ISSN 0280-5316
ISRN LUTFD2/TFRT--5908--SE

Speed Control of a Peristaltic Blood Pump

Gabriel Ingesson
Helena Sandberg

Lund University
Department of Automatic Control
January 2013

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
January 2013

Document Number
ISRN LUTFD2/TFRT--5908--SE

Author(s)
Gabriel Ingesson
Helena Sandberg

Supervisor
Anders Wallenborg, Gambro, Sweden
Tore Hägglund, Dept. of Automatic Control, Lund University, Sweden (examiner)

Sponsoring organization

Title and subtitle
Speed control of a peristaltic blood pump (Varvtalsreglering av peristaltisk blodpump)

Abstract
Hemodialysis is today the main treatment used for patients with renal impairment, a serious medical condition. Hemodialysis treatment handles a very delicate system which is why it is of the utmost importance that the dialysis machine has got a reliable safety monitoring system that can detect severe complications during treatment. A new such system is currently under development at Gambro, Lund, which focuses on early detection of venous needle dislodgement (VND). The safety monitoring system determines the heart pulses from the patient on both the venous and arterial side. VND is indicated when no venous heart pulses can be detected. Large pressure disturbances are induced by the rollers on the peristaltic pump which makes it dicult to extract the patient's heart pulses. In order to successfully filter out the large disturbances it is required that the pump has a nearly constant period time. The objective of this thesis was to evaluate the current control system which is a hardware motor speed controller connected in cascade with a software blood flow controller. The project objective was also to find an improved control system for the peristaltic pump which can fulfill the required standard deviation of the relative period time of less than 0.1 %. This was done by first analyzing the pump process in open loop. Results showed that the process had high-frequency disturbances possibly originating from some mechanical unevenness in the motor since the frequency of the disturbances was dependent on the motor speed. It was also found that these disturbances were enhanced by the existing control system. In order to find a new improved control system, a hardware controller similar to the existing controller, was evaluated. With the alternative hardware controller the standard deviations of the relative period times were improved, except at low blood flows (100-150 ml/min). A PI controller was implemented in software using LabView. The PI controller fulfilled the relative period time standard deviation requirement at all blood flows and showed improved performance compared to the original controller. Used together with a feed forward implementation, the performance was further improved.

Keywords

Classification system and/ or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-83

Recipient's notes

Security classification

Acknowledgements

We would like to take this opportunity to direct our special thanks to everyone who helped us during this project. First of all, we would like to thank Anders Wallenborg who has been our dedicated supervisor at Gambro. His knowledge and support has been invaluable in all aspects of this thesis. Our appreciation is also directed to our examiner Tore Hägglund at Lunds Technical University and our external supervisor Ole Ravn at Denmarks Technical University for all their guidance and feedback.

We also owe great thanks to the members of the Treatment System Research group at Gambro for making us feel welcome during our time there.

Abstract

Hemodialysis is today the main treatment used for patients with renal impairment, a serious medical condition. Hemodialysis treatment handles a very delicate system which is why it is of the utmost importance that the dialysis machine has got a reliable safety monitoring system that can detect severe complications during treatment. A new such system is currently under development at Gambro, Lund, which focuses on early detection of venous needle dislodgement (VND). The safety monitoring system determines the heart pulses from the patient on both the venous and arterial side. VND is indicated when no venous heart pulses can be detected. Large pressure disturbances are induced by the rollers on the peristaltic pump which makes it difficult to extract the patient's heart pulses. In order to successfully filter out the large disturbances it is required that the pump has a nearly constant period time.

The objective of this thesis was to evaluate the current control system which is a hardware motor speed controller connected in cascade with a software blood flow controller. The project objective was also to find an improved control system for the peristaltic pump which can fulfill the required standard deviation of the relative period time of less than 0.1 %. This was done by first analyzing the pump process in open loop. Results showed that the process had high-frequency disturbances possibly originating from some mechanical unevenness in the motor since the frequency of the disturbances was dependent on the motor speed. It was also found that these disturbances were enhanced by the existing control system.

In order to find a new improved control system, a hardware controller similar to the existing controller, was evaluated. With the alternative hardware controller the standard deviations of the relative period times were improved, except at low blood flows (100-150 ml/min).

A PI controller was implemented in software using LabView. The PI controller fulfilled the relative period time standard deviation requirement at all blood flows and showed improved performance compared to the original controller. Used together with a feed forward implementation, the performance was further improved.

List of variables

Symbol	Description	Unit	First appearing in section
d	Load disturbance		4.1.5
e	Control error		4.2.1
e_s	Control signal saturation error		5.4.1
f_c	Low-pass filter cutoff frequency	Hz	5.3.1
f_m	Motor frequency	Hz	4.1.4
f_n	Nyquist frequency	Hz	4.1.4
f_r	Rotor frequency	Hz	4.1.3
h	Sampling interval	s	4.1.7
K	Proportional gain		5.1
n	Oscillative disturbance signal	Hz	4.5
n_g	Gear ratio		4.1.3
t_n	Time at sample number n	s	5.4.1
T	Low-pass filter time	s	5.2.1
T_d	Derivative time constant	s	5.4
T_{dist}	Disturbance period time	s	4.1.5
T_i	Integral time	s	4.2.3
T_p	Pump period time	s	4.4
T_{pR}	Relative period time		4.4.1
u	Control signal		4.1.2
V	Pump stroke volume	ml	4.1.3
v	Unsaturated control signal		5.4.1
y	Pump speed feedback signal (tacho signal frequency)	Hz	4.1.3
y_{filt}	Filtered tacho signal frequency	Hz	4.4.1
y_R	Relative tacho frequency		6.4
y_{set}	Tacho frequency setpoint	Hz	4.2.1
Qb	Blood flow	ml/min	4.1.3
Qb_{filt}	Filtered blood flow	ml/min	4.2.3
Qb_{set}	Blood flow setpoint	ml/min	4.1.4
Qb_{setAct}	Corrected blood flow reference signal	ml/min	4.2.3
$max\lvert\frac{\Delta Qb_{set}}{\Delta t}\rvert$	Allowed Qb_{set} change/second	$\frac{ml/min}{s}$	5.1.3
θ	Motor angle	rad	4.1.7
τ	Time constant in process transfer function	s	4.1.7
ω	Angular frequency	rad/s	4.5
Γ	Motor torque	Nm	4.1.7
	Transfer functions		
$C(s)$	Controller transfer function		4.5
$F(s)$	Filter transfer function		5.3.1
$P(s)$	Process transfer function		4.5
$T(s)$	Complementary sensitivity function		4.5

Contents

Chapter 1

Introduction

1.1 Background

This section includes a brief introduction to the hemodialysis treatment and the peristaltic blood pump. It also includes an overview of the company Gambro, and finally the purpose of the thesis is presented.

1.1.1 Hemodialysis

The human body is very sensitive to changes in its internal environment. For example, our body temperature is relatively constant, but a slight change of no more than a few degrees immediately has a negative effect. Therefore, it is of the highest importance for the body to maintain a relatively constant internal environment, a state known as homeostasis [1].

Homeostasis is preserved by the action of many processes in the body. A largely simplified rule states that inputs should be equal to outputs in order for the body to maintain homeostasis. We digest food and drink water, but we also need to excrete fluid and metabolic by-products in order to establish a balance. The two main organs that possess excretory abilities are the lungs and kidneys, but the kidneys are the most effective.

Roughly 1200 ml of blood flow through and are filtered in the kidneys each minute. Here, a number of important regulatory functions take place. The kidneys perform regulation of the blood volume, blood pressure, plasma concentration of ions and help stabilize the pH in the blood. They also filter out harmful waste products such as urea and, at the same time, prevent valuable nutrients from leaving the body with the urine [1]. Clearly, the kidneys are very important for homeostasis and for the body to function correctly, which is why we need a good substitute in those cases where the kidneys fail to function.

The purpose of hemodialysis is to perform the actions that the kidneys would normally have done, i.e. returning the extracellular fluid to a normal physiological composition. This is done extracorporeally, that is, in a machine outside the body. Figure 1.1 shows an overview of the hemodialysis process. The blood needs to be cleared from toxic metabolites and acids as well as excessive water [2]. This is done in the heart of the dialysis machine; the filter. The filter consists of a semipermeable membrane where the blood is kept on one side

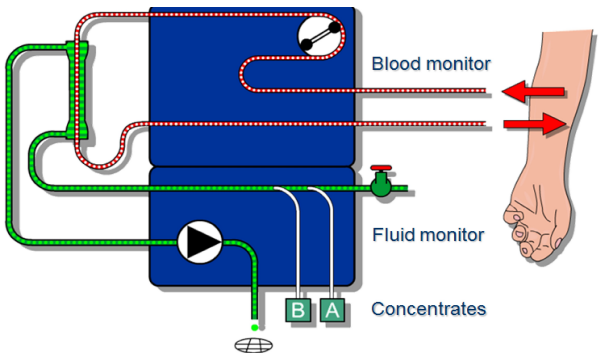


Figure 1.1: Schematic diagram of the dialysis treatment procedure

and the dialysis fluid on the other side. The two fluids are pumped in opposite directions to enable the most effective filtering possible. Since the semipermeable membrane only allows small molecules to be transported over the membrane, larger molecules such as proteins and blood cells stay within the blood. The dialysis fluid is a solution composed of very clean water together with electrolytes, glucose and bicarbonate [2].

There are two physical processes that enable the transportation of molecules over the membrane; diffusion and ultrafiltration. Diffusion occurs when there is a difference in concentration between initially separated fluids and gases. A balance between the concentrations is found when particles are permitted to move along the concentration gradient for a sufficient time period.

Ultrafiltration is a physical process that forces water molecules through the semipermeable membrane. Through ultrafiltration it is possible to remove exessive water from the blood. This is done by keeping a negative pressure on the side of the membrane that contains the dialysis fluid and a positive pressure on the side containing the blood.

Before a dialysis treatment, the blood contains toxic waste products, an excess of water and it is slightly acidic. This puts requirements on the dialysis fluid. It should of course not contain any metabolic by-products in order to enable optimal removal of these products from the blood. The concentration of electrolytes should be slightly smaller than the electrolyte concentration in the blood and a bicarbonate buffer helps returning the blood to a physiologically normal pH value.

Normal blood flows during treatment range between 200-400 ml/min. The treatment is normally performed three days a week and takes approximately 3.5-5 hours [2]. Hemodialysis treatment saves lives and handles a very delicate system which is why it is of the utmost importance that these machines are safe and reliable. This is ensured through numerous safety monitors such as pressure sensors and air detectors.

1.1.2 The peristaltic pump

The blood flow through the dialysis machine is driven by a peristaltic pump. It is a quite simple construction, but it has its advantages in these kind of treatments. The pump consists of a rotor with two rollers attached to it, see Figure 1.2.

An elastic tube is fitted to a circular chamber that partly surrounds the rotor. When the pump is set into motion, the rollers will pinch the elastic tube and force the blood to move forward. The compression of the plastic tube induces a resistance on the pump motor which will create a relatively large disturbance in the fluid flow if no motor speed controller is used.

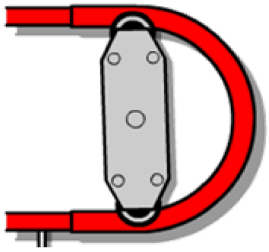


Figure 1.2: Schematic figure of the peristaltic pump

The great advantage of using a peristaltic pump in dialysis treatment is that the blood is contained in a tube and is thus in no contact with anything else which lowers the risk of contamination. The tube set is changed between each patient. However, the pump must be constructed so that the rollers do not pinch the tube too much and destroy the blood cells [4].

The pump treated in this project had a diameter of approximately 7.5 cm. Using a tube with an inner diameter of about 0.5 cm gave a stroke volume of 4.47 ml which had been experimentally determined. The pump was driven by a 24 V DC motor. The angular speed of the motor was transferred to the pump driveshaft through a set of elastic driving belts in a gear box. The gear ratio between the motor shaft and the pump rotor was 49.34.

1.1.3 Safety supervision of blood access to the patient

About 200 patients die every year during their dialysis treatment due to so called venous needle dislodgement (VND). This is an accidental removal of the venous needle during treatment that causes the patient to loose a lot of blood if no life saving actions are taken. It is a serious complication which requires that the dialysis machine has an effective safety supervision system that can detect the VND and stop the treatment immediately. Currently, a new safety system is under development in order to prevent this problem. The idea is to determine the patient's heart pulse on both the arterial and venous sides of the dialysis machine and then compare them to each other. When VND occurs the venous heart pulses will vanish. Hence, if the venous heart pulses are not detected the arterial and venous heart pulses will no longer correlate and thereby indicate VND [3].

Determining the heart pulses is challenging, since a peristaltic pump induces a dominating periodic disturbance in the blood pressure. These disturbances originate from the rollers pinching the tube during pumping. This is overcome by extracting the heart pulses through digital filtering which removes the pressure disturbances caused by the pump. Although, it is expected that if the pump has a nearly constant period time, the large disturbances can be more efficiently filtered.

1.1.4 Gambro

Gambro was founded in 1964 in Lund, Sweden, by Holger Crafoord in collaboration with Nils Alwall and in 1967 they launched their first dialysis machine. Today, Gambro is a leading global medical technology company in developing, manufacturing and supplying products and therapies for kidney and liver dialysis, myeloma kidney therapy, and other extracorporeal therapies for chronic and acute patients. Gambro has 7500 employees, 13 production facilities in nine countries, and sales in more than 100 countries [5].

Over the years Gambro has developed a number of dialysis machines with new improved features and the company owns the rights to more than 1800 patents. In 2007 a new, robust, easy-to-use machine was introduced [5]. It was named AK 96 and its blood pump control system will be subject to analysis in this project.

1.2 Objective

The primary objective is to find a control system that can keep the relative period time of the peristaltic pump within a standard deviation of 0.1 %. The secondary goal is to keep the variation in blood flow as constant as possible within a pump revolution. This is carried out in two steps. First, the existing control system is examined and analyzed. Then the objective is to find, through theoretical and experimental investigations, a new controller that improves the performance of the pump.

Chapter 2

Test setup

2.1 Background

This chapter will provide an overview and description of the test setup. The first section gives an introduction to the hardware setup and in the second section the software setup is explained. Post processing of data will also be briefly described. For an overview of the test setup, see Figure 2.1. The setup was provided and installed by Gambro's department of Research and Development.

LabView was the programming language used throughout the project. It was used to implement a software-based controller and to enable communication between the process and the operator. Through data acquisition (DAQ) cards it was possible to obtain feedback signals from the process and send back control output signals. The LabView code was written in a computer which hereafter will be called the host computer. An external central processing unit (CPU) was also used to process the real time code which implemented the measurement and control functions. This CPU will hereafter be called PXI computer.

2.2 Hardware

The hardware pieces in the experimental setup will be discussed in the same order as in Figure 2.1 begining with the motor and the peristaltic pump.

Three different signals were obtained from the pump unit;

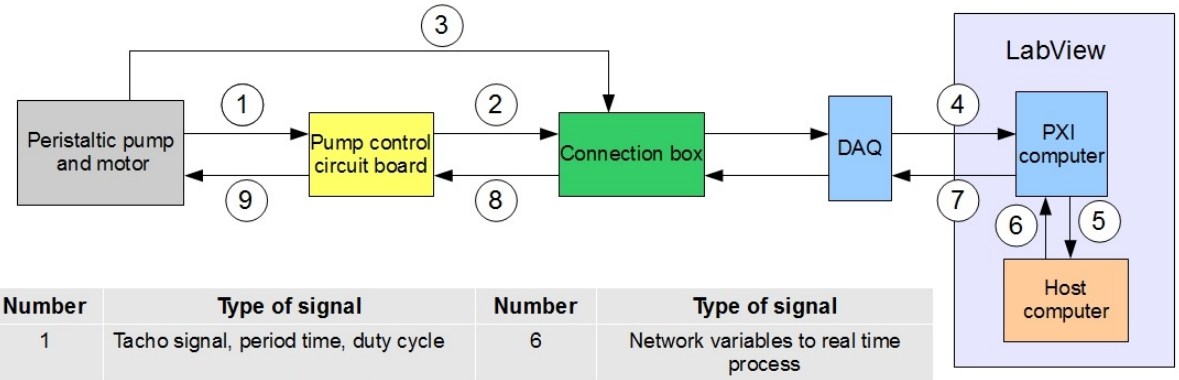
i) The rotational frequency of the motor, a pulsed signal called the tacho signal. The tacho signal was obtained through a stroboscope inside the motor which gave 200 pulses per revolution.

ii) The period time of the peristaltic pump. It was measured with a hall sensor associated with the rotor of the pump.

iii) The arterial and venous pressures. Two pressure sensors were placed before (the arterial pressure) and after the pump (the venous pressure).

Two different pump control circuit boards were available and examined during the project. They were developed and manufactured by Gambro. Each circuit board contained a hardware controller which was implemented in a field programmable gate array (FPGA) circuit. They differed in some aspects, but this will be explained in more detail in Section 4.2.1. Something of interest is that one of the circuit boards was modified in order to bypass the built-in controller and thereby enable the use of a software based controller implemented in LabView.

The circuit board was connected through two connection boxes (NI TBX-68), to a chassis (NI PXI 8183) containing two DAQ modules and a CPU (PXI computer). These read input signals, wrote output signals and processed LabView code. All of the signals were read/written by a multifunction DAQ module (NI PXI-6229), which was able to handle both analog and digital signals, and a counter/timer board (NI PXI-6602), only handling digital signals. The PXI computer (NI PXI 8183) featured an 850 MHz Intel Celeron processor. Almost all the software-based control algorithms and signal processing, which will be explained in the next section, were executed in the PXI computer.



Number	Type of signal	Number	Type of signal
1	Tacho signal, period time, duty cycle	6	Network variables to real time process
2	Tacho signal, period time, duty cycle	7	Control signal / reference signal
3	Arterial and venous pressure	8	Control signal / reference signal
4	Tacho signal, period time, duty cycle, pressures	9	Control signal / reference signal
5	Real time variables to network		

Figure 2.1: Overview of the experimental setup.

2.3 Software

2.3.1 Overview of LabView application

LabView was the programming language used for the implementation of the control algorithms, signal processing and visualization of signals. It is a graphical programming language which differs from regular programming languages in essentially two ways; icons

represent functions used instead of lines of text and the flow of data determines the execution rather than instructions. Labview is especially helpful when working with real time systems since it automatically takes care of prioritized tasks with just a priority number as input [6].

The programs written in LabView are called virtual instruments (VIs) and as the name implies, the program is meant to resemble the appearance of real physical instruments. The VI consists of a front panel and a block diagram. The front panel serves as the user interface. Here, blocks such as graph windows, numeric indicators, controllers and ON/OFF buttons can be built. The block diagram contains all the source code that defines the functionality of the VI. The code is written by adding graphical representations of functions and then wiring them together. The final code in the block diagram slightly resembles a flowchart [6]. Figure 2.2 serves as an example of a VI with its front panel and block diagram. It shows a VI used in the project where disturbance spikes were filtered out from the tacho frequency signal.

