Towards autonomous antibody purification

Start-up → Step

Load → Wash → Elution → Regeneration → CIP → Close down
Master Thesis

Towards autonomous antibody purification

by

Frida Heskebeck

Department of Chemical Engineering
Lund University
Sweden
June 5, 2019

Supervisor: Doctor Niklas Andersson
Examiner: Professor Bernt Nilsson

Front page picture: PCC process scheme by Frida Heskebeck

Postal address
PO-Box 124
SE-221 00 Lund, Sweden

Visiting address
Naturvetarvägen 14

Web address
www.lth.se/chemeng

Telephone
+46 46-222 82 85
+46 46-222 00 00
Preface

The report which you are currently reading is the result of my master thesis. It is the final project of my education for a Master of Science in Engineering in Biotechnology.

The overall idea is that antibody production should be autonomous. In simple words, it means that we should be able to click the start button for the process and then to go home.

The project consisted of two parts, integrating the ÄKTA pcc system to the Orbit software and developing a buffer machine. The report follows the traditional layout with background, materials and methods, results and discussion, and finally, conclusions and future. Both parts of the project are discussed in each section of the report.

In the later years of my education, I have profiled myself towards bioprocess engineering, programming, and automatic control, the three areas that are the central core of this thesis. Antibodies are produced by organisms in a bioprocess, and purification is necessary before the antibodies can be used in an application. The integration of the PCC system in Orbit is essentially programming in Python. Finally, while designing the buffer machine, I got to use my knowledge in automatic control.

I hope that you enjoy reading this report as much as I have enjoyed doing the project!

Best wishes,
Frida Heskebeck
Lund, 31st of May 2019
Acknowledgement

First of all, I want to thank my examiner professor Bernt Nilson and my supervisor Niklas Andersson, who have helped and inspired me throughout my master thesis. I would also like to thank the rest of the department for welcoming me with open arms, we have shared many laughs in the "fika"-room.

A special thanks to my husband Erik, whom I married just a month ago, for being there for me every day, supporting and caring, and for proofreading this thesis.

Finally, I want to thank my family for supporting me my whole life. Without you I would not be where I am today.

Thank You!
Abstract

The vision of antibody purification is to have the whole process fully automatic. It is often desirable to have a continuous process since that reduces the costs for production and the size of the used equipment. It is necessary to supply buffer solutions to the purification steps, and as part of the vision; this should be done automatically. An in-house piece of software called Orbit is used for advanced control of ÄKTA chromatography systems from GE Healthcare, that are used in antibody purification.

An ÄKTA pcc system can be used for continuous processes in the first purification step, which is a capture step based on affinity chromatography. The PCC system has four columns that are used simultaneously; two are loaded continuously, one is eluted, and one is regenerated. The first part of this master thesis was to integrate the ÄKTA pcc system into the Orbit software. The second part was to design a PCC process to demonstrate that Orbit can control the PCC process. Experiments and simulations were done to verify the functionality of the PCC system in Orbit. In the future, the ÄKTA pcc system can be used together with other ÄKTA systems in an antibody purification process where all systems are controlled from Orbit.

The final part of this master thesis was a proof of concept of automatic buffer production. A buffer machine that produces a buffer solution by diluting stock solutions and adjusting pH was developed. The buffer machine was designed to be able to deliver the produced buffer solution to another system, in line with the overall vision for antibody purification.
Sammanfattning

Visionen för upprening av antikroppar är att processen ska vara helautomatisk. Ofta eftersträvas en kontinuerlig process eftersom det i de flesta fall innebär lägre kostnader för produktionen och att utrustningen kan vara mindre. Det är nödvändigt att tillföra mer buffertlösning under processens gång, och som en del av visionen ska även detta göras automatiskt. Orbit är en mjukvara som används av forskargruppen för avancerad kontroll av ÄKTA kromatografisystem från GE Healthcare, system som ofta används för antikroppsupprening.


Sista delen av examensarbetet var att visa att konceptet av en automatisk produktion av buffertlösningar är möjlig. En buffertblandare som producerar buffertlösningar genom utspädning av stamlösningar och därefter pH-justering skapades. Buffertblandaren var designad att kunna leverera buffertlösningar till ett annat system, i linje med visionen för antikroppsupprening.
Popular science summary

Antibodies are a natural part of the immune system and can, for example, be used in pharmaceuticals. They are produced in bioproduction from living organisms and need to be purified before they are used. The vision is that antibodies should be automatically and continuously produced. When antibodies are purified, a lot of buffer solutions are needed. As part of the vision, buffer solutions should be delivered automatically to the antibody purification steps.

Affinity chromatography is a technique where the product of a sample purified by being captured in a column. When all impurities are washed away, the product is released again and have by then been purified. This technique is used in the first antibody purification step, called the capture step. An ÄKTA pcc system from GE Healthcare uses four columns simultaneously to do the capture step continuously. Two of the columns are loaded with product, one is releasing the product it has captured, and one is cleaned to be ready to capture new product. In this master thesis, the ÄKTA pcc system was integrated into a software called Orbit. Orbit is used for advanced control of chromatography systems from GE Healthcare. A PCC process was designed in Orbit, and experiments and simulations were done to demonstrate that Orbit can control the PCC system. The ÄKTA pcc can in the future be used together with other system controlled by Orbit to create a whole automatic antibody purification process.

As proof of concept of automatic buffer production, a buffer machine was developed. The buffer machine produced a buffer solution by diluting stock solutions and adjusting pH. Experiments were done to test the buffer machine and to make the design of the process better. The machine was designed to be able to deliver the buffer solution to a system that is part of an antibody purification step, in line with the overall vision.
Populärvetenskaplig sammanfattning


Som ett bevis av konceptet att automatiskt kunna producera buffertlösningar skapades en buffertblandare. Buffertblandaren producerade buffertar genom att späda stamlösningar och justera pH. Experiment gjordes för att testa buffertblandaren och för att förbättra processdesignen. Buffertblandaren var designad för att kunna leverera buffertlösningar till andra system som är del av uppreningen av antikroppar, i linje med visionen för antikroppsupprening.
## Contents

1 Introduction ............................................. 1  
   1.1 Overview ........................................... 1  
   1.2 Aim ................................................... 1  

2 Background .............................................. 2  
   2.1 Chromatography basics ............................... 2  
   2.2 Periodic counter-current chromatography .......... 3  
   2.2.1 ÄKTA pcc .......................................... 3  
   2.3 Buffer need ........................................... 7  
   2.4 Orbit software ....................................... 7  

3 Materials and Methods ................................. 10  
   3.1 Orbit software ....................................... 10  
   3.1.1 System configuration ............................. 10  
   3.1.2 Process design .................................... 11  
   3.1.3 Simulation ........................................ 13  
   3.2 Laboratory work ...................................... 14  
   3.2.1 PCC system ....................................... 14  
   3.2.2 Buffer solutions .................................. 14  
   3.2.3 Raw material ...................................... 15  
   3.2.4 Filter station ..................................... 15  
   3.3 Buffer machine ...................................... 15  
   3.3.1 System setup ...................................... 16  
   3.3.2 Process design .................................... 18  
   3.3.3 Usage of buffer machine ......................... 20  

4 Result and Discussion ................................ 21  
   4.1 Orbit implementation ............................... 21  
   4.2 Dynamic control of loading ....................... 22  
   4.3 Dynamic control of elution ....................... 24  
   4.4 Simulation .......................................... 24  
   4.5 Long run ............................................ 27  
   4.6 Buffer machine ...................................... 29  

5 Future Work ........................................... 32  
   5.1 PCC system .......................................... 32  
   5.2 Buffer machine .................................... 32
6 Conclusion
   6.1 PCC system .................................................. 34
   6.2 Buffer machine .............................................. 34

References .......................................................... 35

A Unit library .................................................... 37
B PCC script file .................................................. 41
C Buffer script file ............................................... 47
1 Introduction

1.1 Overview

When using a bioprocess for production, downstream processing is essential for product extraction. One part of the downstream process is usually a chromatographic step (Doelle et al. 2009). Chromatographic processes are often run in a batch mode. Though today, it is becoming more popular to run them in a continuous mode since this often is more economically feasible due to smaller equipment and faster processes (Steinebach, Müller-Späth, and Morbidelli 2016). GE Healthcare provides an ÄKTA pcc system for running continuous chromatographic processes. In the ÄKTA pcc system, all phases are run in parallel, phases such as loading, elution, and regeneration. GE Healthcare also provides the Unicorn software that is used to control the system (Bio-Sciences 2016).

