Angular Velocity Estimation for Sensorless Brushless DC Motors

Victor Martinsson

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
TFRT-6241
ISSN 0280-5316

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

Printed in Sweden by Tryckeriet i E-huset
Lund 2024

Abstract

In the automotive industry today, safety is of highest priority. ASIL is a safety standard that quantifies risk. This project investigates how a redundant speed estimate in a sensorless BLDC motor can be developed to meet ASIL B requirements. The proposed solution involves tracking the shape of the phase curve to determine the angular velocity. This is accomplished by changing the boundaries on the ADC, making it work as a comparator, to find limit events.

The results show that the algorithm successfully tracks the angular velocity. The error residual is noisy, with a maximum mean error of -5.64 rad/s. The error is believed to be due to unexpected disturbances on the phase curve. To get the final estimate, a moving average filter with window size $n = 6$ and $n = 18$ is applied, corresponding to one and three mechanical revolutions. Window size $n = 6$ is preferred to reduce delay, but $n = 18$ is more stable with the present disturbances.

Acknowledgements

I would like to thank my supervisors, Karl-Erik Årzén, Jacob Svendenius and Bertil Ströberg, for their guidance. I would also like to thank BorgWarner in Landskrona for the opportunity, and all the employees that have helped guide me during this project.

Contents

1

Introduction

This thesis project has been done at BorgWarner in Landskrona, Sweden. BorgWarner is a company that develops clean and efficient solutions for the automotive industry, where the Landskrona site develops and produces driveline and propulsion systems. The product that has been used during this thesis project is a coupling device used to transfer torque between axles in a vehicle. The coupling can be used between the rear and front axles in a four-wheel drive vehicle, or between the left and right wheel as a differential brake.

1.1 Background

Today's development in the automotive industry is progressing rapidly, and the same goes for the safety. In recent decades, cars have become more digital each year, and today, almost everything is digitally controlled. With the increased usage of computers in the automotive industry, the possibility of more automatic safety precautions has opened up. In today's cars, many subsystems are implemented, whose assignment is to monitor and check that the different components in the car are functioning as intended. For car manufacturers, certain standards must be followed when developing a new car. The Automotive Safety Integrity Level (ASIL) risk assessment scheme, based on the ISO 26262, Functional Safety for Road Vehicles standard, is a risk assessment scheme that defines what safety precautions are needed, based on how often a specific system malfunctions and how severe the outcome are to passengers if the system is malfunctioning. There are four levels in the ASIL scheme, ASIL A-D, where an ASIL A product has the lowest safety requirements and an ASIL D product has the strictest requirements.

In the coupling device, the torque is transferred between axles by controlling an electric motor. When this electric motor is spinning, a hydraulic pressure is produced that couples the two axles together. The motor is a brushless DC (BLDC) motor without sensors. To be able to control the speed of the motor, a technique called block commutation is used. The speed of the motor is not only essential for the overall control of the transfer torque, it is also important for the safety supervi-

sion of the functionality. The product has been given a ASIL classification, and to satisfy this, two redundant speed estimations are needed. The two speed estimates should be compared so that malfunction can be identified, and safety precautions can be taken.

Figure 1.1 Brushless DC motor used in BorgWarner's clutch. Picture from BorgWarner, Sweden AB.

1.2 Master thesis goals

In this thesis project, a new redundant speed estimate of the BLDC motor will be investigated. This report will cover the thought process behind how this was derived, and also how the two speed estimates should be compared to detect malfunction. An implementation of this estimation algorithm, suitable for the microcontroller that controls the speed of the motor, will also be developed. In this report, the extra functionality is also discussed in terms of extra CPU load.

1.3 Literature review

Estimating the angular velocity in a sensorless BLDC motor can be done in several ways. Literature suggests that the angular velocity can be estimated using the magnitude of the back EMF [Shrutika et al., 2021]. The back EMF, is also periodic, opening up for possible solutions where the angular velocity is estimated using the periodicity of the voltage phase curve. There are literature presenting how auto correlation can be utilized to find periodicity in time series [Breitenbach et al., 2023]. This thesis will investigate how these methods can be used and if they are suitable solutions for the problem given the components available.

1.4 Limitations

In this thesis project, the focus has been on implementing an algorithm that estimates the speed of the motor inside the operating span of 65 to 400 rad/s. This has been done without any modifications to the BLDC motor.

2

Theory

2.1 Brushless DC motor

An electric motor mainly consists of a rotor and a stator and can be either brushed or brushless. The rotor is the part that is rotating, and the stator often sits around the rotor. The motor can be driven by either direct current (DC) or alternating current (AC).

There are some major differences in how the motor works depending on if it is brushed or brushless. In a brushed DC motor, the stator mainly holds field magnets, and the rotor, wire windings. To make the rotor rotate, a commutator is used. The commutator sits on the shaft of the rotor and is used to reverse the polarity of the rotor windings so that the motor keeps rotating. When the motor is rotating because of the attraction between field magnets and rotor windings, the commutator, which is made out of conducting materials, loses its connection with the brushes. The brushes later gets contact with the commutator and the polarity is switched.

Brushed DC motors are easily constructed and does not need any computers since the hardware handles the commutations. The speed of the motor is controlled by changing the supply voltage. One problem with the brushed DC motor is that it needs maintenance, since the brushes are worn out over time because of the contact with the commutator.

In BLDC motors, there are a stator and a rotor as well, but they function differently from those in the brushed DC motor case. In a BLDC motor, the rotor often has the field magnets, and the stator has the wire windings. The biggest difference, which is also one of the main advantages of the BLDC motor, is that the commutator and brushes are removed, reducing its need for maintenance. In a BLDC motor, the commutation is done by changing the direction of the current through the phase windings via an inverter controlled by a microcontroller. The number of stator windings can be chosen arbitrarily, and also the number of pole pairs in the rotor.

One phase in an BLDC motor can be modelled as a resistance R, an inductance L and the back electromagnetic force (EMF) E. The schematic can be seen in Figure 2.1. One phase can be described mathematically by (2.1) [Hughes, 2019].

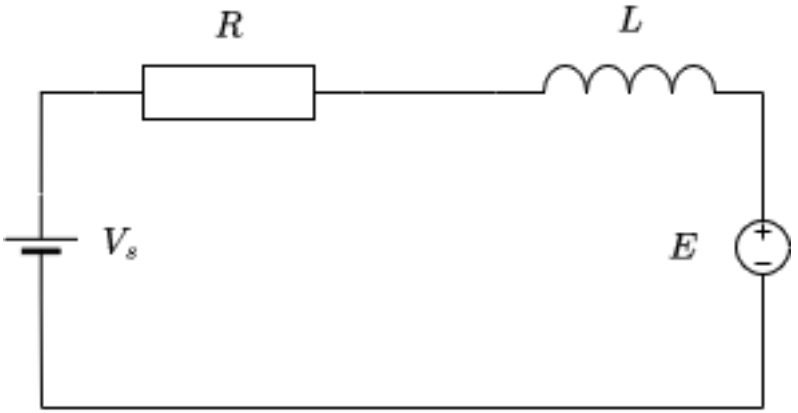

Figure 2.1 Model of one phase with supply voltage V_s, resistance R, inductance L and back EMF E.

$$V_s = Ri + L\frac{di}{dt} + E \tag{2.1}$$

The inductance in the motor is dependent on the change of current inside the motor. For a BLDC motor with three phases, these phases can be connected in two formations called, delta or wye formation. The delta formation can be seen in Figure 2.2 and the wye formation in Figure 2.3. The BLDC motor used in this project uses the wye formation.

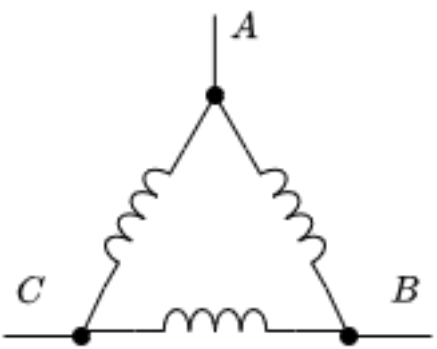

Figure 2.2 Three phases connected in delta formation.

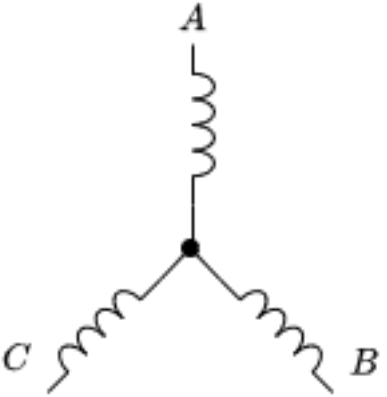

Figure 2.3 Three phases connected in wye formation with the neutral point where the three phases are connected together.

2.2 Inverter

For a three-phase motor, the inverter is built by combining three half bridges, or six transistors, as can be seen in Figure 2.4. By using an inverter, the polarity of the

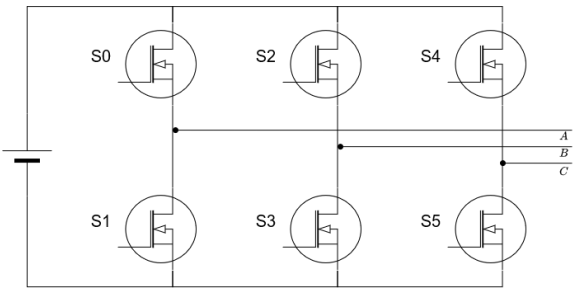

Figure 2.4 Wire diagram for a three-phase inverter with transistors $S0$-$S5$.

windings can be chosen independently. In reality, the six transistors can be chosen to be high or low independently. With these transistors, the states of the three phases A, B and C, are controlled. With six transistors that can be either high or low, there are 64 combinations. The commutation patterns can be represented by a six dimensional vector of logic values, 1 and 0, representing high or low

$$\begin{pmatrix} S0 & S1 & S2 & S3 & S4 & S5 \end{pmatrix}, \tag{2.2}$$

where, Sx represents the state of the transistor x. From (2.2), transistor $S(2n)$, $n = 0,1,2$, determine if the corresponding phase should be high, and $S(2n+1)$ if the

phase should be low. If both $S(2n)$ and $S(2n+1)$ are zero, then that phase is floating, which will be covered later. From this, (2.2) can be written as (2.3).

$$\begin{pmatrix} S0 & S1 & S2 & S3 & S4 & S5 \end{pmatrix} = \begin{pmatrix} A & A' & B & B' & C & C' \end{pmatrix} \tag{2.3}$$

In (2.3), A is representing if phase A should be high, and A', if phase A should be low. From (2.3), both $S(2n)$ and $S(2m)$, $m = 0,1,2$ and $n \neq m$, can be high at once, meaning that two or more phases can be positive at the same time. It is also true that $S(2n)$ and $S(2n+1)$ can be high at the same time, which would mean that the corresponding phase is both low and high at the same time. These combinations are in reality of no use and are ignored. Since the state of each transistor can be changed independently, all the 64 combinations are feasible, but for the purpose of driving the motor only six combinations are valid. By removing the combinations which do not fulfil all the constraints (2.4), (2.5) and (2.6), the patterns in Table 2.1 remains.

$$\sum_{n=0}^{2} S(2n) = 1 \tag{2.4}$$

$$\sum_{n=0}^{2} S(2n+1) = 1 \tag{2.5}$$

$$S(2n) + S(2n+1) \leq 1, n = 0,1,2 \tag{2.6}$$

Table 2.1 Valid combinations for transistor states with corresponding high, low and floating phases.

High	Low	Floating	S0	S1	S2	S3	S4	S5
C	B	A	0	0	0	1	1	0
B	C	A	0	0	1	0	0	1
C	A	B	0	1	0	0	1	0
B	A	C	0	1	1	0	0	0
A	C	B	1	0	0	0	0	1
A	B	C	1	0	0	1	0	0

2.3 Back electro-magnetic force

Back EMF is a voltage generated inside the motor to counter the supplied voltage to the motor. For a three-phase motor, the back EMF can be measured on the floating phase, the phase that the supply voltage does not pass through. When the motor rotates, the magnetic field changes. This magnetic field from the rotor magnets induces a voltage in the stator winding that is not active, in the opposite direction of

the supply voltage, therefore the name back EMF. Since the back EMF is changing due to the rotor magnet's position, this can give information about the rotors angular position.

The back EMF E is proportional to the electric angular velocity [Shrutika et al., 2021]. The electrical angular velocity ω_e can be calculated as

$$\omega_e = \frac{E}{k_e}, \tag{2.7}$$

where k_e is a back EMF constant specific to the motor.

2.4 Block commutation

There are a couple of techniques for how to drive electrical motors. These techniques can generally be divided into two groups: control using sensors and sensorless control, where sensorless control is not entirely sensorless, as voltages and currents are measured. When using a BLDC motor with sensors, the position of the rotor can be determined directly and thus the correct commutation pattern be chosen. Examples of sensors that can be used are hall sensors and encoders. Hall sensors sense the magnetic field produced from the permanent magnets in the rotor and can thus read its position. Encoders on the other hand uses reflected light to determine the position and speed of the rotor.

