Development of a Bioreactor Simulator for supporting automation software test and verification

Viktor Liljequist
Abstract

Development of a Bioreactor Simulator for supporting automation software test and verification

Viktor Liljequist

The GE Healthcare Life sciences organization develop and manufacture bioreactors, mixers, filtration skids and chromatography systems used together in a biomanufacturing platform. The platform is monitored and controlled by a distributed control system through a Programmable Logic Controller (PLC). The automation software controlling the platform is today tested and verified together with the physical units. The software use PROFIBUS, an industry standard for industrial automation, for communication and control of the units. Limited access to the physical units is usually a bottleneck and it’s difficult to test abnormal situations to make sure the correct alarms are triggered. To reduce the hardware dependency and to provide support during test and verification, a virtual environment is developed to simulate the behavior of a bioreactor during execution. A .NET application has been developed together with a mathematical framework to simulate a cell culture and to return relevant process parameters such as pH, dissolved oxygen, temperature and weight. The results show that it’s possible to simulate a bioreactor and to communicate with the control system. The software can be a valuable tool when developing and testing automation software but should not be used for process optimization or tuning of control parameters.
Sammanfattning

GE Healthcare Bio-Sciences AB utvecklar och tillverkar bioreaktorer, mixrar, filtreringsutrustning och chromatografisystem som levereras tillsammans i en komplett plattform för tillverkning av proteinläkemedel. Plattformen styrs idag av ett automationssystem med ett distribuerad styrsystem på en redundant PLC som kan kontrolleras av en operatör genom ett grafiskt användargränssnitt. Styrsystemet använder PROFIBUS, en industristandard för industriell automation, för kommunikation och styrning av plattformens enheter. All utveckling av styrsystemets mjukvara måste idag ske tillsammans med de fysiska enheterna vilket är en begränsande faktor under test och verifieringsfaserna. Det är dessutom svårt att testa avvikande situationer och gränsvärden som kan vara svåra, eller direkt farliga, att uppnå på de fysiska enheterna. För att minska hårdvaruberoendet och underlätta testning samt verifiering av plattformens bioreaktorer ska en virtuell miljö utvecklas som kan simulera beteendet hos en bioreaktor under exekvering.

Genom ett system av matematiska modeller har en Windowsapplikation utvecklats i syfte att simulerar en cellodling samt returnera relevanta miljöparametrar som pH, grad av syresättning och temperatur till det befintliga styrsystemet. Applikationen exekverar tillsammans med ett program som kan ta emot och returnera PROFIBUS och beräknar processvärden med avseende på värden från syra/baspumpar, massflödesregulatorer, agitator och temperaturregulator.

Denna rapport har redogjort för den matematiska modellering som används för att simulera cellodlingen och miljöparametrarna. Rapporten har även förklarat grundläggande begrepp inom automation, reglerteknik och mjukvaruutveckling för att sedan analyseras i det slutgiltiga resultatet.

Resultatet har visat att det är möjligt att med en virtuell miljö simulera beteendet hos en bioreaktor utan att det befintliga styrsystemet märker någon större skillnad. Däremot är de matematiska modellerna mycket begränsade och bör inte användas i syfte att optimera PID-parametrar eller som verktyg vid processoptimering. Simulatorn kan däremot vara mycket värdefull som verktyg vid utveckling av automationsmjukvara och bör även kunna användas i de tidiga faserna av test och verifiering för att minska beroendet av hårdvara.
Preface

This report is the result of my Master’s thesis during spring of 2017 on the Master’s program in Sociotechnical Systems Engineering. The work has been conducted at GE Healthcare Bio-Sciences AB at the department of Research and Development together with the Division of Systems and Control at Uppsala University.

First of all, I would like to thank my mentors Anders Nygård and Peter Toreheim from GE Healthcare and my supervisor Prof. Bengt Carlsson from the Division of Systems and Control. I would also like to thank Simon Åhr for providing the control system interface. A big thank you to Håkan Klang, Andreas Castan, Martin Karlsson, Christian Nes Sjögren, Johan Bäckman, Alesander Gil Almamdoz and Björn Stenquist for great support throughout the whole project.

Viktor Liljequist, 2017-05-31, Uppsala
Table of contents

Glossary and definitions ................................................................. 4
1. Introduction .................................................................................. 5
  1.1 Purpose ................................................................................... 5
  1.1.1 Limitations .......................................................................... 5
  1.1.2 Delimitations ........................................................................ 6
  1.2 Disposition ................................................................................ 6
  1.3 Technical resources ................................................................. 6
2. Background .................................................................................. 8
  2.1 Bioprocess technology .............................................................. 8
    2.1.1 Reactor Design .................................................................... 8
    2.1.2 XDR cell culture bioreactor systems ...................................... 9
    2.1.3 Online Measurement .......................................................... 9
    2.1.4 Modes of Operation ........................................................... 10
    2.1.5 Applications ....................................................................... 10
  2.2 Process Parameters .................................................................. 11
    2.2.1 pH ...................................................................................... 11
    2.2.2 Dissolved Oxygen ............................................................. 12
    2.2.3 Agitation ............................................................................ 14
    2.2.4 Temperature ....................................................................... 14
  2.3 General Automation ............................................................... 15
    2.3.1 Process Signals .................................................................. 16
    2.3.2 PID Control ....................................................................... 17
    2.3.3 Data types .......................................................................... 18
    2.3.4 PLC ................................................................................... 20
    2.3.5 PROFIBUS ........................................................................ 21
  2.4 .NET Framework .................................................................... 21
3. Theory and model framework ...................................................... 22
  3.1 Cell Growth ............................................................................ 23
  3.2 Dissolved Oxygen Tension ...................................................... 25
  3.3 pH Function ............................................................................ 26
    3.3.1 Buffer Systems .................................................................. 27
  3.4 Temperature Model ................................................................. 29
    3.4.1 Heat Transfer Equations .................................................... 30
  3.5 Model summary ....................................................................... 31
4. Result .......................................................................................... 32
  4.1 Software .................................................................................. 34
## Glossary and definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protein</td>
<td>Long chains of amino acids with unique structures and functions</td>
</tr>
<tr>
<td>Recombinant proteins</td>
<td>Proteins produced by a host cell defined by the recombinant DNA (Protein blueprint)</td>
</tr>
<tr>
<td>Recombinant DNA</td>
<td>A combination of DNA sequences that is created with the help of enzymes (DNA scissors) and does not exist naturally.</td>
</tr>
<tr>
<td>PLC</td>
<td>Programmable Logic Controller</td>
</tr>
<tr>
<td>MFC</td>
<td>Mass Flow Controller mainly used for gas control</td>
</tr>
<tr>
<td>DOT</td>
<td>Dissolved Oxygen Tension</td>
</tr>
<tr>
<td>Cell Culture Medium</td>
<td>Composition of energy sources for supporting cell growth and production of product.</td>
</tr>
<tr>
<td>DCS</td>
<td>Distributed Control System</td>
</tr>
</tbody>
</table>
1. Introduction

The GE Healthcare Life sciences organization develop and manufacture bioreactors, mixers, filtration skids and chromatography systems used together in industrial production of biopharmaceuticals. The process can be divided into an upstream and a downstream phase. The upstream phase contains steps for expressing the desired gene that produces the protein of interest, creating the recombinant DNA, inserting it into a host cell DNA and creating a cell culture with optimal environmental conditions. The downstream phase consists of protein purification through chromatography. The bioreactor is a stainless-steel vessel ranging from a volume of only 10 liters up to 2000 liters. The vessel also consists of an agitator, Mass Fluid Controllers (MFCs) and pumps for controlling necessary parameters.

When creating the cell culture for expanding the cells environmental parameters like pH, dissolved oxygen and temperature need to be controlled and kept on certain levels to achieve high cell density and viability without decline in production of the desired product. With the help of PLC hardware and automation software, a PID-controller shall be able to adjust the parameters to a desired level without harming the cell culture. Beside online measurement and control of pH, O₂, temperature and CO₂ the automation software shall be able to trigger alarms and perform correct process, set point and control values. During the software development, the testing and verification is often hardware dependent and this is usually a bottleneck when providing software updates and patches. By introducing software solutions that connects to the PLC control unit and simulates the behavior of a certain hardware unit the hardware dependency can be postponed. This will also support the automation software developers when testing new software and abnormal situations.

1.1 Purpose

To reduce the hardware dependency when testing and verifying automation software a software simulator shall be developed as a .NET application. The software shall simulate the behavior of a cell culture and shall be able to send process values and receive set point and control values from a PLC. The cell culture model shall be composed of a system of ordinary differential equations describing the cell density, dissolved oxygen tension, volume and substrate concentration. To be able to provide a step response for all parameters controlled by the PLC, the simulator will also return approximate process values of pH, pressure and agitation. The simulator shall also provide a graphical user interface (GUI) where process parameters can be specified and simulation behavior can be modified during execution.

1.1.1 Limitations

Most of the models presented in this report regarding cell culture are equations based on empirical studies of bacterial growth. Although the overall dynamic of a cell culture is
similar to bacterial culture, there are important differences like composition of cell medium and maximum growth rate. The similarities and differences will be further explained in this report.

1.1.2 Delimitations

Due to time limitations and lack of profound biotechnical knowledge the result of this thesis shall not be used for process optimization and tuning of control parameters. The software is strictly a tool for automation engineers to reduce the hardware dependency and the simulator will not be useful without a connection to the specific control system developed by GE Healthcare. The development of strategies necessary for the control system connection is not a part of this thesis but is of most importance to fulfill the purpose.

