Examensarbete

Sensor-less Control of a Permanent Magnet Synchronous Motor

Examensarbete utfört i Reglerteknik vid Tekniska högskolan i Linköping
av

Fredrik Petersson

LITH-ISY-EX--09/4186--SE
Linköping 2009
Sensor-less Control of a Permanent Magnet Synchronous Motor

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan i Linköping
av

Fredrik Petersson

LITH-ISY-EX--09/4186--SE

Handledare: Rikard Falkeborn
ISY, Linköpings universitet
Martin Larsén
SAAB Avitronics, Jönköping

Examinator: Johan Löfberg
ISY, Linköpings universitet

Linköping, 31 January, 2009
# Title
Sensor-less Control of a Permanent Magnet Synchronous Motor

# Abstract
A permanent magnet synchronous motor is traditionally controlled from measured values of the angular velocity and position of the rotor. However, there is a wish from SAAB Avitronics to investigate the possibility of estimating this angular velocity and position from the current measurements. The rotating rotor will affect the currents in the motor’s stator depending on the rotor’s angular velocity, and the observer estimates the angular velocity and angular position from this effect.

There are several methods proposed in the article database IEEE Xplore to observe this angular velocity and angular position. The methods of observation chosen for study in this thesis are the extended Kalman filter and a phase locked loop algorithm based on the back electro motive force augmented by an injection method at low velocities.

The extended Kalman filter was also programmed to be run on a digital signal processor in SAAB Avitronics’ developing hardware. The extended Kalman filter performs well in simulations and shows promise in hardware implementation. The algorithm for hardware implementation suffers from poor resolution in calculations involving the covariance matrices of the Kalman filter due to the use of 16-bit integers, yielding an observer that only functions in certain conditions.

As simulations with 32-bit integer algorithm performs well it is likely that a 32-bit implementation of the extended Kalman filter would perform well on a motor, making sensorless control possible in a wide range of operations.

# Keywords
Extended Kalman Filter, Non-linear Observer, Permanent Magnet Synchronous Motor
Abstract

A permanent magnet synchronous motor is traditionally controlled from measured values of the angular velocity and position of the rotor. However, there is a wish from SAAB Avitronics to investigate the possibility of estimating this angular velocity and position from the current measurements. The rotating rotor will affect the currents in the motor’s stator depending on the rotor’s angular velocity, and the observer estimates the angular velocity and angular position from this effect.

There are several methods proposed in the article database IEEE Xplore to observe this angular velocity and angular position. The methods of observation chosen for study in this thesis are the extended Kalman filter and a phase locked loop algorithm based on the back electro motive force augmented by an injection method at low velocities.

The extended Kalman filter was also programmed to be run on a digital signal processor in SAAB Avitronics’ developing hardware. The extended Kalman filter performs well in simulations and shows promise in hardware implementation. The algorithm for hardware implementation suffers from poor resolution in calculations involving the covariance matrices of the Kalman filter due to the use of 16-bit integers, yielding an observer that only functions in certain conditions.

As simulations with 32-bit integer algorithm performs well it is likely that a 32-bit implementation of the extended Kalman filter would perform well on a motor, making sensorless control possible in a wide range of operations.
Acknowledgments

I would like to thank SAAB Avitronics for having a genuine interest in the field of sensorless control and giving me the opportunity to perform this master thesis. A special thank you to Martin Larsén for being my supervisor and for teaching me many things about permanent magnet synchronous motors and guiding me through all your application code and hardware. Anders Gustafsson, for being an understanding manager and helping me with all technicalities and welcoming me to SAAB Avitronics. Rikard Falkeborn, for being my supervisor at ISY, you have helped me with a lot of questions and for proof reading my thesis and ideas many times. Johan Löfberg, for being my examiner, helping me with formalities and for showing a big interest in my thesis. Maria Malm, for letting me rent her apartment while spending 6 months in Perth, making living in Jönköping possible. At last but not least thank you Birgitta for your support, love and warm meals every time I come home to Linköping.

