Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Stina Holen Friisø

Automation of system identification during animal and human trials

Master's thesis in Cybernetics and Robotics
Supervisor: Øyvind Stavdahl and Maria Vatshaug Ottermo
December 2019

NTNU
Norwegian University of
Science and Technology

Stina Holen Friisø

Automation of system identification during animal and human trials

Master's thesis in Cybernetics and Robotics
Supervisor: Øyvind Stavdahl and Maria Vatshaug Ottermo
December 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Abstract

Climate change is a hot topic nowadays, and with ever growing cities comes increased air pollution. This may lead to a number of diseases, with chronic obstructive pulmonary disease (COPD) being one of them. COPD is one of the leading causes of death in the world, mostly due to smoking, but the number of cases is predicted to increase as the air becomes more polluted.

Today's solution for patients who need long term oxygen therapy (LTOT) is manual adjustment of the oxygen delivered from an oxygen tank. The delivered oxygen level is simply based on the doctor's recommendations and the patient's experience of breathlessness. In other words, this is not an ideal solution, and can lead to reduced life quality for the patients. The research project OxyAid, of which this project is a part, aims to automatically adjust the oxygen level based on the patient's actual need.

In order to achieve automatic control of the oxygen level there are several sub systems that must work together to form the whole control system in total. Thus, an overview of the system in its entirety is given. The identification process is one of these subsystems. The aim of this project is to design and evaluate an identification process within the given scope. This entails using some existing programs, and identify and implement new ones where needed.

The identification process has been designed, including implementation of a mathematical model used for parameter estimation developed by Øyvind Stavdahl and Maria V. Ottermo. Tools and techniques that have been used for the process design have been elaborated and explained, along with with theory necessary to understand the results from the system identification itself, supported by literary findings.

The process has been thoroughly tested as possible within the scope of time and available datasets. The new process was proved to be significantly more effective than the previously used procedure. As time is of the essence during a trial, it is important that a process is as effective as possible. The parameter estimation yielded good results, unlike previously used mathematical models. However, more datasets are needed in order to validate the conclusion. It would also be greatly beneficial to test the system in a clinical trial. Further suggestions for relevant future work has been provided.

Sammendrag

Klimaendringer er et veldig dagsaktuelt tema, og med voksende byer kommer økt luftforurensing. Dette kan føre til en rekke sykdommer, med kronisk obstruktiv lungesykdom (KOLS) som en av dem. KOLS er en av de vanligste dødsårsakene i verden, og skyldes som oftest røyking, menantall tilfeller er spådd å øke samtidig som luften blir mer forurenset.

Dagens løsning for pasienter som trenger langtidsbehandling med oksygen (LTOT) er manuell justering av oksygennivået levert til pasienten fra en oksygentank. Nivået blir justert basert på legens anbefalinger og pasientens følelse av kortpustethet. Dette er med andre ord ikke en optimal løsning, og kan føre til redusert livskvalitet. Forskningsprosjektet OxyAid, som dette prosjektet er en del av, har som mål å automatisk justere oksygennivået basert på pasientens faktiske behov.

For å oppnå automatisk kontroll av oksygennivået må en rekke undersystemer fungere sammen for å utgjøre det totale kontrollsystemet. Derfor er en oversikt over systemet i sin helhet gitt i denne rapporten. Identifikasjonsprossen utgjør en av disse undersystemene. Målet med dette prosjektet er å designe og evaluere en identifikasjonsprosess innenfor gitte rammer. Dette innebærer bruk av eksisterende programvare, og identifisering og implementasjon av nye der det er nødvendig.

Identifikasjonsprosessen har blitt designet og implementert, inkludert implementasjon av en matematisk modell utviklet av Øyvind Stavdahl og Maria V. Ottermo, som er brukt i parameterestimeringen. Verktøy og teknikker som er brukt i systemdesignet er beskrevet og forklart, sammen med teorien som er nødvendig for å forstå resultatene av identifikasjonsprosessen.

Prosessen har blitt testet så grundig som mulig innenfor gitt tidsramme og tilgjengelige datasett. Den nye prosessen viste seg å være betydelig mer effektivt enn prosedyren som ble brukt tidligere. Siden tiden er begrenset i et forsøk er det viktig at prosessen er så effektiv som mulig. Parameterestimering ga gode resultater, til forskjell fra tideligere brukte matematiske modeller. Derimot må prosessen testes med flere datasett for å kunne validere konklusjonen. Det ville også vært verdifullt å teste prosessen i et klinisk forøk. Videre forslag for fremtidig arbeid er blitt gitt.

Preface

The work presented in this project has been done in association with the Department of Engineering Cybernetics at the Norwegian University of Science and Technology (NTNU). The report presents my masters project as the final part of the Master of Science education in Cybernetics and Robotics in the fall of 2019.

Getting to work with a project for the OxyAid project has been very rewarding as it has made the work feel meaningful. It has also been very educational, both in regards to the acquired academic knowledge and the knowledge obtained by being a part of a research team. In addition, it has been motivational to work on a project which was intended to be used in real trials. I therefore wish to express my thanks for letting me be part of such an important project, and trusting me with the task.

The project has been completed under the supervision of Øyvind Stavdahl and Maria Vatshaug Ottermo, to whom I would like to express my sincere thanks. Øyvind for our weekly meetings, positive attitude and helpful feedback and discussions. I would like to thank Maria for being available whenever needed either through e-mails or with meetings consisting of feedback on the technical system, proofreading the report and always providing good advice if needed.

I would also like to thank my boyfriend, Andreas, for his support throughout the semester, cheering me on and discussing the project whenever needed. Finally, I would like to thank my dad for proofreading and valuable advice.

The focus of this project has been to evaluate the system identification process, and find more effective solutions where it was possible within the system frames. This includes implementing a mathematical model and evaluating the results, both of the new system and of the parameter estimation. In the identification process, different tools have been used. This includes: hardware and a LabView program form Inventas, MATLAB, Cybernetica's program ModelFit, Eline Malde's masters thesis, where ModelFit was used for offline and online parameter estimation, and a mathematical model developed by Øyvind Stavdahl and Maria V. Ottermo.

Throughout the project, invaluable input has been provided. In order to clarify which parts that are entirely my work and where necessary help has been provided, a summary will be given here. The system in its entirety was thoroughly discussed on a guidance meeting with Øyvind Stavdahl and Maria V. Ottermo, where a draft of the figure included in this report was developed. The claims for the identification process was developed and discussed with Maria V. Ottermo. Øyvind Stavdahl was greatly involved in the discussion of choosing the software platforms. The initial values used in the parameter estimation are developed from research by Maria V. Ottermo and Øyvind Stavdahl. Finally, input on which parameters to estimate has been provided by Maria V. Ottermo. Kasper Linnestad from Cybernetica has contributed with quick replies through e-mail whenever questions about Cybernetica's tools have arisen.

As it was crucial to understand the system and needs before implementing the identification process, a lot of time has been spent on the design phase of the project.

Contents

List of Tables

List of Figures

Nomenclature

ATP	Adenosine-5'-Triphosphate. page 9
CO_2	Carbon Dioxide. page 2, page 4, page 9, page 11–page 13, page 30, page 37, page 45, page 63, page 69–page 72, page 81
COPD	Chronic Obstructive Pulmonary Disease. page 1, page 2, page 5, page 31, page 81
DFD	Data Flow Diagram. page 19, page 35, page 37
$ETCO_2$	End Tidal Carbon Dioxide. page 61, page 63–page 69, page 71, page 72, page 74, page 77
FAT	Factory Acceptance Test. page 19
IDP	Identification Process. page xi, page 2–page 4, page 7, page 8, page 31, page 33, page 35–page 37, page 39, page 42, page 44, page 45, page 49, page 52, page 59, page 78, page 79, page 81, page 82
LTOT	Long Term Oxygen Therapy. page 2, page 38
MPC	Model Predictive Control. page 2–page 5, page 7, page 22, page 26–page 28, page 39, page 46, page 56, page 71, page 72, page 75, page 77, page 80, page 81
MT_{CO_2}	Carbon Dioxide Production. page 63–page 66, page 71, page 72, page 75
MT_{O_2}	Oxygen Consumption. page 63, page 64, page 68, page 71, page 72, page 75

NLP	Non-Linear Programming Problem. page 22
NMPC	Nonlinear Model Predictive Control. page 28
O_2	Oxygen. page 2, page 4, page 9, page 12, page 13, page 30, page 44, page 45, page 62, page 63, page 65, page 67, page 68, page 81
SA/SD	Structured Analysis and Structured Design. page 18, page 19
SaO_2	Oxygen Saturation. page 61, page 64–page 70, page 72, page 74, page 77
SQP	Sequential Quadratic Programming. page 23
STD	State Transition Diagram. page 20
UML	Unified Modeling Language. page 18, page 20
V_{T,CO_2}	Effective Volume of Carbon Dioxide in the Tissue Compartment. page 71, page 72, page 75
V_{T,O_2}	Effective Volume of Oxygen in the Tissue Compartment. page 71–page 73, page 75

Chapter 1

Introduction

In this chapter the motivation for this project is presented with a brief introduction to COPD and today's solution for oxygen therapy. Then, an introduction to the system is entirety, the setup for a trial and previous work is presented. Finally, the problem description and objective, along with the contributions of this project and an outline of the report is presented.

1.1 Background and Motivation

In this section an introduction to COPD and the OxyAid project will be given. As this was also presented in [10], Section 1.1.1 is largely based on the this. In addition, an introduction to the system as a whole will be given in Section 1.1.2 with both a textual and a graphical introduction.

1.1.1 COPD and the OxyAid Project

According to the Global Initiative for Chronic Obstructive Lung Disease's report from 2019, COPD is the fourth leading cause of death in the world[7]. However, already by 2030 COPD is projected to be the third leading cause of death. According to the World Health Organization (WHO) as much as 65 million people have moderate to severe COPD [19]. Globally, this corresponds to as much as 5% of all deaths, and the number of people with COPD is supposed to rise with 30% over the next 10 years. A disease with such global impact clearly comes with a great economic cost for the society. The yearly medical costs alone associated with COPD was 276 billion NOK (BNOK) in 2010, with an estimated increase to 421BNOK by 2020 [22].

COPD is an umbrella term for chronic progressive lung diseases which are characterized by breathing problems and obstructive airways [8]. Hence, the term covers multiple different pulmonary diseases. The three main diseases are emphysema, where the alveoli are damaged, chronic bronchitis where there is an inflammation of the breathing tubes, and refractory asthma; a disease where the bronchial airways tighten up and swell. The main reason for developing COPD in developed countries is smoking. However, other factors such as air pollution, pollution in the work environment and genetics may also lead to COPD [23].

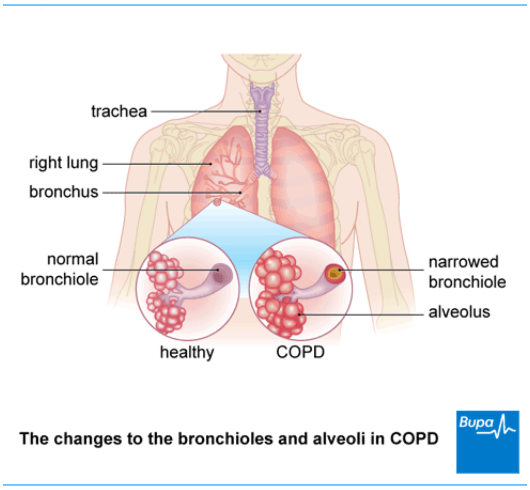


Figure 1.1: COPD's effect on bronchioles and alveoli, from [3]

Today long term oxygen therapy (LTOT) is offered for patients with COPD, and mainly for patients living at home. In LTOT the patient gets the oxygen through a tube in the nose which is connected to an oxygen tank. In this solution the patient must regulate the dosage of oxygen by themselves only based on the level of breathlessness they are experiencing, and the recommendations prescribed by their doctor. This represents a serious danger as the patient may forget to turn off the oxygen when it is no longer needed, or turn the oxygen level too high. This can lead to a very high dosage of O_2 in the blood, which again leads to the rising levels of CO_2 in the blood which this may be fatal [22]. Many patients are therefore very hesitant about turning up the oxygen even if needed as they are scared. Most patients have a tendency to have too low O_2 levels which can effect the quality of life. The tank may also be too far away for the patient to reach if the oxygen need is imminent. In other words, LTOT is not an optimal solution.

The OxyAid project has proposed a more optimal solution where the O_2 and CO_2 levels of the patient would be regularly measured, and through a control system calculate how much O_2 which is needed based on the patient's need and activity level. Achieving this is the aim of the OxyAid project, a project at the NTNU [22]. This report is written as a contribution to the project.

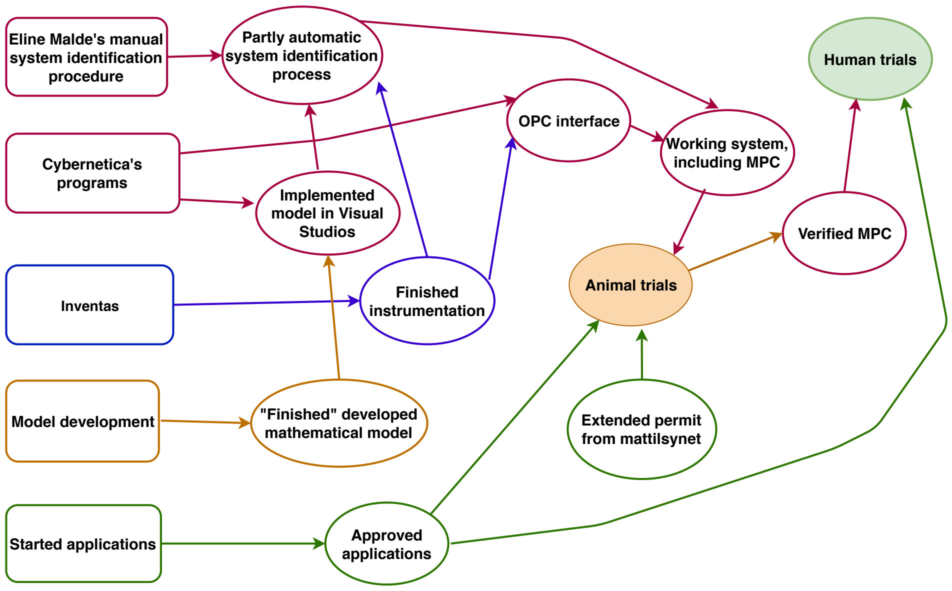
In order to regulate the O_2 level and ensure that the CO_2 level never reaches a dangerous level, a control system is needed. The project plans on utilizing model predictive control (MPC) in order to regulate the control system. In order to achieve good estimates of the needed oxygen it is important with a mathematical model which represents reality well, and parameters tuned to fit each patient. By having a well defined identification process (IDP), precious time can be spared during trials and the performance of the controller can be optimized.

1.1.2 The System in its Entirety

The system in its entirety consists of several subsystems, with the identification process (IDP) being one of them. In this section, an introduction to the system will be given, with a short description of each subsystem along with a graphical display. In addition to an overview of the complete system, the two different cases of pig trial and human trial will be presented.

Description of the System

The description of the system is based on a discussion with Øyvind Stavdahl and Maria V. Ottermo. The description of the system is true as to September 2019, and as stages of the system are completed, the figure will, naturally, change.



Eline Malde's manual system identification procedure
Partly automatic system identification process
Human trials
OPC interface
Cybernetica's programs
Working system, including MPC
Implemented model in Visual Studios
Verified MPC
Finished instrumentation
Animal trials
Inventas
Model development
"Finished" developed mathematical model
Extended permit from mattilsynet
Started applications
Approved applications

Figure 1.2: The OxyAid system in its entirety

In Figure 1.2 the pink part of the system is the software, the blue is hardware, the yellow is model development and the green is applications. Animal trials is marked with orange and human trials is marked green as these are the clear goals for the project.

In order to achieve the system identification process (IDP) and the model predictive control (MPC), a mathematical model must be developed. This model is used both to estimate optimal parameters and later optimal reference values.

By using Cybernetica's programs, which will be introduces later in the report, the mathematical model can be implemented in Visual Studios and used in the system identification process. The finished instrumentation is used to log data, which again is used by the system identification process to estimate parameters. In addition, the instrumentation is also used in the OPC interface.

The OPC interface is used to transfer the measured values needed in the MPC from the hardware (finished instrumentation) to the MPC (Cybernetica's program CENIT), and is thus needed in order to have a working system.

A number of applications must be approved in order to get permission to conduct the pig and human trials. In addition, an extended permit from Mattilsynet is needed to conduct the pig trials, as the current one expires in October 2019. The pig trials will be used to verify the MPC, which is necessary in order to get approval to conduct the human trials.

Pig/Human Trial

In addition to an introduction to the system in its entirety, an introduction to the setup of a trial is included as it is important for a better understanding of the environment of the system identification process. The trial setup will be further discussed in Chapter 3.

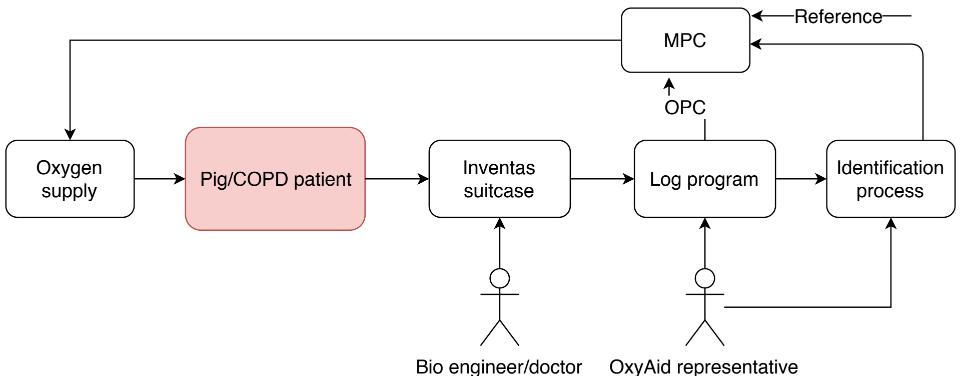


Figure 1.3: The setup of a pig/human trial

Figure 1.3 illustrates the setup of a pig/human trial. The pig/patient will be connected to a suitcase of hardware from Inventas, a company which offers design services and has been hired by the OxyAid project, which measures the necessary data of the subject. This data is sent to a program, also made by Inventas, that can log the data to a file or send it via OPC. The program also displays the O_2 and CO_2 levels at all times, which is crucial as a bio engineer or clinician supervise the levels and interrupts the pig or human trial, respectively, if the levels are too low/high. In addition, the bio engineer also ensures that the pig is in "equilibrium" before starting the IDP in the pig trials.

The logged data is used in the parameter estimation, and the parameters are sent to the MPC before the system is run in closed loop. In closed loop, the measured values are sent to the MPC from Inventas' program via OPC. The handling of the computer programs is done by an OxyAid representative.

The Identification Process - Definition

In this report the term "identification process" or "IDP" for short will be used a lot. It is therefore important to understand its meaning.

Each individual person is different. It differs how much lung capacity a person has, what stage the COPD is at, how much oxygen is consumed, how much carbon dioxide is produced, etc. This means that even though a mathematical model describing the pulmonary system has been developed, it needs to be fitted to each person's (or pig's) individual differences. If the oxygen level is controlled without any adjustments it is likely that the user will not receive the optimal amount of oxygen. Hence, by identifying the values of an individual's respiratory dynamics, it is more likely that optimal control will be achieved. It is therefore an important process in order for a patient to get as much value from the oxygen control system as possible.

1.1.3 Previous Work

In Malde [14] a different mathematical model was implemented and parameter estimation in ModelFit was performed by using logged data from two previous pig trials. In other words, it was completely manual, and not designed to be used in a trial.

The parameter estimation is important as this form the basis for a well controlled system through MPC. This will be further explained throughout the report. Malde's manual process is the basis used in this project for developing the partially automatic system identification process, and is referred to as the "previous procedure" throughout the report.

1.2 Problem Description

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

Masteroppgave

Studentens navn:	Stina Holen Friisø
Fag:	Teknisk kybernetikk
Tittel (norsk):	Automatisering av systemidentifikasjon under dyre- og menneskeforsøk
Tittel (English):	Automation of system identification during animal and human trials

Beskrivelse:

Menneskekroppens celler forbruker oksygen (O_2) og danner karbondioksid (CO_2), og er derfor avhengig av stadig å få tilført O_2 og få transportert bort CO_2 for å kunne opprettholde livet.

Hos pasienter med nedsatt lungefunksjon er det vanlig å få ekstra oksygenrik pustegass for å avhjelpe respirasjonen. I dag stilles oksygenmengden inn manuelt etter legens vurdering og pasientens opplevde behov. Imidlertid kan for mye oksygen gi pasienten redusert behov for å trekke pusten og dermed opphopning av CO_2. En ønsker derfor å utvikle et system for automatisk regulering av tilført oksygen som kan øke pasientens livskvalitet og samtidig hjelpe leger til bedre å vurdere pasientens tilstand.

Denne oppgaven inngår som en del av et større prosjekt som sikter mot å ta fram et slikt system. Systemet vil bli bygget rundt en modellbasert regulator (MPC), og studentens arbeid vil primært være konsentrert om å tilrettelegge for effektiv identifikasjon av respirasjonsdynamikk og integrering av resultatet i regulatoren, slik at systemet kan testes ut i praksis i en klinisk situasjon. Utgangspunktet for arbeidet er tidligere utførte studentarbeider (primært Malde, 2019), foreliggende tilstandsrommodell og andre prosjektresultater.

Oppgaven omfatter følgende momenter:

1. Gi en overordnet beskrivelse av de praktiske prosedyrene som vil inngå i kommende dyreforsøk/kliniske forsøk, der identifisering av individuell respirasjonsdynamikk inngår som én delprosess. Denne delprosessen omtales heretter som identifikasjonsprosessen (IDP)
2. Sett opp en funksjonsspesifikasjon for IDP, og gjør et begrunnet valg av overordnet prosedyre, programvare, programmeringsplattform osv.
3. Implementer IDP i henhold til forrige punkt. I tillegg til programkode skal implementasjonen også inneholde nøyaktig beskrivelse av eventuelle manuelle arbeidstrinn.
4. Evaluer resultatet i kontekst av det øvrige systemet så langt det er mulig innenfor de gitte rammene.

Veileder(e): Øyvind Stavdahl, Maria Vatshaug Ottermo

Trondheim, 05.08. 2019

Øyvind Stavdahl
Faglærer

1.3 Problem Objective

The overall goal for this project is to design and implement an identification process, simply referred to as IDP throughout the report, that is to be used in clinical trials. This entails logging of data, formatting of data, parameter estimation and transferring the estimated values to the model predictive control (MPC. In order to achieve this, the following steps will be pursued:

1. In order to evaluate the result of the design and parameter estimation, it is important to understand the different available tools for the process design. In addition, a good understanding of system identification is needed.
2. In order to evaluate the IDP in context of the system as a whole it is necessary to understand the system in its entirety. Thus, a graphical and textual description of this is valuable.
3. The IDP will be designed at both a high and low level using existing modeling tools. The high level design will result in a function specification.
4. The designed system will be implemented according to the low level design, and manual steps will be described.
5. System tests will be designed and performed.
6. The results of the system tests will be presented and evaluated. This includes the results from the offline and online parameter estimation. In addition, the system will be evaluated with regards to the function specification.