Sensorless control is often used in applications where the rotor position is of no particular interest. A sensorless motor is also cheaper since it does not require any additional components. For sensorless control of a BLDC motor, the speed of the motor is determined through the time between commutations. When the inverter should switch between the different commutation patterns in Table 2.1 is determined by measuring the back EMF on the floating phase. In a BLDC motor, the back EMF has a trapezoidal waveform. When the trapezoidal back EMF crosses a certain threshold, based on the supplied voltage to the motor, the commutation pattern is changed. From Table 2.1, each phase is floating during two periods in one electrical revolution, and one phase is floating in each pattern representing 60° of an electrical revolution. The position of the rotor can thus be estimated every 60°, and the commutation pattern be changed.

The correct sequence of the switching patterns in Table 2.1 can be seen in Table 2.2, where the switching sequence is 1-2-3-4-5-6-1 [Naveen and Isha, 2017]. The typical phase voltage shape for one phase during one electrical revolution can be seen in Figure 2.5. The spikes during the floating phase are introduced due to change of commutation pattern, when the phase goes from high or low to floating. When the phase changes to floating, a discharge, due to the inductance, is occurring. With a greater angular velocity, more current is flowing through the motor, implying a larger discharge, hence a larger spike.

Table 2.2 Sequence of commutations patterns for driving motor forward.

Sequence	High	Low	Floating
1	C	B	A
2	A	B	C
3	A	C	B
4	B	C	A
5	B	A	C
6	C	A	B

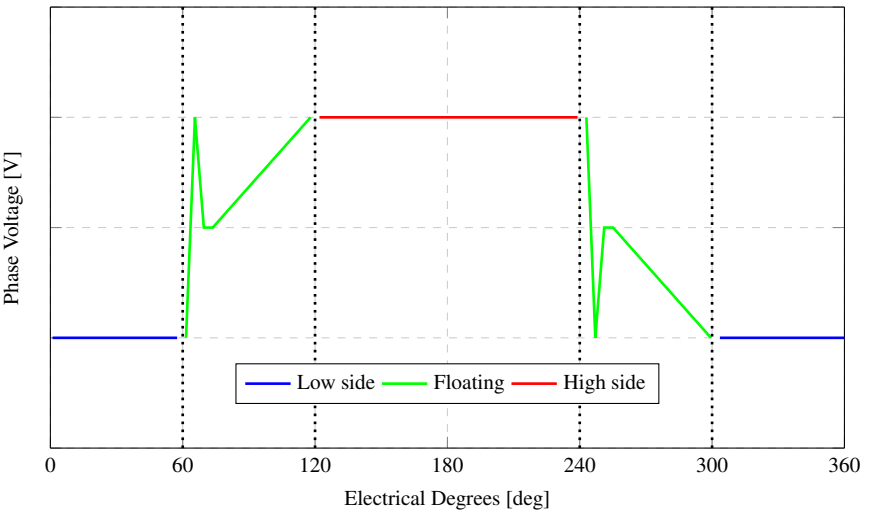

Figure 2.5 A typical shape of the phase voltage during one electrical revolution. The commutation pattern is changed every 60 electrical degrees. Black lines indicate when the phase is changing state between high, low and floating.

The speed of the motor can be controlled by either changing the supplied voltage to the motor, or by keeping the supply voltage constant and using pulse width modulation (PWM). With PWM, the control signal is the duty cycle of the PWM signal, and the duty cycle is used to turn the high or low side transistors in the inverter on and off.

In this project, the motor uses block commutation. The electrical angular velocity ω_e in rad/s can be calculated as

$$\omega_e = 2\pi \frac{1}{6t_c}, \tag{2.8}$$

where t_c is the time in seconds between two commutations. The mechanical angular

velocity ω_m in rad/s is then equal to

$$\omega_m = \frac{2}{P}\omega_e, \tag{2.9}$$

where P is the number of poles in the rotor [Shrutika et al., 2021]. From (2.9), the electrical angular velocity is N times the mechanical velocity, where N is the number of pole pairs.

2.5 Pulse width modulation

PWM is used to control the speed of the motor. PWM is a type of voltage modulation, where one cycle has a period of T seconds. During these T seconds, the signal is high during a duty cycle of T_c seconds, $T_c \leq T$, which controls how much voltage passes through the circuit during a period of time. The ADCs used in this project are synced to sample with a sample period of $T_s = T$. To get useful voltage data, the samples are taken in the middle of the duty cycle of the PWM.

2.6 Autocorrelation

The autocorrelation function (ACF) is a measure of linearly dependency between every two data points y_i and y_l, where $i - l = k$. The AFC is defined as

$$\rho(k) = \frac{r(k)}{r(0)}, \tag{2.10}$$

where $r(k)$ is the autocovariance function, and $r(0)$ is the variance [Jakobsson, 2013]. The ACF is a powerful tool that gives information about seasonality in a series of data points. The ACF can be used to identify the time between periodically occurring events. The lag k that maximizes

$$k = \arg\max_k \rho(k) \tag{2.11}$$

can be used to calculate the time between periodically occurring events. Lag k in (2.11), corresponds to the number of data points from y_i to y_{i+k}, where $0 \leq i < N - k$ and N is the number of data points used to calculate the ACF. The data points y_i and y_l, with $i - l = k$, are therefore the most linear dependent and k, the period of the time series. With k, the period time T_p can be calculated as

$$T_p = \frac{k}{f_s}, \tag{2.12}$$

where f_s is the sample time. To find the k that corresponds to the number of data points between the most negatively linear dependent data points, can be calculated

using (2.13).

$$k = \arg\min_k \rho(k) \tag{2.13}$$

With k from (2.13), a half period can be found.

When calculating the ACF $\rho(k)$ at lag k for a sequence of N data points, there are $N-k$ combinations that are used to calculate the auto correlation. From this, it follows that the largest lag k for the ACF is $k = N-1$, since, $N-k > 0$ must hold, otherwise, no combinations of data points are used to calculate the correlation. From above, it is clear that for $k = N-1$, only one combination is available, which for a time series is not enough to determine seasonality. If any noise is present in the measurements, even lower lags are untrustworthy. A rule of thumb is to only use and trust the ACF for lags up to at most $N/4$ [Jakobsson, 2013].

For perfect periodic time series without noise, the ACF will have a local maxima equal to 1. With additional noise, the time series no longer is periodic in the sense that it is a time shifted copy of the past. The ACF, however, still can pick up on the regularity in the time series and the auto correlation approaches 1 [Breitenbach et al., 2023].

2.7 Kalman filter

The Kalman filter is a filtering and prediction technique that utilizes the linear dynamics of a system, and system and measurement noise to estimate states [*Introduction and Implementations of the Kalman Filter. [Elektronisk resurs].* 2019]. The process model can be mathematically described as

$$\boldsymbol{x}_k = \Phi \boldsymbol{x}_{k-1} + \Gamma \mathbf{u}_{k-1} + \boldsymbol{w}_{k-1}, \tag{2.14}$$

where $\boldsymbol{x}_k$ is the state vector at time k, Φ and Γ are state transition and control matrices respectively, $\mathbf{u}_{k-1}$ is the control input and $\boldsymbol{w}_{k-1}$ is the system noise, with noise levels defined in the covariance matrix $\boldsymbol{Q}$. Together with the measurement model

$$\boldsymbol{y}_k = \mathbf{C}\boldsymbol{x}_k + \boldsymbol{v}_k, \tag{2.15}$$

where $\boldsymbol{y}_k$ is the measured states at time k, $\boldsymbol{C}$ is the measurement matrix that determines what states are measured and $\boldsymbol{v}_k$ is the measurement noise, the states can be filtered and estimated. The noise levels of $\boldsymbol{v}_k$ is defined in the covariance matrix $\boldsymbol{R}$. The idea is that given accurate information about system and measurement noise, estimate the current states, and then update covariance matrices and Kalman gain, given the error between estimates and measurements. The result is that hidden states from measurements, disturbed by noise, can be estimated in an adaptive way.

3

Equipment

The equipment used in this work includes both hardware and software. Constraints on the equipment will be discussed in Section 3.3.

3.1 Hardware

The hardware consists of the BLDC motor itself, the test rig, and the Motor control unit (MCU).

BLDC motor

The BLDC motor used is in-house built at BorgWarner and is used in products today. The motor is a three-phase motor with three pole pairs. The motor with ECU and hydraulic pump can be seen in Figure 3.1.

Figure 3.1 BLDC motor used in this thesis work. Picture from BorgWarner, Sweden AB.

Test rig

The equipment used in the test rig is the BLDC motor, the MCU with circuit boards, and a module that simulates the hydraulic pressure built up when the BLDC motor is spinning. To run the BLDC motor, a power supply is used. To measure the phase voltages, a PicoScope oscilloscope module is used that connects to a PC. The PC is also used to send commands via CAN. The module in the test rig, simulating the oil pressure, can be seen in Figure 3.2. In Figure 3.3, the power supply from DELTA ELEKTRONIKA can be seen.

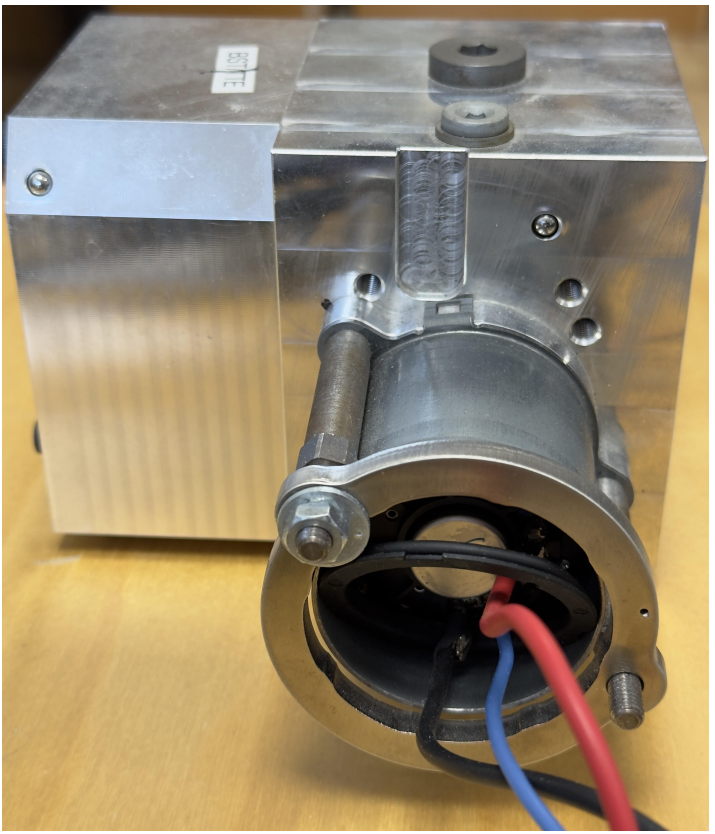

Figure 3.2 Module filled with oil, simulating the increased oil pressure for different angular velocities. The BLDC motor is mounted in front, where also the three cables, red, blue and black, for the three phases can be seen.

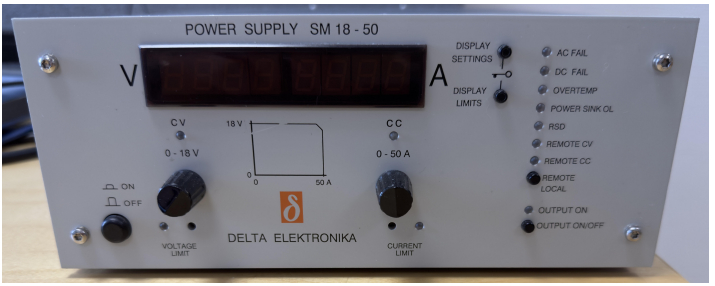

Figure 3.3 Power supply SM 18 - 50 from DELTA ELEKTRONIKA used during experiments.

MCU

The MCU used is a simpler 32-bit microcontroller, with 16-bit timers, in the TriCore Aurix series from Infineon. The MCU has ADC channels that can be configured. In current products, only one channel is used, but for this project, another channel has been set up using the circuit board in Figure 3.4. The motor control is implemented with fix-point arithmetic.

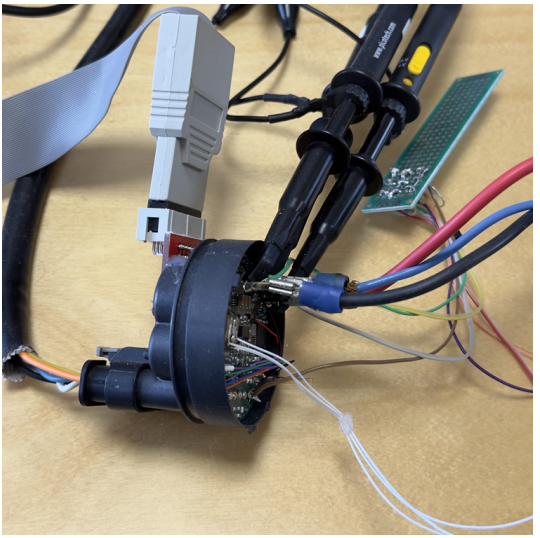

Figure 3.4 The MCU is mounted in the black part in the figure, where all wires are connected. An additional circuit board is used to use the additional ADC channel. The three cables from the BLDC motor, red, blue and black, are connected to the MCU.

