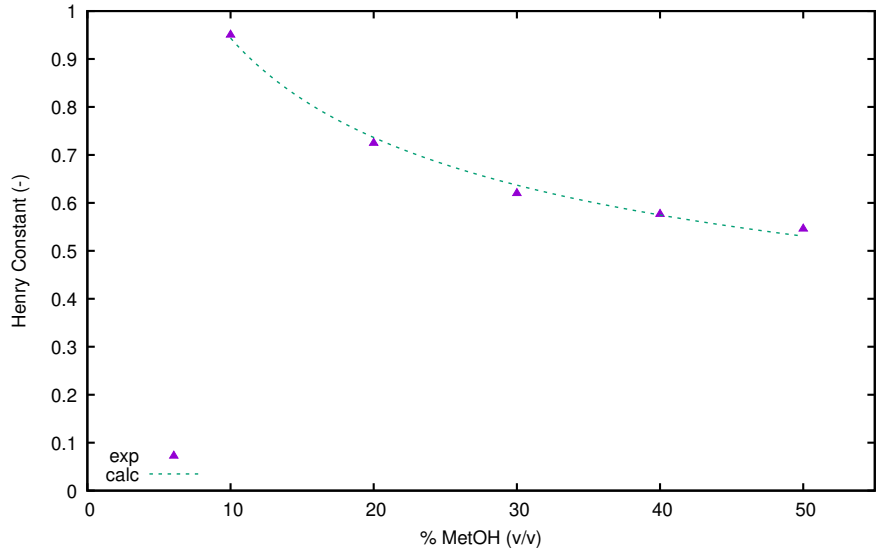

Figure 4.20: Real Henry constant data for Uridine.

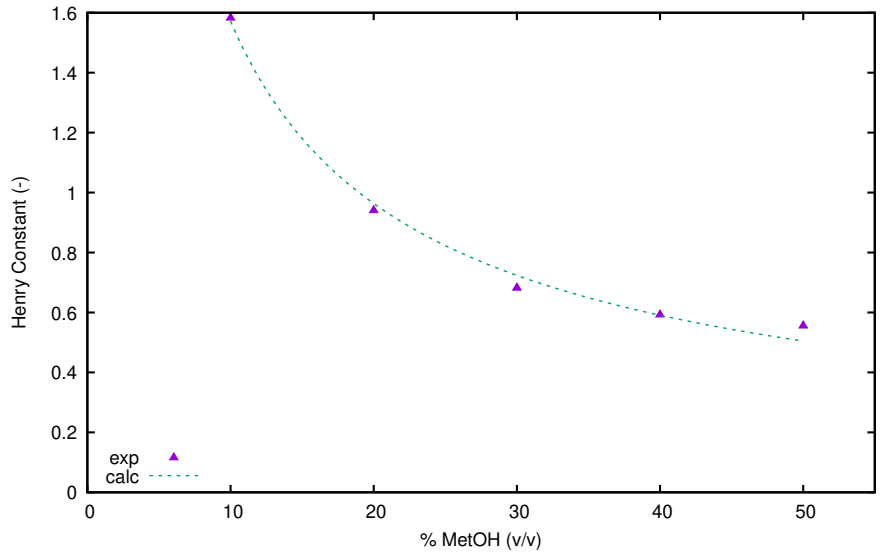

Figure 4.21: Real Henry constant data for Guanosine.

since we were going to perform several experiments, some of them running for 2, 3 hours, we decided to fix the elution flowrate, Q_A, to 2ml/min to ensure a longer lifetime.

After some poorly performed experiments, we accidentally broke the RESOURCE15RPC columns twice, so we decided to switch the columns used, since these were too fragile and any small mistake could cost us another column. We had in our Laboratory several Superformance glass columns with 26mm diameter from Merck and also 200ml of SOURCE30Q resin with a bead size of 30μm from Cytiva, so we packed three Superformance columns with the SOURCE30Q resin with an average bed height of ~63mm. After packing we determined the total porosity, ϵ_t, by using pulse injections of NaCl and following the conductivity through time. The resulting column parameters are shown in Table 4.13. Since the packing was manually made, it is quite difficult to obtain three columns with

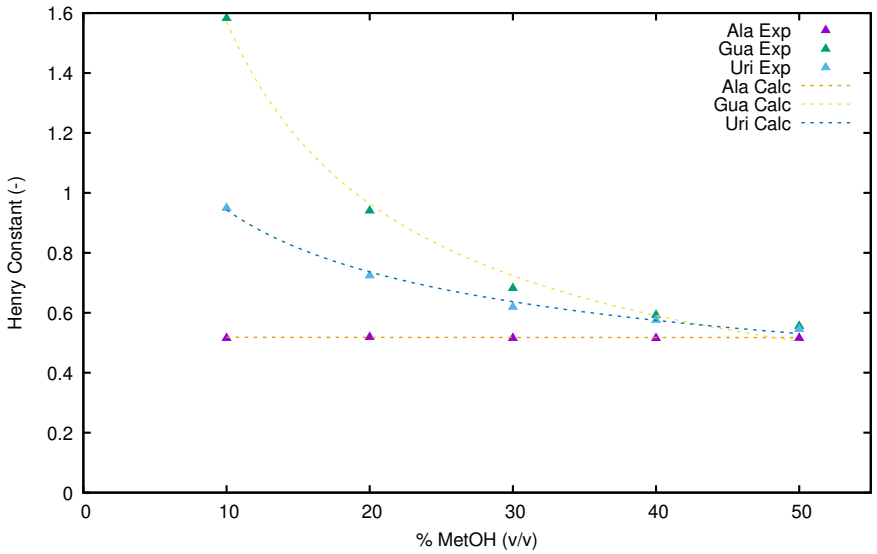

Figure 4.22: Henry constant comparison of all components.

Table 4.12: Fitting parameters for the Henry data.

Component	H^0	n
Alanine	0.534	0.0
Uridine	2.147	0.349
Guanosine	7.831	0.693

Table 4.13: Column parameters after manual packing of SOURCE30Q.

Column	H_c (mm)	ϵ_t (-)
1	61	0.72
2	64	0.71
3	63	0.73
Mean	62.7	0.72
Rel Std Dev (%)	2.44	1.13

the exact same amount of resin, so a relative standard deviation below 5% for both the bed height and the total porosity is very satisfactory. With these results, we can use an average bed volume in our simulations, rather than using different column volumes for each, which would complicate the optimization process.

4.3.1 Determination of the Henry constants

Since we decided to change the resin, we had to determine the Henry constants concerning the adsorption behavior of the chosen components. For that we added tryptophan again to these analyses to verify which component set would be the best to move forward with the GSSR experiments.

The experimental protocol used was the same as described in Section 4.2.3 and the

Table 4.14: Henry constants for each component in the SOURCE30Q resin.

Component	Henry (-)
Alanine	0.00
Uridine	0.30
Guanosine	0.67
Tryptophan	0.83

Table 4.15: Optimized operating conditions using the miscalculated Henry constant set.

Parameter	Value	Units
Q_A	7.0	ml/min
Q_F	7.37	ml/min
α_D	0.68	-
τ_I	4.59	min

results are presented in Table 4.14. As seen, since Alanine does not adsorb in this resin, we decided to follow with the ternary solution of Uridine, Guanosine and Tryptophan.

4.3.2 Preliminary Experiments

As explained in Section 3.3.1, the 8-step GSSR cycle can be simplified to a 4-step cycle, and due to the discrepancy of the Henry constants, it is valid to assume an isocratic elution would be sufficient for an efficient separation. In that case, we simply set $c_A(\tau_I) = c_A(0)$.

The maximum operating pressure of the Superformance columns is 50bar, so we tested the maximum elution flowrate that we can operate. With the three columns in series, we could reach flowrates of 8ml/min with the total pressure not exceeding 20bar. Since we are using 10ml/min pump heads, we decided to fix the elution flowrate to 7ml/min in order to not be too close to the maximum operating flowrate. The optimization was intended to maximize the productivity while having a minimum purity and recovery requirements of 95%.

Despite the Henry constants in Table 4.14 are correct, the first system optimizations were made using a different set of Henry constants, which were 0.02, 0.3 and 0.53 for Uridine, Guanosine and Tryptophan, respectively. This miscalculation was caught only after the first cycle experiments were made, that is why the first set of optimized operating conditions are not the correct ones. Nevertheless, the sole purpose of the first GSSR experiments was to verify the integrity of our experimental set-up and also estimate the Péclet constants of each component, that is why we did not repeat said experiments. The operating conditions after optimization using the miscalculated Henry constants are presented in Table 4.15.

For each component, two runs were performed, each with a different switching interval and feed step, to test if the system integrity holds for different cases. For each experiment

Table 4.16: Estimated Péclet constants for each component.

Component	Péclet (-)
Uridine	500.0
Guanosine	800.0
Tryptophan	800.0

5 cycles were run, starting with all columns completely empty from solute. We followed the concentration profiles in the outlet of column 3, which is where the purified product exits the system. The Péclet was estimated by trial and error using those experiments. In Table 4.16 we can see the Péclet constants determined for each component by adjusting the simulated data with the experimental data by trial and error.

Figures 4.23, 4.24 and 4.25 represent the outlet concentration profiles of column 3 for Uridine, Guanosine and Tryptophan, respectively. Even though the Henry constants were wrong, both Uridine and Guanosine showed in the column 3 outlet, which was not the case for Tryptophan. This is why for Tryptophan there is no experiment with $\tau_I = 4.59$, simply because with that switching interval, Tryptophan was completely rejected from the product stream, exiting the system from columns 1 and 2.

The dynamically simulated curves were performed using the *gPROMS* software, version 4.0. The dimensionless time θ is the total experimental time divided by the switching interval:

$$\theta(t) = \frac{t}{\tau_I} \tag{4.23}$$

These preliminary experiments were used to both determine the Péclet constants and test the physical integrity of the experimental set-up. There is a high concordance between the simulated and the experimental curves, which proves the robustness of both the physical and mathematical systems created in-house.

4.3.3 Separation Experiments

We packed three Superformance glass columns because two of our RESOURCE15RPC broke during the GSSR experiments. Switching to the new columns, packing them and rerunning all the experiments with bigger columns and a different resin cut us a big chunk of the necessary time to perform real separation experiments. To add up, some issue with the installation appeared after we finished the preliminary experiments, which made us lose reproducibility in the experiments after. Due to time constraints we did not have time to fix the issue and perform the separation experiments. Regardless of everything that happened, the experiments we did have helped us verify that our models were in line with reality, so we were able to perform a study of what the separation experiments would look like.

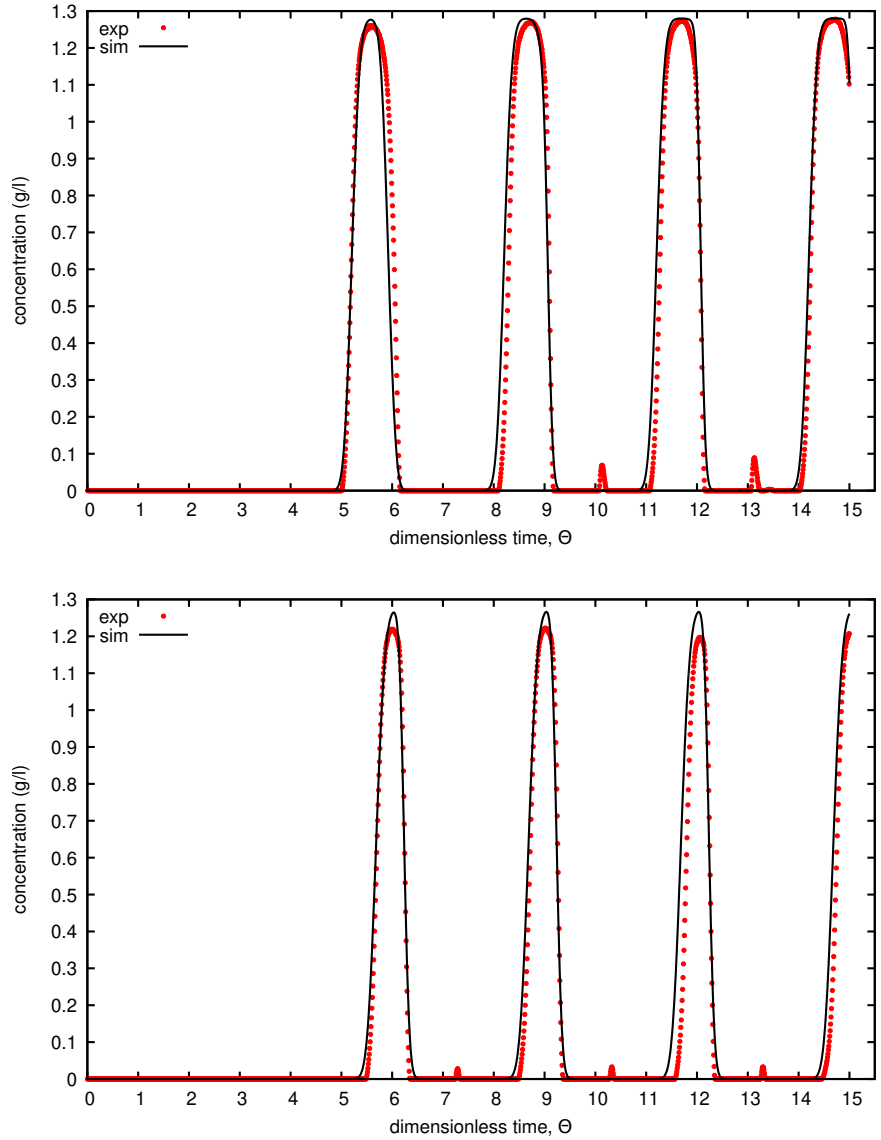

Figure 4.23: Uridine concentration profiles in the column 3 outlet after running 5 GSSR cycles. The experiment represented in the top figure was run using $\tau_I = 4.59$, while for the bottom one was used $\tau_I = 4.0$. The other operating conditions were the ones in Table 4.15.

With the preliminary experiments we could determine the Péclet constants of each component, so we re-optimized the process in order to obtain the new operating conditions. After the optimization, we obtained the parameters presented in Table 4.17.