1.2 Disposition

This thesis will begin with an introduction to bioprocess technology in Section 2.1 and the different types of systems that exists for growing cells in an industrial scale. The Section will briefly describe different modes of executions in bioreactors and some of the most common applications within the biopharmaceutical industry. Section 2.2 will describe the most fundamental environmental control parameters for optimizing the product yield from a cell culture. To understand how this control is executed Section 2.3 will explain the main fundamentals behind industrial automation. Section 3 will present the mathematical framework used to describe the dynamic behavior of cell growth and important process parameters. Section 4 will present the architecture and implementation of the simulator software together with a presentation of the Graphical User Interface (GUI). An analysis and reflections regarding further development of the result will be presented in Section 5 together with some conclusions.

1.3 Technical resources

In order to implement and simulate the behavior of a bioreactor, and to fulfill the purpose of this thesis, all models presented in Section 3 must be implemented in a software environment. This software shall also be connected to the same distributed control system (DCS) where the real bioreactors are controlled. To achieve this a bioreactor simulator shall be developed in a .NET environment with Visual Studio 2015. This simulator must run together with a PROFIBUS simulator that can receive PROFIBUS signals from the PLC, repackage into a .NET compatible data structure and send to the bioreactor simulator. The bioreactor shall generate process values and send the values back to the PROFIBUS simulator and to the PLC.

All software development will be performed in Visual Studio 2015 with the .NET framework and C#. A more detailed description of the .NET framework can be found under Section 2.4. During development, an Allen Bradley PLC will execute the control software and generate input signals and receive process values. To interpret PROFIBUS
signals, the PC used for development must be equipped with a PROFIBUS Multi-Slave PCI Card and a PROFIBUS simulator software for signal handling between the PLC and the Bioreactor Simulator. In the test and verification part, the Bioreactor Simulator will be tested in a real industrial manufacturing environment with connection to the DCS through a network of PLCs. For more information regarding PROFIBUS, PLC and DCS see Section 2.3.


2. Background

The following section will provide a thorough explanation of the three most important areas of this project. Bioprocess and bioreactor technology, the most important process parameters and an introduction to industrial automation. The purpose is to enough knowledge about these areas to understand the theoretical framework and results in Section 3 and 4.

2.1 Bioprocess technology

In the process of producing biopharmaceuticals, the upstream process consists of combining raw materials with biocatalysts to create a fermentation process. The raw materials, also called medium, is the composition of components known as substrate, e.g. sugar, phosphate. The main energy source in most application is carbon, mostly supplied as glucose or some other carbohydrate. In aerobic processes the oxygen demand must be satisfied through continuous aeration and a complementary oxygen supply. Other components such as nitrogen and trace elements are supplied through ammonia and salt. The components and the amount of each component differs between the applications and should be carefully considered and optimized. The main purpose of the medium is to obtain a high growth rate and to maximize the yield since growth medium can be expensive. The biocatalyst is the microorganism, e.g. mammalian host cells from a Chinese hamster ovary or E. coli bacteria, that contains the recombinant DNA. Together with the growth medium the microorganisms will create a reaction called Fermentation. The definition of fermentation in industrial processes is “production with microorganisms or cultured animal cells” [1, p. 9]. [1, pp. 11-14]

2.1.1 Reactor Design

The most conventional bioreactor is the Stirred Tank Reactor (STR). The vessels used in the bioprocess systems developed by GE Healthcare contains between 10-2000 liters of cell medium and biomass and the main applications is production of recombinant proteins and animal cell products. Some of the most common features of the STR is an agitator/stirrer, MFCs, pumps and a surrounding water container used for temperature control. The agitator provides mixing and turbulence for improvement of the gas-liquid transfer and to achieve a uniform pH distribution in the tank. The agitator can operate with a speed up to 360 RPM depending on the vessel volume where a smaller vessel can tolerate higher speeds. High agitator speed might result in faster mixing with better temperature and acid/base distribution and higher solubility of oxygen. But high speeds will also expose the microorganisms to higher levels of stress which will harm the cell culture. The MFCs are used for control of dissolved oxygen and pH. Usually MFCs are often placed in the bottom of the reactor to achieve long exposure time for the gas bubbles towards the fluid. The MFCs are mostly connected to air for providing continuous aeration, pure O2 for supporting control of dissolved oxygen and CO2 for controlling pH. [1, pp. 19-21]
2.1.2 XDR cell culture bioreactor systems

The XDR bioreactor system is a complete bioreactor designed and produced by GE Healthcare. The platform is delivered with a STR vessel, an I/O cabinet and a mobile control console. The system use single-use technology with consumable components to eliminate the risk of cross-contamination between batches. The single-use technology also eliminates time consuming and costly cleaning and sterilization of multiple components between each batch. The system accommodates flexible needs with reactors built for both process development and optimization up to large scale, industrial production of biopharmaceuticals. Embedded scaling methodology for necessary bioprocess parameters supports the transition from small-scale process development to manufacturing and avoids unnecessary and time-consuming process redesign. [2, p. 1]

The single-use technology is mainly delivered with the disposable XDA bioreactor bag assembly containing the agitator, MFCs, pumps and many other components. Everything is disposable and replaceable after every batch. The bag is placed inside the stainless-steel STR vessel equipped with a temperature controlling jacket. The jacket provides an additional container around the STR vessel for circulating a cooling or heating fluid [3]. Together with an external Temperature Control Unit (TCU) the fluid, usually water, can be circulated inside the jacketed vessel and brought back to the TCU for heating or cooling. The vessel can be equipped with sensors for online measurement of pH, dissolved oxygen/carbon dioxide and temperature. The liquid and gas management together with the online measurement of pH, temperature, dissolved oxygen and carbon dioxide can all be monitored and controlled through automation systems like Rockwell, DeltaV, Honeywell, Siemens or Mitsubishi systems. An operator can monitor the process and provide set point values for desired environmental parameters through a Human Machine Interface (HMI). The set point values will be transmitted to the automation system and the PLC will transmit control values to the devices responsible for changing the selected parameter. The XDR bioreactor system is controlled by a Rockwell PLC and the operator interacts with an HMI called Wonderware produced by Schneider Electric. [2, p. 2].

2.1.3 Online Measurement

To obtain an ideal environment for the cell culture online measurement of the most important process parameters is critical. Mainly physical parameters like pressure, liquid weight, temperature and agitator speed are measurable, but also chemical parameters like pH and dissolved oxygen concentration. A major obstacle in bioprocess systems is the strict requirement on aseptic function inside the cell culture which means that all sensors must be sterilizable. This is to avoid cross contamination between batches and is one of the reasons behind the lack of suitable sensors. [1, pp. 28-31]
2.1.4 Modes of Operation

A fermentations process can be executed in three different modes; Batch, fed-batch and continuous batch. Figure 1 illustrates a typical batch culture. It begins with a lag phase where the growth rate is neglectable and is followed up by an exponential phase were the concentration of biomass increase at an exponential rate. This will proceed until all nutrients are consumed and the growth enters the stationary phase. During batch execution, all necessary nutrients are added from start at high enough concentrations to make the growth rate unconstrained with respect to the substrate. The biomass concentration will after an initial dead time increase at an exponential rate until the low substrate concentrations reduces the growth rate. This will slow down the increase in biomass concentration until it enters a stationary phase. If the culture proceeds without additional supplement of nutrients the biomass will enter a death phase and the viable cell concentration will decrease. During a batch culture, only substances with the purpose to control pH, dissolved oxygen and foaming are added. A batch culture process duration is generally between 7-10 days. In a fed-batch culture nutrients are added during the cell growth to keep the nutrition concentration at a relative stable level. The reaction rate can then be controlled and kept on a constant level for a longer time which will result in higher viable cell concentration, increase in volume and higher product yield. This type of process can have a duration for 10 – 20 days. In continuous fermentation, the reactor is fed with medium and at the same time biomass is withdrawn at a constant rate. The volume is therefore kept constant and the process duration can be kept up to several months. [1, p. 105]

![Typical growth curve for cell and bacterial culture.](4)

2.1.5 Applications

Production of therapeutic proteins through biological manufacturing with animal cells is the main technology when producing insulin, monoclonal antibodies and vaccines. Insulin can also be produced using bacteria, e.g. *E. coli*, as a host, but due to the complex construction of monoclonal antibodies and vaccines a more sophisticated host is desired. The insulin protein is made up of about 800 atoms while a monoclonal antibody and a vaccine can be made up of over 10 000 and 100 000 atoms. Monoclonal
antibodies can be used in the treatment of breast cancer as Herceptin that inhibits uncontrolled cell growth. Other applications are treatment of autoimmune disorders such as Rheumatoid arthritis and Crohn’s disease. [5]

2.2 Process Parameters

2.2.1 pH

One of the most important parameters to control for a successful cell culture is pH. A wide range of research studies exists within the area of pH-control in bacterial fermentation and mammalian cell culture. According to studies conducted by Yoon et al. [6] changes in pH have significant impact on cell viability and the specific growth rate. The study shows two experiments with CHO-cells at 32.5 and 37.0 °C where the pH varied between 6.80 and 8.00 in a batch culture. The ideal pH level was at 7.00 with a maximum viable cell density of 3.0 \(10^6\) cells/mL at 32.5 °C and 4.2 \(10^6\) cells/mL at 37.0 °C. The pH level was controlled with base addition through NaOH and CO\(_2\) gas for decreasing pH. By changing the pH to 6.8 and 7.2 an immediate decrease in cell viability could be observed. The results from the study can be seen in Figure 2.