3.2 Software

The software flashed onto the MCU is written in C. To send commands to the MCU from a PC for different run modes and to change the reference speed for the BLDC motor, CAN communication is used.

CANalyzer

CANalyzer is a software tool from Vector Informatik GmbH that allows the user to send CAN messages. The software requires a license that is provided by Borg-Warner and is located in the USB module that can be seen in Figure 3.5.

Figure 3.5 CANalyzer usb module.

3.3 Constraints

The MCU used in this thesis project is a 32-bit fixed point microcontroller, with 16-bit timers. With only 16-bits timers, overflows can be a problem if more than one turn around on the clock has happened between two events where the time is retrieved. Also, when working with fixed point, the resolution is affected when making calculations with decimals.

4

Method Selection

There are in theory a couple of methods that can be used to estimate the angular velocity of the rotor. In this section, three methods will be covered, where two are tested using simulations in Matlab and Simulink, and a third is discussed. In Section 4.1, the magnitude of the back EMF on the floating phase will be used to estimate the angular velocity of the rotor. Section 4.2 will cover a more statistical approach, based on time series. In Section 4.3, a simpler algorithm will be discussed that tries to find periodic patterns in the back EMF. Section 4.4 will conclude what method best suits the problem, given the constraints in Section 3.3.

4.1 Back EMF proportionality to angular velocity

The magnitude of the back EMF is proportional to the electrical angular velocity of the rotor, as can be seen from (2.7). To utilize this phenomenon, it is vital to extract the back EMF from the three phases. As can be seen in Table 2.2, the floating phase will in the forward direction switch in the sequence A-C-B-A. By identifying the floating phase and then follow this sequence, given that the rotational direction is known, the back EMF values can be used to calculate the angular velocity. The back EMF samples can be noisy, and the method presented in this section is therefore based on filtering the noise with a Kalman filter [Wang and Lee, 2015].

The measurement is the back EMF E, and the hidden state that will be estimated is the angular velocity ω_m. The Kalman filter predicts the back EMF E_{est} using the estimated electrical angular velocity ω_e in $\hat{\boldsymbol{x}}_k$, and the noise levels $\boldsymbol{w}_k$ and $\boldsymbol{v}_k$, specified in the process and measurment noise covariance matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$. The measurement matrix $\boldsymbol{C}$ can be set to

$$\boldsymbol{C} = k_e \tag{4.1}$$

to produce the estimated back EMF $\hat{\boldsymbol{y}}_{k+1} = E_{est}$ through (4.2).

$$\hat{\boldsymbol{y}}_k = \boldsymbol{C}\hat{\boldsymbol{x}}_{k-1} \tag{4.2}$$

The Kalman gain $\boldsymbol{K}_k$ is then computed as

$$\boldsymbol{K}_k = \boldsymbol{P}_{k,k-1}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_{k,k-1}\boldsymbol{C}^T + \boldsymbol{R})^{-1}, \tag{4.3}$$

where covariance matrix $\boldsymbol{P}$ is predicted and updated as (4.4) and (4.5).

$$\boldsymbol{P}_{k+1,k} = \Phi\boldsymbol{P}_{k,k}\Phi^T + \boldsymbol{Q} \tag{4.4}$$

$$\boldsymbol{P}_{k,k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{C})\boldsymbol{P}_{k,k-1}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{C})^T + \boldsymbol{K}_k\boldsymbol{R}\boldsymbol{K}_k^T \tag{4.5}$$

The Kalman gain $\boldsymbol{K}_k$ is used to update the hidden states depending on the magnitude of the error through

$$\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_k + \boldsymbol{K}_k(E_k - \hat{\boldsymbol{y}}_k), \tag{4.6}$$

to better match the true back EMF E_k implying that the error between ω_e and the estimated mechanical angular velocity $\hat{\boldsymbol{x}}_k$ decreases.

One assumption is that the motor, at time k and $k+1$, runs at the same speed. Therefore, the state transition matrix Φ is set to $\boldsymbol{I}$, implying $\hat{\boldsymbol{x}}_{k+1} = \hat{\boldsymbol{x}}_k$. In this implementation, where $\hat{\boldsymbol{x}}_k$ only has one hidden state, the state transition matrix Φ can be set to 1. The summarized state space representation can be seen in (4.7).

$$\begin{cases} \Phi &= 1 \\ \Gamma &= 0 \\ \boldsymbol{C} &= k_e \end{cases} \tag{4.7}$$

In each iteration of the Kalman filter, after predicting and filtering, the hidden state $\hat{\boldsymbol{x}}_k$ is saved. Since only the magnitude of the back EMF is of interest, the absolute value of the back EMF is used.

To evaluate this method, a couple of simulated step responses are used. The constant k_e is motor specific and found trough experiments. In the simulations, two poles are used in the motor, which implies that $\omega_m = \omega_e$.

Below in Figure 4.1, 4.2 and 4.3, 1-second step responses for 104.7 rad/s (1000 RPM), 209.4 rad/s (2000 RPM), and 314.2 rad/s (3000 RPM) can be seen. In all figures, the same k_e constant is used.

In the first step response, it can be determined that the estimation performs well during steady state conditions. Less delay can be achieved by changing the system and measurement matrices in the filter. In this case, the trade-off is that the magnitude of the oscillations increases.

In the second step response, the filter is performing well during steady state conditions, but small oscillations are present. There are also less delay in the rising phase.

In the third step response, the oscillations are not visible during steady state conditions. During the rising phase, the estimate first overshoots, but later converges towards the true angular velocity.

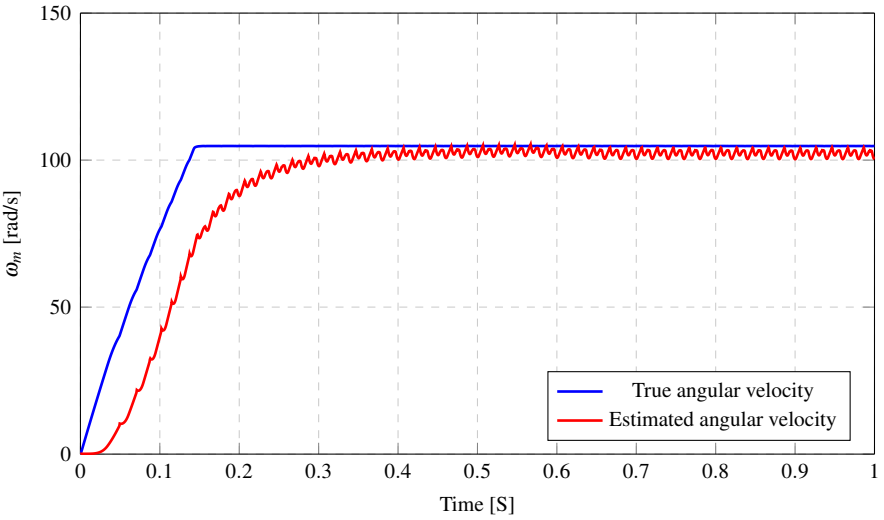

Figure 4.1 1-second step response with a reference speed of 104.7 rad/s or equivalently 1000 RPM. The filter performs good during steady state conditions but are delayed during the rising phase of the step response.

In the simulations, a constant rotating inertia is used, even though the oil pressure in reality is increasing when the speed increases. This simplification should not affect the behavior, since the back EMF is proportional to the angular velocity during steady state, as the figures above also suggests.

4.2 Period detection using auto correlation function

When using auto correlation, there are two theoretical choices of time series that can be used. The first being one phase as in Figure 2.1, and the other being the current floating phase. Which one is chosen does not affect the calculations, although, it is worth noting that the seasonality in the ACF will change.

Maximizing the ACF

The amount of input data points used to calculate the ACF must first be determined. This is due to that the number of input data points sets the limit for which seasonalities can be detected. The mechanical angular velocity ω_m can be calculated as

$$\omega_m = 2\pi \frac{2}{P\alpha\beta T_s k}, \tag{4.8}$$

where T_s is the sampling period, P is the number of poles, $\alpha = 1$ if the time series is for one phase and $\alpha = 3$ if the time series is the current floating phase, and $\beta = 1$

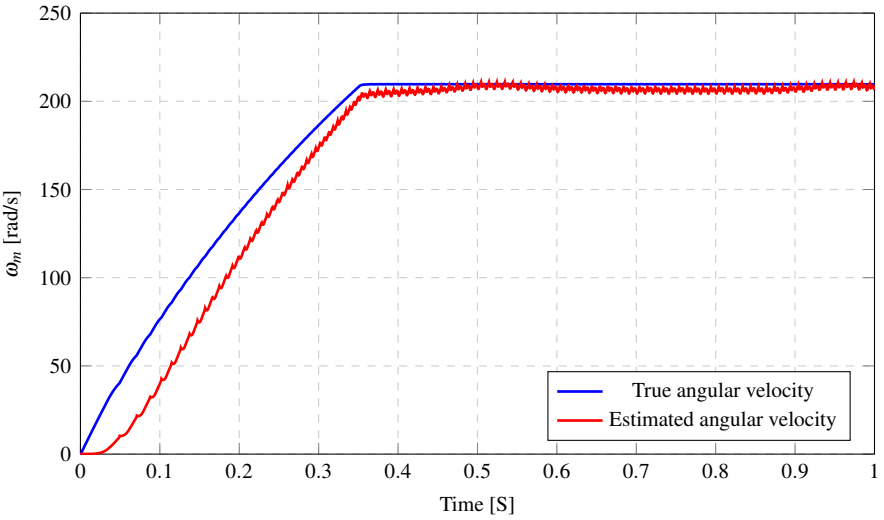

Figure 4.2 1-second step response with a reference speed of 209.4 rad/s or equivalently 2000 RPM. The filter performs good during steady state conditions, but some oscillations are present. Less delay during the rising phase of the step response.

if the ACF are maximized using (2.11) and $\beta = 2$ if the ACF are minimized using (2.13). If the rule of thumb, $k_{max} = N/4$, is applied, (4.8) can be written as (4.9) [Jakobsson, 2013].

$$\omega_{m_{min}} = 2\pi \frac{8}{P\alpha\beta T_s N} \tag{4.9}$$

The result is a function describing the lowest electrical angular velocity that can be calculated given N. From (4.8) and (4.9), the relationship can be written as

$$2\pi \frac{8}{P\alpha\beta T_s N} \leq \omega_m \leq 2\pi \frac{2}{P\alpha\beta T_s}, \tag{4.10}$$

where it can be seen that N only affects the lower limit. If $\alpha = 3$, $\beta = 1$, $f_s = 20$ kHz, and $P = 6$, then the relationship between $\omega_{m_{min}}$ and N can be written as

$$\omega_{m_{min}}(N) = 16\pi \frac{20 \cdot 10^3}{18N}, \tag{4.11}$$

where the plot of the function can be seen in Figure 4.4. Figure 4.4 shows that the number of data points needed for $\omega_{m_{min}} \approx 0$ requires a large N. To determine the size of N, $\omega_{m_{min}}$ must be set, to be able to solve (4.11) for N. Since the minimum and maximum operating speed is 65 and 400 rad/s, setting $\omega_{m_{min}} \approx 25$ rad/s seams reasonable, to detect the rising phase of a startup, yielding $N \approx 2234$. With a larger N, a larger delay is introduced when more than one period fits into the N data point.

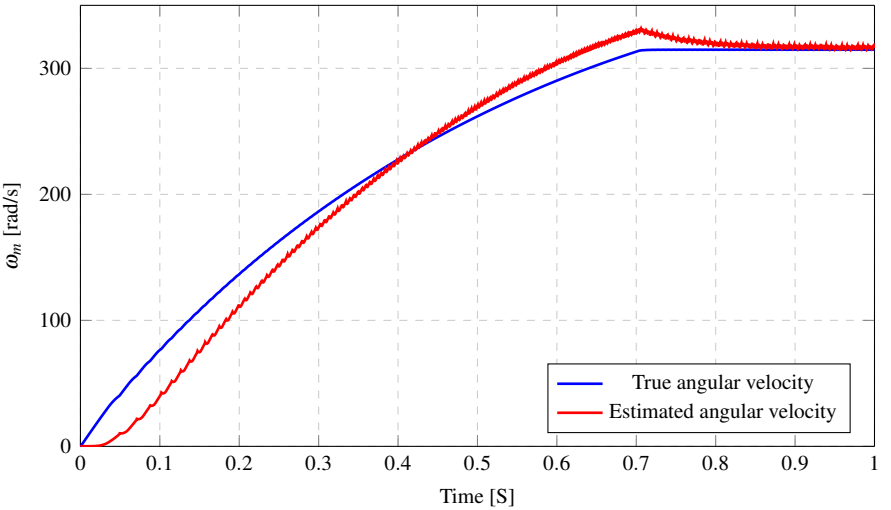

Figure 4.3 1-second step response with a reference speed of 314.2 rad/s or equivalently 3000 RPM. The filter performs good during steady state conditions, but some oscillations are present. Less delay during the rising phase of the step response.