After the re-optimization, we ran 100 simulated cycles to determine when the cyclic steady state starts. Since our minimum requirements for purity and recovery were 95% for each and our optimizations only needed to fulfill them by an infinitesimal margin, we assumed that the cycle when the process reaches the CSS is when both recovery and purity reach the 95% mark. This can be seen in Figure 4.26. The recovery takes significantly more cycles than the purity to reach its CSS value. This is due to the fact that the recovery is

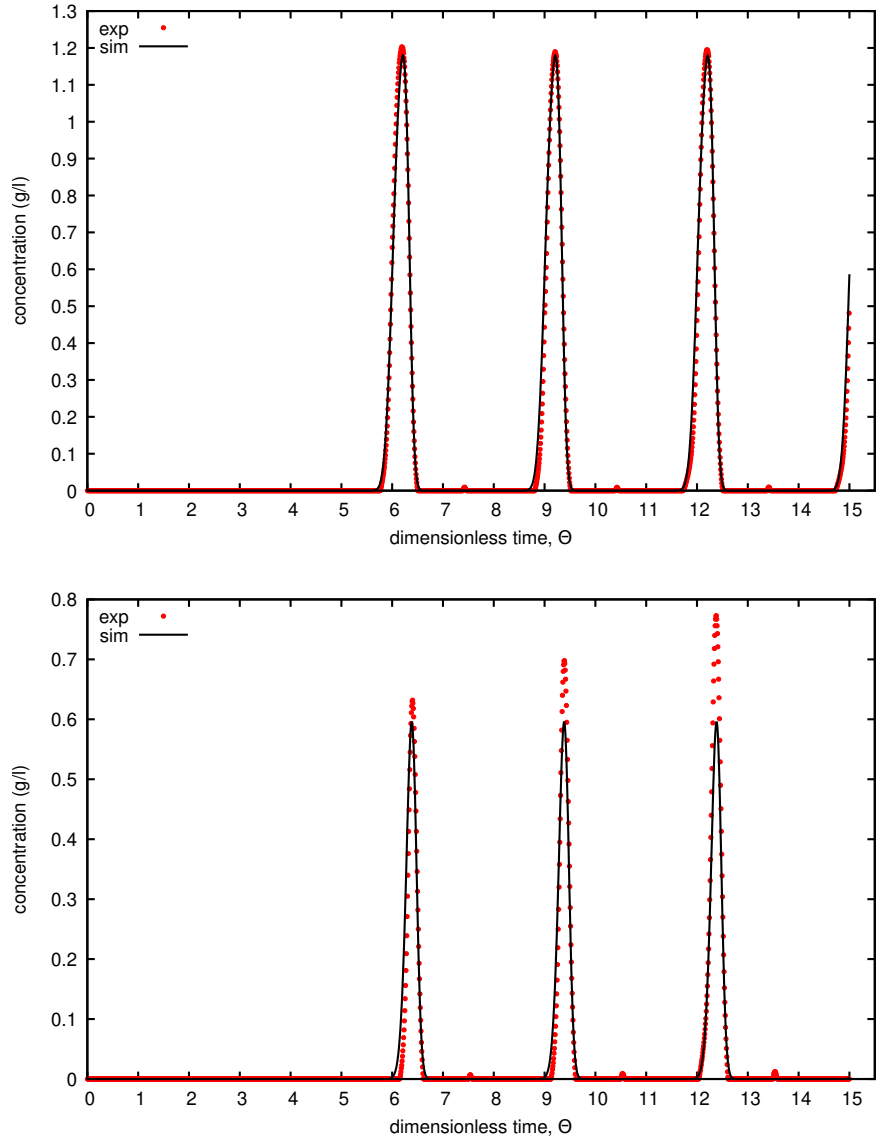

Figure 4.24: Guanosine concentration profiles in the column 3 outlet after running 5 GSSR cycles. The experiment represented in the top figure was run using $\tau_I = 5.0$ and $\alpha_D = 0.5$, while for the bottom one was used $\tau_I = 4.6$ and $\alpha_D = 0.7$. The other operating conditions were the ones in Table 4.15.

way more sensitive to changes, such as component or gradient concentration, as shown in Section 3.3. In the end, it takes 71 cycles for the process to reach CSS, which can be converted to hours:

$$t_{CSS} = \frac{71 \times 3 \times \tau_I}{60} = 23.96h \tag{4.24}$$

It is of our interest to also evaluate the concentration ratio, CR, and the solvent consumption ratio, SCR, as well as the productivity, $\overline{Q_F}$, of our system. The results are in Table 4.18.

Despite not being able to present experiments using these operating conditions, we

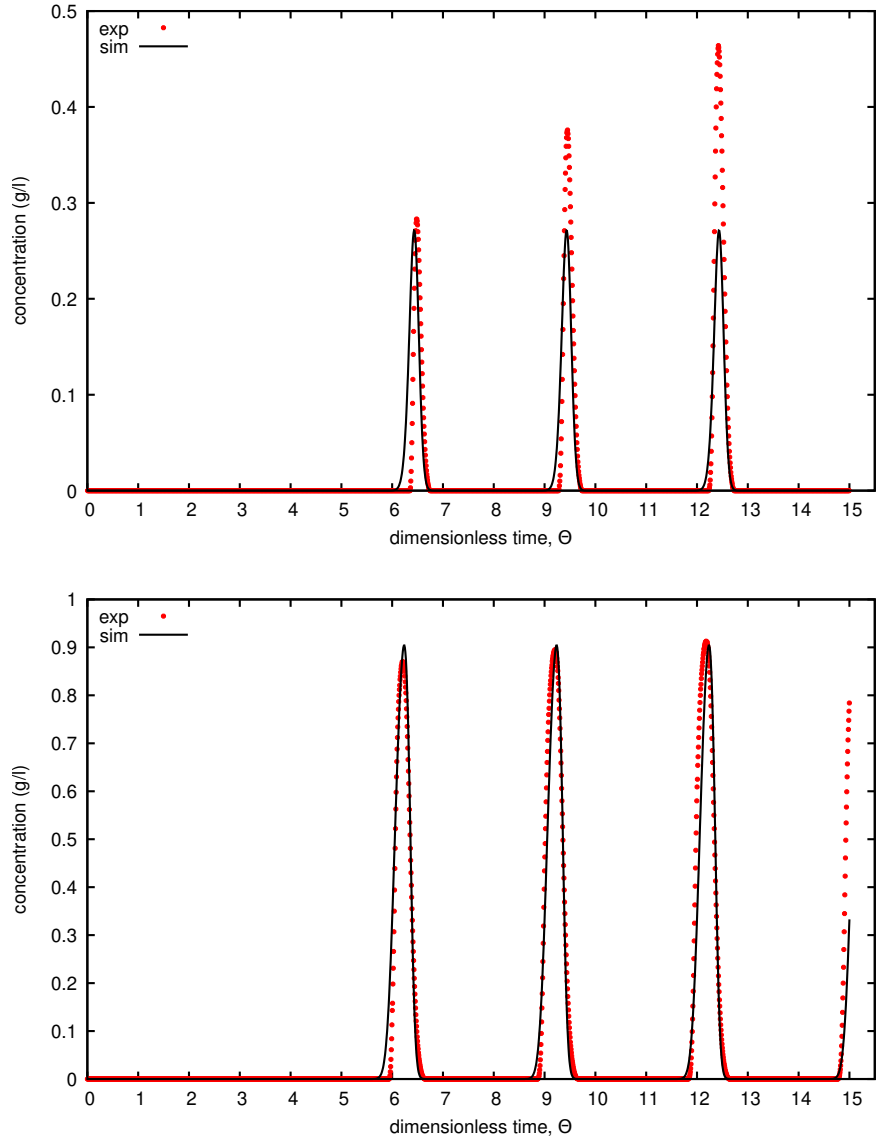

Figure 4.25: Tryptophan concentration profiles in the column 3 outlet after running 5 GSSR cycles. The experiment represented in the top figure was run using $\tau_I = 5.0$, while for the bottom one was used $\tau_I = 5.5$. The other operating conditions were the ones in Table 4.15.

are able to predict, with a good degree of confidence, the results of said experiments, thanks to the preliminaries that helped us estimate some parameters and confirm that our mathematical system was correctly built and is able to predict actual experiments.

Table 4.17: Newly optimized operating conditions with the new Péclet constants.

Parameter	Value	Units
Q_A	7.0	ml/min
Q_F	6.73	ml/min
α_D	0.32	-
τ_I	6.75	min

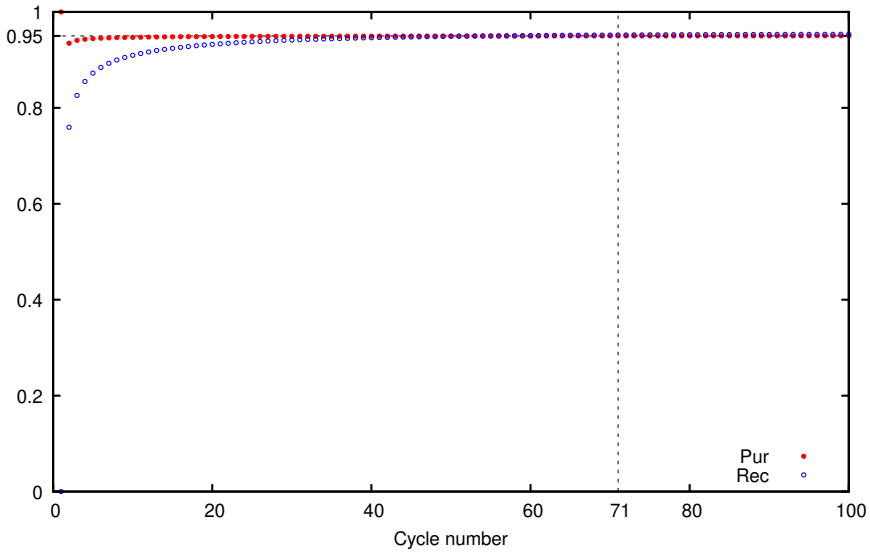

Figure 4.26: Purity and Recovery at each cycle. The system reached the CSS at the 71st cycle.

Table 4.18: Key Performance Indicators of the process with the newly optimized parameters.

Parameter	Value	Units
CR	0.914	-
SCR	9.796	-
$\overline{Q_F}$	0.717	ml/min

5

Conclusions and Future Remarks

The GSSR is an 8-step process that was created to perform central-cut separation of ternary or pseudo-ternary mixtures. The purpose of this work was to create numerical tools to optimize and simulate the GSSR process that would work for any separation system. To validate them, we assembled an experimental set-up and created software to control and automate the equipments in-house in order to implement any pathway required. Not only our set-up can replicate GSSR cycles, but can also implement any process that requires 3 or less columns, such as the 3-col MCSGP, 3-col PCC, 2-col SMB, etc.

In order to improve it, we first used a genetic algorithm. The GA was able to significantly improve the productivity around 3-fold, the concentration ratio around 2-fold and the solvent consumption ratio around 10-fold and was able to reduce the number of steps from 8 to 6, which simplifies the problem.

Secondly, we used a deterministic algorithm, the interior-point solver *IpOpt*. The best optimization approach greatly improved the results from the GA optimizations, increasing the productivity around 4-fold, increasing the concentration ratio around 5-fold and decreasing the solvent consumption around 2-fold. Moreover, the process was able to be simplified from 6 to 4 steps, while eliminating all isocratic solvent streams, leaving only the gradient stream.

Finally, experimental validation of a ternary separation was performed by using Uridine, Guanosine and Tryptophan as the ternary mixture. The separation experiments were compared to simulations using the optimized decision variables and they were a match, which means our optimizations are reliable and our simulations can predict the outcome of experimental results.

In the future, it would be interesting to perform experiments that reach the CSS. Furthermore, further testing our set-up by performing the same separation using the 3-col MCSGP or any other 3 column process would be of interest as well, also with actual peptide mixtures.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. i).

[2] R. D. Noble and P. A. Terry. "Adsorption". In: *Principles of Chemical Separations with Environmental Applications*. Cambridge Series in Chemical Engineering. Cambridge University Press, 2004, pp. 182–213. DOI: 10.1017/CBO9780511616594.008 (cit. on p. 1).

[3] *Adsorption vs Absorption – Differences and Examples*. https://sciencenotes.org/adsorption-vs-absorption-differences-and-examples/. Accessed: 2022-06-06 (cit. on p. 2).

[4] M. Pan, H. M. Omar, and S. Rohani. "Application of Nanosize Zeolite Molecular Sieves for Medical Oxygen Concentration". In: *PubMed* 7 (July 2017). DOI: 10.3390/nano7080195 (cit. on p. 1).

[5] A. K. Bajpai and M. Rajpoot. "Adsorption Techniques - A Review". In: *Journal of Scientific & Industrial Research* 58 (1999), pp. 844–860 (cit. on p. 2).

[6] A. El-Baz et al. "Adsorption technique for pollutants removal; current new trends and future challenges -A Review". In: (Jan. 2021), pp. 1–24. DOI: 10.21608/eijest.2020.45536.1015 (cit. on p. 2).

[7] W. S. Chai et al. "A review on conventional and novel materials towards heavy metal adsorption in wastewater treatment application". In: *Journal of Cleaner Production* 296 (2021), p. 126589. ISSN: 0959-6526. DOI: https://doi.org/10.1016/j.jclepro.2021.126589. URL: https://www.sciencedirect.com/science/article/pii/S095965262100809X (cit. on p. 2).

[8] J. T. Cheng et al. "A Review of Caffeine Adsorption Studies onto Various Types of Adsorbents". In: *The Scientific World Journal* 2021 (July 2021). DOI: 10.1155/2021/9998924 (cit. on p. 2).

[9] F. Poorsharbaf Ghavi, F. Raouf, and A. Dadvand Koohi. "A Review on Diclofenac Removal from Aqueous Solution, Emphasizing on Adsorption Method". In: *Iranian Journal of Chemistry and Chemical Engineering (IJCCE)* 39.1 (2020), pp. 141–154. ISSN: 1021-9986. DOI: 10.30492/ijcce.2020.33337. eprint: http://www.ijcce.ac.ir/article_33337_ae1f2fbdc3b961d1777214d48b7bfb96.pdf. URL: http://www.ijcce.ac.ir/article_33337.html (cit. on p. 2).

[10] J. Farradane. "History of Chromatography". In: *Nature* (1951) (cit. on p. 5).

[11] E. Robens and S. A. A. Jayaweera. "Early History of Adsorption Measurements". In: *Adsorption Science & Technology* 32.6 (2014), pp. 425–442. DOI: 10.1260/0263-6174.32.6.425 (cit. on p. 5).

[12] *What Is HPLC (High Performance Liquid Chromatography)?* https://www.waters.com/waters/es_ES/HPLC---High-Performance-Liquid-Chromatography-Explained/nav.htm?cid=10048919&locale=es_ES. Accessed: 2021-01-19 (cit. on p. 5).

