

ISSN 0280-5316
ISRN LUTFD2/TFRT--5868--SE

Grating motor control

Henrik Björk

Department of Automatic Control
Lund University
December 2010

| | | | |
|---|-------------------------------------|--|-------------|
| Lund University Department of Automatic Control Box 118 SE-221 00 Lund Sweden | | <i>Document name</i> MASTER THESIS | |
| | | <i>Date of issue</i> December 2010 | |
| | | <i>Document Number</i> ISRN LUTFD2/TFRT--5868--SE | |
| <i>Author(s)</i> Henrik Björk | | <i>Supervisor</i> Magnus Tunklev, Foss, Höganäs Tore Hägglund Automatic Control, Lund (Examiner) | |
| | | <i>Sponsoring organization</i> | |
| <i>Title and subtitle</i> Grating motor control (Reglering av gittermotor) | | | |
| <i>Abstract</i> <p>A monochromator is a part of a spectrometer in which the grating scans the different wavelengths over the exit slit. A full spectrum is generated as the grating scans over a predefined interval. The grating must accelerate to a set speed very fast, then keep the speed constant over a predefined interval, and break to reverse the scan. On top of this come disturbances from an optical filter, causing vibrations in the housing, which can be seen in the spectra if not corrected for in the control algorithm. A model-based control system composed of Proportional-Integral (PI) control, and Iterative Learning Control (ILC), is implemented on a Field-Programmable Gate Array, using LabVIEW. The proposed controller shows promising results, with closed loop rise time within given bounds, and reduced disturbance effects.</p> | | | |
| <i>Keywords</i> | | | |
| <i>Classification system and/or index terms (if any)</i> | | | |
| <i>Supplementary bibliographical information</i> | | | |
| <i>ISSN and key title</i> 0280-5316 | | | <i>ISBN</i> |
| <i>Language</i> English | <i>Number of pages</i> 40 | <i>Recipient's notes</i> | |
| <i>Security classification</i> | | | |

Contents

| | |
|---|----|
| Acknowledgments | 3 |
| 1. Introduction | 4 |
| 1.1 Tasks | 4 |
| 1.2 Outline | 4 |
| 1.3 Methods | 4 |
| 2. Modeling | 5 |
| 2.1 Brief Overview of the Motor | 5 |
| 2.2 Model of the DC Motor | 6 |
| 2.3 Parameters | 7 |
| 2.4 Model Characteristics | 9 |
| 2.5 Pulse-Width Modulation | 11 |
| 2.6 Incremental Rotary Encoder | 11 |
| 2.7 Summary | 13 |
| 3. Control Design | 14 |
| 3.1 Control Objective | 14 |
| 3.2 Reference Trajectory | 14 |
| 3.3 Controller Choice | 14 |
| 3.4 Design | 15 |
| 3.5 Experimental Results | 19 |
| 3.6 Summary | 21 |
| 4. Stepping Motor Disturbance | 22 |
| 4.1 Optical Filter and Stepping Motor | 22 |
| 4.2 Disturbance Properties | 22 |
| 4.3 Disturbance Rejection | 26 |
| 5. Iterative Learning Control | 27 |
| 5.1 Design | 28 |
| 5.2 Nonminimum-Phase Systems | 29 |
| 5.3 Implementation Notes | 30 |
| 5.4 Experimental Results | 31 |
| 5.5 Summary | 35 |
| 6. Conclusions and Future Work | 36 |
| 6.1 Conclusions | 36 |
| 6.2 Future Work | 36 |
| 7. Bibliography | 37 |
| A. Nomenclature | 38 |
| A.1 Abbreviations | 38 |
| A.2 Motor Parameters | 38 |
| A.3 Additional Symbols | 38 |

Acknowledgments

The work leading up to this report was performed between June 2009 and June 2010 at Foss R&D department in Höganäs, Sweden. I would like to thank Prof. Tore Hägglund for giving me the opportunity of doing the thesis at FOSS. I would also like to thank the whole team at FOSS for making my visit a pleasant and inspiring one. Special recognition goes to my adviser Dr. Magnus Tunklev for his support throughout the project, and Mr. Johan Granath for all the interesting discussions. Finally, I am grateful to Mr. Peter Andersson, and Mr. Sven-Anders Nilsson for all their assistance in the laboratory.

Henrik Björk

1. Introduction

FOSS is one of the world's leading supplier of analytical instruments. Notably, 85% of the milk and 80% of the traded grain in the world today is tested by a FOSS solution. In 41 of the 50 largest pharmaceutical manufacturers world-wide, a FOSS solution is used.

The instruments are based on technology from many different fields. Some examples are, infrared spectroscopy, dual x-ray absorptiometry, image analysis, artificial neural networks, and control theory. The underlying principle is, however, to measure light, either transmitted through the sample or reflected from the sample. Instruments based on this principle are called spectrometers.

A monochromator is a part of a spectrometer in which the grating scans the different wavelengths over the exit slit, thus illuminating the detector one wavelength at the time. A full spectrum is generated as the grating scans over a predefined interval. Each instrument must be calibrated and behave in a very similar manner. The monochromator is the core module of the instrument and the grating control is the heart of the core module. The grating must accelerate to a set speed very fast, then keep the speed constant over a predefined interval, and break to reverse the scan. The grating is attached to the shaft of a dc motor, hence the control objective is velocity control of a dc motor.

On top of this come disturbances from an optical filter attached to a stepping motor, causing vibrations in the housing, which can be seen in the spectra if not corrected for in the control algorithm.

1.1 Tasks

The tasks of this thesis are the following:

- Implement velocity control of the dc motor.
- Reduce the effects of disturbances caused by the stepping motor.

1.2 Outline

The thesis begins with a chapter on modeling. A short introduction to the dc motor is given, and a model suitable for control purposes is presented. The D-A and A-D converters used are also discussed here. Chapter 3 deals with the velocity control of the dc motor. The stepping motor, and the effect it has on the velocity control is discussed in Chapter 4. Finally, Chapter 5 deals with the attenuation of the disturbance caused by the stepping motor.

1.3 Methods

Control design is done using Matlab/Simulink. Everything regarding the physical motors, e.g., control algorithms and experiments, are implemented on a Field-Programmable Gate Array (FPGA), using LabVIEW.

2. Modeling

This chapter discusses modeling aspects of the motor used. An input-output model suitable for control design is presented. For a more extensive reading on dc motors, see [Rashid, 2001] and [Krishnan, 2001].

2.1 Brief Overview of the Motor

A simple dc motor is shown in Figure 2.1. It consists of two main parts, the stationary *stator*, and the rotating *rotor*. The rotor is connected to the shaft that couples the motor with its mechanical load. The stator has slots with a coil of wire, called the *armature windings*. For simplicity, only one single turn coil is shown in the figure. When current is flowing in the windings, interacting with the magnetic field from the permanent-magnet rotor, the torque needed for the rotor to rotate is produced. The physical law behind this is the *Lorentz force law*, which can be written as [Griffiths, 2003]

$$F = \int I(dl \times B), \quad (2.1)$$

where F is the magnetic force on a segment of current-carrying wire, I is the current, dl is a line segment in the direction of the current, and B the magnetic field. Applying this to the motor in Figure 2.1 gives a force to the right at c and a force to the left at cc . However, by invoking Newton's third law, forces equal in magnitude but opposite in direction are exerted on the permanent-magnet, making it rotate counterclockwise. As the rotor turns, the torque is getting smaller and when the rotor is in horizontal position, the torque is zero. To keep a constant torque over a wider range of angles, the stator generally has more slots with windings. In order for the rotor to continue rotating in the same direction, the polarity of the current needs to be reversed when the rotor passes the horizontal position. This is known as *commutating*, and can be done in a few different ways. However, in this thesis a *Limited Angle Torquer (LAT)* is used, i.e., it is intentionally designed to have a limited range with constant torque. So no such commutating exists.

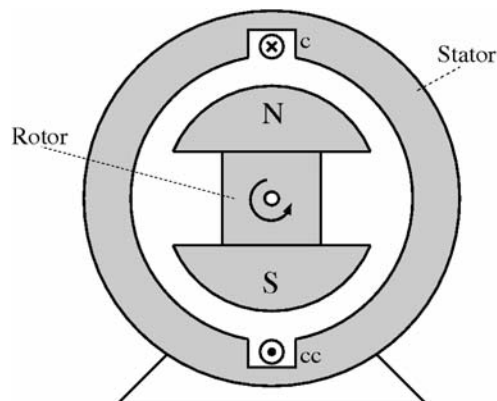


Figure 2.1 Cross section of a simple dc motor.

2.2 Model of the DC Motor

The motor used is a two pole permanent magnet brushless dc (PMBLDC) LAT, brushless in this case just means that there are no mechanical or electrical parts supporting commutating. An equivalent circuit for the motor is shown in Figure 2.2 [Zhang, Smith & Kettleborough, 1999].

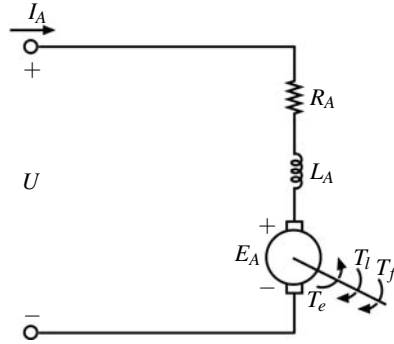


Figure 2.2 Equivalent circuit for the dc motor.

Applying Kirchhoff's voltage law to the electrical loop gives

$$U = R_A I_A + L_A \frac{dI_A}{dt} + E_A, \quad (2.2)$$

where

U is the supply voltage.

R_A is the armature resistance.

I_A is the armature current.

L_A is the armature inductance.

E_A is the induced emf in the armature windings.

A torque balance around the shaft gives

$$J_R \frac{d\omega}{dt} = T_e - T_l - T_f, \quad (2.3)$$

where

J_R is the moment of inertia of the rotor.

ω is the angular velocity of the rotor.

T_e is the electromagnetic torque delivered.

T_l is the load torque.

T_f is the friction torque.

For a two pole motor the relation between angular velocity and angular position is simply

$$\omega = \dot{\theta}. \quad (2.4)$$

The emf can be expressed as

$$E_A = K_e \omega, \quad (2.5)$$

where K_e is the emf constant. A similar relation for the electromagnetic torque is given by

$$T_e = K_t I_a, \quad (2.6)$$

where K_t is the torque constant.

The load and friction torque are subject to modeling on their own. The load consists of a diffraction grating, made approximately symmetrical with respect to the rotational axis, and is modeled as

$$T_l = J_l \frac{d\omega}{dt}. \quad (2.7)$$

Several possible modeling schemes for the friction torque are available in the literature, see for example [Canudas, Åström & Braun, 1986]. Choosing an appropriate model can be difficult a priori. However, the desired angular velocity will be one of two values, $\pm\omega_d$, so a model that is accurate in the vicinity of ω_d should be sufficient. Based on this, the friction is modeled as

$$T_f = \begin{cases} \beta + \alpha\omega & \text{if } \omega > 0 \\ -\beta + \alpha\omega & \text{if } \omega < 0. \end{cases} \quad (2.8)$$

The model of the motor is thus given by equations 2.2-2.8, and can be written in state-space form as

$$\begin{pmatrix} \dot{I}_A \\ \dot{\omega} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -\frac{R_A}{L_A} & -\frac{K_e}{L_A} & 0 \\ \frac{K_t}{J} & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} I_A \\ \omega \\ \theta \end{pmatrix} + \begin{pmatrix} \frac{1}{L_A} & 0 \\ 0 & -\frac{1}{J} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} U \\ T_f \end{pmatrix}, \quad (2.9)$$

where $J = J_R + J_l$. An important limitation of this model is that it is only valid for angles where the electromagnetic torque can be considered independent of the angle. The specific range is supplied by the manufacturer, see Figure 2.3.