In Figure 4.5, a simulated time series for the current floating phase can be seen, where the BLDC motor has 2 poles. By performing the same calculations as above, $N \approx 6700$. The corresponding ACF for this time series is plotted in Figure 4.6. Clearly, maximizing k yields lag $k = 0$ since this is the variance of the time series, and the same data point y_k is 100% linearly dependent with itself. Worth noting is that (4.8) is undefined for $k = 0$. Setting $k = 1$ in (4.8), yields $\omega_m \approx 42000$ rad/s, where $\omega_m >> 400$ rad/s, which is the maximum operating speed. Setting the allowed maximum mechanical angular velocity that can be calculated to $\omega_{m_{max}} \approx 500$ rad/s, yields a minimum $k_{min} \approx 84$, implying that the first 84 ACF values in $\rho(k)$ can be ignored, and can be set to 0. The resulting ACF can be seen in Figure 4.7, where the maximum k in (2.11) now clearly is inside the limits.

Minimizing the ACF

Another approach for estimating the motor speed is to use (2.13) and, $\beta = 2$ in (4.8), to find a half period. Again, the relationship between $\omega_{m_{min}}$ and N can be written using $P = 6$, $\alpha = 3$, $\beta = 2$, and $f_s = 20$ kHz. The relationship is

$$\omega_{m_{min}}(N) = 16\pi \frac{20 \cdot 10^3}{36N}, \tag{4.12}$$

which for $\omega_{m_{min}} = 25$ rad/s yields $N \approx 1117$, which dramatically changes the number of calculations needed to compute the ACF compared to $N \approx 2234$, when maximizing. Another advantage with minimizing rather than maximizing the ACF is that

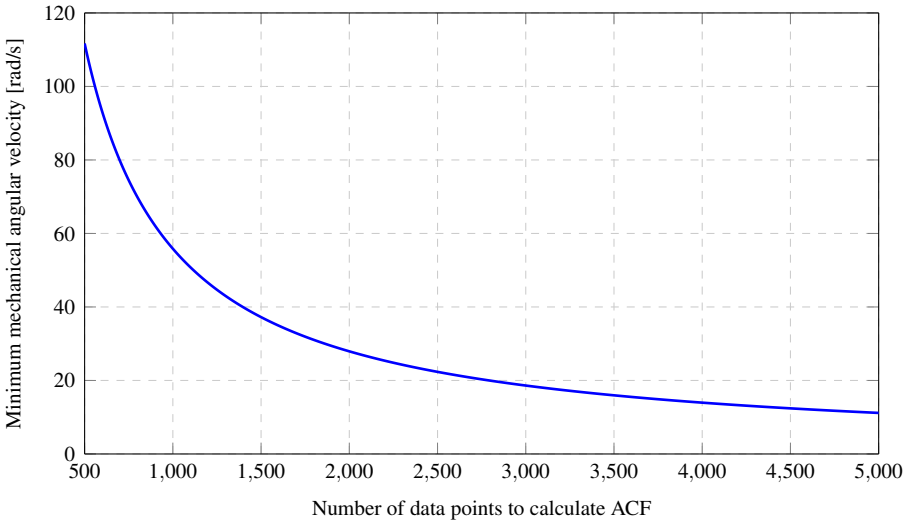

Figure 4.4 Relationship between $\omega_{m_{min}}$ and N for $P = 6$, $T_s = 20$ kHz, $\alpha = 3$, and $\beta = 2$.

k in (2.13) can be found directly in $\rho(k)$, without worrying about the first values in the ACF, since $\rho(k)$ always starts at 1, as, again, can be seen in Figure 4.6.

Using simulated data, the number of data points is reduced down to $N = 3350$ trough (4.9) with $\alpha = 3$, $\beta = 2$, $f_s = 20$ kHz, and $P = 2$, implying that the ACF for the floating phase input sequence is minimized for k. The pseudocode for the implemented algorithm is presented in Algorithm 1. During the first phase, there are not enough data points to calculate the ACF. There is therefore a counter that is checked to be at least N. If this is not fulfilled, the input to the ACF function is filled with zeros so that the resulting input vector is N long. The ACF is calculated, but only the first quarter of the resulting vector is considered as reliable. The argument that minimizes the ACF is then used to calculate the current speed. To evaluate this method, data from a 1 second step response is used together with Algorithm 1. In Figure 4.8, the true mechanical angular velocity as well as the estimated mechanical velocity can be seen. Algorithm 1 can be seen to perform well during steady state, where it tracks the angular velocity perfectly. During the rising phase of the step response, the estimate is not very smooth, but also a bit delayed. To make the estimate smoother, a simple Kalman filter without any dynamics can be used, as presented

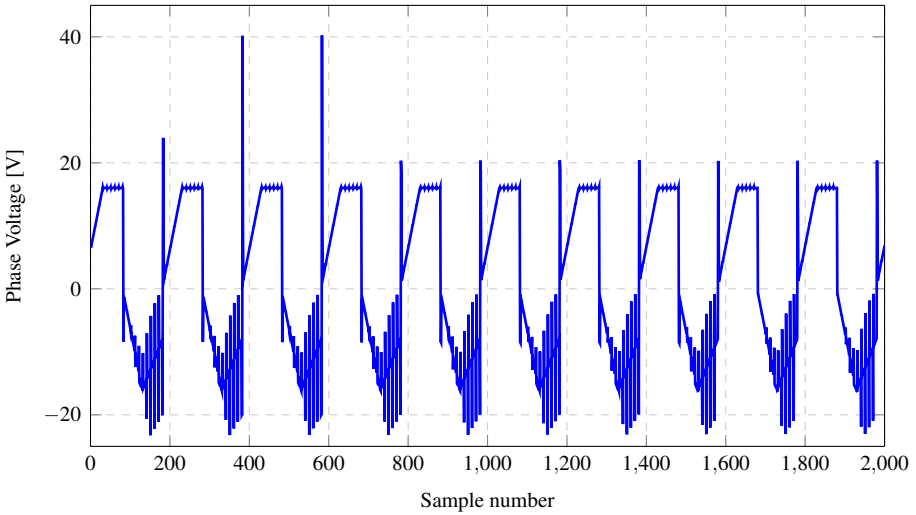

Figure 4.5 Simulated time series, where data is taken from the floating phase at the current time.

Algorithm 1 Period detection for speed estimation

$y \leftarrow input$
$N \leftarrow number of elements$
$t \leftarrow 1$ ▷ Set counter to 1.
while True **do**
  if $t \geq N$ **then** ▷ When N data points are available.
    $x \leftarrow y(t-N:t)$
  else ▷ When only t data points are available, fill with $N-t$ zeros.
    $x \leftarrow [zeros(N-t), y(1:t)]$
  end if
  $\rho \leftarrow ACF(x)$ ▷ Compute the ACF.
  $\rho \leftarrow \rho(1:round(N/4))$ ▷ Only the first fourth of the ACF is used.
  $k \leftarrow \arg\min_k \rho(k)$ ▷ Find k that minimizes the ACF.
  $speed \leftarrow calculate\_speed(P, \alpha, \beta, T_s, k)$ ▷ Calculate the speed based on k.
  $t \leftarrow t+1$
end while

in (4.13).

$$\begin{cases} \Phi & = 1 \\ \Gamma & = 0 \\ \boldsymbol{C} & = 1 \\ \boldsymbol{Q} & = 5 \cdot 10^{-3} \\ \boldsymbol{R} & = 100 \end{cases} \tag{4.13}$$

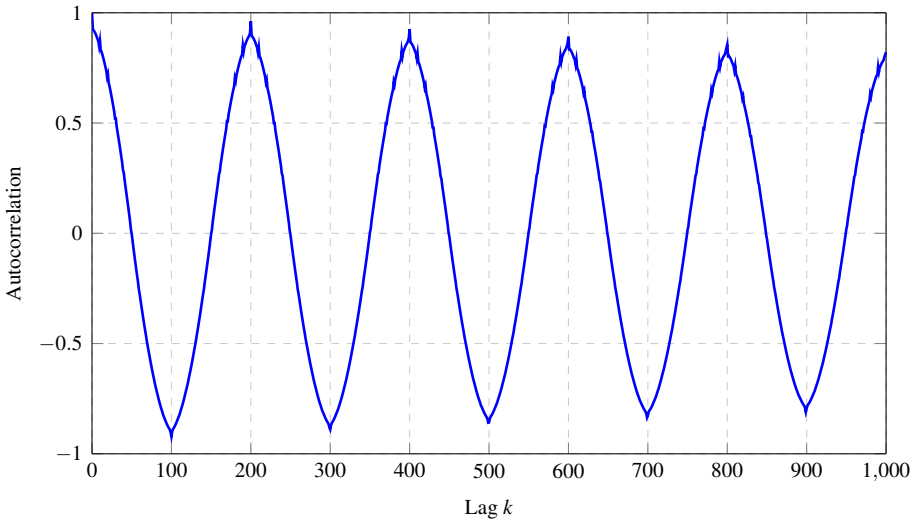

Figure 4.6 Auto correlation function plotted for 1000 lags, with $N = 6700$. Maximum is found at $\rho(0)$ which is of no interest.

The resulting estimation when filtered can be seen in Figure 4.9, where the estimate is smoothened out. As mentioned earlier, the size of N has an impact on the delay for the estimate to settle in during steady state. By ignoring the rule of thumb, $k_{max} = N/4$, and letting $k_{max} = N$, where N is reduced to a fourth from $N = 3350$ to $N = 838$, fewer periods fit. The drawback is that fever combinations are available for computing lower angular velocities.

The resulting estimate can be seen in Figure 4.10, where it can be seen that the rising phase of the step response still is uneven. It can however be seen that there is less delay. By applying the same Kalman filter with no dynamics with the system and measurement matrices,

$$\begin{cases} \boldsymbol{Q} &= 5 \cdot 10^{-3} \\ \boldsymbol{R} &= 100 \end{cases}, \tag{4.14}$$

the estimate is smoothened out. The resulting filtered estimate without the rule of thumb, using $N = 838$, and the estimate with the rule of thumb, using $N = 3350$, can be seen in Figure 4.11. The estimate without the rule of thumb can clearly be seen to perform better, since the delay is heavily reduced and the estimate is tracking the angular velocity perfectly during steady state.

Important noting is also that N can be heavily reduced by decreasing f_s. This, however, comes at the cost of decreased resolution in the estimate.

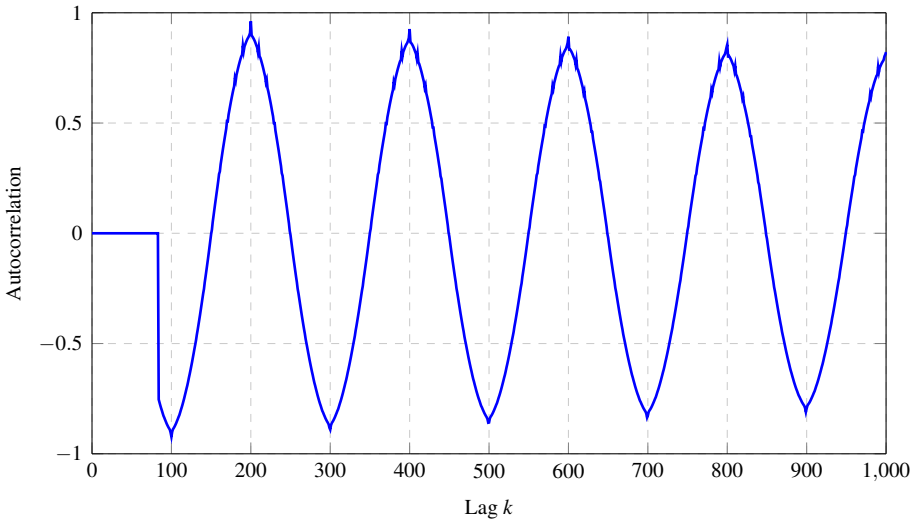

Figure 4.7 Auto correlation function plotted for 1000 lags, with $N = 6700$, where the first 84 data points are set to 0. Maximum is now found at $k = 200$ which corresponds to a mechanical angular velocity of 209.4 rad/s or 2000 RPM.

4.3 Interrupt based pattern recognition

The MCU has functionality that allows the user to set up an analog-to-digital converter (ADC) channel. This ADC channel has the sample frequency $f_s = 20$ kHz and works as a comparator. When the sampled value falls outside or inside a defined range $[b_0, b_1]$, an interrupt is requested from the hardware to the operating system (OS). The interrupt is handled by the OS on the MCU and is routed to an interrupt event handler when it has occurred.

By choosing the boundaries b_0 and b_1 wisely at certain time points, the time between interrupts can be converted to angular velocity.