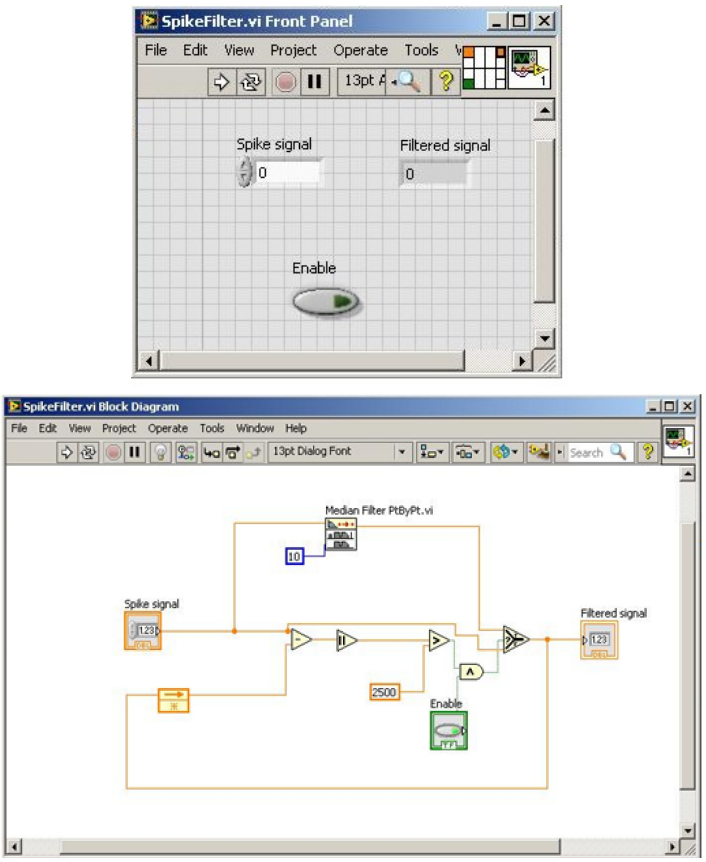


Figure 2.2: Top: front panel containing numeric indicators to visualize input signal and output signal and ON/OFF button to enable/disable the filter. Bottom: block diagram showing the source code.

The software-based experimental setup consisted of two toplevel VIs called the target application and the host application. The front panels of these applications can be viewed in the Appendix. The target application handled all the code associated with the processing of signals and controller implementations as well as the reading and writing of input/output signals. Processing action of this VI was managed by the PXI computer described in the previous section. The host application had a much smaller amount of code. It handled the graphical user interface (GUI) functions such as visualization of signals through graph windows and numeric indicators. It also handled user commands such as the reference signal and controller modes (e.g. open loop, existing controller or new PI controller). This VI was processed by the CPU in the host computer. The variables used in the host application was called network variables while the variables in the target application was called real time variables. The network variables was used in the communication process between host and target application. Some choosen real-time variables were logged to a text file on the PXI computer. Below follows a description of the tasks written in the target application.

Since the target application handled many different real-time calculations it was necessary to use a software construction called "timed loops". These loops enabled tasks to have different priorities and execution period times. The function of the loops will be discussed briefly in order of priority.

Read loop *priority: 100, sampling interval: 5 ms*
All the input signals from the process were read in this loop and therefore it set the limit for the smallest sampling interval possible for the rest of the loops. Also, this loop contained VIs handling signal processing calculations, e.g. converting blood flow to frequency and low-pass filtering of tacho frequency.

Blood pump speed control *priority: 90, sampling interval: 5 ms*
This loop contained the main calculations for a PI controller algorithm as well as a feed forward function. Here, the output control signal was written and sent back to the process.

Rate limit and gain scheduling calculation *priority: 65, sampling interval: 100 ms*
A ramping function was implemented in this loop to limit the rate of change in the reference signal. A gain scheduling function was also used in order to enable change of the PI parameters and time constant of the low-pass filter as a function of the blood flow set point.

Data log *priority: 60, sampling interval: 5 ms*
This loop wrote sampled data, such as time stamp, reference signal, control signal and period time to a text file in order to enable post processing of data.

Qb software control *priority: 55, sampling interval: 5 ms*
A master blood flow controller that was part of the existing control system, which will be explained in Section 4.2.1, was implemented in this loop.

Host communication *priority: 50, sampling interval: 10 ms*
The host and target applications communicated through this loop. Selected signals

were forwarded to the host application to be visualized in plots and manually chosen parameters from the host applications such as reference signal and duty cycle were read.

Period time measurement This loop was a while loop and had less priority than the timed loops and no fixed sampling interval. Its main function was to wait until it received a signal from the hall sensor and to register the period time.

2.3.2 Post processing of logged data

Post processing of logged data was performed using MatLab. Text files obtained from LabView included columns with different logged data as mentioned before. The MatLab scripts read the columns, created vectors and then various calculations were performed. The main functions used were calculating mean and relative standard deviation and plotting graphs. Also, the fast fourier transform (FFT) function was used in order to create frequency spectra.

Chapter 3

Performance requirements

As mentioned in Section 1.2, the primary objective of this project is to find a control system that can keep the peristaltic pump at the most constant period time as possible, with the main goal to achieve a relative period time standard deviation lower than 0.1 %. However, other behavioral aspects are also important. It is preferable that variations in the blood flow within a revolution stays within certain limits so that the treatment is not disturbed.

Another obvious requirement is that the control system should always be stable. The use of needles with various sizes and different sets of tubes should not affect the performance of the pump.

During the dialysis treatment, it is important not to shock the patient by starting with a high blood flow or changing the flow too quickly. Therefore, the blood flow is gradually increased in steps until it reaches its final value. Hence, a fast step response is not a requirement on the control system. It is actually quite the opposite; after a new setpoint value has been selected the process should slowly adapt without any overshoot.

Chapter 4

Analysis of existing control system

4.1 Open Loop

4.1.1 Background

A series of measurements were done in open loop with and without the blood tube to understand the blood pump system. This was done by manually selecting a control signal u (defined in the following section). The control signal was then sent to the DC motor through the pump control circuit board. Measurements were made on the tacho signal frequency y which was sampled at a 5 ms interval. The relation between u and y is illustrated in Figure 4.1. The data was then analyzed in order to better understand the motor speed properties, blood tube load disturbances and step response dynamics. The measurements were also made in order to make a crude linear model of the input output relation between u and y.

4.1.2 Control signal u

The control signal was of pwm (pulse width modulated) type and consists of pulses with fixed height and variable length, while the pulse frequency is constant. It is the pulse length that defines the mean effective voltage, the pwm signal is thus determined by $\frac{\text{pulse length}}{\text{period time}}$ which is called duty cycle. It is the duty cycle (in %) that we define as the control signal u.

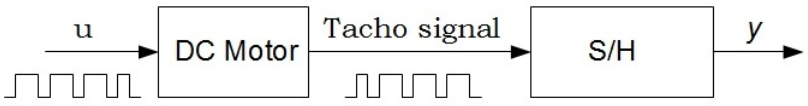


Figure 4.1: A diagram of the DC motor as an input/output system with input signal u (duty cycle) and measured value y (tacho signal frequency).

4.1.3 Motor speed signal y

The measured signal y is the sampled frequency of the tacho signal, y [Hz] determines the motor frequency f_m [Hz] according to

$$f_m = y/200.$$

The factor 200 comes from the fact that the stroboscopic plate generates 200 pulses each motor revolution.

With gear ratio n_g (49.34), rotor frequency f_r and pump period time T_p are given by

$$f_r = \frac{f_m}{n_g}, \quad T_p = 1/f_r.$$

Each full revolution two pump stroke volumes V [ml] (4.47 ml) are delivered by the pump. This gives the blood flow Qb [ml/min]:

$$Qb = f_r \cdot 2V \cdot 60 = \frac{f_m \cdot 2V \cdot 60}{n_g} = \frac{y \cdot 2V \cdot 60}{200 \cdot n_g} = 0.054 \cdot y.$$

The factor 60 comes from the conversion from seconds to minutes.

4.1.4 Frequency analysis

A characteristic property of the motor speed signal y is its high frequency oscillations at constant u. Figure 4.2 shows y at constant u without blood tube disturbances (see section 4.1.5). The mean y corresponds to a Qb of 100 ml/min.

In Figure 4.3, the discrete fourier transform (DFT) of y from Figure 4.2 can be viewed. In Table 4.1 the dominant DFT peak amplitude and corresponding frequency for blood flows 100-300 ml/min are displayed, all measurements contained 5 minutes of data sampled at 200 Hz. At higher blood flows frequency aliasing effects made the analysis more difficult. The sampling frequency at 200 Hz gave the Nyquist frequency, $f_n = 100$ Hz, which means that a sampled frequency higher than 100 Hz will be aliased as a frequency below f_n.

Table 4.1: Frequency and amplitude of dominant peak in the y DFT, as well as motor frequency f_m

Qb_{set}	Frequency	f_m	Amplitude
100 ml/min	18.7 Hz	10.1 Hz	106.2
200 ml/min	36.5 Hz	19.4 Hz	51.66
300 ml/min	55.4 Hz	28.0 Hz	27.35

As seen in Table 4.1, the oscillating frequency and amplitude are both dependent on blood flow Qb. The amplitude decreases with the blood flow while the frequency increases with the blood flow. Interestingly the dominant peak seems to stay around approximately two times the motor frequency.

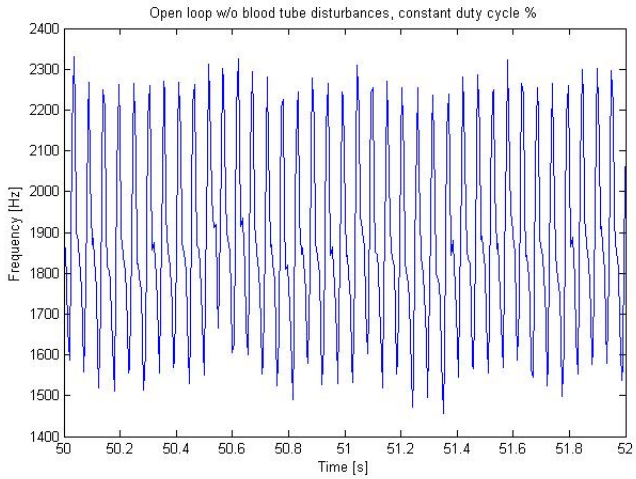


Figure 4.2: Tacho signal frequency at constant u, the oscillations goes from 1500-2300 Hz which corresponds to a Qb interval of $\pm$ 21.5 ml/min. This measurement were done without the blood tube.

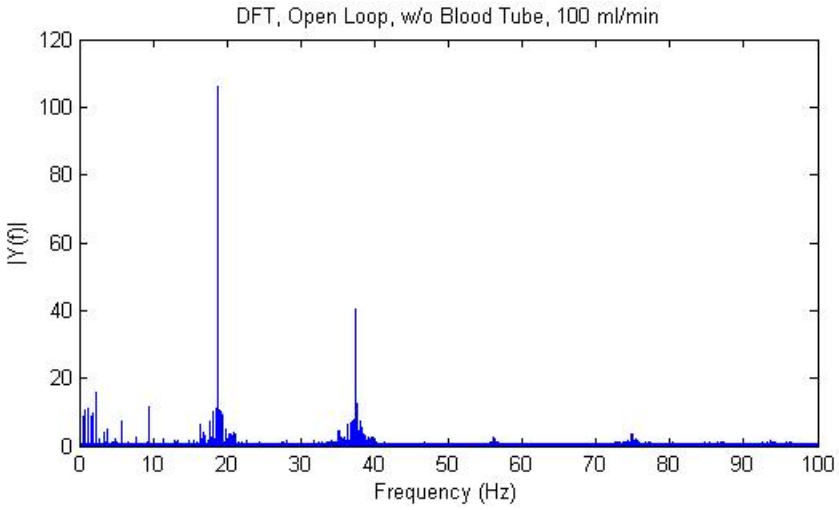


Figure 4.3: DFT of tacho signal frequency, open loop at 100 ml/min blood flow. Two dominant peaks are found around 19 and 37 Hz.

The overall variance of motor speed was also dependent on blood flow, it was found to be especially high at lower blood flows (around 100 ml/min) and decreased with the blood flow.

The causes of the oscillations could not be identified but could be the result of varying friction or uneven revolutions. The elastic drive belts could also be a source of resonance

effects. It is also possible that the oscillations stem from an uneven stroboscopic plate.

The problem of having high frequency oscillations in the motor speed signal y is that the oscillations would be induced in the control signal when using y as a feedback signal in a feedback control loop. In worst case the oscillations could even be amplified by a feedback controller.

4.1.5 Blood tube load disturbances

Measurements were also done with a blood tube to investigate blood tube effect on the motor speed. When the pump rollers squeeze the blood tube, the tube resistance acts as a load disturbance d on the motor, see Figure 4.4. d depends on pump angle and has a period time T_{dist} which equals half of the pump period time.

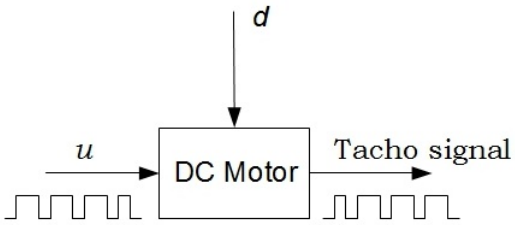


Figure 4.4: Inputs to the DC motor are the control signal u and the periodic tube disturbances d.

The tube disturbance level depends on whether one or two rollers are in contact with the tube. In Figure 4.5 the disturbance effect on the motor frequency can be viewed when the motor is driven with constant duty cycle, the mean tacho frequency in Figure 4.5 corresponds to a blood flow at 165 ml/min. The speed dip occurs when the number of rollers in contact with the tube shifts from one to two, in the same way the overshoot occurs when number of rollers in contact with the tube shifts from two to one. Note that one roller is always in contact with the blood tube.

In Figure 4.5, y varies in a 3000 Hz interval which is equivalent to a blood flow interval of 162 ml/min! In order to keep the blood flow at a steady level, it is necessary that the tube load effects on motor speed are removed or at least reduced by a feedback controller.

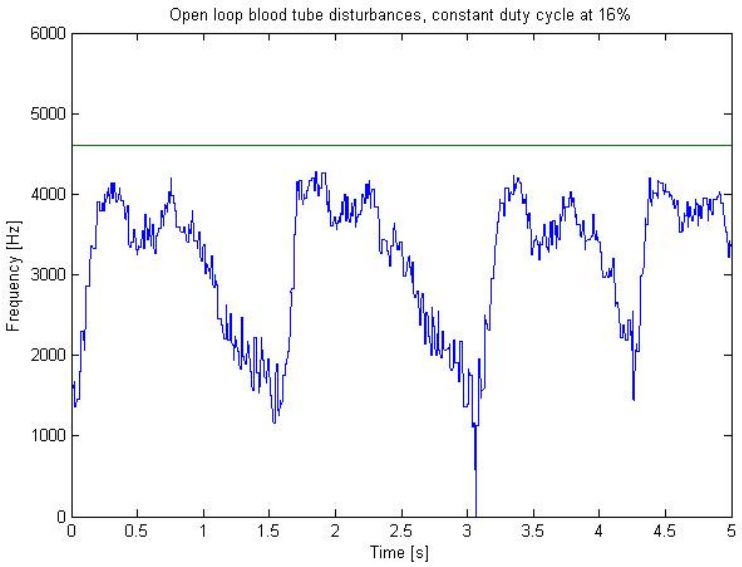


Figure 4.5: Tacho frequency signal y in open loop with constant duty cycle at 16 % with blood tube disturbances. The green line (equivalent to $Qb = 250$ ml/min) indicates where y would have been without blood tube load disturbance.

4.1.6 Lower duty cycle limit

At low duty cycles the blood tube resistance is too high in order for the blood pump to be able to make full revolutions. With duty cycles below 15 % the blood pump will not move at all and above 17 % the pump moves without stopping. At low flows it is thus important to have a controller that does not actuate a too low duty cycle. Sudden pump stops are not wanted during treatment.

4.1.7 Linear system model

In order to understand the process better, the DC motor dynamics between duty cycle u and mean tacho frequency y was approximated with a simple first order linear transfer function, where the electrical dynamics are neglected:

$$P(s) = \frac{K_p}{1 + s\tau}.$$

Where K_p [Hz/%] is the process gain, τ [s] is a time constant, and s is the complex Laplace argument. The model was achieved by assuming that the torque Γ_{motor} generated by the motor is proportional to duty cycle:

$$\Gamma_{motor} = k_u u.$$

And that the rotation dynamics can be described by the first order equation

$$J\ddot{\theta} + b\dot{\theta} = \Gamma_{motor}.$$

Where θ is the motor angle. The transfer function above is then obtained by observing that the motor frequency y is proportional to $\dot{\theta}$. J is the moment of inertia and b is a damping factor. This is an attempt to model a linear approximation of the input/output transfer function in Figure 4.1. The unknown constants K_p and τ were estimated through measurements in open loop without tube load disturbances. K_p was estimated to 386 $\frac{Hz}{\%}$ by measuring the frequency signal at different duty cycles, without tube disturbance, see Figure 4.6.

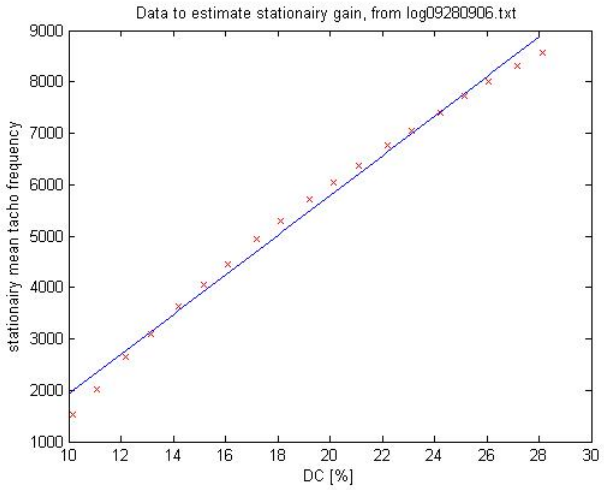


Figure 4.6: Mean tacho frequency plotted against duty cycle, the slope coefficient was estimated to 386 $\frac{Hz}{\%}$.

T was appreciated to 0.02 seconds by studying step responses in open loop without tube disturbance. In the model, T is the time it takes for the tacho frequency to reach 0.63 of the final value in a unit step response [8, p. 48-49]. A step response can be seen in Figure 4.7.

The reason for estimating model parameters from measurements done without tube load disturbances was that the y signal is more easily analyzed without the disturbance variations. On the other hand if the blood tube affects the DC motor dynamics, our estimated model parameters might be inaccurate for the system with a blood tube. The introduction of a blood tube probably increases the damping.

Since discrete controllers will be used in the project, it is interesting to have a transfer function of the sampled model. With sample interval h and a zero order hold input signal, the sampled process pulse transfer function is given by: [7, p. 16-19]

$$P(z) = \frac{K_p(1 - e^{-h/\tau})}{z - e^{-h/\tau}}.$$

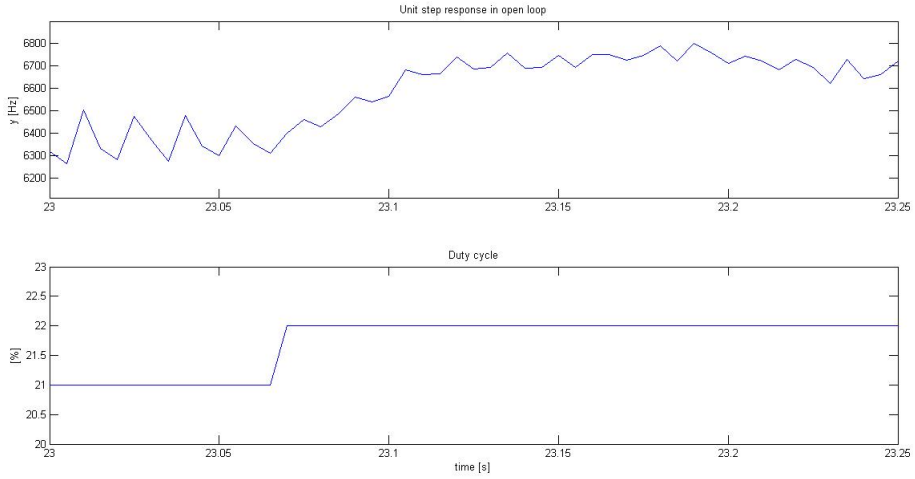


Figure 4.7: Open loop y step response around 300 ml/min, the duty cycle was increased by 1 %. The rise time T was estimated to 20 ms.