2.3 Parameters

The dc motor described in the previous section has a few parameters that need to be determined. Some of these are supplied by the manufacturer. The unknown parameters are

- L_A ,
- J_l ,
- α and β in T_f .

Armature Inductance

L_A can be determined from a step response. By fixing the rotor, making it unable to rotate, equations 2.2 and 2.5 give

$$U = R_A I_A + L_A \frac{dI_A}{dt}, \quad I_A(0) = 0.$$

The solution to this equation when U is a step is given by

$$I_A(t) = \frac{U^0}{R_A} (1 - e^{-\frac{t}{T}}),$$

where U^0 is the magnitude of the step and $T = \frac{L_A}{R_A}$. The inductance can thus be determined by measuring the time constant.

Moment of Inertia of Load

A value for J_l is supplied by FOSS from a detailed CAD model.

Friction

When stationary conditions apply, $\frac{dI_A}{dt} = 0$ and $\frac{d\omega}{dt} = 0$, equations 2.8 and 2.9 give

$$U = R_A I_A + K_e \omega \quad (2.10)$$

$$K_t I_A = \begin{cases} \beta + \alpha \omega & \text{if } \omega > 0 \\ -\beta + \alpha \omega & \text{if } \omega < 0. \end{cases} \quad (2.11)$$

From Equation 2.11 it is seen that α and β can be obtained by plotting the armature current as a function of the angular velocity. This also gives an indication if the proposed friction model is appropriate.

There are two practical problems with this method. First, the experiment needs to be done for angles where the model is valid. This is not possible in open-loop operation, as the rotor angle will be outside the valid range before it reaches stationarity. Hence, it has to be done in closed-loop. Secondly, to obtain a good estimate of β , the velocity needs to be small. This means a small current, however, the input to the motor is the voltage U , and there is a lower limit on how small this can be before it is coerced to zero. This is due to the *Pulse-Width Modulation* (PWM) circuit, more on this in Section 2.5. An extrapolated value from the α estimate is used, see Figure 2.4.

Numerical values for all parameters can be found in Section A.2.

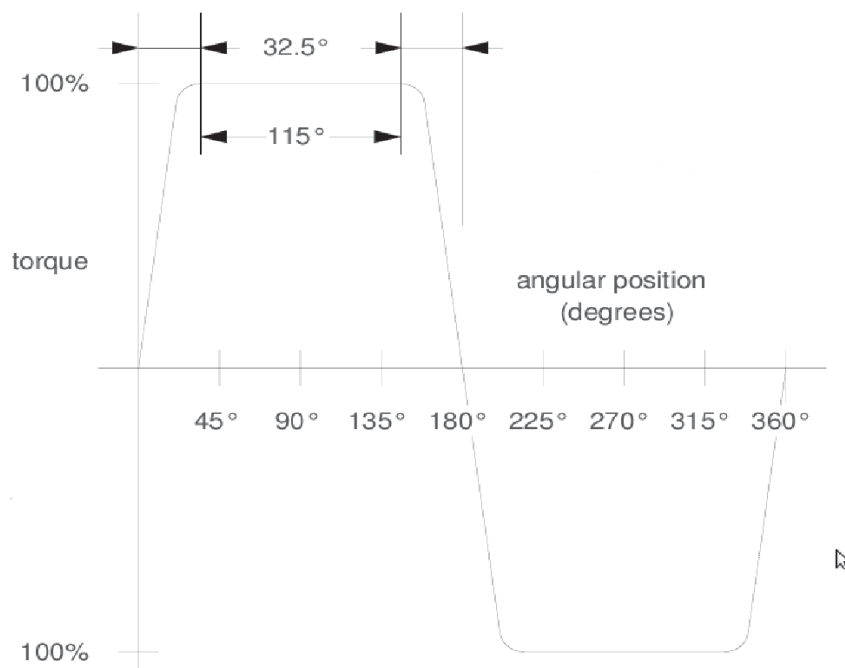


Figure 2.3 Torque curve for the motor.

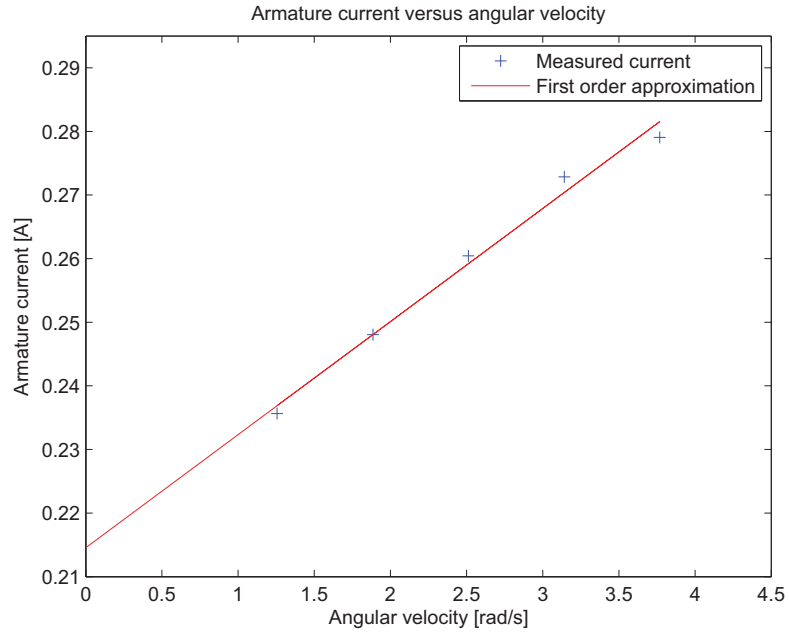


Figure 2.4 Armature current versus angular velocity. The motor showed similar dependence for negative velocities.

2.4 Model Characteristics

The transfer function from supply voltage to angular position is

$$P(s) = \frac{K_t/JL_A}{s(s^2 + (R_A/L_A + \alpha/J)s + (\alpha R_A + K_e K_t)/JL_A)}. \quad (2.12)$$

A bode diagram and a pole-zero map are shown in Figure 2.5 and Figure 2.6, respectively. The motor is well damped with three poles. Besides the pole in the origin, one is located at -6.5 and one at -3700 , on the real axis. It is seen that for frequencies up to about 200 [rad/s], $P(s)$ can be approximated by a second order system.

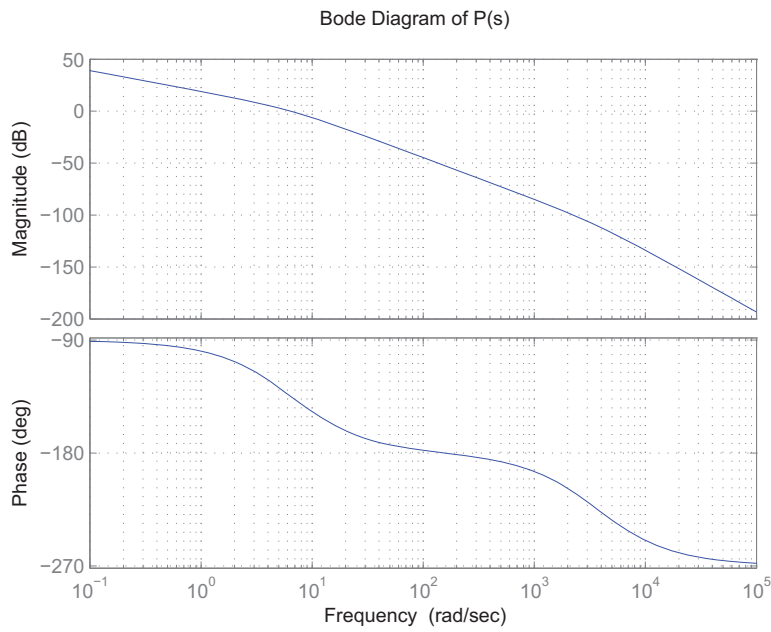


Figure 2.5 Bode diagram of the dc motor.

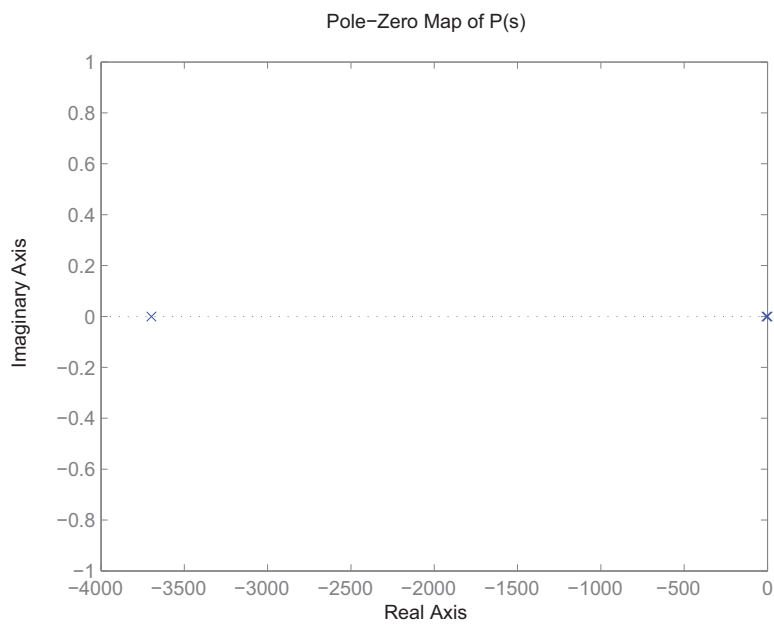


Figure 2.6 Pole-zero map of the dc motor.

2.5 Pulse-Width Modulation

The input to the motor given in Section 2.2 is the supply voltage, and is taken from a constant voltage dc source. Velocity control is obtained by varying the voltage in a suitable way. In order to do that a circuit that converts a constant voltage to a variable voltage is needed. For the motor used in this thesis, this is achieved with an H-bridge and PWM. The PWM output is determined by three parameters [Årzén, 2008], the minimum output U_{min} , the maximum output U_{max} , and the period T_{pwm} . For every period, $0 \leq t < T_{pwm}$, the output is given by

$$U = \begin{cases} U_{max}, & 0 \leq t < DT_{pwm} \\ U_{min}, & DT_{pwm} \leq t < T_{pwm} \end{cases} \quad (2.13)$$

where $0 \leq D \leq 1$ is the *duty cycle*. For the circuit used in this thesis $U_{min} = 0$, which gives an average output signal $\bar{U} = DU_{max}$. The desired voltage is then obtained by varying the duty cycle. An additional bit is used to determine the sign of the output voltage.

One problem with PWM output is that it introduces harmonics in the armature current, causing *torque ripples*. This can be seen from Equation 2.2, when the supply voltage is the output from the PWM, Equation 2.13. Using a shorter PWM period reduces this effect. By following recommendations provided by the hardware manufacturer, the period T_{pwm} is set to 50 [μ s]. The next parameter to consider is the maximum output. The value of U_{max} is given by the constant dc voltage source. The allowed range according to the manufacturer is 8 – 30 [V]. A higher value would mean possibilities for faster control. However, there is a lower bound (and upper) for the duty cycle before it is coerced to zero (or one). This is to avoid too fast switching in the H-bridge, which can cause unpredictable results. A suitable value can be found by considering the desired angular velocity ω_d , which is approximately 2.5 [rad/s]. Then, either from Equation 2.9, or by testing on the real motor, the corresponding input voltage needed can be obtained. The voltage needed was found to be around 0.8 [V]. Furthermore, the PWM is implemented on an FPGA with an internal clock frequency of 40 [MHz], i.e., each clock cycle is 25 [ns]. The PWM period can thus be expressed as 2000 clock cycles. To obtain a desired voltage, a boolean is specified each clock cycle, *true* means output U_{max} , and *false* output 0. The control signal is then a number in the range $[0, 2000]$, representing the amount of clock cycles with output U_{max} . The lower limit for coercion is 80. With a needed voltage of 0.8 V, U_{max} needs to be smaller than $0.8 \frac{2000}{80} = 20$ [V]. However, there is an advantage in choosing a lower value, and that is output resolution. The smallest change in the control signal is $\Delta u_c = \frac{1}{2000}$, resulting in a supply voltage change $\Delta U = \frac{1}{2000} U_{max}$, i.e., the smaller U_{max} , the higher resolution. Based on this, U_{max} is set to 12 [V].