Jönköping, December 2008
Fredrik Petersson
Contents

1 Introduction .......................... 1
  1.1 Background ............................................. 1
  1.2 Aim and Purpose ...................................... 2
  1.3 Method ................................................ 2
  1.4 Limitations ......................................... 2
  1.5 Related Work ....................................... 2
  1.6 Outline .............................................. 3

2 System Overview ...................... 5
  2.1 Frames ................................................. 6
    2.1.1 Transformation Equations ........................ 7
    2.1.2 Inverse Frame Transformations ................... 8
  2.2 Controller ............................................ 8
  2.3 Duty-Cycle Calculation, Inverter Bridge ............... 9
  2.4 Resolver ............................................... 10

3 Motor Model .......................... 11
  3.1 Motor equations in the $\delta$-$\gamma$ frame ............ 13
  3.2 Motor used in this Thesis ............................ 14
  3.3 Simulink Model ....................................... 14

4 Estimators ............................ 17
  4.1 Error Dynamics ................................. 17
  4.2 Extended Kalman Filter ............................ 19
    4.2.1 Theory ............................................. 20
    4.2.2 Algorithm ......................................... 21
    4.2.3 Filter Tuning ..................................... 22
    4.2.4 Convergence ...................................... 22
    4.2.5 Simulink Model .................................. 24
  4.3 Phase Locked Loop ............................. 24
    4.3.1 Estimating Position Error ..................... 24
    4.3.2 Algorithm and Observer Gains ................. 25
    4.3.3 Convergence ...................................... 26
    4.3.4 Simulink Model .................................. 28
  4.4 Low Frequency Injection .......................... 29
Chapter 1

Introduction

1.1 Background

SAAB Avitronics currently use Permanent Magnet Synchronous Motors (PMSM) in their product that is fitted into the high lift control system of the Boeing 787 aircraft. The motors are used to operate the wing flaps and slats of the aircraft. Traditionally these have been driven by hydraulics, but in the later years the interest for electric systems has risen.

A PMSM is a form of electrical motor, it operates by having permanent magnets in its rotor and by applying alternating currents in the stator synchronously with the angular position and angular velocity of the rotor.

Due to their design, PMSMs have a lot of advantages compared to other electrical motors. Here follows a list of positive characteristics which will be explained more in detail later in this report:

- Current is only applied in the stator, so cooling of the motor is simple.
- Magnetic field in the rotor is caused by permanent magnets so few moving parts are required.
- Generated torque is high for all speeds, eliminating the need for gearboxes.
- The PMSM has a high power density.

It is of vital importance to know the position and angular velocity to control the applied voltages in the stator correctly, otherwise the motor cannot be operated as intended. To know these quantities, a sensor on the drive shaft is used. The sensor can operate by several methods, such as measuring Hall effect or by optical encoding. These sensors use valuable space as well as being expensive. A wish to operate the PMSM without this sensor exists and this thesis was proposed.

There are certain requirements on an observer to be useful for sensorless control of PMSMs for avionic applications. The main requirement is that the observer estimates the angular position and the angular velocity of the rotor correctly under all possible working conditions. The requirement can be divided into a few demands:
Introduction

- The error in estimation must be small in stationary operations.
- The observer must converge toward the true rotor position from any unknown starting position, this is also referred to as global convergence in the thesis.
- The observer must converge within a small time period for any steps in reference velocity or the application of an unknown external load.

1.2 Aim and Purpose

The purpose of this thesis is to be a preliminary study of the possibility and suitability for sensorless control of PMSMs in SAAB Avitronics' current and future products. The aim is to find a working solution to obtain sensorless control of the PMSM used by SAAB Avitronics in their current products in both simulations and in a test bench.

1.3 Method

The approach to solve the problem of sensorless control of a PMSM is to first gather information on already proposed solutions to this problem in published academic articles in a literature study. Two promising methods of observation will be chosen for implementation in Matlab/Simulink so their performance, robustness, convergence and complexity can be evaluated. The method yielding the best results will then proceed to be coded in C and implemented on a DSP and validated in a test bench with SAAB Avitronics’ development hardware.

1.4 Limitations

Due to time constraints, only two methods of sensorless control will be evaluated and the motor control algorithm designed by SAAB will be used. Since the thesis uses ideas and software which is property of SAAB Avitronics, most notably the motor control algorithm, those areas will not be covered in this report.