[13] I. Smith et al. "Inhaler Devices: What Remains to be Done?" In: *Journal of aerosol medicine and pulmonary drug delivery* 23 Suppl 2 (Dec. 2010), S25–37. DOI: 10.1089/jamp.2010.0853 (cit. on p. 6).

[14] *Why Particle Size Distribution is Important in Chromatographic Resins*. https://www.chromatographytoday.com/news/purification/66/purolite/why-particle-size-distribution-is-important-in-chromatographic-resins/49992. Accessed: 2022-04-25 (cit. on p. 6).

[15] *What is Jetting Technology*. https://www.purolite.com/index/Company/About/manufacturing/Jetting-Technology. Accessed: 2022-04-25 (cit. on p. 6).

[16] J. Nimmo. "Porosity and Pore Size Distribution". In: *Reference Module in Earth Systems and Environmental Sciences* (Dec. 2013). DOI: 10.1016/B978-0-12-409548-9.05265-9 (cit. on p. 6).

[17] C. F. Poole. "Chapter 4 - The Column in Liquid Chromatography". In: *The Essence of Chromatography*. Ed. by C. F. Poole. Amsterdam: Elsevier Science, 2003, pp. 267–429. ISBN: 978-0-444-50198-1. DOI: https://doi.org/10.1016/B978-044450198-1/50017-3. URL: https://www.sciencedirect.com/science/article/pii/B9780444501981500173 (cit. on p. 7).

[18] J. Bear and M. Corapcioglu, eds. *Advances in Transport Phenomena in Porous Media*. Springer, Dordrecht, 1987. ISBN: 978-94-009-3625-6. DOI: https://doi.org/10.1007/978-94-009-3625-6 (cit. on p. 8).

[19] G. B. Arfken et al. "Chapter 16 - FLUID MECHANICS". In: *University Physics*. Ed. by G. B. Arfken et al. Academic Press, 1984, pp. 306–325. ISBN: 978-0-12-059860-1. DOI: https://doi.org/10.1016/B978-0-12-059860-1.50021-7. URL: https://www.sciencedirect.com/science/article/pii/B978012059860150021 7 (cit. on p. 8).

[20] *Polar and Nonpolar Molecules*. https://sciencenotes.org/polar-and-nonpolar-molecules/. Accessed: 2022-06-07 (cit. on p. 11).

[21] Z. El Rassi. "Chapter 2 - Reversed-phase and hydrophobic interaction chromatography of carbohydrates and glycoconjugates". In: *Carbohydrate Analysis by Modern Liquid Phase Separation Techniques (Second Edition)*. Ed. by Z. El Rassi. 2nd. Amsterdam: Elsevier, 2021, pp. 35–124. ISBN: 978-0-12-821447-3. DOI: https://doi.org/10.1016/B978-0-12-821447-3.00017-2. URL: https://www.sciencedirect.com/science/article/pii/B9780128214473000172 (cit. on p. 11).

[22] L. G. Berruex and R. Freitag. "Separation and Purification of Biochemicals". In: *Encyclopedia of Physical Science and Technology (Third Edition)*. Ed. by R. A. Meyers. 3rd. New York: Academic Press, 2003, pp. 651–673. ISBN: 978-0-12-227410-7. DOI: https://doi.org/10.1016/B0-12-227410-5/00683-9. URL: https://www.sciencedirect.com/science/article/pii/B0122274105006839 (cit. on pp. 11–13).

[23] J. T. McCue. "Chapter 25 - Theory and Use of Hydrophobic Interaction Chromatography in Protein Purification Applications". In: *Guide to Protein Purification, 2nd Edition*. Ed. by R. R. Burgess and M. P. Deutscher. Vol. 463. Methods in Enzymology. Academic Press, 2009, pp. 405–414. DOI: https://doi.org/10.1016/S0076-6879(09)63025-1. URL: https://www.sciencedirect.com/science/article/pii/S0076687909630251 (cit. on p. 11).

[24] J. Fausnaugh, L. Kennedy, and F. Regnier. "Comparison of hydrophobic-interaction and reversed-phase chromatography of proteins". In: *Journal of Chromatography A* 317 (1984). EIGHT INTERNATIONAL SYMPOSIUM ON COLUMN LIQUID CHROMATOGRAPHY PART II, pp. 141–155. ISSN: 0021-9673. DOI: https://doi.org/10.1016/S0021-9673(01)91654-1. URL: https://www.sciencedirect.com/science/article/pii/S0021967301916541 (cit. on p. 11).

[25] N. W. Sadiq and D. Beauchemin. "Chapter 4 - Liquid chromatography". In: *Sample Introduction Systems in ICPMS and ICPOES*. Ed. by D. Beauchemin. Amsterdam: Elsevier, 2020, pp. 213–254. ISBN: 978-0-444-59482-2. DOI: https://doi.org/10.1016/B978-0-444-59482-2.00004-X. URL: https://www.sciencedirect.com/science/article/pii/B978044459482200004X (cit. on pp. 12, 13).

[26] G. L. Mayers and C. J. van Oss. "Affinity Chromatography". In: *Encyclopedia of Immunology (Second Edition)*. Ed. by P. J. Delves. Second Edition. Oxford: Elsevier, 1998, pp. 47–49. ISBN: 978-0-12-226765-9. DOI: https://doi.org/10.1006/rwei.1

999.0012. URL: https://www.sciencedirect.com/science/article/pii/B012226765600013X (cit. on p. 13).

[27] Y.-C. Chen and M.-K. Yeh. "Introductory Chapter: Biopharmaceuticals". In: *Biopharmaceuticals*. Ed. by M.-K. Yeh and Y.-C. Chen. Rijeka: IntechOpen, 2018. Chap. 1. DOI: 10.5772/intechopen.79194. URL: https://doi.org/10.5772/intechopen.79194 (cit. on p. 14).

[28] P. Nestola et al. "Improved virus purification processes for vaccines and gene therapy". In: *Biotechnology and Bioengineering* 112.5 (2015), pp. 843–857. DOI: https://doi.org/10.1002/bit.25545. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bit.25545. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/bit.25545 (cit. on pp. 14, 15).

[29] S. S. Ozturk and W.-S. Hu, eds. *Cell Culture Technology for Pharmaceutical and Cell-based Therapies*. 1st. CRC Press, 2005. DOI: https://doi.org/10.1201/9780849351068 (cit. on p. 15).

[30] M. W. Wolf and U. Reichl. "Downstream processing of cell culture-derived virus particles". In: *Expert Review of Vaccines* 10.10 (2011). PMID: 21988309, pp. 1451–1475. DOI: 10.1586/erv.11.111. eprint: https://doi.org/10.1586/erv.11.111. URL: https://doi.org/10.1586/erv.11.111 (cit. on p. 15).

[31] N. Dejardin, M. Alamir, and J.-P. Corriou. "Model predictive control of a simulated moving bed". In: Jan. 2005 (cit. on p. 18).

[32] C. D. Luca et al. "Boosting the Purification Process of Biopharmaceuticals by Means of Continuous Chromatography". In: *LCGC* 6.28 (June 2020), pp. 30–34 (cit. on p. 18).

[33] A. Rodrigues. *Simulated Moving Bed Technology: Principles, Design and Process Applications*. 1st. Butterworth-Heinemann, Dec. 2015 (cit. on pp. 18, 19, 34).

[34] P. S. Gomes, M. Minceva, and A. E. Rodrigues. "Simulated moving bed technology: old and new". In: *Adsorption* 12 (2006), pp. 375–392. DOI: 10.1007/s10450-006-0566-9 (cit. on p. 19).

[35] D. B. Broughton and C. G. Gerhold. "Continuous sorption process employing fixed bed of sorbent and moving inlets and outlets". U.S. pat. 660790. May 23, 1961 (cit. on p. 19).

[36] J. M. Ferraro et al. "Process for production of paraxylene". Worldwide pat. 97/27162. July 31, 1997 (cit. on p. 19).

[37] M. Minceva and A. E. Rodrigues. "Understanding and revamping of industrial scale SMB units for p-xylene separation". In: *AIChE Journal* 53.1 (2007), pp. 138–149. DOI: https://doi.org/10.1002/aic.11062. eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.11062. URL: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.11062 (cit. on p. 19).

[38] M. S. P. Silva, A. E. Rodrigues, and J. P. B. Mota. "Modeling and simulation of an industrial-scale parex process". In: *AIChE Journal* 61.4 (2015), pp. 1345–1363. DOI: https://doi.org/10.1002/aic.14732. eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.14732. URL: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.14732 (cit. on p. 19).

[39] *The Theory and Advantages of Simulated Moving Bed (SMB) Chromatography*. https://www.azom.com/article.aspx?ArticleID=14708. Accessed: 2022-06-06 (cit. on p. 20).

[40] S. Vignesh et al. "Optimal Strategies for Transitions in Simulated Moving Bed Chromatography". In: *Computers & Chemical Engineering* 84 (Aug. 2015). DOI: 10.1016/j.compchemeng.2015.08.004 (cit. on p. 20).

[41] J. Araújo and R. Rodrigues. "A Streamlined Two-Column SMB Process for Chiral Separation". In: Nov. 2008 (cit. on p. 20).

[42] L. Aumann and M. Morbidelli. "A continuous multicolumn countercurrent solvent gradient purification (MCSGP) process". In: *Biotechnology and Bioengineering* 98.5 (2007), pp. 1043–1055. DOI: https://doi.org/10.1002/bit.21527. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bit.21527. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/bit.21527 (cit. on p. 21).

[43] L. Aumann and M. Morbidelli. "A semicontinuous 3-column countercurrent solvent gradient purification (MCSGP) process". In: *Biotechnology and Bioengineering* 99.3 (2008), pp. 728–733. DOI: https://doi.org/10.1002/bit.21585. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bit.21585. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/bit.21585 (cit. on p. 22).

[44] F. Steinebach et al. "Experimental design of a twin-column countercurrent gradient purification process". In: *Journal of Chromatography A* 1492 (2017), pp. 19–26. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2017.02.049. URL: https://www.sciencedirect.com/science/article/pii/S002196731730287X (cit. on p. 23).

[45] M. Bryntesson, M. Hall, and K. Lacki. "Chromatography Method". U.S. pat. 7901581. Mar. 8, 2011 (cit. on p. 24).

[46] M. Angarita et al. "Twin-column CaptureSMB: A novel cyclic process for protein A affinity chromatography". In: *Journal of Chromatography A* 1389 (2015), pp. 85–95. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2015.02.046. URL: https://www.sciencedirect.com/science/article/pii/S0021967315003039 (cit. on p. 25).

[47] R. J. Silva et al. "A new multicolumn, open-loop process for center-cut separation by solvent-gradient chromatography". In: *Journal of Chromatography A* 1217.52 (2010), pp. 8257–8269. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2010.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0021967310015372 (cit. on pp. 26, 75, 79, 92, 94).

[48] J. W. Lee. "Expanding Simulated Moving Bed Chromatography into Ternary Separations in Analogy to Dividing Wall Column Distillation". In: *Industrial & Engineering Chemistry Research* 59.20 (2020), pp. 9619–9628. DOI: 10.1021/acs.iecr.0c00572. eprint: https://doi.org/10.1021/acs.iecr.0c00572. URL: https://doi.org/10.1021/acs.iecr.0c00572 (cit. on p. 27).

[49] J. Hur and P. Wankat. "New Design of Simulated Moving Bed (SMB) for Ternary Separations". In: *Industrial & Engineering Chemistry Research - IND ENG CHEM RES* 44 (Feb. 2005). DOI: 10.1021/ie040164e (cit. on p. 27).

[50] R. Godawat et al. "Periodic counter-current chromatography – design and operational considerations for integrated and continuous purification of proteins". In: *Biotechnology Journal* 7.12 (2012), pp. 1496–1508. DOI: https://doi.org/10.1002/biot.201200068. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/biot.201200068. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/biot.201200068 (cit. on p. 27).

[51] J. W. Lee and P. C. Wankat. "Design of pseudo-simulated moving bed process with multi-objective optimization for the separation of a ternary mixture: Linear isotherms". In: *Journal of Chromatography A* 1217.20 (2010), pp. 3418–3426. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2010.03.020. URL: https://www.sciencedirect.com/science/article/pii/S0021967310003845 (cit. on p. 27).

[52] R. J. Silva, R. C. Rodrigues, and J. P. Mota. "Relay simulated moving bed chromatography: Concept and design criteria". In: *Journal of Chromatography A* 1260 (2012), pp. 132–142. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2012.08.076. URL: https://www.sciencedirect.com/science/article/pii/S0021967312013106 (cit. on p. 27).

[53] O. LUDEMANN-HOMBOURGER, R. M. NICOUD, and M. BAILLY. "The "VARICOL" Process: A New Multicolumn Continuous Chromatographic Process". In: *Separation Science and Technology* 35.12 (2000), pp. 1829–1862. DOI: 10.1081/SS-100100622. eprint: https://doi.org/10.1081/SS-100100622. URL: https://doi.org/10.1081/SS-100100622 (cit. on p. 27).

[54] M. Holzer, H. Osuna-Sanchez, and L. David. *Multicolumn Chromatography*. Sept. 2008. URL: https://bioprocessintl.com/downstream-processing/chromatography/multicolumn-chromatography-183211/ (cit. on p. 27).

[55] D. Antos and A. Seidel-Morgenstern. "Continuous step gradient elution for preparative separations". In: *Separation Science and Technology* 37.7 (2002), pp. 1469–1487. DOI: 10.1081/SS-120002732. eprint: https://doi.org/10.1081/SS-120002732. URL: https://doi.org/10.1081/SS-120002732 (cit. on p. 27).

[56] A. S. Chibério et al. "Batch chromatography with recycle lag. I—Concept and design". In: *Journal of Chromatography A* 1623 (2020), p. 461199. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2020.461199. URL: https://www.sciencedirect.com/science/article/pii/S0021967320304635 (cit. on pp. 27, 121).

[57] A. S. Chibério et al. "Batch chromatography with recycle lag. II—Physical realization and experimental validation". In: *Journal of Chromatography A* 1623 (2020), p. 461211. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2020.461211. URL: https://www.sciencedirect.com/science/article/pii/S002196732030474X (cit. on pp. 27, 121).

[58] V. Warikoo et al. "Integrated continuous production of recombinant therapeutic proteins". In: *Biotechnology and Bioengineering* 109.12 (2012), pp. 3018–3029. DOI: https://doi.org/10.1002/bit.24584. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bit.24584. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/bit.24584 (cit. on p. 28).

[59] D. Baur et al. "Comparison of batch and continuous multi-column protein A capture processes by optimal design". In: *Biotechnology Journal* 11.7 (2016), pp. 920–931. DOI: https://doi.org/10.1002/biot.201500481. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/biot.201500481. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/biot.201500481 (cit. on p. 28).