4.1.8 Tacho frequency signal spikes

In the tacho frequency signal y, high amplitude spikes frequently occur. An example of a spike is shown in Figure 4.8. The spikes could be a result of measurement noise on the tacho signal, and is unlikely to stem from the process. In order to compute DFT or to use the signal as a feedback signal it is necessary that the spikes are removed.

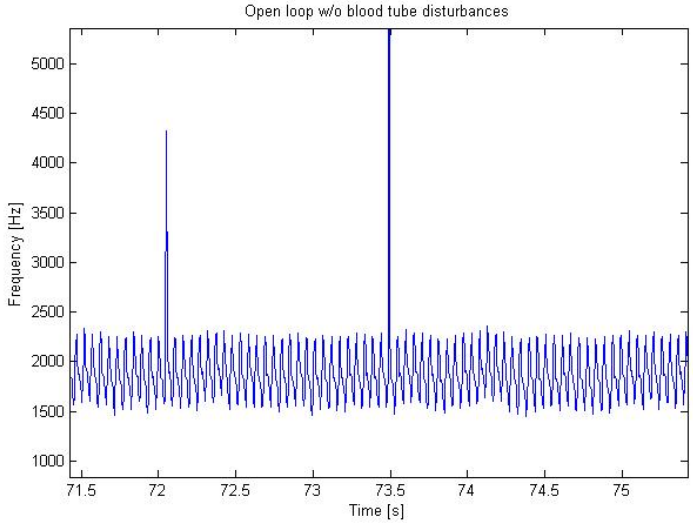


Figure 4.8: Tacho frequency signal spike.

4.2 Existing Control System description

The existing control system consists of two controllers connected in cascade. The slave controller is a fast motor speed controller implemented in hardware while a slower master controller controls the blood flow and is implemented in the software. Both controllers use integral action.

4.2.1 Hardware control algorithm

The hardware control function can be described by two 4-bit counters and a comparator, see Figure 4.9. Counter 1 receives two signals.

1. A feedback signal: the tacho signal from the motor with frequency y.

2. A reference signal: a generated pulse train signal with frequency y_{set}.

When a pulse from the reference signal is recieved by counter 1 it counts up one step, and when a pulse from the feedback signal is recieved by counter 1 it counts down one step. Hence the feedback signal has a negative impact on the counter value while the reference signal has a positive impact. A DC signal is generated from counter 1 with a DC level proportional to the counter value.

If both y and y_{set} have the same frequency (the frequency error equals 0), counter 1 should ideally send a constant DC signal to the comparator.

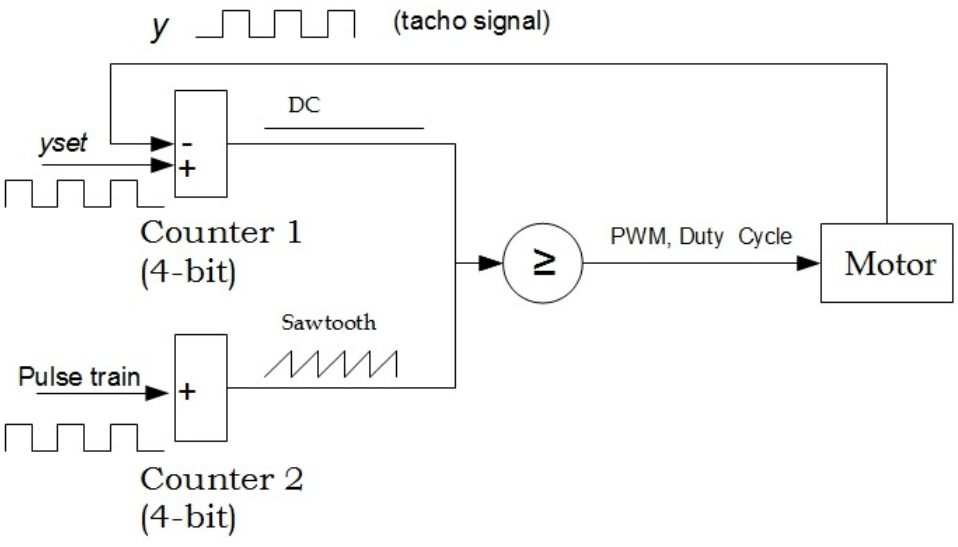


Figure 4.9: Hardware controller algorithm scheme. The +,- signs indicates if the counter counts up/down when recieving a flank.

Counter 2 counts the pulses from a high frequency clock circuit, when the maximum counter value is reached the counter wraps around and starts to count at its minimum value. Counter 2 generates a signal with a voltage proportional to counter value, this creates a signal with sawtooth appearance. The signals from both counters are then compared in the comparator which generates a pwm signal. The pwm signal is high (24 V) when the DC signal (from the first counter) is higher or equal to the sawtooth signal, and low (0 V) when the sawtooth signal is higher than the DC signal. The pwm signal is then sent to the DC motor where the pwm duty cycle acts as control signal amplitude. When counter 1 is at its maximum level the duty cycle is 100 %, and when at its lowest value the duty cycle is 6.25%.

The hardware controller can be seen as a purely integrating controller. For instance if y_{set} is higher than y, counter 1 starts to count up at a rate proportional to frequency difference. The increased counter value then leads to an increased duty cycle. This increases the motor speed and the frequency difference decreases towards zero and the feedback loop would reach stationarity. Unfortunately, perfect stationarity cannot occur since the control signal is quantized in steps determined by the counter resolution.

In order to estimate the controller performance, the following thought experiment was made:

Imagine that a frequency error $e = y_{set} - y, e > 0$ is sent to the hardware controller in open loop. This can be done by setting the reference frequency y_{set} to e and the feedback signal y to 0. The constant error will cause the reference signal to count up counter 1 $e\Delta t$ times during a time interval Δt. This leads to an increase in duty cycle by $\Delta u = e\Delta t 6.25\%$. Where 6.25 % is the resolution in duty cycle. From this example one could argue that the controller gain (integration time) depends on the counter resolution (number of bits).

4.2.2 Software controller

The 4-bit controller creates a negative stationary error (see Section 4.4.1), a master controller implemented in the AK96 software is used to lower y_{set} to compensate for the too high motor speed.

4.2.3 Software control algorithm

The blood flow Qb is estimated by calculating the tacho pulse count increment $\Delta tacho$ during time Δt and dividing it by Δt, this gives the tacho frequency average during that time interval. In AK 96, $\Delta t = 0.5$ s. Qb is then given by

$$Qb = \frac{\Delta tacho \cdot 2V \cdot 60}{\Delta t \cdot 200 \cdot n_g}.$$

A filtered blood flow Qb_{filt} is then calculated by taking the average of the current and previous Qb value:

$$Qb_{filt}(t) = \frac{Qb(t) + Qb(t-1)}{2}.$$

With the algorithm described in the following pseudocode, the software controller calculates a corrected Qb_{set} value, Qb_{setAct}, that is being sent as a corresponding y_{set} to the hardware controller.

This code is executed once every 0.5 seconds :

```
e = Qb_set-Qb_filt;

I = I + e;

Qb_setAct = Qb_set + 0.005 * I;
```

Since the blood flow reference correction is a sum of blood flow errors, this controller can be seen as integrating;

$$\text{correction} \propto \int e(t)dt.$$

The complete controller scheme is illustrated in Figure 4.10. If a negative blood flow error occurs (which is the case for the 4-bit hardware controller) the integrator I becomes negative, the software controller then calculates a new lower frequency set point which compensates for the stationary error. The blood flow controller is equivalent to a controller with discretized integral action:

$$Qb_{setAct} = Qb_{set} + \frac{h}{T_i}\sum_{i}^{n} e_i,$$

with $h = 0.5s$ the integral time T_i becomes 10 s.

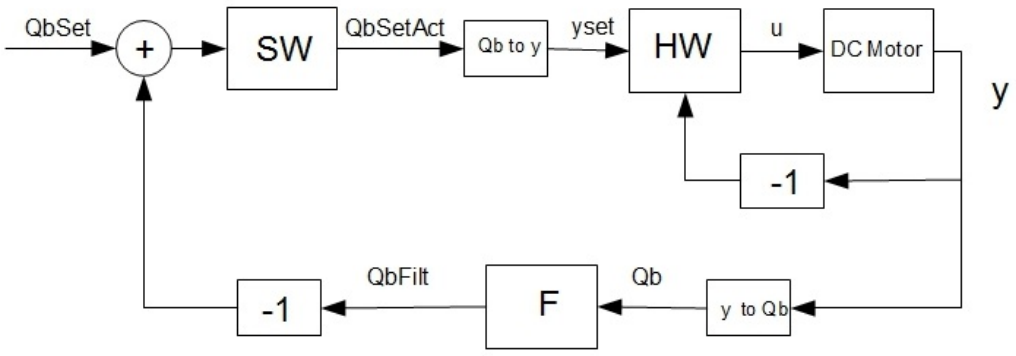


Figure 4.10: Schematic figure on how the software controller (SW) can correct for a stationary error in the hardware (HW) control system. F is the Qb filter.

The controller also has a Qb_{setAct} lower bound, *lowlimit*, at 10 ml/min. When the lower bound is reached, I is set to $lowlimit - Qb_{set}$ to prevent integrator windup. When a change is made in Qb_{set}, I is reset to 0.

4.3 6-bit hardware controller

In another Gambro dialysis product, AK 200, a similar hardware controller is implemented. The controller works by the same principle as explained in Section 4.2.1, but has a 6-bit counter memory instead of 4-bit. The 6-bit controller can actuate a 4 times more resolved duty cycle, and is thus closer to an ideal continous integrating controller that could actuate an exact duty cycle to eliminate stationary error.

A different counter resolution also results in a different controller gain. Whereas the 4-bit controller would have an integral time proportional to $\frac{1}{6.25}$ (see Section 4.2.1), the 6-bit controller would have a 4 times longer integral time, proportional to $\frac{1}{1.5625}$. As a part of the project we controlled the blood pump with a 6-bit pump control circuit board and compared controller properties with the 4-bit controller. See Section 4.4.4.

4.4 Control system performance analysis

Measurements were done with blood flows on each multiple of 100 from 100 to 500 ml/min, with and without software controller. The measured signals were y, u and pump period time T_p (sampled at 5 ms). Control properties where investigated at stationarity and when steps were made in Qb_{set}. The hardware controller with 6-bit counters was also evaluated in the same way.

4.4.1 4-bit hardware controller

Stationary conditions

In Figure 4.11, two seconds of stationary blood flow with $Qb_{set} = 200$ ml/min is plotted. y, y_{set} and a filtered tacho frequency signal y_{filt} can be seen in the upper part of the plot and u in the lower part. In Table 4.2 the standard deviation of the relative period time T_{pR} is displayed for all investigated blood flows. T_{pR} is given in percent and is computed according to:

$$T_{pR} = \frac{T_p}{\text{mean}(T_p)} \cdot 100,$$

where T_p is a vector with period time data. $\text{Std}(T_{pR})$ was computed using 60 period time data samples.

As seen in 4.11 both u and y are very oscillative, it also seems as if the two signals oscillate at the same frequency. In Figure 4.12, a discrete fourier transform (DFT) of the signals in Figure 4.11 are shown. Interestingly, the dominant frequencies are close to the same frequency found in open loop at 200 ml/min, see Table 4.1. Worth noticing is that the amplitude of the dominant peak in Figure 4.12 is higher than the one found in open

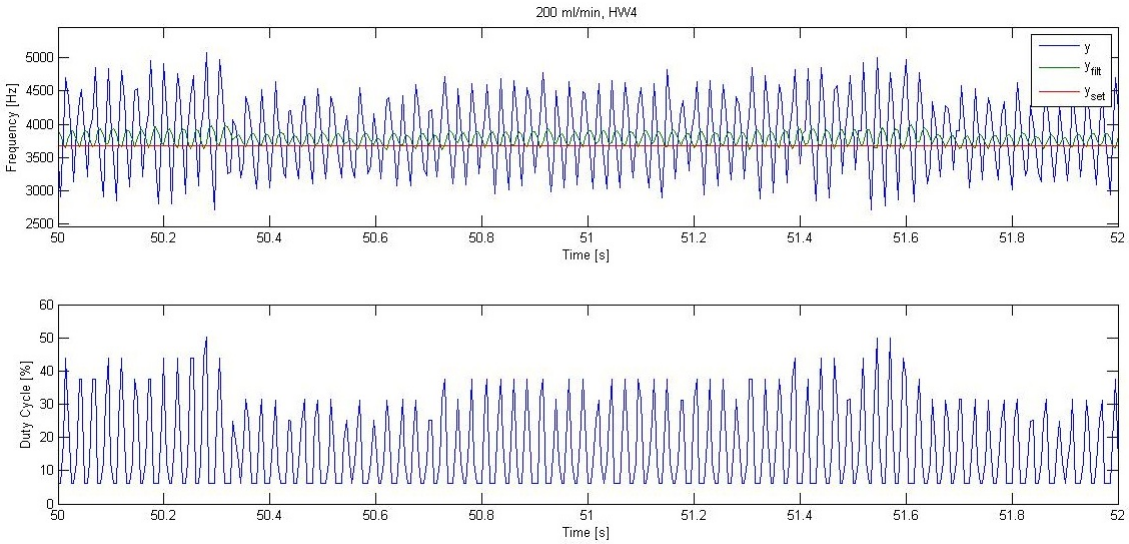


Figure 4.11: Stationary tacho frequency and duty cycle measurements at $Qb_{set} = 200$ ml/min, 4-bit hardware controller.

Table 4.2: T_{pR} standard deviation, 4-bit controller. The computed standard deviations fulfil the specified requirement $\text{Std}(T_{pR}) < 0.1$ %

Qb_{set}	$\text{Std}(T_{pR})$	$\text{mean}(T_p)$
100 ml/min	0.088	4.9454
200 ml/min	0.063	2.6035
300 ml/min	0.075	1.7706
400 ml/min	0.049	1.3344
500 ml/min	0.030	1.0708

loop Table 4.1, even though both measurements have the same amount of sampled data points. It was discovered that the y and u variance and oscillation amplitude decrease while the oscillation frequency increases with Qb_{set}. The T_p variability is also blood flow dependent and decreases with Qb_{set}, see Table 4.2.

In the two second time window in Figure 4.11, two tube load disturbances occur. The disturbance effects are completely hidden in the oscillations.

A stationary error in mean blood flow can also be seen in Figure 4.11 by looking at the filtered signal. The mean blood flow was 10 ml/min higher than the wanted Qb_{set}. When investigating controller properties at higher Qb_{set} the frequency error decreased, these results can be seen in Table 4.3 where stationary error (Qb_{Set} - mean(Qb)) is displayed for different blood flows.

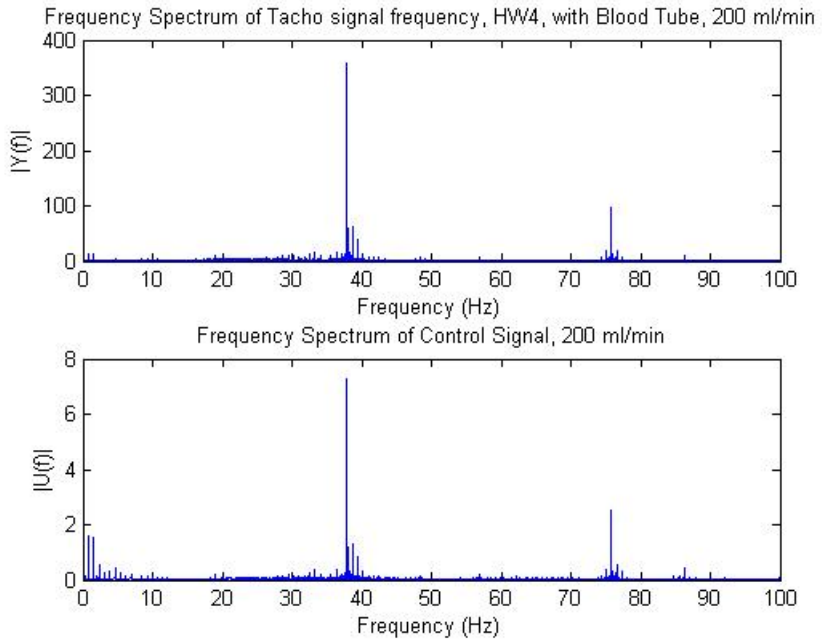


Figure 4.12: DFT of tacho signal and duty cycle in Figure 4.11.

Table 4.3: Stationary error 4-bit hardware controller.

Qb_{set}	Stationary error
100 ml/min	-10 ml/min
200 ml/min	-9.89 ml/min
300 ml/min	-4.5 ml/min
400 ml/min	-4.2 ml/min
500 ml/min	-3.67 ml/min

Step response analysis

To further investigate the controller properties, steps of 100 ml/min were made in Qb_{set}. In Figure 4.13, a step from from 200 ml/min to 300 ml/min is plotted. It takes about 0.1 seconds for the controller to adjust to the new set point. It was hard to analyze step responses with smaller Qb_{set} changes because of the oscillations in y. Here, the settling time is defined as the time it takes for the mean y to enter and remain in a range around the final stationary mean y. The range used was $\pm$ 5 % of the y step size.

In Table 4.4, estimated settling times are shown for different Qb_{set} steps, the settling time seems to slightly decrease with blood flow.

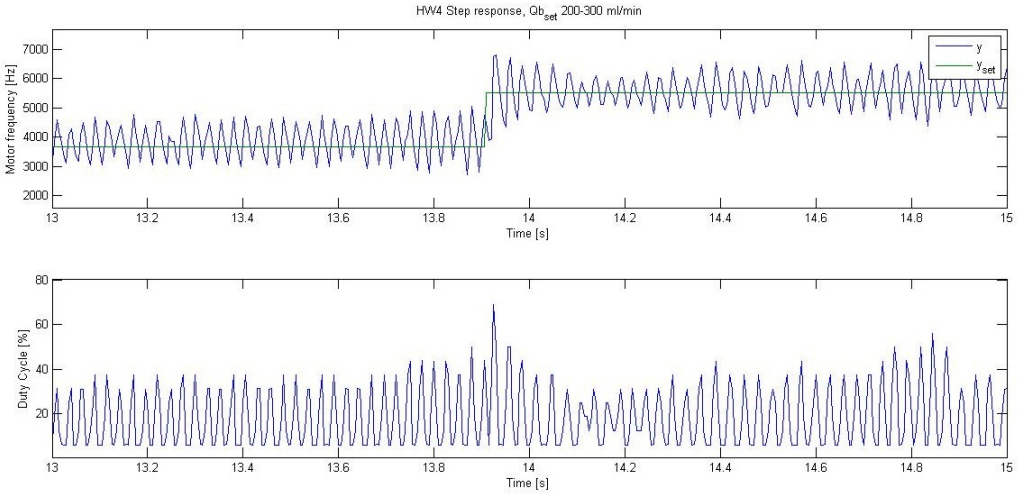


Figure 4.13: 4-bit controller step response, Qb_{set} was increased from 200 to 300 ml/min. The plotted signals are y and y_{set} in the upper plot and u in the lower plot.