As mentioned above, the control signal when using PWM is a number, whereas the input to the motor is a voltage. This needs to be taken into account when doing control design. One way of doing this is to consider the PWM as a part of the motor, that converts a number to a voltage. This is modeled as a gain, $K_{pwm} = \frac{12}{2000}$.

2.6 Incremental Rotary Encoder

An optical incremental rotary encoder is used to measure position relative to a reference point. It consists of two gray coded tracks for quantized position measurements,

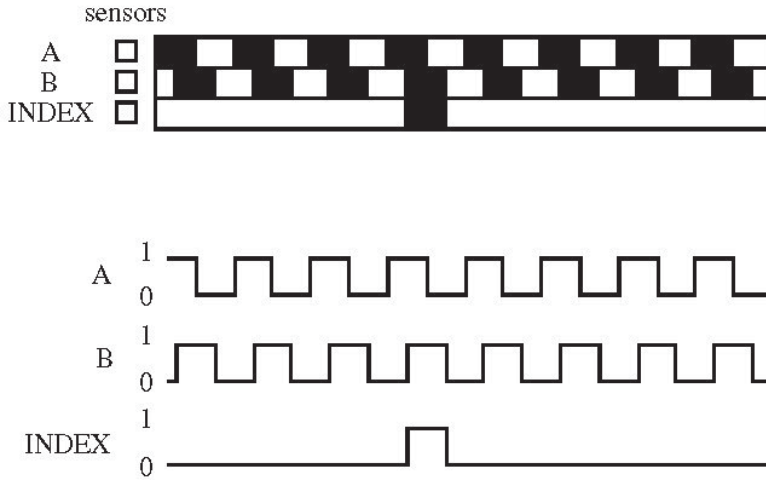


Figure 2.7 A schematic view of the of the encoder and its output.

channels A and B, and one additional track to set the reference point, the index channel. See Figure 2.7. As the shaft rotates, the outputs from channels A and B are pulse trains. The position is obtained by counting the number of edges that have passed, and by knowing how many there are on a full revolution. The phase of the pulse trains determine the direction of motion. At any given time the output from channels A and B can be in one of four states, which can be represented by a binary number, $\{00, 01, 10, 11\}$. To see how this works, lets define the direction of positive motion to the left in Figure 2.7, and start in state 00. An output change to 01 represents positive motion, and 10 negative motion. It is important to read the encoder output often enough, so that no pulses are missed. The encoder used has 12500 pulses per revolution on each channel, i.e., 50000 edges per revolution when using both channels. The implementation is done by polling the channel outputs each clock cycle, making it possible to track velocities up to roughly 5000 [rad/s], well above what is necessary. The reason for doing this is to get highest possible time resolution.

Velocity Estimation

To do velocity control of the motor, it is necessary to estimate the velocity from the quantized position measurements given by the encoder. The estimate used is given by a first order filter,

$$\hat{\Omega}(s) = \frac{s}{1 + sT_f} Y(s) = H_f(s)Y(s), \quad (2.14)$$

where $\hat{\Omega}$ is the estimated velocity, and Y the position output from the encoder. By choosing T_f appropriately, the filter will approximate a derivative well for low frequencies, and limit the effects of high frequency measurement noise. Filters of higher order were tested, but showed no significant improvement. How T_f is chosen is discussed in Section 3.4.

To implement 2.14 on the FPGA, a discretization is needed. Here, the numerator and denominator of $H_f(s)$ will be treated differently. The denominator is discretised using Tustin approximation with sampling period equal to that of the controller. The pure derivative is approximated as

$$\dot{y} = \frac{\Delta y}{\Delta t}. \quad (2.15)$$

If Δt is fixed, Equation 2.15 is a standard finite difference approximation. However, it is implemented with Δy fixed, where Δy is one state change in the encoder loop. Δt is then the time elapsed between a state change. This takes advantage of the fact that the time resolution is a lot higher than the encoder resolution.

The model given by Equation 2.9 has angular velocity in [rad/s]. However, the velocity estimate is in [pulses/s], and so is the specified reference velocity given by Foss. To account for this, the encoder is modeled as a gain, $K_{enc} = \frac{50000}{2\pi}$.

2.7 Summary

In this chapter modeling of the dc motor has been discussed. Experiments to determine unknown parameters were covered in Section 2.3, where the friction parameters are subject to some uncertainty. However, the used friction model simplifies the control design. The D-A and A-D converters are discussed in Section 2.5 and Section 2.6, respectively. These are important to take into account when doing control design, as they change the input-output behavior.

3. Control Design

3.1 Control Objective

The main control objective is to keep the angular velocity constant when the monochromator is performing its scan. The scan is synchronized on position readings from the encoder, and is done in the interval $[-3300, 3900]$ [pulses]. One full scan consists of a forward scan when the motor has positive velocity, and one backward scan when the motor has negative velocity.

- The velocity during forward and backward scan should be 20000 and -20000 [pulses/s], respectively.
- One full scan should take less than one second.
- The estimated velocity should be within 0.5% of the desired velocity, i.e., it should be within $\pm 20000 \pm 100$ [pulses/s].

Note that the third control objective is highly dependent on how the velocity estimate is done. A long T_f in Equation 2.14 would mean a lower cutoff frequency, and hence more attenuation of noise. The objective is, however, a good guideline of what can be considered acceptable.

3.2 Reference Trajectory

Considering the objectives, the reference velocity is ± 20000 [pulses/s] in a predefined interval, depending on the position. There are many possible ways to accomplish this, one is to let the reference be a square wave synchronized on encoder position. The choice on when, i.e., for what position, to switch from positive to negative reference velocity, and vice versa, depends on the rise time of the closed loop system. This is decided through experiments.

3.3 Controller Choice

The controller should make the motor follow the reference trajectory without any steady state error, and speed up the transient response in order to satisfy the objectives. This motivates the use of a Proportional-Integral (PI)-controller. The integral term will also account for unmodeled dynamics. The PI-controller with set point weighting can be described by the equation [Åström & Wittenmark, 1997]

$$u_c(t) = K_c(b\omega_d(t) - \hat{\omega}(t) + \frac{1}{T_i} \int^t (\omega_d(\tau) - \hat{\omega}(\tau)) d\tau). \quad (3.1)$$

The use of set point weighting, b , provides a way to separate reference signal response from the response to disturbances. With large changes in reference signal, b can be used to reduce a plausible overshoot.

3.4 Design

There are several possible ways to design a PI-controller. Although they differ in methodology, the end result is often similar. The approach used in this thesis is loop shaping. The design should result in a stable system that fulfills the objectives, and ensure robustness to model uncertainties, time delays, and disturbances. For a system described as in Figure 3.1, the loop transfer function is given by $L(s) = P(s)C(s)$.

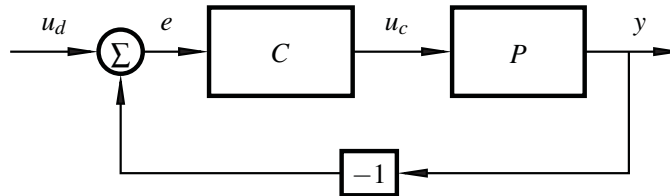


Figure 3.1 Simple feedback structure.

The well-known *Nyquist criterion* provides a simple way to determine stability when the loop transfer function is available. However, only stability does not say much about how the system will behave, it is important to have stability with good margins. Two common stability margins are the *amplitude margin* and the *phase margin*, [Hägglund, 2009].

Phase Margin

The phase margin is a measurement of how much the phase can decrease before the system becomes unstable. It is defined as

$$\varphi_m = \pi + \arg L(i\omega_c) \quad (3.2)$$

where ω_c is the crossover frequency, that is, the frequency where

$$|L(i\omega_c)| = 1. \quad (3.3)$$

The crossover frequency is closely related to the bandwidth, and thus to the speed of the closed loop system.

Amplitude Margin

Similar to the phase margin, the amplitude margin is a measurement of how much the gain can be increased until the system becomes unstable, and is defined as

$$g_m = \frac{1}{|L(i\omega_o)|}. \quad (3.4)$$

Here, ω_o is the frequency where the loop transfer function has phase $-\pi$,

$$\arg L(i\omega_o) = -\pi. \quad (3.5)$$

When designing the controller there are a few rules of thumb considering the margins. Common values are $g_m \in [2, 6]$ and $\varphi_m \in [45^\circ, 60^\circ]$.

Loop Shaping

The process transfer function was given in Section 2.4. From Equation 3.1,

$$U_c(s) = K_c(b\Omega_d(s) - \hat{\Omega}(s) + \frac{1}{sT_i}(\Omega_d(s) - \hat{\Omega}(s))) = \underbrace{K(1 + \frac{1}{sT_i})}_{C_{FB}} E(s) + \underbrace{K(b-1)}_{C_{FF}} \Omega_d(s) \quad (3.6)$$

where $E(s) = \Omega_d(s) - \hat{\Omega}(s)$. Together with Equation 2.14, this can be represented as in Figure 3.2. The similarity with Figure 3.1 is clear, and by defining

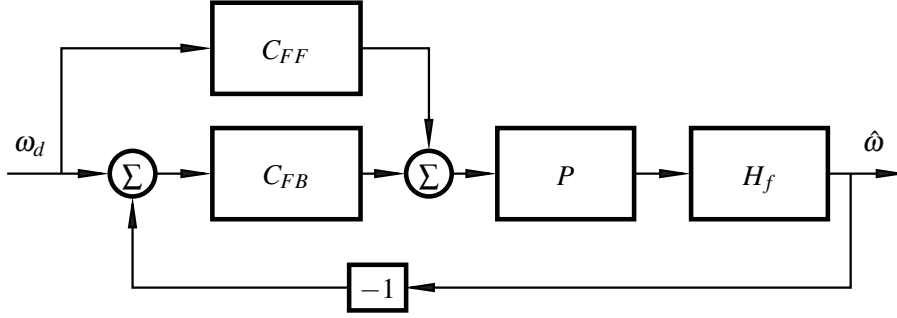


Figure 3.2 Block diagram of the system. Note that the PWM and encoder are not included.

$L(s) = H_f(s)P(s)C_{FB}(s)$, the margins discussed above apply. The use of set point weighting introduces an additional degree-of-freedom, and can be seen as a feedforward term from the reference signal.

When combining the PI-controller and the velocity filter, the loop transfer function becomes

$$L(s) = \frac{K_c}{T_i} \frac{1 + sT_i}{1 + sT_f} P(s). \quad (3.7)$$

This is a lead filter (if $T_i > T_f$) in series with the process. The lead filter lifts the phase of the loop, and when considering the phase margin, it is desired to have maximum phase lift at the crossover frequency. This is achieved by placing the zero and pole on either side of ω_c , at equal logarithmic distance. The further apart, the larger phase lift (max 90°). However, short T_i and long T_f are desired for rejection of process disturbances and measurement noise. As a compromise, they are chosen as $T_i = d\omega_c^{-1}$ and $T_f = d^{-1}\omega_c^{-1}$, with $d \approx 5$. This gives a phase lift of approximately 65° . The crossover frequency, ω_c , is then left to be decided through experiments, from which K_c is determined,

$$K_c = \frac{1}{\omega_c |P(i\omega_c)|}. \quad (3.8)$$

The PWM and encoder gains have so far been disregarded, but are easily accounted for by a scaling of K_c ,

$$K_{new} = \frac{K_c}{K_{pwm}K_{enc}}. \quad (3.9)$$

Bode diagram of the lead filter and loop transfer function are shown in Figure 3.3 and Figure 3.4, respectively.