![Figure 2](image)

**Figure 2.** Effect of pH change on specific growth rate and cell viability. The black dots represent 32.5 °C and the white 37.0 °C [6, p. 348].

pH [7] can be formally defined as the logarithm concentration of hydrogen ions according to equation 1 and is a measurement for acidity.

\[
pH = -\log_{10}[H^+]
\]  

(1)

The pH scale is between 0-14 and can be derived from the equilibrium of hydrogen and hydroxyl ions in pure water which is neutral with a pH of 7. At 25 °C the concentrations can be written like

\[
[H^+] \cdot [OH^-] = 10^{-14}
\]  

(2)
where the dissociation of hydrogen and hydroxyl ions must be equal to $10^{-7}$ which will result in pH for neutrality.

$$pH = -\log[10^{-7}] = 7$$

Since pH is a critical control parameter rapid changes in pH must be prevented and changes in desired pH levels should be satisfied by the process without long delays. Due to the non-linear logarithmic behavior, this is very hard to achieve in practice. A suggestive example of this is the titration curve where a strong acid, e.g. HCL, is continuously neutralized by a strong base. As Figure 3 illustrates, the pH has a non-linear relationship to the volume of added base until a rapid change occurs between pH 4 and 10. Since mammalian cells prefer a cell culture at pH 7 with only small deviations, an advanced control strategy must be used [7].

![Figure 3. Titration curve [7]](image)

### 2.2.2 Dissolved Oxygen

Another parameter essential for cell culture is the level of dissolved oxygen (DO) in the cell growth medium. The dissolved oxygen is critical to maintain cell respiration and refers to the level of free oxygen molecules available for the cells to consume. In biotechnical applications, complete aeration is often desired through the inlet of air from the bottom of the reactor. The oxygen will then transfer from the bubble into the medium through diffusion until the medium is in equilibrium with the air. This state of complete equilibrium with air is often expressed as 100 % Dissolved Oxygen Tension (DOT) which is the most common unit for measuring dissolved oxygen in industrial applications. The electrodes that provides the measurement of DOT must be calibrated to 100 % by measuring the partial pressure of oxygen in air. The reading from the electrode during execution will be dependent on the partial pressure of oxygen inside the bioreactor and can be described as
\[ DOT = 100 \frac{P_{O_2}}{P_{O_2,\text{cat}}} \]  

where \( P_{O_2} \) is the partial pressure of oxygen inside the reactor and \( P_{O_2,\text{cat}} \) is the partial pressure of oxygen during calibration of the electrode. The dissolved oxygen concentration can be written according to Henry’s law [1, p. 89] as

\[ C = \frac{P_{O_2}}{k_H} \]  

where \( k_H \) is a Henry constant. By combining these equations, an expression mapping the dissolved oxygen concentration to its corresponding DOT through a Henry’s law derived constant can be written as

\[ DOT = C \frac{100 k_H}{P_{O_2,\text{cat}}} = C \cdot H \]  

where \( H \) is the Henry’s law constant with a common value of 14000 % used during simulations.

If pure oxygen is sparged into the cell medium, levels above 100 % DOT can be reached since pure oxygen have a lot higher oxygen concentration than air. [8]

The oxygen required by the cells depends a lot on the cell species [1, pp. 79-81]. Almost all oxygen is consumed through cell respiration and the total oxygen demand in a cell culture can be described by equation

\[ r_o = X q_o = X \frac{\mu}{Y_{x/o}} \]  

where \( X \) is the total biomass concentration, \( q_o \) is the specific volumetric oxygen consumption rate that can be described as the ratio between the cell growth rate, \( \mu \), and the yield from oxygen to biomass, \( Y_{x/o} \). This yield is often denoted in kg biomass produced per kg oxygen consumed. In bacterial fermentation, this yield is much lower than for cell culture which results in a higher oxygen demand.

![Diagram](image)

*Figure 4. Oxygen pathway from gas phase to cells*
The oxygen transfer from the gas phase to the liquid phase can be illustrated according to Figure 4 where the mass transfer coefficient \( K_L \) determines the rate of transfer from gas to liquid through the bubble layer. Both the gas to liquid transfer and the liquid to cell transfer is mainly done through diffusion. To enhance the oxygen transfer, an agitator is used to remove the resistance of the air bubble layer by breaking the bubbles. The transfer resistance over the bubble film is also affected by the oxygen concentration difference between the bubble and the liquid. The parameter \( K_L \) can be described as

\[
K_L = \frac{D}{\delta}
\]

where \( D \) is the gas diffusivity over the bubble film and \( \delta \) is the film thickness. By expanding this expression, we obtain the parameter \( K_L a \) (h\(^{-1}\)), which is called the volumetric transfer coefficient and can be described as

\[
K_L a = \frac{D}{\delta} \cdot \frac{\sum A}{V}
\]

where \( \sum A \) is the total bubble surface area and \( V \) is the reactor volume. Parameters such as agitation and aeration rate have positive effect on \( K_L a \), while volume have a negative effect [9] [1, pp. 83-86]. To increase gas hold-up in the reactor the vessel should have a cylinder-shaped design with a length to width ratio that supports long exposure for the bubbles towards the cell medium. The viscosity of the medium has great influence where increased viscosity will reduce \( K_L a \).

### 2.2.3 Agitation

In a STR reactor the stirrer provides mixing and turbulence for creating a homogenous environment inside the reactor. It supports gas mixing by breaking gas bubbles, reduce mixing time when adding acid or base for controlling pH and create an even temperature distribution. The stirrer speed is variable and depends on the vessel volume and motor capacity where smaller reactors can have higher speeds. High agitator speed will result in better mixing but can also cause a stressful environment for the cells which will reduce product quality. Often in bioreactors for mammalian cell cultures the stirrer is classified as an impeller to reduce stress but still provide appropriate mixing. [1, p. 20]

### 2.2.4 Temperature

The cell culture temperature has tremendous impact on the growth rate and cell viability, as can be seen in figure 2. Although it is also shown by Furukawa et al. [10] that a low culture temperature may result in maintained high cell viability, improved productivity of the recombinant protein together with less medium consumption. According to this result a slight deviation from the optimal temperature of 37 °C made the cells more efficient during production. This result is very likely to be very
dependent on the cell line and equipment used during the experiment and cannot be
generalized.

2.3 General Automation

According to Simutis and Lübbert [11] bioreactor control is of most importance when
producing high quality products with minimum variability in product properties. With
the help of simple feedback techniques instead of open loop control the performance of
the fermentation process can be drastically improved. Different control strategies can be
used depending on the process variable but often a dynamic model is used to derive an
appropriate control algorithm. The model must be able to sufficiently catch the
necessary dynamics of the system and represent system responses to process
disturbances. Models can be used as transfer functions when deriving traditional PID
controllers or in more advanced controllers like Model Predictive Control (MPC).

Automation of industrial processes has been a revolutionary development for companies
producing high quality and cost-effective products [12]. The different applications are
wide spread and with the development of the networked society and Industry 4.0 even
more opportunities arise. The main purpose behind an automation system is to maintain
consistency in required quality and cost in production without any manual intervention.

The automation process can generally be divided into three steps according to Figure 5.
In a continuous industrial process the automation system will be designed to monitor
and control some physical properties necessary for the process to deliver consistent and
high quality results. The process must be equipped with sensors capable of reading and
transmitting analog values corresponding to the physical properties. The automation
system will then compare this value with the desired value and provide a control value
to eliminate the control errors. [12, p. 11]

---

**Figure 5. Overview of the automation process control system**
2.3.1 Process Signals

The input and output signals in the automation control system correspond to some physical property depending on the application and purpose. Signals can be divided into three main categories: Discrete, continuous and fluctuating. These names can also be translated to digital, analog or pulse. A digital signal is a state-changing signal represented as a discrete value of 1 or 0, e.g. on and off. The analog signal corresponds to an analog value of a physical property like temperature, pH or flow. When using sensor technology analog values will often be represented as a voltage or current. A fluctuating signal is a discrete signal that repeatedly change state during a given period.

A common application of pulse signals is Pulse Width Modulation (PWM) \([13]\). PWM is used to encode an analog signal through fast switching between a binary high and low value. The ratio between the on and off period will result in a mean value corresponding to the desired analog value. When controlling the voltage supplied to a DC motor PWM is used to switch between e.g. 0 and 5 volts resulting in different speeds depending on the pulse width. \([12, pp. 12-14]\)

In an open-loop system no feedback control is used. For example, if a DC-motor is only controlled by the applied voltage the system will have no information regarding the motor position \([14]\). The embedded controller will therefore never receive information about the actual position and will not be able to adjust possible errors. By introducing feedback a closed-loop controller can produce an output signal dependent on the control error calculated from the obtained process value and the desired set point value.

Figure 6 shows an overview of a typical closed loop system with components for manipulation and measurement of a process variable.