One part of the research done by the research group led by Bernt Nilsson aims to make the downstream process in bioproduction fully automatic. Orbit is a software developed within the research group that allows advanced control of ÄKTA pure and ÄKTA explorer systems. One of the advantages of Orbit compared to Unicorn, is that it easily allows running multiple phases in parallel on one system (Andersson 2018). Hence, it would be favorable to use the Orbit software to control the continuous ÄKTA pcc system, since it requires multiple phases running in parallel. Even though Orbit automatically controls the systems, one obstacle remains before the whole downstream process is fully automatic: automatic production of buffer solutions is not yet possible.

1.2 Aim

The project aims to make it possible to control an ÄKTA pcc system with Orbit, by extending the Orbit software, and to investigate the possibility of automatically mixing buffer solutions.
2 Background

Production of biopharmaceuticals or other products from a bioprocess requires downstream processing for purification (Doelle et al. 2009; Steinebach, Müller-Späth, and Morbidelli 2016). The common way to do it is by having the following process steps in sequence: Cell culture harvest, Capture, Viral inactivation, Polishing, Virus filtration, UF/DF, and Formulation. The capture step and the polishing step are chromatographic steps, where the capture step removes most of the impurities, often by affinity chromatography (Diefenbach-Streiber et al. 2010; Steinebach, Müller-Späth, and Morbidelli 2016). Continuous chromatography is widely used to increase efficiency and reduce costs. Periodic counter-current chromatography (PCC) is one way of running the capture step continuously. Compared to batch mode, PCC allows a higher loading of the column and consumes less buffer solution. However, the PCC process is more complicated than the batch process (Svedberg 1976; Mahajan, George, and Wolk 2012; Steinebach, Müller-Späth, and Morbidelli 2016). The capture step can also be run continuously with a simulated moving bed process (SMB) for example, stepSMB or captureSMB (Grabski and Mierendorf 2009; Angarita et al. 2015; Steinebach, Müller-Späth, and Morbidelli 2016). The product to be purified in this project was monoclonal antibodies.

2.1 Chromatography basics

The basic idea with chromatography is to separate the components of a sample into different fractions, based on size, affinity, charge, or hydrophobicity (Wilson 1940; Doelle et al. 2009). A packed column is used for the separation. Fixed inside the column is the stationary phase, and moving through the column is the mobile phase. For liquid chromatography, the mobile phase is liquid, and the stationary phase is a solid porous material. The sample moves through the column, with the mobile phase, and interacts with the stationary phase. Each component of the sample has a different interaction with the stationary phase. More interaction leads to a longer time for the component to go through the column. The separation is a result of
these different interactions with the stationary phase, hence the different times for elution (Jandera and Henze 2011). The time it takes for a compound to elute, go through the column, is called retention time (IUPAC 1997).

Affinity chromatography is used for separation of monoclonal antibodies. The basics of affinity chromatography are that the column is filled with molecules that capture the target molecule. The molecules that are in the column are called ligands and are attached there, immobilized to the column. The target molecule is adsorbed in the column when it is captured by a ligand. The target molecule is reversibly adsorbed to a ligand, which means that the target molecule can be released again, and be desorbed from the column (Cabrera, Brünner, and Müller 2011). The antibodies are adsorbed to a protein A ligand and are desorbed by lowering the pH (Deisenhofer 1981; Bio-Sciences 2007).

### 2.2 Periodic counter-current chromatography

In PCC, multiple columns packed with the same stationary phase are connected in series. The loading feed enters the first column, any product that escapes the first column enters the next and may be captured there instead. As a result, almost all capacity of the columns is used (Svedberg 1976; Mahajan, George, and Wolk 2012; Steinebach, Müller-Späth, and Morbidelli 2016). An ÄKTA pcc system from GE Healthcare can use either three or four columns in parallel. Two columns are loaded, and for three columns PCC (3C-PCC), the third column is eluted and regenerated. As for the four columns PCC (4C-PCC), the third column is regenerated, and the fourth column eluted. To optimize the overall process, the process for each column should take approximately the same time (Bio-Sciences 2016; Bio-Sciences 2018).

#### 2.2.1 ÄKTA pcc

In this project, the 4C-PCC was used. One cycle has four steps. In step 1, the first column is the primary loading column, the second column is the secondary loading column, the third column is the regenerating column, and lastly, the fourth column is the eluting column. In the following step, step 2, the columns shift position. The one that was the primary loading column is changed to be the eluting column, the one that was the secondary loading column is changed to be the primary loading
column, the one that was the regenerated column is changed to be the secondary loading column, and finally, the one that was the eluted column is changed to be the regenerated column. The same principle goes for step 3 and 4, resulting in that all columns have had all positions when one cycle is completed (see figure 2.1) (Bio-Sciences 2016; Bio-Sciences 2018).

**Figure 2.1** – Schematic picture of the four steps in a 4C-PCC cycle. Two columns are loaded (red), one is regenerated (black) and one is eluted (yellow). For each step in the cycle the columns shift position (remade picture from: Bio-Sciences 2016).

Before the elution step, a post load wash is performed to reduce product loss. The post-load wash washes the unbound product in the elution column to the secondary
Figure 2.2 – Schematic picture of post-load wash (left) and the following main step (right), corresponding to step 2 in figure 2.1. Valves used to change the flow path to the columns are depicted as gray circles, sensors used to control and/or observe the process are depicted as white squares with text specifying what sensors are connected. Inlet valves, pumps and fraction collector are not depicted (Bio-Sciences 2017b).

loading column. After the wash, the flow paths are changed to the following main step (see figure 2.2). (Bio-Sciences 2016; Bio-Sciences 2017b; Bio-Sciences 2018).

The ÄKTA pcc system has three flow paths. One from the sample pump, loading the columns, one from pump A, washing and eluting the columns, and lastly, one from pump B, cleaning in place (CIP) and regenerating the columns. Each pump has an inlet valve to select the buffer solution to use. The valves called “column in valve” and ”column out valve” change the flow path through the columns. The outlet valve directs the sample flow path and flow path A, either back to the column valve, to waste, to a specific outlet for collection or to the fraction collector (see figure 2.2) (Bio-Sciences 2016; Bio-Sciences 2017b; Bio-Sciences 2018).

The sensors in the ÄKTA pcc system are placed in a pathway after a column, rather than after a specific column. The system has conductivity sensors and pH sensors after both the eluted column and the regenerated column. Four UV sensors are available (see figure 2.2) (Bio-Sciences 2016; Bio-Sciences 2017b; Bio-Sciences 2018):

- **UV sample** measures the UV absorbance in the sample solution before it enters any column.
• **UV breakthrough** measures the UV absorbance after the primary loading column. At the beginning of the loading phase, the column captures all antibodies, and this sensor only measures the impurities in the sample. With time, more antibodies escape through the column, and as a result, the UV breakthrough absorbance increases.

• **UV flowthrough** measures the UV absorbance after the secondary loading column. It should in a well-tuned system, only measuring impurities since no antibodies should escape the secondary loading column.

• **UV elution** measures the UV absorbance after the eluted column. Its signal is used to find and fraction peaks during elution to gather the product.

A breakthrough curve displays how the product is absorbed in the column. Since the columns in a PCC system switch place, both the UV breakthrough and the UV flowthrough signals have to be studied to get the full breakthrough curve. The breakthrough curve starts at zero since the column is empty when the loading is started. The UV signal from impurities in the sample is visible after a short while since the impurities merely flow through the column. The baseline value corresponds to the UV signal of impurities. As the column is filled, more and more product flows through the column and the breakthrough curve eventually reaches the UV sample signal (see figure 2.3) (Bio-Sciences 2016; Bio-Sciences 2018).