Referring back to Figure 2.5 in Section 2.4, where a typical wave form of the phase voltage can be seen, the waveform has peaks when the phase switches to floating. These peaks occur every half electrical revolution, meaning that the electrical angular velocity in rad/s can be calculated using

$$\omega_e = 2\pi \frac{1}{2\Delta t}, \tag{4.15}$$

where Δt is the time in seconds between two peaks. Further, the mechanical angular velocity can be calculated using (2.9), which simplifies down to

$$\omega_m = 2\pi \frac{1}{P\Delta t}, \tag{4.16}$$

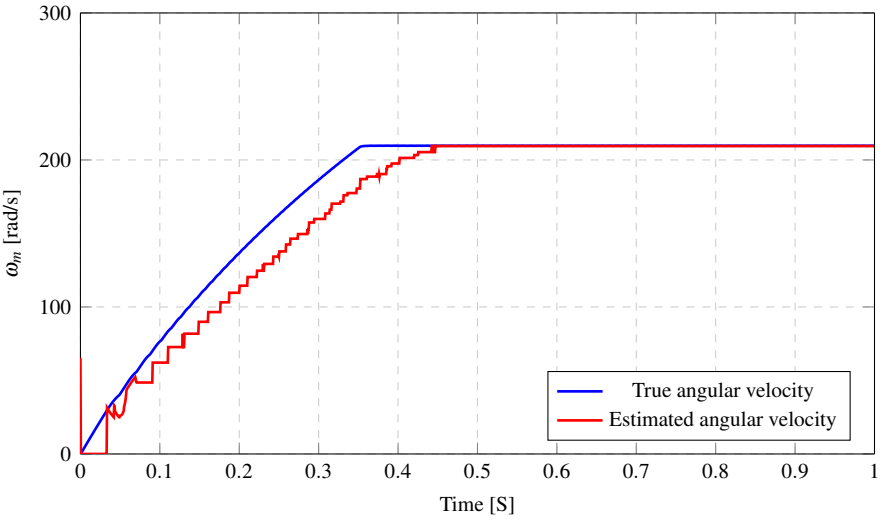

Figure 4.8 1 second step response and the estimated mechanical angular velocity from Algorithm 1. The algorithm uses $N = 3350$ to derive the mechanical angular velocity.

where again P is the number of poles in the rotor.

A first approach to the problem would be to choose two boundary pairs B_1 and B_2. Since the ADC has 12 bits, the maximum value from the ADC is 4095. For B_1, the chosen values would be $[b_0, b_1] = [0, b]$, and for B_2, $[b_0, b_1] = [b, 4095]$. From this, it is clear that b_0 in B_1, and b_1 in B_2 covers the whole range of possible values $[0, 4095]$. But by alternating between B_1 and B_2 where b is the cross-section, information about the phase curve's shape, such as peaks, can be gathered.

As in previous theoretical methods, the algorithm can be based on only one motor phase or the current floating phase. In this method, this can be an important decision for several reasons. When looking at one phase at all times, two peaks will occur each electrical revolution. When switching between phases to see the floating phase, instead, six peaks will occur each electrical revolution. This means that the estimate can be updated two or six times each electrical revolution. If all phases are used, also extra logic must be implemented to handle the switching. By looking at only one phase, this can be ignored at the cost of the lowest possible angular velocity that can be calculated. For a 16-bit timer, the turnaround time T_{ta} is equal to,

$$T_{ta} = (2^{16} - 1)\frac{1}{f_{timer}}, \tag{4.17}$$

where f_{timer} is the frequency of the timer. To be able to calculate the angular velocity

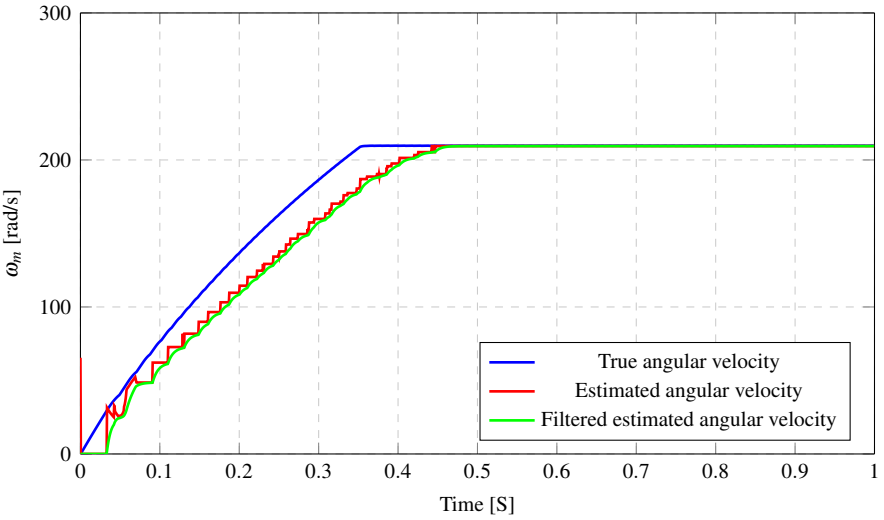

Figure 4.9 Rising phase of 1 second step response. The original estimation is delayed and not very smooth, whereas the filtered estimation is only delayed.

in the motor,

$$T_{ta} \geq T_c, \tag{4.18}$$

where T_c is the time between two commutations when looking at current floating phase. It is clear that this does not hold for all angular velocities, so T_c must be the time between two commutations at a minimum angular velocity $\omega_{m_{min}}$. When looking at only one phase, the time between two peaks T_p is equal to,

$$T_p = 3T_c, \tag{4.19}$$

which implies that $\omega_{m_{min}}$ is three times as large if $T_{ta} = T_c$.

The ADC channels on the microcontroller are separated from the processing unit. This reduces the load on the processor since the processor only acts when an interrupt is fired, implying fewer calculations per time unit.

4.4 Chosen method

There are pros and cons with all above methods. By using the magnitude of the back EMF, or calculating the ACF for the voltage time series, a lot more calculations are needed per time unit. Using interrupts from the ADC to find patterns in the phase voltage is not equally resource expensive and is therefore chosen as the most suitable method for this problem. One potential problem with this method could be that the ADC can be fooled by disturbances in the phase curve.

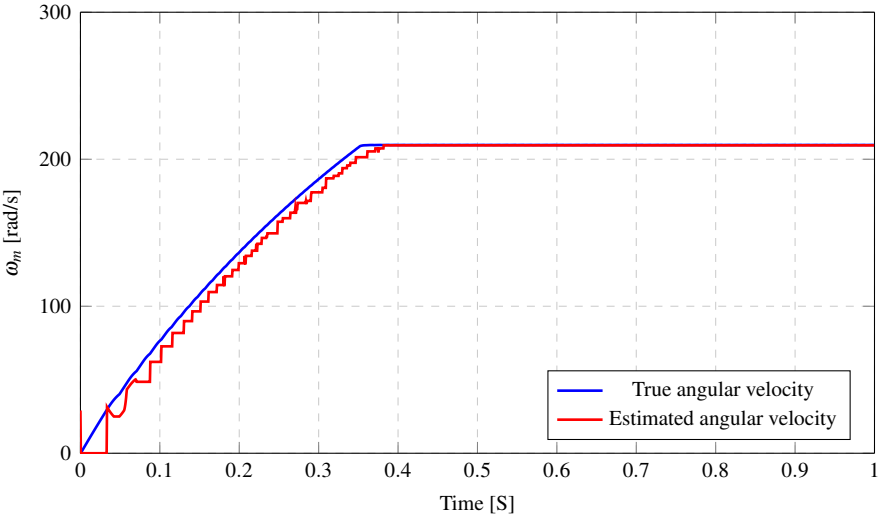

Figure 4.10 1 second step response and the estimated mechanical angular velocity. The algorithm uses $N = 838$ and is ignoring the rule of thumb to derive the mechanical angular velocity.

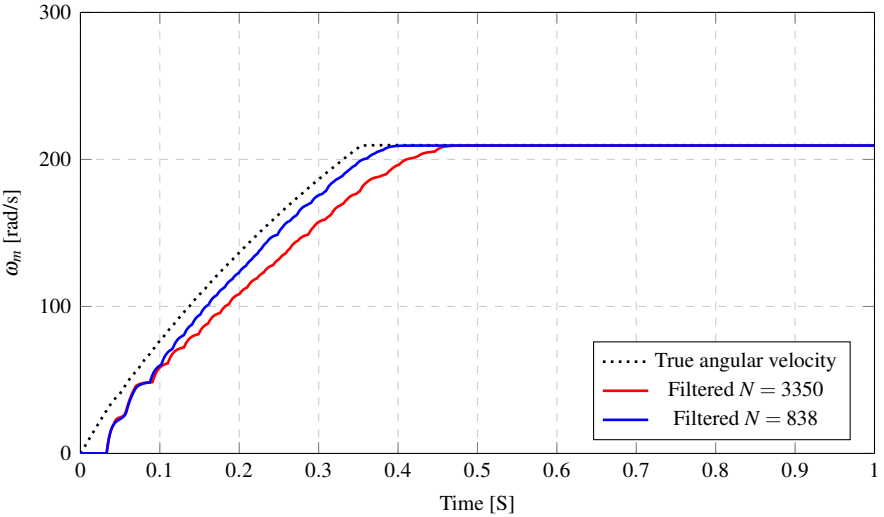

Figure 4.11 Rising phase of 1 second step response. The original filtered estimate is delayed, whereas the new filtered estimate's delay is reduced.

5

Implementation

In this section, the implementation phase of the thesis work will be described in detail. The problems that were encountered and how they were solved are also discussed.

5.1 First iteration

As mentioned in Section 4.3, a vital part of implementing an algorithm based on interrupts, is to choose when B_1 and B_2 should be used. In the first prototype of the algorithm, the idea was to detect the rising phase of a peak and then blank, meaning, ignore and not act when the phase is constantly outside the interval. After a duration of time based on the time between the two earlier peaks, the boundaries are changed from B_1 to B_2 to detect the falling phase of the next peak. The time between the interrupts from rising and falling peaks can then be converted to mechanical angular velocities through (4.16). The thinking can be seen in Figure 5.1, where the changed boundaries as well as the interrupts are visualized.
In the plot, four times, t_i, $i = 1,2,3,4$, are highlighted on the time axis. To begin with, boundary pair B_1 is used and the raw voltage value is low. At time t_1, one peak has begun to rise. The raw voltage value falls outside B_1 and an interrupt is fired. Given the time between t_1 and the previous interrupt, not visible in the plot, the blanking time is calculated to be $t_2 - t_1$. If the interrupt is the first since startup and no earlier time can be used to calculate the blanking time, a default blanking time is used. When t_2 is reached and the blanking time is over, the boundary pair is changed from B_1 to B_2. The raw voltage values are then inside the interval B_2 until the next falling peak begins. At t_3, the falling peak is detected. The blanking time is again based on the time between t_3 and t_1 and is calculated to be $t_4 - t_3$. When t_4 is reached, the boundary pair is changed back to B_1 and repeated.

The above approach was later about to be changed after viewing the phase curve from the MCU's perspective. By generating interrupts at a frequency of 20 kHz, and saving the raw values from the ADC, the true phase curve could be seen. Figure 5.2 shows one electrical revolutions at 65 rad/s.

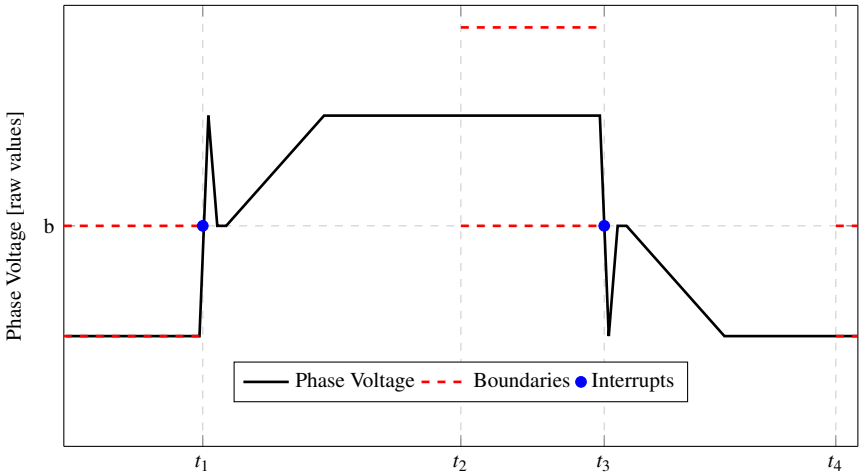

Figure 5.1 Visualization of when interrupts (blue) are fired, how the boundaries (red) are changed, and the phase voltage (black) over time. In the plot is *b* the cross-section value.

At first sight, the phase curve did not match the theoretical shape in Figure 5.1. However, after analyzing it a bit, the theoretical characteristics could be spotted, where the biggest difference between the true shape and the theoretical one was that the peaks were harder to detect during the floating phase at lower angular velocities. In Figure 5.3 and 5.4, phase curves during 150 and 350 rad/s respectively can be seen. Comparing those with Figure 5.2, it is clear that the curve's shape is changing for different angular velocities. To successfully estimate the angular velocity in the motor, the algorithm must be able to handle the changing shape, which is harder and more complicated to achieve by only using two pairs of boundaries, especially if disturbances are present, fooling the algorithm to fire interrupts wrongly.