![Figure 6. Overview of a general control loop](image)

The manipulating part of the control loop consist of a converter, an actuator and a valve. For bioprocess control the most common type of control is varying flow of either gas or fluid through a valve. The valve receives mechanical movement as an input signal from the actuator and responds by producing a flow of gas or fluid into the process. This
signal starts as a CV from the controller and is often specified from 0 to 100 %. The converter translates the percentage into an electrical signal in milliampere or voltage, or a pneumatic signal. The actuator will then produce a mechanical movement adjusting the valve. The measurement part begins with a sensor in direct contact with the process. A broad range of sensors are available for multiple applications and works as a low power device capable of producing a signal depending on some physical property. This signal must be interpreted, amplified and filtered from noise by the transducer before being transmitted back to the controller. [7, pp. 15-18]

Table 1. Description of signal types in figure 4

<table>
<thead>
<tr>
<th>Signal type</th>
<th>Description</th>
<th>Acronym</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Value</td>
<td>Measured process value</td>
<td>PV</td>
<td>(y(t))</td>
</tr>
<tr>
<td>Set Point Value</td>
<td>Desired process value</td>
<td>SP</td>
<td>(y_{ref}(t))</td>
</tr>
<tr>
<td>Control Value</td>
<td>Value sent from controller to actuator</td>
<td>CV</td>
<td>(u(t))</td>
</tr>
<tr>
<td>Control Error</td>
<td>Error between SP and PV</td>
<td>e</td>
<td>(e(t))</td>
</tr>
</tbody>
</table>

### 2.3.2 PID Control

In figure 4 the controller block receives the difference between the SP and the measured output PV as an input signal. The goal is to eliminate this error and there are many different approaches towards this problem. One of the oldest and most common solutions within industrial process and manufacturing applications is the PID controller [15]. The acronym PID stands for Proportional, Integral and Derivative and can be written as

\[
u(t) = K_P \left( e(t) + \frac{1}{T_I} \int_0^t e(\tau) d\tau + T_D \frac{d}{dt} e(t) \right) \tag{9}\]

The control error can be described as

\[
 e(t) = y_{ref}(t) - y(t) \tag{10}\]

where \(y_{ref}(t)\) is the SP value and \(y(t)\) is the measured output value.

The output from the controller will act as the input to the system and will depend on the control error. The first part of the equation corresponds to the proportional control and will increase proportionally according to \(K_P\) when \(e(t)\) grows. Although a simple P-controller will try to eliminate the control error, disturbances affecting the process and static control errors might still be present. By introducing the integral part, the input
signal will depend on a term that is proportional to the integral of the control error. This will compensate for process noise and eliminate static control errors but might also cause the system to become unstable. Due to the complex nature and slow response times of some systems the PI-controller might overcompensate the control error. When the system finally responds to the input signal it might overshoot the SP value and the control error will remain the same size, or larger, but with opposite sign. This will escalate the output signal and force the system into an unstable state. By observing the current change in the control error and introducing a derivative term to the signal the risk of unstable behavior can be reduced. A major drawback with the derivative term is the problem with deriving the derivative from noise corrupt output measurements. Due to this problem, the derivative term is often ignored and a PI-controller is used. In Figure 7 an overview of the PID-controller is shown.

Figure 7. The PID-controller

2.3.3 Data types

Digital communication through a serial protocol like PROFIBUS requires the data to be represented as multiples of binary bits. A bit is the smallest entity of information in a computer system and can hold either 1 or 0. By grouping multiple bits together in “words” with lengths of 4, 8 or 16 bits the representation of larger values becomes possible. The purpose of different data types is to define constants and variables in computer programs like integers, floating point values and Booleans. The binary number representation has base 2, but there are other representations like hexadecimal with base 16 and the decimal system with base 10.
Table 2 shows the range of representation between the different data types. [7, pp. 35-36]

In the C programming language, the data types character, integer, float and double are called primary data types. Integers can be stored in 8, 16 or 32 bit sizes where one bit can be used as a sign. The float data type is used to represent fractional numbers with a precision of 6 digits allocating 32 bits of memory. If more precision is needed the data type double can be used which allocates 64 bits, corresponding to a double float. A character can be either signed or unsigned and is often used when defining non-arithmetic variables like letters. In more modern programming languages, like C#, other data types like strings are predefined which is basically an array of characters. [16]
### Table 2. Primary data types in C

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Bits Required</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character</td>
<td>8</td>
<td>-128 – 127</td>
</tr>
<tr>
<td>Unsigned Character</td>
<td>8</td>
<td>0 – 255</td>
</tr>
<tr>
<td>Integer</td>
<td>16</td>
<td>-32768 – 32767</td>
</tr>
<tr>
<td>Unsigned Integer</td>
<td>16</td>
<td>0 – 65535</td>
</tr>
<tr>
<td>Float [17]</td>
<td>32</td>
<td>±1.18×10^{-38} to ±3.4×10^{38}</td>
</tr>
<tr>
<td>Double [17]</td>
<td>64</td>
<td>±2.23×10^{-308} to ±1.8×10^{308}</td>
</tr>
</tbody>
</table>

### 2.3.4 PLC

A Programmable Logic Controller, or PLC, is “a special form of microprocessor-based controller that uses programmable memory to store instructions and to implement functions such as logic, sequencing, timing, counting, and arithmetic in order to control machines and processes. [18, p. 3]”. PLCs are similar to regular computers with a CPU, program and data memory, communication interface but more adapted for control tasks in an industrial environment rather than calculations and graphical illustrations. The PLC have interfaces to interact with input devices like sensors and output devices like valves and motors. A PLC can only handle digital values corresponding to either state changes or analog values. Therefore the PLC must be equipped with digital to analog and analog to digital converters. [18, pp. 1-7]

The purpose of an input device is to provide information regarding a state or process value from the outside world. A state can e.g. be if a certain water level is reached or if a motor is on or off etc. This state is often represented as a discrete value of 1 or 0 that corresponds to a high and low voltage e.g. 5 V and 0 V. To change between the states a mechanical, photoelectric or proximity switch is often used together with some logic gates to decide an input value. A process value like temperature, fluid flow and position are represented by analog values with a resolution that depends on the materials used in the sensors. An analog temperature value can be obtained by measuring the change of electrical resistance in materials. [18, pp. 22-39]

The input value is processed in the PLC and a control value is generated and sent to an output device. This value is always a digital representation of a state or an analog value. If the device has an analog value as input, e.g. a current, the value is converted through PWM. A common way to operate a switch is with a relay. A relay can create a magnetic field through a current and attract metal components to create a switch. The result is a
switch that can control very large currents with only a small current through a solenoid. [18, p. 40]

### 2.3.5 PROFIBUS

In order to integrate all components in an automation system a standard for digital communication called PROFIBUS [19] is widely used in industrial applications. PROFIBUS is an open fieldbus standard for serial communication with a three-layer protocol structure; Physical, data link and application.

The evolution of fieldbuses began in the early 20th century with the development of distributed control systems. To control a large industrial process, multiple steps in the production must be coordinated and synchronized. This was mainly done by control operators reading analog meters and manually adjusting the actuators. The demand for a centralized control strategy led to the development of pneumatic and 4-20 mA current standards used together with analog transmission between control rooms and the industrial process. With the advancements in computer science during the 1960s central computers using digital signals replaced many of the analog systems. But it was not until the invention of the integrated circuit and the microprocessor that distributed control systems became capable of real-time computing and control. With digital signal transmission, more and more information can be sent on the same transmission line. And with the development of smart sensors and embedded systems, components become more integrated and the line between converters, actuators and valves become even more indistinct. [19, pp. 143-145]

![Figure 8. Evolution of field communication. (19, p. 144)](image)

### 2.4 .NET Framework

The .NET Framework is a collection of a class library and a runtime environment for applications created to run on a Windows operating system. The framework supports the developer by offering a large class library of precoded solutions for ordinary programming tasks called the Framework Class Library (FCL). Together with the Common Language Runtime (CLR), applications can be written in several different languages and still be compiled and executed together in a single application. FCL and CLR are the two components that constitutes the .NET framework.
CLR is the implementation of the larger Microsoft standard Common Language Infrastructure (CLI). The purpose of CLI is to compile and run applications written in different languages. Instead of compiling the code directly into machine code, CLI use a Common Intermediate Language (CIL) as a platform-neutral language before compiling it to the machine code compatible with the desired platform. This makes it possible to use the same source code for many different operating systems and hardware platforms. Figure 9 illustrates how the CLI can compile and run several languages on a single platform. One of the most common languages in the .NET Framework is C#, pronounced C-Sharp. [20]

![Figure 9. Overview of the .NET framework](image)

3. Theory and model framework

This Section will introduce the mathematical framework my bioprocess model is built upon. It will present a system of ordinary differential equations (ODEs) that provides the possibility to simulate the behavior inside a bioreactor during execution of a cell culture. It will also explain how different parameters like $K_La$ are estimated and how to solve ODEs with a 4th order Runge-Kutta solver. The Section will also briefly explain the software development framework Visual Studio 2015 and the design and development of the hardware interface and connection with an Allen Bradley PLC.
3.1 Cell Growth

To model the cell growth a framework of ODEs for bacterial fermentation has been used. Although there are a lot of differences between cell growth and fermentation a similar behavior can be observed and by adapting necessary parameters the models will be sufficient for my purpose. According to Enfors [21], a bioreactor fermentation system can be represented as a batch, fed-batch or continuous process. The fed-batch and continuous systems have a continuous inlet flow of substrate where the volume in the continuous process will be kept steady and in the fed-batch it will increase.