1.4 Contributions

In this project, the author has made the following contributions:

- a presentation of the entire OxyAid system with sub processes,
- high and low level design of the identification process,
- implementation of Ottermo and Stavdahl's two compartment model, and evaluation of the results of simulations and parameter estimation with said model,
- implementation of the designed identification process,
- evaluation of the identification process with regards to defined function specifications and system tests.

1.5 Outline

In the following, a brief overview of the content of each chapter is presented.

Theory

The theory chapter presents the knowledge necessary to understand the system design and the results and discussions of the parameter estimation. This includes a presentation of the implemented mathematical model, the V-model and other modeling tools used a guidelines for the system design, theory about online and offline parameter estimation and a brief introduction to model predictive control.

High Level Design of the IDP

This chapter presents the high level design of the identification process. This includes textual and graphical descriptions of the system's interactions with the environment and definitions of the modules. In addition, the claims that must be satisfied are presented. The chapter concludes with a function specification.

Low Level Design of the IDP

In this chapter, low level design of the identification process is presented. The functions of each module are defined along with unit and integration tests. In addition, a discussion concerning the software options is included.

Implementation

The implementation chapter provides a presentation of the implemented system, including the implementation of the mathematical model in Visual Studios. Also, the chapter presents precautions before running the system, and a detailed description of the manual steps that must be performed to run the identification process.

Results and Discussion

In this chapter, the results of the identification process are presented. This entails both the presentation and discussion of the results of both the offline and online parameter estimation, and the results of the system, integration and unit tests of the process. In addition, an evaluation of the system in regards to the system specification is included.

Conclusion and Future Work

Finally, in this chapter the conclusion is presented. The chapter concludes with a section on recommended improvements and future work.

Chapter 2

Theory

In this chapter background theory for the project will be presented. First, an introduction to the pulmonary system will be given, as this is important for understanding the mathematical models. As this section also was included in the project report, this section will be based on the work done in [10]. Stavdahl and Ottermo's two compartment mathematical model will be introduced as this is used for the parameter estimation. Then, an introduction to the V-model, structured analysis and design (SA/SD) is presented. Finally, an introduction to system identification along with model predictive control will be presented.

2.1 Introduction to the Pulmonary System

Breathing is absolutely essential for life, not only for humans, but for animals and even plants as well. We breathe in oxygen (O_2), which is essential for the cells in the body to burn sugars and fatty acids to produce adenosine-5'-triphosphate (ATP) [6]. This provides the energy needed for the body to function. We breathe out carbon dioxide (CO_2), which is, along with water, the waste product of the energy production. This process of breathing in oxygen and breathing out carbon dioxide and vapor is repeated as many times as 15-20 times for adults every minute [11].

The pulmonary system can be divided into three parts; the respiratory tracts, the blood and the cells [6]. The respiratory tracts consist of everything from the nasal cavity to the lungs and is where the gas exchange takes place. The oxygen is breathed in through the mouth and nose, and then precedes to the lungs through the trachea and the two bronchi; one for each lung. The air then enters the alveoli, and here the exchange of oxygen and carbon dioxide to and from the blood occurs [27].

In the blood, the red blood cells binds the oxygen. The hemoglobin is transported throughout the body via the bloodstream. The oxygen rich blood from the lung capillaries enters the pulmonary veins, and is pumped into the aorta by the heart. Here the oxygen will travel to the different parts of the body, and the red blood cells will release the oxygen where needed and bind carbon dioxide. The blood, which is now rich on carbon dioxide instead, is then pumped back to the heart and into the pulmonary artery and to the lungs.

The carbon dioxide is then released and diffused to the airways and the red blood cells once again binds oxygen [6]. In the cells, the oxygen, as previously mentioned, is used to create ATP needed for the body to function.

2.2 The Mathematical Model

Here, one of the mathematical models developed by the OxyAid project will be presented. This is important in order to discuss the results of the parameter estimation, as Stavdahl and Ottermo's two compartment model is the one that has been implemented. The model is based on Batzel, and further reading can be found in [2].

2.2.1 Ottermo and Stavdahl's Two Compartment Model

After several different models were developed and tried, the following model was proposed and is now the one in use. The model is developed by Maria V. Ottermo and Øyvind Stavdahl, and the presentation of the model is a summary based on [26].

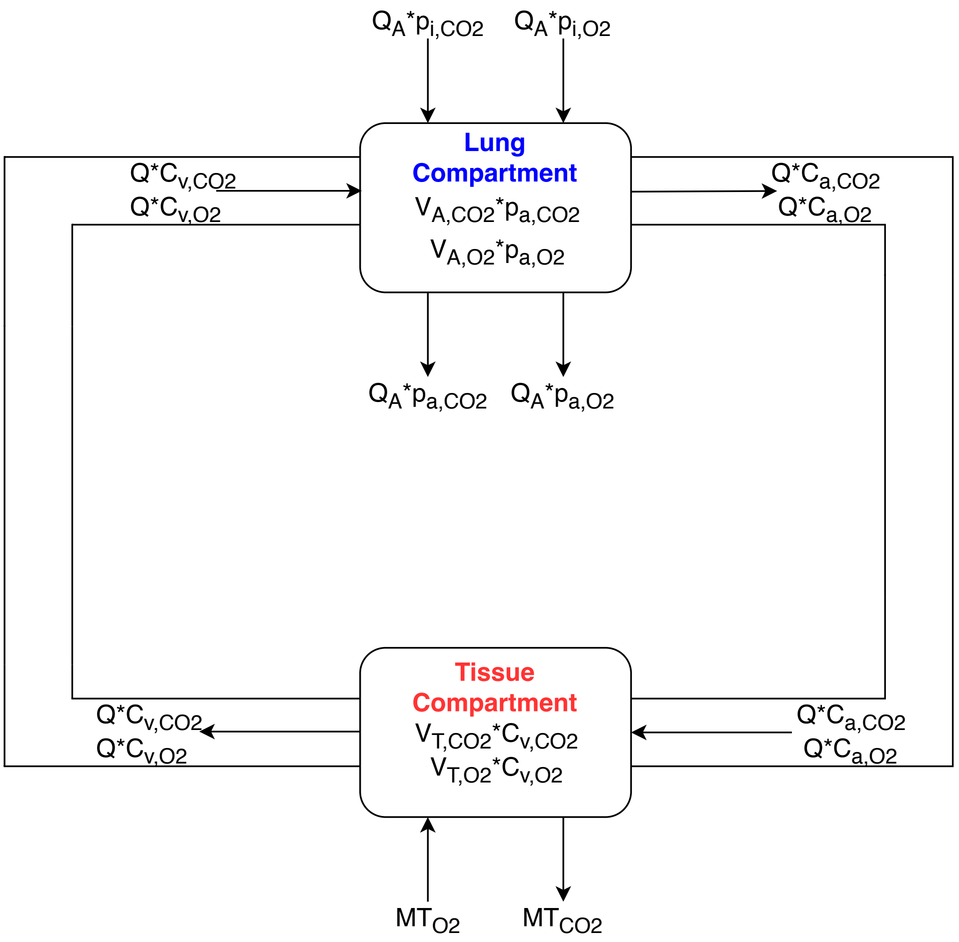


Figure 2.1: Ottermo and Stavdahl's two compartment model

The following assumptions were made:

- The body temperature, humidity and ambient pressure are constant.
- The arterial and alveolar CO_2 are assumed equal, though a small arterial gradient is included. It is also assumed that tissue and venous gases are in equilibrium.
- The alveoli and pulmonary capillary spaces are two well-mixed interfaced compartments with detailed structure and flows omitted.
- Gas exchange happens by diffusion. This is a reasonable assumption as the alveolar space represents a very large surface area, and the blood gas barrier between the alveoli and capillaries is extremely thin. Thus, allowing for efficient diffusion of gases in both directions.

- The left and right lung behave exactly the same, and the air flows are the same in both lungs.
- The intercardiac shunting, which is about 2% is negligible.
- The partial pressure of CO_2 in inhaled air is negligible.

The model is also based on some fundamental laws:

Dalton's law states that the total pressure of a mixed gas is the sum of all the individual (partial) pressures of the independent gas components. Mathematically, this is expressed as:

$$p_i = f_i p_{tot} \tag{2.1}$$

where f_i is the molar fraction of the i'th gas, p_i is the partial pressure of each gas, and $p_{tot} = \sum_i p_i$ is the total pressure.

For fixed pressure and temperature, Avogadro's law states that the molar and volume fractions of a gas constituent are the same.

In addition, a conversion between two of the most common types of conditions based on the ideal gas law has been utilized:

$$V_{BTPS} = \frac{863}{P_{amb} - 47} V_{STPD} \tag{2.2}$$

where BTPS is for when body temperature and ambient pressure are water saturated, with $T = 37°C = 310°K$, $P = P_{amb}$ is the ambient pressure and $p_{H_2O} = 47mmHg$ and STPD is the standard temperature and pressure when dry, with $T = 0°C = 273°K$, $P_0 = 760mmHg$ os at the sea level barometric pressure and $p_{H_2O} = 0mmHg$.

For both compartments the equations are developed by using mass balance; change in compartment j = (sum of all transfers into compartment j) - (sum of all transfers out of compartment j) + (creation within compartment j) - (destruction within compartment j).

Thus, the equations for the compartments are as follows:

Lung Compartment

The mass balance equations for the rate of change of the lung CO_2 and O_2 volumes are:

$$V_{A,CO_2} \frac{dp_{a,CO_2}}{dt} = 863Q(C_{v,CO_2} - C_{a,CO_2}) + Q_A(p_{i,CO_2} - p_{a,CO_2}) \tag{2.3}$$

$$V_{A,O_2} \frac{dp_{a,O_2}}{dt} = 863Q(C_{v,O_2} - C_{a,O_2}) + Q_A(p_{i,O_2} - p_{a,O_2}) \tag{2.4}$$

where V_{A,CO_2} and V_{A,O_2} denote the effective CO_2 and O_2 volumes in the lung compartment, respectively. p_{a,CO_2} and p_{a,O_2} are the partial pressures of CO_2 and O_2 in the arterial blood, Q is the blood flow and can be written as:

$$Q = f_H \cdot SV \tag{2.5}$$

where f_H is the heart rate/pulse, and SV is the stroke volume.

C_{v,CO_2} and C_{v,O_2} are the CO$_2$ and O$_2$ concentration in venous blood, and C_{a,CO_2} and C_{a,O_2} are the concentrations of CO$_2$ and O$_2$ in the arterial blood. Q_A is the alveolar ventilation, with the following mathematical expression:

$$Q_A = f_T \cdot V_{eff} \tag{2.6}$$

where f_T is the breathing frequency and V_{eff} is the effective volume.

p_{i,CO_2} is the partial pressure of CO$_2$ in inhaled air. Though, from the assumptions this is so small that it's negligable. It has therefore not been taken into consideration when implementing the model. p_{i,O_2} is the partial pressure of oxygen in inhaled air.

The equations for the lung compartment are found by using equation 2.1 and 2.2 for substitution for the fractional concentrations, f_{A,CO_2}, f_{A,O_2}, f_{i,CO_2} and f_{i,O_2}.

The first term on the right-hand side of the equations accounts for the net change in CO$_2$ and O$_2$ concentrations, respectively, due to gas transport by blood flow. The second term on the right-hand side accounts for the net change in CO$_2$ and O$_2$ concentration, respectively, due to alveolar ventilation. Together, the sum of the two terms gives the total change in alveolar CO$_2$ and O$_2$ concentrations, respectively.

Tissue Compartment

The mass balance equations for the rate of change of the tissue CO$_2$ and O$_2$ volumes are:

$$V_{T,CO_2}\frac{dC_{v,CO_2}}{dt} = MT_{CO_2} + Q(C_{a,CO_2} - C_{v,CO_2}) \tag{2.7}$$

$$V_{T,O_2}\frac{dC_{v,O_2}}{dt} = -MT_{O_2} + Q(C_{a,O_2} - C_{v,O_2}) \tag{2.8}$$

where V_{T,CO_2} and V_{T,O_2} are the effective volumes of CO$_2$ and O$_2$, respectively, in the tissue compartment. MT_{CO_2} is the metabolic production of CO$_2$ in the tissue, and MT_{O_2} is the metabolic consumption of O$_2$ in the tissue.

The left hand side of these equations account for the change in CO$_2$ and O$_2$ volumes in the tissue compartment, respectively. The right hand sides represent the net volume changes in terms of the metabolic production of CO$_2$ and consumption of O$_2$, and the removal of CO$_2$ and O$_2$ described by the net change in the arterial and venous blood CO$_2$ and O$_2$ concentrations for the blood flowing through the tissue. The net volume change is found by multiplying concentration by the tissue blood flow, Q.

As both C_{a,CO_2} and C_{a,O_2} are time varying with time delay, these must be expressed by mathematical expressions. For C_{a,CO_2}, the equation is:

$$C_{a,CO_2} = K_{CO2}p_{a,CO_2} + k_{CO_2} \tag{2.9}$$

where $K_{CO_2} = 0,0051[1/mmHg]$ and $k_{CO_2} = 0,224[]$, and are both constants.

For C_{a,O_2} both a linear and non-linear relationships are described in [26]. In order to describe reality as best as possible, a non-linear case is used:

$$C_{a,O_2} = S_{a,O_2}C_{Hb}b_{Hb} + s_{,O_2}p_{a,O_2} \tag{2.10}$$

where S_{a,O_2} is oxygen dissociation, $C_{Hb} = 0,15$ is the constant concentration of haemoglobin in the whole blood for a healthy person, $b_{Hb} = 1,34$ is a constant and represents the oxygen-binding capacity of haemoglobin and finally $s_{,O_2} = 0,000031[L/LmmHg]$ is constant and represents the solubility coefficient for oxygen.

Finally, an expression for S_{a,O_2} is needed:

$$S_{a,O_2} = \frac{1}{\frac{23400}{p_{a,O_2}^3 + 150 p_{a,O_2}} + 1} \tag{2.11}$$

This is known as the oxygen dissociation formula, which relates p_{a,O_2} and S_{a,O_2} (O_2-saturation).

In Table 2.1 a summary of the symbols in Stavdahl and Ottermo's two compartment model is presented.

Symbol	**Description**
C_{a,CO_2}	Carbon dioxide concentration in arterial blood
C_{a,O_2}	Oxygen concentration in arterial blood
C_{v,CO_2}	Carbon dioxide in venous blood
C_{v,O_2}	Oxygen in venous blood
Q	Blood flow
Q_A	Alveolar ventilation
f_T	Breathing frequency
f_H	Heart rate
P_{amb}	Ambient pressure
MT_{CO_2}	Metabolic production of carbon dioxide in tissue
MT_{O_2}	Metabolic consumption of oxygen in tissue
p_{a,CO_2}	Partial pressure of carbon dioxide in arterial blood
p_{a,O_2}	Partial pressure of oxygen in arterial blood
p_{i,CO_2}	Partial pressure of carbon dioxide in inhaled air
p_{i,O_2}	Partial pressure of oxygen in inhaled air
T	Temperature
V_{A,CO_2}	Effective volume of carbon dioxide in the lung compartment
V_{A,O_2}	Effective volume of oxygen in the lung compartment
V_{T,CO_2}	Effective volume of carbon dioxide in the tissue compartment
V_{T,O_2}	Effective volume of carbon dioxide in the tissue compartment
$s_{,O_2}$	Solubility coefficient of oxygen
K_{CO_2}	constant
k_{CO_2}	constant
C_{Hb}	constant, concentration of haemoglobin in whole blood for healthy person
b_{Hb}	constant, oxygen-binding capacity of haemoglobin
SV	Stroke volume
V_{eff}	Effective volume

Table 2.1: Summary of symbols in Stavdahl and Ottermo's compartment model

2.2.2 State-Space Form

In order to use ModelFit, the model must be represented on the state-space form and implemented in Visual Studios. Therefore, the model has been rewritten to state-space form:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2.12a}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u} \tag{2.12b}$$

Where $\mathbf{x}$ are the states, $\mathbf{A}$ is the system matrix, $\mathbf{B}$ is the input matrix, u is the input, $\mathbf{C}$ is the output matrix, $\mathbf{D}$ is the feedthrough matrix and y is the output [4].

Defining the states:

$$\mathbf{x} = \begin{bmatrix} p_{a,CO_2} \\ p_{a,O_2} \\ C_{v,CO_2} \\ C_{v,O_2} \end{bmatrix} \tag{2.13}$$

Defining input:

$$\mathbf{u} = \begin{bmatrix} p_{i,O_2} \\ f_T \\ f_H \end{bmatrix} \tag{2.14}$$

Defining output:

$$\mathbf{y} = \begin{bmatrix} ET_{CO_2} \\ S_{a,O_2} \end{bmatrix} \tag{2.15}$$

In addition, matrices for parameters and constants are defined:

Parameters:

$$\mathbf{p} = \begin{bmatrix} MT_{CO_2} \\ MT_{O_2} \\ V_{T,CO_2} \\ V_{T,O_2} \\ V_{A,CO_2} \\ V_{A,O_2} \\ SV \\ V_{eff} \end{bmatrix} \tag{2.16}$$

Constants:

$$\mathbf{c} = \begin{bmatrix} s_{,O_2} \\ K_{CO_2} \\ k_{CO_2} \\ C_{Hb} \\ b_{Hb} \end{bmatrix} \tag{2.17}$$

This yields the following state equations:

$$\dot{x_1} = \frac{1}{p_5}(863Q(x_3 - c_2x_1 - c_3) + Q_A(-x_1)) \tag{2.18a}$$

$$\dot{x_2} = \frac{1}{p_6}(863Q(x_4 - y_2c_4c_5 - c_1x_2) + Q_A(u_1 - x_2) \tag{2.18b}$$

$$\dot{x_3} = \frac{1}{p_3}(p_1 + Q(c_3 + c_2x_1 - x_3)) \tag{2.18c}$$

$$\dot{x_4} = \frac{1}{p_4}(-p_2 + Q(y_2c_4c_5 + c_1x_2 - x_4)) \tag{2.18d}$$

where $Q = u_3p_7$ and $Q_A = u_2p_8$.
With output equations:

$$y_1 = x_1 \tag{2.19a}$$

$$y_2 = \frac{1}{\frac{23400}{x_1^3+150x_1} + 1} \tag{2.19b}$$

Thus, it is clear that the model has 4 states, 8 parameters, 5 constants, 3 inputs and 2 outputs.

2.3 System Modeling Tools

As the identification process designed in this project is to be used in practical trials, it is important that it works according to the system claims and produces correct values. If the parameter estimation gives wrong values or uses incorrect data it can lead to the patient getting either too much oxygen, which in turn can lead to an accumulation of carbon dioxide which is very dangerous, or too little oxygen. Therefore, it is very important with a thorough design procedure. Unified Modeling Language (UML), The V model and structured analysis (SA) and structured design (SD) are different tools for a good design process. They each contain a lot of different tools, and the tools used in this project will be presented in this section.

2.3.1 The V Model

The V model is a model which describes a design procedure for software systems. It is divided into multiple steps in order to ensure software quality [15]. These steps include requirements analysis, specifications, architectural design, detail design, coding, unit testing, integration testing, system testing and finally acceptance testing. These can be arranged in a V, as shown in Figure 2.2, where the developer starts at the requirement analysis, moves downwards to coding, before stepping up again all the way to acceptance testing.

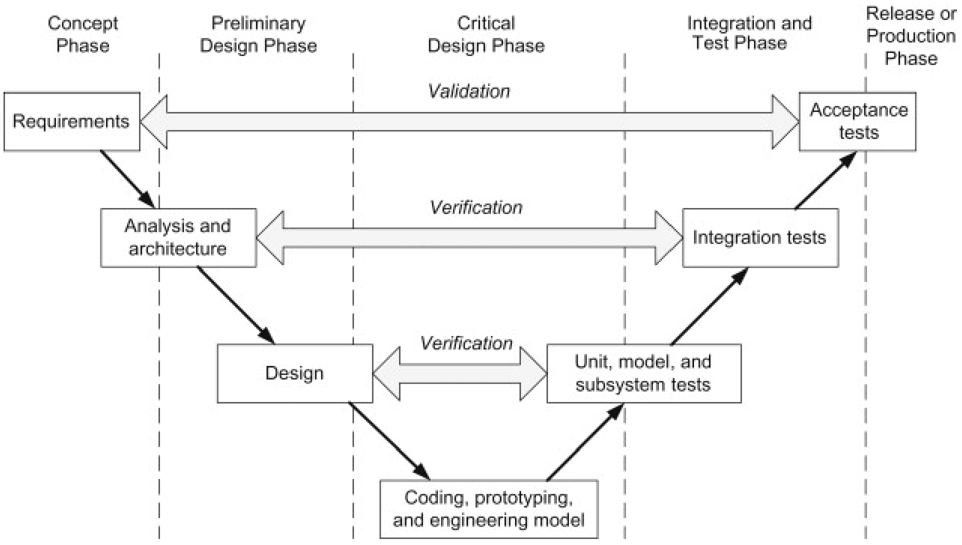

Concept Phase
Preliminary Design Phase
Critical Design Phase
Integration and Test Phase
Release or Production Phase
Requirements
Validation
Acceptance tests
Analysis and architecture
Verification
Integration tests
Design
Verification
Unit, model, and subsystem tests
Coding, prototyping, and engineering model

Figure 2.2: The V-model, from [9]

Requirement Analysis

In the requirement analysis, the user requirements are established. These requirements are at a high level, meaning that they do not include specific details about the system, and does not include any hard requirements of the code. For instance, for a gas pump system, the user requirements could be:

- The gas pump must the able to deliver unlimited gas to cars.
- The gas pump must be able to give instructions to the user on how to use the pump.
- The system must be able to process payment with both debit and credit cards.
- etc.

Specifications

In the next step, the system specifications are established. This is a very important and extensive step, which should end in a function specification. In this step it is useful to use tools such as SA/SD or unified modeling language (UML), which are described in Section 2.3.2. A function specification includes the requirements a system should fulfill, and is normally a list of hard and loose demands listed as "the system shall..." and "the system should..." respectively. In this project, the development of specifications and high level design overlap.

Architectural Design

Architectural design is the step where the high level design of the system is designed. For instance, for a larger software system, the result of this step should be the names of the different modules, how they communicate and what each module is supposed to do. In this setting, a module is a part of the system which has one defined task and contains multiple functions. For instance, for a gas pump the handling of payment may be one module. It is in the development of the architectural and detail design SA/SD is a useful tool.