The set point weighting parameter $b = 0.15$ is used to eliminate the overshoot.

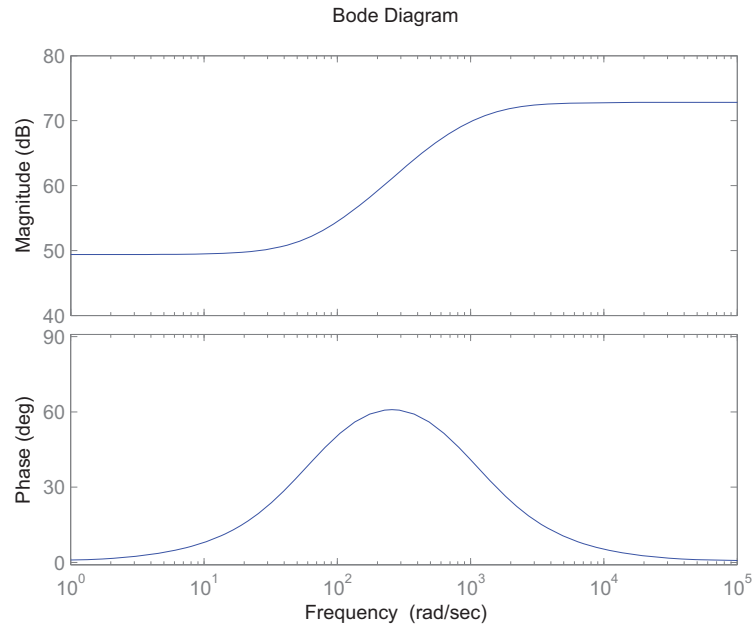


Figure 3.3 Bode diagram of the lead filter. The phase peak is placed at the crossover frequency.

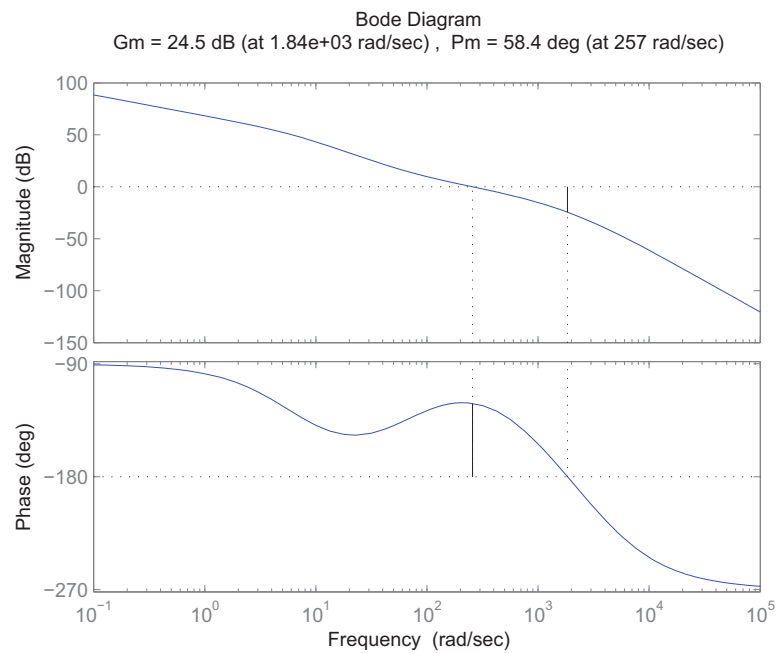


Figure 3.4 Bode diagram of $L(s)$. It is seen that the gain and phase margins are roughly 24 dB and 58° , respectively.

Discretization

To implement the controller on the FPGA, a discretization of the control algorithm is necessary. For a PI controller, only the integral term needs to be approximated. From

Equation 3.1, the integral term can be rewritten as

$$\frac{dI(t)}{dt} = \frac{K_c}{T_i} e(t).$$

Using forward approximation, $\dot{x} = \frac{x(t+T_s) - x(t)}{T_s}$, gives

$$I(t + T_s) = I(t) + \frac{K_c T_s}{T_i} e(t), \quad (3.10)$$

where T_s is the sampling period. A rule of thumb for selecting sampling period can be found in [Åström & Wittenmark, 1997], and read

$$T_s \omega_c \approx 0.05 - 0.14. \quad (3.11)$$

Based on this, the sampling period is set to $T_s = 0.0005$.

Anti-Windup

As the controller includes integral action, there is a risk of windup. Different methods exist to overcome the problem, see for example [Visioli, 2006]. In this thesis, an approach often called *tracking* is used. An extra feedback path is added as illustrated in Figure 3.5. The error signal e_s between the actuator model output and the control output is fed through an integrator with gain $\frac{1}{T_i}$. As long as the actuator is not saturated, the error signal is zero. When the actuator is saturated, the added feedback path tries to reduce e_s to zero. This effectively resets the integrator. The *tracking-time* constant T_i determines the rate at which the integrator is reset. A value of $T_i = 0.5T_s$ was found appropriate in simulations.

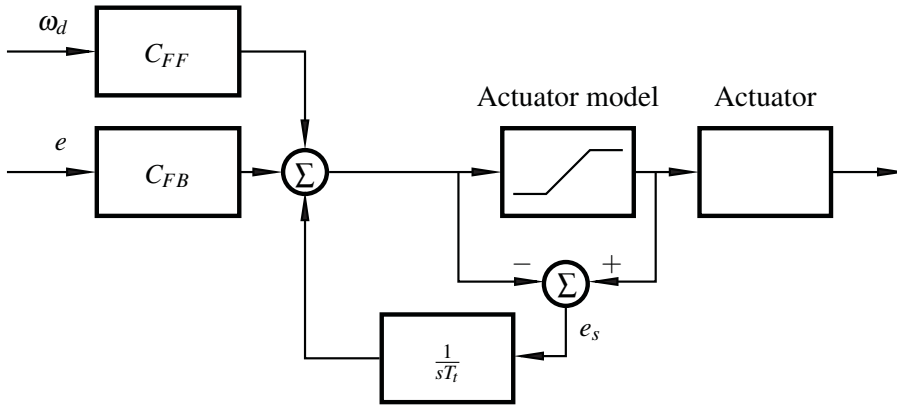


Figure 3.5 Block diagram of controller and actuator, with an added feedback path to prevent windup.

Fixed-Point Arithmetic

The FPGA does not have any hardware support for floating point arithmetic. To overcome this, all control parameters are stored using integers. Each parameter is scaled by a fixed power of 2 and rounded to nearest integer.

As it turns out, all parameters are smaller than one in value, so no bits are needed to store the integer part. The fractional part is stored with 16 bits precision, giving a resolution of 2^{-16} , which was considered sufficient. For more information on fixed-point arithmetic, see [Årzén, 2008].

3.5 Experimental Results

In the following experiments the reference trajectory changes sign at 5200 and -4600 [pulses].

The PI-controller is evaluated in accordance with the objectives in Section 3.1. Figure 3.6 and Figure 3.7 show the estimated angular velocity and control signal, respectively, when the reference velocity is a square wave. As can be seen the scan time objective is fulfilled. The control signal is within its range, $[-2000, 2000]$, and is showing low control effort. The latter is desired since a noisy control signal would cause more wear and tear.

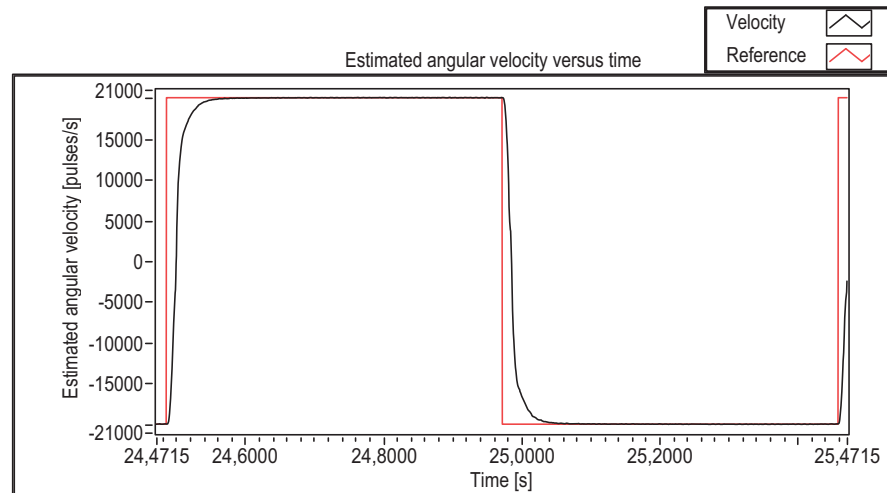


Figure 3.6 Estimated angular velocity versus time with PI-controller. The scan time objective of less than one second is achieved.

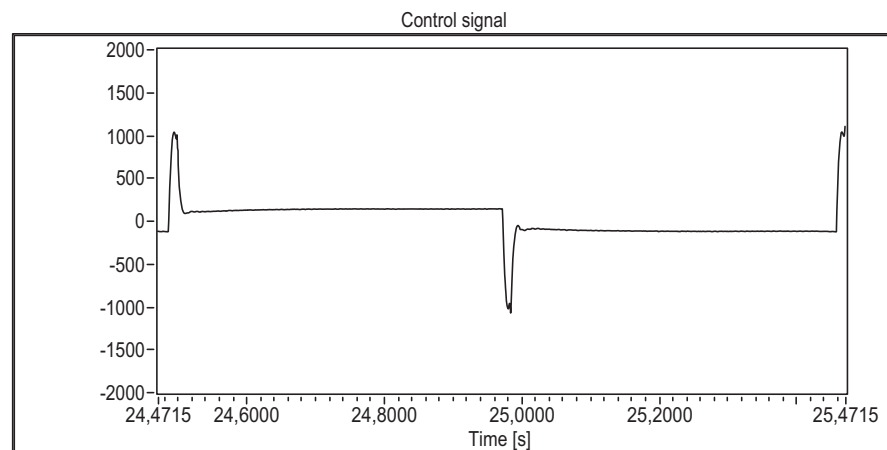


Figure 3.7 Control signal for the square wave experiment in Figure 3.6. A value of 2000 corresponds to the maximum of 12 [V].

In order to evaluate the second and third objective, it is appropriate to look at the estimated velocity versus position. This is shown in the figures on the following page. The control objectives are fulfilled, albeit not with much margin. The velocity has not fully reached steady state at -3300 and 3900 [pulses], although it is within $\pm 20000 \pm 100$ [pulses/s].

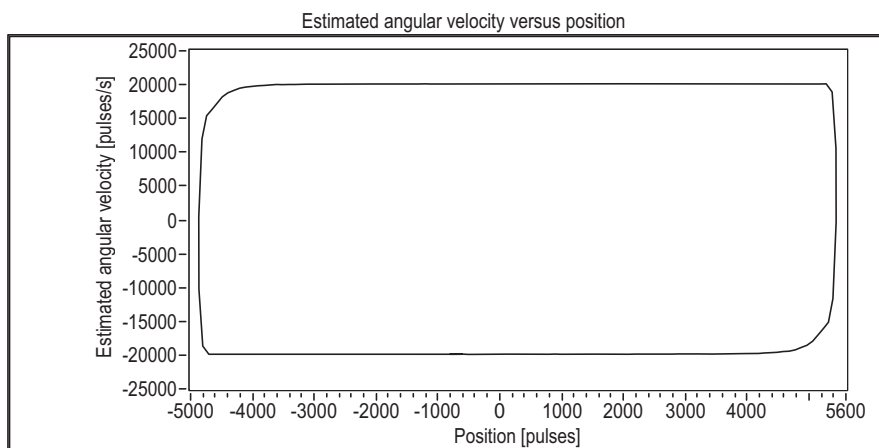


Figure 3.8 Estimated angular velocity versus position with PI-controller. The velocity should be ± 20000 [pulses/s] in the interval $[-3300, 3900]$ [pulses].