1.5 Related Work

Performing a search for “observer <and> PMSM” in the article database IEEE Xplore yields 268 articles, all published within the last 10 years. This is a clear indication on that the subject of sensorless control of electric motors and PMSM especially has been of interest in the academic world during the last decade. A lot of these articles were not read during the literature study, but in those studied there has been several proposed methods to solve the problem that is in the scope of this thesis. Just to mention a few:

- Extended Kalman filter. [1–5]
1.6 Outline

- Phase locked loop. [6,7]
- Low frequency injection. [8]
- High frequency injection. [9,10]
- Sliding mode observer. [11,12]
- Reduced-order linear Kalman filter. [13]
- Adaptive flux observer. [14]
- Model reference adapting system. [15,16]

It is also worth noting that the methods of observation between internal mounted and surface mounted PMSM differs, this matter will be explained further in Chapter 3.

1.6 Outline

Chapter 2 will give an overview of the system as well as describing the frames where calculations take place and the transformations between them. The chapter will also describe the components surrounding the motor and observer. Chapter 3 will describe the general theory of PMSMs and also present the mathematical model of the PMSM that is used to evaluate observers in Matlab. Chapter 4 will describe error dynamics and convergence, describe the chosen observers with derivations, algorithms and Simulink models. Chapter 5 will discuss the aspects of implementation of algorithms on hardware. Chapter 6 will cover results of simulation of algorithms in Simulink and tests of algorithms in hardware. Chapter 7 will outline proposals for future work in this field and discuss the results of this thesis.
Chapter 2

System Overview

To give the reader a better understanding of the observers a brief explanation about the system where the observer operates in, illustrated in Figure 2.1, will be given. The complete system is built of a few components as introduced in Table 2.1. Later in this chapter the controller, space vector modulation (SVM), inverter bridge and the resolver will be described. The PMSM will be explained in Chapter 3 and the methods of estimation will be described in Chapter 4. A description of the frames used for calculations will be presented first in this chapter, as they are an important part of calculations in this thesis.

Figure 2.1. An overview of the system.
System Overview

Table 2.1. Components used in the system.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>Contains the controllers used for controlling speed and currents by the use of an output voltage. Also includes the observer of choice.</td>
</tr>
<tr>
<td>SVM and inverter bridge</td>
<td>Device used to convert the controller output voltages into the voltages used to drive the PMSM through vector control.</td>
</tr>
<tr>
<td>PMSM</td>
<td>The real motor / the motor model implemented in Simulink including resolver to measure angular position and angular velocity.</td>
</tr>
</tbody>
</table>

2.1 Frames

Different frames are used in this thesis to make calculations easier. There are four frames used in this thesis, Table 2.2 describes the use of each frame and where they are used and they are illustrated in Figure 2.2. The use of each frame may seem abstract as of now, but they will be explained in the following chapters.

Table 2.2. The used frames.

<table>
<thead>
<tr>
<th>Frame</th>
<th>Description and their use.</th>
</tr>
</thead>
<tbody>
<tr>
<td>a - b - c</td>
<td>The frame of the stator windings. Voltages are applied and currents are measured in this frame.</td>
</tr>
<tr>
<td>α - β</td>
<td>Stator static frame. Frame used for actuating voltages to the PMSM in the PWM duty-cycle calculations and inverter bridge.</td>
</tr>
<tr>
<td>d - q</td>
<td>Static frame in the estimated rotor position. Motor equations are calculated in this frame, the observer and controller also operates in this frame.</td>
</tr>
<tr>
<td>δ - γ</td>
<td>Static frame in the true rotor position. Frame used when discussing estimation errors in the observers.</td>
</tr>
</tbody>
</table>
2.1 Frames

2.1.1 Transformation Equations

To travel between frames so called transformation equations are used, depending on the relation between the bases these transformations are either projection- or rotation transformations. To go from a-b-c to \( \alpha-\beta \) base we apply a projection transformation. The a-b-c frame represent the axes of the 3-phase current, so they are separated by an angle of \( 2\pi/3 \) in a 2D plane, and the \( \alpha-\beta \) is the base of this plane. Due to this projection transformation is overdetermined, a normalization factor of 2/3 is added.

\[
\begin{bmatrix}
\alpha \\
\beta
\end{bmatrix} = \frac{2}{3} \begin{bmatrix} 1 & \cos(\frac{2\pi}{3}) & \cos(-\frac{2\pi}{3}) \\ 0 & \sin(\frac{2\pi}{3}) & \sin(-\frac{2\pi}{3}) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}
\]