Detail Design

The detail design entails the design of each module, i.e. which functions that are included in each module, dependency issues, local and global variables, etc. This also includes specifying the purpose of each function.

Coding

Finally, when all the design steps are completed, the developer can start the actual implementation of the system. In this step a first draft of the system code is completed.

Unit Testing

The final steps of the V model all includes different forms of system testing. The first step is Unit Testing. Here, smaller tests which only tests smaller units, or parts, of the system are used. This is to ensure that there are no mistakes in the functions and smaller units of the system before moving to a higher level. The tests are developed during the detail design step. The units tests are used to eliminate smaller bugs at an early stage.

Integration Testing

After establishing that the unit tests are passed, the developer can move on to the next step of testing; the integration testing. The integration testing tests how the different units work together. In other words, the previous step tested the units alone, while this step tests how the units work together. The tests are developed during the architectural design step.

System Testing

In the system tests, the system is tested as a whole. These tests are developed during the design process phase of the V-model. The system tests determine whether the system has achieved the desired functionality.

Acceptance Testing

Finally, if the system has passed all the previous testing steps, it is ready for acceptance testing. The acceptance tests are often developed by the customer/people who ordered the system. For instance, for larger systems Factory Acceptance Tests (FAT) often used. Normally this is simply a list with the function specifications, which are agreed upon beforehand by the developers and customers, with a claim followed by an empty box which is ticked off if the claim is satisfied and finally a box for comments. This step is to secure that the customer is satisfied with the delivered system.

2.3.2 SA/SD

Structured Analysis and Design (SA/SD) is a tool used by developers to help understanding their system better. The aim of SA/SD is to improve quality and reduce risk of system failure. It is a process where the developer starts at a high level with textual description and context diagrams before delving deeper into the system [21].

SA/SD involves 2 phases:

1. The analysis phase (SA)
2. Design phase (SD)

In the analysis phase different textual descriptions, diagrams and lists are utilized in order to map what the system is going to do, and how it logically is going to do it [25]. This phase includes both mapping of the interface of the environment surrounding the system or process, and the actual behaviour of the system [25]. The different features will be discussed in further detail later in the subsection.

The design phase includes structure charts and pseudo code [21]. According to [25], the second phase is the "Structured Construction", which consists of the architecture model, the construction model and implementation. The goal in this phase is to construct the best possible solution of the problem by using the behavior model from the analysis phase [25].

Structured Analysis

In the structured analysis phase, the first step is to develop *system claims*. This is normally a detailed textual description that describes how the system is going to work. This provides a good starting point, but yields little overview of the problem [25].

In order to get a better overview of the system, the next step is to make an *environment model*. In this case the system's environment consists of the units the system can exchange information with. The model is made by first making a detailed list of all units in the system's environment, and then graphically representing this as a *context diagram*. A context diagram is a data flow diagram (DFD) at the highest abstraction level. In other words, the purpose of the context diagram is to graphically express how the data flows between the system and its environment. A DFD is composed of the following components:

- Terminator - symbolizes units in the system's environment with which the system communicate. It is a square box.

- Whole line with double arrow - continuous data stream, meaning an "analogue" data stream.
- Whole line with simple arrow - discrete data streams where the data is sent as impulses.
- Dotted line with double arrow - continuous event stream, meaning Boolean information that lasts over time.
- Dotted line with simple arrow - discrete event stream, meaning Boolean information that does not lasts over time.

After the context diagram is made, the next step in the analysis phase is to make an event list. An event list is a list of relevant external events. For each external event, name/description of the event, a named input stream that the system detects the event through, if the input stream is continuous or discrete and the data stream is noted. This is an efficient way of getting the essential logic of the system.

When the context diagram and event list are fully developed, it is time to start on the *behaviour model* of the system. This describes the behaviour of the system, including what subsystems the system consist of, how they interact, etc. This is developed from the environmental model, so it is very important that this is correct. The behaviour model can also be said to describe the architectural design step of the V-model.

In a context diagram, the system is simply represented as a "black box", i.e. the system is represented as a box with no detailed information of the content. In order to model the system's behaviour the context diagram will be "broken down" to more detailed data flow diagrams in a so-called top-down approach. This means that the system will be divided into subsystems, with each subsystem having an (as much as possible) isolated task [25]. For each of the subsystems a textual description should be included.

Finally, a state transition diagram (STD) is made. An STD specifies how much time each function will take to execute and data access triggered by events. In addition, it describes all of the states an object can have and when it changes state, including the triggers of the next state [25].

Structured Design

The structured design process describes the architectural and detailed design steps. This includes description of the different modules, how the modules communicate and which functions that need to be defined. As not much of the tools from structured design is used in this project, they will not be presented in this report. For further reading, see [25].

Sequence Diagram

In addition to structured design, a sequence diagram is used as a tool to represent the detailed design graphically. A sequence diagram is a tool from the Unified Modeling Language (UML), and describes how the modules interact arranged in a time sequence.

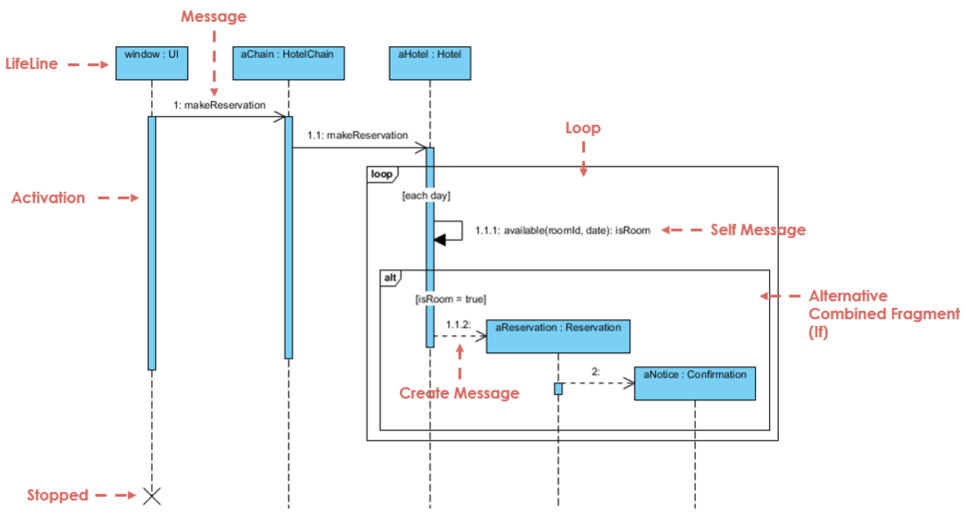


Figure 2.3: Example of a sequence diagram for hotel reservations, from [20]

Figure 2.3 shows an example of a sequence diagram for reserving a room at a hotel. The boxes at the top shows the different modules in the system, and the blue vertical boxes along the stippled lines are how long the different modules are active. The boxes symbolize loops (loop) or if-sentences (alt) in the code. Continuous arrows to the right represent messages, while dotted lines to the left are responses. Dotted lines such as the one marked "Create Message" represents the instantiation of (target) lifeline [20].

2.4 System Identification

System identification is a tool used for the optimizing of parameter values in the process model, so that the output resembles the output of the plant model as much as possible [5]. There are two ways to do this parameter estimation; online and offline parameter estimation, and both will be presented in the following subsections.

2.4.1 Optimization

As both offline parameter estimation and the Kalman filter is based on optimization, a brief introduction will be given in this section. This section is based on the presentation of optimization from [10].

Optimization is finding an optimal solution for a given problem. In mathematical optimization you have an objective function, which is the output that is to be either minimized or maximized. Many problems also include constraints, which limit the area of which a given algorithm can search for an optimal solution. This is an important tool in many different areas from decision science to the analysis of biological systems [17].

By defining:

- x as the list of variables/unknowns/parameters

- f as the objective function, that is a scalar function of x that is to be either maximized or minimized
- c_i are the constraint functions, which are scalar functions of x that define equations and inequalities that the unknown vector x must satisfy

Mathematically, an optimization problem can be formulated as:

$$\begin{array}{rlcll} \min\limits_{x \in R^n} & f(x) & & & \\ \text{s.t.} & c_i(x) & = & 0, & i \in Y, \\ & c_i(x) & \geq & 0, & i \in W. \end{array} \tag{2.20}$$

Where Y and W are sets of indices for equality and inequality constraints, respectively.

In biology, the use of mathematical optimization has enabled scientists to describe patterns, mechanisms and predict how organisms should be designed [1]. Most of the biological problems are described as a non-linear programming problem (NLP), that is an optimization problem where the objective is to find an optimal solution to a problem with either non-linear objective function and/or dynamic and algebraic constraints.

2.4.2 Offline Parameter Estimation

The aim of MPC is to make good predictions. Therefore, the model quality is essential in MPC. By performing parameter estimation, the distance between the model predictions and the experimental data is reduced [10]. This can be done by optimizing the parameter values, and is advantageous as it reduces the deviation between the model and the plant. As the parameter estimation is done without any tuning or interfering, that is without running the process, this is called offline parameter estimation.

In Cybernetica's ModelFit both the model and measurements are added, which enables comparison between the model and plant with the same input values. With enough variation in the dataset, it is possible to optimize more parameters than measurements. In other words, the program solves an optimization problem [5].

Sequential Quadratic Programming

In ModelFit, the parameters are estimated by solving the following optimization problem:

$$\min_{\theta} \quad J(\boldsymbol{\theta}) = \frac{1}{2}(\mathbf{y} - \mathbf{y_m})^T \mathbf{Q} (\mathbf{y} - \mathbf{y_m}) \tag{2.21a}$$

s.t.

$$\mathbf{x_k} = \mathbf{x_{k+1}} + \int_{t_{k-1}}^{t_k} \mathbf{f}(\mathbf{x}(\tau), \mathbf{u}(\tau); \boldsymbol{\theta}) d\tau \qquad \forall k \in \{1, \ldots, n\}, \tag{2.21b}$$

$$\mathbf{y_k} = \mathbf{g}(\mathbf{x_k}, \mathbf{u_k}; \boldsymbol{\theta}), \tag{2.21c}$$

$$\boldsymbol{\theta_{min}} \leq \boldsymbol{\theta} \leq \boldsymbol{\theta_{max}} \tag{2.21d}$$

where $\boldsymbol{\theta}$ is the parameters, $\mathbf{f}()$ is the process model, $\mathbf{g}()$ is the measurements, k is the sampling number and n is the number of samples. The vectors y and y_m are all model values and measurements for each sample, respectively. The $\mathbf{Q}$ matrix is the weight matrix. This means that it contains how much each variable is weighted. This is used to make the cost function dimensionless and can also be used to weigh different measurements and samples. The cost function, $J(\boldsymbol{\theta})$, is the squared deviation [14].

The optimizing problem can be solved by using the sequential quadratic programming (SQP) algorithm, as described by Nocedal and Wright [17] in Chapter 18 [14]. This is known as one of the most effective methods for non-linearly constrained optimization, as it generate steps by solving quadratic sub problems [17].

2.4.3 Online Parameter Estimation

Mathematical models will always be a simplification of reality. Hence, a model will not be able to regenerate the measured output values for the process perfectly. In other words, there will be a difference between the measured values and the values predicted by the model even if offline parameter estimation is utilized. This can, to a certain degree, be corrected by different recursive state estimation techniques where the states are updates based on the difference between the predicted and measured values. There are several different techniques that can be utilized, where Moving Horizon Estimator and a Kalman filter are two different options offered by ModelFit. [14].

In this thesis, a Kalman filter will be used, and for the online parameter estimation the Kalman filter will be tuned in open loop. In order to understand Kalman filters, a basic understanding of the problem a Kalman filter solves is essential.

Mathematical Description of the Problem

As mentioned, in a generalized system there are inputs, states and outputs. Mathematically, these are denoted as $\mathbf{u}$, $\mathbf{x}$ and $\mathbf{y}$ respectively. The goal is to estimate the states. However, there is generally a deviation between the measured and estimated state values of a system. In other words, there is an error that mathematically can be described as:

$$\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}} \tag{2.22}$$

where $\mathbf{e}$ is the deviation between the actual states $\mathbf{x}$ and the estimated states $\hat{\mathbf{x}}$.

The general system can be described as:

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \tag{2.23a}$$
$$\mathbf{y} = \mathbf{C}\mathbf{x} \tag{2.23b}$$

In order to estimate the states, an observer is introduced. This is generally an estimator that estimates the states based on the measured output from the plant. Mathematically, this is described as:

$$\dot{\hat{\mathbf{x}}} = \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} + \mathbf{K}(\mathbf{y} - \hat{\mathbf{y}}) \tag{2.24a}$$
$$\hat{\mathbf{y}} = \mathbf{C}\hat{\mathbf{x}} \tag{2.24b}$$

where $\hat{y}$ is the estimated outputs and $\mathbf{K}$ is the observer gain [13].

This can be represented graphically:

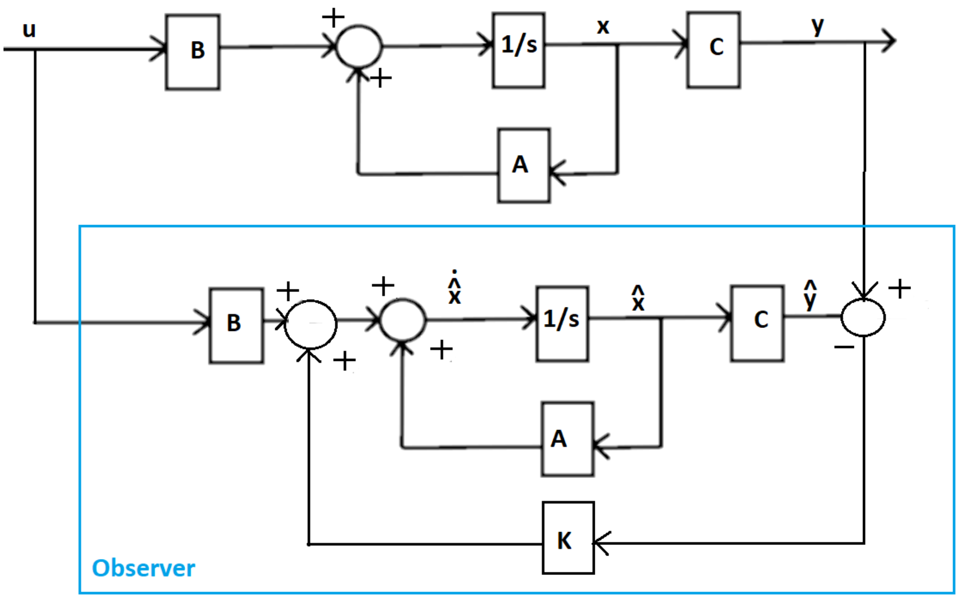


Figure 2.4: General feedback system with observer represented graphically

By subtracting the observer from the plant and substituting $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$, the error dynamics are obtained:

State expression:

$$\dot{\mathbf{x}} - \dot{\hat{\mathbf{x}}} = \mathbf{A}\mathbf{x} - \mathbf{A}\hat{\mathbf{x}} + \mathbf{B}\mathbf{u} - \mathbf{B}\mathbf{u} - \mathbf{K}(\mathbf{y} - \hat{\mathbf{y}}) \tag{2.25a}$$

$$\implies \dot{\mathbf{e}} = (\mathbf{A} - \mathbf{K}\mathbf{C}) \cdot \mathbf{e} \tag{2.25b}$$

$$\implies \mathbf{e}(t) = e^{(\mathbf{A}-\mathbf{K}\mathbf{C})t} \cdot \mathbf{e}(0) \tag{2.25c}$$

Output:

$$\mathbf{y} - \hat{\mathbf{y}} = \mathbf{C}(\mathbf{x} - \hat{\mathbf{x}}) \tag{2.25d}$$

$$\implies \mathbf{y} - \hat{\mathbf{y}} = \mathbf{C}\mathbf{e} \tag{2.25e}$$

As the equation in 2.25c is an exponential function that will approach zero if the exponent is less than zero, $\mathbf{K}$ is chosen such that $(\mathbf{A} - \mathbf{K}\mathbf{C}) < 0$. Thus, $\mathbf{e} \to 0$ when $t \to \infty$, and furthermore $\hat{\mathbf{x}} \to \mathbf{x}$ when $t \to \infty$. Since $\mathbf{y} = \mathbf{C}\mathbf{x}$, $\hat{\mathbf{y}} \to \mathbf{y}$ when $t \to \infty$ [14].

Kalman Filtering

One way of choosing K is by utilizing a Kalman filter. A Kalman filter is an optimal state estimator for stochastic processes, and is used on linear dynamical systems in state space form in its conventional form [14].

Mathematically this can be represented as:

$$\mathbf{x_k} = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k\mathbf{v}_k \tag{2.26a}$$

$$\mathbf{y}_k = \mathbf{C}\mathbf{x}_k + \mathbf{w}_k \tag{2.26b}$$

where $\mathbf{v}_k$ is the process noise and $\mathbf{w}_k$ is the measurement noise, and is assumed to be zero-mean Gaussian and normally distributed with covariance Q and R, respectively ($v \sim \mathcal{N}(0,\, Q)$ and $w \sim \mathcal{N}(0,\, R)$) [12].

A mathematical description of the Kalman filter is given by:

$$\begin{aligned}\hat{\mathbf{x}}_k &= \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}(\mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k)) \\ &= \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-\end{aligned} \tag{2.27}$$

where $\hat{\mathbf{x}}_k^-$ is the prediction and is called the *a priori* estimate, while $\hat{\mathbf{x}}_k$ is called the *a posteriori* estimate [14].

The algorithm for the Kalman filter consists of two steps; 1: prediction of the states, $\mathbf{x}$, and 2: updating of the state predictions, Kalman gain, $\mathbf{K}$, and covariance of the error, $\mathbf{P}$, for all time steps, k. Mathematically, this can be expressed as the following:

Step 1, prediction:

$$\hat{\mathbf{x}}_k^- = \mathbf{A}\hat{\mathbf{x}}_{k-1} + \mathbf{B}\mathbf{u}_k \tag{2.28a}$$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + Q \tag{2.28b}$$

Step 2, updating state predictions, covariance of error and Kalman gain:

$$\mathbf{K}_k = \frac{\mathbf{P}_k^-\mathbf{C}^T}{\mathbf{C}\mathbf{P}_k^-\mathbf{C}^T + R} \tag{2.29a}$$

$$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{C}\hat{\mathbf{x}}_k^-) \tag{2.29b}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{C})\mathbf{P}_k^- \tag{2.29c}$$

The Kalman algorithm is a recursive algorithm. This means that the algorithm is not dependent on all previous information in order to estimate the current states. The only information needed is the estimated states and the covariance matrix from the previous time step and the current measurements [14].

Nonlinear Kalman Filters

As mentioned, the Kalman filter described in the previous section is used on linear systems. As most systems are nonlinear, the Kalman filter algorithm utilized by ModelFit is an extension of the general Kalman filter for nonlinear systems. There exists a number of different extensions, but the ones used in ModelFit are Nørgaard et al.'s [18] first and second order divided difference filters (DD1 and DD2) [14].

In Malde [14], Kasper Linnestad from Cybernetica explains the difference between DD1 and DD2 in an attached e-mail. He says that in theory DD2 is more accurate than DD1, though it is also less robust and the difference in accuracy is not large enough to justify using DD2. Thus, DD1 is used in most cases in Cybernetica's tools.

Tuning

The tuning of the Kalman filter is very important, and it is indirectly by tuning that the parameters are estimated. The Kalman filter is tuned by changing the standard deviation of the noise. If the standard deviation of the process noise is set too high the measurements can be recreated perfectly as the states are updated such that the measurements and plant are matching. This will also be possible if the standard deviation of the parameter noise is too high. However, the parameters will then change dramatically from sample to sample which will lead to very bad prediction when utilizing the MPC. This is further explained in [14]. If the Kalman filter follows the measurements perfectly it either indicates an extremely good model or that the Kalman filter is updating the parameters and states too much. As it is unrealistic that the model is that good, the most likely explanation for the Kalman filter to follow the measurements perfectly is over estimation [14].

In order to achieve good predictions from the MPC, it is optimal if the parameters vary smoothly and evenly. The amount of process noise required depends on the application. For instance, if the process has many disturbances it can be a wise to set the process noise high in order for the states to change rapidly enough to get the effect of the disturbances. It is possible to change process, measurement and parameter noise in model fit [14].

2.5 Model Predictive Control

There are many different controllers that can be used when controlling a system in closed loop. Model Predictive Control, or MPC, is one of them. In Malde [14] MPC was compared to a PI-controller, where it was concluded that an MPC controller was the most suited for the OxyAid system. This was also based on general investigation of the controller with the system and thorough testing. Thus, an MPC controller is used in the project, and as the online parameter estimation is performed in order to optimize the MPC behavior, knowledge of MPC is advantageous. Therefore, an introduction of the controller will given in this section.

MPC is an advanced control technique used for process control, and other multivariable control problems. It uses available measurements of a system to predict the future output values of the system. By basing the changes in input on both the predictions and the measurements, the correct input values can be chosen [24]. Furthermore, in Mayne et. al [16] MPC is described as a controller where the upcoming input values are found by, for each time step, solving an open loop optimization problem with finite time horizon, by using the current states of the system as initial states. The optimization yields an optimal sequence of the input values, where the first value of the sequence is the input value of the system. Thus, MPC can be described as a closed loop optimization problem, where the feedback is exploited, as illustrated by Figure 2.5.

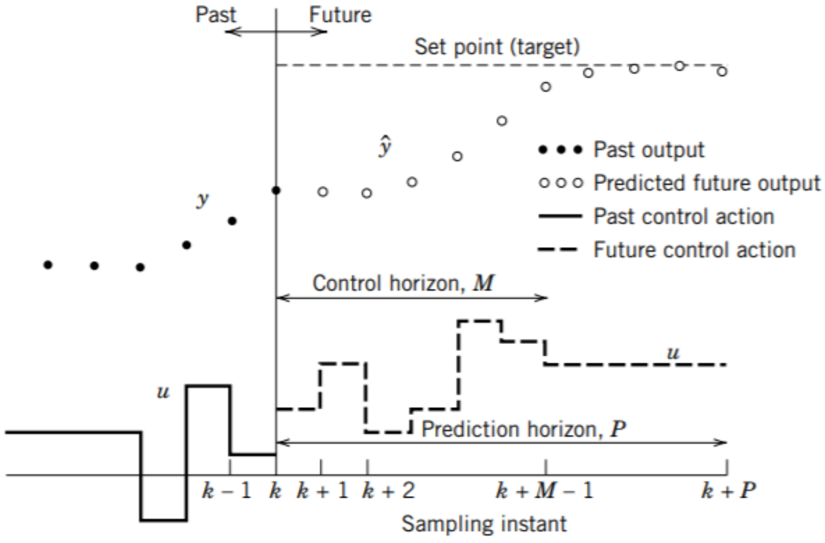


Figure 2.5: The principle of MPC, from [24]

Figure 2.5 shows the MPC principle, where y the past output, u is the past control action, $\hat{y}$ is the predicted, future output and u (dotted line) is the future control action. The figure illustrates how both the past and predicted output sequence effect the future control action. However, only the first value of the control action sequence is used. At the next time step, the optimization problem is solved again with regards to the now past output sequence and control values, along with the predicted sequences. Thus, the optimal control input relies on both the past and predicted measurements and control actions.