Table 4.4: Estimated settling times, 4-bit controller.

Qb_{set} step	Settling time
100-200 ml/min	0.13 s
200-300 ml/min	0.07 s
300-400 ml/min	0.06 s
400-500 ml/min	0.05 s

4.4.2 4-bit controller with blood flow software control

Stationary conditions

In Table 4.5, T_{pR} standard deviation are displayed for all investigated blood flows and compared with the measurements from the 4-bit controller without software controller. The standard deviation was computed using 50 sampled period times. In Figure 4.14, Qb_{filt} and Qb_{setAct} are plotted in steady state when $Qb_{set} = 300$ ml/min.

A small positive mean Qb stationary error < 0.1 ml/min was measured at all blood flows, this is probably due to tube load disturbances that lowers the mean Qb over longer time periods.

Step response analysis

When the 4-bit controller is run together with the software controller, the step response dynamics change, see Figure 4.15. The software controller succesively adjusts Qb_{SetAct} until the stationary error is 0, this increases the settling time of a Qb_{set} step response. In

Table 4.5: T_{pR} standard deviation, 4-bit controller with software blood flow control

	4-bit + software			4-bit controller only	
Qb_{set}	std(T_{pR})	mean(T_p)	Qb stationary error	std(T_{pR})	mean(T_p)
100 ml/min	0.1296	5.3711	0.908 ml/min	0.088	4.9454 s
200 ml/min	0.0949	2.687	0.3732 ml/min	0.063	2.6055
300 ml/min	0.1066	1.7928	0.3538 ml/min	0.075	1.7706
400 ml/min	0.0758	1.3430	0.5956 ml/min	0.049	1.3344
500 ml/min	0.0466	1.0736	0.3825 ml/min	0.03	1.0708

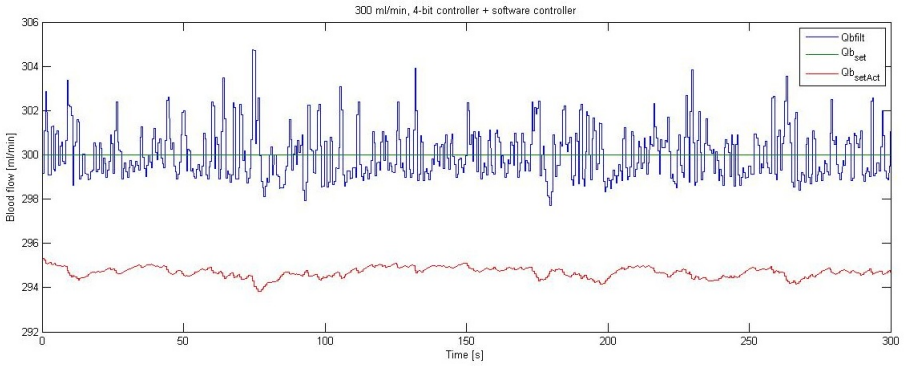


Figure 4.14: HW + SW controller in steady state, at 300 ml/min, the graphs show 5 seconds of Qb_{filt}, Qb_{setAct}, Qb_{set} and u.

Figure 4.15, the settling time is estimated to 40 seconds. In Figure 4.16 a step response without I reset on Qb_{set} change can be viewed, this was investigated in order to see if the I reset affected the step response dynamics.

4.4.3 Measurement result discussion

Oscillations

Since the control objective is to keep the motor speed constant, the large variance in y is clearly an unwanted control property.

The fact that u oscillates at the frequecies found in open loop (see Figure 4.2), indicates that the hardware controller is trying to control the high frequency variations. Since the controller has high gain due to low counter resolution, the controller amplifies the high frequency variations in y. Compare the peak amplitudes in Table 4.1 and Figure 4.12.

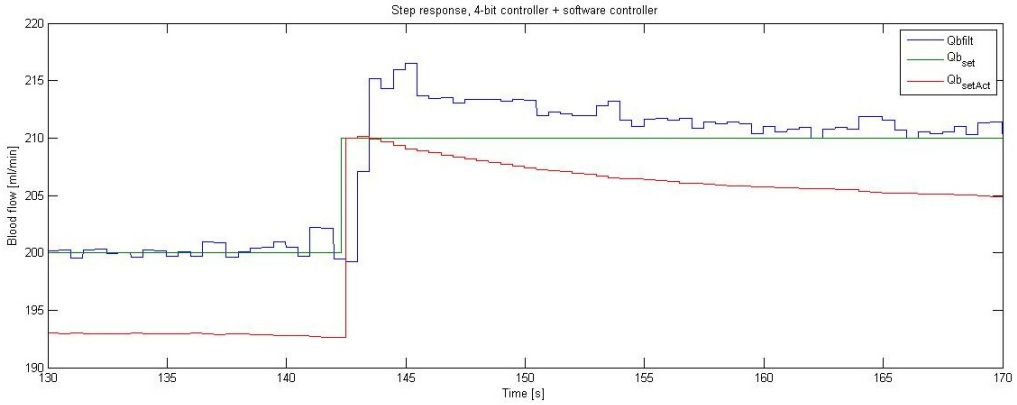


Figure 4.15: Blood flow step response from 200 to 210 ml/min. Qb_{filt}, Qb_{set} and Qb_{setAct} are displayed together.

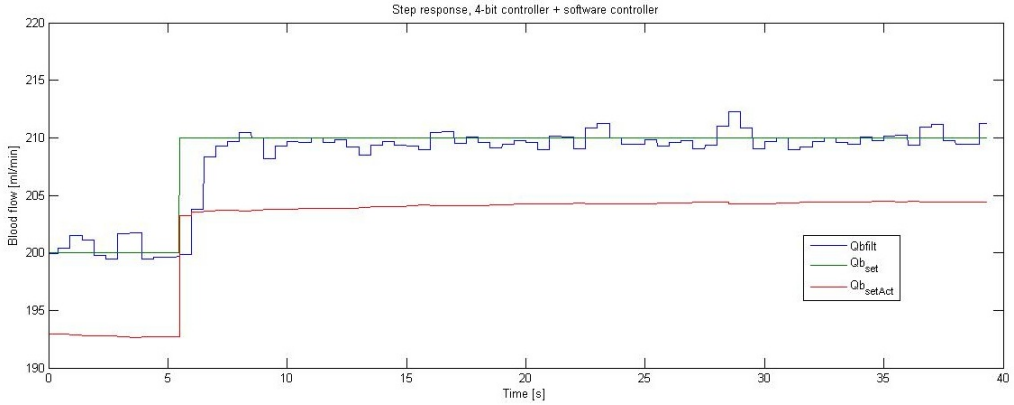


Figure 4.16: Blood flow step response from 200 to 210 ml/min. Here the software controller is modified so that I is not reset upon setpoint change.

Stationary error

The measured stationary error with the hardware controller could be a result of the duty cycle not being able to go lower than 6.25 % which also can be seen Figure in 4.11. When both u and y oscillate around a mean level, the controller counter 1 probably reaches its lowest counter value, but instead of actuating a 0 duty cycle, u is bounded from below by 6.25 % (see Section 4.2.1). This results in a somewhat higher mean u which creates a stationary error. The reason for the lower bound is the $\geq$ comparator. When counter 1 is at its lowest value it is still equal to the sawtooth signal during 6.25 % of the time, this happens when counter 2 is at its lowest value. Since this implies that $\geq$ is fulfilled, the duty cycle cannot be zero. A solution to the problem would be to change $\geq$ to a strictly larger than $>$ comparator.

When Qb_{set} is increased, the u oscillations are smaller in amplitude and also further away from the lower 6.25 % bound, u would then be saturated less often which could explain why stationary error decreases with Qb_{set}.

The stationary error is the reason why a master software controller is needed even though the hardware motor speed controller has integral action.

Period time variation

The measurements show that the 4-bit controller alone fulfills the requirement std(T_{pR}) ≤ 0.1, but std(T_{pR}) is still not optimal and can be reduced even further by using other controllers, this will be covered later. High tacho frequency variance at lower blood flows creates more randomness and variability in the pump period time. The fact that std(T_{pR}) decreases with blood flow is a result of the reduced high frequency disturbance on y at higher blood flows. A hypothesis is that the T_p variation may be reduced by finding a controller that does not amplify the variations in y.

Step responses

Since the settling time decreased at higher blood flows, the system dynamics seem to change with blood flow. The root cause of this change is unknown.

Software Controller

The introduction of a software master controller increases the period time variation, which can be seen in Table 4.5. The added variation is most probably a result of the variations in Qb_{setAct}. The 4-bit controller together with a software contoller does not fulfill the requirement std(T_{pR}) ≤ 0.1 at all blood flows.

The conclusion is that it would have been beneficial if the motor speed controller would not have a stationary error since the master controller adds variation in pump period time.

Both the settling time and overshoot decreased significantly when the integrator reset was removed. The reset increases the time it takes for the software controller to adjust to the new Qb_{set} level. Since the integrator reset has a negative impact on the step response dynamics it seems reasonable to question if the reset is really needed.

4.4.4 6-bit hardware controller

The hardware controller with 6-bit counters (see Section 4.3) was investigated in the same way as the 4-bit controller. The measurements were made to compare the 4-bit and the 6-bit controller properties and to see if a counter with higher resolution gives better motor speed control. Section 4.3.

Stationary behavior

In Figure 4.17, y and u are plotted in stationary state at $Qb_{set} = 200$ ml/min. Interestingly both u and y are less oscillative than with the 4-bit hardware controller (compare

Figure 4.17 with Figure 4.11). This also holds for blood flows up to 500 ml/min and is illustrated in the discrete fourier transform, see Figure 4.18. The dominant peak amplitude around 38 Hz is lower than in Figure 4.12. The y variance is also lower for 200-500 ml/min with the 6-bit controller than with the 4-bit controller.

Another interesting property was that the 6-bit controller had smaller Qb errors than the 4-bit controller together with the software controller between 200-500 ml/min, see Table 4.6, the reason for this is unknown. It was also possible to see how the controller behaves when two tube load disturbances occur in Figure 4.17, something that was not possible with the 4-bit controller.

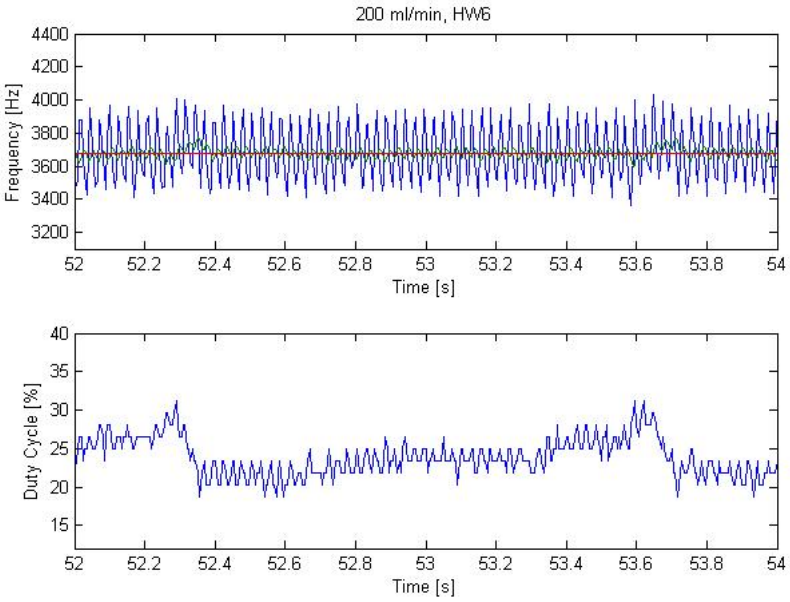


Figure 4.17: Stationary tacho frequency and duty cycle at 200 ml/min with the 6-bit hardware control, this figure has different scale than Figure 4.11.

However, at lower Qb_{set} values $\leq$ 150 ml/min the 6-bit controller behaves differently, the control signal is more oscillative and occasionally y reaches zero. At 100 ml/min the stationary error was 5.22 ml/min. This behavior is displayed in Figure 4.19 and will be further discussed in Section 4.5.

Table 4.6 contains the computed T_{pR} standard deviation for different blood flows, note that the 6-bit controller gives much better period time control than the 4-bit controller except at 100 ml/min blood flow. The standard deviations were computed with 50 period time samples each.

Step responses

y and u were also sampled when 100 ml/min Qb_{set} steps were made. In Figure 4.20, a step response is shown where Qb_{set} goes from 200 to 300 ml/min.

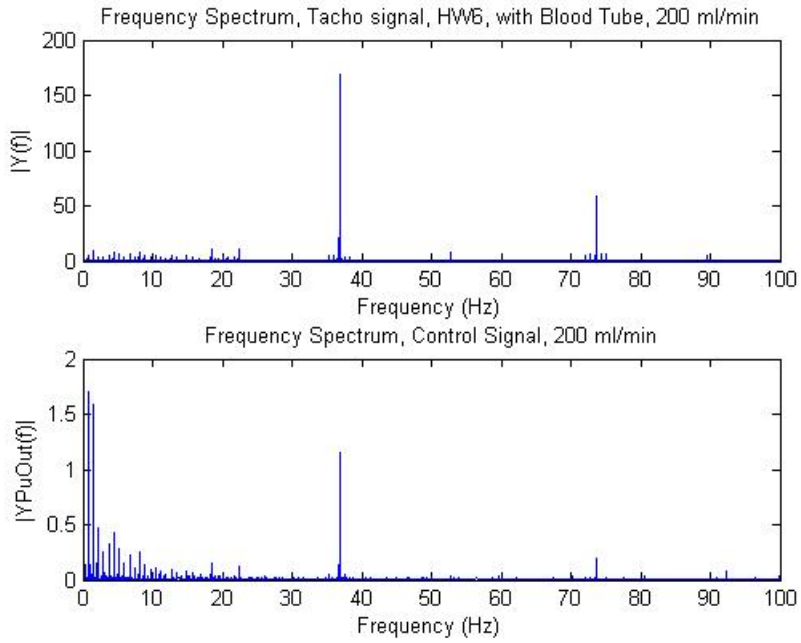


Figure 4.18: DFT of tacho signal frequency y and duty cycle u from Figure 4.17.

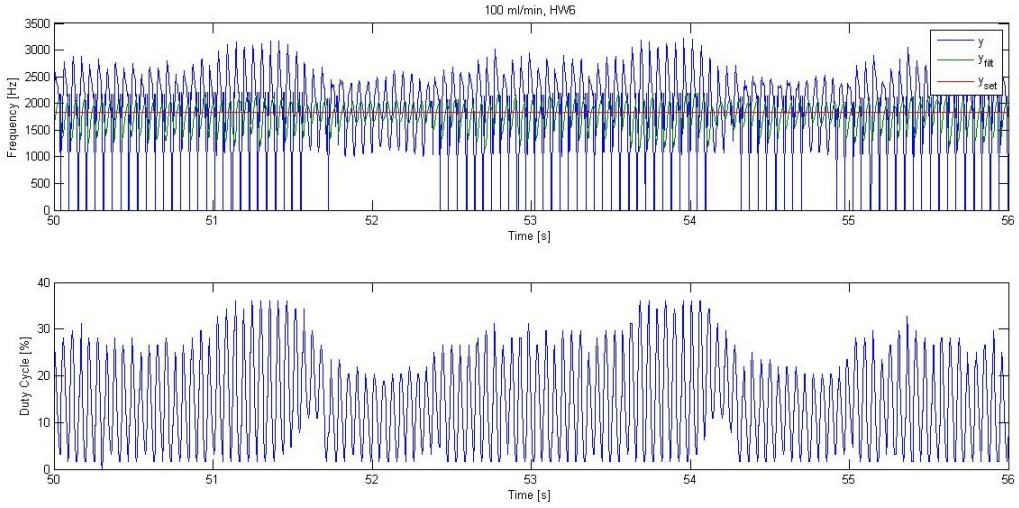


Figure 4.19: 6 seconds of y and u with Qb_{set} at 100 ml/min

In Table 4.7, settling times for various step changes are displayed. Just like the 4-bit controller, settling time is somewhat shorter at higher blood flows. Overall, settling times were longer for the 6-bit controller than the 4-bit controller and overshoot and ringing

Table 4.6: T_{pR} standard deviation, 6-bit and 4-bit controller and stationary error 6-bit controller. The stationary errors are in the most cases negligibly small.

	6-bit controller			4-bit controller	
Qb_{set}	std(T_{pR})	mean(T_p)	stationary error Qb	std(T_{pR})	mean(T_p)
100 ml/min	0.1192	5.3393 s	-5.22 ml/min	0.088	4.9454 s
200 ml/min	0.0215	2.6820 s	-0.0014 ml/min	0.063	2.6035 s
300 ml/min	0.0190	1.7879 s	-0.0091 ml/min	0.075	1.7706 s
400 ml/min	0.0181	1.341 s	0.018 ml/min	0.049	1.3344 s
500 ml/min	0.0175	1.0728 s	-0.018 ml/min	0.03	1.708 s

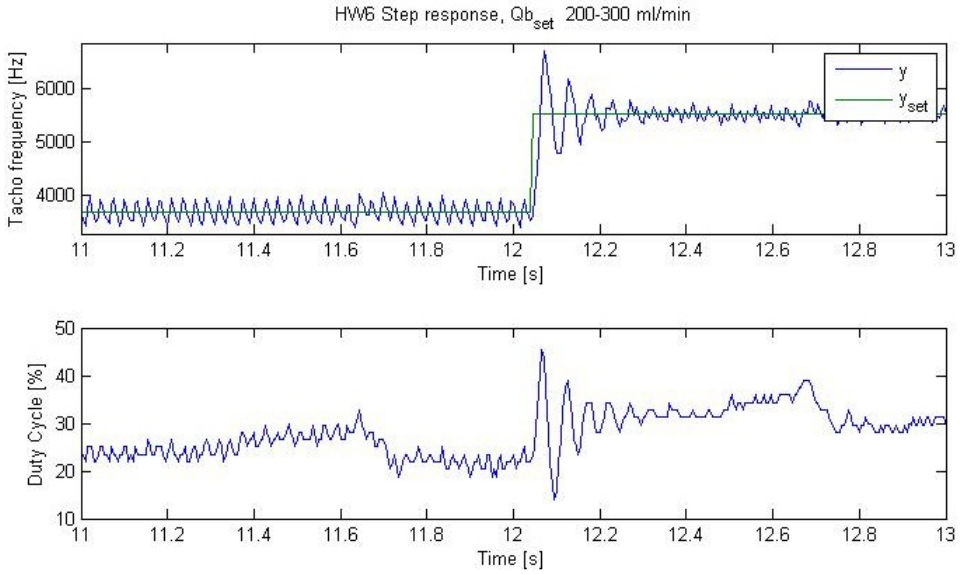


Figure 4.20: 6-bit hardware controller step response where Qb_{set} was increased from 200 to 300 ml/min.

were more noticeable. Compare Figure 4.20 with Figure 4.13.