(2.1)

To travel from \( \alpha-\beta \) to d-q we apply a rotational operator

\[
\begin{bmatrix} d \\ q \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}
\]

(2.2)

Combining the projection operator with the rotational operator we obtain the transformation from a-b-c frame to d-q frame

\[
\begin{bmatrix} d \\ q \end{bmatrix} = \frac{2}{3} \begin{bmatrix} \cos(\theta) & \cos(\theta - \frac{2\pi}{3}) & \cos(\theta + \frac{2\pi}{3}) \\ -\sin(\theta) & -\sin(\theta - \frac{2\pi}{3}) & -\sin(\theta + \frac{2\pi}{3}) \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}
\]

(2.3)

When discussing estimation error an error \( \vartheta \) between the real rotor position and the estimated rotor position is assumed. The error in estimated position yields a rotational operator to travel from d-q frame to \( \delta-\gamma \) frame

\[
\vartheta = \theta - \hat{\theta}
\]

(2.4)

\[
\begin{bmatrix} \delta \\ \gamma \end{bmatrix} = \begin{bmatrix} \cos(\vartheta) & \sin(\vartheta) \\ -\sin(\vartheta) & \cos(\vartheta) \end{bmatrix} \begin{bmatrix} d \\ q \end{bmatrix}
\]

(2.5)
2.1.2 Inverse Frame Transformations

To obtain the inverse transformation from $\alpha-\beta$ frame to a-b-c frame a Moore-Penrose pseudo inverse\(^1\) of (3.1) is used. Worth noting is that the pseudo inverse is underdetermined, so the result of the transformation is a set of solutions. Due to the fact that all solutions will yield the same result when used in later calculations, this is not of concern. This transformation is only used in the Simulink model, in the hardware this is handled in the PWM duty-cycle calculations and the inverter bridge.

\[
\begin{bmatrix}
 a \\
 b \\
 c
\end{bmatrix} =
\begin{bmatrix}
 1 & 0 \\
 \cos(\frac{2\pi}{3}) & \sin(\frac{2\pi}{3}) \\
 \cos(\frac{-2\pi}{3}) & \sin(\frac{-2\pi}{3})
\end{bmatrix}
\begin{bmatrix}
 \alpha \\
 \beta
\end{bmatrix}
\]

(2.6)

The inverse transformation of (2.2) is calculated analytically by the 2x2 matrix special case matrix inversion.\(^2\)

\[
\begin{bmatrix}
 \alpha \\
 \beta
\end{bmatrix} =
\begin{bmatrix}
 \cos(\theta) & -\sin(\theta) \\
 \sin(\theta) & \cos(\theta)
\end{bmatrix}
\begin{bmatrix}
 d \\
 q
\end{bmatrix}
\]

(2.7)

These two transformations combined yields the following transformation

\[
\begin{bmatrix}
 a \\
 b \\
 c
\end{bmatrix} =
\begin{bmatrix}
 \cos(\theta) & -\sin(\theta) \\
 \cos(\theta - \frac{2\pi}{3}) & -\sin(\theta - \frac{2\pi}{3}) \\
 \cos(\theta + \frac{2\pi}{3}) & -\sin(\theta + \frac{2\pi}{3})
\end{bmatrix}
\begin{bmatrix}
 d \\
 q
\end{bmatrix}
\]

(2.8)

By using the same definition of 2x2 matrix inversion, we also have the following equation from (2.5)

\[
\begin{bmatrix}
 d \\
 q
\end{bmatrix} =
\begin{bmatrix}
 \cos(\dot{\vartheta}) & -\sin(\dot{\vartheta}) \\
 \sin(\dot{\vartheta}) & \cos(\dot{\vartheta})
\end{bmatrix}
\begin{bmatrix}
 \delta \\
 \gamma
\end{bmatrix}
\]

(2.9)

2.2 Controller

The controller used for the work in this thesis is property of SAAB Avitronics and will not be described fully. However, a brief description will be given to give the readers a feel for how it affects the system.

First a torque request is controlled from the difference between the reference value and the measured value of the angular velocity.

This torque request is calculated into a current reference in the q-axis by the use of an inverted formula from the PMSM equations presented in Chapter 3, the current reference in d-axis is set to zero.