There are many advantages of MPC. These advantages are also a big part of why the controller is suitable for this project. Seberg et. al describes the following advantages of MPC:

> Model predictive control offers several important advantages: (1) the process model captures the dynamic and static interactions between input, output, and disturbance variables, (2) constraints on inputs and outputs are considered in a systematic manner, (3) the control calculations can be coordinated with the calculation of optimum set points, and (4) accurate model predictions can provide early warnings of potential problems.

2.5.1 Non Linear MPC (NMPC)

There exist different forms of controllers. For some systems linear controllers can be utilized. However, for some systems, nonlinear controllers, must be used. Examples of these kind of systems are systems that never reach steady-state, i.e. unstable systems, or systems that have too much disturbances so that the system is not in an "operatable region" [14].

For MPC, there is not really a linear an nonlinear variant, but nonlinear model predictive control (NMPC) is used for systems with a nonlinear model. As MPC is solving an open loop optimization problem for each time step, solving a nonlinear problem would mean to add non linear constraints, and thus the optimization problem would no longer be convex.

When the model is nonlinear there are two options; linearizing the model or using NMPC. There are pros and cons with both approaches; the pros of linearization is that the problem is much easier to solve than a nonlinear optimization problem. The algorithms needed to solve the problem demand much longer run time than a linear problem, and there are few ways to determine the quality of the solution. However, a nonlinear model is often a more precise description of the real world, and thus more precise predictions. As keeping the model as close to real life as possible is an important feature in the OxyAid project, Malde [14] concluded that the NMPC was the best solution for the OxyAid project.

Chapter 3

High Level Design of the IDP

In order to understand what the identification process entails, it is very useful to do a high level design of the system. This will ensure that it is clear what the process should do, what hard demands that exist and give a more thorough understanding of the process.

This chapter evaluates the identification process interface by using the first steps of the V-model, that is the requirement analysis and specifications, including textual descriptions, context diagrams, event lists, data flow diagrams and state transition diagrams. The chapter concludes with a function specification.

3.1 Identification Process Interface

The first step in the V-model, as according to Section 2.3.1, is a simple requirement analysis.

3.1.1 Requirement Analysis

- The system shall be able to perform both online and offline parameter estimation.
- The parameters must be realistic.
- The new system shall be more efficient than the previous, and with fewer manual steps.
- The parameters should be easily moved to the MPC.

3.2 High Level Description of IDP (SA)

This is the specification step of the V model. As previously stated, the Structured Analysis (SA) is a useful tool in this step, and in this section both context diagrams and event lists are used for both the pig trial case and the human patient case. The details of the textual description have been provided by Maria V. Ottermo and is based on previous experience in pig trials, and the plans of the human trials.

3.2.1 Textual Description

In this subsection, a concise textual description of the two scenarios at a high level will be described. This includes the order different events happens, and the action the event triggers.

The Pig Trial Case

In the pig trial, the pig is put under sedation and needs to rest for about half an hour before it has stabilized. It is the bio engineer/veterinarian responsible for the pig who tells when the pig has reached a steady state and is ready to be connected to the equipment.

When the pig is ready, it is connected to the equipment delivered by Inventas. The equipment delivers the oxygen to the pig and measures end tidal CO_2 and O_2 saturation, the pulse and breathing frequency. It also displays several lights that indicates that the equipment is on and working.

After it is connected the data collection can begin. This demands that someone starts the data collection in LabView, and starts a timer. After reasonable time, for instance half an hour, someone has to stop the data collection, and the LabView program will then automatically create csv files containing the collected data. While the data is measured it is available to be displayed at the computer screen at all times.

The next step will then be to start a program which changes the data files into a useable format, and sends it to Cybernetica's program ModelFit. This is where the parameter estimation is done, and someone will have to start the process unless a script is made that runs this automatically.

Finally, when the parameters are estimated, they need to be verified before they are transmitted to the MPC.

The Human Patient Case

The human patient case is very similar to the pig trial case. However, there is a big difference in that the patient will be awake and responsive while the pig will be under anesthesia. The patient can immediately be connected to the sensors and oxygen mask, and the data collection can start as soon as the patient is stably calm.

Otherwise, most of the procedure is the same as for the pig trial, with the exception that the patient can stop the trial if feeling discomfort or if there is any danger to the patient. In addition, a doctor will act as a clinical supervisor to ensure that mainly the patient's CO_2 level never reaches a dangerous level.

3.2.2 Detailed List of Units in the System's Environment

A detailed list of units in the system's environment is used in order to formalize the textual description, and making it easier to make the context diagrams. The units in both cases are very similar, with only minor differences which will be emphasized.

The Pig Trial Case

- Timer to keep track of data collection.
- Safety routine to ensure correct transfer of data.
- Inventas' suitcase with hardware for oxygen supply control and sensors.
- Model predictive control used for the closed loop control.
- Bioengineer to ensure the pig's well being.

The Human Patient Case

The only difference from the pig trial case to the human patient case is that instead of a bioengineer, there is a patient and a clinical supervisor.

- Timer to keep track of data collection.
- Safety routine to ensure correct transfer of data.
- Inventas' suitcase with hardware for oxygen supply and sensors.
- Model predictive control used for the closed loop control of oxygen.
- Patient with COPD that will say if it is feeling discomfort, and otherwise is assumed to be okay.
- Clinical supervisor to ensure the patient's safety. Though, during the IDP the oxygen the patient receives will not be controlled by a control system, and thus the chances that the clinical supervisor needs to intervene are very small.

3.2.3 Context Diagrams

The context diagram represent the interface of the system identification process in the system. For the context diagrams it was discussed whether or not the data logging process should be a part of the IDP or not, as it is made by Inventas. Nevertheless, as the data logging can be defined as a part of the actual identification process, it was defined as being a part of the IDP, and thus not represented as a unit in the environment. The context diagrams are developed according to the description in Section 2.3.2.

The Pig Trial Case

The context diagram for the pig trial is the graphical representation of signals going in and out of the system identification process.

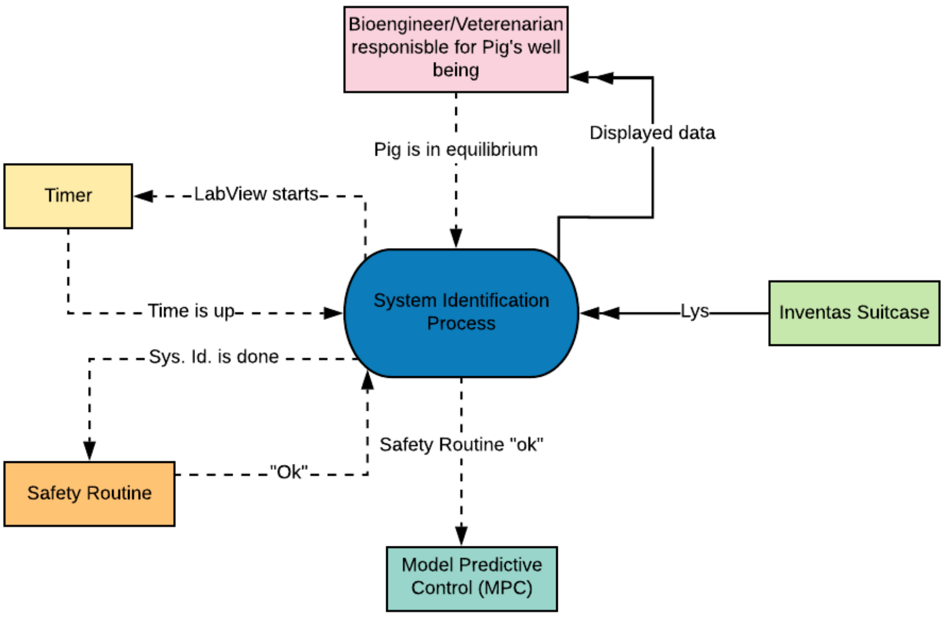


Figure 3.1: Context diagram for the pig trial case

The Human Patient Case

The human patient case is almost the same as for the pig trial case, with the exception of that there is a patient instead of a pig, and a clinical observer in addition.

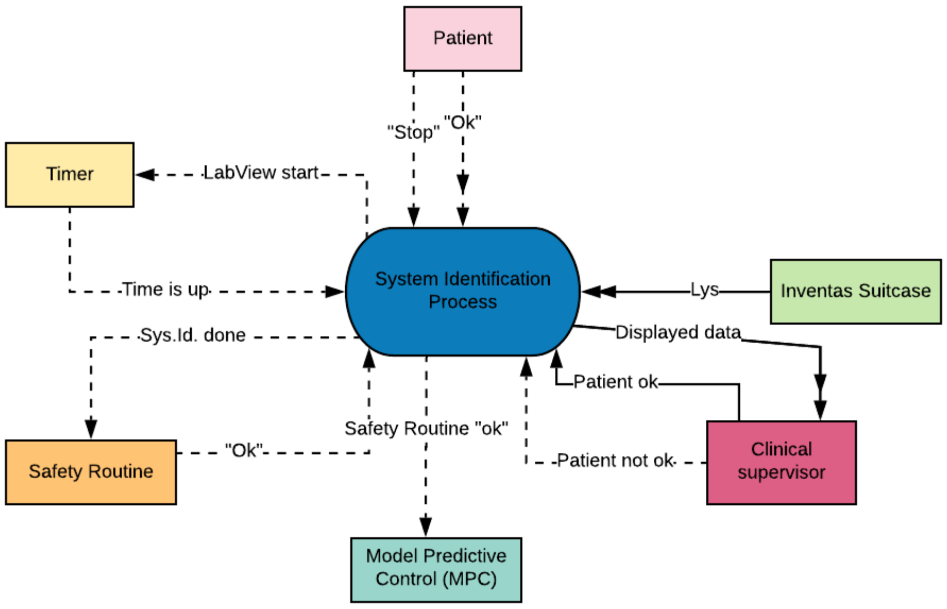


Figure 3.2: Context diagram for the human patient case

3.2.4 Event Lists

In this sub section the event lists based on the context diagrams will be presented. These are useful in regards to getting overview of the time line of the IDP, i.e. in what order different events happen.

The Pig Trial Case

Event	**Input stream**	**Type**	**Reaction**
Bioengineer informs that the pig is in equilibrium	"Ok"	Continuous control	Inventas' equipment is connected and started, the sensors and oxygen mask is placed on the pig, LabView is opened.
Inventas' equipment is connected	Data in Labview	Discrete data	The logging in LabView is started.
Logging is started	Data is logged	Discrete data	The timer is started.
The timer reaches time limit	Time	Continuous control	The logging in LabView is stopped.
Log files are created	Files	Continuous control	Script for formatting of the log files is run.
Script finishes	Csv files are created	Discrete data	Data is transferred to ModelFit.
Data in ModelFit	Datastream	Discrete data	Program in ModelFit is started.
ModelFit finishes	Data	Discrete data	Data is transferred to the MPC.
Data is transferred to MPC	Data	Discrete data	Safety routine is activated.
Safety routine is enabled	"Ok/not ok"	Continuous control	If ok: start control system, else: start over.

Table 3.1: Event list for the pig trial case

The Human Patient Case

There are not many differences between the two cases as this is an event list, and continuous information streams, such that the patient is okay, is not included.

Event	**Input stream**	**Type**	**Reaction**
Bioengineer informs that the pig is in equilibrium	"Ok"	Continuous control	Inventas' equipment is connected and started, the sensors and oxygen mask is placed on the pig, LabView is opened.
Inventas' equipment is connected	Data in Labview	Discrete data	The logging in LabView is started.
Logging is started	Data is logged	Discrete data	The timer is started.
The timer reaches time limit	Time	Continuous control	The logging in LabView is stopped.
Log files are created	Files	Continuous control	Script for formatting of the log files is run.
Script finishes	Csv files are created	Discrete data	Data is transferred to ModelFit.
Data in ModelFit	Datastream	Discrete data	Program in ModelFit is started.
ModelFit finishes	Data	Discrete data	Data is transferred to the MPC.
Data is transferred to MPC	Data	Discrete data	Safety routine is activated.
Safety routine is enabled	"Ok/not ok"	Continuous control	If ok: start control system, else: start over.

Table 3.2: Event list for the human patient case

3.2.5 Data Flow Diagrams (DFD)

A data flow diagram shows the subsystems in a system, and how the information flow between the subsystems. For the IDP the subsystems are the data collection process, the data formatting process and the actual parameter estimation. This is the first step in the behaviour model. As previously mentioned, the data collection process is performed using equipment and a program made by Inventas, it is still included as a subsystem as it contains many manual steps that must be performed by an OxyAid representative. In addition, it is crucial for the parameter estimation, and thus defined as a part of the IDP.

As shown in the event lists, of which the DFD is based on, there are not many differences to the actual IDP between the pig trial case and the human patient trial case. Therefore, only one data flow diagram will be presented, while the difference from the pig trial case to the human patient trial case will be listed below.

The Human Trial Case

As the human patient trial case is the main goal of the OxyAid project, this is the main focus of this report. Therefore, when it was only deemed necessary to make one data flow diagram the human patient trial case was chosen.

For the human patient trial case, there are, as mentioned three subsystems. While the communication with the environment is very much the same as for the context diagram, there are naturally also some communication between the subsystems. This is important to be aware of when implementing the system.

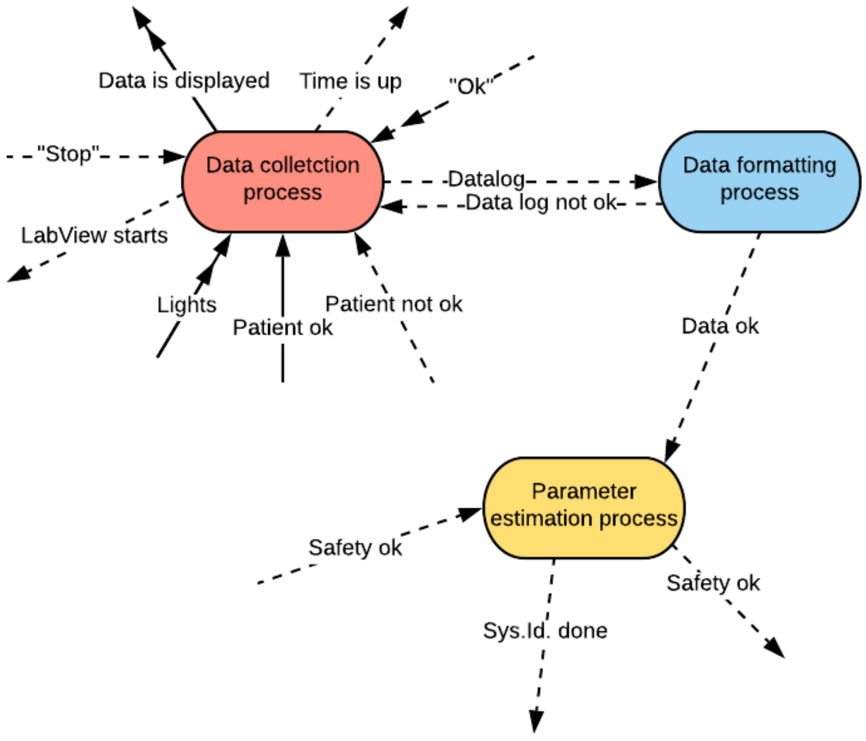


Figure 3.3: Data flow diagram for the human patient case

As seen from the figure, information about the patient's well being is only included in the data collection process. The reason for this is that the only time the patient is connected to the IDP is during the data collection. The other subsystems run while the patient is not connected, and thus should not feel any discomfort caused by the system at that time.

The Pig Trial Case

The only difference between the human patient trial case and the pig trial case is some of the information flow with the environment. While there is a clinical observer and a patient with continuous information about the patient's well being in the human patient case, there is a bioengineer that is in charge of the pig's well being during the trial in the pig trial case. Otherwise, the subsystems are the same for both cases, and most of the communication with the environment is the same as well as the communication between the subsystems.

3.2.6 Textual Description of the Subsystems

Based upon the DFD, a textual description of each of the subsystems is included. This is a detailed description of each process, and is similar to the textual description of the general IDP though a lot more detailed. As for the DFD, only a textual description for the human patient trial case will be given.

The Data Collection Process

- In the "waiting state", the computer screen shows the LabView program made by Inventas. A graph displaying the patient's CO_2 level is shown on the screen.
- When the user starts the logging, the "log button" is pressed and a bright green light is shown on the button. A timer is started.
- When the time is up, the user must press the "log button" again. The log files are saved to a folder. This information is sent to the data formatting process

The Data Formatting Process

- The user can run the data formatting process once the log files are saved to a folder. A program is then ran which changes the files into a format that can be put directly into the parameter estimation program.
- When the program has run, it will either notify that the data formatting fails (if this is the case) to the data collection process. Else, information that the data has been successfully formatted is sent to the parameter estimation process.

The Parameter Estimation Process

- When getting information that the files are formatted, the user can start the parameter estimation program, and submit the data from the files.
- The user can then run the offline parameter estimation, and choose to update the parameters with the estimated values.
- Secondly, the online parameter estimation can take place. This is changing the standard deviation of the noise, as described in Section 2.4.3.
- Finally, the values from the parameter estimation can be sent to the MPC.

3.2.7 State Transition Diagrams (STD)

As stated previously, the final step in the behaviour model is the state transition diagrams for each of the subsystems. However, as the previous sections cover the system quite well, the state transition diagrams were excluded from this report as it would be mostly redundant information.

3.3 Claims for IDP

In this section, the claims for the IDP will be presented. These are requirements the system must fulfill, and has been developed and discussed with Maria V. Ottermo.

3.3.1 Real Time Claims

There are several reasons why it is important to have real time claims in for the system.

Firstly, a trial lasts for a limited time and to spend too much time on the system identification would mean that there will be less time to run the actual control system. This would be disadvantages as the control system is the main part of the project and needs thorough testing.

Secondly, collecting the data needed for the parameter estimation requires the patient to stay calm and still without any oxygen supply. This can be challenging for patients who partake in the trial as they are reliant on LTOT, and therefore may become uncomfortable with being without oxygen supply for an extensive period of time.

Hence, the real time claim for the system is:

- The IDP shall take no more than 40 minutes (reasonable time). This includes the data collection process.

3.3.2 Other Claims

In addition to the real time claim, the system also has other claims it must fulfill:

- Minimize erroneous parameter estimations and other errors.
- The IDP shall be as autonomous as possible.
- Important data shall be transferred digitally to avoid human errors.
- There shall be no manual scaling of data.
- There shall be only one sample per second as the system is too slow to note changes more than once per second.

3.4 Function Specification

The obtained function specification is also the base for the acceptance test. All of the discussed features in the chapter results in the following list:

- The system shall autonomously convert the data files from the Labview program into a form that can be used. This includes conversion from .csv file to .xlsx form, divide the data from one to multiple cells in the rows and change the time to date-time format (originally the time is given in seconds since January 1 1970).
- The system shall autonomously check if there are several samples within a second, and if so only keep one sample to reduce the data size.
- The system shall ensure that the sample time from the different logging instruments match each other in the file containing the data for the parameter estimation.
- The system should automatically insert the formatted data sheets in ModelFit for the parameter estimation.
- If it is not possible to automatically insert the formatted data in ModelFit, it should be on a form that enables easy copying.
- The IDP shall take no more than 40 minutes.
- There shall be a safety mechanism to ensure that the estimated parameters are correctly transferred to the MPC.
- There shall be a safety mechanism to ensure that the estimated parameters are likely and realistic before being transferred to the MPC.
- It shall be easy to move the parameters to the MPC, and this should be done autonomously.
- The estimated parameters should be presented in a way that is easy to read and should be accepted before being sent to the MPC.
- All manual steps of the process shall be thoroughly explained in a way so that a person with minimal training is able to perform the IDP.
- The chosen mathematical model shall be implemented and present when opening the parameter estimation program.
- The IDP shall perform both online and offline parameter estimation.
- The parameters estimated should be realistic and probable.

3.4.1 System Tests

From the function specification it is clear that there are especially two things that must be tested on system level; the performance of the system, i.e. that everything works as supposed to and that the identification process produces realistic results, and the effectiveness, i.e. that time is saved on using the new process. Thus this will be tested when the system is implemented.

Chapter 4

Low Level Design of the IDP

The previous chapter aimed to plan the system on a high level. This chapter goes more in depth, and contains the low level design. Thus, this chapter represents the last steps in the V-model before the actual coding. In addition, this chapter also contains an evaluation of software options.

4.1 Overall IDP Procedure

This section will present the evaluation of the design steps of the V-model for this system; the architectural and detail design. These steps also includes the development of the unit and integration tests used in the later steps of the V-model.

4.1.1 Architectural Design

Chapter 3 presented the high level design of the system. Thus, this subsection will simply present how the defined modules communicate by a graphic representation, and define the integration tests.

High Level Design of the System

From Section 3.2.5, it is clear that the system will consist of three different modules. The different modules are a data logging module, a file formatting module and a parameter estimation module. This is defined by the function specification developed in the previous chapter, and is presented here through a sequence diagram:

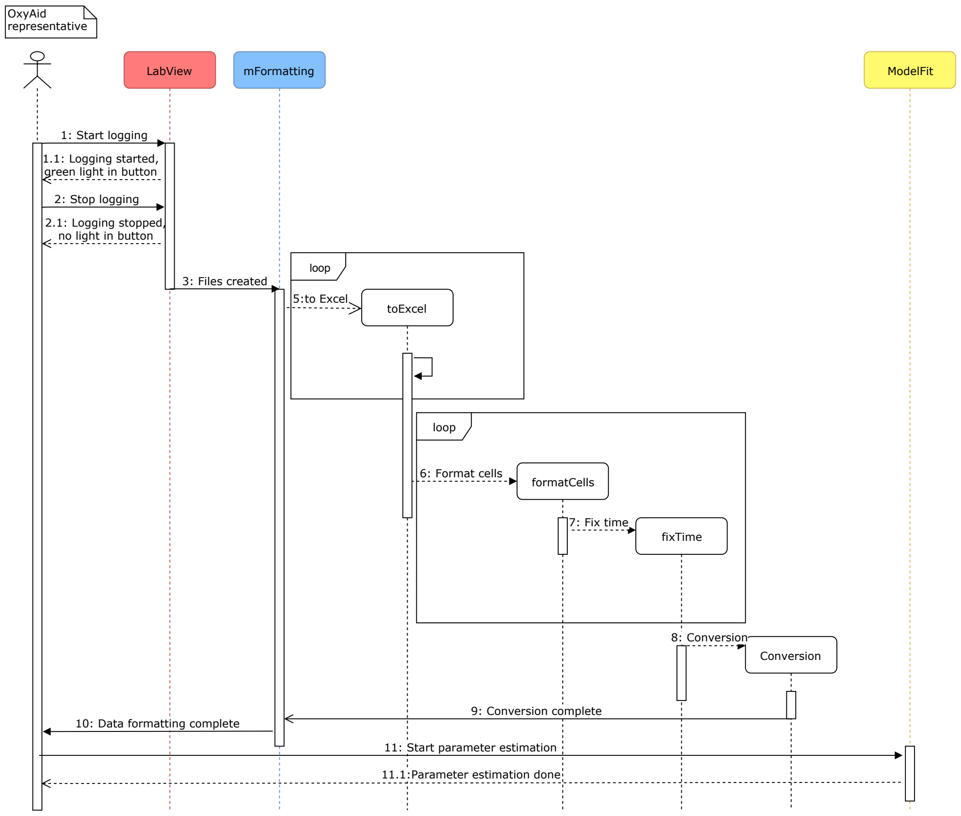


Figure 4.1: Sequence diagram of the IDP

In figure 4.1 the IDP is represented with a sequence diagram. The first object represents the LabView program (the data logging module), the blue is the main function of the data formatting module and the yellow is the parameter estimation module. The white ones represent important functions in the file formatting module, and describes its behaviour.