Table 4.7: Estimated settling times

	6-bit controller	4-bit controller
Blood flow set change	Settling time	Settling time
100-200 ml/min	0.25 s	0.13 s
200-300 ml/min	0.23 s	0.07 s
300-400 ml/min	0.18 s	0.06 s
400-500 ml/min	0.19 s	0.05 s

4.5 Result-discussion

The 6-bit controller provided a less variable pump period time for all blood flows except at 100 ml/min. For 200-500 ml/min the requirement std(T_{pR}) ≤ 0.1 is fullfilled with good margin. The requirement is on the other hand not fulfilled at lower blood flows. The improved period time variation is a result of the reduced y variance, compare Figure 4.11 and 4.17. The integral time of a controller with integral action affects how an oscillating feedback signal is amplified in the feedback loop. For example consider the simple process $P(s)$:

$$P(s) = \frac{K_p}{1 + s\tau},$$

controlled with the analog integrating controller $C(s)$,

$$C(s) = \frac{1}{sTi},$$

in a simple feedback loop. Suppose an oscillative signal n with angular frequency ω is added to the feedback signal, see Figure 4.21. n is then transferred to the output signal y

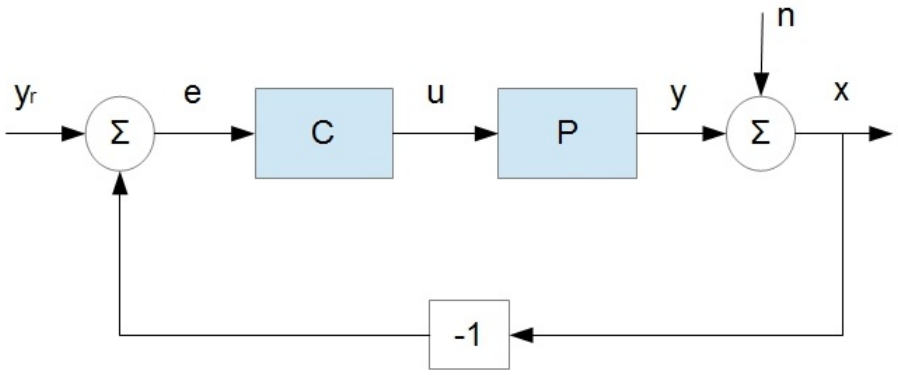


Figure 4.21: A feedback loop where an oscillative signal n is added to the output signal y.

through the complementary sensitivity function $T(s)$;

$$T(s) = \frac{K_p/T_i}{s(s\tau + 1) + K_p/T_i}.$$

n will thus be superimposed on y with an amplitude proportional to the absolute value of $T(i\omega)$,

$$|T(i\omega)| = \frac{K_p/T_i}{\sqrt{(K_p/T_i - \omega^2\tau)^2 + \omega^2}}.$$

When decreasing T_i, the effect of the oscillating signal n on y will be larger for any given frequency ω. [9, p.123]

By this example one could argue that the high frequency y variation will be more amplified in the 4-bit controller feedback loop with the shorter integral time (higher gain), as seen in Figure 4.11 and 4.17.

The 6-bit controller has worse steady state properties than the 4-bit controller at lower blood flows. Figure 4.22 and Figure 4.23 show steady state 100 ml/min blood flow with the 6-bit and the 4-bit controller and software controller.

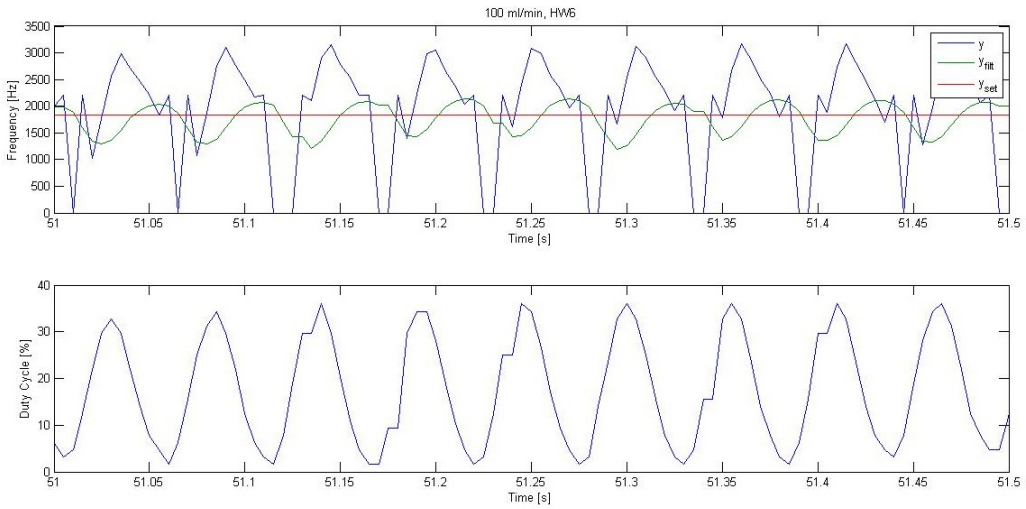


Figure 4.22: 6-bit controller 100 ml/min steady state.

The problems with the 6-bit controller are the high amplitude y oscillation and motor stops. Motor stops can be a result of too low duty cycles and internal fricition, see Section 4.1.6.

The combination of motor stops and integral action can cause stick/slip motion which explains the oscillating 6-bit control behavior. [8][p.333] The reason why this does not happen with the 4-bit controller could be explained by its short integral time, it reacts faster to disturbances and the speed does not reach zero. Also, the 4-bit controller can not actuate as low duty cycles as the 6-bit controller.

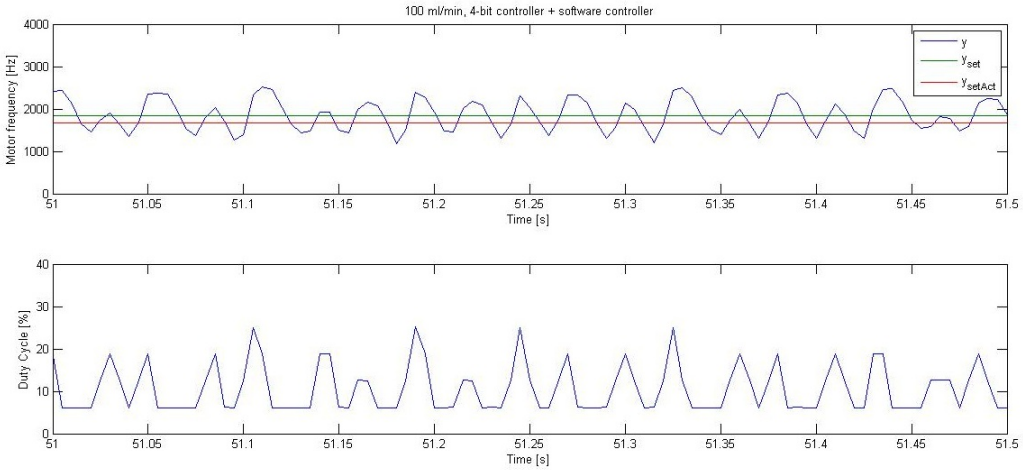


Figure 4.23: 4-bit + software controller 100 ml/min steady state.

Stick-slip motion

If the blood pump gets stuck due to internal friction and tube load resistance, the feedback loop will be broken and the integral action will increase the duty cycle until the motor starts to move again. When this happens the motor speed probably reaches the other side of the speed set point, the integral action then lowers the duty cycle. Unfortunately, the motor gets stuck again by the lowered duty cycle. This phenomenon is called stick-slip motion [8][p.333], and could explain the behavior in Figure 4.22.

Chapter 5

New PI control strategy

5.1 Control principles

The main control objective is to keep the pump period time as constant as possible at stationary blood flow. This will be achieved with good motor speed control based on the hypothesis that the pump period time variance and the motor speed variance are positively correlated. As an alternative to the hardware motor speed controller, a manually tuned discretized PI controller was chosen. The PI controller computes a duty cycle u with motor frequency error $e(t) = y_{set} - y$ as input signal, where y_{set} is computed from Qb_{set}. The idea was that by being able to choose control parameters, K and T_i, freely and by actuating a highly resolved duty cycle, improved motor speed control could be achieved.

The continous PI controller computes the control signal $u(t)$ according to

$$u(t) = K\left(e(t) + \frac{1}{Ti}\int_0^t e(t)\,\mathrm{d}t\right),$$

where $e(t)$ is the control error, the discretized version will be presented in Section 5.4.1. The reasons for chosing a PI controller were that it is a simple, well known, conventional controller, frequently used in the industry. Another reason was that it is a controller that does not demand an accurate mathematical model of the controlled process.

5.1.1 Gain scheduling

From measurements with a manually tuned PI controller it was found that PI control parameters with good control performance at one specific blood flow did not necessarily give good control performance at another. When using a PI controller it would clearly be beneficial to let the control parameters vary with Qb_{set}, i.e. to use gain scheduling. The reason for letting the control parameters vary with the flow setpoint Qb_{set} and not the measured Qb was that Qb_{set} is constant in steady state which ensures constant control parameters. Having varying control parameters in steady state was not a desireable control property. The idea was to optimize the PI controller parameters at each indivudial blood flow in the set of blood flows: 100, 200, 300, 400 and 500 ml/min, and then let the PI

controller parameters vary linearly with Qb_{set} between the found optimized parameters. The gain scheduling implementation is described in more detail in Section 5.6.

5.1.2 Signal processing

Since the feedback controller is used on a process with a high frequency disturbance on the output signal y, prefiltering of the feedback signal is necessary in order to avoid oscillations in the feedback loop. Since the amplitude and frequency of the y oscillations varies with Qb_{set}, a lowpass filter with a cutoff frequency that varies with Qb_{set} was implemented. To get rid of tacho frequency spikes (see Figure 4.8), a spike filter was implemented prior to the lowpass filter in the feedback loop. Filter implementation is described in more detail in Section 5.3.

5.1.3 Rate limit

It was discovered that PI parameters optimized to reject tube load disturbances often gave unsatisfactory, way too aggressive step responses. In order to obtain smoother step responses a rate limit on Qb_{set} was implemented. The rate limit limits how much Qb_{set} can change in a given time interval. Since there are no requirements on Qb_{set} step response time, the allowed Qb_{set} change/second, $max|\frac{\Delta Qb_{set}}{\Delta t}|$, can be chosen arbitrarily small to give smooth step responses with aggressive PI controller parameters. It was decided to set $max|\frac{\Delta Qb_{set}}{\Delta t}|$ to 10 $\frac{ml/min}{s}$.

5.2 Algorithms and implementation

To test a PI controller in our test setup (see chapter 3) we wired the tacho feedback signal to the LabView application. The PXI computer sampled and prefiltered the tacho frequency and then computed a duty cycle with a discretized PI algorithm. The generated duty cycle was wired to the modified pump control circuit board. Qb_{set} was selected by the operator.

5.2.1 LabView program structure

Figure 5.1 shows a block diagram of the LabView PI controller implementation. Each big box represents a timed loop with given execution period time and priority:

- The 'Read' loop has the highest priority (100), and runs with a 5 ms period time. Read samples the tacho frequency y [Hz] and filters the signal with a spike filter and a lowpass filter with variable time constant T [s], the 'Read' loop has the filtered tacho frequency y_{filt} as output.
- 'Rate limit and gain scheduling' has the manually chosen Qb_{set} [ml/min] as input. This loop includes Qb_{set} rate limit and the calculation of PI parameters (K, T_i [s]) and filter time constant T [s] from the rate limited Qb_{set}. Since these computations do not have to be made very often or with strict timing requirements, 'Rate limit and gain scheduling' has the longest period time (100 ms) and lowest priority (65).

- The PI algorithm is executed in the 'Blood pump speed control' loop. The rate limited Qb_{set} is converted to a frequency set value y_{set} that together with the filtered tacho signal y_{filt} defines the control error e. With the PI controller parameters calculated in the gain scheduling block, the PI algorithm computes a duty cycle control signal (in %) that is generated and sent to the pump control circuit board. The control loop has priority 90 and a 5 ms period time.

  The period time h of the read and control loop was chosen as short as possible in order to maximize tube load disturbance rejection at the higher blood floows. Our implementation was unable to run with shorter period times due to the computational load on the CPU. The tacho signal has a frequency around 2 kHz at 100 ml/min which means that it is not meaningful to have a sampling period time lower than 0.5 ms, since then we would sample faster than new information could be obtained, this sets a lower bound on h.

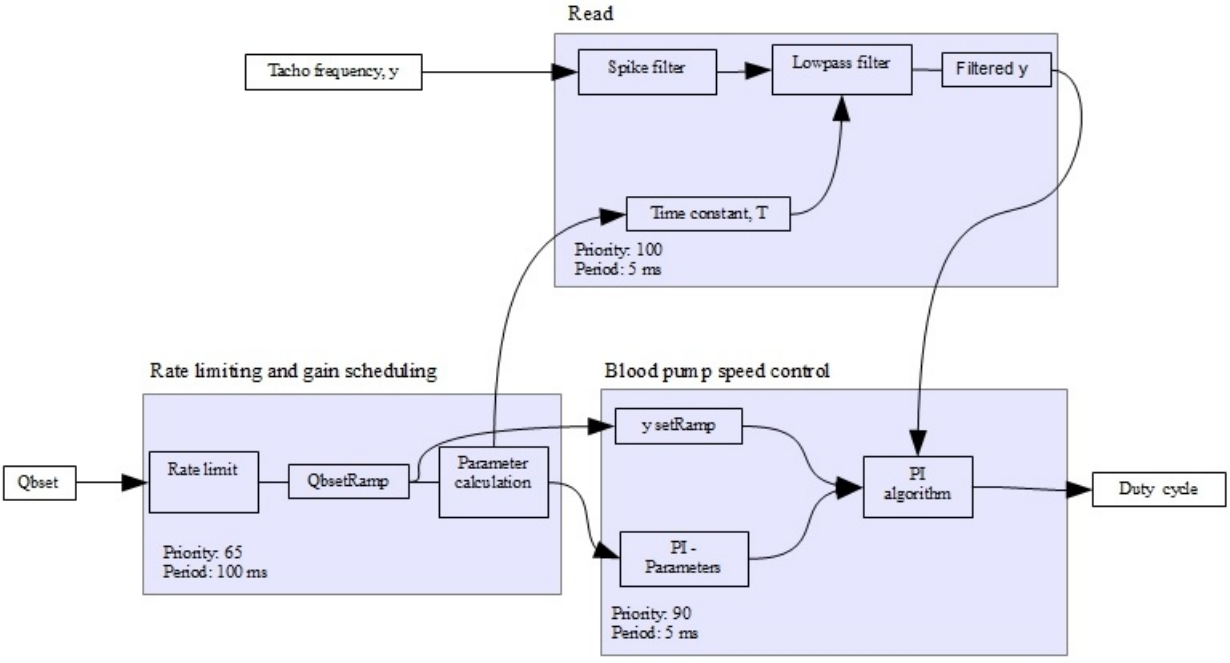


Figure 5.1: PI implementation in LabView.

5.3 Filters

5.3.1 Lowpass Filter

The lowpass filter used to reduce the oscillations in the feedback signal y is a first order IIR (infinite impulse response) filter, with a difference equation obtained by discretizing a continous filter. The reason for chosing a first order lowpass filter was that this allowed for changing of the filter time constant T during runtime, something that would have been more difficult to implement with higher order filters in LabView.

The contionous time filter transfer function is given by

$$F(s) = 1/(sT + 1).$$

The filter cutoff frequency f_c in Hz is given by $\frac{1}{2\pi T}$ and has to be chosen so that most of the oscillating signal components are removed. A problem is that the filter works as a delay in the feedback loop. This means that a too low cut off frequency would result in poor tube load disturbance rejection.

In order to implement the filter digitally, the continous filter was discretized using the forward difference approximation:

$$\frac{dy}{dt} \approx \frac{y(t+h) - y(t)}{h}.$$

The filter above in continous time:

$$T\frac{\mathrm{d}y}{\mathrm{d}t} + y(t) = u(t).$$

Is discretized to

$$T\frac{y(t+h) - y(t)}{h} + y(t) = u(t),$$

which gives the IIR filter

$$y(t+h) = \frac{h}{T}u(t) + (1 - \frac{h}{T})y(t).$$

The forward approximation accuracy is dependent on the factor $\frac{h}{T}$, and gets less accurate when $\frac{h}{T}$ gets larger.

5.3.2 Spike filter

The properties of a tacho frequency spike are short time duration and very high amplitude. This makes it easy to determine a spike by detecting a sudden large change in y. The following pseudocode describes how the filter works:

```
y(n) = new tacho frequency value;

y(n) = max(min(y(n),100000),0);      // (range limit)
```

```
if (|y(n)-y(n-1)| > largeValue) {
y(n) = median([y(n-1),y(n-2),...,y(n-10)]);
} else {
do nothing(); }
```

When a spike (large y change) is detected, the spike y value is replaced by the median value of the ten last sampled (non-spike) y values. Depending on how $largeValue$ is chosen, practically all spikes can be eliminated. $largeValue$ was set to 2500 Hz in our implementation.

5.4 PI controller algorithm

The PI algorithm block was a PID Virtual Instrument (VI) written by our Gambro supervisor. The block uses a discrete PID algorithm with I part discretized with forward difference approximation, explained in Section 5.4.1 (and D part with backward difference approximation). The PID block has the following inputs:

- Setpoint, y_{set}.
- Feedback value, y_{filt}.
- Sampling interval, $h = 0.005$.
- (Feed forward signal).
- (External control signal).
- Proportional gain K.
- Integral time, T_i (s).
- (Derivative time, T_d (s)).

and control signal u as output. The PID VI also provides the following functionalities:

- Gain scheduling - the ability to change controller parameters during runtime.
- Integrator windup protection, using tracking.
- Output range limit.
- (Feed forward from external input).
- (Derivative action with limited high frequency gain).
- (Output reset).
- (Output rate-of-change limit).

Inputs and functions in parenthesis were not used when implementing the discrete PI controller. A pseudocode of the block will be shown in Section 5.4.2.

5.4.1 The discrete PI controller algorithm

The PI controller

$$u(t) = K\left(e(t) + \frac{1}{Ti}\int_0^t e(t)\,\mathrm{d}t\right),$$

was implemented using the discrete algorithm:

$$u(t_n) = Ke(t_n) + I(t_n).$$

Where t_n is the time at sample n, for times inbetween samples $t_n \leq t < t_n + h$ the constant $u(t_n)$ is actuated. For simplicity, notation t will from now on be used instead of t_n. The integral part $I(t)$ approximates the integral of errors with an accumulated sum of errors:

$$I(t) = \frac{K}{T_i}\sum_{i=0}^{n} e(h \cdot i)h$$

This corresponds to using the forward difference approximation of the I derivative:

$$\frac{dI}{dt} \approx \frac{I(t+h) - I(t)}{h}.$$

The integral term I can thus be computed recursively:

$$I(t+h) = I(t) + \frac{Kh}{T_i}e(t).$$

After computation of $u(t)$, $u(t)$ is bounded by a range limit. Since duty cycle has to be in the interval $[0, 100]$, u was limited to stay in the range $[0.01, 0.99]$. Duty cycle 0 % and 100 % were not allowed in the LabView implementation due to a limitation in the counter/timer output driver.

When saturating a control signal actuated by an integrating controller, undesired integrator windup effects can occur. To avoid this, an anti-windup protection has to be implemented, e.g tracking.