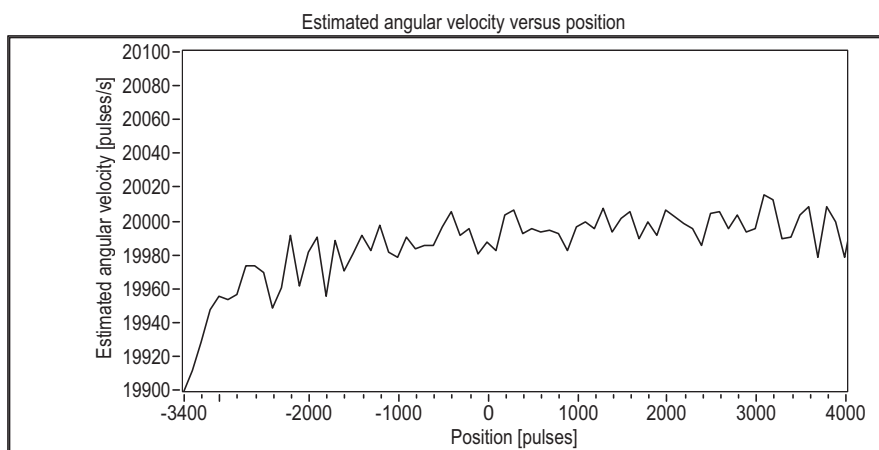


Figure 3.9 Figure 3.8 zoomed in when the velocity is positive.

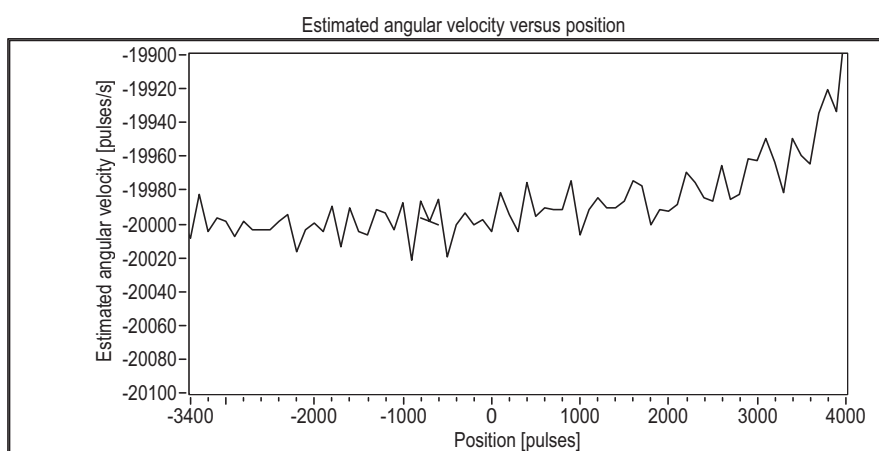


Figure 3.10 Figure 3.8 zoomed in when the velocity is negative.

3.6 Summary

This chapter has been devoted to the velocity control of the dc motor. Stability with good margins has been shown in Section 3.4. Implementation details, such as discretization and fixed-point arithmetic, have also been covered.

The proposed controller performs well, fulfilling the objectives, although not with much margin.

4. Stepping Motor Disturbance

4.1 Optical Filter and Stepping Motor

The diffraction grating splits the light into different wavelengths before it goes through the exit slit. These will, however, overlap, and to reduce this effect an optical filter is placed in front of the exit slit. The optical filter actually consists of many filters placed in series. This is needed since different wavelengths demand a different filter. Depending on the position of the grating, a specific wavelength will be passed through the filter and exit slit. In order to have the correct filter in front of the exit slit, the filter is mounted on a stepping motor that rotates depending on the grating position. Both the dc motor with grating, and stepping motor with filter are attached to the same housing. Each time the stepping motor takes a step, it causes vibrations in the housing. The vibrations can be seen as a disturbance acting on the dc motor.

4.2 Disturbance Properties

The disturbance is considered unknown, since the grating position is the only thing measured. However, the time, or rather, the grating position when the disturbance will enter, is known. Each step taken by the stepping motor is determined by a predefined array, where each element corresponds to a grating position. The direction (clockwise or counterclockwise), is dependent on the reference velocity. It should be noted that a different array is used for each direction. However, the total number of elements are the same, and consists of 34 nonuniform steps.

To get an idea of how the disturbance affect the velocity control of the dc motor, the same experiment as in Section 3.5 is performed, both with and without the stepping motor. Recall that the scan is made in the position range $[-3300, 3900]$ [pulses], the stepping motor is working in the range $[-3100, 3700]$ [pulses]. Figures 4.1 and 4.2 show the velocity during the forward- and backward scan, respectively. These should be compared to figures 4.3 and 4.4, when the stepping motor has been disabled.

The effect of the stepping motor is most evident in the beginning when it takes its first steps, where the velocity has a peak deviation of almost 200 [pulses/s] from the reference. One might reason that it is because the velocity has not fully reached steady state. However, a simple experiment where the period of the square wave reference was increased, discarded this idea. By comparing figures 4.1 and 4.2, it is seen that the behavior is slightly different. There are many possible reasons for this. As mentioned, the array containing the stepping information is different during positive and negative reference, but also the housing where everything is attached is not symmetric with respect to the dc motor. The latter might cause the vibrations to propagate differently, depending on the stepping motor direction of motion.

The way the housing is positioned plays an important part of how the disturbance will affect the velocity control. Figures 4.5 and 4.6 show the result when the housing is attached to a metallic casing. This indicates that a mechanical solution could be possible. However, since different instruments have different mechanical parts, finding a mechanical solution that works for all might be difficult. There is of course also the economic aspect to consider, where a software solution is most likely more lucrative, especially if it is instrument independent.

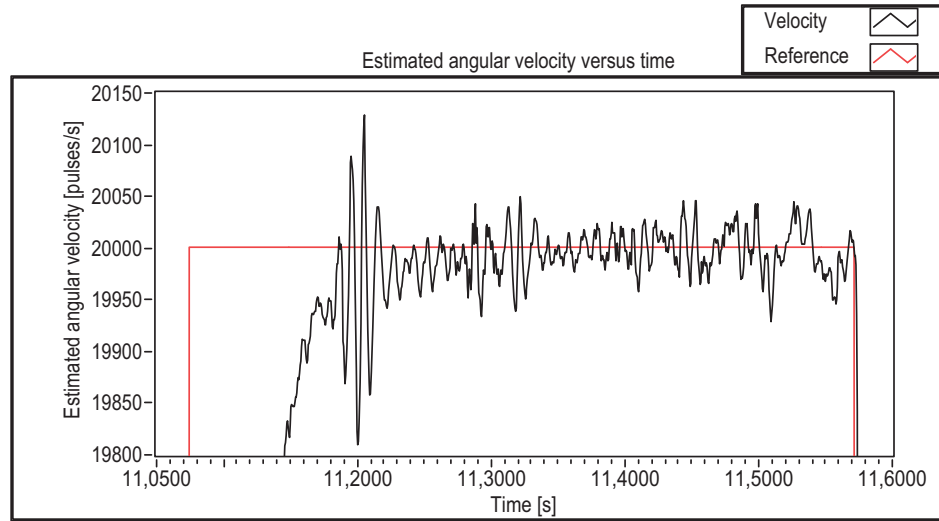


Figure 4.1 Estimated angular velocity versus time with PI-controller and stepping motor enabled, during forward scan.

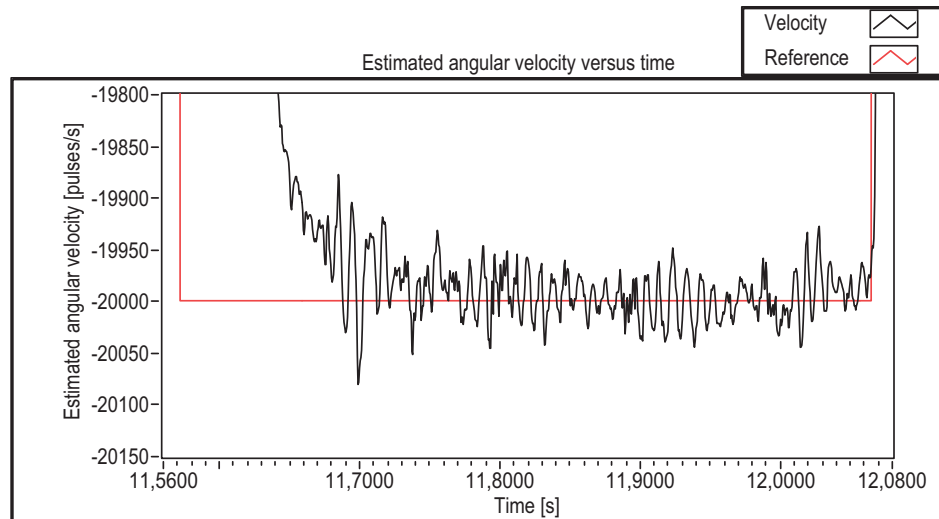


Figure 4.2 Estimated angular velocity versus time with PI-controller and stepping motor enabled, during backward scan.

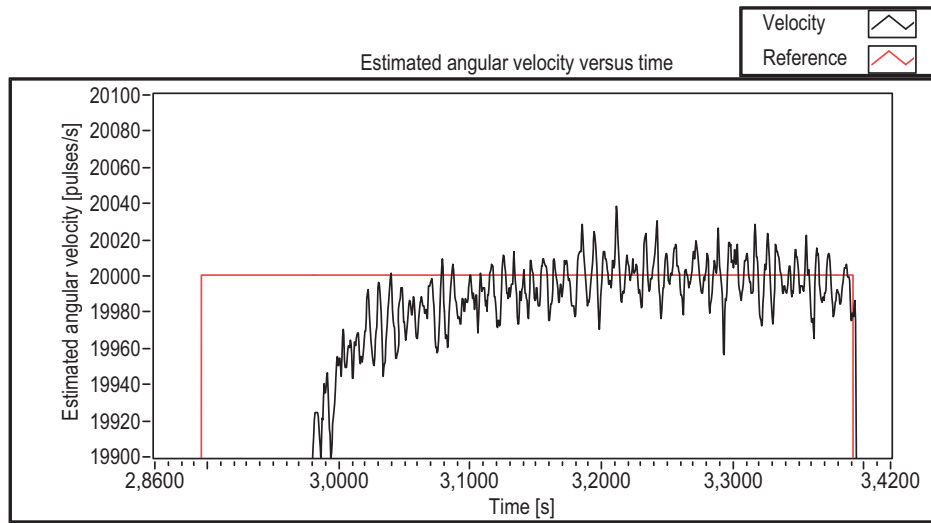


Figure 4.3 Estimated angular velocity versus time with PI-controller and stepping motor disabled, during forward scan.

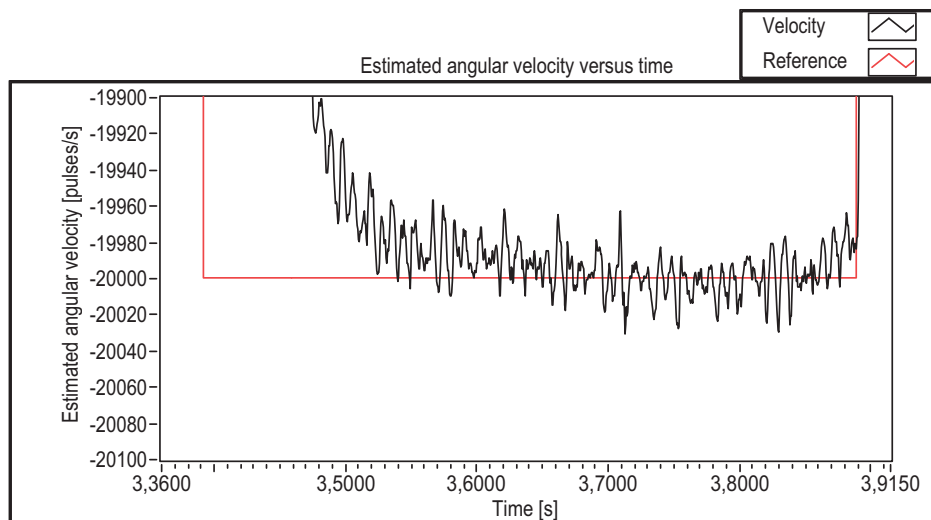


Figure 4.4 Estimated angular velocity versus time with PI-controller and stepping motor disabled, during backward scan.