Integration Tests

Based on the sequence diagram, the following integration tests are suggested:

- Test that the MATLAB program works with the Labview program by including the current date in the evaluation of the files.
- Test that the MATLAB program works with the LabView program by making a function that returns an error if the files are not found where they are expected to be.
- Test that the ModelFit program works with CENIT by transferring the estimated parameters and standard deviations and running the MPC.
- Test that all of the modules work together by testing the system, first on test files, but later on in a pig trial.

4.1.2 Detail Design

In the detail design, each module is planned in detail. This includes planning of the functions that are to be included in the modules. In addition, the unit tests are also created.

Low Level Design of the System

The LabView program has already been made by Inventas, and in the parameter estimation Cybernetica's program ModelFit will be utilized. Thus, the only module requiring detail design is the MATLAB program for the data formatting.

From the function specification that concluded Chapter 3, it is clear that the following functions must be included:

- A function that gets the path of the MATLAB program in order to know where the log files are to be moved.
- A function that gets the current date and format it in the same manner as the LabView program, i.e. YYMMDD.
- A function that moves the log files to a program folder is required.
- A function that converts the files from .csv to .xlsx format.
- A function that splits the cells so that one piece of data is in one cell each is necessary.
- A function that changes from number of seconds to standard time, as the LabView program logs the time as seconds since January 1. 1970.
- A function that ensures that there is only one sample per seconds as this is one of the requirements.
- A function that changes the oxygen input from concentration in percent to partial pressure.
- A function that changes the patient's oxygen level from percent to fraction.
- A function that gathers the relevant data in one file in order to make the transaction to the parameter estimation program as easy as possible.
- A function that transposes the columns is beneficial as it then is possible to simply copy and paste the data into ModelFit.

The function for changing the input oxygen to partial pressure is needed for the pig trials, as the oxygen level will be changed during the data collection. For the human trials this may be held constant, or the patient may not receive any extra oxygen at all.

In addition, the mathematical model should be implemented in Visual Studios and imported to ModelFit beforehand.

Unit Tests

The unit tests are tests that test each function in the different modules. The following unit tests have been proposed:

LabView

Although Inventas has made the LabView program which logs the data, it is useful to test the program to ensure that it works as expected on the respected computer.
The following tests are thus proposed:

- Test that the data that is supposed to be displayed is displayed.
- Test that the program logs the data as it is supposed to do.

MATLAB

As the MATLAB program is made from scratch, it is important with thorough testing before using the program in a trial.

- Test each function separately to ensure that they work as they are meant to.
- Test that functions work together.
- Test that the calculations of the time, partial pressure and O_2 level yields the correct values.
- Test that they work while in a for-loop which iterates through all the files in a folder.

ModelFit

Although Cybernetica has made the program used for parameter estimation in this project, it is still important to perform unit tests as the mathematical model has been implemented and it is important to ensure that the estimated parameters are likely and realistic.

- Test whether the implemented mathematical model behaves as expected when running the model.
- Test that the estimated parameters in the offline parameter estimation are likely and realistic.
- Test that the estimated parameters in the online parameter estimation are likely and realistic.

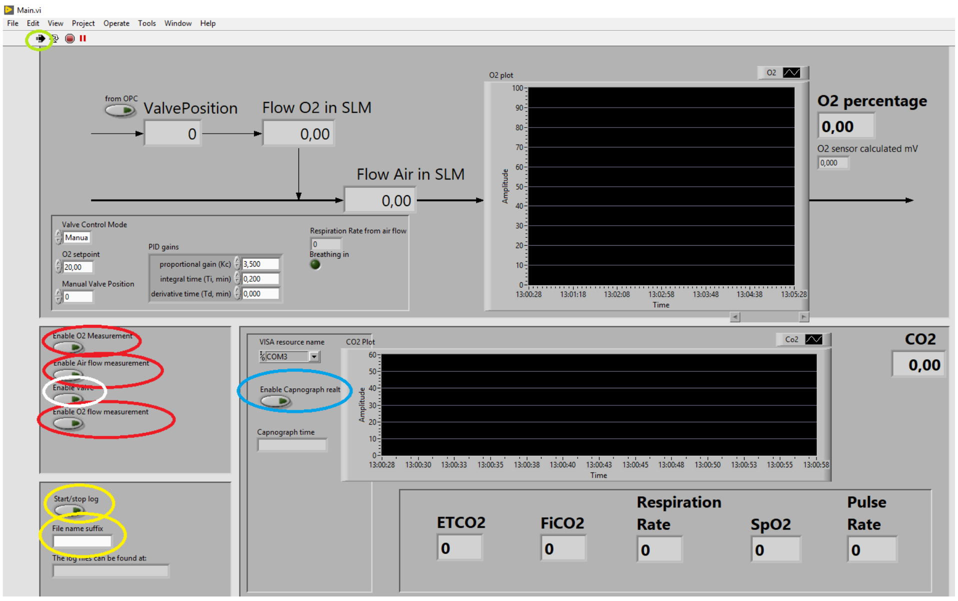
4.2 Software and Programming Platform Options

One of the problem objectives is to discuss the software and programming platform options that are chosen in the IDP. Along with a discussion of the opted programming platform, an introduction to each of the programs is included.

4.2.1 Inventas' Labview program

The first part of the identification process is to record the data that will be used in the parameter estimation. In previous trials, medical equipment available at St.Olavs Hospital was used, and the parameter estimation done after the trial. Thus, this has not been included in previous definitions of the IDP.

The data needed is the level of end tidal CO_2 ($ETCO_2$), the oxygen saturation (SaO_2), the oxygen level given to the patient, the heart rate and the breathing frequency, i.e. the input and output of the mathematical model. The OxyAid project hired Inventas to connect all the devices needed for controlling the oxygen level and recording the data, and make a program where the CO_2 and O_2 level of the patient would be graphically visible and log the data when wanted. Obviously, this program is the one being used for data logging in this project as well.



ValvePosition: 0
Flow O2 in SLM: 0,00
Flow Air in SLM: 0,00
O2 percentage: 0,00
O2 sensor calculated mV: 0,000
Valve Control Mode: Manua
O2 setpoint: 20,00
Manual Valve Position: 0
PID gains
proportional gain (Kc): 3,500
integral time (Ti, min): 0,200
derivative time (Td, min): 0,000
Respiration Rate from air flow: 0
VISA resource name: COM3
Capnograph time:
File name suffix:
The log files can be found at:
CO2: 0,00
ETCO2: 0
FiCO2: 0
Respiration Rate: 0
SpO2: 0
Pulse Rate: 0

Figure 4.2: Inventas' LabView program

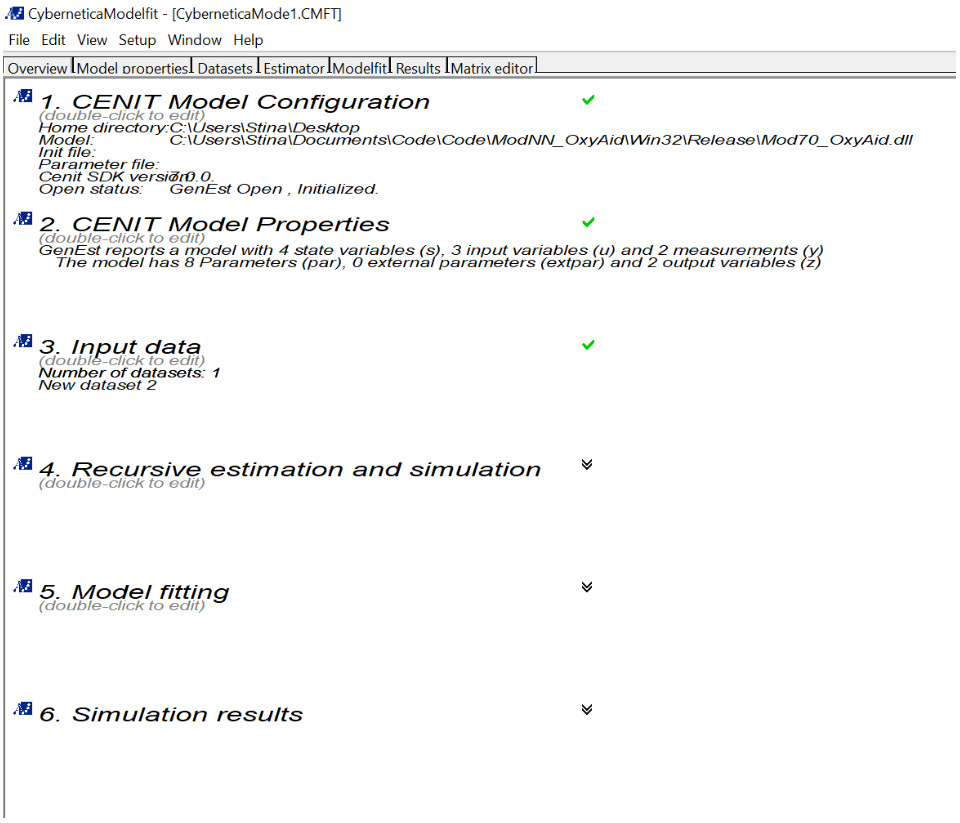
Figure 4.2 shows a print screen of the LabView program made by Inventas. In order to log any data, the buttons with the red circle around them must be pressed. These enable different information to be logged. In addition, the button with the yellow circle around it must be pressed as this starts the logging. Other important features are also circled, and will be discussed in Chapter 5.

4.2.2 MATLAB

The file format program could have been made in a variety of different programming languages, such as Python or C#. However, as MATLAB offers easy access of .csv and .xlsx files and is a very familiar tool, MATLAB was chosen, and the file formatting program was implemented according to the function specification. Further details of the code and manual steps in order to run the program will be given in Chapter 5.

4.2.3 Cybernetica's ModelFit

For the parameter estimation, MATLAB was also considered as it has build-in packages and other toolboxes that work well. However, in [14] ModelFit was explored, and proved to be a very competent tool for parameter estimation, both offline and online. In addition, as Cybernetica has made a program for MPC that the OxyAid project planned to use, using Cybernetica's program ModelFit was deemed the best solution for the parameter estimation after discussions with Øyvind Stavdahl.


CyberneticaModelfit - [CyberneticaMode1.CMFT]
File Edit View Setup Window Help
Overview
Model properties
Datasets
Estimator
Modelfit
Results
Matrix editor

1. CENIT Model Configuration ✓
(double-click to edit)
Home directory:C:\Users\Stina\Desktop
Model: C:\Users\Stina\Documents\Code\Code\ModNN_OxyAid\Win32\Release\Mod70_OxyAid.dll
Init file:
Parameter file:
Cenit SDK version: 7.0.0.
Open status: GenEst Open , Initialized.

2. CENIT Model Properties ✓
(double-click to edit)
GenEst reports a model with 4 state variables (s), 3 input variables (u) and 2 measurements (y)
The model has 8 Parameters (par), 0 external parameters (extpar) and 2 output variables (z)

3. Input data ✓
(double-click to edit)
Number of datasets: 1
New dataset 2

4. Recursive estimation and simulation
(double-click to edit)

5. Model fitting
(double-click to edit)

6. Simulation results

Figure 4.3: An overview of Cybernetica's ModelFit

Figure 4.3 shows an overview of ModelFit, and this window is shown when selecting "File" -> "New", or clicking on "Overview".

In figure 4.3, there are different options at the top, where the "overview" is the one being shown in the figure. In "Model Properties" all the values for parameters, states, input, output etc. are displayed. Under "Datasets" the input values and measured values can be entered, and the initial values of the parameters and states can be changed, along with other data like for instance process and measurement noise variance, Kalman gain and upper an lower limits for system variables. In "Estimator" the model can be ran and input data, states and output data can be shown graphically. This is a useful tool to ensure that the input data looks good and to compare the measured and estimated output.

By running the model in the estimator before and after the offline parameter estimation, it is easy to compare the difference in how well the estimated output follows the measured output before and after the parameter estimation. The online parameter estimation is also done in "Datasets" by using "Estimator" to evaluate the performance. Finally, in "ModelFit" the offline parameter estimation can be done.

Chapter 5

Implementation

The main focus of this thesis has been the design and implementation of a program that optimizes the system identification process of the OxyAid project. As one of the objectives of the thesis is to explain the code and provide a "recipe" on how to run the system identification process, this chapter will present the general implementation of the system, any precautions that must be made before starting the IDP, and instructions on how to run it.

5.1 The Code

In this section the code for the IDP will be discussed. As the whole file formatting module was coded from scratch, this will be discussed. In addition, although ModelFit was used for the parameter estimation, the mathematical model still needed to be implemented and therefore the parameter estimation will also be a subsection.

5.1.1 File Formatting

As previously mentioned in Chapter 4, a number of functions were implemented. The purpose of the functions are all listed in Section 4.1.2, and is thus unnecessary to repeat. However, Inventas' made a second program intended to be used in human trials in December 2019. Therefore, some minor functions were implemented in order to adjust to the new format. This included a function where the input oxygen level is sat, with the user deciding the value. Otherwise, the file formatting program is the same for the human and pig trials. The functions were implemented as described with the main function of the program presented below. As this gives a good overview of the file formatting program implemented, this will not be addressed any further in this section. The sequence diagram in Figure 4.1 gives a good graphical representation of the general implementation of the module.

```
function [] = main()
```

```
path = getCurrentPath();
dateMonthYear = getTodaysDate();

newMatlabOxyPath = path + dateMonthYear;

%Check that LabView ran correctly, prints error if not
checkIfFilesExist(dateMonthYear, 'C:\OxyAid_logs')

moveFilesToProgramFolder(matlabOxyPath,
'C:\OxyAid_logs');

%Iterates through the files and converts them to .xlsx
files=fullfile(matlabOxyPath,'*.csv');
fileInfo=dir(fileInfo);
for k=1:numel(fileInfo)
    filename=fullfile(matlabOxyPath,fileInfo(k).name);
    fromCsvToExcel(filename);
end

%Iterates through the files and changes the format and
time to the correct format
files=fullfile(matlabOxyPath,'*.xlsx');
fileInfo=dir(fil);
for k=1:numel(fileInso)
    filename=fullfile(matlabOxyPath,d(k).name);
    formatCellsInExcelSheet(filename);
    fromSecondsToDate(filename);
    onlyOneSamplePerSec(filename);
    nameTimeColumn(filename);
    deleteLastRow(filename);
end

clear dir

fileList = dir(fullfile(matlabOxyPath,'*.xlsx'));

airO2File = fileList(1).name;
fromConcentrationToPartialPressure(AirO2);

newFile = "raw_data_trial" + dateMonthYear + ".xlsx";

CapnoNumeric = fileList(3).name;
fromPercentToFraction(CapnoNumeric);
```

```
    saveDataToOneFile(CapnoNumeric, newFile);
    correctNamesToColumns(newFile);
    transposeColumns(newFile);

end
```

5.1.2 Parameter Estimation

In order to use ModelFit, a model template must be filled out in Visual Studios.

There are several different files in the template in which the model information must be updated.

From Section 2.2.2 it is clear that the model has 4 states ($pa_{CO_2}, pa_{O_2}, Cv_{CO_2}, Cv_{O_2}$), 3 inputs ($pi_{O_2}$, F_T, F_H), 8 parameters (MT_{CO_2}, MT_{O_2}, V_{T,CO_2} V_{T,O_2}, V_{A,CO_2}, V_{A,O_2}, SV, V_{eff}), 5 constants (S_{O_2}, K_{CO_2}, k_{CO_2}, C_{Hb}, b_{Hb}) and 2 outputs (ET_{CO_2}, Sa_{O_2}). In addition, for the pig trial case two extra constants are added; one offset for the partial pressure of oxygen and one offset for the oxygen saturation.

In "model_tags.c", all the names of the states, inputs, parameters, constants and outputs along with the units must be defined. While in "model_index.h", the vectors for the states, input, parameters, constants and outputs must be defined, along with the index number of each variable. "model_variables.h" is almost exactly the same as "model_index.h", except that the time derivative of the states are also defined and instead of the index number, the variables are defined as, for instance, xs[i_xs_paCO2] and so forth.

In «model.h», the dimensions are defined. For the implemented mathematical model, the following must be defined:

```
    #define NS       4  //Number of ordinary states
    #define NPAR     8  // Number of parameters
    #define NINIT    NS  // Number of initialization variables
    #define NCON     7  // Number of constants
    #define NU       3  // Number of process inputs
    #define NY       2  // Number of measurements
    #define NZ       2  // Number of derived output variables
    #define NV       2*NPAR+NS // Number of process
    disturbances
    #define NW       NY  // Number of measurement noise
    variables
    #define NCONPAR 3*NZ  // Number of reference trajectory
    parameters
    #define NCONVAR 2*NZ  // Number of reference trajectory
    variables
    #define DT       1  // Sampling time for estimator
    #define DT_INT  0.1  // Sampling time integrator[s] DT_INT
    <= DT. Set small enough to ensure stable integration.
    #define NDT      1  // Sampling time for Nmpc: NDT * DT
```

Finally, in "model.c" the model equations are implemented along with the initial conditions. The initial values for the states and parameters are developed from research by Maria V. Ottermo and Øyvind Stavdahl. They are:

```
//State inits
init_paCO2 = 40;
init_paO2 = 95;
init_CvCO2 = 0.5;
init_CvO2 = 0.27;

//Parameter inits
par_MTCO2 = 0.22;
par_MTO2 = 0.29;
par_VTCO2 = 15;
par_VTO2 = 6;
par_VACO2 = 3.2;
par_VAO2 = 2.5;
par_SV = 0.05;
par_Veff = 0.3;

//Constants
con_sO2 = 0.0031;
con_KCO2 = 0.0057;
con_kCO2 = 0.224;
con_CHb = 0.15;
con_bHb = 1.34;

//Offsets added for the pig trial case
con_pigOffset1 = 70;
con_pigOffset2 = 0.05;
```

5.2 Preparations

As mentioned, in order to run the IDP certain preparations are in order. In this section, the necessary precautions will be discussed.

5.2.1 LabView

In order to run the LabView program the 2018 version of LabView along with some important packages (Real Time Engine, NIOPCUA and a DAQ driver) must be installed on the computer. In addition, the suitcase with the hardware must be connected to the computer before the program is started. Finally, in order for the log files to be saved, a folder named "OxyAid_logs" must be created in the path "C:\".

5.2.2 MATLAB

MATLAB must be installed. As long as the data logging has been performed, no other preparations are needed in order to run the MATLAB program.

5.2.3 ModelFit

In order to run the parameter estimation, ModelFit must be downloaded and installed. If the model has been updated, or it is the first time starting ModelFit, the model must be added.

The model is added by opening ModelFit, and choosing the option "Start without document". Then, click on "File" and "New". This will lead to a new page opening, such as the one in Figure 4.3, though without the green check marks. Double click on the option "CENIT model configuration", and choose the .dll file under the folder ModNN_OxyAid -> Win32 -> Release. This will add the implemented mathematical model to ModelFit, and "CENIT Model Configuration" and "CENIT Model Properties" should now have green check marks as depicted.

5.3 Running the IDP

In this section, the manual steps required in order to run the implemented system will be presented. The steps will be presented in the order they need to be executed when running the program.

5.3.1 Data Collection in LabView

Assuming all preparations are completed successfully, the data collection can begin.

- Start LabView. Open the project "OxyAid_IO.lvproj", and then "main.vi" from there.
- Run the program by clicking the white arrow. This should lead to the CO_2 level appearing in the largest graph.
- In order to start logging, all the buttons with the desired things to be logged need to be clicked. The light in the button should go from dark green to light green when clicked. Finally, the log button must be clicked.
- In order to stop logging the log button must be pressed again. The log files can now be found at "C:\OxyAid_logs"

5.3.2 File Formatting in MATLAB

The file formatting program in MATLAB must be run after logging the data to be used in the parameter estimation. The program changes the files from the LabView program to the correct units and on a form that can simply be copied to ModelFit.

- Open the Matlab file "fileFormatting.mat"
- Run the script. This should create a folder in the program folder named "OxyAid_logsDATE" with DATE being the current date. In this folder the original .csv files can be found, the updated .xlsx files, and a file containing all the data needed for the parameter estimation with transposed columns to enable smoothest possible copying.

5.3.3 Parameter Estimation i ModelFit

Finally, the actual parameter estimation can take place. This is done in Cybernetica's program ModelFit.

- Open ModelFit, and choose the option "Start without document"
- Click on "File" and "Open..." and choose CyberneticaMode1.CMFT. It should be in the folder ModNN_OxyAid -> Win32 -> Release.
- Double click on "Datasets". Open the file "raw_data_trialDATE.xlsx" that can be found in the "OxyAid_logsDATE" folder in the MATLAB program folder. Check how many rows there are, and enter this number where it says "Data points in dataset" in ModelFit, and click "Add new dataset".
- Copy the data from the excel sheet to the correct places. The place the input data goes can be found under "New DataSet 1" -> "Time series" -> "input values (u)", and the measured output values under "New Dataset 1" -> "Time series" -> "Measurement values (ym)".