Tracking is implemented by adding an extra feedback path from the actuated (possibly saturated) control signal u to the controller. The control signal saturation error $e_s = u - v$ is then computed, where v is the computed unsaturated control signal. The I update is then modified to:

$$I(t+h) = I(t) + \frac{Kh}{T_i}e(t) + \frac{h}{T_t}(u(t) - v(t)).$$

The extra term is zero when u is not saturated, but when saturated the term acts to keep the control signal saturation error e_s at zero. v will then be kept at the saturation limit. T_t is the tracking time constant and was set to h, in order to get deadbeat tracking.

5.4.2 PI VI pseudocode

The following pseudocode executes once every blood pump speed control loop iteration. First the latest computed y_{filt} value is called from the read loop.

```
e = y_set - y_filt;              //control error
P = K*e;            //proportional part
v = P + I;            //computed control signal
u = max(min(v,umax),umin);      //control signal output from the PI block

e_s = u-v;
// I update:

if( T_i <= 0 ){           //to avoid division by zero.
I = (umin+umax)*0.5;
else {
I = I + K*h/T_i*e + e_s;
}
```

5.4.3 Ratelimit

The ratelimit block algorithm is described in the following pseudocode:

```
maxinc = 10;       // Maximum allowed change/second

maxincit = maxinc * h;    // Maximum allowed change / loop iteration

increment = Qb_set - Qb_setold;     //  Qb_set change.

increment = min(max(increment,-maxincit),maxincint);     // change saturation

Qb_setAct = Qb_setold + increment;     // output

Qb_setold = Qb_setAct;
```

5.5 PI controller and filter tuning

When the sampling interval is chosen sufficiently small, the discrete PI controller can be tuned as its continous counterpart. [7][p.55]

The tuning was based on measurement observations at 100, 200, 300, 400 and 500 ml/min, rather than using model assumptions or existing tuning methods. The control parameters K, T_i and T were chosen to fulfill the following performance requirements (listed in order of priority):

- A stable and robust control loop.
- Small pump period time variation.
- Good rejection of tube load disturbances.

5.5.1 Period time variance minimization

Measurements have shown that there are several factors that seem to affect period time variation. The most significant discovered factors are presented in this section:

Tube load disturbance rejection

One factor is the controller's ability to reject tube load disturbances. A controller with good disturbance rejection also seems to keep a more constant period time. During a half pump revolution the control error has a dip and fast peak due to blood tube load disturbance, see Figure 4.5. Experiments have shown that it is easier to reject the disturbances with a high proportional gain K than a short integral time T_i. A short T_i results in a larger overshoot when the disturbance level quickly decreases. The reason is that the proportional controller part better follows quick control error variations.

Phase margin

Another factor that seems to have a negative influence on period time variation is the ringing in y and u when large controller gains K or short integral times T_i are used. Too aggressive control can even result in self-oscillation in the feedback loop, which has a devastating effect on period time variance. The hypothesis is that these are results of a too low phase margin in the process open loop. Smaller phase margin gives a larger ringing amplitude, and in a PI controller feedback loop, phase margin decreases with controller gain K and $\frac{1}{T_i}$. The filter (that can be seen as a time delay) also affects the phase margin. Larger filter time constants T gives increased phase lag. Another negative effect of having a too large K is the fact that the unwanted high frequent content in y becomes more amplified in the feedback loop.

Filter time constant T

A third observed factor that effects period time variation is the filter's ability to reduce high frequency y content, see Section 4.1.4. With a too low T the oscillations are not attenuated sufficiently, this gives a very oscillative control signal u. Having a too large T on the other hand gives a longer delay in the feedback loop which leads to a worsened tube load disturbance rejection. This increases the period time variation. At each blood flow, the filter time constant is a trade-off between attenuation of high frequent disturbances and low frequent load disturbance rejection.

Table 5.1: K, T_i and T choices.

	100 ml/min	200 ml/min	300 ml/min	400 ml/min	500 ml/min
K	0.05	0.05	0.03	0.03	0.02
T_i	1 s	1 s	1 s	1 s	1 s
T	0.318 s	0.159 s	0.0318 s	0.0159 s	0.0106 s

5.5.2 Tuning procedure

Below, a brief overview is presented of the parameter tuning procedure at 100, 200, 300, 400 and 500 ml/min. The parameters are displayed in Table 5.1. Basically the same tuning approach was used at all blood flows; first T was chosen to give an acceptable attenuation of the high frequent disturbances. K and T_i were then chosen to optimize the tube load rejection without too much ringing.

100 ml/min

At this blood flow, y oscillates around 20 Hz with high amplitude. This implied that a long T should be chosen. Good tube load rejection was then achieved with a relatively high K and a long T_i.

200 ml/min

At 200 ml/min the y oscillations are more high frequent and have lower amplitude. This allows for a shorter T. Good tube load rejection was attained with the same PI control parameters as for 100 ml/min.

300 ml/min

The y oscillations are now even more high frequent and not as dominating as for 100-200 ml/min. This allows for an even shorter T. Optimized tube load rejection was then attained for a somewhat lowered K and the same T_i as for the previous two blood flows.

400 ml/min

At 400 ml/min, T could be even lower while still obtaining sufficient oscillation attenuation. The best measured period time variation was obtained with the same PI parameters as for 300 ml/min.

500 ml/min

With the same parameters as at 400 ml/min, the phase margin became too low and ringing occured in y. K and T where then lowered to increase phase margin and to get less variation in y.

5.6 Gain scheduling implementation

With the obtained parameters, the gain scheduling block in Figure 5.1 could be implemented. The block maps the rate limited Qb_{set} (in this section denoted Qb_{set}) to a set of parameters K, T_i and T. In Figure 5.2 and 5.3 the graphs of K and T against Qb_{set} are displayed while $T_i = 1$ for all blood flows. The gain scheduling block is executed once every 100 ms. The computations are only made when the ramped Qb_{set} is changed.

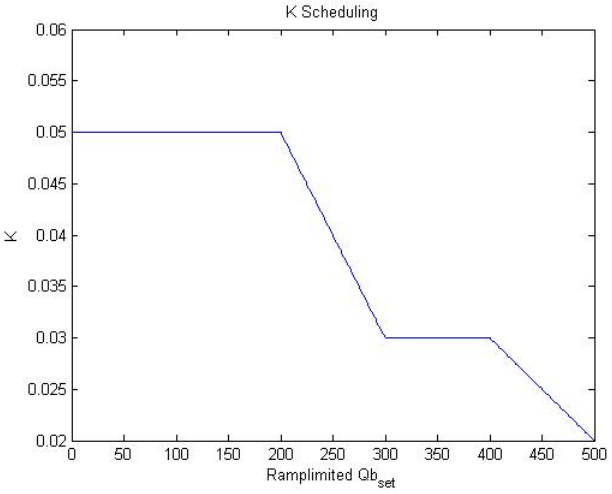


Figure 5.2: K gain scheduling.

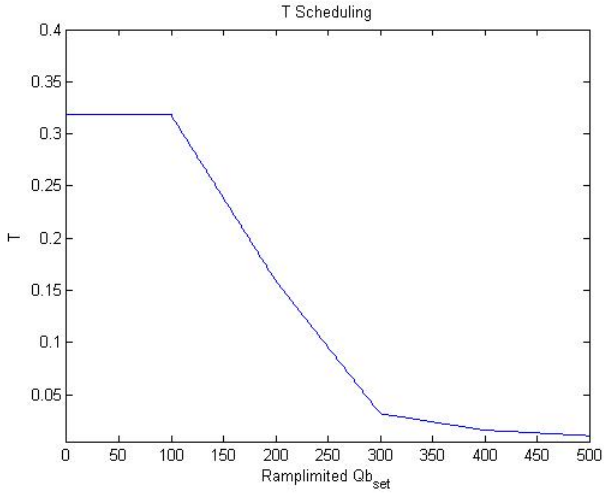


Figure 5.3: T gain Scheduling.

5.7 Performance

In this section, properties of the PI controller are evaluated. In Section 5.7.1 the stationary properties, period time variation, tube load disturbance rejection and stationary error are investigated. Section 5.7.2 describes Qb_{set} change dynamics, and Section 5.7.3 handles controller robustness.

5.7.1 Stationarity

To test stationary properties, 5 minute measurements were made at 100, 200, 300, 400 and 500 ml/min at a sampling interval of 5 ms. Data from one full pump revolution for each blood flow can be seen in Figures 5.4, 5.5, 5.6, 5.7 and 5.8. In the figures 6.4-6.8, y is plotted together with y_{filt}, y_{set} and u. In all plots, frequency and duty cycle are plotted in the same range.

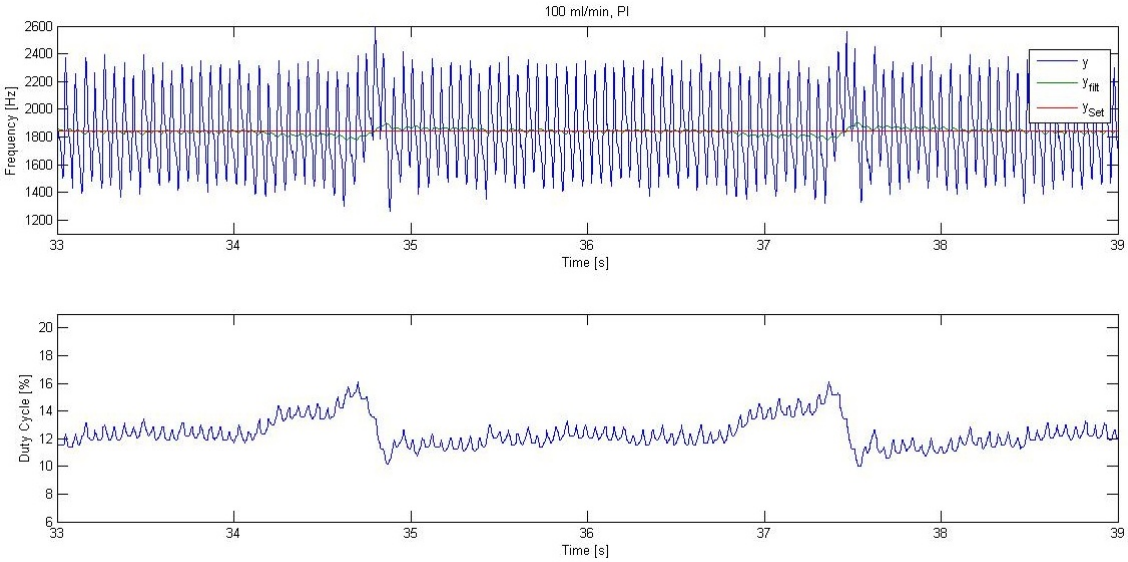


Figure 5.4: PI Control, stationarity at 100 ml/min.

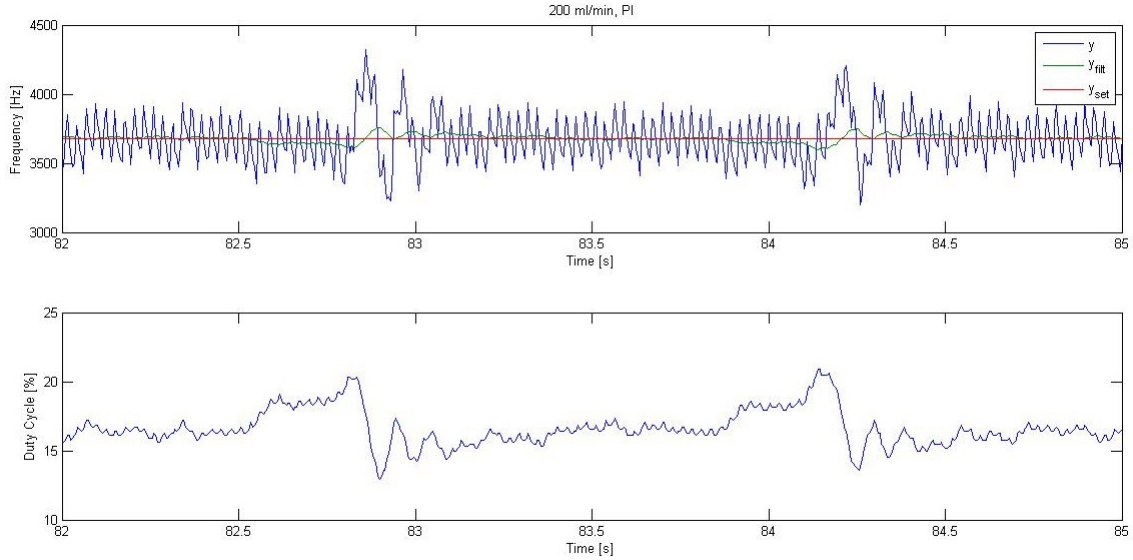


Figure 5.5: PI Control, stationarity at 200 ml/min.

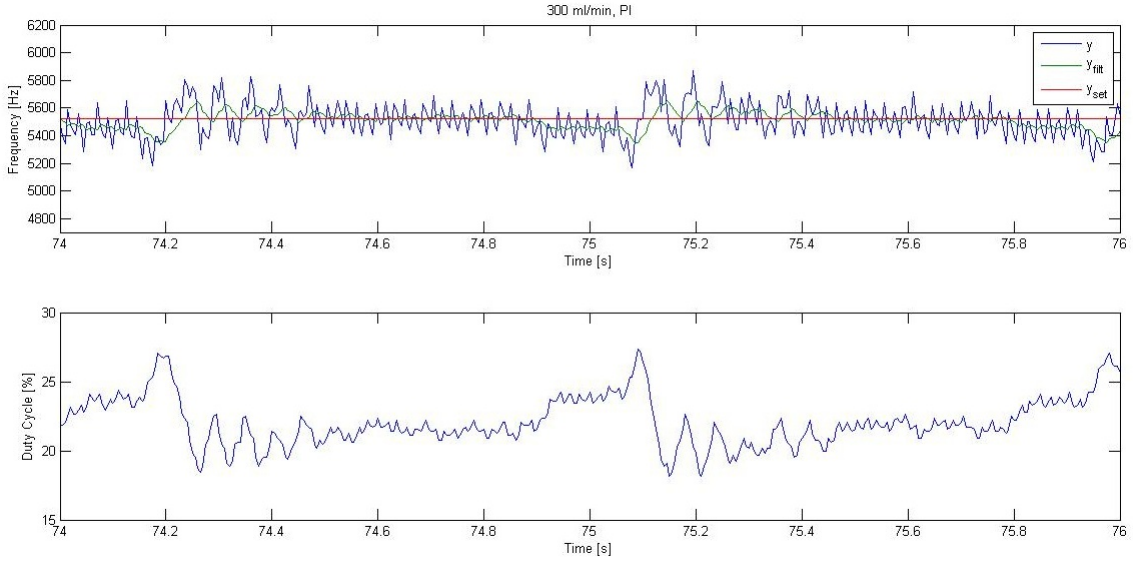


Figure 5.6: PI Control, stationarity at 300 ml/min.

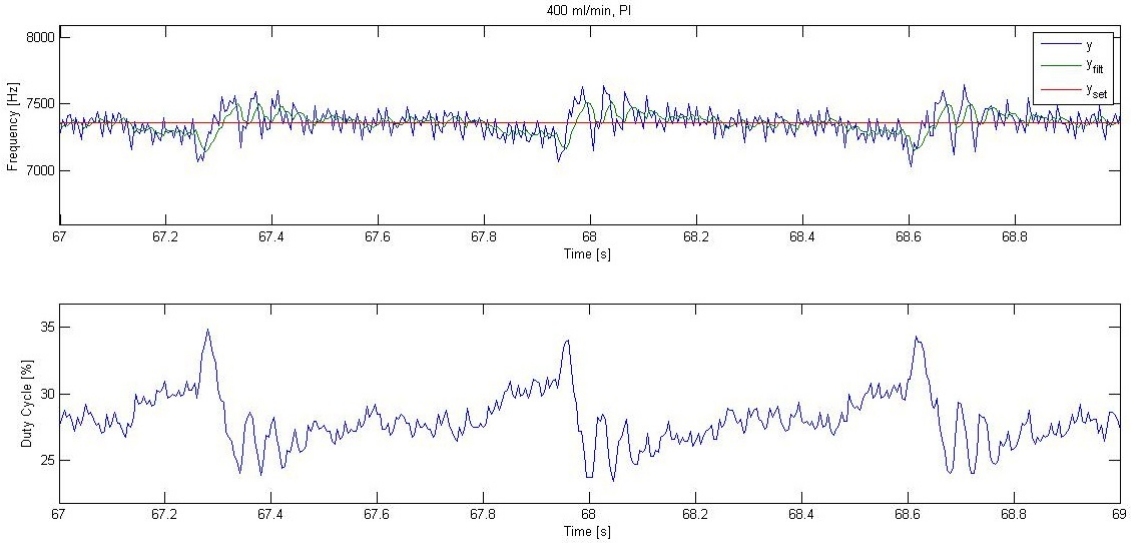


Figure 5.7: PI Control, stationarity at 400 ml/min.

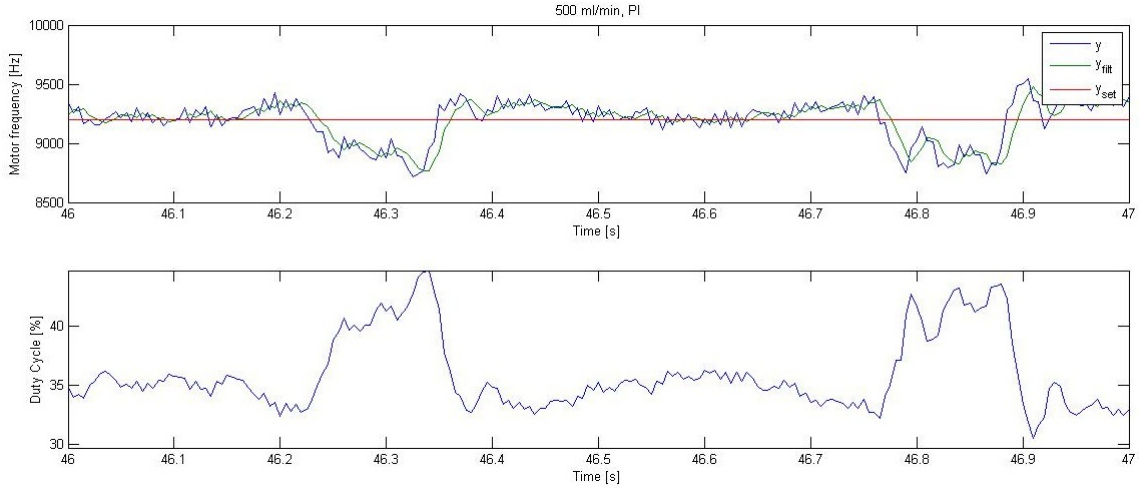


Figure 5.8: PI Control, stationarity at 500 ml/min.

Tube load disturbance rejection

Precisely as in the measurements from open loop and with the hardware controllers, the y oscillation increases in frequency and decreases in amplitude with Qb_{set}. The filter time constant T is chosen so that the oscillations barely are transferred to the control signal, this can be seen in the figures. T affects how well the filtered signal y_{filt} follows the motor speed signal y, which in turn determines how well the tube load disturbances can

Table 5.2: T_{pR} standard deviation and stationary Qb error, PI controller. The stationary errors are negligibly small.