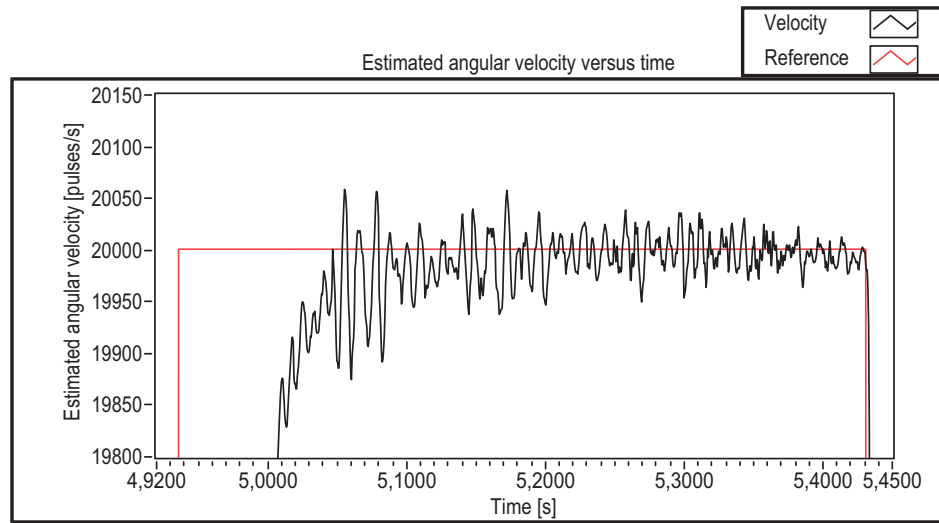


Figure 4.5 Estimated angular velocity versus time with PI-controller and stepping motor enabled, during backward scan. Housing attached to a metallic casing.

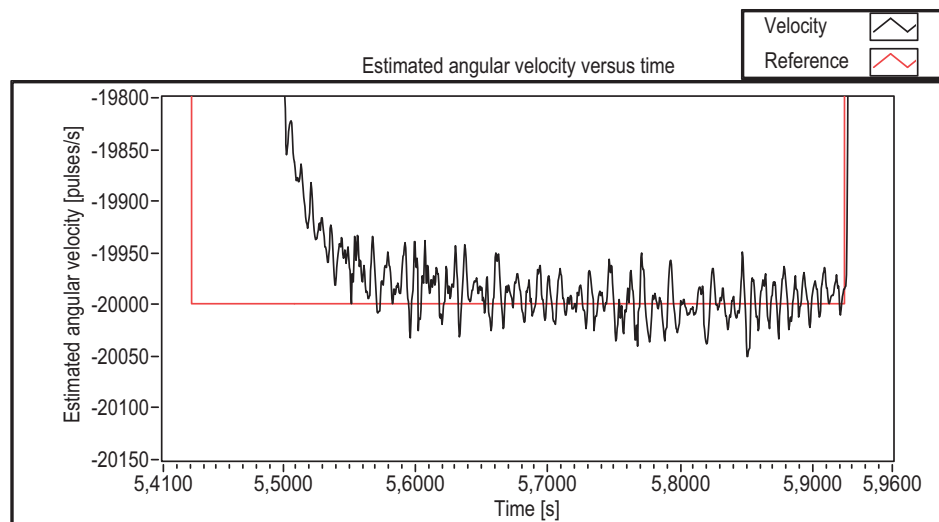


Figure 4.6 Estimated angular velocity versus time with PI-controller and stepping motor enabled, during backward scan. Housing attached to a metallic casing.

4.3 Disturbance Rejection

One approach to reduce disturbance effects is some kind of feedforward control scheme, where the goal is to find an input to the system, such that the output given by that input negates the effect caused by the disturbance. However, since the disturbance in this case is highly dependent on housing position, which differ from instrument to instrument, such a scheme would have to be able to find the feedforward control signal automatically in order to be successful. This could be achieved by augmenting the system description with a model of the disturbance, where the parameters of the model are identified online. This is often referred to as *adaptive control*. In order for this to be successful the structure of the model needs to be correct. Consider for example Figure 4.1, the disturbance seems to have at least one dominating frequency. Depending on the complexity of the model needed, one might run into problems with excitation in the identification of parameters [Åström & Wittenmark, 1995].

The disturbance exhibits repetitive behavior, in the sense that during each forward- and backward scan, the disturbance is almost identical compared to the previous forward- and backward scan. This is expected, as the stepping motor takes its steps at the exact same grating positions each scan. This important property can be used for disturbance rejection, and is discussed in the following chapter.

5. Iterative Learning Control

Iterative Learning Control (ILC) methods deal with systems that perform the same task repeatedly. Given a reference trajectory and a system, the idea is to find, with information from a previous run, an input to the system such that the output follows the reference trajectory as well as possible. For an in-depth reading on ILC, the reader may benefit from [Norrlöf, 2000] and [Johansson, 2008].

ILC Formulation

Let $\omega_d(t)$ be a reference trajectory defined on a finite interval $[0, t_f]$, and that the system should follow this trajectory repeatedly with high accuracy. At every iteration an external ILC control signal will be added to the input. The goal is to, for every iteration, reduce the tracking error, ideally converging to zero. Figure 5.1 shows the system considered. Here $u(t)$ and $d(t)$ are the ILC control signal and disturbance, respectively. Define the tracking error $e(t) = \omega_d - \hat{\omega}$. The following equations describe the problem:

$$\hat{\omega}_k(t) = T_c(q)\omega_d(t) + T_u(q)u_k(t) + T_d(q)d_k(t) \quad (5.1)$$

$$e_k(t) = \omega_d(t) - \hat{\omega}_k(t) \quad (5.2)$$

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t)) \quad (5.3)$$

where q is the time shift operator. The subscript k is called the *iteration index*, and indicates how many times the iterative motion has been repeated. Note that $T_d(q)$ and $d(t)$ are unknown. In this thesis the ILC control signal is applied at the same input as the reference trajectory, i.e., $T_u(q) = T_c(q)$, where $T_c(q)$ is a discrete-time equivalent of $T_c(s)$. From Figure 3.2 in Section 3.4, the closed loop transfer function can be found,

$$T_c(s) = \frac{(C_{FF}(s) + C_{FB}(s))P(s)H_f(s)}{1 + C_{FB}P(s)H_f(s)}. \quad (5.4)$$

The external ILC control signal is given by Equation 5.3, and the filters $Q(q)$ and

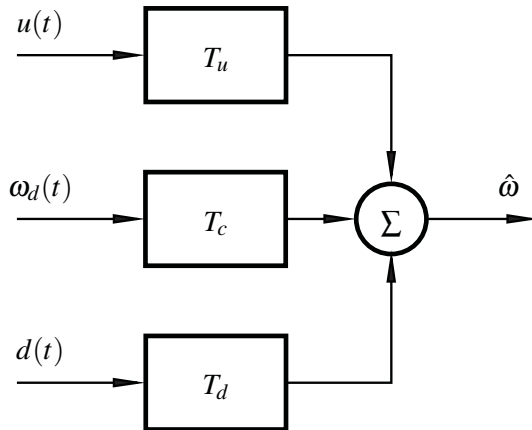


Figure 5.1 System representation considered for ILC.

$L(q)$ are to be designed. An important observation is that $Q(q)$ and $L(q)$ do not need to be proper, since in the calculation of u_{k+1} , $Q(q)$ and $L(q)$ operate on the whole time sequences u_k and e_k , already known from the previous iteration.

Error Equation

A successful ILC algorithm must be stable, i.e., the tracking error should be reduced as a function of iteration. From equations 5.1-5.3, and the fact that $T_u(q) = T_c(q)$, a recursive expression for the tracking error is derived:

$$\begin{aligned}
 e_{k+1}(t) &= \omega_d(t) - \hat{\omega}_{k+1}(t) = \omega_d(t) - (T_c(q)\omega_d(t) + T_c(q)u_{k+1}(t) + T_d(q)d_{k+1}(t)) \\
 &= \omega_d(t) - T_c(q)\omega_d(t) - T_c(q)(Q(q)(u_k(t) + L(q)e_k(t)) - T_d(q)d_{k+1}(t)) \\
 &= Q(q)(1 - T_c(q)L(q))e_k(t) + (1 - Q(q))(1 - T_c(q))\omega_d(t) \\
 &\quad + T_d(q)(Q(q)d_k(t) - d_{k+1}(t))
 \end{aligned} \tag{5.5}$$

Consider first the disturbance free case, i.e., $d_k(t) = 0, \forall k$. The following stability criterion can be found in [Norrlöf, 2000].

THEOREM 5.1

If the system

$$\hat{\omega}_k = T_c(q)\omega_d(t) + T_c(q)u_k(t)$$

is controlled using the ILC updating equation

$$u_{k+1}(t) = Q(q)(u_k(t) + L(q)e_k(t))$$

and

$$|1 - T_c(e^{i\omega t_s})L(e^{i\omega t_s})| < |Q^{-1}(e^{i\omega t_s})|, \tag{5.6}$$

where $\omega \in [-\pi, \pi]$ and t_s is the sampling time, then the ILC system is stable. \square

Note that the stability criterion is sufficient, it is still possible to achieve stability even though Equation 5.6 is not fulfilled. However, in [Norrlöf, 2000] it is shown that when Equation 5.6 is fulfilled, the 2-norm of the error decreases as function of iteration. This would be a desired property in a practical application. Running the system with ILC, even if only for a few iterations, should make the tracking better, than without ILC. Ideally the error should converge to zero value, from Equation 5.5 it is seen that this is only possible if $Q(q) = 1$. Considering Theorem 5.1, this can be seen as a restriction on the choice of $L(q)$, it is clear that another choice of $Q(q)$ could increase the stability region. In order to have robustness to modeling errors, $Q(q)$ is often chosen as a low pass filter. Although this means the error will not converge to zero, an appropriate choice of $Q(q)$ will make the asymptotic error close to zero.

Now consider the case when a disturbance is acting on the system. If $d_{k+1}(t) = d_k(t)$, and $Q(q) = 1$, the disturbance would be completely attenuated by the ILC. Since $Q(q)$ will be chosen differently, this will not be the case. However, it is still possible to reduce the effect of the disturbance. In a typical case $T_d(q)$ and $Q(q)$ are of low pass character, i.e., $1 - Q(q)$ is of high pass character. Assuming repetitive disturbance, $d_{k+1}(t) = d_k(t)$, the multiplication of $T_d(q)$ with $1 - Q(q)$ results in a band pass filter. Hence, the disturbance attenuation will depend on the frequency content of $d(t)$ and the cutoff frequencies of $T_d(q)$ and $Q(q)$.

5.1 Design

A few different design methods are proposed in [Norrlöf, 2000]. They all have in common that they use some knowledge about the system being controlled. In this thesis, the focus is on one particular design method.

Model-Based Approach

The following steps constitute the model-based design procedure.

1. Build a model of the relation between the ILC input and the resulting correction on the output, i.e., find a model $\hat{T}_c(q)$ of $T_c(q)$.
2. Choose a filter $H_B(q)$ such that it represents the desired convergence rate for each frequency. Normally this means a high pass filter.
3. Calculate $L(q)$ by $L(q) = \hat{T}_c^{-1}(q)(1 - H_B(q))$.
4. Choose the filter $Q(q)$ as a low pass filter with cutoff frequency such that the bandwidth of the resulting ILC is high enough and the desired robustness is achieved.