[60] J. Gomis-Fons, N. Andersson, and B. Nilsson. "Optimization study on periodic counter-current chromatography integrated in a monoclonal antibody downstream process". In: *Journal of Chromatography A* 1621 (2020), p. 461055. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2020.461055. URL: https://www.sciencedirect.com/science/article/pii/S0021967320302673 (cit. on p. 28).

[61] C. Shi et al. "Model-based process development of continuous chromatography for antibody capture: A case study with twin-column system". In: *Journal of Chromatography A* 1619 (2020), p. 460936. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2020.460936. URL: https://www.sciencedirect.com/science/article/pii/S0021967320301230 (cit. on p. 28).

[62] D. DeVault. "The Theory of Chromatography". In: *Journal of the American Chemical Society* 65.4 (1943), pp. 532–540. DOI: 10.1021/ja01244a011. eprint: https://doi.org/10.1021/ja01244a011. URL: https://doi.org/10.1021/ja01244a011 (cit. on p. 34).

[63] A. Fick. "Ueber Diffusion". In: *Annalen der Physik* 170.1 (1855), pp. 59–86. DOI: https://doi.org/10.1002/andp.18551700105. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/andp.18551700105. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/andp.18551700105 (cit. on p. 35).

[64] R. M. Flores. "Chapter 4 - Coalification, Gasification, and Gas Storage". In: *Coal and Coalbed Gas*. Ed. by R. M. Flores. Boston: Elsevier, 2014, pp. 167–233. ISBN: 978-0-12-396972-9. DOI: https://doi.org/10.1016/B978-0-12-396972-9.00004-5. URL: https://www.sciencedirect.com/science/article/pii/B9780123969729000045 (cit. on p. 35).

[65] S. Patankar. *Numerical Heat Transfer and Fluid Flow*. 1st. CRC Press, 1980. DOI: \url{https://doi.org/10.1201/9781482234213} (cit. on p. 36).

[66] C. K. Lee et al. "Mass transfer effects in isocratic non-linear elution chromatography". In: *Journal of Chromatography A* 484 (1989), pp. 29–59. ISSN: 0021-9673. DOI: https://doi.org/10.1016/S0021-9673(01)88961-5. URL: https://www.sciencedirect.com/science/article/pii/S0021967301889615 (cit. on p. 37).

[67] G. Guiochon, A. Felinger, and D. G. Shirazi. *Fundamentals of Preparative and Nonlinear Chromatography*. 2nd. Elsevier, Feb. 2006 (cit. on p. 39).

[68] J. van Deemter, F. Zuiderweg, and A. Klinkenberg. "Longitudinal diffusion and resistance to mass transfer as causes of nonideality in chromatography". In: *Chemical Engineering Science* 5.6 (1956), pp. 271–289. ISSN: 0009-2509. DOI: https://doi.org/10.1016/0009-2509(56)80003-1. URL: https://www.sciencedirect.com/science/article/pii/0009250956800031 (cit. on p. 39).

[69] D. M. Ruthven. *Principles of Adsorption and Adsorption Processes*. 1st. Wiley-Interscience, 1984 (cit. on p. 39).

[70] DrSmith321. *Shape of moisture isotherm types. Generally % adsorbed or absorbed by weight on y (vertical) axis and Relative Humidity on x (horizontal) axis. Sorption normally falls into one of these five shapes, represented by Type.* Wikimedia Commons, Dec. 2019 (cit. on p. 42).

[71] H. Erbil. *Surface Chemistry of Solid and Liquid Interfaces*. Wiley, 2006. ISBN: 9781405119689. URL: https://books.google.pt/books?id=oV1HwpVUPDYC (cit. on p. 42).

[72] I. Langmuir. "THE CONSTITUTION AND FUNDAMENTAL PROPERTIES OF SOLIDS AND LIQUIDS. PART I. SOLIDS." In: *Journal of the American Chemical Society* 38.11 (1916), pp. 2221–2295. DOI: 10.1021/ja02268a002. eprint: https://doi.org/10.1021/ja02268a002. URL: https://doi.org/10.1021/ja02268a002 (cit. on p. 43).

[73] H. R. P. "Kapillarchemie, Eine Darstellung der Chemie der Kolloide und verwandter Gebiete". In: *Nature* 85 (Feb. 1911). DOI: 10.1038/085534a0 (cit. on p. 44).

[74] N. Ayawei, A. N. Ebelegi, and D. Wankasi. "Modelling and Interpretation of Adsorption Isotherms". In: *Journal of Chemistry* 2017 (Sept. 2017). DOI: 10.1155/2017/3039817 (cit. on p. 44).

[75] T. Benzaoui, A. Selatnia, and D. Djabali. "Adsorption of copper (II) ions from aqueous solution using bottom ash of expired drugs incineration". In: *Adsorption Science & Technology* 36.1-2 (2018), pp. 114–129. DOI: 10.1177/0263617416685099. eprint: https://doi.org/10.1177/0263617416685099. URL: https://doi.org/10.1177/0263617416685099 (cit. on p. 44).

[76] R. A. Koble and T. E. Corrigan. "ADSORPTION ISOTHERMS FOR PURE HYDROCARBONS". In: *Industrial & Engineering Chemistry* 44.2 (1952), pp. 383–387. DOI: 10.1021/ie50506a049. eprint: https://doi.org/10.1021/ie50506a049. URL: https://doi.org/10.1021/ie50506a049 (cit. on p. 44).

[77] A. Kapoor, J. A. Ritter, and R. T. Yang. "An extended Langmuir model for adsorption of gas mixtures on heterogeneous surfaces". In: *Langmuir* 6.3 (1990), pp. 660–664. DOI: 10.1021/la00093a022. eprint: https://doi.org/10.1021/la00093a022. URL: https://doi.org/10.1021/la00093a022 (cit. on p. 45).

[78] F. B. Hildebrand. *Introduction to Numerical Analysis: Second Edition (Dover Books on Mathematics)*. 2nd. Dover Publications, Dec. 1987 (cit. on p. 60).

[79] S. Garrett. "Chapter 13 - Introductory Numerical Methods". In: *Introduction to Actuarial and Financial Mathematical Methods*. Ed. by S. Garrett. San Diego: Academic Press, 2015, pp. 411–463. ISBN: 978-0-12-800156-1. DOI: https://doi.org/10.1016/B978-0-12-800156-1.00013-3. URL: https://www.sciencedirect.com/science/article/pii/B9780128001561000133 (cit. on pp. 63, 70).

[80] K. H. Borgwardt. *The Simplex Method. A Probabilistic Analysis*. 1st. Springer Berlin, Heidelberg, 1987. DOI: 10.1007/978-3-642-61578-8 (cit. on p. 66).

[81] W. L. Winston and M. Venkataramanan. *Introduction to Mathematical Programming: Operations Research*. 4th. Thomson Learning, Oct. 2002 (cit. on p. 66).

[82] S. J. Wright. *simplex method*. In: *Encyclopaedia Britannica*. Accessed: 20 September 2022. Aug. 2021 (cit. on p. 66).

[83] D. M. Gay. "The AMPL Modeling Language: An Aid to Formulating and Solving Optimization Problems". In: *Numerical Analysis and Optimization*. Ed. by M. Al-Baali, L. Grandinetti, and A. Purnama. Cham: Springer International Publishing, 2015, pp. 95–116. ISBN: 978-3-319-17689-5 (cit. on pp. 67, 82).

[84] I. I. Cplex. "V12. 1: User's Manual for CPLEX". In: *International Business Machines Corporation* 46.53 (2009), p. 157 (cit. on p. 67).

[85] K. Levenberg. "A method for the solution of certain non-linear problems in least squares". In: *Quarterly of Applied Mathematics* 2 (1944). DOI: 10.1090/qam/10666 (cit. on pp. 68, 71).

[86] D. W. Marquardt. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". In: *Journal of the Society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441. DOI: 10.1137/0111030. eprint: https://doi.org/10.1137/0111030. URL: https://doi.org/10.1137/0111030 (cit. on pp. 68, 71).

[87] R. W. Hamming. *Numerical Methods for Scientists and Engineers (Dover Books on Mathematics)*. 2nd. Dover Publications, 1987 (cit. on p. 69).

[88] S. Basu, R. Pollack, and M.-F. Coste-Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics, Vol. 10)*. 2nd. Springer, 2006 (cit. on p. 69).

[89] J. Solomon. *Numerical Algorithms: Methods for Computer Vision, Machine Learning, and Graphics*. 1st. A K Peters/CRC Press, 2015. DOI: 10.1201/b18657 (cit. on p. 69).

[90] S. Chapra and R. Canale. *Numerical Methods for Engineers*. 8th. McGraw-Hill Education, 2020 (cit. on p. 69).

[91] A. Gil, J. Segura, and N. M. Temme. *Numerical Methods for Special Functions*. Society for Industrial and Applied Mathematics, 2007. DOI: 10.1137/1.9780898717822. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9780898717822. URL: https://epubs.siam.org/doi/abs/10.1137/1.9780898717822 (cit. on p. 70).

[92] J. H. Holland. "Genetic algorithms". In: *Scholarpedia* 7.12 (2012). revision #128222, p. 1482. DOI: 10.4249/scholarpedia.1482 (cit. on p. 72).

[93] L. Vanneschi. "Fitness Landscapes and Problem Hardness in Genetic Programming". In: *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*. GECCO '10. Portland, Oregon, USA: Association for Computing Machinery, 2010, pp. 2711–2738. ISBN: 9781450300735. DOI: 10.1145/1830761.1830916. URL: https://doi.org/10.1145/1830761.1830916 (cit. on p. 72).

[94] A. Wilde et al. *Evolutionary.jl*. Version v0.10.0. July 2021. DOI: 10.5281/zenodo.5110647. URL: https://doi.org/10.5281/zenodo.5110647 (cit. on pp. 73, 75, 79).

[95] H. Mühlenbein and D. Schlierkamp-Voosen. "Predictive Models for the Breeder Genetic Algorithm I. Continuous Parameter Optimization". In: *Evolutionary Computation* 1.1 (1993), pp. 25–49. DOI: 10.1162/evco.1993.1.1.25 (cit. on p. 74).

[96] A. Wächter and L. Biegler. "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical programming* 106 (Mar. 2006), pp. 25–57. DOI: 10.1007/s10107-004-0559-y (cit. on pp. 75, 82, 89).

[97] J. F. Bonnans et al. *Numerical Optimization*. 2nd. Springer Berlin, Heidelberg, 2006. ISBN: 978-3-540-35447-5. DOI: https://doi.org/10.1007/978-3-540-35447-5 (cit. on p. 75).

[98] A. Abbood et al. "Retention mechanism of peptides on a stationary phase embedded with a quaternary ammonium group: A liquid chromatography study". In: *Journal of Chromatography A* 1216.15 (2009), pp. 3244–3251. ISSN: 0021-9673. DOI: https://doi.org/10.1016/j.chroma.2009.02.031. URL: https://www.sciencedirect.com/science/article/pii/S0021967309002532 (cit. on p. 77).

[99] T. P. Santos. *Evolutionary.jl*. https://github.com/tpdsantos/Evolutionary.jl. 2020 (cit. on pp. 79, 81).

[100] I. Dunning, J. Huchette, and M. Lubin. "JuMP: A Modeling Language for Mathematical Optimization". In: *SIAM Review* 59.2 (2017), pp. 295–320. DOI: 10.1137/15M1020575 (cit. on p. 82).

[101] A. Gupta. "WSMP: Watson Sparse Matrix Package Part I - direct solution of symmetric sparse systems". In: (Jan. 2002) (cit. on p. 82).

[102] T. P. Santos et al. "Model-based design and optimization of GSSR chromatography for peptide purification". In: *Digital Chemical Engineering* 6 (2023), p. 100081. ISSN: 2772-5081. DOI: https://doi.org/10.1016/j.dche.2022.100081. URL: https://www.sciencedirect.com/science/article/pii/S2772508122000722 (cit. on p. 89).

[103] P. Nestola et al. "Robust design of adenovirus purification by two-column, simulated moving-bed, size-exclusion chromatography". In: *Journal of Biotechnology* 213 (2015). Integrated Continuous Biomanufacturing: A New Paradigm for Biopharmaceutical Production, pp. 109–119. ISSN: 0168-1656. DOI: https://doi.org/10.1016/j.jbiotec.2015.01.030. URL: https://www.sciencedirect.com/science/article/pii/S0168165615000760 (cit. on p. 94).

[104] J. P. B. Mota, J. M. M. Araújo, and R. C. R. Rodrigues. "Optimal design of simulated moving-bed processes under flow rate uncertainty". In: *AIChE Journal* 53.10 (2007), pp. 2630–2642. DOI: https://doi.org/10.1002/aic.11281. eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.11281. URL: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.11281 (cit. on p. 94).

[105] J. Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM Review* 59.1 (2017), pp. 65–98. DOI: 10.1137/141000671. URL: https://epubs.siam.org/doi/10.1137/141000671 (cit. on p. 100).

[106] T. van Beelen. *RS-232 for Linux, FreeBSD and Windows*. https://www.teuniz.net/RS-232/index.html (cit. on pp. 100, 161).

[107] clendinning et al. *SeaBreeze*. https://sourceforge.net/projects/seabreeze/ (cit. on pp. 100, 167).

[108] D. Schleef et al. *Comedi: linux control and measurement device interface*. https://www.comedi.org/ (cit. on pp. 100, 163).

[109] A. Cerepi, L. Humbert, and R. Burlot. "Pore-Scale Complexity of a Calcareous Material by Time-Controlled Mercury Porosimetry". In: *Characterisation of Porous Solids V*. Ed. by K. Unger, G. Kreysa, and J. Baselt. Vol. 128. Studies in Surface Science and Catalysis. Elsevier, 2000, pp. 449–458. DOI: https://doi.org/10.1016/S0167-2991(00)80050-6. URL: https://www.sciencedirect.com/science/article/pii/S0167299100800506 (cit. on p. 104).

[110] https://pubchem.ncbi.nlm.nih.gov/. Accessed: 25-05-2021 (cit. on p. 108).

[111] K. Steward. *Essential Amino Acids: Chart, Abbreviations and Structure*. Accessed: 25-01-2023. Sept. 2019 (cit. on p. 110).

[112] https://www.handymath.com/cgi-bin/methanoltble3.cgi?submit=Entry. Accessed: 28-12-2022 (cit. on p. 120).

[113] A. Chibério. "Single-Column Chromatography with Recycle Lag Analog to Simulated Moving Bed Processes". PhD thesis. NOVA School of Science and Technology, 2019 (cit. on p. 121).