Qb_{set}	std(T_{pR})	mean(T_p)	stationary error
100 ml/min	0.0843	5.369 s	0.0895 ml/min
200 ml/min	0.0221	2.682 s	-0.0017 ml/min
300 ml/min	0.0218	1.788 s	0.0008 ml/min
400 ml/min	0.0215	1.328 s	0.0008 ml/min
500 ml/min	0.0201	1.0728 s	0.0064 ml/min

be rejected. When comparing the control at 200 ml/min and 300 ml/min, the oscillation amplitude and frequency at 200 ml/min 'forces' T to be set high. This gives bad following of y in y_{filt} and poor disturbance rejection. At 300 ml/min T can be set lower, which gives better following of y in y_{filt}, and much better disturbance rejection. At 400 ml/min, the disturbance rejection is even better. It is clear that there are ringing effects in y at 300 and 400 ml/min, this could be taken care of by lowering the controller gain, but at the expence of worsened disturbence rejection.

Pump period time variation

In Table 5.2, the T_{pR} standard deviations for the sampled data are listed. The standard deviations were computed with 80 period time samples each. The standard deviation is highest at 100 ml/min. The variance of the motor frequency is also highest at the lower blood flows, a variance that the PI controller cannot do much about, since it is too high frequent to be controlled. At 200-400 ml/min the relative period time standard deviation is lower together with the motor frequency variation. When comparing the disturbance rejection at 200 and 400 ml/min, it is better at 400 ml/min, but there is just a small improvement in std(T_{pR}). It is therefore hard to say how much std(T_{pR}) can be improved by just improving disturbance rejection.

Stationary error

The largest measured Q_b stationary error was obtained at 100 ml/min and was < 0.1 ml/min, stationary errors are presented in Table 5.2. Since the controller has integral action the stationary error should be zero, but the tube load disturbances lower the motor speed, there will be an average Qb error over time.

5.7.2 Step response dynamics

In Figure 5.9, a series of step responses are plotted. 100 ml/min Qb_{set} steps where made and Qb_{set} goes from 100-500 ml/min. The settling time was estimated to 10 seconds for all steps, something that was expected with the chosen $max|\frac{\Delta Qb_{set}}{\Delta t}| = \frac{10ml/min}{s}$. In

Figure 5.10, the step when Qb_{set} goes from 100-200 ml/min is zoomed in. The duty cycle pulsations are caused by tube load disturbances.

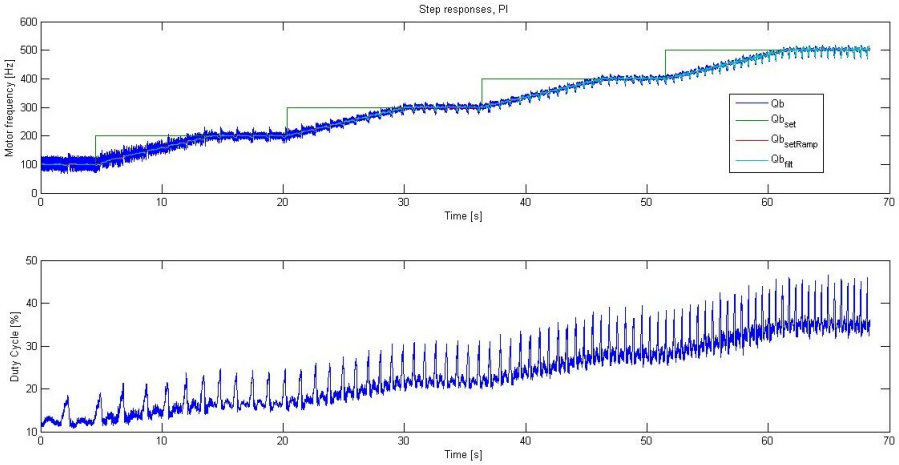


Figure 5.9: Step responses where 100 ml/min steps are made in Qb_{set}.

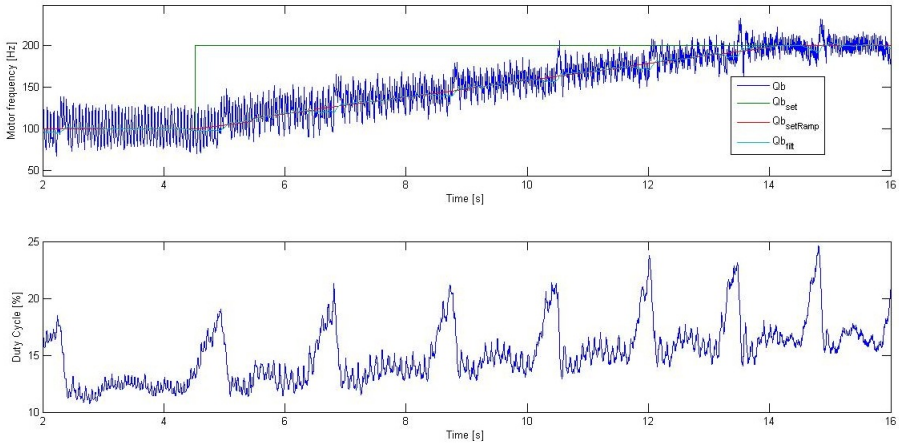


Figure 5.10: The zoomed in 100-200 ml/min step from the picture above.

5.7.3 Robustness

A common source of process variation for the blood flow control system during hemodialysis treatments is the needle size. The needle size affects blood flow resistance and determines the pressure difference over the blood pump for a given blood flow. The blood pump creates a positive pressure after the pump (venous side) and a negative pressure before the pump (arterial side).

Measurements were done with needle sizes 1.8 mm, 1.7 mm and 1.3 mm. The data in Table 5.3 illustrates how the needle affects the blood pump pressure difference at 300 ml/min blood flow. It was found that the needle did not have any significant effect on the

Table 5.3: Mean pressure PrA on the arterial side and venous side PrV with varying needle thickness, at 300 ml/min.

Needle thickness	PrA	PrV	std(T_{pR})
1.8 mm	-44 mPa	61 mPa	0.0205
1.7 mm	-74 mPa	80 mPa	0.0190
1.3 mm	-137 mPa	135 mPa	0.0197

PI control performance. The T_{pR} standard deviations for the different needle thicknesses are displayed in Table 5.3.

5.8 PI controller result discussion

The best found PI controller with the sampling interval 5 ms was a proportional controller with weak integral action. The PI controller was superior to the currently used control system in terms of pump period time variation, motor speed variation and stationary error. The standard deviations fullfills the performance requirement $\leq 0.1\%$ at all blood flows. The PI std(T_{pR}) was not better than the 6-bit controller, except at lower blood flows where the 6-bit controller seems to suffer from stick-slip motion effects. In Table 5.4, the controllers are compared in terms of period time variation.

Table 5.4: T_{pR} standard deviation, 6-bit, 4-bit+software control and PI controller.

	6-bit controller		4-bit + software controller		PI controller	
Qb_{set}	std(T_{pR})	mean(T_p)	std(T_{pR})	mean(T_p)	std(T_{pR})	mean(T_p)
100 ml/min	0.1192	5.3393 s	0.1296	5.3711 s	0.0843	5.3690 s
200 ml/min	0.0215	2.6820 s	0.0949	2.687 s	0.0221	2.6820 s
300 ml/min	0.0190	1.7879 s	0.1066	1.7928 s	0.0218	1.788 s
400 ml/min	0.0181	1.341 s	0.0758	1.343 s	0.0215	1.3281 s
500 ml/min	0.0175	1.0728 s	0.0466	1.736 s	0.0201	1.0728 s

The PI controller tests show that sufficiently good control properties can be achieved with a PI controller. The tests also show that tacho frequency filtering is necessary and that filter tuning is a vital part in order to reach good control properties.

5.8.1 Derivative action

Control performance could probably be improved by introducing derivative action, i.e. by using a PID algorithm. This was in fact tried out and evaluated as a part of the project. However, the found performance improvements were too small in order for the PID control alternative to take part in this report.

Chapter 6

Feed forward

6.1 Background

Relatively large disturbances in blood flow come as a result of the two rollers on the peristaltic pump pinching the tube twice per revolution. Fortunately, since the rollers are placed exactly opposite to each other, the disturbances come periodically with a time interval which is half the period time of the rotor. Therefore, a simple feed forward signal was introduced in order to improve the tube load disturbance rejection and thereby obtain less variation in motor speed.

The objective was to add a feed forward signal to the control signal and use it together with a PI controller.

6.2 Design

The feed forward loop was implemented by adding a counteracting pulse to the control signal in order to dampen the disturbance induced by the roller, see Figure 6.1. However, before starting with the design of the feed forward control it was necessary to obtain some information of when the disturbances occur during the revolution of the pump rotor.

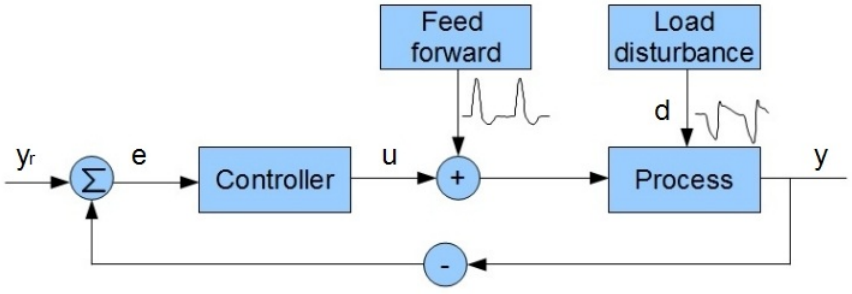


Figure 6.1: Block diagram showing the feed forward realization in the control loop.

The period times were measured using a hall sensor and two magnetic pieces placed opposite to each other in the rotor. The hall sensor was located next to the pump rotor and changed state as the magnets passed during revolutions. Hence, the time it took for the sensor to change state two times was equivalent to one period time. The placement of the magnets in the rotor relative to the rollers was determined simply by plotting the measured period time together with the tacho frequency in open loop at a duty cycle of 18 %. From this analysis it was concluded that the first disturbance occurs approximately after 16 % of the period time and the second after approximately 66 %, see Figure 6.2.

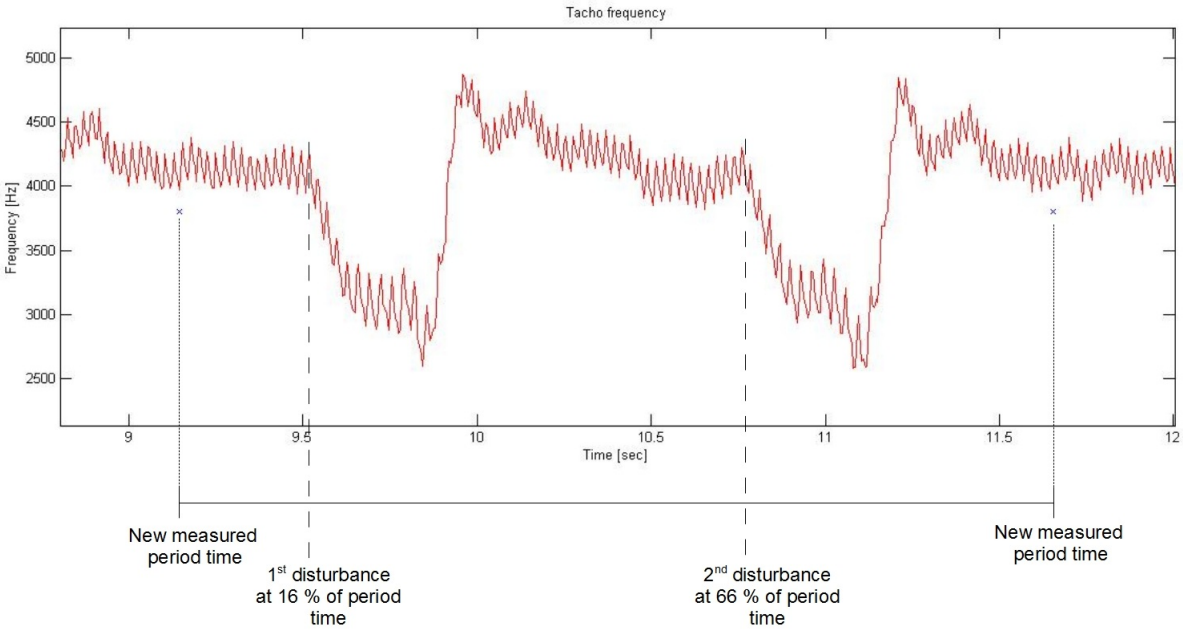


Figure 6.2: Determination of disturbance occurence relative to period time measurements.

In order to effectively suppress the disturbances, a counteracting pulse in the control signal with similar, but opposite, appearance had to be found. After considering the behavior of the disturbances it was decided that the feed forward pulse should include two sine waves, one positive and one negative, see Figure 6.3.

The purpose of the first positive sine was to dampen the large dip caused by the first roller pinching the tube and the second slightly smaller, negative sine wave was included to reject the overshoot caused by the second roller detaching from the tube.

The amplitude, phase shift and frequency of each sine wave were given as global variables which enabled the user to change them during run time, see Appendix. Thereby, a fine-tuning of the feed forward pulses was possible where the effect of the changes was immediately recognized in the control signal. Also, the start and stop times of the pulses were implemented as global variables which could be changed during run time.

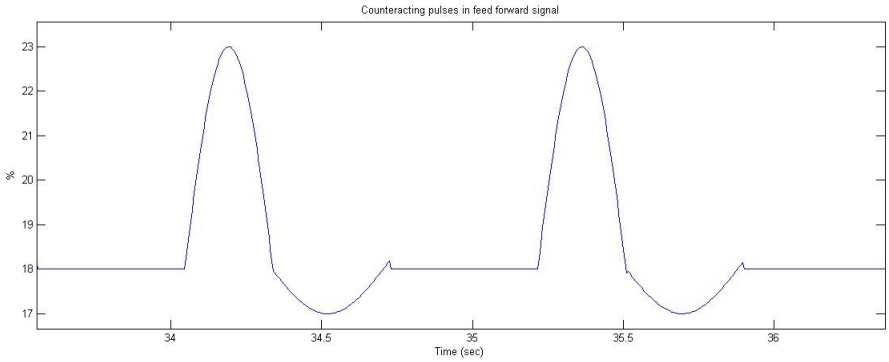


Figure 6.3: Two sine waves were used in the feed forward signal.

The feed forward function was implemented with two VIs in the blood pump speed control loop. Since the sampling interval of this timed loop was not synchronized with the period time measurements it was necessary to implement one VI which flagged as soon as a new period time was received from the period time measurement loop. This flagging VI was then connected to a second VI which contained the calculations needed to make a feed forward signal. Here follows a description of the different parts and variables of the calculating VI:

Period time counter: As soon as a new period time measurement was received, this counter started to count with an interval of 5 ms. The counter was reset when it received a new period time flag. This was to ensure that the feed forward signal was applied at the right time.

Start time: This global variable gave the starting time of a new feed forward pulse, given in percent of the period time, i.e. 16 %.

Stop time: This was also a global variable which stated the stop time of the pulse, given as percent of the period time.

True/false loops: The true/false loops contained the sine wave calculations. When the period time counter was equal to the start time, a true-value was sent to the loop until the stop time was reached. During this time the sine waves were added to the signal. When no true-value was sent, nothing was added to the signal. The second counteracting pulse during a pump revolution was also implemented by a true/false loop, but 50 % was automatically added to the start and stop times.

The implementation of the feed forward signal was effective, but contained one drawback; since we cannot predict exactly what the next period time would be we use the latest measured one. This means that the previous period time is used when the feed forward signal is made, and the period times cannot differ too much in order for this to work.

Table 6.1: K,T_i and T choices from approach two.

	100 ml/min	200 ml/min	300 ml/min	400 ml/min	500 ml/min
K	0.03	0.01	0.01	0.01	0.01
T_i	1 s	1 s	1 s	1 s	1 s
T	0.318 s	0.159 s	0.0318 s	0.0159 s	0.0106 s

6.3 Tuning

To obtain a high suppression of the disturbance, two different procedures were tested, both with the objective to reduce the variation in tacho frequency.

Approach one: The counteracting pulses were first coarsely tuned in open loop mode in order to find the start and stop times of the pulses. Then the amplitude, phase and frequency were further optimized in closed loop using the gain scheduled PI controller found in Chapter 5 in order to further reduce the disturbances.

Approach two: This approach was quite the opposite from approach one. Here, the counteracting pulses were only optimized in open loop. The configurations of the sine waves were then kept fixed in the closed loop mode. As before, a PI controller was used. The optimized PI controller parameters found in Chapter 5 functioned as a starting point and then K and T_i were slightly changed to find a tacho frequency with the lowest variance. Shorter T_i resulted in large overshoots and therefore it was kept at a relatively long time of 1 second. For simplicity the filter time constant was not changed. However, it was found that a lower gain was necessary. Using original K gave a ringing effect in the signal. The resulting PI controller parameters and filter time constants can be viewed in Table 6.1.

6.4 Open loop performance

Figures 6.4 and 6.5 shows the tacho frequency in open loop, before and after the feed forward signal was enabled as well as the corresponding duty cycle for both approaches. In both figures it can be seen that the feed forward implementation takes approximately one period time before it settles in a new less disturbed stationary state. It can clearly be seen that both approaches created a much less disturbed signal. In order to evaluate how much the disturbances was suppressed the standard deviation of the relative tacho frequency was calculated. The relative tacho frequency, y_R is given as:

$$y_R = \frac{y}{\text{mean}(y)} \cdot 100.$$

The standard deviation of y in open loop was calculated to be 22.4488. Using approach one, the same calculation gave a standard deviation of 8.2368 and for approach two it was

6.3790. Since approach two was optimized to give the smallest variation possible in open loop it is expected that it would give a better result than approach one.

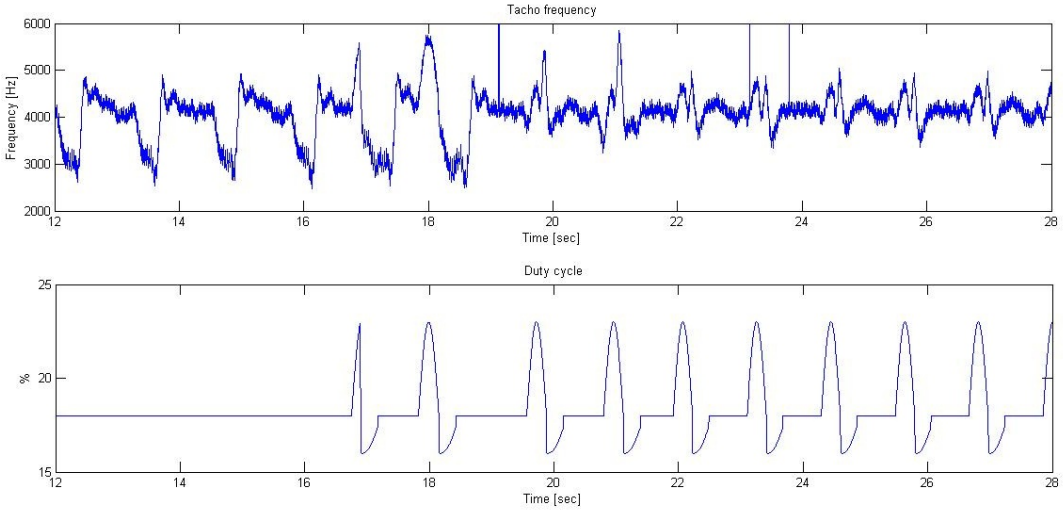


Figure 6.4: Top: tacho frequency before and after feed forward using approach one was enabled. Bottom: corresponding duty cycle. The feed forward was enabled at t = 17 seconds. $Qb \approx 220$ ml/min.