The four phases of a cell culture, illustrated in Figure 1, are the lag phase, exponential phase, stationary phase and the death phase. To obtain a high cell concentration and to avoid entering the death phase critical environmental parameters like pH, temperature and dissolved oxygen need to be controlled. Before execution the bioreactor system will be prepared with high concentrations of cell medium to avoid limiting the cell growth from start. One big difference when modeling cell growth compared to microbial fermentation is the complexity of the cell medium. In microbial growth, the limiting substance is usually some carbon source, like sugar, and the medium consists of only a few ingredients. In cell culture the medium can consist of up to 100 different ingredients optimized for a specific cell type. It is therefore very hard to identify a single limiting substance and to derive a mathematical expression for this.

After a lag phase where the growth is neglectable, the culture will enter an exponential phase with rapid growth. The definition of exponential growth [22] is when the cell concentration is doubled during each fixed time interval. This time interval is called the doubling time $t_D$ and can be estimated by offline sampling from the cell culture during execution. By initiating the cell culture with an initial concentration $X_0$ the cell concentration at a specific time can be describes as

$$X(t) = X_0 2^{(t-t_0)/t_D}$$

From this equation, we can use the natural logarithm to derive

$$\ln X(t) - \ln X_0 = \frac{1}{t_D} \ln 2$$

Here we can use the definition of the derivative and let $t \to t_0$ which gives

$$\frac{d}{dt} \ln X(t) = \frac{1}{X(t)} \frac{d}{dt} X(t) = \frac{1}{t_D} \ln 2 = \mu$$

where $\mu$ is the specific growth rate [21, p. 11]. From this equation, we can continue to derive the expression for the cell concentration $X(t)$.

$$\frac{dX}{dt} = \mu X$$  \hspace{1cm} (11)
It is not enough to derive the specific growth rate only from the doubling time since the doubling time will change according to the changes in the environment. Therefore, the parameter $\mu$ is not constant and its dependency to the environmental parameters must be described. The most common approach is to describe the specific growth rate with the Monod function

$$\mu(S) = \mu_{\text{max}} \frac{S}{S + K_s} \quad (12)$$

where $\mu_{\text{max}}$ is the maximum growth rate and $K_s$ is a half saturation constant which denotes the substrate concentration that gives half the maximum growth rate. $\mu_{\text{max}}$ can be obtained by observing the shortest doubling time. This is also a critical difference between microbial fermentation and cell culture where the growth in a fermentation process is far faster than in a cell culture [21, p. 12] [6, p. 348]. The maximum growth in a microbial process is usually around $1 \text{ h}^{-1}$ and in cell culture $0.03 \text{ h}^{-1}$ with ideal conditions. Apart from the substrate concentration the specific growth rate is also dependent on environmental variables like pH, temperature and DOT and the Monod function can be extended like

$$\mu(S, pH, T, DOT) = \mu_{\text{max}} \frac{S}{S + K_s} \cdot f(pH) \cdot f(T) \cdot f(DOT) \quad (13)$$

where the product of the environmental variables is equal to 1 if all conditions are ideal. The environmental functions can all be described as second order polynomials with zeroes located at critical levels where the growth rate is completely inhibited. The pH, temperature and DOT limits can be approximated with the help of studies performed by Yoon et al, Furukawa et al and Sellick et al [6] [10] [23].

According to the studies the pH interval where the growth rate is still measurable is between 6 and 8. Outside this interval the viable cell density will rapidly decrease and the growth rate will be close to zero.

$$f(pH) = \begin{cases} 
-(8 - \text{pH})(6 - \text{pH}) & \text{if } 6 \leq \text{pH} \leq 8 \\
0 & \text{else} 
\end{cases} \quad (14)$$

The temperature interval is harder to estimate since the cells are more resistible to low temperatures. The ideal temperature is $37 \degree C$ and the growth rate will be 0 when the temperature reach $30 \degree C$ or less. To obtain a symmetric second order polynomial the upper limit will be set to $44 \degree C$.

$$f(T) = \begin{cases} 
-\frac{1}{49}(44 - T)(30 - T) & \text{if } 30 \leq T \leq 44 \\
0 & \text{else} 
\end{cases} \quad (15)$$

The constant $\frac{1}{49}$ is used to scale the maximum function value to 1.
The dissolved oxygen function can be described with a Monod function similar to the substrate function. This is since the cell growth is not limited by the level of DOT until it reaches a critical level of about 30%. For higher concentrations, no significant inhibitory effects can be observed. The DOT function will therefore be defined as

\[ f(DOT) = \frac{DOT}{DOT + K_{DOT}} \]  

(16)

where \( K_{DOT} \approx 6 \) is the DOT concentration where the specific growth rate will be half of \( \mu_{max} \) with all other environmental parameter equal to 1.

When modeling the substrate consumption considerations must be made if the model shall represent a batch or fed-batch. A fed-batch process model must include a continuous flow of substrate into the vessel and therefore account for change in the total volume. A model for the substrate concentration can be described as

\[ \frac{dS}{dt} = \frac{F}{V} (S_i - S) - \frac{\mu}{Y_{s/x}} X \]  

(17)

where \( F \) is the inlet flow, \( V \) the current volume, \( S_i \) the inlet flow substrate concentration, \( S \) the current substrate concentration, \( Y_{s/x} \) is the yield from unit substrate to cells, \( \frac{F}{V} \) is sometimes called the dilution rate which will account for different concentrations between the inlet flow and current medium. The parameter \( Y_{s/x} \) ranges between 0 and 1 and depends heavily on the cell line used and the composition of the medium [21, p. 15]. The biomass concentration will also be affected by the inlet flow and therefore the dilution rate shall be a part of the model.

\[ \frac{dX}{dt} = \left( \mu - \frac{F}{V} \right) X \]  

(18)

The rate of volume change can be described by the inlet flow

\[ \frac{dV}{dt} = F \]  

(19)

To reduce the dilution, the feed solution should have very high concentrations of the limiting component. This is, as mentioned before, very hard to accomplish for cell media with many possible limiting components.

3.2 Dissolved Oxygen Tension

Most of the theory and some of the necessary parameters behind dissolved oxygen was presented in Section 2.2.2. The complete model for the dissolved oxygen tension can be described as
\[ \frac{d\text{DOT}}{dt} = K_L a (\text{DOT}^* - \text{DOT}) - q_o X H \]  

(20)

where H is a Henry’s law derived constant that relates the dissolved oxygen concentration in g/L to % saturation [21, p. 39]. The constant depends on the partial pressure of oxygen in the supplied gas and Henry’s constant which depends on the current temperature and pressure. For simplicity, we assume H = 14000 % which corresponds to a solubility of oxygen in growth medium of 7.14 mg/L in equilibrium with air. The mass balance equation stated above can be described as the difference between the oxygen transfer rate (OTR) and the oxygen consumption rate (OCR) where

\[ \text{OTR} = K_L a (\text{DOT}^* - \text{DOT}) \]

and

\[ \text{OCR} = q_o X H \]

The OTR can be described as the solubility of oxygen in the given medium at a specific time. \text{DOT}^* denotes the highest possible levels of dissolved oxygen in the medium and is equal to 100 % if the supplied gas is air. If the gas is complemented with additional supply of pure oxygen this value will be higher and if complemented with N2 the value will be lower. DOT denotes the current level of dissolved oxygen in the medium and if this value is equal to 0 we get the maximum oxygen transfer capacity \( K_L a \text{DOT}^* \). The OTR is heavily dependent on the parameter \( K_L a \) where a large value is necessary to achieve efficient oxygen transfer. Internal research conducted by GE Healthcare [9] derives a multivariate regression model for estimating \( K_L a \) with respect to volume, agitation rate and gas supply.

\[ K_L a = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 \]  

(21)

where \( x_1 = \text{volume, } x_2 = \text{agitation, } x_3 = \text{air flow, } x_4 = \text{gas}^2 \text{ and } x_5 = \text{agitation} \cdot \text{gas}. \)

The OCR corresponds to the oxygen demand in the cell culture [21, p. 31]. Most of the oxygen consumed during cell culture is used for respiration and the cells will therefore produce carbon dioxide. A part of the CO2 will be transferred back to the bubbles and removed through ventilation when the bubbles reach the surface and some parts will be kept dissolved in the medium.

3.3 pH Function

As described in Section 2.2.1 pH is close to impossible to describe with conventional linear dynamic models. According to internal studies conducted by GE Healthcare [24] the dynamic behavior of pH can be simplified with a first order system with time delay with the transfer function
\[ G_{pH}(s) = e^{-ls} \frac{K_{pH}}{\tau s + 1} \] (22)

where \( K_{pH}, \tau \) and \( L \) are three lumped parameters estimated through empirical step response experiments with \( CO_2 \). \( K_{pH} \) is the system gain, \( \tau \) sets the system speed and \( L \) is the time delay. Other research [25] studied the time to reach 95% of set point in a pH step change using acid and base addition. The study suggests a multivariate regression model to predict the response time with respect to volume and agitation rate.

\[ T_{95} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 \] (23)

where \( x_1 = \text{volume}, x_2 = \text{agitation}, x_3 = \text{agitation}^2, x_4 = \text{volume} \cdot \text{agitation}. \)

Generally, pH change in a reactor can have many different causes and through addition of acid, base and \( CO_2 \) the pH is controlled within certain limits to avoid harming the cell culture. One source of natural addition of \( CO_2 \) is through cell respiration. Studies of large-scale and high-density CHO cell cultures [26] show that the production rates of \( CO_2 \) from CHO cells can reach \( 5.36 \cdot 10^{-17} \text{mol cell}^{-1} \text{sec}^{-1} \). With cell densities in the range of \( 10^6 \text{cells ml}^{-1} \) this will certainly lower the pH value.