During the implementation phase of the project, a strange behavior was discovered in the phase curve. From experiments, it could be seen that the phase curve switched between a smooth curve, and a curve with high frequently occurring spikes, as can be seen in Figure 5.5. This behavior is harder to handle, since interrupts are fired at the wrong places. However, since this is not an expected behavior, it is assumed to be present because of some fault in the test equipment specifically made to be used in this thesis, or the project specific circuit board.

5.2 Second iteration

Instead of continuing with the above approach, the changing shape of the phase voltage forced the development into a more incremental algorithm. Since the shape

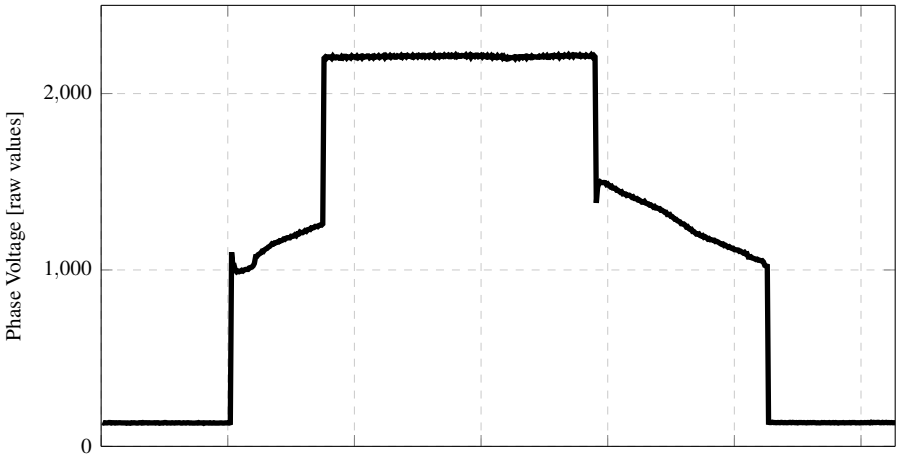

Figure 5.2 Phase voltage in raw ADC values during one revolution at an angular velocity of 65 rad/s.

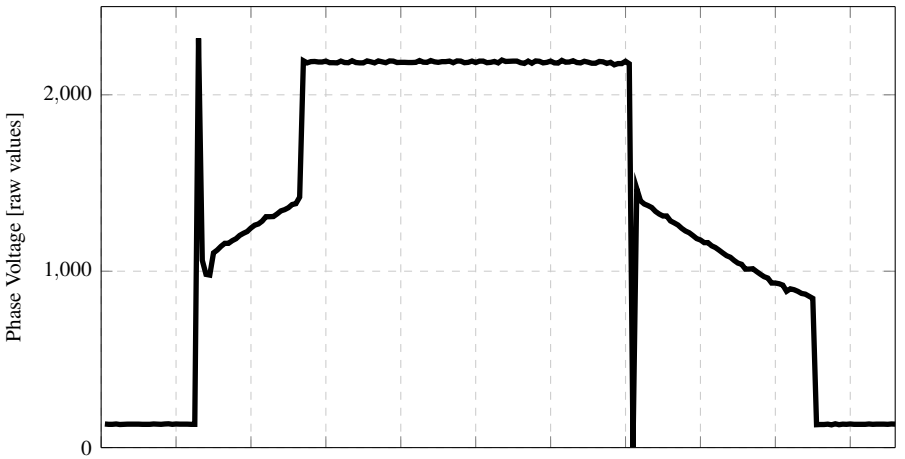

Figure 5.3 Phase voltage in raw ADC values during one revolution at an angular velocity of 150 rad/s.

is different for different angular velocities, an algorithm described as a state machine can be developed. By trying to change the boundaries more often, a step by step algorithm can keep track of where, in the voltage shape, it currently is.

To begin with, only the curve without the unwanted spikes were considered. One electrical revolution is divided into 6 states, where each state contains a char-

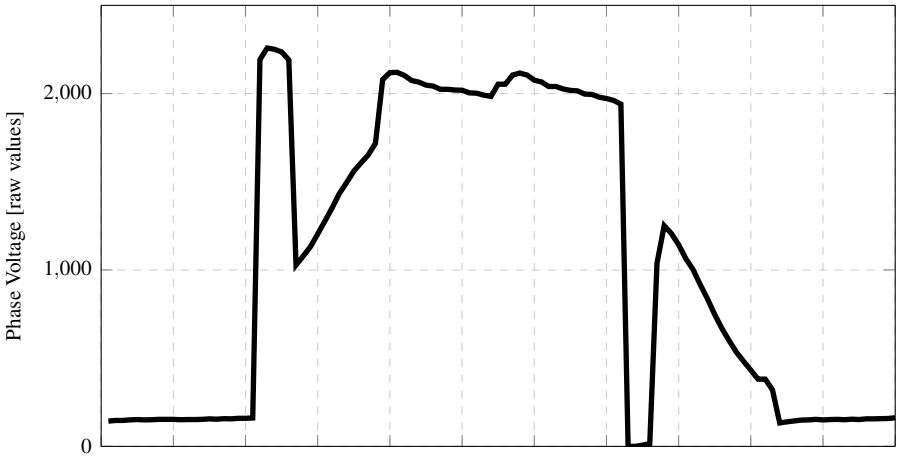

Figure 5.4 Phase voltage in raw ADC values during one revolution at an angular velocity of 350 rad/s.

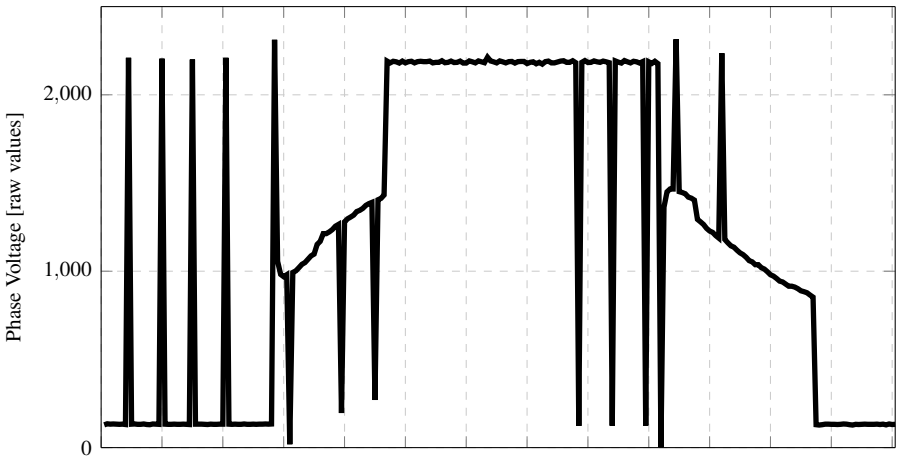

Figure 5.5 Phase voltage in raw ADC values during one revolution at an angular velocity of 150 rad/s with highly frequent occurring spikes.

acteristic feature such as high, low, local maxima, or local minima. Each state is highlighted in Figure 5.6.

Instead of firing an interrupt when the curve moves outside the interval, an interrupt is fired when the voltage falls inside the interval. From Figures 5.2, 5.3 and 5.4, the magnitudes in the different states can be compared. In state one, the bound-

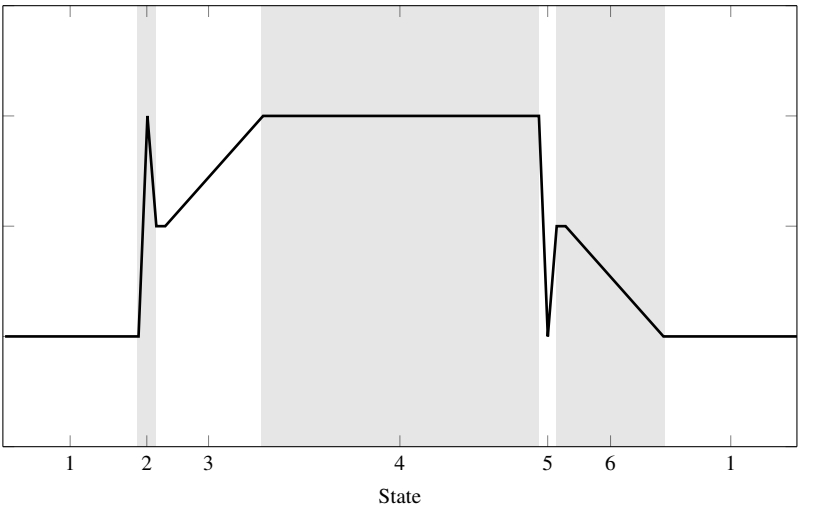

Figure 5.6 Typical phase voltage curve with the different states numbered out.

ary pair $[0, 500]$ is used to capture the beginning of the low phase. In state two, the spikes are different between the figures. Therefore, the boundary pair $[1000, 4095]$ is used to capture the peak, no matter its magnitude. To find the area around the local minima in state three, the boundary pair $[1000, 1200]$ is used. In state four, the boundary pair $[2000, 4095]$ is used to capture the beginning of the high phase. To capture the downwards spike in state five independently of its magnitude, boundary pair $[0, 1200]$ is used. In the last state, again, the boundary pair $[1000, 1200]$ is used to capture the area around the local maxima. The boundary pairs are summarized in Table 5.1.

When the phase curve is spiky, interrupts are fired at the wrong times, no matter the approach. To reduce this influence, only angular velocities below a maximum angular velocity are accepted. This will affect the overall performance negatively but will instead make the estimates more stable and usable.

Table 5.1 Boundary pairs to detect the different states.

State	b_0	b_1
1	0	500
2	1000	4095
3	1000	1200
4	2000	4095
5	0	1200
6	1000	1200

In state two and five, where the peaks are located, the current times are retrieved, corresponding to a half revolution. The tick frequency of the timer used is $f_{timer} = 260417$ Hz. This corresponds to a turnaround time of $T_{ta} \approx 0.252$ seconds, which in turn corresponds to a mechanical angular velocity of $\omega_m \approx 4.16$ rad/s, which is below the idle speed that the motor is driven at.

The velocity is calculated on the MCU using

$$\omega_m = 2\pi \frac{f_{timer}}{P n_{ticks}}, \tag{5.1}$$

where n_{ticks} is the number of ticks between interrupts. To reduce the number of calculations, a conversion constant C_{conv} is defined. This constant is calculated as

$$C_{conv} = 2\pi \frac{f_{timer}}{P} \approx 272708, \tag{5.2}$$

rounded to the closest integer, simplifying the calculations down to

$$\omega_m = \frac{C_{conv}}{n_{ticks}}. \tag{5.3}$$

Code

Below, example code in C of the implemented algorithm is presented. To calculate the time elapsed between two peaks, the `timeElapsed` function is used. It must be checked if the current time t_1 is less than the previous time t_0 to handle a turn around on the timer. If this is false, the time is calculated normally.

```
PRIVATE UINT timeElapsed(UINT t0, UINT t1, UINT max){
    if (t1 <= t0){
        return (max - t0) + t1;
    }
    return t1 - t0;
}
```

The angular velocity is calculated through the function `calculateAngularVelocity`, where a moving average filter is applied with window size n. This function has the current timestamp as argument and updates the time buffer, which has size `timeBufferSize` $= n$. Only tick values greater than what corresponds to 400 rad/s are added to the buffer to prevent corrupted interrupts to influence the estimated velocity. The mean is calculated if all values in the time buffer are none-zero. The mean is later used with C_{conv} to calculate the angular velocity.

```
PRIVATE UINT calculateAngularVelocity(UINT t1){
    UINT t = timeElapsed(t0, t1, 65535);
    UINT angularVelocity = 0U;
    UINT meanTick = 0U;
```

```
    UINT i;
    t0 = t1;
    if (timeBufferIdx == timeBufferSize){
        timeBufferIdx = 0U;
    }
    if (t >= minimumTickValue){
        timeBuffer[timeBufferIdx] = t;
        timeBufferIdx++;
    }
    for (i = 0U; i < timeBufferSize; i++){
        if (timeBuffer[i] == 0U){
            return 0U;
        }
        meanTick = add(meanTick,
            div(timeBuffer[i], timeBufferSize));
    }
    return div(C_conv, meanTick);
}
```

The function `redundantInterruptHandler` is invoked when an interrupt is fired. The boundary pair during startup is set to detect the first upwards peak, hence state two is set. In each state, the boundaries are changed. When state two and five are found, the current times are retrieved and passed with function `calculateAngularVelocity`.

```
PUBLIC void redundantInterruptHandler(void){
    if (startUp){
        state = 2;
        startUp = False;
    }

    if (state == 1){
        setBoundaries(1000, 4095);
        state = 2;
    }
    else if (state == 2){
        UINT t1 = getCurrentTime();
        setBoundaries(1000,1200);
        state = 3;
        redundantAngularVelocity =
            calculateAngularVelocity(t1);
    }
    else if (state == 3){
        setBoundaries(2000,4095);
```

```
        state = 4;
    }
    else if (state == 4){
        setBoundaries(0,1000);
        state = 5;
    }
    else if (state == 5){
        UINT t1 = getCurrentTime();
        setBoundaries(1000,1200);
        state = 6;
        redundantAngularVelocity =
            calculateAngularVelocity(t1);
    }
    else if (state == 6){
        setBoundaries(0,500);
        state = 1;
    }
    else{
        /* Do Nothing */
    }
}
```

5.3 Comparing estimates

The redundant estimate is a safety feature, running in the background at all times. Therefore, the estimate used as input to the controller, controlling the motor speed, is seen as the true angular velocity ω. The redundant estimate can be used as an additional measurement, that should not diverge from the true estimate ω more than a threshold of $\pm L$, for more than n estimates. The allowed range could be calculated as $L = p\omega$, where p is a percentage level.