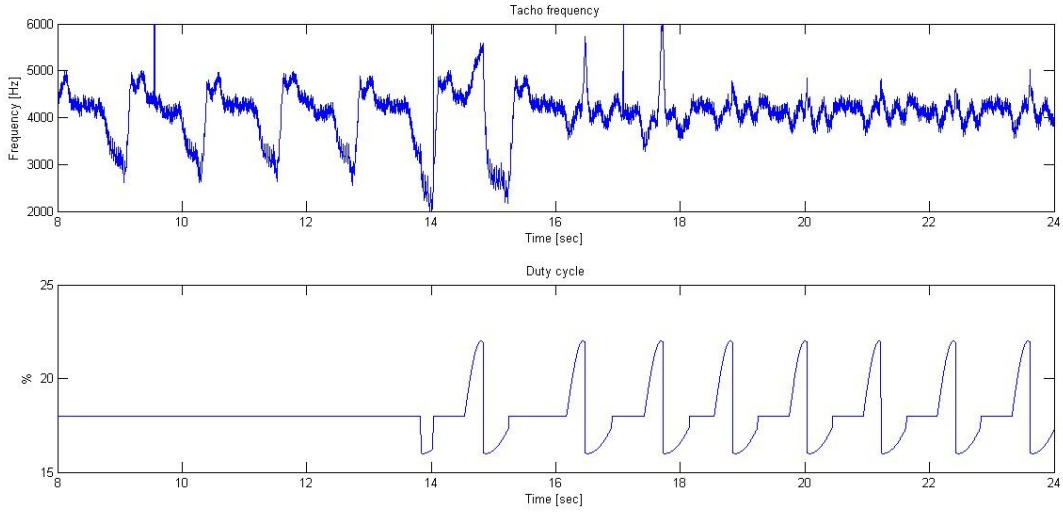


Figure 6.5: Top: tacho frequency before and after feed forward using approach two was enabled. Bottom: corresponding duty cycle. The feed forward was enabled at t = 14 seconds. $Qb \approx 220$ ml/min.

6.5 Closed loop performance

The closed loop performance using the two approaches can be viewed in Figures 6.6 and 6.7 which shows the tacho frequency before and after feed forward was enabled at a flow of 300 ml/min. The figures display the filtered tacho frequency in order to show more clearly the difference between disabled/enabled feed forward signal.

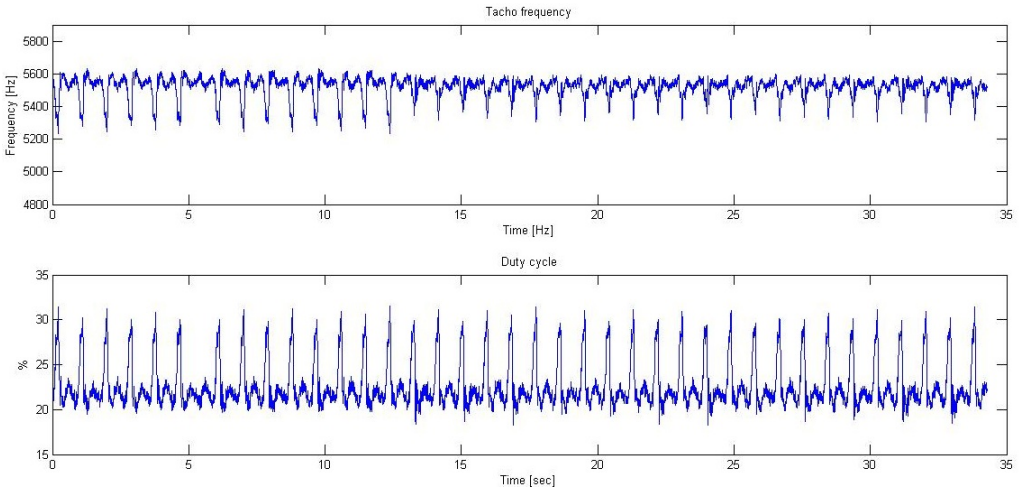


Figure 6.6: Approach one: Closed loop behavior before and after feed forward was enabled. Feed forward was enabled at t = 12 seconds. Qb = 300 ml/min.

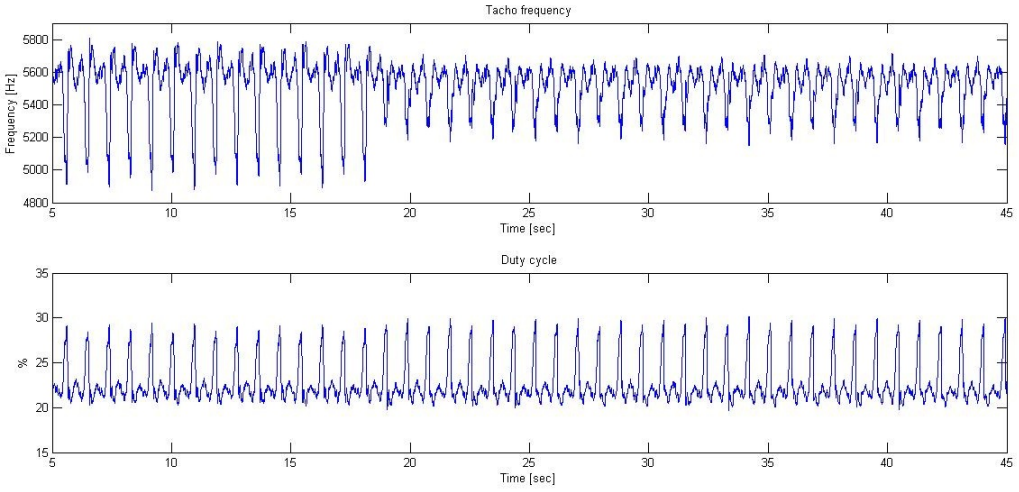


Figure 6.7: Approach two: Closed loop behavior before and after feed forward was enabled. Feed forward was enabled at t = 18 seconds.Qb = 300 ml/min.

The relative standard deviation of the relative tacho frequency in stationary state in closed loop for the two approaches can be viewed in Table 6.2 as well as the PI controller found in Chapter 5. Approach one clearly reduces the tacho frequency variation. However, approach two gave a tacho frequency variation that was even worse at all flows except at 200 ml/min. An optimized feed forward signal in open loop did apparently not give better performance in closed loop.

Table 6.2: Relative standard deviation of relative tacho frequency, y_R, using the PI controller found in Chapter 5 as well as the closed loop feed forward approaches 1 and 2.

Flow (ml/min)	PI controller	Approach 1	Approach 2
100	15.2847	15.6403	19.6778
200	4.6244	4.3086	4.3422
300	2.1413	1.9499	2.2699
400	1.4387	1.2117	1.5781
500	1.4009	1.1160	1.5143

Table 6.3 shows the standard deviation of the relative period time using the PI controller and closed loop feed forward approaches one and two. The standard deviations were slightly improved when using the feed forward, except at low flows. To summarize; the feed forward approach one showed the overall best performance even though only small improvements were obtained.

Table 6.3: Relative standard deviation of relative period time, T_{pR}, using the PI controller found in Chapter 5 as well as the closed loop feed forward approaches 1 and 2.

Flow (ml/min)	PI controller	Approach 1	Approach 2
100	0.0843	0.0889	0.1336
200	0.0221	0.0275	0.0216
300	0.0218	0.0191	0.0197
400	0.0215	0.0186	0.0159
500	0.0201	0.0200	0.0179

We must keep in mind that this feed forward application is probably sensitive to changes in load disturbance and other process changes. Changing the tube set or using blood instead of water are things that could affect the load disturbance. Using the originally chosen parameters for the sine waves, e.g. the amplitude, could therefore be a poor fit to other disturbances.

Chapter 7

Summary and conclusions

The blood pump requires a well designed control system in order to deliver a steady blood flow, although the main obective of this thesis was to maintain a constant pump period time.

The two main control obstacles are that the motor that drives the pump is subjected to blood tube load disturbances and that the measured signal used for feedback is highly oscillative.

The currently used hardware controller with integral action has an insufficient control signal resolution, which results in high pump period time variance. A master blood flow controller used to correct for stationary errors adds extra variance to the pump period time.

We have shown that an increase in the hardware controller control signal resolution gives a more constant pump period time at most bloodflows but shows signs of stick-slip motion at lower blood flows.

We have also demonstrated that a gain scheduled, manually tuned, discretized PI controller together with a lowpass filter fulfill the pump period time goals, with no stationary errors and without stick-slip motion effects.

Using the fact that the disturbances are periodic, a feedforward control stategy can be used in order to get better blood tube load disturbance rejection. This can be used in combination with the PI controller.

Chapter 8

Suggestions for further work

This chapter contains a collection of recommended topics that we think could be further investigated.

Current system

High frequency disturbances

By identifying the source of the high frequency disturbances on the tacho frequency signal y, the disturbance amplitude could hopefully be reduced in some way. This would allow for an improved motor speed control.

Duty cycle resolution efficiency

The 4-bit hardware controller duty cycle was never found above 40 % at 100 ml/min, which means that more than half of the duty cycle resolution is unused. A rescale of the 4-bit control signal could allow for a more efficient use of the duty cycle resolution.

PI controller

Shortened sampling interval h

Our choice of sampling interval h was limited by the computational load on the CPU. It would be interesting to see if better speed control properties could be achieved with a shorter h. This could be done by using a different PXI computer or by optimizing the control algorithm even further.

FPGA implementation

The PI algorithm used in chapter 5 has a too short sampling interval to be implemented in the software of AK 96. To be able to reach short sampling intervals an idea is to implement a PI controller algorithm in a FPGA circuit.

Derivative action

Derivative action could be investigated further, it is possible that a PID controller could give better pump speed control performance than a PI controller even though our experiments did not indicate on this.

Feed forward

The tuning of feed forward signal parameters such as amplitude and start/stop times could ideally be handled by an adaptive algorithm rather than being handled manually. This would make the feed forward control algorithm more robust to process variation. There are probably other, more easily implemented or more efficient feed forward design approaches. An evaluation of alternative feed forward methods should be done prior to a feed forward implementation in AK 96.

Appendix

LabView front panels

Host application

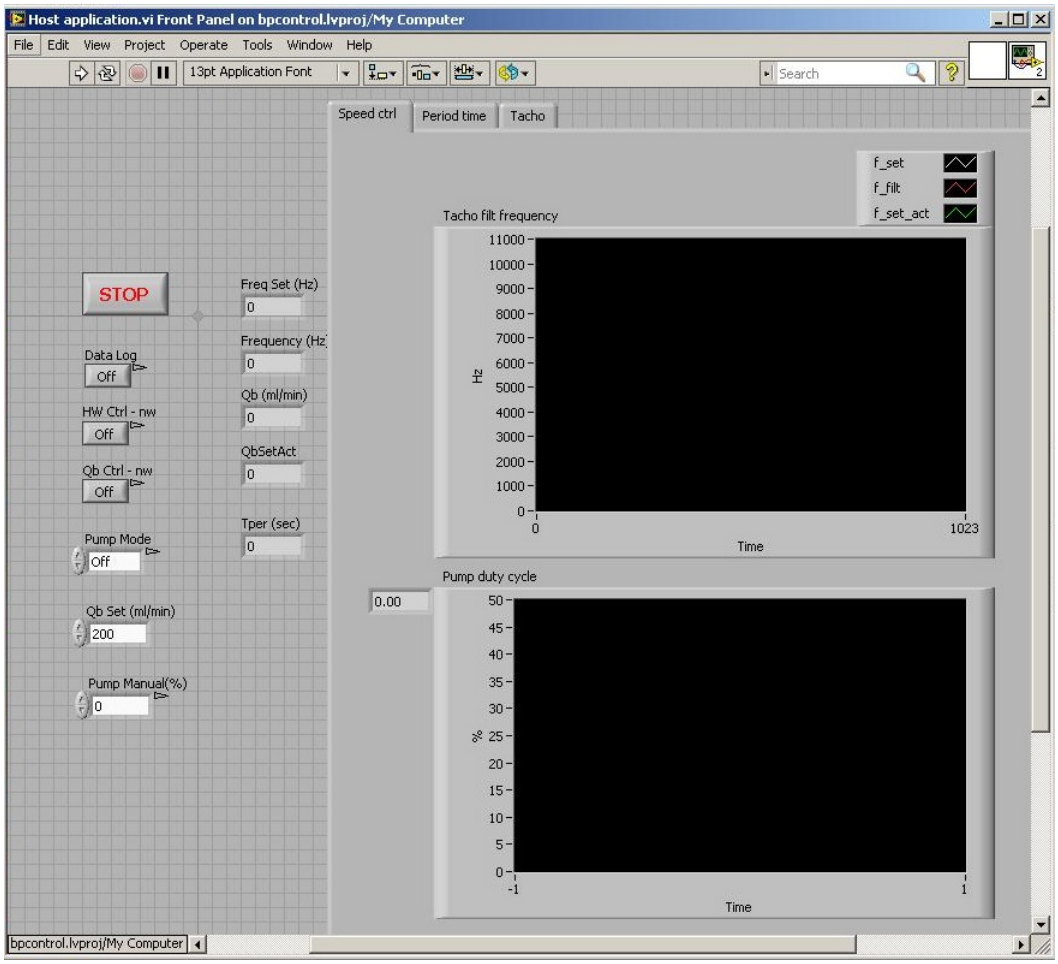


Figure 8.1: Front panel of the host application.

Figure 8.1 shows the graphical user interface used for the host computer, the host VI front panel. Numerous of actions was performed here. Signals such as tacho frequency, duty cycle and period time was shown in the graphs and numeric indicators showed the measured real time value of the set point value, the blood flow, period time, tacho frequency and corrected blood flow. The pump mode was manually selected as well as the blood flow set point and manual duty cycle in open loop. ON/OFF buttons enabled or disabled different controllers and the writing to data log.

Target application

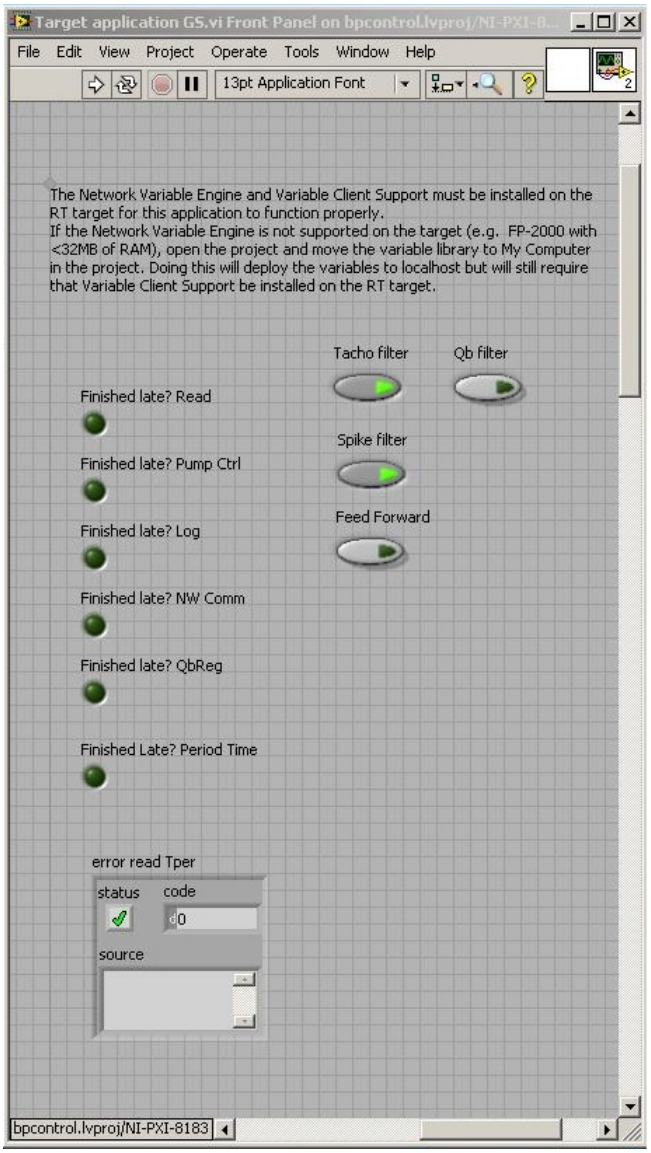


Figure 8.2: Front panel of the target application.

The front panel of the target application can be seen in Figure 8.2. Boolean indicators was flashing with a red light as soon as the time requirement was not fulfilled by the timed loops. When the period time was not successfully measured, a error message appeared in the error box. The filters and the feed forward signal was manually enabled through ON/OFF buttons.

Global variables

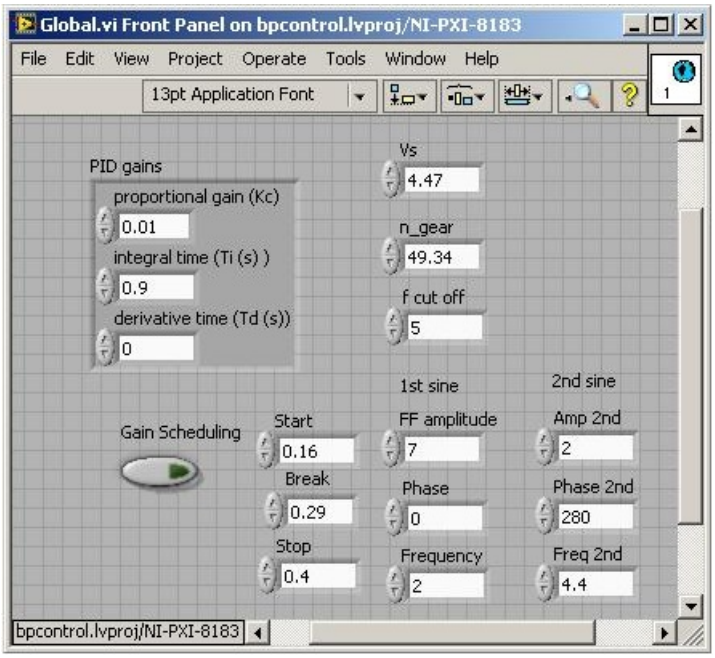


Figure 8.3: Front panel of the global variables.

The global variables could be manually selected during runtime. This was an important feature when fine tuning the PI parameters and the feed forward signal. Figure 8.3 shows the front panel of the global parameters where the PID parameters are shown at the top left. The top right variables give the stroke volume, gear ratio and cut-off frequency of the lowpass filter. The bottom of the front panel shows the variables used when constructing a feed forward signal. An ON/OFF button enabled gain scheduling.

Bibliography

[1] Frederic H. Martini, Edwin F. Bartholomew, *Essentials of Anatomy and Physiology.* 4th ed. Pearson education, 2007

[2] Mattias Aurell, *et al.*, *Njurmedicin.* Liber AB, 2007

[3] Venous needle monitoring, visited November 2012.
http://ckj.oxfordjournals.org/content/4/suppl_2/4.s2.39.abstract

[4] Gunnar Hillerström, Jan Sternby, *Application of repetitive control to a peristaltic pump*, J. Dyn. Sys., Meas., Control 1994 Volume 116, Issue 4, 786

[5] Gambro's official website, visited November 2012.
www.gambro.se

[6] National Instruments LabView webpage, visited November 2012.
sweden.ni.com/labview

[7] Wittenmark, B., Åström, K-J.,Arzen, K-E. *Computer Control: An overview.* Lund, Sweden. Educational Version 2012

[8] Astrom, K-J., Hägglund, T. *Advanced PID Control.* ISA, North Carolina, USA. 2005.

[9] Glad, T., Ljung, L. *Reglerteknik, grundläggande teori* Studentlitteratur, Lund, Sverige. 2006.