### 3.3.1 Buffer Systems

A common way to avoid rapid changes in pH is to use a buffer system. There are many different types of buffers but in cell culture the most common is the Bicarbonate buffer system [27]. The purpose of a buffer system is to balance the pH through neutralization of waste products produced by the cells with a weak acid and its conjugated weak base. In the case of the bicarbonate buffer the weak acid is \( H_2CO_3 \) and the conjugated base is \( HCO_3^- \) when a strong acid such as HCL is added. This might be the case when an operator wants to lower the pH in the reactor. The balance equation can be written like

\[ NaHCO_3 + HCL \leftrightarrow H_2CO_3 + NaCL \] (24)

If the pH must be higher, a strong base such as NaOH can be supplied and the following buffer reaction can be described as

\[ NaHCO_3 + NaOH \leftrightarrow 2Na^+ + CO_3^{2-} + H_2O \] (25)

where the carbonate ion \( CO_3^{2-} \) is the conjugated base of the more acidic bicarbonate \( HCO_3^- \). As described above the buffer system will reach an equilibrium between an acid and a conjugated base and therefore reduce the impact on the pH. Equilibrium can also be reached with dissolved \( CO_2 \) together with \( H_2O \) according to the following balance equation.

\[ CO_2 + H_2O \leftrightarrow H_2CO_3 \leftrightarrow HCO_3^- + H^+ \] (26)
More generally [28] a buffering capacity can be described as “the number of moles of H\(^+\) that must be added to 1 L of solution to decrease the pH by 1 unit” [29]. The better buffering capacity, the larger number of H\(^+\) ions need to be added to provoke a pH change. The general balance equation between an acid (HA) and its conjugated base (A\(^-\) ) can be described as

\[
HA \leftrightarrow H^+ + A^- \tag{27}
\]

where the forward reaction rate is dependent on the acid concentration [HA] and can be written like \(k_f[HA]\). The reverse reaction rate can be expressed as \(k_r[H^+][A^-]\) and if the system reach equilibrium the \(k_f\) and \(k_r\) will be equal. The fraction between the two constants is called the equilibrium constant or the acid constant \(K_a\).

\[
K_a = \frac{[H^+][A^-]}{[HA]} = \frac{k_r}{k_f} \tag{28}
\]

From this expression, we can derive the Henderson-Hasselbalch equation [28].

\[
pK_a = -\log_{10} \left( \frac{[H^+][A^-]}{[HA]} \right) \tag{29}
\]

\[
pK_a = -\log_{10} \left( \frac{A^-}{HA} \right) - \log_{10}([H^+]) \tag{30}
\]

\[
pK_a = -\log_{10} \left( \frac{A^-}{HA} \right) + pH \tag{31}
\]

\[
pH = pK_a + \log_{10} \left( \frac{A^-}{HA} \right) \tag{32}
\]

The \(pK_a\) for the bicarbonate buffer system will obtain different values depending on the balance equation. When adding a strong acid such as HCL to the buffer the pH will be derived according to

\[
pH = pK_{a_{H_2CO_3}} + \log \left( \frac{[HCO_3^-]}{[H_2CO_3]} \right) \tag{33}
\]

where \(pK_{a_{H_2CO_3}} = 6.1\). For addition of NaOH the \(pK_a = 10.3\) since the conjugated acid is \(HCO_3^-\). For addition of CO\(_2\), the \(pK_a\) value must be derived with respect to the amount of dissolved CO\(_2\) instead of the \(H_2CO_3\) concentration.

\[
pH = pK_{a_{H_2CO_3}} + \log \left( \frac{[HCO_3^-]}{[CO_2]_{aq}[H_2O]} \right) \tag{34}
\]

where \([CO_2]_{aq} = k(pCO_2)\) and \(pCO_2\) is the partial pressure of CO\(_2\) in the gas phase and \(k\) is a Henry’s constant for CO\(_2\). Since the value of \(k = 0.0229\ mol\ atm^{-1}\) we can
derive a new expression relating the pH to the partial pressure of CO₂. The \( H_2O \) term is constant and can be ignored.

\[
pH = pK_{aH_2CO_3} + \log \left( \frac{[HCO_3^-]}{k(pCO_2)} \right)
\]

\[
pH = pK_{aH_2CO_3} + \log \left( \frac{[HCO_3^-]}{pCO_2} \right) - \log(k)
\]

\[
pH = 6.1 + \log \left( \frac{[HCO_3^-]}{pCO_2} \right) + 1.64
\]

\[
pH = 7.74 + \log \left( \frac{[HCO_3^-]}{pCO_2} \right)
\]

We now have three different \( pK_a \) values for each pH changing medium together with a bicarbonate buffer system.

### 3.4 Temperature Model

The temperature control in an agitated bioreactor system is often performed with a separate *Temperature Control Unit (TCU)* that control the flow of a secondary fluid through the jacket surrounding the steel vessel. The control procedure is illustrated in the figure below where the medium and biomass is the primary fluid and water is the secondary fluid. For the XDR systems two temperature transmitters are located inside the vessel in contact with the primary fluid to provide process values for the PLC in order to control this temperature with a PID algorithm. The PID will produce a control value that will work as the set point for the PID inside the TCU for controlling the secondary fluid.
3.4.1 Heat Transfer Equations

Heat transfer in an agitated vessel [30] such as the XDR is performed with a half-pipe jacket as illustrated in Figure 11. The secondary fluid will flow from the TCU, through the half-pipes and heat or cool the primary fluid before returning to the TCU for heating or cooling.

Figure 11. The half-pipe jacket [30]

The flow of heat from the secondary to the primary fluid, or in the opposite direction, can be described by convection and heat conduction. Convection [31] is a terminology within thermodynamics to describe the heat transfer from molecular movement of fluids and gases. Convection will cause the heat transfer from the secondary fluid to the vessel wall and from the wall to the primary fluid. Heat conduction will cause the heat to transfer from one side of the wall to the other side. To reduce heat loss to the surrounding environment a layer of insulation is place between the jacket and the outside wall.

To calculate the heat transfer from the secondary fluid to the primary fluid through a stainless-steel surface, the following equation can be used.

\[
\frac{dE}{dt} = kA\Delta T
\]  

(36)

where \(E\) = Heat flow through wall \([W]\), \(A\) = Transfer area \([m^2]\), \(\Delta T\) = Temperature differens between primary and secondary fluid \([K]\).

The heat transfer coefficient \(k\) can be derived according to

\[
k = \frac{1}{\left(\frac{1}{\alpha_v}\right) + \left(\frac{1}{\alpha_k}\right) + \left(\frac{\delta}{\lambda}\right)}
\]

(37)

where \(\alpha_v\) = Heat transfer coefficient for the warmer fluid \([\frac{W}{m^2\cdot K}]\), \(\alpha_k\) = Heat transfer coefficient for the colder fluid, \(\delta\) = wall thickness \([m]\), \(\lambda\) = Heat transfer coefficient for wall.
To derive the temperature in the primary fluid one can use the following equation

\[ T = \frac{Q}{cm} \]  

(38)

where \( Q = \frac{E \Delta t}{1000} \) is the total heat added in kJ and \( \Delta t \) is a limited time interval, \( c \) is a specific heat constant \( \left[ \frac{kJ}{kgK} \right] \) and \( m \) is the mass in Kg. For water the constant \( c = 4.1814 \ kJ/KgK \).

To account for heat loss from the secondary fluid to the insulation the same equations can be used but with a different \( k \) value with heat transfer coefficient and wall thickness for the insulation. And since the heat loss is between the secondary fluid and the outside air the heat transfer coefficient for air should be used.

### 3.5 Model summary

The following section will provide a summary of the final models used to simulate the behavior of a bioreactor. The dynamics of cell growth, medium concentration and volume can be written as

\[ \frac{dX}{dt} = \left( \mu - \frac{F}{V} \right) X \]  

(39)

\[ \frac{dS}{dt} = \frac{F}{V} (S_i - S) - \frac{\mu}{Y_{s/x}} X \]  

(40)

\[ \frac{dV}{dt} = F \]  

(41)

where the parameter \( \mu \) is the specific growth rate dependent on substrate concentration, pH, temperature and DOT. \( \mu \) can be described as

\[ \mu(S, pH, T, DOT) = \mu_{max} \frac{S}{S + K_s} \cdot f(pH) \cdot f(T) \cdot f(DOT) \]  

(42)

where

\[ f(pH) = -(8 - pH)(6 - pH) \]  

(43)

\[ f(T) = -\frac{1}{49} (44 - pH)(30 - pH) \]  

(44)

\[ f(DOT) = \frac{DOT}{DOT + K_{DOT}} \]  

(45)
The Dissolved Oxygen Tension (DOT) can be derived through the difference between the oxygen solubility in the medium and the Oxygen Consumption Rate (OCR). Together with the parameter $K_L a$ the mass balance equation can be written as

$$\frac{d \text{DOT}}{dt} = K_L a (\text{DOT}^* - \text{DOT}) - q_o X H$$  \hfill (46)

The pH will be derived with the equations described in Section 3.3.1 with a $H_2CO_3$ buffer system depending on addition of HCL and CO$_2$. The equation can be written as

$$pH = pK_a H_2CO_3 + \log \left( \frac{[HCO_3^-]}{[H_2CO_3]} \right)$$  \hfill (47)

for the equilibrium with HCL and as

$$pH = 7.74 + \log \left( \frac{[HCO_3^-]}{pCO_2} \right)$$  \hfill (48)

for CO$_2$. The heat transfer balance equation can be written as

$$\frac{dE}{dt} = k A \Delta T$$  \hfill (49)

where

$$k = \frac{1}{\left( \frac{1}{\alpha_v} + \frac{1}{\alpha_k} \right) + \left( \frac{\delta}{\lambda} \right)}$$  \hfill (50)

$$T = \frac{Q}{cm}$$  \hfill (51)

4. Result

This Section will present the results from the software implementation of the theoretical framework presented in Section 3. The connection to the control system through simulation of the PROFIBUS slaves will also be briefly explained, although most of this development is not included in this thesis. Illustrations of the system and software architecture together with screenshots of the PROFIBUS Simulator and sample code implementations can be found in Appendix A, B and C.