![Figure 2.3](image) – The breakthrough curve is seen from the UV breakthrough and from the UV flowthrough (Bio-Sciences 2016).

The process has specific start-up and close down sequences. During the first step of the 'start-up', only loading is done. In the next step, loading and elution are done.
Then the following steps are done as described above (see figure 2.4). The first step in ‘close down’ is a wash phase of the latest loaded columns. The next two steps are elution and regeneration of the not yet eluted and regenerated columns. The final step is a regeneration of the final column (see figure 2.5) (Bio-Sciences 2018)

![Schematic picture of start-up of the PCC process.](image)

**Figure 2.4** – Schematic picture of start-up of the PCC process.

### 2.3 Buffer need

Even though a PCC process consumes less buffer solutions than a process in batch mode, buffer solutions are needed (Mahajan, George, and Wolk 2012). The common way to prepare a buffer solution is to make it by hand, which means to weight salts, measure volumes, and to adjust pH. It is a procedure that consumes surprisingly much time. The PCC system is designed to run for a long time, meaning that a lot of buffer solution will be consumed. The production of buffer solution needs to be automatized to make the overall process fully automatic. Ge Healthcare has one solution where buffers are produced in-line. The flow of different stock solutions is adjusted to the concentration of the final buffer solution (Bio-Sciences 2017a).

### 2.4 Orbit software

The conventional way to control ÄKTA systems is with the Unicorn software that GE Healthcare provides. The software has some limitations, e.g., it is hard to run
Figure 2.5 – Schematic picture of closedown of the PCC process.
complex scripts such as having multiple phases in parallel, which is the reason why the Orbit software was developed at the Department of Chemical Engineering at the Faculty of Engineering (LTH). Orbit is programmed in the object oriented programming language Python and controls the ÄKTA system via Unicorn. API and OPC are two ways to connect Orbit to Unicorn. In either case, special communication links are used to control the units in the ÄKTA system. Orbit is constructed from many classes, e.g., for the system, for units in the system and for the process. Within these classes, the communication links to Unicorn are specified (Andersson et al. 2017; Löfgren et al. 2018; Andersson 2018).

The system file defines the units that are part of the system. Precisely what each unit can do is defined in its class in the unit library. Typically, a unit class contains methods to read or set values for that unit, for instance, flow rate, UV signal, or valve position. When the system file is set up, the next step for running a process is to create a process file and a script. A process consists of different phases, e.g., washing, loading, or regeneration. For each phase, the system should have different settings such as flow rate or valve positions. The script file defines the phases of the process and their order of execution. The process file defines the exact instructions for the phase execution (Andersson 2018).
3 Materials and Methods

The materials and methods are divided into three sections, reflecting the order of which they were performed:

- Orbit software - Programming of the PCC system.
- Laboratory works - Everything done in the lab regarding the PCC system.
- Buffer machine - The development of a system that automatically mixes buffer solutions.

3.1 Orbit software

The first part of programming, system configuration, included making it possible to control the ÅKTA pcc system from the Orbit software. The second part, process design, involved creating the script and the process file for running the PCC process. The third part, simulation, was about finding models for relevant signals and use them to study how the system behaved over a long time in a simulation.

3.1.1 System configuration

The units were implemented as new classes in the unit library. Similar units shared one class, to reduce the amount of code. For example, pump A, pump B, and pump Sample all shared one class. The supervisor gave the connection links for OPC and API connection. A list of the implemented classes and their methods can be found in appendix A. The general process file was extended with some method calls for the PCC system, to enable logging of data during a run.

Commands were sent to the PCC system from the python command window, to verify that Orbit could control the ÅKTA pcc system. A saltwater solution and blue dextran solution were used to get signal for conductivity and UV absorbance.
3.1.2 Process design

In the script file, phases were created for loading, washing, elution, cleaning in place (CIP), and regeneration. A cycle was completed when all four steps, discussed in section 2.2.1 had been done. Each step started by setting the column valves to the correct positions, then the process was branched. One branch contained the loading phase, one branch the wash phase and elution phase, and one branch the CIP phase and regeneration phase. When all branches were done, they merged, and a new step was initiated. The start-up and close down both had separate phases (see figure 3.1). Each branch controlled one of the pumps.

Figure 3.1 – Schematic picture of script structure. After setting the step, three branches of phases were created, one for each flow path. When all branches were done, they merged, and a new step started. The process had special phases for start-up and close down.

In the process file, the details to perform the phases were coded. In the first trial, the phases were controlled only by a time limit. In the final version, dynamic control of the loading phase and the elution phase was implemented.

Dynamic control of the loading phase was based on the $\Delta$UV value, calculated by the ÅKTA pcc system (see equation 3.1 and figure 3.1). The loading phase was stopped when the $\Delta$UV value reached 70 %. A time limit stopped the phase if the control with $\Delta$UV value did not work, for example, if the sample solution had a
low concentration of the product. The signals for UV breakthrough and UV sample was measured continuously during the run. The PCC system provided four ways of baseline search: \textit{search once}, \textit{continuous search}, \textit{set value}, and \textit{set to current value}. The system found a baseline when the UV signal for UV breakthrough had not changed more than a given value (5 mAu in these experiments) during a given time (5 min for the experiment with continuous baseline search and 10 min for all other experiments). The \textit{continuous search} mode was used for every step in the first experiment. The \textit{search once} mode was used for every step in the rest of the experiments.

\[ \Delta UV = \frac{UV_{\text{breakthrough}} - \text{Baseline}}{UV_{\text{sample}} - \text{Baseline}} \]  

\textbf{Figure 3.2} – Delta UV visualized in a breakthrough curve (Bio-Sciences 2016).

The dynamic control of the elution phase was based on the UV elution signal, both for collecting peaks and for stopping the eluent flow. The collection of peaks was done with a built-in method of the ÄKTA pcc system. The method detected peaks and directed them to a specified outlet. The peaks could also be collected in the fraction collector. The stopping of eluent flow was done with a pooling loop that detected when a peak started and when a peak ended. A time limit stopped the flow if no peak was detected.
The full script file of the PCC process can be found in appendix B.

### 3.1.3 Simulation

The process was simulated to verify stability and functionality over time before a long-running experiment was done. It was only the loading and elution phases that were dynamically controlled, hence only the signals used in those phases were simulated (UV sample, UV breakthrough, and UV elution).

The signal for UV sample was simulated as a constant value.

The signal for UV breakthrough was simulated with a polynomial function (see equation 3.2). The parameter $time$ [min] corresponds to the current time, $timeStart$ [min] to the starting time of the interval, $top$ [mAu] to the height of the peak at the end of the interval, $interval$ [min] to the time of the interval and finally, $baseline$ [mAu] to the baseline value. To fit the model to experimental data, the parameters in the model where adjusted.

\[
UV = (time - timeStart)^4 \times \frac{top}{interval^4} + baseline \tag{3.2}
\]

The UV elution signal was simulated with an adapted lognormal distribution curve (see equation 3.3) (Limpert, Stahel, and Abbt 2001). The parameter $height$ [mAu] corresponds to the height of the peak, $width$ [-] to the shape of the curve, $time$ [s] to the current time, $timeStart$ [s] to the time when the peak start, and finally, $timePeak$ [s] to the time of the top of the peak. By adjusting the parameters, the model was fitted to experimental data.

\[
UV = height \times \exp \left( - \frac{(\ln(time - timeStart) - \ln(timePeak))^2}{2 \times width^2} \right) \tag{3.3}
\]

The models for UV sample, UV elution and UV breakthrough were integrated into the process file and the PCC process was simulated for ten cycles. During the simulation, variations in the UV breakthrough model were generated by normalized
random values for the parameters: \textit{top} (mean = 70, std = 4), \textit{interval} (mean = 105, std = 20) and \textit{baseline} (mean = 100, std = 10) (see equation 3.2). Variations in the UV elution signal were generated by normalized random values for the parameters: \textit{timePeak} (mean = 5.4, std = 10.2), \textit{height} (mean = 600, std = 150) and \textit{width} (mean = 0.45, std = 0.04) (see equation 3.3).

### 3.2 Laboratory work

First, one experiment with one cycle and continuous search mode was done. A second experiment with one cycle and search once mode was done. A third experiment with six cycles and search once mode was done.