Step 1 was done in Chapter 2 and Chapter 3, and can be summarized by Equation 5.4, with the exception that a discretization is needed. To explain step 2 and step 3, consider Equation 5.5 with $Q(q) = 1$ and $d_{k+1}(t) = d_k(t)$. The error equation then becomes

$$e_{k+1}(t) = (1 - T_c(q)L(q))e_k(t).$$

The filter $H_B(q)$ clearly decides the convergence rate for the error, and in this thesis $H_B(q) = 0$, which can be considered an aggressive choice. The filter $Q(q)$ is then left to be chosen through experiments.

Discretization

In order to use the design procedure above, and also to implement the ILC on the FPGA, a discretization of Equation 5.4 is needed. Most parts of the discretization have been covered already, $C_{FB}(s)$ and $C_{FF}(s)$ in Section 3.4, and $H_f(s)$ in Section 2.6.

The discretization of $P(s)$ is done using *zero-order hold*. The use of zero-order hold is based on how the D-A converter, i.e., the PWM is constructed and implemented. An important property of zero-order hold sampling is that a minimum-phase continuous-time system may become a nonminimum-phase discrete-time system. In fact, if the pole-excess of the continuous-time system is larger than 2, and if the sampling period is sufficiently short, the resulting discrete-time system will always be nonminimum-phase [Åström & Wittenmark, 1997]. From Section 2.4 it is seen that the pole excess is 3. Now, the zeros of $P(s)$ will become zeros of $T_c(s)$, and the same applies to the discrete-time equivalent. In step 3 of the design procedure, $T_c(q)$ is inverted, resulting in an unstable inverse. There are a few ways to get around this problem, which are discussed in the following section.

5.2 Nonminimum-Phase Systems

Perhaps the most common way to get around the inversion of nonminimum-phase systems is some type of model approximation.

From the discussion in the previous section, by approximating $P(s)$ with a system of lower order, the introduction of nonminimum-phase zeros by the zero-order hold discretization can be avoided. In Section 2.4 it was mentioned that $P(s)$ could be approximated with a second order system. This would, however, give rise to another problem in practice, as a pole excess of 2 will result in a zero that moves to -1 as the sampling period decreases [Åström & Wittenmark, 1997]. As the implementation

environment has limited numerical precision, one would need to carefully consider how the filter is implemented.

Now consider the case when $T_c(q)$, with z -transform $T_c(z)$, is nonminimum-phase. There are several methods in the literature that focus on approximating the transfer function with one that is minimum-phase. Most of them involve mirroring the zeros with respect to the unit circle, see for example [Butterworth, Pao & Abramovitch, 2009]. However, the filtering done by $L(q)$ does not need to be causal, and an approximation is not needed. The technique used here is based on the minimum-phase and all-pass decomposition found in [Mitra, 2006]. Factorize $T_c(z)$ as

$$T_c(z) = T_c^+(z)A(z), \quad (5.7)$$

where $T_c^+(z)$ is minimum-phase, and $A(z)$ is an all-pass transfer function. The zeros of $A(z)$ are the nonminimum-phase zeros of $T_c(z)$, and the poles of $A(z)$ are placed so that they exhibit mirror-image symmetry with the zeros, with respect to the unit circle. To clarify, if $z = re^{i\omega}$ is a zero of an all-pass transfer function, then it has a pole at $z = \frac{1}{r}e^{-i\omega}$. The poles of $A(z)$ are then zeros of $T_c^+(z)$ in order to satisfy Equation 5.7. It is assumed that no zeros of $A(z)$ lie exactly on the unit circle. The filtering operation, $u(t) = T_c(q)^{-1}y(t) = T_c^+(q)^{-1}A(q)^{-1}y(t)$ can now be done as follows: Filter $y(t)$ through $T_c^+(q)^{-1}$,

$$v(t) = T_c^+(q)^{-1}y(t).$$

Now, since $A(z)^{-1}$ will have poles outside the unit circle, it has an unstable causal impulse response. However, by choosing the *region of convergence* (ROH) as the interior of a circle passing through the pole nearest to the origin, $A(z)^{-1}$ has a stable noncausal impulse response. This is true since the ROH will always include the unit circle, as the pole nearest to the origin will be strictly outside the unit circle. Hence, the filter operation

$$u(t) = A(q)^{-1}v(t)$$

can be done in reversed time. This can be implemented by filtering the reversed sequence $w(t) = v(N-t)$, where N is the number of data elements, through the stable causal filter $A(q^{-1})^{-1}$, and then reversing the filtered sequence again,

$$g(t) = A(q^{-1})^{-1}w(t)$$

$$u(t) = g(N-t).$$

5.3 Implementation Notes

In the beginning of the chapter it was mentioned that the ILC works iteratively on data defined on a finite interval $[0, t_f]$. Since reference trajectory, and the stepping motor are synchronized on grating position, it is natural to also synchronize the ILC on grating position.

Due to limited size of the FPGA, the ILC could not be applied to a full scan, and the implemented ILC is applied only to the beginning of the forward scan. However, as was seen in Figure 4.1, this is where the disturbance is most evident. The implementation is done as follows: Each time the grating passes position -3400 [pulses], and the reference velocity is positive, the ILC starts logging the tracking error, a total of 192 samples are collected. The same sampling period as for the PI-controller is used, i.e. 0.5 [ms]. When the data has been collected, filtering according to Equation 5.3 is performed. The ILC control signal is applied in the same way, i.e., simultaneously with the logging. It is initially set to zero.

5.4 Experimental Results

The ILC is now evaluated with focus on its disturbance rejection properties. As mentioned, $L(q) = T_c(q)^{-1}$. $Q(q)$ is chosen as

$$Q(q) = F(q)F(q^{-1}),$$

where $F(q)$ is a second order butterworth filter with cutoff frequency 20% of the nyquist frequency. This choice of $Q(q)$ will result in zero phase distortion. Generally, a lower cutoff frequency will make the ILC more robust, but will limit the bandwidth.

As mentioned in Section 5.3, the ILC is applied to the beginning of the forward scan only. Figure 5.2 shows the the result at iteration 0, i.e., when no ILC is running. Figure 5.3 shows the corresponding tracking error. Note that it is only the tracking error that the ILC is taking into account that is shown, i.e., 192 samples. Figures 5.4 to 5.7 show the velocity and tracking error when the ILC is running. The disturbance has been reduced by the ILC, and satisfactory results are obtained after two iterations. Actually, not much improvement can be seen after two iterations, implying that the ILC has converged. Figure 5.8 and Figure 5.9 show the result after five iterations. Since the error is not zero initially, there will be an initial transient as seen in the tracking error figures. However, the instrument starts scanning at grating position -3300 [pulses] and the ILC control sequence is applied at position -3400 [pulses].

The possibility to also use ILC to speed up the closed loop system is clear. It might, however, be advantageous to low pass filter the reference signal in this case, to avoid transients.

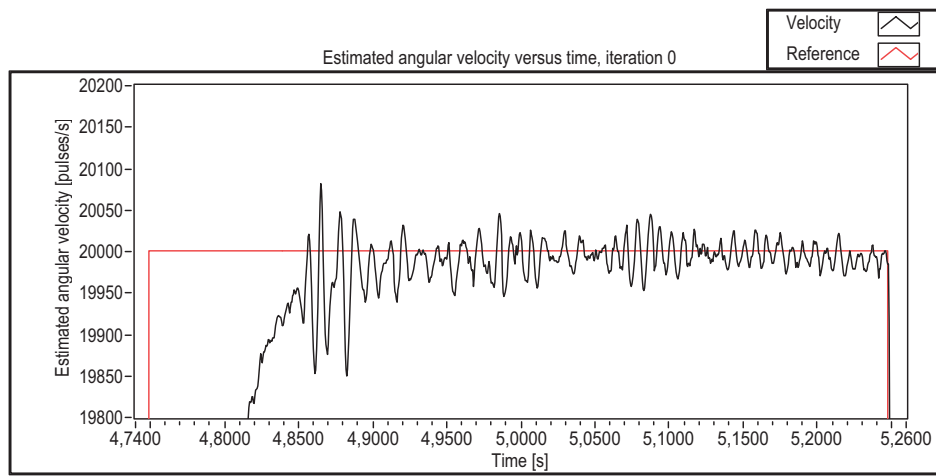


Figure 5.2 Forward scan at iteration 0, i.e., no ILC running.

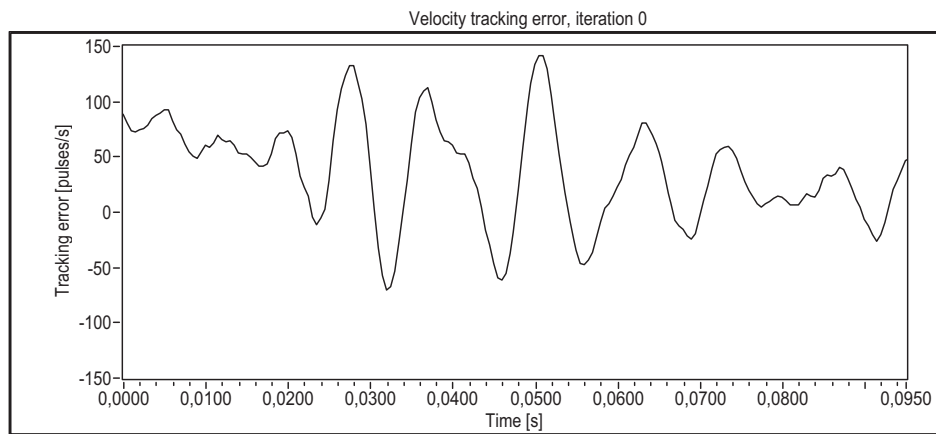


Figure 5.3 Tracking error that is taken into account by the ILC, during forward scan.

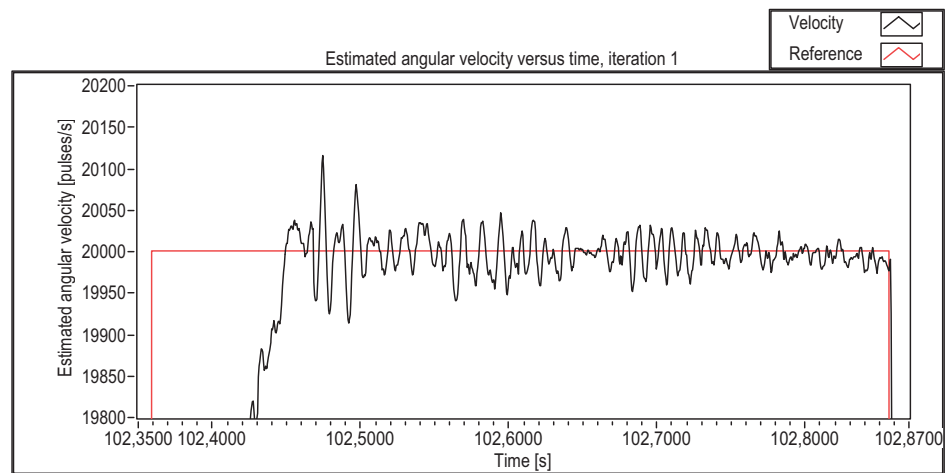


Figure 5.4 Forward scan with ILC, after iteration 1.