I

ChromatographyStudio

This and the following annexes are part of a standalone document that is intended to be a documentation guide for the automation software created in-house to control all machines in our experimental set-up, called *ChromatographyStudio* . We chose *Julia* as the programming language simply because we were already quite familiar with the language and we also developed a data analysis toolkit in the same language for full integration. Since the documentation guide was written separately from the main dissertation, the language might look and feel more informal and personal, since it is supposed to be understandable by anyone that wants to handle our set-up. It is intended to be as simple and practical as possible, in order to achieve said goal. Without further ado, let us get started.

II

Getting Started

This annex covers some simple and not so simple uses of *ChromatographyStudio* . By reading this chapter you will be able to perform basic experiments and some less basic ones, just by using some simple and straightforward commands. For more information and more in-depth explanations of what these commands do and how they do it, carefully read Annex III.

II.1 How to Start

There's a different way of starting *Julia* , depending on the Operating System (OS) used. Both Windows and Linux will be covered in this section.

II.1.1 On Windows

The 1-col Analog system runs on Windows 10, and the only thing you need to do is to open the shortcut named `julia-1.6.3`.

II.1.2 On Linux

The Combined GSSR-MCSGP-SMB system works in Linux, using the flavour Mint, that is based on Ubuntu. In this computer you only need to open the terminal and type `work` to change the working directory and then type `julia`. That's what you'll type:

```
# after opening terminal, the directory is
> work
# now it's  /Desktop/Repositories
> julia
```

After this you're ready to go.

II.2 How to Run

After opening *Julia* , we need to import the *ChromatographyStudio* package with:

```
using ChromatographyStudio
```

You'll see several messages, do not be frightened, you'll learn how to read them, and they're only bad if they're red, so don't be alarmed. If you don't see any red messages or *Julia* doesn't shut down (sometimes it can happen), you'll see another *Julia* prompt and you're ready to start making experiments. If you do have red messages or *Julia* breaks down, then check Chapter IV for a list of several common errors that you may encounter.

In these messages you'll see the equipments that are plugged in the computer and are actually recognizable by the system, so make sure, beforehand, that all the equipments are turned on and correctly connected to the computer.

II.3 Controlling Equipments

Currently, the way of running an experiment is different depending on the OS. After *Julia* 0.7, we dropped support for the Windows platform, but we restarted supporting Windows after *Julia* 1.6. Another thing is, the physical systems built do not have the exact same hardware, so different drivers and libraries were needed for each. This will change in the future, since we're working on trying to write some wrappers in pure *Julia* , so we can easily reuse code for each type of equipment. Nonetheless, most of the code is the same for both, so only a few steps will be discussed regarding each OS.

II.3.1 Controlling a HPLC pump

This is the same for both OSes. The way of controlling a K501 HPLC pump is through the `@pump` macro, like this:

```
@pump F = 1.0
```

This command turns the pump with ID `F` flowrate to $1ml/min$. The values can go from 0.0 to the pump head value (10 or $50ml/min$). This macro accepts many more arguments, but for this tutorial we'll only learn how to set flowrates (for both K501 and S1050) and channel composition (only for S1050). If you're on the Combined system, you will use a S1050 pump. This pump can mix four channels, so it's useful for you to change channel compositions at any given time. This can be done with:

```
@pump G A=0 B=50 C=0 D=50
@pump G comp=0,50,0,50
```

These two ways were created just for convenience. Using the first syntax you don't need to specify the composition of all four channels, the channels that are not specified will maintain their previous values. The second syntax is more compact but needs the values of all four channels for it to work. The choice is yours, you just need to make sure that the sum of all four channels is 100 (including the ones you're not specifying using

the first way), since these values are percentages of each channel. In here the ID `G` is the default one for gradient pump.

II.3.2 Controlling pneumatic valves

This is different for each system, simply because the hardware used is different.

II.3.2.1 1-Col Analog

This is made using the `@valve` macro. The syntax is simple:

```
@valve +V1 +V2 -V3 -V4
@valve -all
@valve +all
```

This one is quite intuitive, positive sign for opening a valve and negative sign for closing a valve. You also have the command `all` to open or close all valves.

II.3.2.2 Combined System

This is made using the `@valves` macro. The syntax is simple:

```
@valves 1,2,-3,-4
@valves -all
@valves +all
```

Again, is quite intuitive: positive value for opening and negative value for closing. The command `all` is for opening or closing all valves.

II.3.3 Setting Flow Pathways

The `@step` macro is the most general macro. In it you specify the flow path that you want and the macro opens/closes valves accordingly. Since the two systems available are different, this macro is different for both systems, so let's see each of them separately.

II.3.3.1 1-Col Analog

This system has only one chromatographic column and a recycling device, so the steps implemented are simpler. Some examples are:

```
@step F => COL => P
@step E => COL => R
@step E => COL => W
@step R => COL => W
```

First step says the liquid will be pumped by pump with ID `F`, pass through the column and go straight to the product container. The second pumps with `E`, passes through column and goes to the recycling device (always denominated with ID `R`). The third

pumps the column outlet to waste. The last one is a nuance of this specific system. The recycling device, which is basically a container, can also pump liquid. There is a way to pass the liquid through a column bypass, but that's controlled by a manual valve, so just ask someone in the lab to help you in case you don't know how to do it.

II.3.3.2 Combined System

This system has three columns and we can use any number of columns in each step of an experiment. Besides that, any column can receive and transport liquid to any column, so the number of combinations are vast. We had the patience to implement every column combination that the system supports, so the options are quite extensive. Let's take a look at some:

```
@step F => BYPASS => P
@step E => COL1   => W
@step G => COL21  => W
@step E => COL312 => P
```

In here we can use 0, 1, 2 or 3 columns at any point in time (`BYPASS` means a bypass to all columns). The number order indicates the column order from which the fluid will pass through. In the example 4, pump `E` will pump to column 3, the outlet of column 3 goes to column 1 and the outlet of column 1 goes to column 2. The outlet of column 2 goes to the product container.

Another thing you can do with this macro is specify how long this step will take using the syntax:

```
@step F => BYPASS => P : 30.0
```

This way you're saying that step `F => BYPASS => P` will last for 30 seconds (the time has to be given in seconds).

II.4 Running an Experiment

Before starting an experiment, make sure all the solutions are well made, have enough liquid for the entire run and that every bottle is connected to the right pump. After that, you can go to the command prompt .

The first step is to set the UV wavelengths to be monitored by the software. In here we'll use a very important command, which is the `@uvc` macro. We'll do it like this:

```
@uvc A wavelengths=[230.0,250.0,300.0,600.0]
```

`@uvc` is a macro: in *Julia* , macros are really versatile and are especially good for interacting with non-programmers. In this example, `@uvc` is the macro name, `A` is the device ID that appears when starting *ChromatographyStudio* and `wavelengths=[230.0,250.0,300.0,600.0]` tells *Julia* to monitor only those five wavelengths. Note that the ".0" after each number is

important, it says that the number is a real number instead of an integer. In *Julia* and in *ChromatographyStudio* this matters a lot, so make sure that all numbers that you put inside this command are real.

After that we need to set some references for the spectrometer to give us correct readings. We need to tell it how much noise there is when the light is shut down and when the system is completely filled by eluent. We do this with:

```
# set light source button to 'close'
@uvc A set=dark

# set light source button to 'open'
@uvc A set=ref
```

Now we want to see in real time how the absorbance is changing when different concentrations pass through the system. To do that we just ask it to plot the results:

```
@uvc A plot=all
```

We want to save our experimental data for later treatment. That's done with the `@monitor` macro:

```
@monitor file = "path/to/filename.mon"
```

The ".mon" extension is important, only with that extension will the file be read by our data analysis software. We HIGHLY recommend that you use our data extraction and analysis software too, we spent a lot of time creating it to save us and you a lot of time.

It's good practice to set the spectrometer reference again right before or after you start monitoring the data, so let's do it:

```
@uvc A set=ref
```

And now you can start implementing all the steps your experiment needs. A short example of an experiment to get a breakthrough curve of a blue dextran mixture:

```
# set light source parameters before experiment
@uvc A wavelengths=[230.0,250.0,300.0,600.0]
@uvc A set=dark
@uvc A set=ref
@uvc A plot=all

# starting monitor, followed by another UV referencing
@monitor file = "path/to/filename.mon"
@uvc A set=ref

# starting the actual experiment
@step E => COL => W
@pump E = 1.0
sleep(60.0)
@pump E = 0.0
@step F => COL => W
@pump F = 1.0
sleep(60.0)
@pump F = 0.0
@step E => COL => W
@pump E = 1.0
sleep(120.0)
@pump E = 0.0

@valve -all # 1-col analog
@valves -all # combined system
@monitor stop
```

This is a simple procedure, with the last line just meaning to stop writing on the monitor file. There are some things to take into account here:

- NEVER turn on a pump flowrate before setting a pathway, or at least open some discharge valve. It will ruin your experiment and can also break your equipments.
- ALWAYS turn off a pump flowrate before changing steps that imply changing the pump. For instance, a sequence like:

  ```
  @step E => COL => W
  @step F => COL => W
  ```

  Will not work, the flowrate of `E` must be turned off before `@step F => COL => W`, like in the previous example.
- ALWAYS make sure that, when changing a step, you turn on the flowrate of the right pump. Always double-check that the IDs match. Avoid things like:

  ```
  @step E => COL => W
  @pump F = 1.0 # must be @pump E = 1.0
  ```

These three points are specially important in the 1-Col Analog (running on Windows). The Combined System, which runs in Linux, already has some shields that prevent these errors. But make no mistake, those shields only protect the equipment, your experiment will still be ruined, so always be extra careful to not waste time on failed experiments that could be avoided.

After all of this you're now ready to perform some basic experiments. To perform more complex experiments (SMB cycles, for instance), there's an easy way that is not covered in this tutorial, but will be covered in Chapter III.

III

The *ChromatographyStudio* Software

ChromatographyStudio was basically created to perform experimental runs with our homemade systems, and we developed a protocol that works and it's relatively easy to implement. The only reason we say "relatively easy" is because we don't have a graphical interface yet, so it's not super easy for someone that never used or even saw a command prompt . Anyway, we made it as easy as possible for anyone to perform experiments.

In this Chapter we'll cover every command stated in Chapter II with more detail and explain some nuances of those commands, specially all the arguments accepted and some exceptions.

III.1 Macros Explained

This section will cover in more detail all the macros presented in Chapter II.

In *Julia* , a macro is a piece of code that generates code. A function receives its input and converts it in a result, which could be anything. A macro receives its input and converts it into specialized code to use it later on. Macros are really powerful, and it's easy to write the inputs to them, that's why we decided to use macros as wrappers for all the code, and the user only needs to use them.

In table III.1 we have all the macros created.

[1] only works on the Combined System

Table III.1: Available macros

Macro	Description
`@uvc`	controls every aspect of a spectrometer
`@pump`	controls every aspect of a HPLC pump
`@valve` or `@valves`	controls every aspect of a pneumatic valve
`@step`	defines a specific pathway for the fluid
`@csinfo`[1]	detailed information of every capillary, connector and valve in the system

III.1.1 The @uvc macro

The `@uvc` macro stands for UV cell. Every functionality that a spectrometer has can be accessed through this macro. The general syntax is:

```
@uvc ID command1 command2 command3 ...
```

Only `ID` and `command1` are mandatory. As mentioned several times, each equipment has a unique ID associated, and that appears in the initial screen when you start *ChromatographyStudio* . `command1` and so on are just the arguments that you'll pass to the macro. Each command is a key/value pair, like `key=value`. A right command would be:

```
@uvc ID key1=value1 key2=value2 key3=value3...
```

To see all the arguments supported, just type `?@uvc` in the *Julia* command prompt .

III.1.2 The @pump macro

As the name clearly states, `@pump` stands for HPLC pump. Everything regarding pumps can be made with this macro. The general syntax is:

```
@pump ID cmd1 cmd2 cmd3 ...
```

Only `ID` and `cmd1` are mandatory. For setting parameters, `cmd1` and so on are key/value pairs with the syntax `key=value`. If you want only to check the values of some parameters, use `key=get`. Some examples[2]:

```
@pump F Q=1.0 pmin=0.0 pmax=300.0 # setting pump values
@pump F Q=get pmin=get pmax=get   # reading pump values
```

For a list of all arguments type `?@pump` in the *Julia* command prompt .

III.1.3 The @valve/@valves macro

There are two versions because there are two systems that have different hardware to control the pneumatic valves. `@valve` is for the 1-col Analog and `@valves` is for the Combined System. The general syntax is:

```
@valve +V1 +V2 +V3 -V4 -V5 -V6
@valves 1,2,3,-4,-5,-6
```

`@valve` needs a "V" between the positive/negative and the valve number, whether `@valves` needs only positive or negative integer numbers, with a comma separating each number. Positive to open and negative to close the valve. You can also open or close all of the valves simultaneously, using the same analogy:

```
@valve  +all # or -all
@valves +all # or -all
```

[2] `@pump F = 1.0` is simply an abbreviation of `@pump F Q=1.0`.

This information is also available in the command prompt , by typing `?@valve` or `?@valves`.

III.1.4 The `@step` macro

This macro was created for convenience, since the only thing it does is open and close valves. Each pathway has a specific combination of opened and closed valves, and this macro helps us to establish each pathway in a simple way, by explicitly writing that pathway.

III.1.5 The `@csinfo` macro

This macro was created only for the Combined System. Since this set-up has several connectors, capillaries and valves, we came up with a code in order to easily identify each piece of equipment that we have in our system. The letter `A` is for connectors, letter `C` for capillaries and letter `V` for valves. Everything is identified with labels, and if you're having trouble to figure out the exact pathway the fluid is taking, this macro comes in handy:

```
@csinfo A10
```

This example will give you the information regarding connector `A10`. No rush on memorizing every code, if you make a mistake the macro will post an information message about the code. If you're doing some reverse engineering, such as finding the right connector for a specific pathway, you can simply type:

```
@csinfo A
```

And a table with all the connectors will appear on the screen. Everything is applied for both capillaries and valves. You can find some more information by typing `?@csinfo` on the command prompt .

III.2 Equipment Structures

Until now you learned on how to use all the macros that are implemented, and it will probably be the only thing you need. However, since you might need to add or remove equipment from each of the systems, you will need to change a specific file in the source folder, wich is the `new-devices.jl` file. To do that you need to understand how the information is stored and updated in real-time inside the software.