#### 3.2.1 PCC system

A standard connected ÄKTA pcc from GE Healthcare (Uppsala) was used in this study (see figure 3.3). The Orbit software was used to control it.

![Figure 3.3 – The ÄKTA pcc system used in this thesis.](image)

#### 3.2.2 Buffer solutions

A phosphate-buffered saline (PBS) buffer solution (8 g/l NaCl, 0.2 g/l KCl, 1.44 g/l Na$_2$HPO$_4$, 0.24 g/l KH$_2$PO$_4$, pH = 7.4) was used for washing and regeneration. A
sodium hydroxide solution (0.1 M) was used for cleaning in place. A sodium acetate buffer solution (50 mM, pH = 3.5) was used for elution.

### 3.2.3 Raw material

GE Healthcare supplied the antibody solution. It was stored in a freezer at -20°C to reduce bacterial growth, hence before use, it was necessary to thaw the solution. The solution was filtered with the filter station (see section 3.2.4) right before running the process to remove the bacterial growth and other particles that made the solution opaque. The solution was kept in a water bath to not heat it with the heat from the PCC machine while running a process.

A solution with purified antibodies, blue dextran, and PBS buffer solution was used as a substitute for the antibody solution in the run with six cycles to reduce the use of the antibody solution. The blue dextran mimicked the impurities in the antibody solution by giving a UV signal without binding to the column. The antibodies bound to the column as in the real antibody solution. The PBS buffer solution was used since it was the buffer used for the regeneration of the column and should not interfere with the antibodies binding to the column.

### 3.2.4 Filter station

A filter station was built to facilitate filtering of the antibody solution. A peristaltic pump was used to pump the unfiltered antibody solution to a tangential flow filter. The permeate contained the antibodies and was collected. The retentate contained the impurities and was recirculated back into the unfiltered solution (see figure 3.4). The filter was cleaned with distilled water and sodium hydroxide (0.5 M).

### 3.3 Buffer machine

100 ml of a sodium acetate buffer solution (50 mM, pH = 3.5) was made in the buffer machine experiments from a stock solution with sodium acetate (500 mM). The design of the process was improved successively, and it is the final design that is described below. Improvements that were done to the process between the first and the last experiment are described and discussed in section 4.6. About eight
Figure 3.4 – Filter station. (a) Schematic picture of setup. (b) Real setup. The antibody solution can be seen to the left, the purified antibody solution to the right, and the pump and the filter in the middle.

experiments were done. Before each experiment, multiple simulations were done to test the logic in the process design.

3.3.1 System setup

The buffer machine was set up with two modified ÄKTA explorer systems. The gradient pumps of the systems were split, resulting in four individual pumps. Pump A in system one (pump A1) was used to pump stock solutions into the buffer container, via the sample valve. The sample valve was used for pump wash of pump A1. Pump B in system one (pump B1) was used to pump the buffer solution from the buffer container through the pH sensor and back to the buffer container again. The outlet valve made it possible to do pump wash of pump B1 and could be connected to another system to pump the buffer solution directly to the next system. During pH control, pump A in system two (pump A2) was used to add acid, and pump B in system two (pump B2) was used to add base. Mixing was done with an external magnetic stirrer (see figure 3.5).
Figure 3.5 – Buffer machine. (a) Schematic picture of setup. The pumps for system one was denoted A1 and B1. The pumps for system two was denoted A2 and B2. (b) Real setup with two ÄKTA explorer systems. Mixing was done with an external magnetic stirrer.
3.3.2 Process design

The Henderson-Hasselbalch equation is widely used to calculate the pH of buffer solutions (Henderson 1908; Po and Senozan 2001). The design of this buffer machine was based on the dilution of already existing stock solutions, where the pH could be adjusted.

The buffer process started by adding all stock solutions. If no pH control should take place, the remaining water was added. If pH should be controlled; first, about 90% of the water was added, then, the pH was controlled, and finally, the remaining water was added (see figure 3.6).

![Buffer process diagram]

Figure 3.6 – Shematic picture of buffer process.

The volume of stock solution to add was calculated from the stock concentration, the target concentration, and the final volume (see equation 3.4). The volume was then translated to time, and the solution was added for this time (see equation 3.5).

\[ C_1 \times V_1 = C_2 \times V_2 \]  

(3.4)
The reason for only adding 90% of the water before the pH control loop was that the added acid and base also added up to the final volume, and it was not known beforehand how much acid and base that would be needed for pH control. The pH control loop started by measuring the pH value. If the pH value was above the target pH, acid was added, if the pH was below, base was added, and if it had reached the target pH, the remaining volume of water was added, and the buffer was done. pH usually takes a few moments to stabilize after addition of acid or base. Because of this, the pH was measured two times to check if it was stable before more acid or base was added. If the pH was not stable after two measurements, the measurements of pH continued until a stable pH value was found (see figure 3.7). A larger volume of acid and base were added if the measured pH was far from the target pH. Only droplets of acid and base were added if the measured pH was close to the target pH or if the target pH had been passed.

The full script file for the buffer system can be found in appendix C.
3.3.3 Usage of buffer machine

It is essential that all pumps and tubes in system one have been washed and filled with water when starting the system. In system two, it is necessary that pump A and B have been washed and filled with acid, respectively base.

The user has to set the concentration of the stock solutions, the target concentration, and the final volume of the buffer solution before starting a run. The pH is set as a range, $\text{pH} \pm \text{pH margin}$. 
4 Result and Discussion

It was possible to run the ÄKTA pcc system from the Orbit software, and it was possible to design an off-line automatic production of buffer solution. The result and discussion part is divided into the following sections which cover:

- Orbit implementation - Implementation of the ÄKTA pcc system’s units and design of the PCC process.
- Dynamic control of loading - Implementation of dynamic control of the loading phase in the PCC process and experimental PCC runs with one cycle.
- Dynamic control of elution - Implementation of dynamic control of the elution phase in the PCC process.
- Simulation - Simulation of signals in the PCC process and a simulation of a PCC process with ten cycles.
- Long run - Experimental PCC run with six cycles.
- Buffer machine - Development of the buffer machine.

4.1 Orbit implementation

The units in the ÄKTA pcc system were implemented in the unit library and tested one at the time. The only unit methods that could not be implemented were getting the pH values with API connection, meaning that pH could not be used to control the process and could not be logged if the API connection was used.

New classes were created for the ÄKTA pcc system’s units in the unit library. They could have been included in existing unit classes for the ÄKTA pure or ÄKTA explorer systems, but the communication link was different for a unit depending on in what system it was used. If all similar units for all systems were combined in one class, there would need to be some system tag which decides what communication link that should be used.
The script and the process file were successfully developed. The easiest way to control the process was by setting a time limit for each phase. The downside of this approach was the loss of dynamic control during the load phase and the elution phase, which could be done with the Unicorn software. Hence, dynamic control was implemented, see further discussion in section 4.2 and 4.3.

If one branch in the process was done before the others, it waited to be merged before the next phase was started. If two of the branches are waiting most of the time, one should consider using a 3C-PCC instead of a 4C-PCC. If a 3C-PCC were to be used, only two branches would be needed, one branch for the loading phase (sample pump) and one branch for the wash, the elution, and the regeneration phase (pump A and pump B).

With the implemented units and their methods in Orbit, it was possible to design a PCC process. Depending on the use of the system, other designs of the process than the ones presented in this master thesis could easily be programmed in Orbit. The easiness to change the process is one of Orbits advantages compared to the Unicorn software.

### 4.2 Dynamic control of loading

The loading was dynamically stopped when the primary loading column had been loaded to a given percentage of maximum capacity, calculated as $\Delta UV$, discussed in section 3.1.2.

It turned out that the system found new baselines all the time during the loading steps in the first experiment where the *continuous search* was used. The loading should be stopped when the $\Delta UV$ value reached 70%, which never happened since the baseline was changed all the time. Instead, the loading was stopped when the time limit was reached, resulting in overloaded columns (see figure 4.1a). *Baseline search once* was used instead of *search continuous* in the second experiment to solve this problem. Then only one baseline was found in each loading step, and the $\Delta UV$ value reached 70% as it should, meaning that the loading could be controlled dynamically (see figure 4.1b). Another solution could have been to study and fine-
tune the parameters for the baseline search. No fine-tuning was done since the aim of this project was to study if the PCC process could be controlled by Orbit, and not to develop an optimized process.