The voltages that is used for actuation is then controlled from the difference in reference currents and measured currents augmented by a feed forward of current dynamics. The Simulink model of this control algorithm is illustrated in Figure 2.4.

\(^1\)A Moore-Penrose candidate $A^\dagger$ is valid if it satisfies the following conditions: 1. $AA^\dagger A = A$

2. $A^\dagger AA^\dagger = A^\dagger$, 3. $(AA)^T = AA^\dagger$, 4. $(A^\dagger A)^T = A^\dagger A$

\(^2\)\(A = \begin{bmatrix}
 a_{1,1} & a_{1,2} \\
 a_{2,1} & a_{2,2}
\end{bmatrix}
\) \(A^{-1} = \begin{bmatrix}
 \frac{1}{a_{1,1}a_{2,2} - a_{1,2}a_{2,1}} & a_{1,2} \\
 -\frac{1}{a_{1,1}a_{2,2} - a_{1,2}a_{2,1}} & a_{1,1}
\end{bmatrix}
\)
2.3 Duty-Cycle Calculation, Inverter Bridge

The controller implemented in Simulink is run in a discrete time frame with a sample time matching the one used in hardware implementation in Chapter 5.

![Simulink model of the used controller.](image)

**Figure 2.3.** Simulink model of the used controller.

### 2.3 Duty-Cycle Calculation, Inverter Bridge

The PWM duty-cycle calculation and inverter bridge takes the references for applied voltages in $\alpha$-$\beta$ axis and actuates them in the stator specific a-b-c axis. This process is referred to as vector control, as the voltages are actuated with the aid of vectors, calculated from reference voltages.

Depending on the reference voltages and the voltage of the DC supply, different duties for the PWM signals fed to the inverter bridge are calculated, a process also known as SVM. The inverter bridge then actuates the voltages along the three phases on the stator. The inverter bridge, illustrated in Figure 2.3, uses six transistors and diodes to control the application of voltages along the three phases.

The actuation of voltages by the SVM and inverter bridge are not ideal, which is illustrated further in Section 5.2.

![A sketch of the inverter bridge.](image)

**Figure 2.4.** A sketch of the inverter bridge. [17]
2.4 Resolver

The resolver operates by the same principles as the sensor solution made by Analog Devices. [18] The resolver consists of a rotor and a stator and operates as a rotary transformer. A carrier voltage is injected in a primary winding, \( E_0 \sin(\omega t) \), that induces a voltage in the rotor which in turn induces a voltage in two secondary windings. The principle of operation is presented in Figure 2.5. The induced voltages are dependent on the rotor position by the following relations.

The injected signal

\[
V_s(t) = E_0 \sin(\omega t) \quad (2.10a)
\]

The induced voltages

\[
V_a(t) = k_{tr} E_0 \sin(\omega t) \sin(\theta) \quad (2.10b)
\]
\[
V_b(t) = k_{tr} E_0 \sin(\omega t) \cos(\theta) \quad (2.10c)
\]

Here, \( E_0 \) is the amplitude of the injected voltage, \( \omega \) is the frequency of the injected signal, \( k_{tr} \) is a transformation constant and \( \theta \) is the position of the rotor. The rotor position is calculated from this information and is fed back to the EMCU.

![Figure 2.5. The operation principle of the resolver. [18]](image)
Chapter 3

Motor Model

The motor used in this thesis is a so called PMSM. It is also referred to as a brushless DC motor (BLDC), although that name is misleading as the voltages applied in the stator alternates. However, the PMSM shares characteristics with a DC motor such as the linear relation between current - torque and voltage - speed.

The PMSM have an electromagnetic stator and a permanent magnet rotor. The magnetic flux of the rotor is caused by permanent magnets, rather than an alternating current, which is the case in a classical DC motor. Due to no current applied in the rotor there is no need for mechanical contact between stator and rotor, therefrom the description brushless.