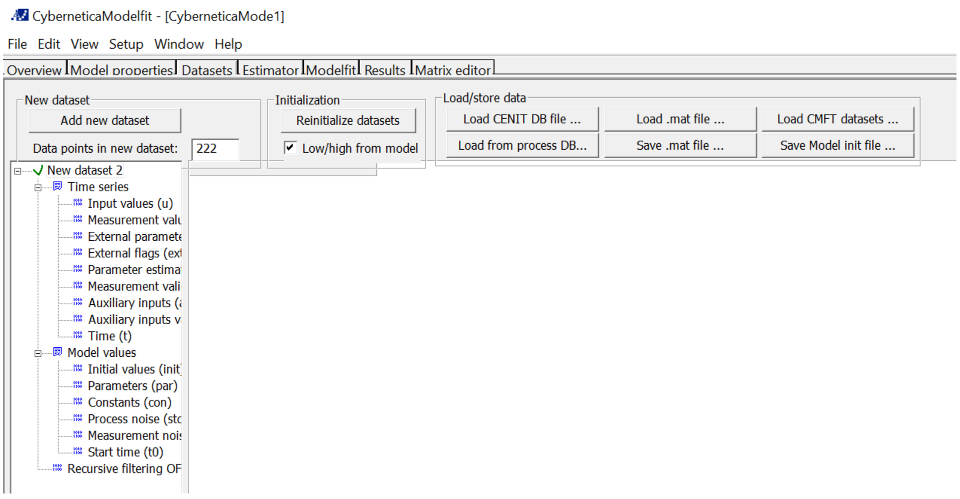


Figure 5.1: "Datasets" in ModelFit

Figure 5.1 shows the "Datasets" page in ModelFit. On the left, the menu showing the added dataset "New dataset 2" is shown. By clicking on the different options input data, measurement data, parameter values, initial values, etc. can be changed.

- Click on "Estimator" and "Run" to see how the model runs with the inserted values before any adjustments are made.
- In order to get canvases, click on "New Canvas". In order to change the number of canvases, insert the number of desired canvases and click on "Resize".
- Right click on a canvas and choose "Select variables" -> "Output" and choose which variables that are to be displayed. As ModelFit does not necessarily automatically adjust the scaling, it may be wise to right click and choose "Zoom 100% both axes" in order to get a better dimension on the graph.

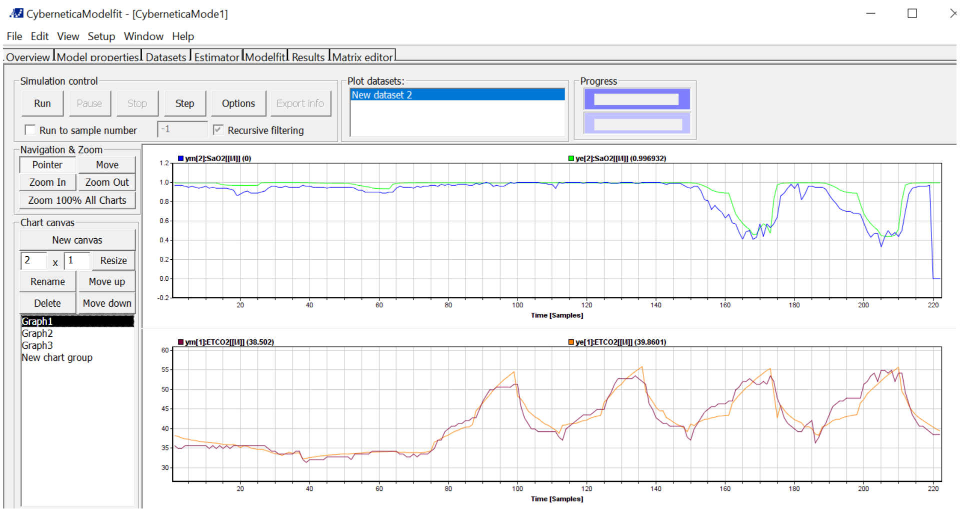
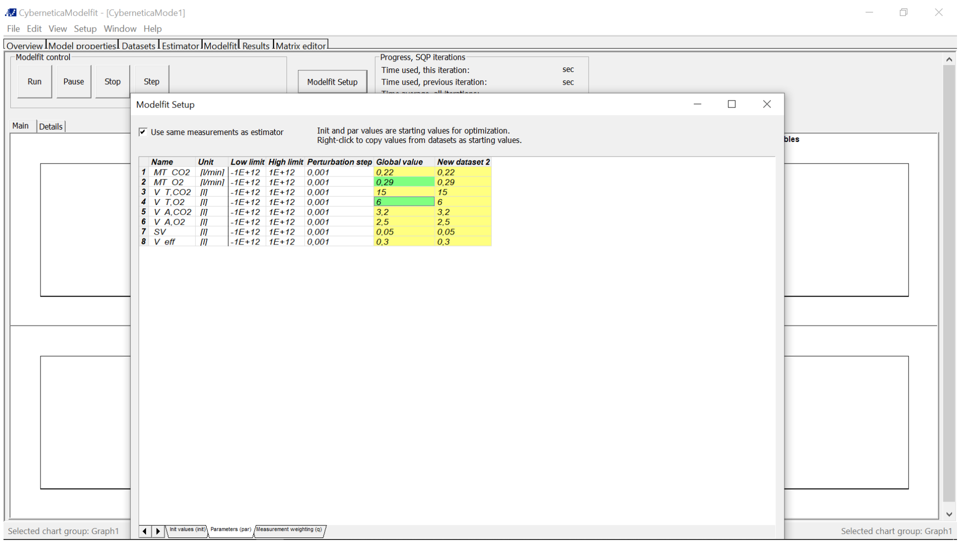


Figure 5.2: "Estimator" in ModelFit

Figure 5.2 shows the "Estimator" page in ModelFit. Here, a model is run, and the measured and estimated outputs are chosen in "select variables". On the left, the canvas options are shown.

- In order to do the offline parameter estimation, click on "ModelFit", and choose "Modelfit Setup".
- It is important that the perturbation step is changed if this is too big (default value is 2000000000, it should be around 0.001). Too big perturbation step will lead to the optimization problem to converge too early (probably after one step) and without finding a suited optimal value. This must be done for both init values and parameters.
- In order to select the parameters chosen for the parameter estimation, right click on the desired parameters and select "Toggle for optimization".
- Finally, the measurement weighting must be sat. This is a number between 0 and 1 and is to what degree the measurements can be trusted.



Progress, SQP iterations
Time used, this iteration: sec
Time used, previous iteration: sec

Modelfit Setup

☑ Use same measurements as estimator

Init and par values are starting values for optimization.
Right-click to copy values from datasets as starting values.

	Name	Unit	Low limit	High limit	Perturbation step	Global value	New dataset 2
1	MT CO2	[l/min]	-1E+12	1E+12	0,001	0,22	0,22
2	MT O2	[l/min]	-1E+12	1E+12	0,001	0,29	0,29
3	V T,CO2	[l]	-1E+12	1E+12	0,001	15	15
4	V T,O2	[l]	-1E+12	1E+12	0,001	6	6
5	V A,CO2	[l]	-1E+12	1E+12	0,001	3,2	3,2
6	V A,O2	[l]	-1E+12	1E+12	0,001	2,5	2,5
7	SV	[l]	-1E+12	1E+12	0,001	0,05	0,05
8	V eff	[l]	-1E+12	1E+12	0,001	0,3	0,3

Figure 5.3: "Modelfit" in ModelFit

Figure 5.3 shows the "Modelfit" page in ModelFit, with "ModelFit Setup" open. As seen from the figure, the perturbation step is sat to 0.001, and the green means that these are the parameters to be estimated in the offline parameter estimation.

- Close "ModelFit Setup" and click on "Run" in order to do the offline parameter estimation.
- When the parameter estimation is done a pop-up window will appear asking to update the parameters, click on "Yes". The offline parameter estimation is now done.
- Go to "Estimator" to perform the online parameter estimation, and choose "Recursive filtering".
- Right click and apply "Fill with ones" to the parameter that is to be tuned under "Datasets" and "Parameter estimation (ixpar).
- In order to adjust the standard deviations, go to "Model properties", and choose "Process noise (std_v)" and "Measurement noise (std_w)" under "GenEst".
- Try changing the standard deviations until the model behaves in a desirable way, as described in Chapter 2.

5.3.4 Transfer of Parameter Values to the MPC

According to Kasper Linnestad, there are two choices of transferring the estimated parameter values and standard deviations for the noise to the MPC. The first one is to hardcode the new values in the .dll file, while the second is to save an initialize file from ModelFit. As the first option is prone to human error, the second option is opted, and thus described below:

- Go to "Datasets", and choose "Save Model init file ...". This will create a text file containing the optimized values.
- Open CENIT MMI, and assign the file in "Assign Model and Estimator" under "Parameter file".
- In the CENIT .csv configuration file, the parameter file must be specified as "FileInitialState".

Chapter 6

Results and Discussion

In this chapter the plots and results from the system simulations, offline and online parameter estimation will, along with discussions, be presented. In addition, evaluations of the system tests will be provided. Finally, the system will be evaluated in regards to the function specifications.

6.1 Results of Parameter Estimation

In this section the results from simulations before any adjustments and the parameter estimations will be presented, both the offline and online.

The data used to simulate the model and perform the parameter estimations is from two different pig trials. The first pig trial was done May 29th 2018, while the second was done June 26th 2019. Hereafter, they will be referred to as pig trial A and pig trial B, respectively. As there were delays with the applications for new trials, this did not happen as planned within the time scope of this project, and thus the IDP designed in this project has only been tested with built in program in the capnograph that generates data, and not in any trials.

6.1.1 Simulations

After implementing the mathematical model described in Section 2.2.2 according to the procedure described in Section 5.1.2, the system was simulated in ModelFit using data from pig trial A and B.

Pig Trial A

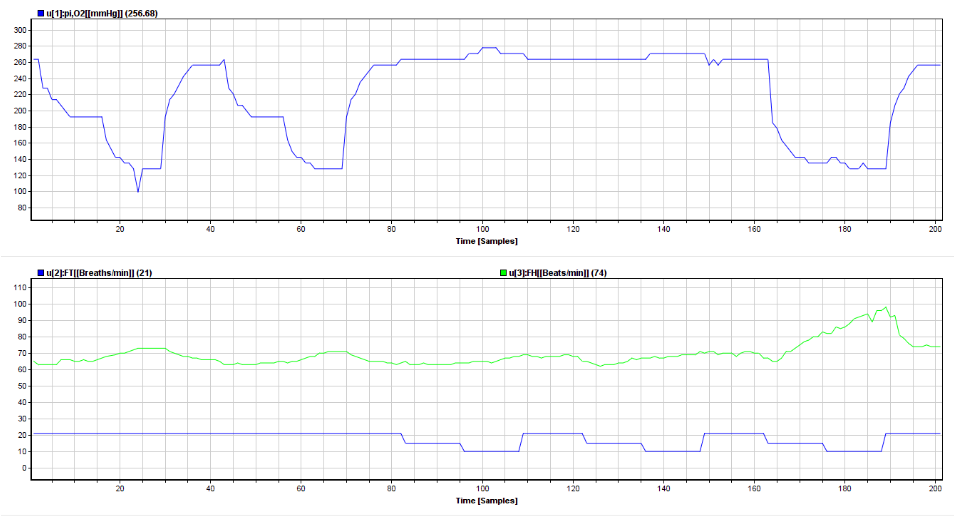


Figure 6.1: Simulated input with data from pig trial A

Figure 6.1 shows the input data displayed graphically for pig trial A. The first plot of the figure shows the input level of oxygen represented as a partial pressure, with the x-axis being the number of samples. The second plot shows the heart beat in green and the breathing frequency in blue. From the figure it is clear that the input level of oxygen is changed first, with the pulse going up as the oxygen level goes down and down as the oxygen level goes up, as expected. Then, the input oxygen level is kept constant as the breathing frequency is varied. This also affects the pulse, though in a smaller scale than the oxygen input. For the last samples both the oxygen level and breathing frequency are varied, resulting in a spike in the heart rate.

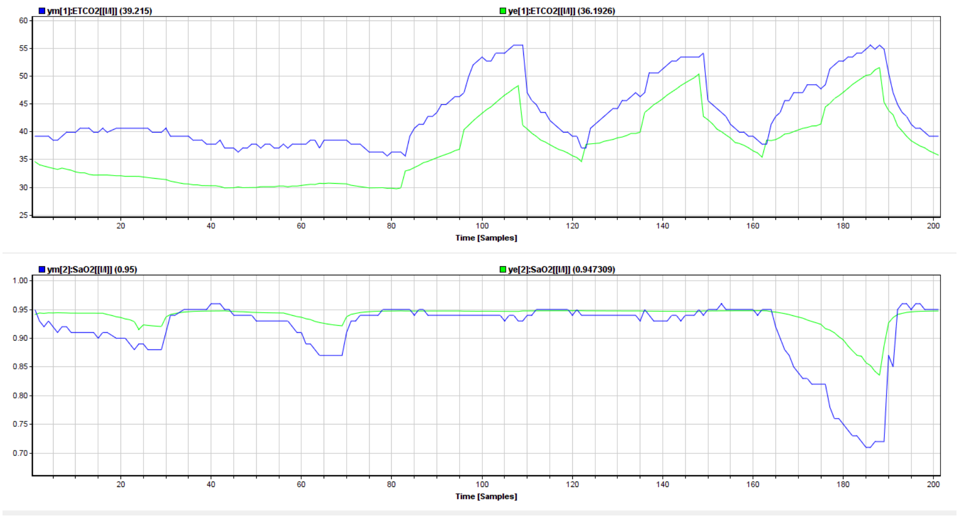


Figure 6.2: Simulated system output with data from pig trial A

Figure 6.2 shows the output of the simulations before parameter estimation of the model with input data from pig trial A. The first plot shows the $ETCO_2$ with the measured data in blue and the estimated in green, while the second plot shows the SaO_2 with the measured data in blue and estimated in green. For $ETCO_2$ it is clear that the curve with the estimated data follows the variations in the measured data curve quite well, but appears to be constantly lower than the measured curve. While for SaO_2 the estimated curve seems to have the same values as the measured one when there are no variation in the curve, but only varies slightly where the measured curve varies.

Pig Trial B

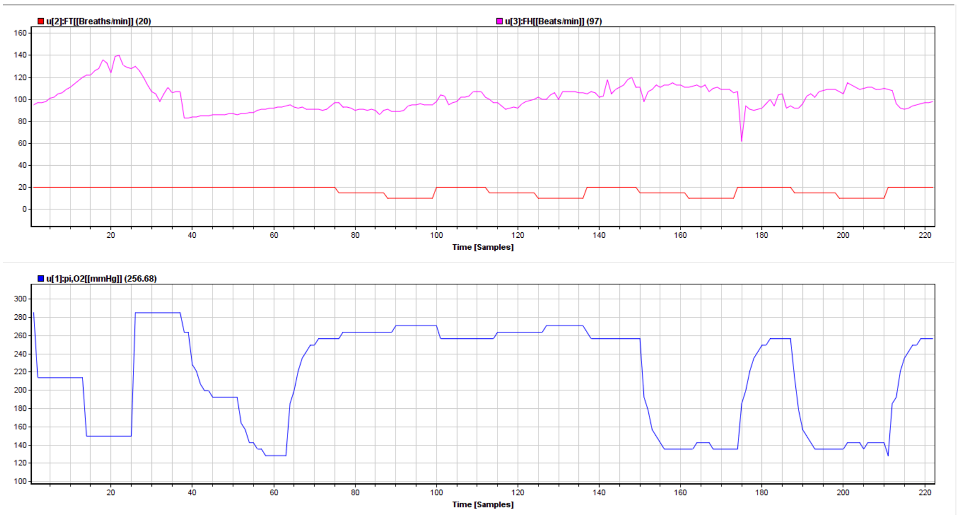


Figure 6.3: Simulated input with data from pig trial B

Figure 6.3 the breathing (red) and heart frequency (pink) in the first graph, and the partial pressure of O_2 in the first graph. As seen from the figure, first the partial pressure of O_2 was varied, then the breathing frequency, while at the end of the trial both were varied. The heart rate is not controllable, even though it is an input variable in the model. Thus, it simply varies naturally due to changes in the oxygen level or breathing frequency.

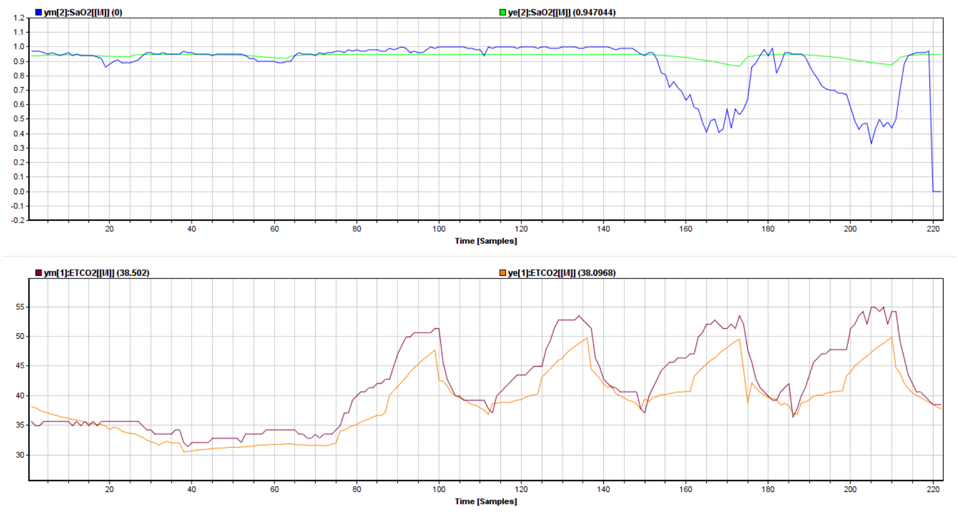


Figure 6.4: Simulated system output with data from pig trial B

As seen from Figure 6.4, the estimated oxygen saturation does not follow the measured oxygen saturation once it varies (the first plot). The estimated SaO_2 varies slightly when the measured SaO_2 does, but not nearly enough according to what would be satisfactory.

While for $ETCO_2$ it is clear that although the estimated values follow the variations of the measured values, it does not follow perfectly.

In other words, parameter estimation is needed in order to get the estimated values to follow the measured values better and more smoothly.

6.1.2 Offline Parameter Estimation

The results of the offline parameter estimation will be presented both graphically from the graphs in ModelFit (by running the Estimator after the parameter estimation) and in a table with the estimated values. Both the parameter estimations from pig trial A and B will be presented. As pig trial B is the newest and varies both breathing frequency and oxygen level simultaneously, and thus represent the real case the most, this will be discussed in more detail and with more plots than pig trial A.

For the offline parameter estimation, Maria V. Ottermo suggested that the production of CO_2 (MT_{CO_2}) and the volume of CO_2 in the tissue (V_{T,CO_2}) were the most important parameters to estimate as they influence the CO_2 level the most. It is very dangerous for a patient to get too much CO_2 in the body, as this will lead to a congestion of CO_2 which may lead to the patient choking. Therefore, the CO_2 level is the most important to estimate correctly. In addition, the volume of O_2 in the tissue (V_{T,O_2}) should be estimated if possible along with the consumption of O_2 (MT_{O_2}). Thus, these are the focus parameters of the offline parameter estimation.

Pig Trial A

The obtained values from the parameter estimation is presented in Table 6.1. The initial values are presented in the "Init." column, and the result from the parameter estimation is given in "Run X". Only the parameters with a value was estimated in this run.

Parameter Values, Pig Trial A					
Parameter	Init.	Run 1	Run 2	Run 3	Run 4
MT_{CO_2}	0.22	0.2557	-	-	0.2557
MT_{O_2}	0.29	-	0.3849	0.3212	0.3849
V_{T,CO_2}	15	14.4696	7.3313	-	14.4696
V_{T,O_2}	6	-	-	1.305022	-
V_{A,CO_2}	3.2	-	-	-	-
V_{A,O_2}	2.5	-	-	-	-
SV	0.05	-	-	-	-
V_{eff}	0.3	-	-	-	-

Table 6.1: Offline parameter estimation values, pig trial A

As seen from Table 6.1, the value for MT_{CO_2} is the same both time it is estimated, while the value for V_{T,CO_2} is nearly halved when estimating it along with MT_{O_2} instead of MT$_{CO_2}$. Thus, this should be represented graphically in order to evaluate which value that yields the best model performance. The value for MT$_{O_2}$ does not change significantly during the runs. Though ModelFit crashed or calculated highly unlikely values when trying to estimate more than three parameters, it is interesting to note that it appears to have estimated three parameters with likely values.

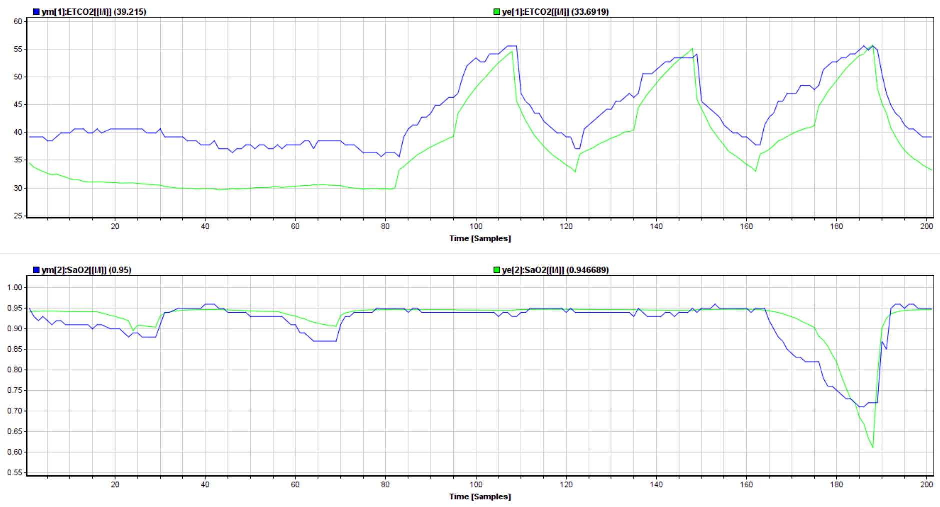


Figure 6.5: Simulated system output for pig trial A with parameter estimation performed on MT_{O_2} and V_{T,CO_2}, with parameters equal to Run 2

In Figure 6.5 the first plot shows ETCO$_2$ with the measured curve in blue and the estimated in green, while for the second plot the color codes are the same but this plot shows SaO$_2$. For the first plot it is clear that the estimated curve is still low compared to the measured one, though the spikes now reaches the measured spikes. For the second plot the estimated curve fits better, though it does not appear to vary enough for the first two small variations, but over estimated at the large drop at the end. Thus, it may appear that neither of the estimated values for the parameters in Run 2 are optimal.