In the second experiment where the search once mode was used, occasionally no baseline was found at the beginning of a loading step, and instead, an incorrect baseline was found at the end of the loading step. The unsuccessful findings of a baseline could be a result of incorrect search parameters. It could also be as a result of one column having a lower binding capacity than the others and as a result, never being stable enough at the beginning of the loading step for a baseline to be found (see figure 4.1b). In other experiments (see section 4.5) it was observed for the same column that it had a lower binding capacity than the others, indicating that this was a probable reason. To prevent overloaded columns due to unstable UV signals, the ∆UV value was calculated from the previous baseline as long as no new baseline was found (see equation 3.1). The loading was stopped if this estimated ∆UV value was above the target ∆UV value.

If the sample solution would have had the same composition all the time, it would be enough to find only one baseline at the beginning of the process and use that for all ∆UV calculations. By finding a new baseline in every step, the process is adaptable to changes in the sample solution. The method of using the previous baseline for ∆UV calculations only works if the sample composition does not change.

**Figure 4.1** – Different behaviors of baseline search. (a) The first experiment, continuous baseline search used for every step. The step-shaped baseline is a result of new baselines being found throughout the loading steps. No correct ∆UV found. (b) The second experiment, search once was used for every step. One baseline found for every loading step and an expected behavior of ∆UV observed.
dramatically between two steps. If there were a dramatic change between two steps, one would have to fine-tune the search parameters and consider lower the time limit, to prevent overfilled columns. It should be noted that the PCC process is designed to fill the primary loading column more than in standard chromatography. The problem arises when the secondary loading column gets overloaded and starts to leak product.

Whether the continuous search mode or search once mode (or other baseline search options) should be used depends on the experiment that is to be run. In either way, the design is done in the process file and script file, meaning that the implementation of units and communication with the Unicorn software in the unit library are independent of the choice.

4.3 Dynamic control of elution

The built-in method to collect peaks took the dead volumes in the system into account when collecting fractions. Hence, the flow was required to remain turned on, until the product was transported to the outlet. Only fractions to the outlet were used during the experimental runs in this project, though it should not be a problem to direct the fractions to the fraction collector instead.

The pooling loop that dynamically controls the stopping of the eluent flow was designed to detect when a peak started and when a peak was done. It was only used to stop the flow after a peak was found, but it could have been used to collect peaks if the built-in method had not been used. The eluent flow was stopped after the peak was collected to reduce the amount of used buffer solution.

4.4 Simulation

The signal for the UV sample during the experimental long time run was quite stable. Therefore, it was reasonable to simulate it as a constant value (see figure 4.5a).

The model for UV breakthrough was compared with the UV breakthrough signal
from the second experimental run. The model corresponded to some of the experimental loading peaks, while for other experimental loading peaks it only captured the start and end points (see figure 4.2a). Though, to use the model for simple simulations, this was enough. The model was easily adjusted to the experimental data by changing the parameters of the model (see equation 3.2). The normalized random values for the model, as described in section 3.1.3, gave simulated UV breakthrough signals with slightly different baselines and time periods before reaching a target ∆UV of 70 % (see figure 4.2b).

The model for UV elution was compared with the UV elution signal from the second experimental run. The simulated UV elution peaks could be fitted precisely to the experimental data. The peaks appeared as pairs, the first peak was from the wash phase and contained impurities, the second peak was from the elution phase and contained the product. The last two peaks are not in a pair since they are from the 'close down' phase of the system where no wash phase was run (see figure 4.3a). Simulation with normally distributed values for the parameters as described in section 3.1.3 gave elution peaks with different height and slightly different shapes and time periods for elution (see figure 4.3b).

The simulation of ten cycles with variations in the UV breakthrough and UV elution
signals as described in section 3.1.3 showed that the process could adjust to deviations in the process. Under the assumption that the system is connected to other downstream processes with a direct connection to the production tank, deviations in the signals could be as suggested below:

- **Baseline** - Higher or lower concentration of impurities and product in the sample.
- **UV breakthrough** - Different loading capacities of the columns or different amount of product in the sample. Different loading capacities could be from manufacturing, or from products or impurities that are stuck in the column.

---

**Figure 4.3** – UV elution signals. (a) Experimental UV elution signals from the second experiment and simulated UV elution signals. The model has a good fit to the experimental data, the peaks from the model are overlapping the experimental peaks. The lower subplots in the figure are zoomed in versions of the peaks in the top subplot, starting with peak 1 to the left. (b) Distribution of simulated UV elution signals with normal distributed parameters as described in section 3.1.3.
- UV elution - Different amount of product loaded to the column or ligands in different columns binding the product with different strength, making desorption of the product slightly different.

Other reasons might also be possible.

It is concluded from the simulation that the process can adapt to these possible deviations in a run over a long time (see figure 4.4). The stopping of the elution flow can be seen by studying the log of the process (not shown).

**Figure 4.4** – Simulation of a process with ten cycles. The UV breakthrough and UV elution signals have variations between the steps as described in section 3.1.3. The variations in the UV breakthrough signal can be seen as different baselines, height, and width of the peaks. The process adjusted the loading phase to the different times for reaching target ∆UV. The phase was stopped if the target ∆UV was not reached within the time limit of the phase. The variations in the UV elution signal can be seen as different heights of the peaks. The differences in peak width are visible when the plot is zoomed in.

### 4.5 Long run

The UV sample signal was more or less constant during the whole six cycle run. Only one column was loaded until 70% ∆UV in every cycle, and the time limit stopped the loading phases for the other columns. This indicates both that one column had a lower loading capacity than the others and that the time limit was too short concerning the concentration in the sample solution. Though, from this run, it can be concluded that the system was able to adjust to differences in column loading capacity and that a loading phase was terminated when its time limit had been reached (see figure 4.5a).
Figure 4.5 – The third experiment. The cycle number (purple) increased every fourth step. (a) UV sample signal (blue) was roughly constant. UV breakthrough signal (green) increased every step in the process. The peak was higher every fourth step, indicating that one column had a lower binding capacity than the others. ΔUV (black) reached 70 % every fourth step in a similar manner as the UV breakthrough signal. The loading phase was stopped after the time limit was reached for the steps where the ΔUV did not reach 70 %. (b) UV elution signal (red) and flow for elution buffer (blue). The flow for elution was stopped after the peak was done.
The functionality over time can also be verified from the UV elution signal. It can be seen that the elution peaks have a repetitive pattern; every fourth peak is lower than the rest, indicating once again that one column had a lower loading capacity. The flow for an elution phase was stopped when the peak was done, this is visible in the log of the run (not shown) and if the figure is zoomed in (not shown). From this, it was concluded that the system could adapt to a different amount of product to elute (see figure 4.5b).

The run over a long time proved that Orbit was able to control the ÄKTA pcc system for a long time. This includes both that Orbit maintained the connection to the Unicorn software and that Orbit was able to keep the process running. The change from a ‘one cycle run’ to a ‘six cycles run’ was done in the script file by a single line of code, meaning that it was very easy to change the process design. The simulation of a run over a long time (see section 4.4) had similar behavior as the experimental run over a long time, implying that the simulation can be used to get an idea of how a newly designed process will behave.

4.6 Buffer machine

100 ml of a sodium acetate buffer solution was made in the experiments (50 mM, pH = 3.5). The stock solution was a sodium acetate solution (500 ml). Between the experiments, improvements were done to the process, which are described in the following paragraphs. The final design of the buffer machine is described in section 3.3.

Pump A1, which pumped stock solution, was a bit slow to reach the target flow rate. If the pump was pumping at another flow rate than the flow rate used for calculating the time, the wrong amount of stock solution was added. The fix to this problem was to set the sample valve to waste and switch it to add the stock solution only when the pump had reached the set flow rate. Another solution could have been to control the addition of stock solution by calculating the added volume from the current flow, but this was not done since the first solution worked. Also, too little stock solution was added as a result of the real flow of pump A1 differing from the set flow. To correct this mistake, the time to add stock solution was calculated
from the real flow rate, instead of the set flow rate. The real flow rate was measured once; it was calculated from the volume that was pumped into a graduated cylinder during a fixed time (see equation 3.5).