III.2.1 The purpose of using structures

As in every other object-oriented programming language, a structure (or object) is basically a container that stores information regarding something. As an example, we can create a virtual HPLC pump by creating a structure that has all the information of a pump (flowrate, pressure, communication port, etc.). To create said virtual pump, we would

need more than the physical parameters, we would need to emulate the communication between the PC and the pump, implement it asynchronously and other more complex features.

Even though this is explained, we developed several ways for users to not be worried about them. In order to create those structures easily, we developed wrapper functions so the users only need to worry about the actual properties that they can get from the equipments used. Each structure has its own wrappers, so they will be shown at the right time.

To simplify, we created a global structure for every type of equipment. The user only needs to know the physical properties of the equipment, since the other more complex tasks are created and run in the background, with no need for user interaction. The next sections will talk about how to add more equipments to your experimental set-up.

III.2.2 Connecting the Equipment to the Computer

First and foremost, we need the equipment to be recognized by the computer. Until now, the software only supports equipments that can be connected through RS-232 communication, either being with 9-pin connectors or USB. There are several ways to connect an equipment to a computer through RS-232 communication (USB to RS-232, RS-232 to RS-232, etc.), and each way has a specific manner to check if the equipment is well connected, so you need to figure out how you can do that with the way you're using to connect the equipment. After that you can find out the name of the port the computer established for that hardware.

After having the port name, you can go to the next sections and learn how to add it to *ChromatographyStudio* .

III.2.3 Off-Limits Structures

There are some structures that exist to add some robustness to this software. However, they are built internally and MUST NOT be tampered with. All the macros and functions described in this guide already perform the necessary changes to these structures. There are two structures with such characteristics.

III.2.3.1 The `ChromStudio` Structure

This is the main structure of the *ChromatographyStudio* software. It handles everything, from saving monitored data, opening and closing background processes, keep track of all equipment parameters, etc. It's internally created and no field can be changed explicitly. All the changes necessary are made through the macros and functions presented in this guide.

However, you can consult this structure to get some information that you might need. When *ChromatographyStudio* is opened in *Julia* , it automatically creates the `cs` variable,

Table III.2: `RS232` fields

Field Name	Type	Description
port	`AbstractString`	port name given by OS
baudrate	`Integer`	baudrate supported by equipment
databits	`Integer`	number of bits of data supported by equipment
parity	`Char`	parity supported by equipment
stopbits	`Integer`	number of stopbits supported by equipment

which is an abbreviation for `ChromStudio`. As long as you don't change any value, you can check any value inside this structure.

III.2.3.2 The `OceanOptics` Structure

This is a structure that handles port opening and closing of the Ocean Optics spectrometers. This is a structure that the user has to create before creating the spectrometer structures, explained in Subsection III.2.9.

The biggest between this and the `ChromStudio` structure is that this one HAS to be created, but is very simple:

```
oo = OceanOptics(1)
```

The integer number passed to the `OceanOptics` function h«is the number of spectrometers connected to the computer. If you have three spectrometers connected, the way to do it would be `oo = OceanOptics(3)`.

After creating it, at no point in time should the structure parameters be changed.

III.2.4 The `RS232` structure

The `RS232` structure is used to interact with equipments that are connected through serial ports. Although the name of the structure implies that only RS-322 connections are supported, the truth is that all types of serial port connections are supported, including some USB connections.

Before creating this one we used a package that depended on *Python* code to communicate to the equipments. However, since *Julia* is a JIT compiled language, using *Python* code meant that the code could not be compiled (*Python* is a purely declarative language, hence not compilable). To overcome this, we started using a C library [106] to perform this communication and built a *Julia* wrapper to use this library. In this way, the code is fully compilable and only the ports have to be opened in run-time.

The fields are listed in Table III.2

These fields are not exactly self-explanatory, so let's try and break them down:

- **port** -This name is given by the operating system. In Linux it will be something like `/dev/ttyS1` or `/dev/ttyUSB0`, depending if is connected through RS-232 or by USB.

In Windows it will be something like `COM2`.

- **baudrate** - It's the speed in which the signals are sent. This speed depends on the machine, so you have to find this information in instruction manuals, for instance.
- **databits** - number of bits transmitted to the machine. It depends on the equipment and can be 7 or 8 bits.
- **parity** - related to the error detection mechanism. It depends on the equipment and can be even, odd or none at all.
- **stopbits** - time period between the current set of data bits and the next one, to make sure no data is loss. It depends on the equipment and can be 1, 1.5 or 2 bits.

The way we create a structure in *Julia* is like calling a function:

```
RS232(port     ::AbstractString,
      baudrate ::Integer       ,
      databits ::Integer       ,
      parity   ::Char          ,
      stopbits ::Integer       )
```

Let's show some examples. For instance, if the instruction manual of our equipment says that it has a baud rate of 9600 (standard value), sends 8 data bits with each signal, has an even error detection mechanism and 1 stop bit, we could write in *Julia* in the following way:

```
sp = RS232("COM2", 9600, 8, 'E', 1)
```

To note that the fourth argument, can be either `'E'` for even, `'O'` for odd or `'N'` for none. `sp` is just a random name for a variable, in this case an abbreviation for serial port.

Now that we created the structure inside the variable `sp`, we can now communicate with the equipment connected to `"COM2"`. First we open the port for communication:

```
open_port(sp)
```

Easy as that. Now we just need to write to and read from the port:

```
write(sp, command)
str_from_serialport(sp)
```

When writing, the second argument has to be a string, meaning a sequence of characters (either letters or numbers) inside quotes. As an example, if we wanted to change the flowrate of a K-501 HPLC pump to 1 ml/min, the command would be:

```
write(sp, "F1000.000\r") # pump only accepts uL/min
```

The character `"\r"` is the representation of the key "Enter". It means that every command you type inside the `write` function has to end with `"\r"`, even if it doesn't show in the instruction manual of the equipment.

Table III.3: `LI_Valve` fields

Field name	Type	Description
cs	`ChromStudio`	ChromStudio instance
sp	`RS232`	serial port
id	`Char`	device ID
model	`AbstractString`	device model
pos	`Symbol`	valve position
atinit	`Function`	function to be run at run time
atexit	`Function`	function to be run at exit time

III.2.5 The `LI_Valve` structure

This structure is built to control the Knauer load/inject valve, a 2-position 6-port electric valve. Each of the 6 ports are connected with the previous and the next ports. In each position, the ports are connected with either the previous or the next port, but never both. For instance, in load position, port 1 is connected to port 2, 3 with 4 and 5 with 6, while in inject position port 2 is connected with port 3, 4 with 5 and 6 with 1. Table III.3 has all the associated fields of this structure.

To control this equipment, the `@livalve` macro was created for this purpose. It accepts only one argument at a time:

```
@livalve L
```

The two most important arguments are `L` and `I`, which stand for load and inject positions, respectively. Another important one is `P`, which is used to check the current valve position. There are other accepted arguments, which you can consult in the command line by running `?@livalve`.

III.2.6 The `ValveType` structure

This structure controls all the 25 pneumatic ON/OFF valves installed. To control it we use a NI-6509 PCI card from National Instruments, in which communication is made through a C library developed by Comedi [108]. Despite the high number of valves, since the PCI card does all the communications, we only need one piece of code for all of them. Table III.4 shows all the fields.

This is a special structure, since **NONE** of these fields are safe to change manually. Nevertheless, let's break down the new ones:

- **dioread:** This a Symbol that has the name of the C function used for reading the valves that are open and the name of the shared library. It is purely informative, it has no impact on the software. However, it is not supposed to be messed around.

Table III.4: `ValveType` fields

Field name	Type	Description
cs	`ChromStudio`	ChromStudio instance
dioread	`Tuple{Symbol,Symbol}`	function name used to read valve positions. Informative only
diowrite	`Tuple{Symbol,Symbol}`	function name used to write valve positions. Informative only
port	`AbstractString`	port name
dev	`Ptr{comedi_t}`	pointer that tells if the PCI card communication was successfully opened
subdev	`Integer`	number of devices connected to the PCI card
basechannel	`Tuple{Int64,Int64}`	start bit of each device
vals_chans	`DataFrame`	table that associates valve number with bit number
paths	`Dict`	dictionary with all the configured path
valves	`NTuple{N,Int64} where N`	configured valves
channels	`NTuple{N,Int64} where N`	configured channels
bits	`NTuple{N,Int64} where N`	configured bits
atinit	`Function`	function to be run at run time
atexit	`Function`	function to be run at exit time

- **diowrite:** Similar to **dioread**, it shows the name of the C function used for opening and closing valves and the name of the shared library. Again, purely informative but should not be tampered with.

- **dev:** It's a pointer to the communication channel with the PCI card. If the channel is successfully opened, this pointer will have a value different than zero.

- **subdev:** The PCI card is our main device, but it is connected to the actuall valve controllers, which can be more than 1. In our case there are two, and this field stated the number of sub-devices connected to the PCI card.

- **basechannel:** For each sub-device, we need to give to the library which bit the counting starts. This is specific to each system, so these parameters are fixed.

- **vals_chans:** Each valve number has a corresponding Comedi channel and bit. This data frame contains the correspondences of all the valves that are currently configured. This is created by reading the file `working_channels_valves.tsv` found in `src/channels_valves` inside the *ChromatographyStudio* project folder. You shall not change the numbers in each row, however you can delete rows of the valves that will not be used or add other valves from the file `all_channels_valves.tsv` in the same folder. The latter is read-only and cannot be changed.

Table III.5: `K501_Pump` fields

Field name	Type	Description
cs	`ChromStudio`	ChromStudio instance
sp	`RS232`	serial port
transmit	`Channel{Bool}`	channel to avoid deadlock communication
id	`Char`	character identifier
model	`AbstractString`	pump model
head	`Integer`	pump head (ml/min) = 10, 50
Q	`Float`	current flow rate (ml/min)
rec_valve	`Integer`	recycle valve associated
calib	`Function`	calibration function
der_calib	`Function`	derivative of calibration function
inv_calib	`Function`	inverse calibration function
taskstate	`Symbol`	true or false (finished or running)
task	`Task`	background task
atinit	`Function`	function to be run at run time
atexit	`Function`	function to be run at exit time

- **paths:** it's a dictionary that contains the combination of valves for each different system step. It's specific to the system, so it mustn't be changed.
- **valves:** The same as the valve number column in `vals_chans`, for convenience.
- **channels:** The same as the channel number column in `vals_chans`, for convenience.
- **bit:** The same as the bit number column in `vals_chans`, for convenience.

Since no field should be changed, you only need to give the port name in order to build the structure:

```
ValveType(port ::AbstractString)
```

III.2.7 The `K501_Pump` structure

This structure controls a Knauer K501 HPLC pump. All the fields are listed in Table III.5.

These are all the fields that you can access in this structure, but only a small portion are safe to modify, most of them are used implicitly by the software and must NEVER be changed. The safe ones[3]are shown in Table III.6.

The easiest way of creating this structure is by using the wrapper function:

[3]**NOTE:** Those are safe to change in the sense that the software will still be able to run without errors. However, if you change for the wrong parameters, your experiments will be ruined, so always be extra careful with those changes.

Table III.6: Safe-to-change fields

Field name	Type	Description
sp	`RS232`	serial port
id	`Char`	character identifier
model	`AbstractString`	pump model
head	`Integer`	pump head (ml/min) = 10, 50
Q	`Float`	current flow rate (ml/min)
calib	`Function`	calibration function
der_calib	`Function`	derivative of calibration function
inv_calib	`Function`	inverse calibration function

```
# for the Combined System
K501_Pump(port      ::AbstractString,
          id        ::Char          ,
          rec_valve ::Integer       ,
          head      ::Integer       ,
          calib     ::Vector{<:Real},
          inv_calib ::Vector{<:Real})

# for the 1Col-Analog System
K501_Pump(port      ::AbstractString,
          id        ::Char          ,
          head      ::Integer       ,
          calib     ::Vector{<:Real},
          inv_calib ::Vector{<:Real})
```

`port` is the name of the port the OS assigned to the pump, `id` is the device ID supplied by the user, `rec_valve` is the recycle valve number for that pump, `head` is the pump head (10 or 50ml/min), `calib` is the polynomial coefficient vector to be used for the calibration function and `inv_calib` is the same as `calib` but for the inverse calibration function.

As an example, let's assume we have a K-501 HPLC pump in our lab. We connected it to the computer, discovered that the name the OS gave to the equipment is "COM2", we decide to give it the ID 'A', the pump head is of 10ml/min and assumed that the pump is perfectly calibrated, meaning that the flowrate stated by the pump is exactly the flowrate we measure in real life. With all the parameters set, we can create the pump structure:

```
pump = K501_Pump("COM2", 'A', 10, [1.0], [1.0])
```

Both of the one-element vectors represent the calibration coefficients, and since the calibration is ideal in this case, there's only one coefficient per vector (polynomial of degree zero) and the coefficient is 1. For how to calibrate a pump, carefully read Section III.2.10.

Table III.7: `S1050_Pump` fields

Field name	Type	Description
cs	`ChromStudio`	ChromStudio instance
sp	`RS232`	serial port
id	`Char`	character identifier
model	`AbstractString`	pump model
head	`Integer`	pump head (ml/min) = 10, 50
Q	`Float`	current flow rate (ml/min)
comp	`Dict{Symbol,Int64}`	channel composition
calib	`Function`	calibration function
der_calib	`Function`	derivative of calibration function
inv_calib	`Function`	inverse calibration function
transmit	`Bool`	channel for data transmission
atinit	`Function`	function to be run at run time
atexit	`Function`	function to be run at exit time

III.2.8 The `S1050_Pump` structure

The `S1050_Pump` structure concerns the Knauer S1050 gradient HPLC pump. It is comprised of four channels (A, B, C and D) that converge into a 10ml pump head where they are mixed. The structure created is quite similar to the `K501_Pump`, although it needs a structure of its own due to its different RS-232 commands. The structure is presented in Table III.7. As you can see, the biggest difference is the composition field, which harbors the composition of each of the four channels. The values are given in percentages.

The structure is created using the following syntax:

```
S1050_Pump(port ::AbstractString,
           id   ::Char          ,
           head ::Integer       )
```

As an example, we can create the gradient pump structure in the following way:

```
pump_G = S1050_Pump("COM3", 'G', 10)
```

III.2.9 The `UVCell` Structure

The `UVCell` structure is used to control the Ocean Optics spectrometers. This structure is basically a wrapper for a C library created by Ocean Optics employees which supports several spectrometer models, all of them stated in [107]. All the fields are listed in Table III.8.