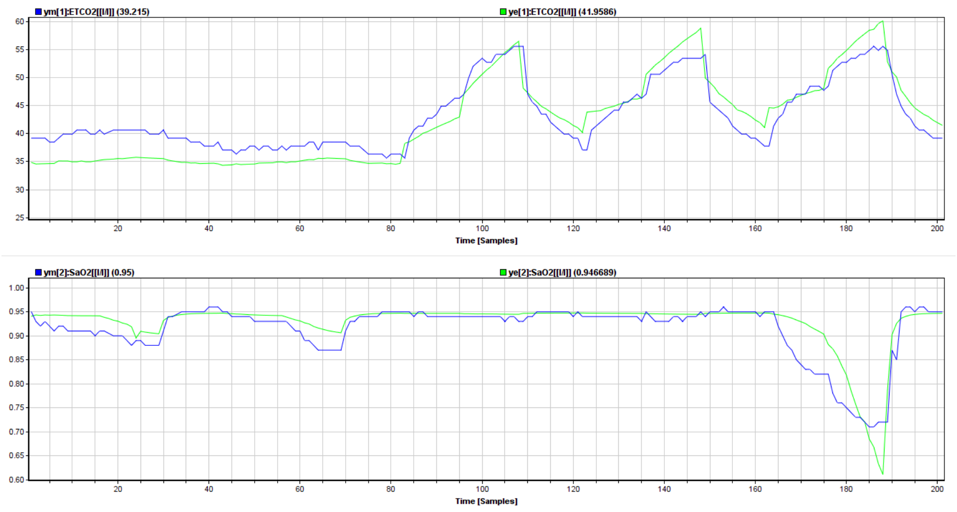


Figure 6.6: Simulated system output for pig trial A with parameter estimation performed on MT_{CO_2}, MT_{O_2} and V_{T,CO_2}, with values equal to Run 4

As Run 4 had the same values as Run 1 for MT$_{CO_2}$ and V_{T,CO_2}, only the simulated model with the results from Run 4 will be presented. This is shown in Figure 6.6, where the first plot shows ETCO$_2$ with blue being the measured curve and green being the estimated, and the second plot shows SaO$_2$ and the same color codes apply. In the first plot the estimated curve follows the measured curve much better than in Figure 6.5. As for Run 3 there appears to be an offset from sample 0 through about 85. However, from the figure it looks to be smaller than for Run 3. The estimated curve appears to fit the spikes much better, and even though it appears to overfit the second and third spikes, it looks to be fitted much better overall. For the second plot, this looks exactly like in Figure 6.5, which is unsurprising as the only parameter affecting the O$_2$ has the same value.

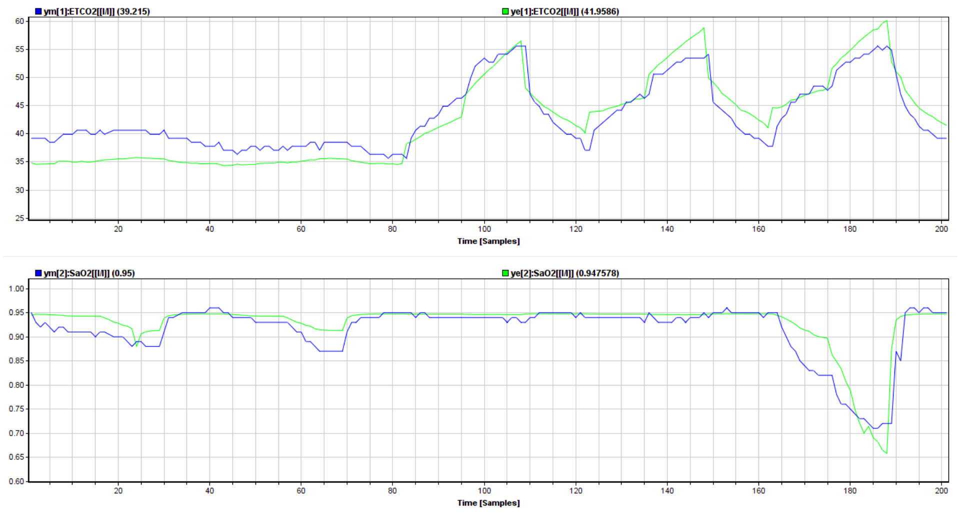


Figure 6.7: Simulated system output for pig trial A with parameter estimation performed on MT_{CO_2}, MT_{O_2}, V_{T,CO_2} and V_{O_2}, with values for MT_{CO_2} and V_{T,CO_2} equal to Run 1 and values for MT_{O_2} and V_{O_2} equal to Run 3

Finally, the plots with all the optimal values for the parameter estimations is presented in Figure 6.7. ETCO$_2$ is in the first plot and SaO$_2$ in the second, and color codes are the same as the previous figures. By comparison, the values from Run 1 was decidedly the optimal value for MT$_{CO_2}$ compared to Run 2, and thus the first plot is as discussed in Figure 6.6. For the second plot the values from Run 3 proved to be the optimal values, though the estimated curve still does not follow the measured curve perfectly. Thus, it appears that performing online parameter estimation may be advantageous.

Pig Trial B

The offline parameter estimation for pig trial B is presented in Table 6.2. As for pig trial A, "Init." is the initial values the parameters had before the parameter estimation, and "Run X" is the values the parameters had after running the offline parameters estimation with these parameters. "-" means that this parameter was not estimated in this run.

Parameter Values, Pig Trial B					
Parameter	Init.	Run 1	Run 2	Run 3	Run 4
MT_{CO_2}	0.22	0.2359	-	-	0.2359
MT_{O_2}	0.29	-	0.4990	0.4151	0.9882
V_{T,CO_2}	15	10.3848	7.2465	-	10.3853
V_{T,O_2}	6	-	-	1.2750	-
V_{A,CO_2}	3.2	-	-	-	-
V_{A,O_2}	2.5	-	-	-	-
SV	0.05	-	-	-	-
V_{eff}	0.3	-	-	-	-

Table 6.2: Offline parameter estimation values, pig trial B

When trying to estimate more than two parameters ModelFit either presented an error message, ran an infinite number of iterations until stopped manually where the parameters went towards large negative numbers, or gave unlikely results such as MT_{O_2} in run 4, which appears to be unreasonably large. This may due to the observability of the model, and suggests that this should be inquired. The values presented in Table 6.2 from the parameter estimation will be discussed along with the plots.

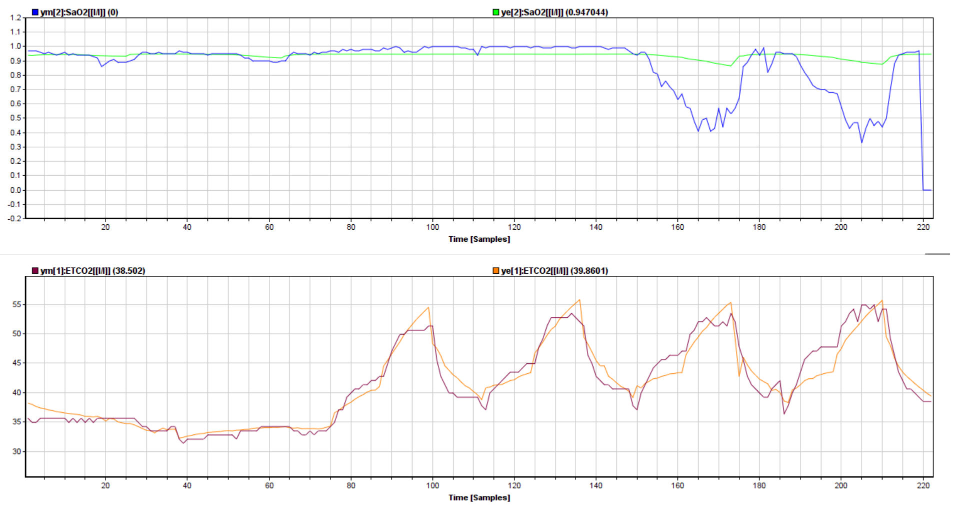


Figure 6.8: Simulated system output for pig trial B with parameter estimation performed on MT_{CO_2} and V_{T,CO_2}, with parameter values equal to Run 1

In Figure 6.8 the first plot is the measured values of SaO_2 in blue and estimated values in green, and the second plot shows the measured values of $ETCO_2$ in red and estimated values in orange. From the figure it is clear that the estimated curve for $ETCO_2$ is following the measured values with great improvement comparing with the simulations in Figure 6.4. This indicated that there are no improvement in the estimated curve for SaO_2, though this is expected as none of the estimated parameters affect the O_2 in the model. Thus, it may be useful to perform and use the estimated parameters affecting the O_2 as well.

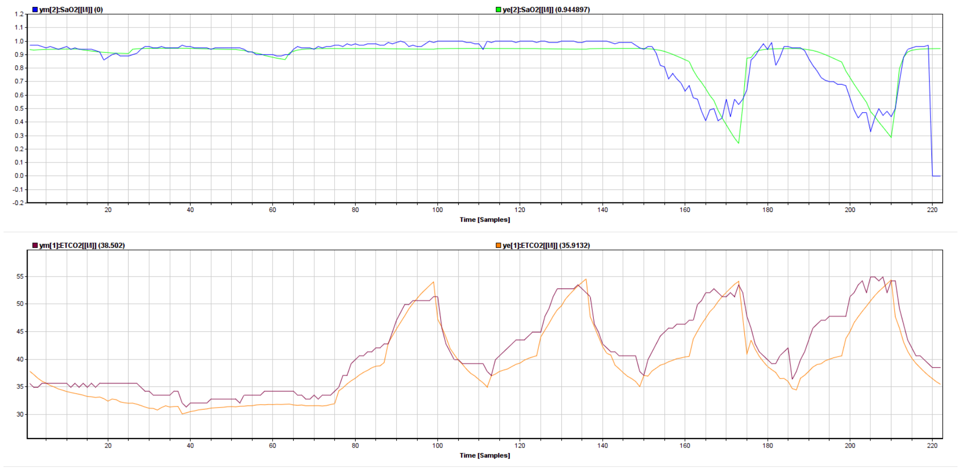


Figure 6.9: Simulated system output for pig trial B with parameter estimation performed on MT_{O_2} and V_{T,CO_2}, with parameter values equal to Run 2

In Figure 6.9 the first plot shows the measured values of SaO_2 in blue and estimated values in green, while the second plot shows the measured values of $ETCO_2$ in red and the estimated values in pink. In the plot for SaO_2 it is clear that the model over estimates at big changes in the curve, which indicates that the estimated value for MT_{O_2} may be too large. In the second plot, the curve for the estimated values of $ETCO_2$ follows much better than in the simulated plot before the parameter estimation in Figure 6.4. This indicated that the estimated value for V_{T,CO_2} gives a better result than the initial value.

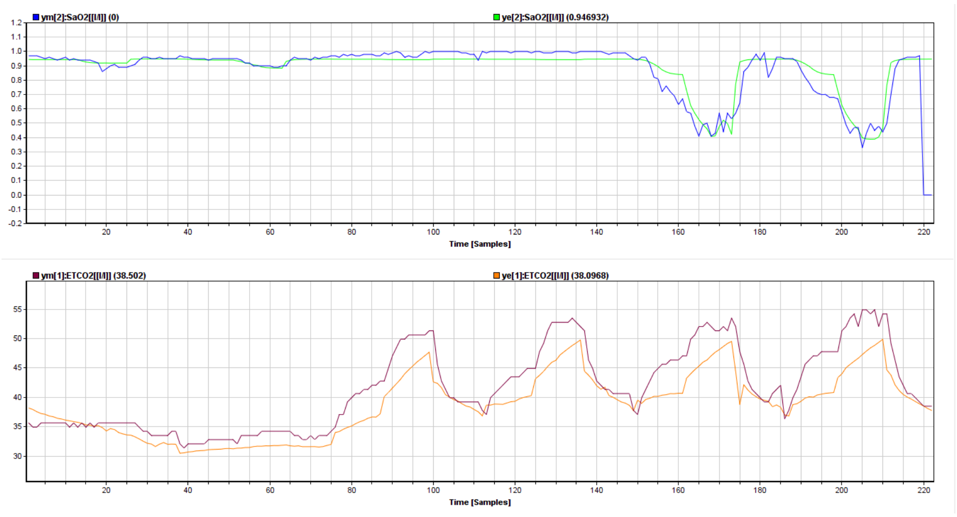


Figure 6.10: Simulated system output for pig trial B with parameter estimation performed on MT_{O_2} and V_{T,O_2}, with parameter values equal to Run 3

As for the previous plots, in Figure 6.10 the first plot shows the measured values of SaO_2 in blue and estimated values in green, while the second plot shows the measured values of $ETCO_2$ in red and the estimated values in orange. Not surprisingly, the figure shows that the plot for $ETCO_2$ is almost identical as the simulation in Figure 6.4 as there are no estimated parameters affecting the CO_2. While for SaO_2 the estimated curve follows the measured curve much better than before the parameter estimation. This indicated that using the estimated parameters for MT_{O_2} and V_{T,O_2} will yield an improved model behaviour, rather than simply using the estimated parameters for MT_{CO_2} and V_{T,CO_2} as first assumed.

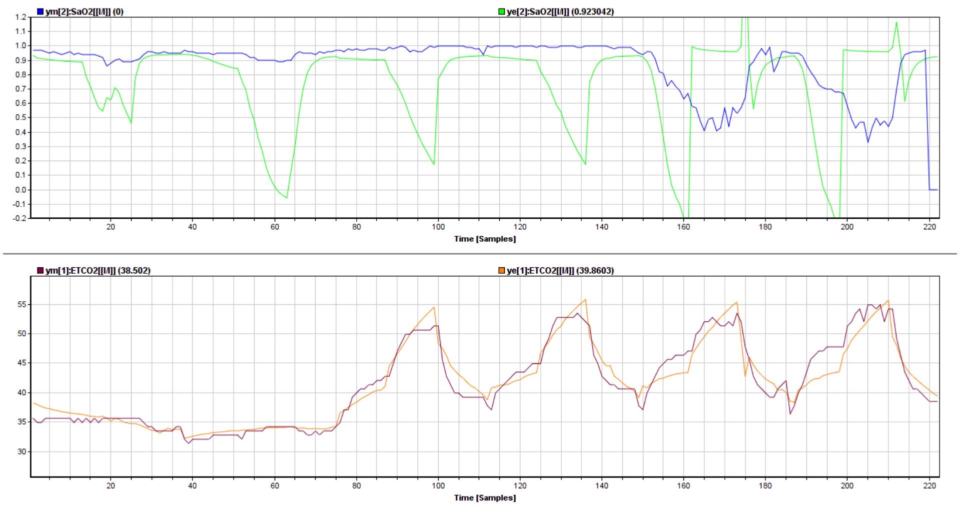


Figure 6.11: Simulated system output for pig trial B with parameter estimation performed on MT_{CO_2}, MT_{O_2} and V_{T,CO_2}, with parameter values equal to Run 4

The plots and colors are the same as the previous figures. As previously mentioned is the value of MT_{O_2} clearly unrealistically large, as can be seen in the first plot in Figure 6.11. This makes the curve unstable and the estimated curve for SaO_2 is oscillating rather than following the curve for the measured values. While for the second plot, the values affecting the CO_2 are the same as in Figure 6.8, and thus the plot is the same as well.

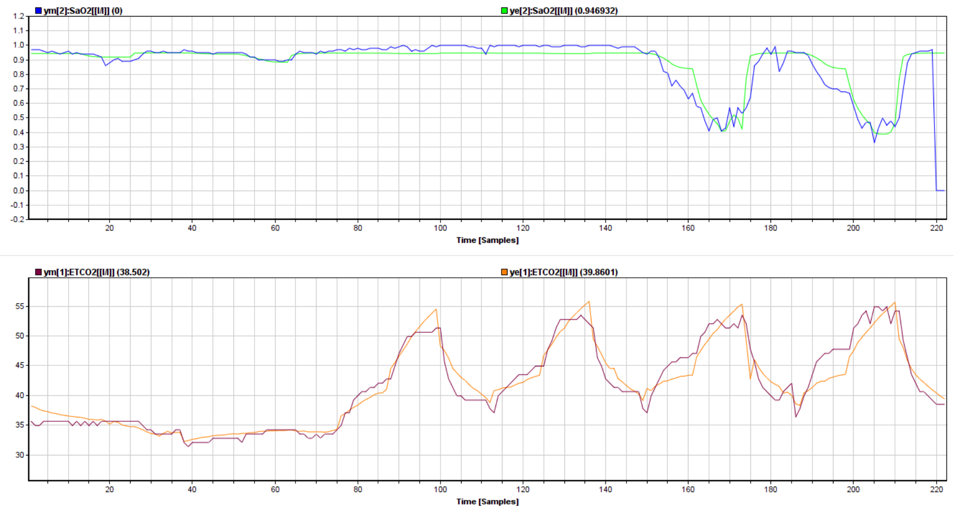


Figure 6.12: Simulated system output for pig trial B with parameter estimation performed on MT_{CO_2}, MT_{O_2}, V_{T,CO_2} and V_{T,O_2}. The parameter values for MT_{CO_2} and V_{T,CO_2} equal to Run 1 and the parameter values for MT_{O_2} and V_{T,O_2} being equal to Run 3

Figure 6.12 shows the model simulated with all the optimal values from the offline parameter estimation. As seen from the figure, the overall performance of the system is greatly improved when using the updated parameter values. However, the curves are not as smooth as desired for the MPC, and thus it is necessary to perform online parameter estimation in order to get the wanted result.

General Discussion

The first important thing to notice is that all the estimated values appear to be reasonable and likely. This indicates that the mathematical model is an improvement compared to the one used in Malde [14], at least as long as only two parameters are estimated at the time.

There is a big difference in the estimated value for V_{T,CO_2} in the different runs, especially with the data from pig trial A. This may be due to the fact that when using the data from pig trial A there was a general offset in the estimated curve for ETCO$_2$, as seen from Figure 6.2, and when it is estimated along MT$_{O_2}$ it is the only parameter affecting the CO$_2$ level, and thus compensate for this. In other words, V_{T,CO_2} should be estimated with MT$_{CO_2}$ and V_{T,O_2} with MT$_{O_2}$ in future trials. In addition, estimating more than two variables at the time should be avoided, as this may lead to unlikely parameter values as in trial B. This argument is also supported considering that the data from trial B is the most realistic.

The estimated parameter values differ significantly when using data from trial A and B. It can can thus be argued that offline parameter estimation should be performed for each trial/patient, despite Malde [14] concluding otherwise. However, as the two datasets differ quite a lot and neither represent data from a human trial it is difficult to conclude absolute without further trials. Though, as the offline parameter estimation does not take long to perform it can be argued that it should be performed every time either way.

6.1.3 Online Parameter Estimation

The online parameter estimation is tuning of the standard deviations of the process and measurement noise until the model has the desired behaviour, as defined in Chapter 2, i.e. that the estimated parameters and outputs have a smooth curve as this is optimal behaviour for the MPC. The online parameter estimation will be performed using data from both pig trial A and B. As Malde [14] concluded that the online parameter estimation yields better results when the offline parameter estimation is done beforehand, the online parameter estimation will be ran using the updated parameters from the offline parameter estimation. In addition, this was an expected result as the model should have improved behaviour when adjusted for each individual. The results will be presented with a table presenting the standard deviations of the parameters and output variables after the online parameter estimation, along with a graphic representation of the simulated model with applied noise and a recursive filter. As it is only expedient to estimate as many parameters as measurements in a recursive estimator, only two parameters will be estimated at the time. As for the offline parameter estimation, MT$_{CO_2}$, MT$_{O_2}$, V$_{T,CO_2}$ and V$_{T,O_2}$ are assumed the most important parameters to estimate. Hence, these have been the focus parameters of the online parameter estimation.

In trials A and B, a capnograph was used to measure the $ETCO_2$ and SaO_2. The instrument measures the end tidal CO_2 in the throat of the pig, it is likely that the measurement will have some noise, though not a lot. As the initial value of SaO_2 is 0.97 while it is 40 for $ETCO_2$, it is likely that more noise is required for the measured $ETCO_2$ in order to make the same impact. In other words, the measurement noise for $ETCO_2$ is expected to have a higher standard deviation than the measurement noise for SaO_2. The default value of the standard deviations is 0.0001.

Pig Trial A

In the figure below the plots of the parameters before tuning are presented.

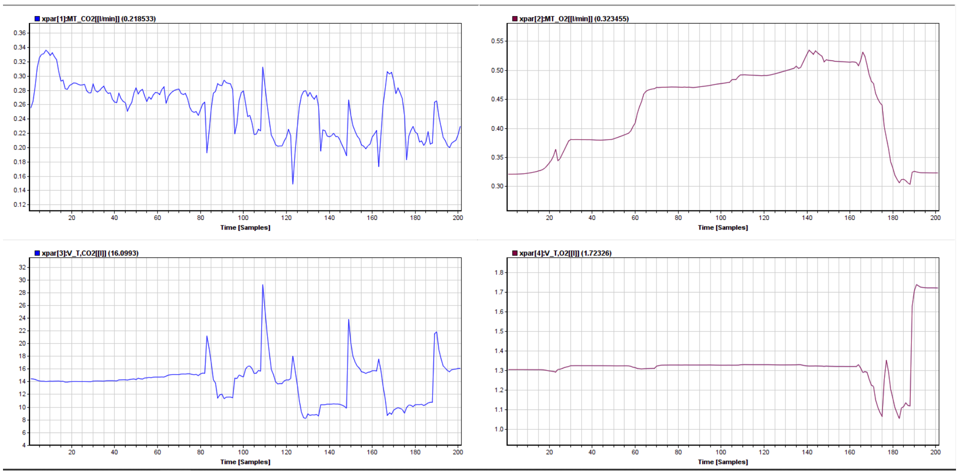


Figure 6.13: Simulated system with parameters that are to be estimated online, with data from pig trial A

In Figure 6.13 the curve for MT_{CO_2} is in the top left corner, V_{T,CO_2} is in the bottom left corner, MT_{O_2} is in the top right corner and V_{T,O_2} is in the bottom right corner. From the plots it is plain that the plots vary much more than the desired behaviour required for an optimal performance of the MPC. Especially the plots associated with CO_2. It is therefore necessary to tune the parameters to obtain the wanted behaviour.

In Table 6.3 the standard deviations acquired after tuning are presented, in addition to the measurement noise used. It appears that in order to achieve the same level of variation between the measured and estimated output curve, a smaller standard deviation of the measurement noise was required. Thus, the values for this is smaller than for pig trial B. This may be due to the fact that the dataset for pig trial A varies less and is less complex than for pig trial B.

Standard deviation, process noise, pig trial A	
Parameter	Standard deviation
MT_{CO_2}	0.0006
MT_{O_2}	0.0003
V_{T,CO_2}	0.0007
V_{T,O_2}	0.0002
V_{A,CO_2}	0.0008
V_{A,O_2}	0.0003
SV	0.001
V_{eff}	0.001
Standard deviation, measurement noise, pig trial A	
Measurement	Standard deviation
$ETCO_2$	0.02
SaO_2	0.001

Table 6.3: Online parameter estimation values, pig trial A

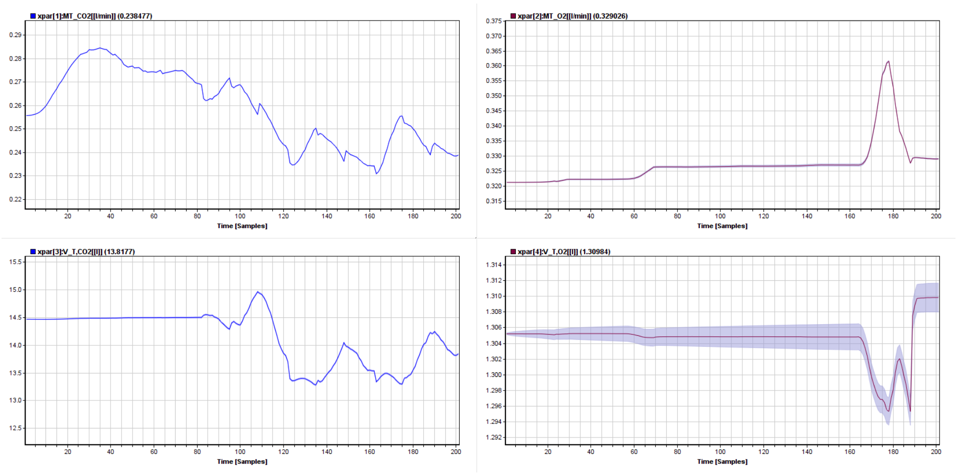


Figure 6.14: Simulated system with online estimated parameters, with data from pig trial A

The estimated parameters after tuning are shown in Figure 6.14. By comparison, the tuned parameters vary much more smoothly than before the tuning. Though, in the plot for V$_{T,O_2}$ it appears that it may be too much noise. However, as the curve varies so little, it is considered that the performance is good enough.