To establish communication between the PLC and the Bioreactor Simulator, a master-slave relationship must be established between the PC and the PLC. The PLC is equipped with a PROFIBUS master that can control multiple slaves. In a bioreactor, the slaves are devices used for controlling the environment inside the reactor, like MFCs,
pumps and transmitters. The Bioreactor Simulator software must therefore be complemented with a slave simulator that can simulate the slaves and communicate with the PLC through PROFIBUS. The slave simulator must also receive PROFIBUS signals from the PLC and communicate them to the Bioreactor Simulator software. Figure 12 shows how the slave simulator, in the figure named PROFIBUS Simulator, receives and transmits PROFIBUS signals from the PROFIBUS card and pass the information on to the Bioreactor Simulator.

The PLC used during development is an Allen-Bradley PLC running software from Rockwell Automation. The PLC software can be monitored and controlled by running Rockwell software on the PC and connecting the PLC with an Ethernet connection. Figure 22 in Appendix A shows a more detailed view of the slave simulation process.

![Figure 12. Overview of the communication paths between PC and PLC](image)

After the main development phase the setup described in Figure 12 was replaced by a complete DCS, illustrated in Figure 13, used for controlling a complete biopharmaceutical factory. This DCS can control up to 9 bioreactors simultaneous together with multiple other units necessary to produce recombinant therapeutic proteins. The control system is composed of two redundant PLCs to have a backup if the main PLC malfunctions. The repeater cabinet is a signal amplifier and a network switch for connecting multiple units to every PLC master.
Figure 13. Communication with the DCS controlled with an Operator Interface Terminal (OIT)

4.1 Software

The Bioreactor Simulator is implemented using Visual Studio 2015 and C#. Since the main purpose of the software is to support automation software developers in the testing and verification process, the focus of the software is around the controlled parameters. The software architecture is designed as layers to have a clear separation of the different tasks. An illustration of the design can be found in Appendix A Figure 23 where the software is divided into function, reactor, device, signal and communication layers.

4.1.1 Communication layer

The layer closest to the PROFIBUS Simulator is called the communication layer and is implemented as the class InputOutputThread. It is responsible to establish the communication pipe with the PROFIBUS Simulator using the C# library NamedPipe. When the application is executed the software creates the server-side instance of the communication and waits for a client to connect. As illustrated in Figure 14, the client in this application is the PROFIBUS simulator that execute a client-side script which will establish a connection with the server. After the client-side connection is created the script will wait for an acknowledgment as a confirmation that the connection now is established. The user will now be able to start the simulation from the GUI and when this happens another signal will be sent to the PROFIBUS Simulator saying the reactor is enabled. The client and server will now be able to send and receive input signals and process values.
The signals that represent the reactor input signal are sent through the NamedPipe as a C# struct according to the code in Figure 15.

```csharp
public struct ReactorInput {
    public MFC mfc_1;
    public MFC mfc_2;
    public MFC mfc_3;
    public MFC mfc_4;
    public MFC mfc_5;
    public MFC mfc_6;

    public Pump pump_1;
    public Pump pump_2;
    public Pump pump_3;
    public Pump pump_4;

    public Agitator agitator;
    public TCU tcu;
}
```

**Figure 14. Connection establishment between Bioreactor Simulator and PROFIBUS Simulator**

**Figure 15. C# structure containing devices with input values**
4.1.2 Signal layer

The signal layer is implemented as the class MessageHandler and receives input signals from the communication layer. The input signal is a data structure of type struct containing structs representing the devices with raw values. The main task for the MessageHandler is to create a mapping from the raw values received from the PLC to the physical units of the input and output devices. A summary of all input devices and the corresponding physical values can be found below in Table 3.

Table 3. Input device raw values and physical units

<table>
<thead>
<tr>
<th>Device</th>
<th>Raw Value</th>
<th>Data Type</th>
<th>Physical Value</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFC</td>
<td>0.0-100.0</td>
<td>Float</td>
<td>0.0-100.0</td>
<td>LPM</td>
</tr>
<tr>
<td>Pump</td>
<td>15 bit 0-32767</td>
<td>UInt16</td>
<td>0 – 360</td>
<td>RPM</td>
</tr>
<tr>
<td>Agitator</td>
<td>0-3600</td>
<td>UInt16</td>
<td>0 – 360</td>
<td>RPM</td>
</tr>
<tr>
<td>TCU</td>
<td>15 bit 0-32767</td>
<td>UInt16</td>
<td>-10 – 60</td>
<td>°C</td>
</tr>
</tbody>
</table>

After the mapping, a new structure like the one in Figure 15 is created with mapped values and is passed on to the reactor layer.

4.1.3 Reactor layer

The reactor layer contains the reactor class with instances of each object in the function layer. The reactor object receives input values from the signal layer and communicates with the corresponding function. The reactor must respect that i.e. the pH function will be affected by both acid and base pumps, and MFCs containing CO$_2$. Therefore, the reactor need information about what type of medium the corresponding device contains. Since the different environmental parameters in the function layer affect each other, the reactor must coordinate the process values and respect the dependency between some of the functions.

4.1.4 Function layer

The function layer is the top layer in Figure 23 in Appendix A. The different boxes each represent a class that is initiated from the reactor class. This is where most of the theory is implemented. Every function is implemented independently from each other but, as mentioned above, since the functions are dependent they will receive all necessary information from the reactor.

The cell growth function contains all models necessary to describe the dynamic behavior of the specific growth rate, substrate concentration and biomass concentration.
The ODEs are defined according to the code in Figure 24 in Appendix B. The ODEs are all solved with a 4th order Runge-Kutta solver. The implementation can be seen in Figure 25 in Appendix B where the solver for the biomass concentration is shown.

The dissolved oxygen function contains both an ODE describing the dynamic behavior of the DOT and a parameter estimation of the kLa. The implementation can be seen in Figure 26 in Appendix B where a call to the static class ParameterEstimator calculates a kLa value with respect to the agitator, air flow and volume. The SetDOTstar method sets the maximum dissolved oxygen tension described in Section 3.2.

The pH function is implemented in three steps where a pH-increment is calculated for acid, base and CO₂ individually. The implementation can be seen in Appendix B Figure 27. The first step is CO₂ where the CO₂ production from the cells and the CO₂ input from MFCs are summarized to calculate the partial pressure of CO₂ of the total gas supply. This percentage can then be used in the Henderson-Hasselbalch equation to calculate an increment. The CO₂ is followed by the acid and base calculations according to the equations presented in Section 3.3.

### 4.1.5 GUI

The purpose of the GUI is to provide a simple way to specify initial values and conditions, and to get an overall perspective of the process during execution. This GUI will work as a complement to the control system HMI. Before executing the software, the reactor must be deployed in the control system and the PC must be properly connected to the DCS through a repeater cabinet. The user must also provide information about the control mapping through the HMI. A mapping can i.e. be a split range control mapping of the pH with acid for lowering and base for increasing. The user can also choose to use CO₂ for lowering the pH and the same mapping must be specified for the dissolved oxygen control. When launching the software, the user will see a dialog according to Figure 16 prompting for reactor and TCU specifications. The user must choose reactor and TCU type with initial weight values and starting temperature.
Figure 16. TCU and reactor settings with initial weight and temperature values

Even more detailed information about reactor measurements and fluid properties can be specified under the **Constant** tab described in Figure 17.

**Figure 17. Constants for heat transfer equations described in Section 3.4.1**

By pressing the **Apply** button the user will be transferred to the main dialog described in Figure 18. The top left of the dialog is composed of a console for providing real time updates of the application status to the user. The bottom left is composed of three views where the user specifies initial parameter values, MFC specifications and pump specifications.
Figure 18. Main view of the simulator when waiting for the PROFIBUS Simulator to connect

Figure 19 shows the setting for the MFCs and pumps. The user must synchronize this information with the deployed reactor in the DCS. A reactor can have a maximum of six MFCs and four pumps but often a reactor is not fully equipped since each control variable can only be controlled by two devices. The user can therefore select which devices to use.
The figure below shows an execution with acid and base as control devices for the pH control. The oscillating behavior of the growth rate is a consequence of the overshoots in the pH as it tries to reach the set point of 7.2.