These parameters need a bit more detailed explanations, which will be given:

- **maxintensity** - maximum light intensity level the equipment can support. Each model has its own maximum

Table III.8: `UVCell` fields

Field name	Type		Description
cs	`ChromStudio`		ChromStudio structure
oo	`OceanOptics`		OceanOptics Structure
serialnumber	`AbstractString`		device serial number
id	`Char`		device ID
maxintensity	`Real`		maximum intensity supported
integrationtime	`Integer`		integration time
lcut	`Integer`		number of left wavelengths to be removed
rcut	`Integer`		number of right wavelengths to be removed
boxcar	`Integer`		parameter for Boxcar averaging technique
average	`Integer`		number of time averages
baseline	`Vector{<:Real}` or `Tuple{<:Real,<:Real}`		values to determine baseline

- **integrationtime** - integration time is the time interval the detector uses to read pixel information. The lower the integration time, the better the resolution, with the downside of increasing the reading time for each data point. Each equipment has a minimum integration time, since it depends on how good the hardware is at reading pixel information.

- **lcut** - each spectrometer can record a specific amount of wavelengths. This parameter, abbreviation for "left cut" is used to remove from the wavelength vector a certain amount of wavelengths to be recorded. For instance, if `lcut` is 5, then the first 5 wavelengths are going to be removed and will not be monitored by the *ChromatographyStudio* software.

- **rcut** - the same as `lcut` but for the right side of the wavelength vector, removing wavelengths from the end of the vector to not be monitored by the *ChromatographyStudio* software.

- **boxcar** - it's a type of signal smoothing, which consists of averaging several pixels, with the purpose to reduce signal to noise ratio, at the expense of optical resolution. The bigger the value, the smoother the signal, with the downside of losing resolution.

- **average** - determines how many measures are taken, in order to improve the signal to noise ratio. The bigger the number, the better the signal to noise ratio, but the measurement time increases significantly.

- **baseline** - for this parameter you need to give two values. Those two values are the interval in which the baseline is calculated. For instance, if the interval given

is `[0,50]`, then the software will use all values between 0 and 50 to determine the baseline value.

For creating a structure, we can use this syntax:

```
UVcell(cs              ::ChromStudio    ,
       oo              ::OceanOptics    ,
       serialnumber    ::AbstractString,
       id              ::Char           ,
       maxintensity    ::Real           ,
       integrationtime ::Integer = 0    ,
       lcut            ::Integer = 0    ,
       rcut            ::Integer = 0    ,
       boxcar          ::Integer = 0    ,
       average         ::Integer = 1    ,
       baseline        ::Union{Vector{<:Real},
                         Tuple{<:Real,<:Real}
                              } = Float[])
```

In the example above, all the fields that have a value in front of it are already default values. However, it's recommended to change them according to the equipment used, since every model has its own optimized parameters. A practical example could be:

```
uv_A = UVcell(cs, oo, "USB4C02624", 'A', 4095,
              integrationtime = 50000    ,
              lcut            = 5        ,
              rcut            = 5        ,
              boxcar          = 10       ,
              average         = 10       ,
              baseline        = [0.0,0.0])
```

The values presented above are optimized for the USB4000 spectrometer, one of the models used in our research group. The variables `cs` and `oo` are explained in Subsection III.2.3.

III.2.10 The `SartoriusBalance` Structure

This structure holds all the information regarding the Sartorius balance implemented in our system. The balance is used mainly for pump calibration, since the flowrate stated by HPLC pumps sometimes are not accurate. The fields are presented in Table III.9.

Since the balance is not an equipment that needs to be controlled, the structure does not need many fields. To create the structure you would do something like:

```
bal_A = SartoriusBalance(cs, 'A')
```

Monitoring the balance is not automatic, so it has to be explicitly specified before starting an experiment:

Table III.9: `SartoriusBalance` fields

Field Name	Type	Description
cs	`ChromStudio`	ChromStudio structure
sp	`RS232`	Serial Port structure
id	`Char`	balance ID
taskstate	`Symbol`	used to verify if it's running in the background
task	`Task`	pointer to the background process
atinit	`Function`	function to run at run time
atexit	`Function`	function to run at exit time

```
monitor_balance(balance          ::SartoriusBalance,
                sampling_interval ::Real           ,
                time_window       ::Integer        ,
                time_unit         ::Symbol         ,
                color             ::AbstractString ,
                fname             ::AbstractString )
```

All parameters except `balance` have default values, which in general are good for the majority of experiments. In practice you would only need:

```
monitor_balance(bal_A)
```

And `bal_A` will start being monitored into the file created by the `@monitor` macro.

III.2.11 Calibrating a HPLC Pump

Creating a balance structure is quite simple, but the most important part is how to use this structure to calibrate a HPLC pump. This is done with the `calibrate_pump` function.

The `calibrate_pump` function accepts as input the pump to be calibrated, the balance to be used for calibration (we can install more than one balance) and the flowrates to be calibrated. Let's look at an example:

```
cal = calibrate_pump(pump_F, bal_A,
                     [1.0,1.5,2.0,2.5,3.0])
```

In this example, we are going to calibrate the HPLC pump called `pump_F` by using balance `bal_A` and measuring flowrates `[1.0,1.5,2.0,2.5,3.0]`. This is a process that takes a good amount of time, depending on how many flowrates you decide to calibrate. With more flowrates used, the calibration is more accurate, but the calibration time increases significantly. The rule of thumb is to use around 5 flowrates inside your operating range. For instance, if you never use flowrates below 0.5ml/min nor above 3ml/min, choose these two flowrates (minimum and maximum) and three more in the middle. Of course this is not a strict rule, but we generally had good calibration accuracies in this way, so you definitely should give it a try.

The next step is to add the calibration parameters to the pump in question. This is made using the `update_calibs!` function:

```
update_calibs!(pump    ::K501_Pump     ,
               cal     ::CalibPumpGraph;
               iswrite ::Bool = true   )
```

In this case, `cal` is the result of a `calibrate_pump` run and `iswrite` is an optional argument that writes the calibration result to a JSON file so we don't need to calibrate the pump every time we start *ChromatographyStudio* . The default is true. Following the same example, to update the calibration parameters for pump F we just need to run:

```
update_calibs!(pump_F, bal_A)
```

To reset a previously performed calibration the `reset_calibs!` function was created:

```
reset_calibs!(pump ::K501_Pump, delete ::Bool = false)
```

The `delete` option is used to delete, or not, the calibration entry from the JSON file.

To remove the JSON file in order to reset all calibrations just type:

```
rm_calibs()
```

The older the pump is, the more important it is to calibrate it regularly, because the flowrates become less and less accurate as the pump age and use increases. As an extra precaution, is advisable to use the balance to monitor the fluid weight that is leaving the bottles to improve the flowrate accuracy.

IV

Good practices in *ChromatographyStudio*

This Chapter will cover common practices that are recommended in order to use this software more efficiently. We will cover best practices to run experiments, how to create more complex chromatographic cycles and some common errors that were still not fixed.

IV.1 Interaction between valves and pumps

The ON/OFF valves and the pumps have a very tight relation. When opening and closing valves, you must always pay close attention to which pumps are on positive flowrates, because if you inadvertently close a valve that should not be closed, you might experience pump shutdowns due to over pressure. With that in mind, for running experiments, you should avoid the `@valves` macro and use the `@step` macro instead. Furthermore, even with the `@step` macro you are not completely safe. Imagine this usage example:

```
@step F => COL1 => P
@pump E = 1.0
```

With this step fluid from pump F enters column 1 and goes to product. Right after activating this step we are activating pump E with a flowrate of 1ml/min. This will lead to an over pressure error and pump E will shutdown. In the future we want to add some protections for these kind of mistakes but for now this has to be explicitly looked after by the user. Always pay extra attention and only activate the flowrate of the pumps which path you opened with `@step`.

IV.2 Use functions

Every experiment has a protocol that must be followed. within *ChromatographyStudio* , this results in a sequence of steps that have to be performed in order to obtain the results we expect. Even before the experiment starts, you need to set the UV parameters, start

monitoring and then start the actual experimental steps. After defining all the steps of an experiment, wrap it inside a function and give it the right mandatory arguments so you do not forget anything.

We will use the example in the tutorial section of a pulse experiment:

```
# set light source parameters before experiment
@uvc A wavelengths=[230.0,250.0,300.0,600.0]
@uvc A set=dark
@uvc A set=ref
@uvc A plot=all

# starting monitor, followed by another UV referencing
@monitor file = "path/to/filename.mon"
@uvc A set=ref

# starting the actual experiment
@step E => COL1 => W
@pump E = 1.0
sleep(60.0)
@pump E = 0.0

@step F => COL1 => W
@pump F = 1.0
sleep(60.0)
@pump F = 0.0

@step E => COL1 => W
@pump E = 1.0
sleep(120.0)
@pump E = 0.0

@valves -all # combined system
@monitor stop
```

The first four steps have to be performed separately, but all the other steps can be put inside a function:

```
function pulse_experiment()

    # column equilibration
    @step E => COL1 => W
    @pump E = 1.0
    sleep(120.0)
    @pump E = 0.0

    # monitoring start, followed by another UV referencing
    @monitor file = "path/to/filename.mon"
    @uvc A set=ref
    sleep(5.0) # give it some time to stabilize

    # injection step
    @step F => COL1 => W
    @pump F = 1.0
    sleep(60.0)
    @pump F = 0.0

    # elution step
    @step E => COL1 => W
    @pump E = 1.0
    sleep(120.0)
    @pump E = 0.0

    @valves -all
    @monitor stop
end
```

This makes it more neat and helps prevent some human error, such as starting the experiment without monitoring the data (it happened to us several times, it's one of the most annoying mistakes, specially for long experiments). This specific example can be made more general if we add arguments to the function:

```
function pulse_experiment(filename ::String,
                          tE       ::Real  ,
                          QE       ::Real  ,
                          tF       ::Real  ,
                          QF       ::Real  )

    # equilibration step
    @step E => COL1 => W
    @eval(@pump E = $QE)
    sleep(tE)
    @pump E = 0.0

    # monitoring start, followed by another UV referencing
    @eval(@monitor file = $filename)
    @uvc A set=ref
    sleep(5.0) # give it some time to stabilize

    @step F => COL1 => W
    @eval(@pump F = $QF)
    sleep(tF)
    @pump F = 0.0

    @step E => COL1 => W
    @eval(@pump E = $QE)
    sleep(tE)
    @pump E = 0.0

    @valves -all # combined system
    @monitor stop
end
```

In this example, `filename` is the name of the monitor file, which needs the `.mon` extension, `tE` is the time of the elution step, `QE` the elution flowrate, `tF` the time of the injection step and `QF` the injection flowrate.

In this case, when you want to give a variable to a macro, you need to use it inside the `@eval` macro. You can see it in the example above, the macros that need the function arguments are inside `@eval`, while the ones that have a fixed input are not. Furthermore, if you want to add a variable to a macro expression, said variable has to be preceded by a `$`.

Adding the monitoring inside the function is definitely an advantage since all your experiments will definitely be monitored, but one must be aware of the fact that the `@monitor` macro, if called with a file name that already exists, it overwrites it completely. This can be problematic when you are running several experiments in sequence. Following the previous example, imagine you are determining the retention time of several components in a

column, and for that you run the `pulse_experiment` function for different raw materials. if you are running the experiment in the command line you would type something like:

```
pulse_experiment("COMP1.mon",1.0,1.0,1.0,1.0)
```

As you start to realize, after component 1 is finished, it is very easy for the operator to change the raw material, go back to the command line to start evaluating component 2 and forget to change the file name. If that happens, the previous experiment will be completely overwritten and destroyed. You deduce correctly, unfortunately this also happened to us. To prevent such disasters, we can introduce in our function a way to verify if the file already exists:

```
function pulse_experiment(filename ::String,
                          tE       ::Real  ,
                          QE       ::Real  ,
                          tF       ::Real  ,
                          QF       ::Real  )

    if ispath(filename)
        error("file $filename already exists.")
    end

    # equilibration step
    @info("Equilibrating...")
    @step E => COL1 => W
    @eval(@pump E = $QE)
    sleep(tE)
    @pump E = 0.0

    # monitoring start, followed by another UV referencing
    @info("Monitor start")
    @eval(@monitor file = $filename)
    @uvc A set=ref
    sleep(5.0) # give it some time to stabilize

    @info("Injecting...")
    @step F => COL1 => W
    @eval(@pump F = $QF)
    sleep(tF)
    @pump F = 0.0

    @info("Eluting...")
    @step E => COL1 => W
    @eval(@pump E = $QE)
    sleep(tE)
    @pump E = 0.0

    @info("Experiment Finished")
    @valves -all
    @monitor stop
end
```

`ispath` is a built-in function that returns true if the given file path exists and false if it doesn't. In this way our function will not allow us to overwrite an existing experiment. If you really want to overwrite it, just manually delete the file. The `@info` macro is just to print in the command line the experiment phase that is currently running.

Another good practice is to automatically shutdown the system when some error

occurs in the experiments. We can achieve that with the `try catch` statement:

```
function pulse_experiment(filename ::String,
                          tE       ::Real  ,
                          QE       ::Real  ,
                          tF       ::Real  ,
                          QF       ::Real  )

    try
        if ispath(filename)
            error("file $filename already exists.")
        end

        # equilibration step
        @info("Equilibrating...")
        @step E => COL1 => W
        @eval(@pump E = $QE)
        sleep(tE)
        @pump E = 0.0

        # monitoring start, followed by another UV referencing
        @info("Monitor start")
        @eval(@monitor file = $filename)
        @uvc A set=ref
        sleep(5.0) # give it some time to stabilize

        @info("Injecting...")
        @step F => COL1 => W
        @eval(@pump F = $QF)
        sleep(tF)
        @pump F = 0.0

        @info("Eluting...")
        @step E => COL1 => W
        @eval(@pump E = $QE)
        sleep(tE)
        @pump E = 0.0

        @info("Experiment Finished")
        @valves -all
        @monitor stop
    catch
        @pump F = 0.0
        @pump E = 0.0
        @valves -all
        @info("An error occurred, shutting down")
    end
end
```

The code inside `try` is the experimental protocol, but if some error occurs inside one of the steps, the code jumps right to `catch` and runs the code inside it. In this way the system will be completely shut down in case something happens inside the experimental system.