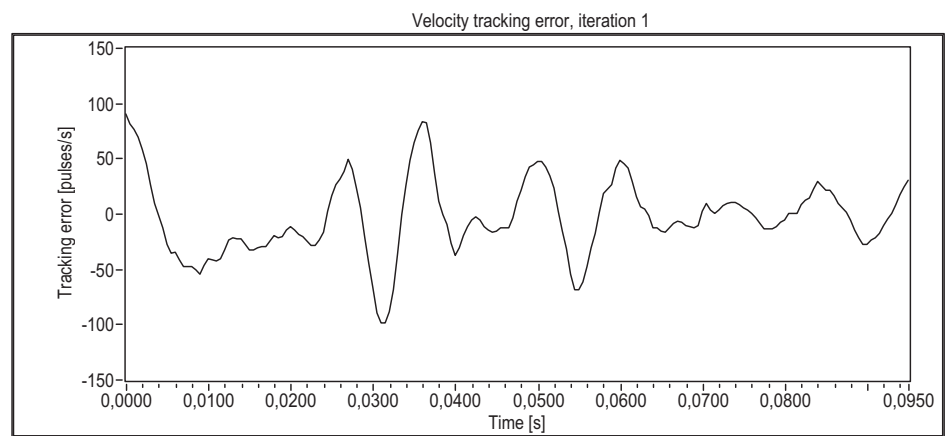


Figure 5.5 Tracking error after iteration 1, during forward scan.

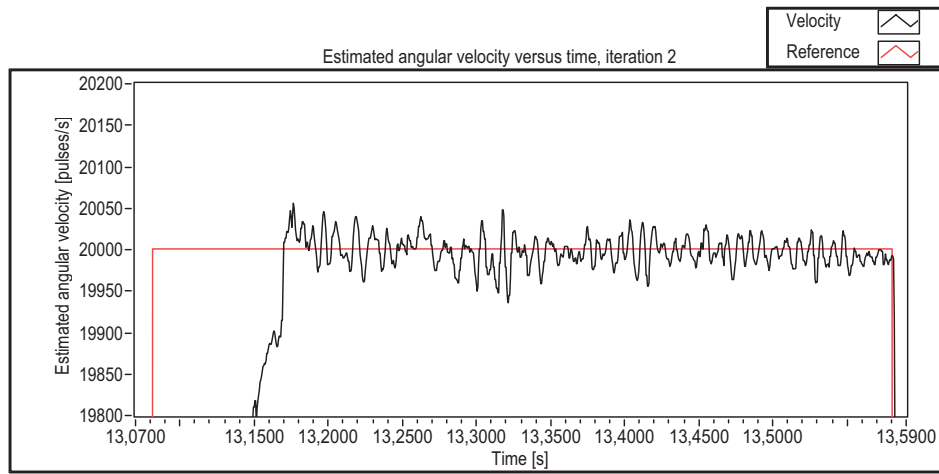


Figure 5.6 Forward scan with ILC, after iteration 2.

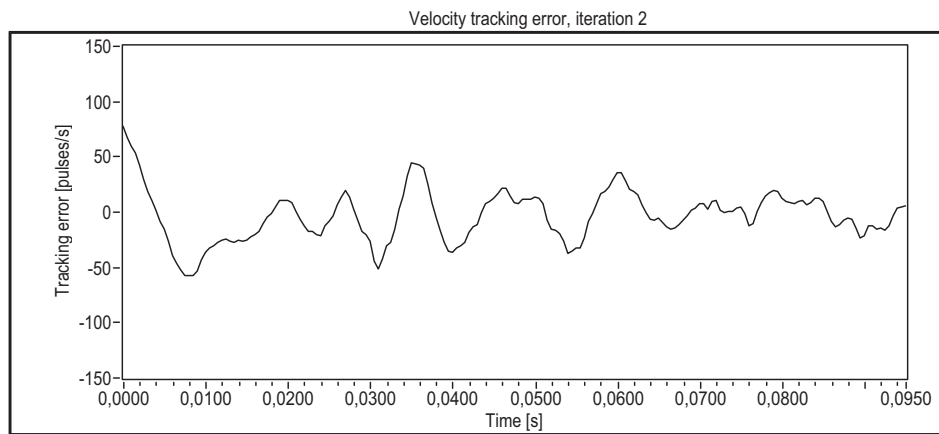


Figure 5.7 Tracking error after iteration 2, during forward scan.

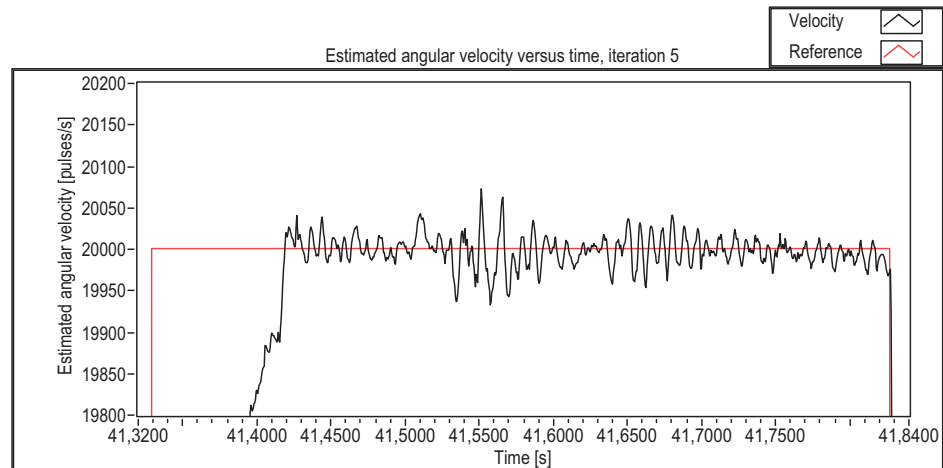


Figure 5.8 Forward scan with ILC, after iteration 5.

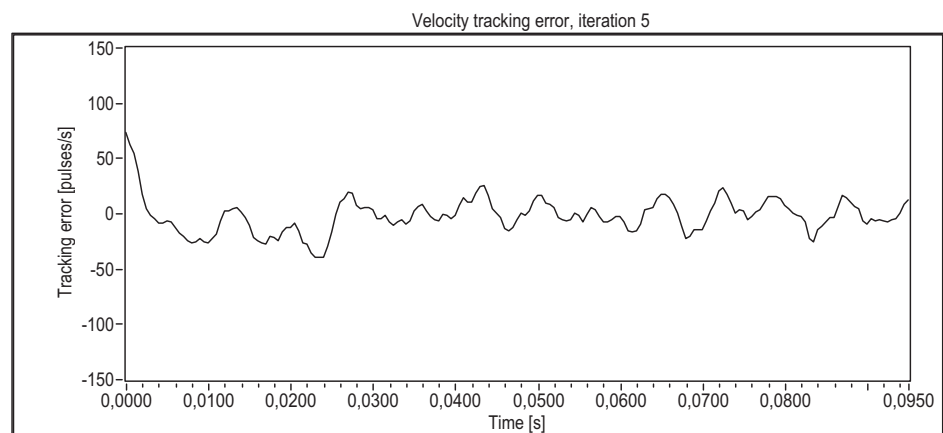


Figure 5.9 Tracking error after iteration 5, during forward scan.

5.5 Summary

In this chapter an ILC control algorithm has been designed for the dc motor. The main objective of the ILC is to attenuate the disturbance introduced in Chapter 4. A few practical issues have been discussed, the discretization leading to a nonminimum-phase system in Section 5.2, and implementation in Section 5.3.

The ILC shows promising results, the disturbance is successfully reduced to acceptable levels after only two iterations. However, due to limited size of the FPGA, an ILC covering the whole scan could not be implemented. Although the focus has been on disturbance rejection, the ILC could also be used to decrease the rise time of the closed loop system.

6. Conclusions and Future Work

6.1 Conclusions

Velocity Control

The focus of this thesis has been on two parts. The first part being velocity control of a dc motor. Experimental results show that the proposed PI-controller performs well. The control objectives are fulfilled, although steady state is not fully reached when the scan begins.

Disturbance Rejection

The second part concerns the attenuation of the disturbance caused by the stepping motor. An iterative control strategy, ILC, shows promising results. Although it was only implemented on a part of the scan, the disturbance on that part was successfully reduced. The problem with reaching steady state in time with only the PI-controller, is also remedied by the ILC. However, a drawback with ILC is that data needs to be stored, which can be troublesome in an embedded system with limited memory.

6.2 Future Work

A few things remain before the proposed control scheme can be considered a complete solution. An ILC implementation that runs on a complete scan remains to be done. Since the disturbance is highly dependent on how the housing is attached and placed, the ILC might not be needed in all cases. Should the ILC be turned off after a few iterations, and how to decide if it should? This should probably be based on some measurement of the tracking error. Perhaps the ILC can be initialized to something more clever than zero values, to reduce the number of iterations needed. These are all questions that should be investigated.

7. Bibliography

- Butterworth, J.A., Pao, L.Y., and Abramovitch, D.Y. 2008, 'The effect of nonminimum-phase zero locations on the performance of feedforward model-inverse control techniques in discrete-time systems', *2008 American Control Conference*.
- Canudas, C., Åström, K.J., and Braun, K. 1986, 'Adaptive Friction Compensation in DC-Motor Drives', *IEEE Journal of Robotics and Automation*, vol. 3, no. 6, pp. 681-685.
- Griffiths, D.J. 1999, *Introduction to Electrodynamics*, third edition, Prentice Hall, New Jersey.
- Hägglund, T. 2009, *Lecture Notes in Automatic Control, Basic Course*, Department of Automatic Control, Lund University, Lund.
- Johansson, R. 2008, *Predictive and Adaptive Control: Lecture Notes*, Department of Automatic Control, Lund University, Lund.
- Krishnan, R. 2001, *Electric Motor Drives: Modeling, Analysis, and Control*, Prentice Hall, New Jersey.
- Mitra, S.K. 2006, *Digital Signal Processing: A Computer-Based Approach*, third edition, McGraw-Hill, New York.
- Norrlof, M. 2000, 'Iterative Learning Control: Analysis, Design, and Experiments', PhD thesis, Division of Automatic Control, Department of Electrical Engineering, Linköping University, Linköping.
- Rashid, M.H. 2001, *Power Electronics Handbook*, Academic Press, San Diego.
- Visioli, A. 2006, *Practical PID Control*, Springer, London.
- Zhang, Y., Smith, I.R., and Kettleborough J.G. 1999, 'Performance Evaluation for a Limited-Angle Torque Motor', *IEEE/ASME Transactions on Mechatronics*, vol. 4, no. 3, pp. 335-339.
- Årzen, K.E. 2008, *Real-Time Control Systems*, Department of Automatic Control, Lund University, Lund.
- Åström, K.J., and Wittenmark, B. 1995, *Adaptive Control*, second edition, Addison-Wesley, Reading, Massachusetts.
- Åström, K.J., and Wittenmark, B. 1997, *Computer-Controlled Systems: Theory and Design*, third edition, Prentice Hall, New Jersey.

A. Nomenclature

A.1 Abbreviations

| | |
|--------|---|
| LAT | Limited Angle Torquer |
| PMBLDC | Permanent Magnet Brushless Direct Current |
| PWM | Pulse-Width Modulation |
| FPGA | Field-Programmable Gate Array |
| PI | Proportional-Integral |
| ILC | Iterative Learning Control |

A.2 Motor Parameters

| Parameter | Symbol | Value | Unit |
|----------------------|----------|-----------|------------------|
| Armature resistance | R_A | 3.7 | Ω |
| Armature inductance | L_A | 0.001 | H |
| Induced emf constant | K_e | 0.0388 | V/rad/s |
| Torque constant | K_t | 0.0388 | Nm/A |
| Load inertia | J_l | 0.000176 | kgm ² |
| Rotor inertia | J_R | 10^{-8} | kgm ² |
| Coulomb friction | β | 0.0083 | Nm |
| Viscous friction | α | 0.00077 | Nm/rad/s |

A.3 Additional Symbols

| Symbol | Description |
|------------|---|
| U | Supply voltage |
| I_A | Armature current |
| E_A | Induced emf in armature windings |
| T_e | Electromagnetic torque |
| T_l | Load torque |
| T_f | Friction torque |
| J | Total inertia, i.e. rotor and load inertia combined |
| ϕ_m | Phase margin |
| g_m | Gain margin |
| ω_c | Crossover frequency |