6

Results

To validate the performance of the redundant estimate, a number of experiments were carried out. The experiments carried out were step responses from standstill to reference speed 65, 250 and 300 rad/s. Experiments have been done with the two window sizes n, 6 and 18, corresponding to the mean speed during one and three mechanical revolutions respectively. These plots are found in Section 6.1. For all experiments, a residual analysis was performed using the ACF, described earlier. The residual error is calculated as $e = \omega_{redundant} - \omega$. These results are found in Section 6.2.

6.1 Step responses

In Figure 6.1, 6.3, and 6.5, step responses can be seen for reference speeds 65, 250 and 300 rad/s respectively, using window size $n = 6$. In Figure 6.2, 6.4, and 6.6, step responses for the same reference speeds can be seen, using window size $n = 18$.

6.2 Residual errors

In Table 6.1, the mean residual error and the residual error variance are summarized. For all the residual errors, a Jarque-Bera hypothesis test was performed to determine if the residual errors were normally distributed. All tests resulted in that the null hypothesis can be rejected at a 5% level and that the data is not normally distributed. The ACF are computed for all residual errors and plotted for the first 200 lags, where in Figure 6.7, the window size $n = 6$ is used, and in Figure 6.8, the window size $n = 18$ is used. A 95% confidence interval for significance is also shown.

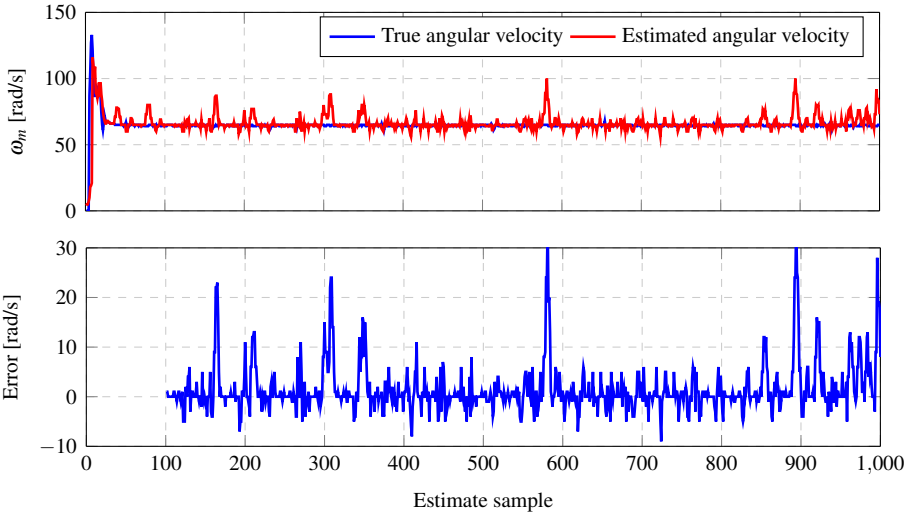

Figure 6.1 Step response from standstill to $r = 65$ rad/s, using $n = 6$ (upper subplot). The residual error is plotted with the rising phase of the step response removed (lower subplot).

Table 6.1 Mean residual error and residual error variance.

r	n	mean	variance
65	6	1.54	24.61
65	18	1.14	5.1
250	6	-1.95	126.16
250	18	-2.78	58.13
300	6	-5.54	291.81
300	18	-5.64	106.82

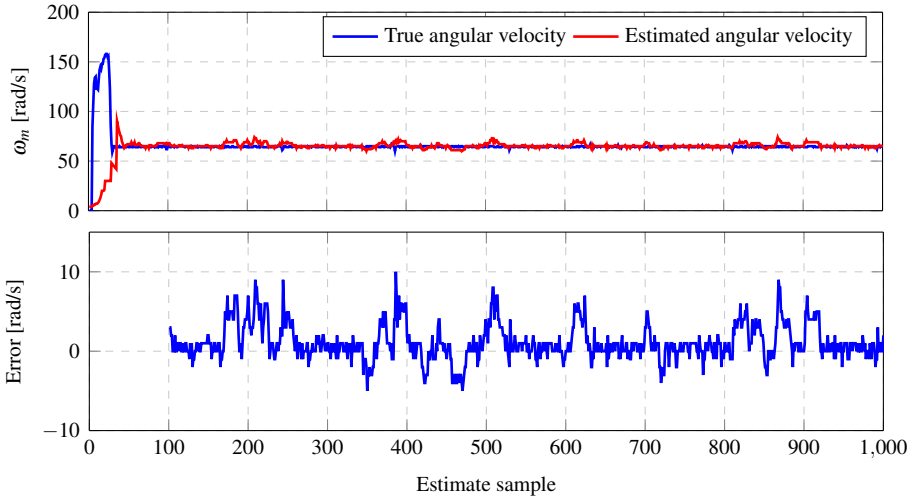

Figure 6.2 Step response from standstill to $r = 65$ rad/s, using $n = 18$ (upper subplot). The residual error is plotted with the rising phase of the step response removed (lower subplot).

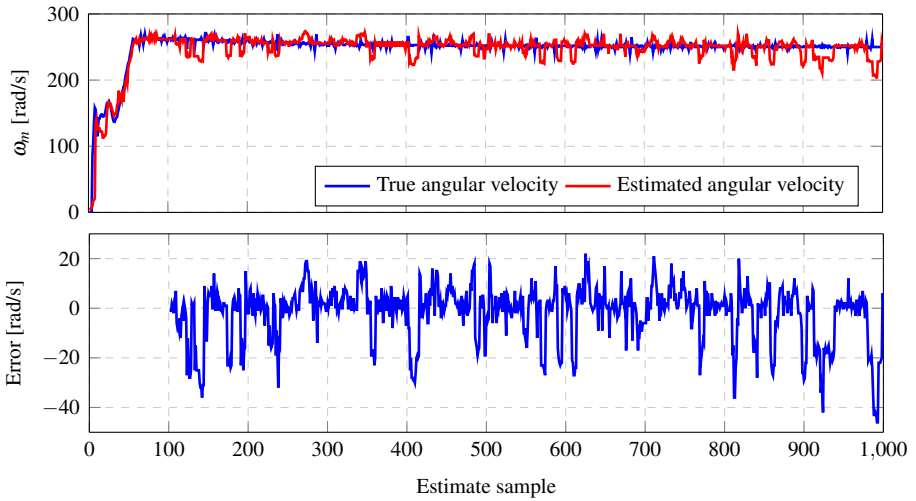

Figure 6.3 Step response from standstill to $r = 250$ rad/s, using $n = 6$ (upper subplot). The residual error is plotted with the rising phase of the step response removed (lower subplot).

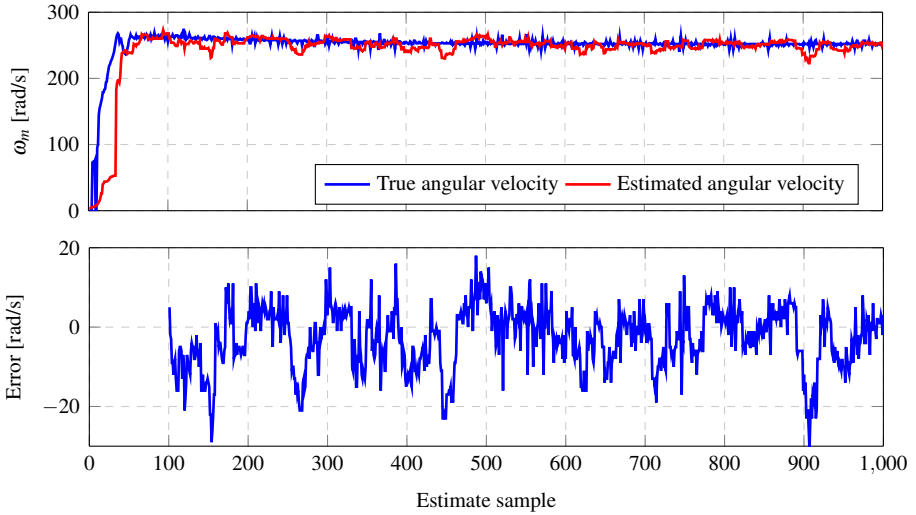

Figure 6.4 Step response from standstill to $r = 250$ rad/s, using $n = 18$ (upper subplot). The residual error is plotted with the rising phase of the step response removed (lower subplot).

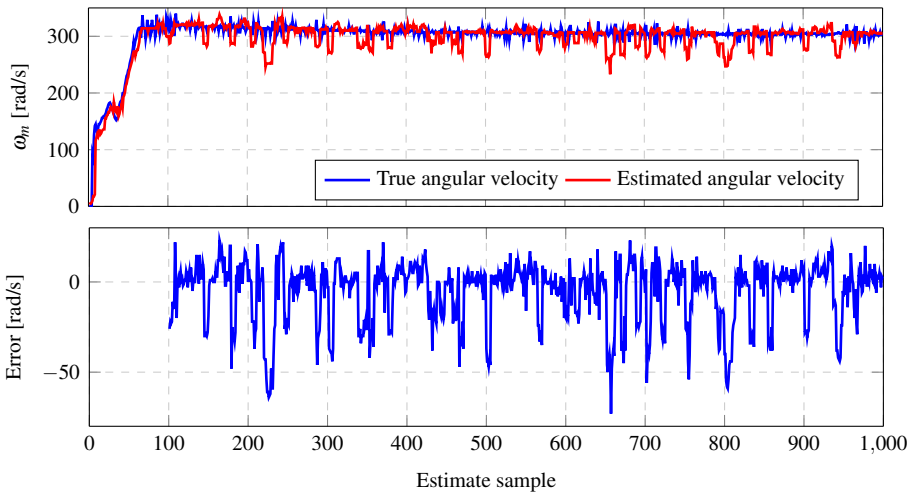

Figure 6.5 Step response from standstill to $r = 300$ rad/s, using $n = 6$ (upper subplot). The residual error is plotted with the rising phase of the step response removed (lower subplot).

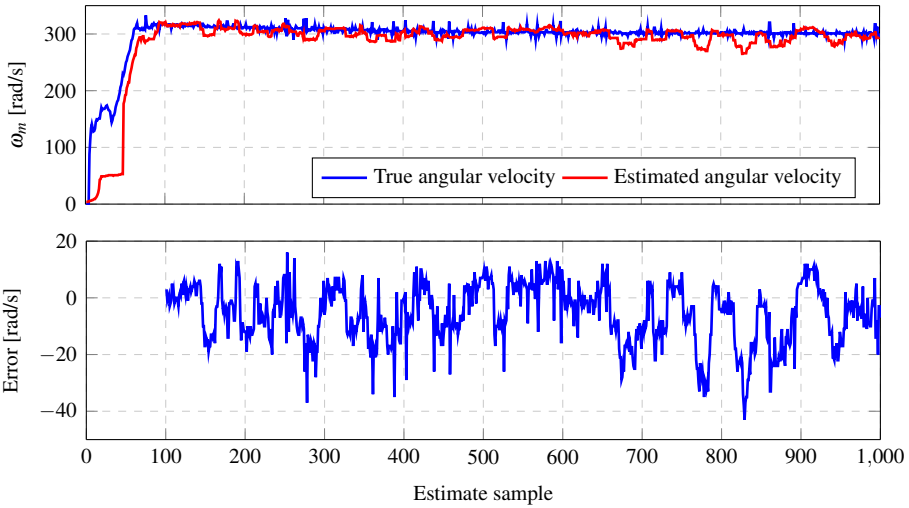

Figure 6.6 Step response from standstill to $r = 300$ rad/s, using $n = 18$ (upper subplot). The residual error is plotted with the rising phase of the step response removed (lower subplot).