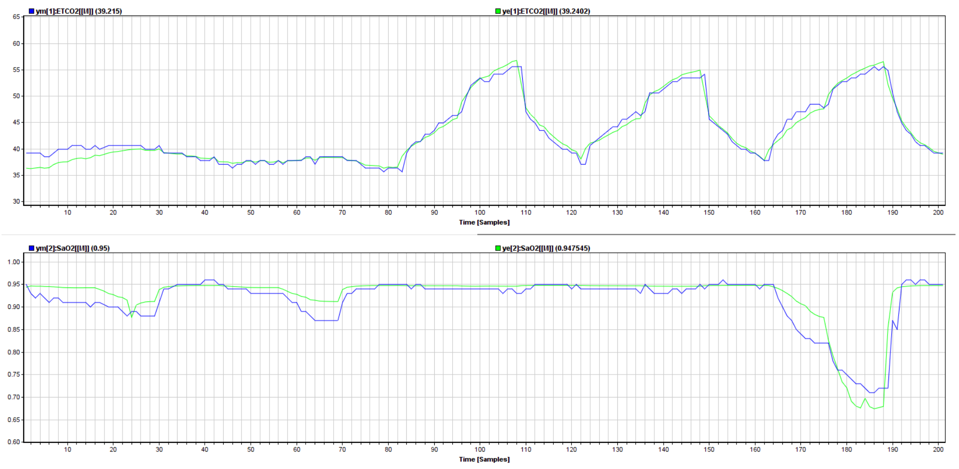


Figure 6.15: Simulated system output after tuning, with data from pig trial A

In Figure 6.15 the plots for measured (blue) and estimated (green) $ETCO_2$ are presented at the top, and the plots for measured (blue) and estimated (green) SaO_2 are presented at the bottom. From the figure it looks like the estimated curve for $ETCO_2$ fits the measured curve too well compared to what is realistic. Although, when trying to apply a higher standard deviation to the measurement noise, the parameters became increasingly unstable and the estimated curve for SaO_2 appeared to over estimate the "drop" even more. Thus, the values presented in Table 6.3 were used.

Pig Trial B

First, the parameters before tuning are presented graphically, as seen in the figures below.

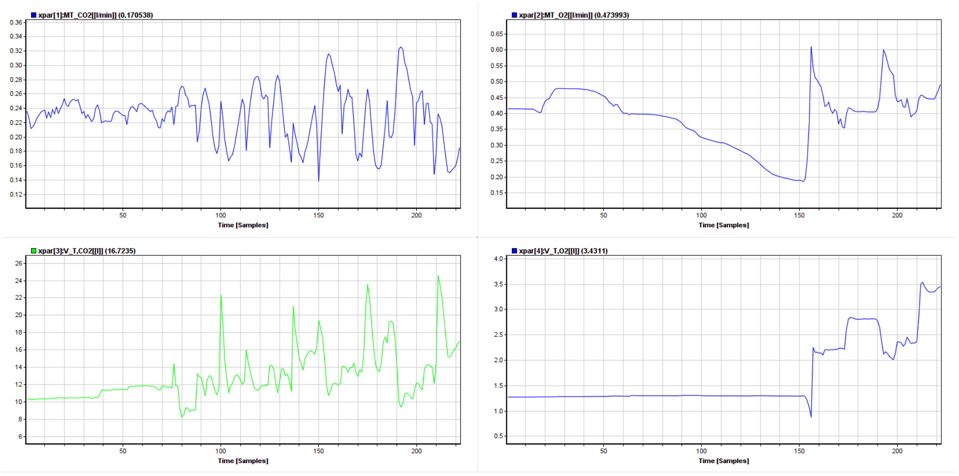


Figure 6.16: Simulated system with parameters that are to be estimated online, with data from pig trial B

In Figure 6.16 MT$_{CO_2}$ is shown in the top left corner, MT$_{O_2}$ in the top right corner, V$_{T,CO_2}$ in the bottom left corner and V$_{T,O_2}$ in the bottom right corner. From the figure it is clear that the parameters do not vary as smoothly and evenly as desired in order for optimal performance of the MPC for pig trial B either.

In Table 6.4 the standard deviations of process and measurement noise after tuning are presented.

Standard deviation, process noise, pig trial B	
Parameter	Standard deviation
MT_{CO_2}	0.001
MT_{O_2}	0.0008
V_{T,CO_2}	0.0005
V_{T,O_2}	0.0004
V_{A,CO_2}	0.0003
V_{A,O_2}	0.0006
SV	0.009
V_{eff}	0.003
Standard deviation, measurement noise, pig trial B	
Measurement	Standard deviation
$ETCO_2$	0.0375
SaO_2	0.001

Table 6.4: Online parameter estimation values, pig trial B

By trial and error, the standard deviations presented in Table 6.4 were achieved. This gave the following plots for the estimated parameters:

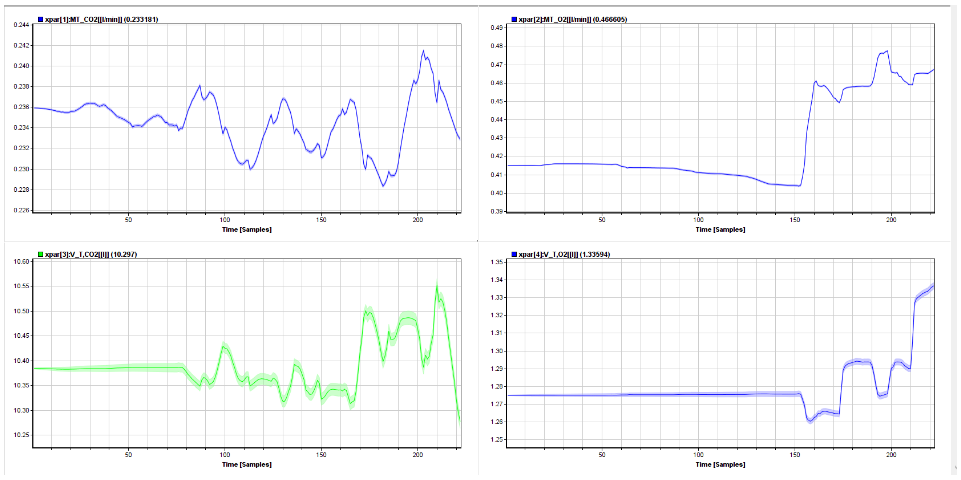


Figure 6.17: Simulated system with the estimated parameters, with data from pig B

Figure 6.17 shows the parameters after tuning. When comparing the results to that of Figure 6.16 it appears that the tuning has been successful, as the parameters seemingly vary more smoothly. There are still some "sharp edges". However, this may be due to the fact that there are no time delays implemented in the model, and thus the model will have much quicker system dynamics than the actual system. As the tuning relies on the standard deviations of both the process and measurement noise, there are many different combinations that will yield an improved result. Therefore, it is impossible to conclude that this is the best result, so it can only be deemed good enough.

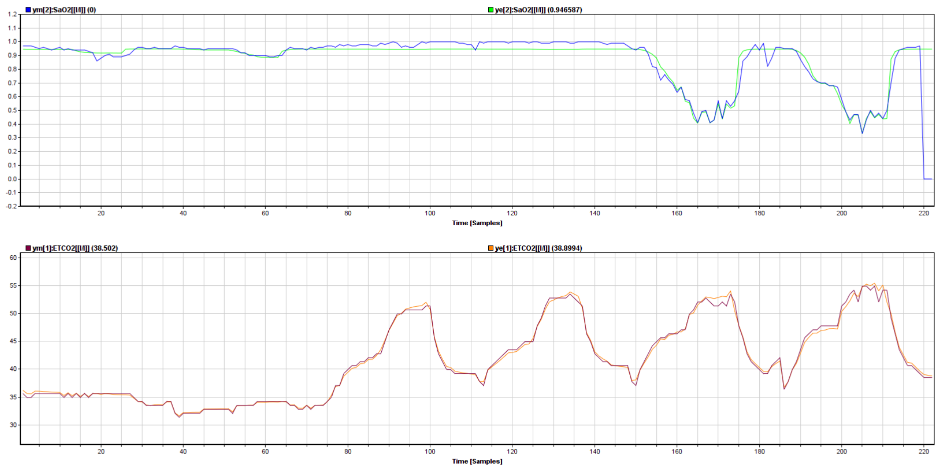


Figure 6.18: Simulated system output after tuning with data from pig trial B

From Figure 6.18 it appears that the estimated curves follow the measured ones very well. When first applying the recursive filter, the estimated curve for $ETCO_2$ followed the measured curve perfectly. This is completely unrealistic, and measurement noise was added. From the figure, it can be argued that the estimated curves still follows the measured curves too well compared to how realistic it is. However, as the parameters proved harder to estimate with more measurement noise, the added standard deviations of the measurement noise were deemed enough.

General Discussion

By comparing the simulations of the parameters before any tuning for pig trial A and B, the parameters for pig trial B seem to be varying more than for pig trial A. This is logical as the dataset from pig trial B has more variation, i.e. reflects reality better. Thus, in order to achieve the desired behaviour for improved performance of the MPC, higher standard deviations for measurement and process noise are required for pig trial B than A.

When running the simulations with a recursive filter before tuning, the $ETCO_2$ is estimated perfectly. This may be due to the fact that $ETCO_2$ is equal to a state. When applying a recursive filter the measurement from the previous time step may be utilized, and an almost perfect value can be applied to the estimated output. However, this is a very unrealistic result. Hence, it is important to add measurement noise, and especially to $ETCO_2$. This is also probably why the model performs well with more measurement noise on $ETCO_2$ than SaO_2, in addition to having a lower a lower accuracy of measurements.

For both trials, there are still some "sharp edges" even after the tuning. A reason for this may be that the model dynamics changes faster than the plant dynamics. This implies that it can be beneficial to implement the model with time delay to better represent reality.

Despite the differences in the datasets, the results of the online parameter estimation are similar enough that the tuning process may go faster the next time. Although, two datasets are not enough to establish that there is indeed a trend.

There are not much literature on the expected standard deviations for the measurement or process noise for this model. Hence, the conclusions are only based on the results presented in this report. In order to reach a solid conclusion, the model should be tested with datasets that better represent the real situation, i.e. with both varying oxygen input and breathing frequency simultaneously, and with more datasets.

6.2 Result of System Tests

Throughout the report several system tests have been defined. The results are presented in this section. Not all the specified tests are addressed, as some tests are necessary for the system to work, and therefore will be fulfilled as long as the system works. Deviations and failed tests will be addressed.

6.2.1 Unit Tests

In this section the result of the unit tests defined in Chapter 4 will be presented. As most of the tests have a binary result (either the test was passed or it was not), the results are accordingly.

After running the system with the suitcase from Inventas and with test data from the capnograph, the following results were achieved for the unit tests:

- LabView: The data is displayed as expected, and the files are saved in the expected folder.
- MATLAB: All functions yield the expected results when tested separately. The module works well as a whole, and all calculations yield the correct values.
- ModelFit: From the results in Section 6.1 it is clear that the estimated parameters from the online and offline parameter estimations are as expected, and likewise for the system simulations.

6.2.2 Integration Tests

The integration tests are how well the different modules of the program communicate. By running the program, all the integration tests were tried.

- The MATLAB program works well with the LabView program.
- The MATLAB program works ok with ModelFit. As the capnograph and oxyometer have different sample rates, it is challenging to get the right time for both files. Two solutions have been proposed: ensuring that there are only unique samples in a file, and collecting every tenth sample from the oxyometer, as the oxyometer has a frequency of 10Hz and the capnograph of 1Hz. Both were implemented and tested, and it was concluded that the solution with unique samples worked best. This is due to that with the seconds solution there are no guarantees for when the sampling starts, and thus an offset in time may occur between the information from the oxyometer and the capnograph. Neither are deemed optimal solutions.
- Otherwise all modules work well together.

6.2.3 System Tests

The system tests that were developed during the design phase have been executed and the results along with a brief description of the tests are presented in section.

Performance

The IDP does what it intends to; it logs data from a pig/patient, formats the data in a form that can be used by ModelFit, estimates parameters online and offline and safely transfers the obtained values to the MPC.

However, as the system is now, all relevant data is saved to an excel-file on a format that enables quick copy and paste to ModelFit, but, according to Kasper Linnestad from Cybernetica, it may also be a solution to save the data to a .mat and load the .mat file directly in ModelFit. This solution was explored, but as the format the .mat file demanded was quite complex, the current solution was deemed good enough.

Effectiveness

The time difference between doing everything the IDP does manually and the IDP was tested. As the available time in a human trial is limited, it is important that the IDP is ran as effectively as possible.

In this test, only the file formatting module was run, as this is where the main difference in effectiveness between the old and new system lies. All conversion formulas were known beforehand, and "Remove duplicates" in excel was used in order to only get one sample per second. In this test the data was only logged for 47 seconds. This time is not accounted for in the times presented. The difference in the amount of data between this test and a real trial would only result in approximately a total of 20 seconds longer for the manual process.

- Time, manual procedure: 17min 43sec
- Time, identification process: 58sec

Hence, it is clear that the identification process takes significantly shorter time than the manual procedure in order to do the same job. It is also a lot more complicated to do the manual procedure as it demands prior knowledge of conversion formulas and where to find different functions in excel (for instance how to split cells, go from decimal number to time, copy the values from formulas not the formula itself, etc.).

6.3 Evaluation of IDP with regard to Function Specifications

Chapter 3 concluded with a set of function specifications being defined. In this section the IDP will be evaluated with regard to these specifications.

- The process converts the data files from the LabView program into a form that can be used.
- As discussed in Section 6.2.2 the process only keeps one sample per second. This ensures that the different instruments match each other, as well as keeping the data size down.
- The process does not automatically insert the formatted data in ModelFit. The most automatic procedure available was making a .mat file with the input data. However, this was deemed too much work with the time available compared to the profit gained. Instead, a file is created with the data presented in such a way that it is very easy to simply copy and paste it into ModelFit.
- The IDP takes less than 40 minutes, given that the logging only takes 30 minutes.
- By saving the estimated parameters from ModelFit, correct transfer of parameters and noise is ensured.

- There has not been implemented a safety mechanism to ensure likely and realistic estimation of parameters. Although, by only estimated two parameters at the time and knowing what regions these should be within should ensure this anyway.
- According to Kasper Linnestad the parameters are not transferred to the MPC autonomously, and did not suggest that there exist a solution that enables automatic transfer. Therefore, the current solution for transfer is as good as it gets with Cybernetica's programs.
- The estimated parameters are presented nicely in ModelFit. The user must accept the update of the parameters from the offline parameter estimation. The user decides when the parameters are tuned well enough in the online estimation. Hence, the safety is ensured.
- All manual steps have been thoroughly explained in Chapter 4.
- The mathematical model has been implemented, and is present when opening the parameter estimation program.
- Both online and offline parameter estimations are performed.
- The estimated parameters appear to be realistic and probable.

Chapter 7

Conclusion and Future Work

In this chapter the conclusion will be presented, along with recommendations for future work.

7.1 Conclusion

The identification process is one of the processes that must be performed before closed loop control. It is an important process as it estimates the optimal parameters and tunes the filter used in the MPC in closed loop. With the basis in a thorough design process, an identification process to be used in pig and human trials for the OxyAid project has been implemented in this project.

In the planned human trials, limited time is available. As much of the available time as possible will go to the closed loop regulation of oxygen for the COPD patients. Thus, it is important that other processes take as little time as possible. When testing the new IDP against the old, the results showed that the new process took about 17 minutes less than the old. This is a significant difference, and proves the new process to be more effective than the old.

The process is not as autonomous as hoped beforehand, though this is mainly due to limitations of the used programs. Thus, one option may be to use a different tool for the parameter estimation such as MATLAB, in order to get an even more effective process. However, Cybernetica's tools work very well, so despite the number of manual steps a different program should perform to the same standards before considering a change.

In order to perform the parameter estimation of the identification process a mathematical model developed by Maria V. Ottermo and Øyvind Stavdahl was implemented. The model showed good performance when simulated, and the estimated parameter values were realistic. This indicated that the mathematical model is an improvement from the one previously used, as it did not give realistic values.

From the offline parameter estimation it is clear that two and two parameters should be estimated together, and the two parameters regarding O_2 and the two regarding CO_2, respectively, are the ideal pairs in order to achieve the best results. Also, offline parameter estimation should be performed for each trial as it does not take long and yields the best overall performance.

When adding a recursive filter to the model it was obvious that measurement noise had to be added as the model predicted the output perfectly without noise, which is a very unlikely result. From the two datasets that were tried, it appeared that the more variation in the dataset the higher standard deviations for measurement noise was needed. Though, this should be tested on more datasets to see if this is indeed a trend. For both trials, there were still some "sharp edges" after tuning. This may be due to the fact that the model dynamics change faster than the plant dynamics, and thus adding time delay to the model may prove beneficial.

In conclusion, the IDP designed and implemented in this project is more effective than previous solutions. In addition, the implemented model appears to yield good results from the parameter estimation, although more datasets should be utilized in order to verify the observed trends. Despite good performance of the process, there are still measures that can be done to automate the process even further.

7.2 Future Work

In order to better evaluate the new IDP solution, there are some measures that should be taken. Firstly, the solution should be tried with more data in order to verify the results. Secondly, the IDP should be tested in trials to see how well it performs there. Thirdly, in order to achieve a more autonomous process it should be considered to implement the data to a .mat file for easier transfer of data to ModelFit. Finally, it may be considered to use a different program for parameter estimation in order to automate the process further. Although, as Cybernetica's programs have very good performance, there may be a trade-off between performance and effectiveness if this solution is tried. It will also demand further research of different toolboxes in order to find one suitable for the project.

Bibliography

[1] E. Balsa-Canto, D. Henriques, A. Gábor, and J. R. Banga. Amigo2 start up guide. https://sites.google.com/site/amigo2toolbox/doc, 2016 (Accessed: 2019.09.21).

[2] J. J. Batzel, F. Kappel, D. Schneditz, and H. T. Tran. *Cardiovascular and Respiratory Systems: Modeling, Analysis and Control*. Society for Industrial and Applied Mathematics, 2006.

[3] L. Blanks and R. Russel. Chronic obstructive pulmonary disease (copd). https://www.bupa.co.uk/health-information/lungs-breathing/copd, 2016 (Accessed: 2019.10.07).

[4] E. Cheever. State space representations of linear physical systems. https://lpsa.swarthmore.edu/Representations/SysRepSS.html, 2019 (Accessed: 2019.10.01).

[5] J. G. Dyrset, K. Linnestad, and P. Singstad. Introduction to cybernetica apc technology. Power point from course held by Cybernetica, 2019.

[6] A. Eldhuset. Dynamisk modell av ph, o2- og co2-metning i blod. Master's thesis, NTNU, 2016.

[7] Global Initiative for Chronic Obstructive Lung Disease Inc. Pocket guide to copd, diagnosis, management, and prevention. https://goldcopd.org/wp-content/uploads/2018/11/GOLD-2019-POCKET-GUIDE-DRAFT-v1.7-14Nov2018-WMS.pdf, 2018 (Accessed: 2019.09.03).

[8] COPD Foundation. What is copd? https://www.copdfoundation.org/What-is-COPD/Understanding-COPD/What-is-COPD.aspx, 2019 (Accessed: 2019.08.24).

[9] K. R. Fowler and C. L. Silver. *Developing and Managing Embedded Systems and Products*. Newnes, 2015.

[10] S. Friisø. Model identification of the human respiratory system using existing matlab toolboxes. Student project, 2019.

[11] Norsk Helseinformatikk. Lungene. https://nhi.no/kroppen-var/organer/lungene/?page=all, 2019 (Accessed: 2019.08.27).

[12] Y. Kim and H. Bang. Introduction to kalman filter and its application. In *Introduction and Implementations of the Kalman Filter*, chapter 2. IntechOpen, 2019.

[13] Y. P. Li. State feedback and observer feedback. www.me.umn.edu/courses/me8281/notes/statebf_abiram.pdf, 2012 (Accessed: 2019.11.10).

[14] E. Malde. Modellprediktiv regulering av oksygenkonsentrasjon hos kolspasienter. Master's thesis, NTNU, 2019.

[15] Malik S. Mathur, S. Advancements in the v-model. *International Journal of Computer Applications*, 1(12), 2010.

[16] D. Q MAyne, J. B Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[17] J. Nocedal and S. J. Wright. *Numerical Optimization, Second Edition*. Springer, 2006.

[18] M. Nørgaard, N. K. Poulsen, and O. Ravn. New developments in state estimation for nonlinear systems. *Automatica*, 36(11):1627–1638, 2000.

[19] World Health Organization. Burden of copd. https://www.who.int/respiratory/copd/burden/en/, 2019 (Accessed: 2019.09.24).

[20] Visual Paradigm. What is sequence diagram? https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/, 2019 (Accessed: 04.12.2019).

[21] P. Patel. Structured analysis and structured design (sa/sd). https://www.geeksforgeeks.org/structured-analysis-and-structured-design-sa-sd/, 2019 (Accessed: 2019.09.26).

[22] The OxyAid Project. Automatisk oksygendosering til kols-pasienter som behandles med langtids oksygenterapi (ltot). For use in OxyAid project, 2017.

[23] Harvard Medical School. Copd. https://lmi.med.harvard.edu/what-copd, 2019 (Accessed: 2019.04.23).

[24] D. D Seborg et al. *Process Dynamics and Control*. John Wiley & Sons, 2010.

[25] Ø. Stavdahl. Strukturert analyse og -konstruksjon [structured analysis and -construction]. Course material, 2001.

[26] Ø. Stavdahl and M. V. Ottermo. Oxyaid compartment model, based on batzel. For use in OxyAid project, 2019.

[27] K. A. Zimmermann. Respiratory system: Facts, function and diseases. https://www.livescience.com/22616-respiratory-system.html, 2018 (Accessed: 2019.11.23).