Adding a different amount of acid and base depending on the "distance" to the target pH turned out to make the pH control remarkably faster than when only droplets were added for all "distances". In the first design of the buffer machine, the pH was measured at intervals long enough for the pH to stabilize. The stability check, implemented in later designs, made the control faster since the pH was measured more frequently, and as soon as it was stable, either acid or base was added (see figure 4.6). There was a problem with the pH electrode giving a different pH value than what an external verified pH electrode gave, meaning that the produced buffer solution had an incorrect pH value. Despite that, the concept of a buffer machine that can control pH has been proved with this project. The accuracy of the pH is only a question of functioning equipment.

Figure 4.6 – The first and last experiments with the buffer machine. Flow rate A1 (orange), flow rate B1 (green) and pH (blue). pH control started after about two minutes. Before that, stock solution and water was added. The remaining water was added after the pH control was done. (a) First experiment (pH = 3.5, pH margin = 0.2). The pH control was slow. The pH dropped below the target pH before reaching the target pH. (b) Last experiment (pH = 3.5, pH margin = 0.05). The pH control was quite fast. The volume of added acid and base was depending on the "distance" to the target pH. Pump B was washed after the remaining water was added.

If strong acids or bases are to be used as stock solutions, one might have to consider adding some of the water to the buffer container before adding the stock solutions, to prevent accidents.
It was possible to use pump B1 to pump the finished buffer solution to another system, though it was not used in this study. An external pH sensor that could communicate with the computer could have been used instead of the pH sensor in the ÄKTA explorer system. In that case, pump B1 would only be used to pump the solution from the storage container to the next system. That would probably result in a faster pH control loop since the pH would be measured directly in the buffer container and not after having been pumped to the pH sensor. An external pH sensor was not used since there was none available in the lab.

The advantage of the batch mode design used in this project compared to the in-line designed mentioned in section 2.3, is that the buffer machine can produce a buffer solution faster than the receiving system consumes it. This allows the buffer machine to deliver buffer solutions to multiple systems, which cannot be done with an in-line system that only delivers the buffer that is currently used.
5 Future Work

5.1 PCC system

The ÄKTA pcc system was, as discussed before, successfully implemented. Future work will be to use the system for purification in continuous processes, connected in a purification chain with more downstream processing steps. If the system is going to be used in big scale production, one would have to look into scale-up of the process. Antibodies were used in this study, but other substances could also be purified. The PCC process has the same principle regardless of substance.

Future work with Orbit could be to make it more user-friendly. One idea could be to make templates for script and process files. Another idea could be to make it easier for the user to find the correct class in the unit library to use. This could either be done by combining all similar units in all systems to one class, to have separate unit libraries for all systems, or to write detailed documentation for the unit library.

5.2 Buffer machine

The buffer machine was only used as an off-line buffer machine, but it would be possible to connect it to other systems and have it deliver buffer solutions, for example, to a PCC system. One possible structure of communication between the systems that has been discussed during the project is to have a master system that gets buffer requests from the receiving systems. The master then organizes the requests in a priority queue and forwards the request with the highest priority to the buffer machine which makes the buffer and delivers it to the correct system (see figure 5.1). Things to consider with this approach are:

- When should a system require new buffer: when it is out of buffer or when it is about to run out of buffer?
- How do the systems know how much buffer they have left? Should they have
an integrator that keeps track of the used amount of buffer solutions?

- How should the master prioritize the requests? What if many separate systems request buffers, which should have the highest priority?
- How should the buffers be transported from the buffer machine to the systems? Should there be long tubes all over the lab? Should there be separate tubes for each type of buffer or should there be a main tube which then needs to be washed in between transfers?

![Diagram](image)

**Figure 5.1** – One discussed structure for communication between systems for buffer delivery.

It could also be studied whether the buffer machine should produce a lot of buffer solution in a batch mode and deliver it to storage where the receiving system takes from, or if the buffer solution should be produced continuously at the same rate as the receiving system consumes the buffer solution, as discussed in section 2.3.

If the buffer machine is going to be used as an off-line system, it would be of interest to develop a user interface for the system. In the user interface, the user should be able to start the production and to set stock solution concentrations, final volume, final concentrations, and pH.
6 Conclusion

This project has contributed to the work of making the downstream process of bioproduction fully automatic, by implementing the continuous chromatography system ÄKTA pcc in the Orbit software and by developing a concept of an automatic buffer machine.

6.1 PCC system

It is concluded that the Orbit software can control the ÄKTA pcc system. The process can be designed from the needs of the experiment by changing the process file and script file. Thus the design is independent of the system implementation in the unit library. The models for the UV signals can be used for simulation of the PCC process before an experiment is run.

6.2 Buffer machine

It was possible to construct a buffer machine that automatically mixes buffer solutions, and it was possible to control the pH of the solution. It should be possible to connect the buffer machine to other systems as to make the overall process fully automatic.
Frida Heskebeck

References


# A Unit library

<table>
<thead>
<tr>
<th>Class</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>InletValve</td>
<td>set Sets valve position.</td>
</tr>
<tr>
<td></td>
<td>get Returns current valve position.</td>
</tr>
<tr>
<td></td>
<td>getAir Returns 1 if there is air in valve, 0 else.</td>
</tr>
<tr>
<td>Pump</td>
<td>setFlowrate Sets the flow rate to the given value.</td>
</tr>
<tr>
<td></td>
<td>getFlowrate Returns the current flow rate.</td>
</tr>
<tr>
<td>UVsensorSingle</td>
<td>getUv Returns the current UV value.</td>
</tr>
<tr>
<td></td>
<td>autoZero Sets current UV value as zero.</td>
</tr>
<tr>
<td></td>
<td>getBaseline Returns value of baseline.</td>
</tr>
<tr>
<td></td>
<td>setBaseline Sets value of baseline or sets parameters for baseline search.</td>
</tr>
<tr>
<td></td>
<td>getDeltaUV Returns deltaUV value.</td>
</tr>
<tr>
<td>UVsensorMulti</td>
<td>getUV1 Returns UV value for the first wavelength.</td>
</tr>
<tr>
<td></td>
<td>getUV2 Returns UV value for the second wavelength.</td>
</tr>
<tr>
<td></td>
<td>getUV3 Returns UV value for the third wavelength.</td>
</tr>
<tr>
<td></td>
<td>setWavelength1 Sets the first wavelength.</td>
</tr>
<tr>
<td></td>
<td>setWavelength2 Sets the second wavelength.</td>
</tr>
<tr>
<td></td>
<td>setWavelength3 Sets the third wavelength.</td>
</tr>
</tbody>
</table>