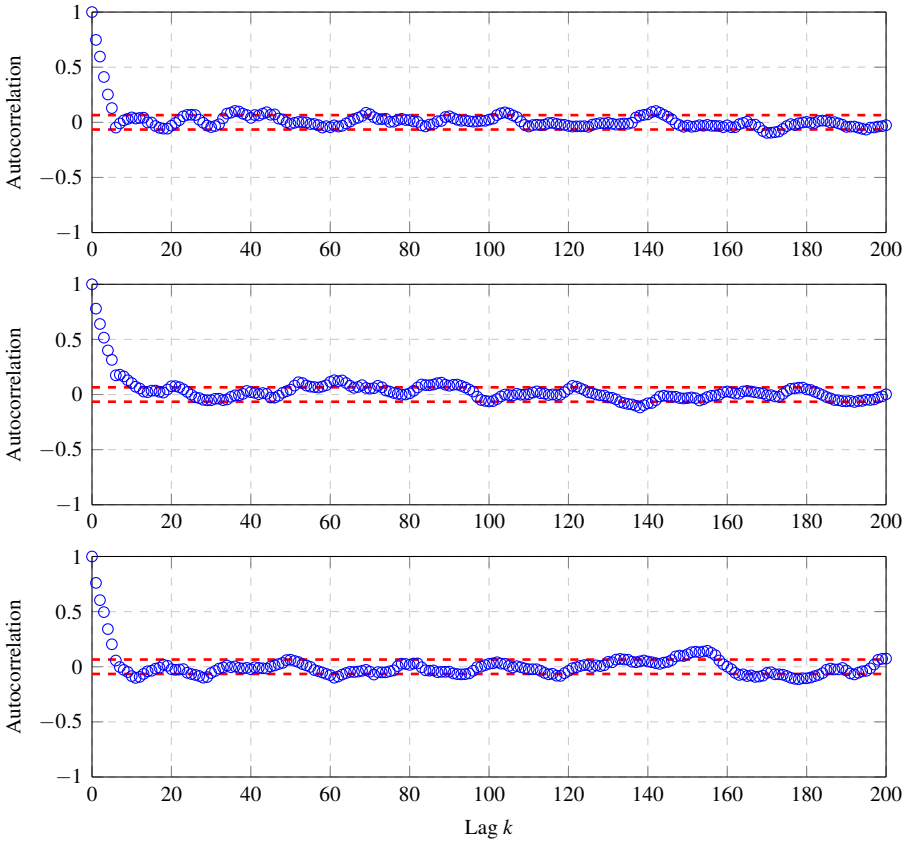

Figure 6.7 ACFs for residual errors with $n = 6$. The first 200 lags are plotted. Red dotted lines represent the 95% significance confidence interval.

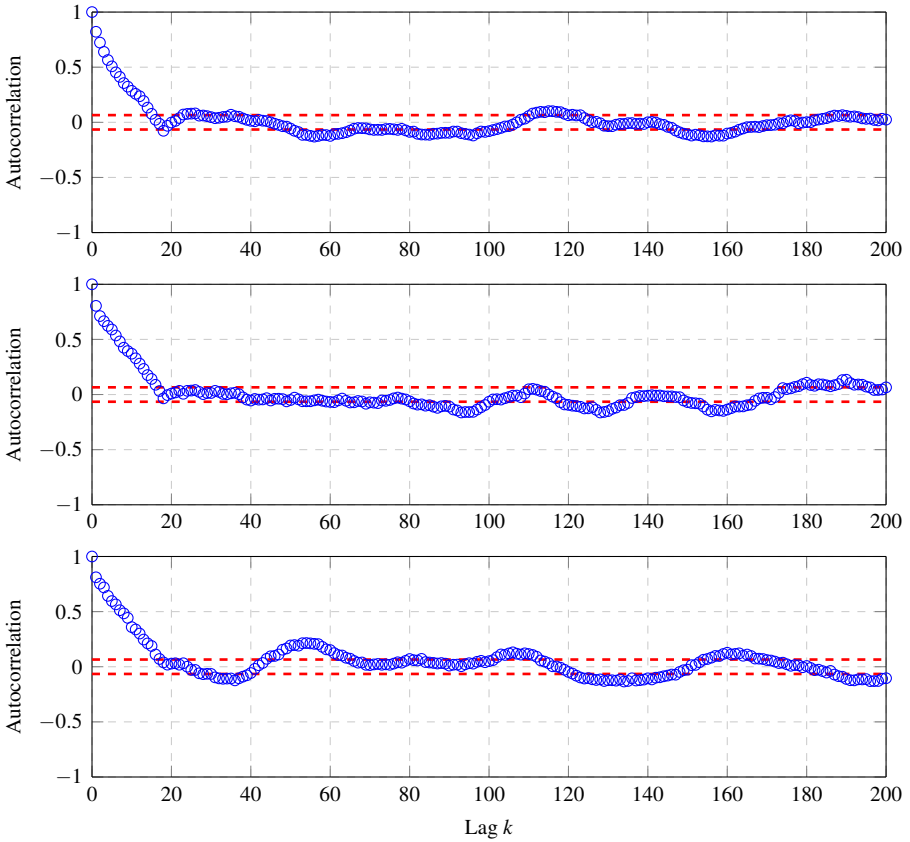

Figure 6.8 ACFs for residual errors with $n = 18$. The first 200 lags are plotted. Red dotted lines represent the 95% significance confidence interval.

7

Discussion

In this section, the results obtained from the implemented method are discussed. The strengths, weaknesses and overall performance are discussed, but also the problems that arose during implementation. Lastly, some conclusions about the extra CPU load are mentioned.

7.1 Performance

From the figures in Section 6.1, the performance differs between the window sizes. In Table 6.1, the variance is considerably greater for the window size $n = 6$. The mean error is greater for $n = 6$ at idle speed, and lower for greater angular velocities, but not dramatically.

Even though the larger window size performs better during steady state conditions, the variance is greater than desired. The variance being that high probably comes down to the influence of disturbed ADC values. The algorithm tracks the shape of the voltage curve, where each section has different lengths. The length of each section also changes with the angular velocity. The time between two correct interrupts are for greater angular velocities, less than the time between a correct and a corrupted one, for smaller angular velocities. This hinders the solution of checking if the time between interrupts are long enough. If the other estimate is used to calculate a window where no interrupts are expected, a gray area emerges around the definition of the term redundant.

If a corrupt interrupt is fired too early, this can result in that the time between, what the algorithm thinks is two peaks, is less than what corresponds to the maximum angular velocity. If the motor, for example, runs at idle speed, this time could influence the estimate a lot. When this occurs, the algorithm also gets lost. After a while, eventually, it will find its way back, but then the measured time takes longer than expected, decreasing the estimate. In Table 6.1, the mean error is positive for idle speed, and negative for greater angular velocities. This supports the reasoning above, since for smaller angular velocities, some corrupt and short times are likelier

to be accepted, and for greater angular velocities, too short times are ignored, and only too long times are accepted.

Overall, a longer window size performs considerably better during steady state, but comes with a longer delay when the reference speed is changed. This is clear from the plots in Section 6.1, where the estimate tracks the angular velocity better for $n = 6$.

The algorithm's weaknesses are that it is sensitive to unexpected disturbances on the ADC channel, and that it is more delayed than the current sensorless estimate. The unexpected disturbances on the ADC channel should be fixable, but the delay is harder to disregard. With two measurements per electric revolution, smaller window sizes than $n = 6$ might make the estimate unstable even with undisturbed ADC measurements.

The strengths coming with the proposed algorithm are that less logic is needed, the algorithm never fully stops working, but also that it can keep track of the angular velocity, when the motor stops commuting. As described above, the algorithm can get lost, but because of the shape of the voltage curve and the changed limits, the algorithm eventually, always finds itself in the right state. When the motor stop commuting, and current stops flowing, but still spins, all three phases will have trapezoidal back EMF. The trapezoidal waveform is very similar to the waveform of the voltage during idle speed. This makes it possible to track the angular speed when the motor slows down, without any modifications.

Using the smaller window size of $n = 6$ is preferred to minimize the delay. The smaller window size should also be enough to get a stable estimate when the unexpected disturbances on the ADC channel is resolved. However, for now, a larger window size is needed to remove the worst spikes in the estimate.

7.2 Residual analysis

Using the ACF, the correlation between samples e_k is calculated. In Figure 6.7, where the window size $n = 6$ is used, there are strong dependencies for lags $1 - 5$, which is expected because of the window size of the moving average filter. The same phenomenon is visible in Figure 6.8, where the lags $1 - 17$ are significant, because of the larger window size. When two ACF plots, with the same reference speed, are compared, no seasonality is standing out. Some lags are outside the confidence interval and are significant, but, since the data is not normally distributed, the confidence interval can not be fully trusted. If the redundant ADC channel was configured wrongly, or if the syncing is not working as expected, this should appear in the ACF. This points in a direction where the high frequent disturbance comes somewhat randomly, which logically would suggest that something could be wrong with the test setup, rather than the configuration of the ADC channels.

7.3 Comparing estimates

With a percentage p of five percent, and letting the estimate exceed the threshold for $n = 100$ times, all runs in the Section 6.1 would comply with that requirement. With an angular velocity of 400 rad/s, $n = 100$ would correspond to approximately 0.26 seconds. During idle speed, it would correspond to approximately 1.61 seconds.

7.4 CPU load

Without the logic of changing what phase to track, and four more peaks per electrical revolution, the algorithm successfully tracks the angular velocity with very low extra load on the CPU. With 4 more peaks per electrical revolution, and logic for phase switching, the operations per electrical revolution is more than three doubled. The other two methods described in Section 4 requires more operations per sample period, which would have impacted the CPU load more. With the proposed solution, fewer calculations are made overall per time unit. Also, fewer calculations are needed for smaller angular velocities, since the number of calculations are decoupled from the sampling frequency.

8

Conclusion

The end goal was to implement and evaluate a redundant algorithm that successfully could track the angular velocity. The proposed solution tracks the angular velocity, but because of unexpected disturbances on the ADC channels, spikes in the estimate are a fact. Two window sizes on the moving average filter applied to the estimate are tested, where the smaller is preferred to get rid of delay. However, the larger window size is needed to lower the impact of spikes in the final estimate with the current problems. An important aspect was also to try to minimize the extra CPU load, which the proposed solution does. By focusing on tracking the shape of only one phase curve, the operations per electrical revolution are decreased with more than a third. To detect malfunction, the two estimates should be compared, where the difference should not exceed a set percentage for more than a set number of samples.

8.1 Future work

In the proposed solution, the algorithm is sensitive to disturbances. To increase the performance of the algorithm, this would be important to address. Also, from Section 4, finding the period using auto correlation seems to have potential. The algorithm appears, from simulations, to be able to perfectly track the angular velocity. Currently, it requires a lot of resources to be able to calculate the ACF inside the sampling period. In the future, it would be interesting to investigate how much the sampling frequency could be reduced to still be able to find the periodicity. The algorithm could also be modified to sample at the same frequency, but calculate the ACF less often to reduce the CPU load.

Bibliography

Breitenbach, T., B. Wilkusz, L. Rasbach, and P. Jahnke (2023). “On a method for detecting periods and repeating patterns in time series data with autocorrelation and function approximation.” *Pattern Recognition* **138**. ISSN: 0031-3203.

Hughes, A. (2019). *"Electric Motors and Drives : Fundamentals, Types and Applications" [Elektronisk resurs]*. ISBN: 0081026153.

Introduction and Implementations of the Kalman Filter. [Elektronisk resurs]. (2019). InTech. ISBN: 9781838805364.

Jakobsson, A. (2013). *An Introduction to Time Series Modeling*. Fourth edition. Studentliteratur AB, pp. 39–40, 47. ISBN: 978-91-44-15894-5.

Naveen, V. and T. B. Isha (2017). “A low cost speed estimation technique for closed loop control of bldc motor drive.” *2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT), Circuit ,Power and Computing Technologies (ICCPCT), 2017 International Conference on*, pp. 1–5. ISSN: 978-1-5090-4967-7.

Shrutika, C., S. Matani, S. Chaudhuri, A. Gupta, S. Gupta, and N. Singh (2021). “Back-emf estimation based sensorless control of brushless dc motor.” *2021 1st International Conference on Power Electronics and Energy (ICPEE), Power Electronics and Energy (ICPEE), 2021 1st International Conference on*, pp. 1–6. ISSN: 978-1-7281-8774-7.

Wang, S. and A.-C. Lee (2015). “A 12-step sensorless drive for brushless dc motors based on back-emf differences.” *IEEE TRANSACTIONS ON ENERGY CONVERSION* **30**:2, pp. 646–654. ISSN: 08858969.

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

Document name
MASTER'S THESIS

Date of issue
June 2024

Document Number
TFRT-6241

Author(s)
Victor Martinsson

Supervisor
Jacob Svendenius, BorgWarner Sweden AB
Bertil Ströberg, BorgWarner Sweden AB
Karl-Erik Årzén, Dept. of Automatic Control, Lund University, Sweden
Kristian Soltesz, Dept. of Automatic Control, Lund University, Sweden (examiner)

Title and subtitle
Angular Velocity Estimation for Sensorless Brushless DC Motors

Abstract

In the automotive industry today, safety is of highest priority. ASIL is a safety standard that quantifies risk. This project investigates how a redundant speed estimate in a sensorless BLDC motor can be developed to meet ASIL B requirements. The proposed solution involves tracking the shape of the phase curve to determine the angular velocity. This is accomplished by changing the boundaries on the ADC, making it work as a comparator, to find limit events.

The results show that the algorithm successfully tracks the angular velocity. The error residual is noisy, with a maximum mean error of −5.64 rad/s. The error is believed to be due to unexpected disturbances on the phase curve. To get the final estimate, a moving average filter with window size n = 6 and n = 18 is applied, corresponding to one and three mechanical revolutions.Window size n=6 is preferred to reduce delay, but n = 18 is more stable with the present disturbances.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
1-55

Recipient's notes

Security classification