IV.2.1 Generalize code as much as possible

This tip is not actually specific to *ChromatographyStudio* but to programming in *Julia* . Functions allow us to create generalized code that can be applied into many different situations. For instance, in our previous example, you might have noticed three protocols that are used should be used in every experiment: 1) checking if we don't overwrite already existing experiments, 2) system equilibration with solvent and 3) start monitoring followed by a setting a reference baseline. These three protocols can easily be moved to their own separate functions. We can start with the file check step:

```
function existfile(filename ::String)
    if ispath(filename)
        error("file $filename already exists.")
    end
    return nothing
end
```

In this example we just copy and pasted the code from our pulse experiment to a separate function. In this way we can reutilize this piece of code when creating other experimental protocols. Now let's do the same for the equilibration step:

```
function equilibration(PUMP     ::Symbol     ,
                       tE       ::Real       ,
                       QE       ::Real       ;
                       shutdown ::Bool = true)

    @info("Equilibrating...")
    @eval(@pump $PUMP = $QE)
    sleep(tE)

    if shutdown
        @eval(@pump $PUMP = 0.0)
    end

    return nothing
end
```

This time we didn't just copy and pasted the code, we did some modifications to generalize it even more. First we removed the `@step` macro, otherwise our code could not be used if we wanted to equilibrate other flow paths. Second we added the variable `PUMP` so we could choose which pump we would use for the equilibration step. Lastly, we added

the optional argument `shutdown`, with the default being `true`. This optional argument is actually a safety measure that prevents us from leaving the pump on even after the time interval has ended. Now you ask, why do we make it an optional argument and not explicitly add `@eval(@pump $PUMP = 0.0)` to always shutdown the pump in the end of the equilibration? On its own, this function should always shutdown the pump, but if to be used inside an experiment function, such as the `pulse_experiment`, it makes no sense to shutdown the pump after equilibration, since the actual experiment will immediately start after it. In those cases, we can set `shutdown=false` when the function is called and the equilibration pump will stay on.

The monitoring step can also be separated into its own function:

```
function monitor(filename ::String)

    existfile(filename)

    @info("Monitor start")
    @eval(@monitor file = $filename)
    @uvc A set=ref
    sleep(5.0)

    return nothing
end
```

In here, the `existfile` function fits perfectly. In this way we can directly apply the `monitor` function inside an experimental protocol without worrying about possible file overwrites. Finally, the function of our pulse experiment would look like this:

```
function pulse_experiment(filename ::String,
                          tE       ::Real  ,
                          QE       ::Real  ,
                          tF       ::Real  ,
                          QF       ::Real  )

    try
        existfile(filename)

        equilibration(:F, QE, tE, shutdown=false)

        monitor(filename)

        @info("Injecting...")
        @step F => COL1 => W
        @eval(@pump F = $QF)
        sleep(tF)
        @pump F = 0.0

        @info("Eluting...")
        @step E => COL1 => W
        @eval(@pump E = $QE)
        sleep(tE)
        @pump E = 0.0

        @info("Experiment Finished")
        @valves -all
        @monitor stop
    catch
        @pump F = 0.0
        @pump E = 0.0
        @valves -all
        @info("An error occurred, shutting down")
    end

    return nothing
end
```

Now our code is much more clean and understandable. Furthermore, it is much easier to be reused, hence it is easier to create other experimental protocols without rewriting everything. In this example, `existfile` is called both explicitly and inside `monitor`. The explicit call is made for time saving, since without it the file check would have been made after the equilibration step, wasting time and resources if there was an overwriting attempt. In this specific case, the explicit call comes in handy.

For pulse experiments, most always more than one injection needs to be performed in

order to verify reproducibility. To achieve this, we can make another generalization:

```
function pulse_injection(tE ::Real, QE ::Real  ,
                         tF ::Real, QF ::Real  ;
                         shutdown ::Bool = true)

    @info("Injecting...")
    @step F => COL1 => W
    @eval(@pump F = $QF)
    sleep(tF)
    @pump F = 0.0

    @info("Eluting...")
    @step E => COL1 => W
    @eval(@pump E = $QE)
    sleep(tE)
    @pump E = 0.0

    if shutdown
        @pump F = 0.0
        @pump E = 0.0
    end

    return nothing
end
```

Now the pulse injection and elution is separated into a different function and our full experiment would take the following form:

```
function pulse_experiment(filename ::String,
                          reps     ::Int64 ,
                          tE       ::Real  ,
                          QE       ::Real  ,
                          tF       ::Real  ,
                          QF       ::Real  )

    try
        existfile(filename)

        equilibration(:E, QE, tE, shutdown=false)

        monitor(filename)

        for i in 1:reps
            @info("Rep $i")

            pulse_injection(tE, QE, tF, QF, shutdown=false)

            @info("Experiment Finished")
            @valves -all
            @monitor stop
        end
    catch
        @pump F = 0.0
        @pump E = 0.0
        @valves -all
        @info("An error occurred, shutting down")
    end

    return nothing
end
```

Where `reps` is the number of replicates that we want. In the end, we created a robust pulse experiment script that is neat, understandable and reusable.

IV.2.2 The `sleep` function

Until now we used the `sleep` function to run some actions during a certain time length. It is a very simple function that serves its purpose for simple experiments. However, for experiments with a high number of small steps (like a pump gradient, for instance), this function starts to lose efficiency. This is due to the fact that inherently has an error associated that is negligible for a small number of big steps but is quite significant when we have a high number of small steps.

As a demonstration, let's take a look at the `gradient` function, which implements

a concentration gradient using two HPLC pumps, each one having a different solvent concentration:

```
function gradient(c1  ::Real, c2 ::Real,
                  ci  ::Real, cf ::Real,
                  tau ::Real, QT ::Real;
                  tst ::Integer = 100  )

    t_sw = range(0.0, tau, length=tst)

    t_step = (t_sw[2]-t_sw[1]) * 60.0
    for t in t_sw
        c  = ci + (cf-ci)/tau * t
        f  = (c - c2) / (c1 - c2)
        Q1 = abs(f*QT)
        Q2 = abs(QT-Q1)

        set_pump(pump_E, Q=Q1)
        set_pump(pump_G, Q=Q2)

        sleep( t_step )
    end

    return nothing
end
```

`c1` and `c2` are the solvent concentrations of each HPLC pump, `ci` and `cf` the initial and final concentrations of the gradient, `tau` is the time length of the gradient, `QT=Q1+Q2` is the total flowrate injected and `tst` is the number of steps in which we divide the time length, the bigger the number of steps, the smoother the gradient. `set_pump` is the function version of the `@pump` macro. Inside other functions, `set_pump` should be used in detriment to `@pump`. You can type `?set_pump` in the command prompt for more information.

To test the real time spent from our gradient function, we will use `BenchmarkTools`, a *Julia* package meant to run code and test their true speed:

```
using BenchmarkTools
c1  = 1.0
c2  = 1.0
ci  = 1.0
cf  = 1.0
tau = 0.5 # minutes
QT  = 1.0

> @btime gradient(c1, c2, ci, cf, tau, QT, tst=300)
30.659 s (1512 allocations: 46.12 KiB)
```

The last line shows the real time spent inside the function. The time length is supposed

to be 30 seconds, but the code ran for 30.659 seconds. This is a significant error, which tends to propagate the longer the experiment and the smoother the gradient. It gets even worse if we implement it in cyclic operations in which the gradient is reset every switching interval. A better solution is to create a customized sleep function:

```
function mysleep(t ::Real)
    t_ref = time()
    while time() - t_ref < t
    end
    return nothing
end
```

The `time` function gives the time, in seconds, passed since the epoch. It has a resolution in the microsecond scale, so it's quite accurate for the cases we deal with. Benchmarking the gradient function using `mysleep` we have the following result:

```
using BenchmarkTools
c1  = 1.0
c2  = 1.0
ci  = 1.0
cf  = 1.0
tau = 0.5 # minutes
QT  = 1.0

> @btime gradient(c1, c2, ci, cf, tau, QT, tst=300)
30.101 s (0 allocations: 0 bytes)
```

The error is still there but it's much smaller. Furthermore, the error propagation of `mysleep` is way smaller than `sleep`. In the end, this is just an alternative, you should do your own testing before choosing the best option for your experimental protocols.

IV.3 Programming a cyclic chromatographic process

Until now we used a pulse experiment to demonstrate how you can use the fundamental functionalities of *ChromatographyStudio* . In this section we will use what we learned to create a cyclic chromatographic process for central-cut separation. As explained in Section 3.3, The GSSR process used for the ternary separation of Alanine, Uridine and Guanosine was simplified into four steps, as shown in Figure **??**. The A stream is usually operated in gradient mode, but since the adsorption equilibrium of the three components is significantly different, operating it in isocratic mode is enough to achieve both a purity and recovery of $\sim 99\%$.

The implementations of steps 1, 2 and 4 in *ChromatographyStudio* is very straightforward. The steps would look like this:

```
@step G => COL123 => W # step 1
@step G => COL231 => W # step 2
@step G => COL312 => W # step 4
```

Step 3 is a little trickier. The reason is that, until now, the `@step` macro does not support the simultaneous injection of two pumps in two different columns. This is because the macro, when called, closes all the valves that are not part of the step. For these types of configurations we have to explicitly open some valves. In this example, the protocol to implement step 3 would be:

```
# step 3: F => COL12 => W ; A => COL3 => P
@step F => COL12 => W # the first part can be used with @step
@valves 4,19 # explicitly open valves for the second part
```

If you don't know which valves are opened when you implement a step, it's very easy to figure that out, implement `@step` with the step you want to analyze and then run `@valves get`. In this case, the output would be:

```
@step A => COL3 => P
@valves get
> OPEN VALVES: V4 , V19
```

In this way you know which valves you need to open for a certain pathway, and using `@valves` you can open them without closing all others. In the future we intend to add this functionality to `@step`.

Finally, the isocratic 4-step GSSR cycle that we used for our experiments is as follows:

```
function gssr_cyc_isocratic(tau      ::Float64    ,
                            aP       ::Float64    ;
                            test     ::Bool = true,
                            shutdown ::Bool = true)

    # 1st SWITCHING INTERVAL
    @info("1st Switching Interval")
    @info("A => COL123 => w")
    if !test
        @step G => COL123 => W ; @valves 1
    end
    mysleep(tau * 60.0)
    println("")

    # 2nd SWITCHING INTERVAL
    @info("2nd Switching Interval")
    @info("A => COL231 => W")
    if !test
        @step G => COL231 => W ; @valves 1
    end
    mysleep(tau * 60.0)
    println("")

    # 3rd SWITCHING INTERVAL
    @info("3rd Switching Interval")
    @info("F => COL12 => W ; A => COL3 => P")
    if !test
        @step F => COL12 => W ; @valves 4,19
    end
    mysleep(aP*tau * 60.0)

    @info("A => COL312 => W")
    if !test
        @step G => COL312 => W ; @valves 1
    end
    mysleep((1-aP)*tau * 60.0)

    if shutdown
        println("")
        system_shutdown(:total)
    end
    return nothing
end
```

In here you see that we open valve 1 in every elution step. In our system valve 1 is a recycling valve, which is connected to the feed tank. In this way, instead of shutting

down the feed flowrate during the elution steps, we simply direct the feed to its own tank, making a closed circuit. This allows for greater accuracy, since opening and closing pneumatic valves is much faster than shutting down a pump and then turning it back on because the flowrate increase is not instantaneous.

And finally, the full experimental protocol, including equilibration and a variable number of cycles:

```
function gssr_isocratic(filename ::AbstractString,
                        ncyc     ::Integer       ,
                        tau      ::Float64       ,
                        aP       ::Float64       ,
                        QF       ::Float64       ,
                        QA       ::Float64       ;
                        test     ::Bool = true   )
    existfile(filename)

    @valves 1; set_pump(pump_F, Q=QF)
    @valves 7; set_pump(pump_G, Q=QA)

    monitor(filename, checkfile = false)

    for cyc in 1:ncyc
        printstyled("\nCYCLE $cyc\n\n",
                    bold=true, color=:magenta)
        data_to_mon("CYCLE $cyc")
        gssr_cyc_isocratic(tau, aP;
                           test=test, shutdown=false)
        println("")
    end

    @eval(@monitor stop)

    # system shutdown
    @info("System Shutdown")
    set_pump(pump_F, Q=0.0)
    set_pump(pump_G, Q=0.0)
    @valves -all

    return nothing
end
```

In here we implement the recycling strategy also for the elution pump, using valve 7. This is only useful before the start of the experiment, since the elution pump is always injecting during the entire cycle. The `data_to_mon` function is part of the *ChromatographyStudio* software and is used to introduce data in the monitoring file. Internally all macros use this function to write the data and we can also add extra messages by using

this function. In this case we add a message at the start of every cycle, so we can easily track the start time of each cycle.

While there is no graphical user interface to make all these steps easier, there increase in flexibility and customization that this approach gives has several benefits. In the future, the goal will be to provide a guide user interface for simple experiments and some chromatographic processes, as well as still being able to hard-code some customized experiments that the user might require. *ChromatographyStudio* was, and still is, a super fun project that needs a lot of attention but also gives us priceless tools and knowledge, as well as high degrees of customization for doing whatever we want with the system.

V

Future Work

Although the software is working and suits all our needs, it still has several things that can be improved and even changed. In here we will list some improvements that are already planned.

The main one is the monitoring system. Currently, we monitor the data of all the equipments into one single file, with the data being written in bytes. When an equipment wants to write data, it writes the device code previously set by the software, the number of data bytes that will be written and then the actual data bytes. This ends in a single byte file with a significant data compression that we can later convert to human readable data. This has two issues: 1) Since all equipments write their data to the file, just one bad communication between equipment and computer can ruin all the data that comes after the bad reading. This is a problem that sometimes happens to us, specially with the S1050 pump, and 2) During the experiment the file has to be internally opened and if, for some reason, we need to stop the experiment, we lose all the data that came before the forced stop. We implemented a backup strategy, but even then we have some issues with data recovery. We want to fully reformulate data monitoring and use databases. In this way we have different tables per equipment, the actual writing to the database is very straightforward and we don't have to worry with backups because databases already do it automatically. We also will have to reformulate the entire data analysis software that we already have but that's a price that we'll gladly pay in order for monitoring in *ChromatographyStudio* to become more reliable.

The second one is the plotting system. For real-time plotting we use the *Matplotlib* written for *Python* and after integrated in *Julia* , which is a very simple plotting package that gives fast plotting capabilities and good degrees of customization. However, there is no functinality for real-time plotting, so we had to come up with workarounds for it to work, which are not very efficient and might actually fail if we need fast UV readings. The plan is to fully move to use *Makie*, a plotting package written in *Julia* that has simple real-time implementation and higher degrees of flexibility, as well as very efficient and compilable code, which makes it tempting for computers with slow processing power. With this change we plan on having shorter and more concise code.

Last but not least, is the user interface. Until now we have no type of user interface and still haven't planned to start it. This would be a much bigger effort and we still have no planned strategy to implement it. Unfortunately it's not a thing that we will do in the near future, but it's always on our minds, and if someone wants to help us create one user interface as a master thesis, for example, we are gladly open to suggestions.