Table A.1 – List of implemented classes and methods in Unit Library
<table>
<thead>
<tr>
<th>Module</th>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>autoZero</td>
<td></td>
<td>Sets current UV value as zero for all wavelengths.</td>
</tr>
<tr>
<td>ColumnValve</td>
<td>set</td>
<td>Sets both inlet and outlet column valve to given position.</td>
</tr>
<tr>
<td></td>
<td>get</td>
<td>Returns position of current column valve.</td>
</tr>
<tr>
<td></td>
<td>setAsymetric</td>
<td>Set current column valve to given position, no change in other column valve.</td>
</tr>
<tr>
<td>PumpWashValve</td>
<td>set</td>
<td>Sets both flow path A and B to given position.</td>
</tr>
<tr>
<td></td>
<td>setA</td>
<td>Sets both flow path A to given position.</td>
</tr>
<tr>
<td></td>
<td>setB</td>
<td>Sets both flow path B to given position.</td>
</tr>
<tr>
<td></td>
<td>getA</td>
<td>Returns position for flow path A.</td>
</tr>
<tr>
<td></td>
<td>getB</td>
<td>Returns position for flow path B.</td>
</tr>
<tr>
<td>OutletValve</td>
<td>set</td>
<td>Sets outlet valve to given position.</td>
</tr>
<tr>
<td></td>
<td>get</td>
<td>Returns position of outlet valve.</td>
</tr>
<tr>
<td>pHsensor</td>
<td>pH_ON</td>
<td>Changes flow path for pH meter and restrictor, includes one, both or none.</td>
</tr>
<tr>
<td></td>
<td>get_pH</td>
<td>Returns pH value. Note, only functioning with opc connection.</td>
</tr>
<tr>
<td>ConductivitySensor</td>
<td>getCond</td>
<td>Returns conductivity value.</td>
</tr>
<tr>
<td></td>
<td>getTemp</td>
<td>Returns temperature.</td>
</tr>
<tr>
<td>Alarm</td>
<td>setPrePresAlarm</td>
<td>Activates alarm for pre-column pressure with given value.</td>
</tr>
<tr>
<td>Module</td>
<td>Function</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>setDeltaPres Alarm</td>
<td></td>
<td>Activates alarm for pressure over column with given value.</td>
</tr>
<tr>
<td>setAirAlarm</td>
<td></td>
<td>Activates alarm for air in inlet valves.</td>
</tr>
<tr>
<td>CycleCounter</td>
<td>activateCounter</td>
<td>Activates cycle counter.</td>
</tr>
<tr>
<td></td>
<td>resetCount</td>
<td>Resets cycle counter to zero.</td>
</tr>
<tr>
<td></td>
<td>get</td>
<td>Returns cycle number.</td>
</tr>
<tr>
<td>FractionCollector</td>
<td>setTemp</td>
<td>Set temperature in fraction collector.</td>
</tr>
<tr>
<td></td>
<td>getTemp</td>
<td>Returns temperature in fraction collector.</td>
</tr>
<tr>
<td></td>
<td>getPosition</td>
<td>Returns position of arm in fraction collector.</td>
</tr>
<tr>
<td></td>
<td>fraction</td>
<td>Activates fraction in fraction collector. Fractions based on volume or time.</td>
</tr>
<tr>
<td></td>
<td>stopFraction</td>
<td>Stops fraction in fraction collector.</td>
</tr>
<tr>
<td></td>
<td>fractionOutlet</td>
<td>Activates fraction in outlet valve. Fractions based on volume or time.</td>
</tr>
<tr>
<td></td>
<td>stopFraction Outlet</td>
<td>Stops fraction in outlet valve.</td>
</tr>
<tr>
<td></td>
<td>peakFraction Parameters</td>
<td>Sets signal and parameter for peak search during peak fraction.</td>
</tr>
<tr>
<td></td>
<td>peakFraction</td>
<td>Activates peak fraction in fraction collector. Used kind of tubes must be specified.</td>
</tr>
<tr>
<td></td>
<td>stopPeakFraction</td>
<td>Stops peak fraction in fraction collector.</td>
</tr>
<tr>
<td></td>
<td>peakFraction Outlet</td>
<td>Activates peak fraction in outlet valve.</td>
</tr>
<tr>
<td>stopPeakFraction</td>
<td>Stops peak fraction in outlet valve.</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Outlet</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
B  PCC script file

```python
from system_PCC import *
import numpy as np
from process_PCC import Phase, Process

S = SystemPCC(mode = 'test', connection = 'api')

# Flows and volumes and positions

# Volume columns (ml)
V = 1.

# Colon volumes to load (CV)
CVcip = 5.
CVreg = 5.
CVwash = 5.
CVelu = 5.
CVload = 90.

# Flow rates (ml/min)
Fcip = 0.25
Freg = 1.
Fwash = 1.
Felu = 1.
Fload = 0.75

pos = {'CIP': {'B':1},
       'Reg': {'B':2},
       'Wash': {'A':1},
       'Elu': {'A':2},
       'Load': {'S':1},
       'WashS': {'S':0}}

# S0 – Binding buffer (PBS)
# S1 – Sample
# A1 – Binding buffer (PBS)
# A2 – Elution buffer (50 mM Acetate)
```
# B1 − NaOH
# B2 − Binding buffer (PBS)

baseLine = {'time': 10.,
            'delta': 5.,
            'delay': V * 3 / Flow * 60.,
            'mode': 'Detect once'}

UV = {'startLevel': 50.,
       'endLevel': 50.,
       'targetDelta': 70.}

# Phases

Settings = Phase(name='Settings', Settings=True,
                  pH={'elu': {'pH_pos': 1, 'restrictor_pos': 1},
                      'reg': {'pH_pos': 1, 'restrictor_pos': 1}})

Step1 = Phase(name='Step 1',
               Time=5.,
               SetColValve={'col': 1},
               SetOutValve='wash',
               pH={'elu': {'pH_pos': 1, 'restrictor_pos': 0},
                   'mode': 'Off'})

Step2 = Phase(name='Step 2',
               Time=5.,
               SetColValve={'col': 2},
               SetOutValve='wash',
               pH={'elu': {'pH_pos': 1, 'restrictor_pos': 0},
                   'mode': 'Off'})

Step3 = Phase(name='Step 3',
               Time=5.,
               SetColValve={'col': 3},
               SetOutValve='wash',
               pH={'elu': {'pH_pos': 1, 'restrictor_pos': 0},
                   'mode': 'Off'})
Step4 = Phase(name = 'Step 4',
    Time = 5.,
    SetColValve = {'col': 4},
    SetOutValve = 'wash',
    pH = {'elu': {'pH_pos': 1, 'restrictor_pos': 0}},
    Baseline = {'mode': 'Off'})

CIP = Phase(name = 'CIP',
    Time = V * CVcip / Fcip * 60.,
    SetFlow = {'B': Fcip},
    SetInlet = pos['CIP'],
    PumpWash = 'B',
    StopFlowAfterPhase = 'B')

Regen = Phase(name = 'Regen',
    Time = V * CVreg / Freg * 60.,
    SetFlow = {'B': Freg},
    SetInlet = pos['Reg'],
    PumpWash = 'B',
    StopFlowAfterPhase = 'B')

Wash = Phase (name = 'Wash',
    Time = V * CVwash / Fwash * 60.,
    SetFlow = {'A': Fwash},
    SetInlet = pos['Wash'],
    PumpWash = 'A',
    StopFlowAfterPhase = 'A')

Elu = Phase(name = 'Elution',
    SetOutValve = 'waste',
    Time = V * CVelu / Felu * 60.,
    SetFlow = {'A': Felu},
    SetInlet = pos['Elu'],
    PumpWash = 'A',
    pH = {'elu': {'pH_pos': 1, 'restrictor_pos': 1}},
    PeakFraction = {'startLevel': UV['startLevel'],
                    'endLevel': UV['endLevel'],
                    'signal': 'UV Elution wl 1',
                    'mode': 'Level',
                    'out': 1,
DynamicStopPeakFraction = { 'out' : 1, 'uvUp' : UV['startLevel'], 'uvDown' : UV['endLevel'], 'timeLimit' : V * CVelu / Felu * 60. })

StopPeakFraction = { 'out':1 }

EluNoPW = Phase(name = 'Elution No PumpWash', SetOutValve = 'waste', # Time = V * CVelu / Felu * 60., SetFlow = { 'A' : Felu }, SetInlet = pos['Elu'], pH = { 'elu':{ 'pH_pos':1, 'restricotor_pos':1 }}, PeakFraction = { 'startLevel' : UV['startLevel'], 'endLevel' : UV['endLevel'], 'signal' : 'UV Elution wl 1', 'mode' : 'Level', 'out' : 1, 'col1' : 1, 'col2' : 1, 'col3' : 1, 'col4' : 1}, DynamicStopPeakFraction = { 'out' : 1, 'uvUp' : UV['startLevel'], 'uvDown' : UV['endLevel'], 'timeLimit' : V * CVelu / Felu * 60. })

StopPeakFraction = { 'out':1 }

Load = Phase(name = 'Loading', # Time = V * CVload / Fload * 60., SetFlow = { 'S' : Fload }, SetInlet = pos['Load'], BaselineFailsafe = { 'mode' : baseline['mode'], 'time' : baseline['time'], 'delta' : baseline['delta'], 'delay' : baseline['delay'], 'targetDelta' : UV['targetDelta']},}
DynamicStopLoading = {'targetUV': UV['targetDelta'],
                     'time': V * CVload / Fload * 60. -
                     baseLine['delay']}) # we start loading and then wait delay time
before starting dynamic stop. This time here is timeout for loading
in total.