Figure 21 shows the process variables view during execution. This offers the opportunity to terminate the dynamic generation of process values and instead force desired values. This is mainly implemented as a tool during testing to test unusual and sometimes dangerous situations to trigger alarms and other safety measures in the HMI. This is something you usually do not want expose the real equipment to since this can cause severe damage to the hardware.
Figure 21. Process variables during execution
5. Discussion and conclusions

The main purpose of this project was to develop a simulator of the XDR bioreactor system and connect it to the DCS controlling the real biopharmaceutical factory. The control system should not be able to distinguish between a real bioreactor and the simulator. The project has therefore been divided into one part of simulator development and one part of establishing connectivity with the control system. The later part has not been a part of this thesis and is excluded from this report. The software has been implemented in a .NET environment to ensure proper execution on a Windows operating system. To simulate the bioprocess and environmental parameters appropriate models have been derived from a theoretical framework and implemented as function blocks in the software. During execution, the PID controller can receive approximate process values from the simulator, derive control values for the appointed devices and transmit the control values back to the simulator to generate a new process value that converge towards the set point value. This section will discuss the results shown in Section 4 and reason about the general advantage of developing virtual environments to postpone hardware dependencies. A discussion regarding the limitations with the current simulator will be presented together with some suggestions of further development and improvements.

The general advantage of using simulators is when a limited access to appropriate hardware becomes a bottleneck in the software development process. Many of the most common procedures during testing and verification can be performed towards a virtual environment as well as towards the real unit. Of course, the final tests must be performed on real hardware, but with access to a virtual environment the developers can be more dynamic and flexible when scheduling tests. Since the simulator environment is only dependent on a PROFIBUS card and a PC, multiple instances can be created and therefore a whole factory can be simulated. This can be very useful when i.e. performing load tests on the control system or the servers logging all traffic. If the bioprocess models are improved, the simulator might also be used for process optimization and tuning of PID parameters before testing on a real cell culture. This is, however, not recommended with the current version of the simulator since the models are still very primitive. Testing of critical and abnormal situations is a crucial part of the verification process and is very hard to perform on real hardware. By using a virtual environment, the developer can test all possible scenarios without jeopardizing equipment worth millions of dollars or being exposed for hazardous risks. The simulator can also be used as an educational tool or as support when demonstrating executions for a potential customer without access to hardware.

Further development of the current software should mainly be focused on the models simulating pH. Development and implementation of this model has been one of the hardest tasks of this project and the framework presented in Section 3.3 is only appropriate for small changes in pH with limited deviation from the pKa value. This is due to the very nonlinear nature of the pH scale explained in Section 2.2.1. This can
also be seen in Figure 20 where the pH is oscillating heavily in the beginning as the control system has a hard time understanding the dynamics of the simulator. A more extensive vision for the simulator concept should be to simulate a complete biopharmaceutical factory with all types of units like Mixers, Filtration systems and Chromatography systems\(^1\).

\(^1\) A system for protein purification developed by GE Healthcare
References


44


45
Appendix A

Figure 22. System architecture from the automation point of view

Figure 23. Software architecture of the Bioreactor Simulator
Appendix B

private void DefineODEs()
{
    GrowthRate =
           (s, temp, ph, dot) =>
           ((my_max * s) / (half_sat + s)) * (-((1 / 49d) * (30d - temp) * 
(44d - temp)) * ((8d - ph) * (6d - ph)));

dXdt =
       (t, x, my_val, volume) =>
       my_val * x;

dSdt =
       (t, s, x_val, my_val, volume) =>
       (-my_val / yield) * x_val;
}

Figure 24. Definition of ODEs in the cell growth function

private double rk4_biomass(Func<double, double, double, double, double> func, 
  double t, double x, double my, double v, double dt)
{
    double halfdt = 0.5 * dt, k1, k2, k3, k4;
    double ret_val;
    k1 = func(t, x, my, v);
    k2 = func(t + halfdt, x + halfdt * k1, my, v);
    k3 = func(t + halfdt, x + halfdt * k2, my, v);
    k4 = func(t + dt, x + dt * k3, my, v);
    ret_val = x + (dt / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
    return ret_val;
}

Figure 25. Implementation of 4th order Runge-Kutta

public double GetDOT(double time, double dot_current, double x_current, double volume, 
  double agitator, double air_flow, double oxygen_flow, 
  double n2_flow)
{
    kla = ParameterEstimator.GetKla(agitator, volume, air_flow);
    dot_star = SetDOTstar(air_flow, oxygen_flow, n2_flow);
    return rk4_oxygen(dDOTdt, time, dot_current, kla, dot_star, qo, x_current, henry, 
           step_length);
}

Figure 26. Method for calculating the dissolved oxygen.
public double GetPH(double time, double ph_current, double agi, double vol, 
        double flow_acid, double flow_base, double co2, double air, 
        double biomass, double temp)

    {
        double acid_input_conc = 0.1;
        double base_input_conc = 0.1;
        double input_acid = (flow_acid / 1000) * acid_input_conc;
        double input_base = (flow_base / 1000) * base_input_conc;
        double gas_end_ph = 0; double fluid_end_ph = 0;
        double gas_increment = 0; double fluid_increment = 0;
        double t95_fluid = ph_estimator.GetPHResponse(agi, vol);

        buffer_concentration = buffer_concentration - input_acid + input_base;
        if (co2 != 0)
        {
            // CO2 AND BUFFER PH CALCULATIONS
            double gas_constant = 8.314; // L*kPa/K/mol
            double temp_kelvin = temp + 273.15;
            double gas_increment = (gas_end_ph - ph_current) / t95_gas;
            double fluid_increment = (fluid_end_ph - ph_current) / t95_fluid;

            // Data from empirical ph study with co2. Unit seconds
            double t95_gas = 4500;
            double pka_co2 = 7.74;
            double pp_co2 = (co2 + co2_cell_vol) / (air + co2 + co2_cell_vol);
            if (buffer_concentration / pp_co2 <= Math.Pow(10, -7.74))
            {
                gas_end_ph = 0;
            } else
            {
                // Henderson Hasselbalch law pH = pKa + log([HCO3]/pCO2)
                gas_end_ph = pka_co2 + Math.Log10(buffer_concentration / pp_co2);
            }
        } else 
        {
            //pH calculations from acid and base addition
            hco3 = hco3 + input_base - input_acid; // Conjugated base
            h2co3 = h2co3 + input_acid - input_base; // Conjugated acid
            // Avoid division with zero and infinite logarithm
            if (hco3 <= 0)
                hco3 = 1e-4;
            if (h2co3 <= 0)
                h2co3 = 1e-4;

            // Henderson Hasselbalch equation
            fluid_end_ph = pka_h2co3 + Math.Log10(hco3 / h2co3);
            // Calculate increment with respect to mixing time
            fluid_increment = (fluid_end_ph - ph_current) / t95_fluid;

            // pH calculations from acid and base addition
            hco3 = hco3 + input_base - input_acid; // Conjugated base
            h2co3 = h2co3 + input_acid - input_base; // Conjugated acid
            // Avoid division with zero and infinite logarithm
            if (hco3 <= 0)
                hco3 = 1e-4;
            if (h2co3 <= 0)
                h2co3 = 1e-4;

            // Henderson Hasselbalch equation
            fluid_end_ph = pka_h2co3 + Math.Log10(hco3 / h2co3);
            // Calculate increment with respect to mixing time
            fluid_increment = (fluid_end_ph - ph_current) / t95_fluid;
        }
    }

Figure 27. Implementation of pH function

49
//Heat transfer area for XDR reactor
A_bath = 2 * Math.PI * R_bath * H_bath + 2 * Math.PI * R_bath;
A_lid = 2 * Math.PI * R_bath;
Transfer_wall_height = (m_primary / 1000) / A_lid; // new
A_bottom = 2 * Math.PI * R_bath;
A_wall = 2 * Math.PI * R_bath * H_bath;
A_bath_transfer = 2 * Math.PI * R_bath * Transfer_wall_height;

Q_primary_start = Cp_primary * m_primary * T_start_primary;
Q_secondary_start = Cp_secondary * V_secondary * T_start_secondary;
Q_change = TCUcounter == 0
 ? ((Q_primary_start * 1000 + Q_trans_rate * preTime_sec) / 1000)
 : (preQ_change - Q_trans_rate * (Time_sec - preTime_sec) / 1000);

T_primary = Q_change / (m_primary * Cp_primary);
U = T_secondary_output_heating > 50
 ? 1 / (1 / h_water2 + bath_thickness / bath_k_value + 1 / h_water2) * 1.15
 : 1 / (1 / h_water2 + bath_thickness / bath_k_value + 1 / h_water2);
Q_trans_rate = TCUcounter == 0?0:
- U * ((T_secondary_output_heating + T_secondary_input) / 2 - T_primary) * A_bath_transfer;

Q_trans = Q_trans_rate * (Time_sec - preTime_sec);
if (TCUcounter == 0)
{
  Q_heat_loss_heating = 0;
}
if (TCUcounter >= 1)
{
  Q_heat_loss_heating = -((1 / (1 / h_water1) + (1 / h_air) + (Ins_thickness / Ins_k_value))) * A_bath * ((T_secondary_output_heating + T_secondary_input) / 2 - T_amb) + (1 / (1 / h_air) + (Lid_thickness / Lid_k_value))) * A_lid * ((T_secondary_output_heating + T_secondary_input) / 2 - T_amb)) * (Time_sec - preTime_sec) + (Cp_bath * 1000 * A_bath * bath_thickness * p_bath) * (T_secondary_output_heating - T_secondary_input));

Figure 28. Implementation of heat transfer functions
Appendix C

Figure 29. The PROFIBUS Simulator with Reactor 2 strategy and Bioreactor function layer