StopFlowAfterPhase = 'S')

Start1 = Phase(name = 'Start 1',
                #
                Time = V * CVload / Fload * 60.,
                SetFlow = {'S': Fload},
                SetInlet = pos['Load'],
                SetColValve = {'col': 4},
                SetOutValve = 'waste',
                Baseline = {'mode': baseLine['mode'],
                           'time': baseLine['time'],
                           'delta': baseLine['delta'],
                           'delay': baseLine['delay']},
                DynamicStopLoading = {'targetUV': UV['targetDelta'],
                                      'time': V * CVload / Fload * 60.
                                      - baseLine['delay']})

StopFlowAfterPhase = 'S')

End1 = Phase(name = 'End 1',
              Time = V * CVwash * 2.5 / Fwash * 60.,
              SetFlow = {'S': Fwash},
              SetInlet = pos['WashS'],
              SetColValve = {'col': 4},
              SetOutValve = 'waste',
              StopFlowAfterPhase = 'S',
              Baseline = {'mode': 'Off'})

FlowOff = Phase(name = 'Flow off',
                SetFlow = {'S': 0., 'A': 0., 'B': 0.})

# % % Process

startup = [Start1, Step1, [[Wash, Elu], [Load]],
           Step2, [[CIP, Regen], [Wash, Elu], [Load]],
           Step3, [[CIP, Regen], [Wash, Elu], [Load]],
           Step4, [[CIP, Regen], [Wash, Elu], [Load]]]
cycle = [Step1, [[CIP, Regen], [Wash, Elu], [Load]]],
        Step2, [[CIP, Regen], [Wash, Elu], [Load]],
        Step3, [[CIP, Regen], [Wash, Elu], [Load]],
        Step4, [[CIP, Regen], [Wash, Elu], [Load]]]

closeDown = [End1, Step1, [[CIP, Regen], [EluNoPW]],
            Step2, [[CIP, Regen], [EluNoPW]],
            Step3, CIP, Regen, FlowOff]

cycles = []
nbrCycles = 1
for x in range(nbrCycles):
    cycles += cycle

phases = [Settings] + startup + cycles + closeDown

#
P = Process('PCC_Simulated', phases, S)

options = {'mode': 'test',
            'timeFactor': 100.0,
            'sampleSignals': ['uvSamp', 'uvFlow', 'uvBreak', 'uv1', '
baseLine', 'condReg', 'condElu', 'deltaUV'],
            'sampleSignals': [P.uvSamp, P.uvBreak, P.baseLine, P.
deltaUV', P.uvElu'],
            'sampleTime': 1.0,
            'loops': [(None, P.e.getTime, {})],
            'logData': True,
            'logRun': True}

P.run(options)
C Buffer script file

```python
# coding: utf-8

Created on Mon Mar 18 11:20:21 2019

@author: fridah

from system_buffer import *
from process_buffer import Phase, Process

strMode = 'real'
S = SystemExplorer1(mode=strMode)
S2 = SystemExplorer2(mode=strMode, connection='remote')

# Simulated pH

def pHsim(P, phaseName = 'Adjust pH and add Water'):
    phase = [phase for phase in P.activePhases if phase.name==phaseName]
    time = phase[0].time
    pH = 8 - time/60*0.5
    return pH

S.pHsensor.pHsim = pHsim

# Things to set

finalVolume = 100. # ml
pH = 3.5 # 7.4 # A float value or None
pHMargin = .05 # A float value or None
emptyFlask = False # True or False. If True, the buffer is pumped to outlet 5

stockConc = { 'A1': 500.,
              'A2': 0,
```
targetConc = {'A1': 50., # NaAcetate
              'A2': 0,
              'A3': 0,
              'A4': 0,
              'A5': 0.,
              'A6': 0.,
              'A7': 0.,
              'Water': 0.}

# automatic things are nice
flow = 100. # ml/min
realFlow = 101. # ml/min
washVol = 10. # ml
bufferVol = 0.
timeTotal = 0.
washTime = washVol / flow * 60 + 20. # + 20 s since this is the extra
time in phaseOption

phases = []

# Stock solutions A1-A7
for stock, conc in sorted(stockConc.items()):
    if conc > 0:
        vol = targetConc[stock] * finalVolume / stockConc[stock] # C1 * V1 = C2 * V2
        printFun(\x1b[36m% stock - %.1f ml \x1b[0m%(stock, vol))
        bufferVol += vol
        timePhase = vol / realFlow * 60.
        timeTotal += timePhase + washVol / realFlow * 60 + 16.
        phaseThis = Phase(name = stock,
                           Time = timePhase, # s
                           SetInletValve = stock,
                           LoadStockAndPumpWash = {'washVol': washVol,
```
phases += [phaseThis]

if pH == None:
    timePhase = (finalVolume - bufferVol) / realFlow * 60.
    timeTotal += timePhase + washVol / realFlow * 60 + 16.
    phaseThis = Phase(name = 'Water',
                       Time = timePhase, # s
                       SetInletValve = 'Water',
                       LoadStockAndPumpWash = {'washVol': washVol,
                                               'flow' : flow})

    phases += [phaseThis]
else: # pH should be adjusted
    timePhase = (finalVolume - bufferVol) / realFlow * 60
    timeTotal += timePhase
    timeTotal += washTime
    phaseThis = Phase(name = 'Adjust pH and add Water',
                       # Time = timePhase, # s
                       SetInletValve = 'Water',
                       pH = {'pH': pH,
                              'phMargin': pHMargin,
                              'finalVolume' : finalVolume,
                              'bufferVol' : bufferVol,
                              'flow' : flow,
                              'realFlow' : realFlow,
                              'washVol' : washVol})

    phases += [phaseThis]

# Empty flask
if emptyFlask: # True
    timePhase = finalVolume / realFlow * 60 + 50/realFlow * 60. # s , 50 ml extra just in case.
    timeTotal += timePhase
    phaseThis = Phase (name = 'EmptyFlask',
                       Time = timePhase,
                       SetOutValve = 'nextSys',
```

---

"Water" = %.1f ml \x1b[0m%(finalVolume−bufferVol)"
SetFlow = \{ 'B': flow \}
phases += [phaseThis]

# Wash pump B
if emptyFlask or pH != None:
timePhase = 5. # s
timeTotal += timePhase + washTime + 15
phaseThis = Phase (name = 'WashPumpB',
                      Time = timePhase,
                      PumpWashB = \{ 'washFlow': 100, 'washVol': 30 \})
phases += [phaseThis]

# Stop flow
timePhase = 5. # s
timeTotal += timePhase
phaseThis = Phase (name = 'StopFlow',
                      Time = timePhase,
                      SetFlow = \{ 'A': 0. \})
phases += [phaseThis]

# Information to user
if pH != None and pHMargin == None:
    raise Exception ('A pH margin must be set. ')
if bufferVol > finalVolume:
    raise Exception ('Volume of added buffers (%.2f) is greater than final volume (%.2f). ' % (bufferVol, finalVolume))
elif bufferVol > finalVolume * 0.99 and pH != None:
    raise Exception ('Buffer volume > 0.99*finalVolume. There is a risk that pH can\’t be adjusted with remaining volume. ' % (bufferVol, finalVolume))
elif pH != None:
    S .printFun ('\x1b[36mBuffer will be done in approx: %.0f min + time for pH adjusting.\x1b[0m' % (timeTotal/60))
else:
    S .printFun ('\x1b[36mBuffer will be done in approx: %.0f min. \x1b[0m' % (timeTotal/60))

P = Process ('Buffer_real', phases, S, S2=S2)
options = {'mode':strMode,
            'timeFactor':1.,
            'sampleSignals':['concb', 'flow', 'pH'],
            'sampleTime':1.0,
            '# loops':[(None, P.e.getTime, {})],
            'logData': True,
            'logRun':True}

P.run(options)