There are several design methods for building the rotor in a PMSM, the most notable differences in design is the number of magnets, or pole pairs, mounted. The number of pole pairs will affect the amount of torque the motor yields for a certain current. Another design difference, illustrated in Figure 3.1, is if the magnets are mounted on the iron core, surface mounted PMSM (SMPMSM), or if they are mounted inside the iron core, internal mounted PMSM (IMPMSM). The type of mounting will affect the inductances caused by the rotor. A SMPMSM will have a very small difference in inductances in $\delta$ and $\gamma$-axis and these are often regarded equal, but for the IMPMSM the interaction between the permanent magnet and the iron core will yield a difference in inductance in $\delta$ and $\gamma$-axis. The SMPMSM is usually called a non-salient motor and the IMPMSM is called a salient rotor motor.
Figure 3.1. An IMPMSM- and SMPMSM-rotor with 4 pole pairs illustrated.

The motor is driven by voltages in three different stator windings. The voltages will control the current which will induce a magnetic field that the permanent magnets in the rotor will react to, causing the rotor to turn. The radial torque of the motor is determined by the number of pole pairs, the magnetic flux of the permanent magnets, the quantity of current and its direction related to the direction of the previously mentioned magnetic flux.

Due to the property of magnets, the permanent magnetic rotor will try to align itself with the applied external magnetic field, in this case caused by the currents in the stator. The torque generated by the motor due to the magnetic fields is governed by the the equation of torque generated in a magnetic dipole from an external magnetic field [19]

\[ \vec{\tau} = \vec{\mu} \times \vec{B} \]

where \( \vec{\mu} \) is the magnetic dipole moment generated by the the permanent magnets in the rotor, \( \vec{B} \) is the magnetic flux density inside the rotor generated by the stator currents.

To suit the quantities used in this thesis, (3.1) can be rewritten with other quantities, as the magnetic dipole causes a magnetic flux \( \vec{\Phi} \) and the external magnetic field is generated by a current \( \vec{i} \) in the stator. Also due to the three phase current, \( \frac{3}{2} \), and the number of pole pairs, \( N \), the equation will be

\[ \vec{\tau} = \frac{3}{2} N \vec{\Phi} \times \vec{i} \]

As the direction of magnetic flux is defined as the \( \delta \)-direction in the rotor, the partial current that will induce a torque in the rotor is the current component aligned with the \( \gamma \)-axis. Due to this, to maximize the torque given by a certain quantity of current, the applied current should be aligned with the \( \gamma \)-axis of the rotor. The definition of the \( \delta \)-axis and the amount of generated torque is illustrated in Figure 3.2.
3.1 Motor equations in the $\delta$-$\gamma$ frame

The derivation of the motor equations will not be covered in this report, as the modeling was not in the scope of this thesis. If the reader is interested in a derivation one can be found in O. Wallmark’s Ph.D Thesis. [6] The equations used to model the engine in Simulink, in the rotor static $\delta$-$\gamma$ frame, are

$$\frac{di_\delta}{dt} = \frac{u_\delta}{L_\delta} - \frac{R}{L_\delta} i_\delta + \frac{L_\gamma}{L_\delta} \omega i_\gamma$$  \hspace{0.5cm} (3.3a)

$$\frac{di_\gamma}{dt} = \frac{u_\gamma}{L_\gamma} - \frac{R}{L_\gamma} i_\gamma - \frac{L_\delta}{L_\gamma} \omega i_\delta - \frac{\Phi}{L_\gamma} \omega$$  \hspace{0.5cm} (3.3b)

$$\frac{d\omega}{dt} = \frac{N}{J} \left(\frac{3}{2} N \Phi i_\gamma - \tau_L - \frac{D \omega}{N}\right)$$  \hspace{0.5cm} (3.3c)

$$\frac{d\theta}{dt} = \omega$$  \hspace{0.5cm} (3.3d)

where $u_{\delta,\gamma}$ and $i_{\delta,\gamma}$ are the voltages and currents in $\delta$ and $\gamma$-direction. $R$ is the stator resistance, $L_{\delta,\gamma}$ is the stator inductances in $\delta$ and $\gamma$-direction, $N$ is the number of pole pairs, $\Phi$ is the magnetic flux of the rotor, $J$ is the moment of inertia of the rotor, $\tau_L$ is the load torque and $D$ is a constant of friction.

$\delta$ and $\gamma$ denotes the true rotor frame, although in literature and in this thesis the true rotor frame is also regarded as the d-q frame, i.e. assuming that there is no error in estimation. The only time the $\delta$-$\gamma$ frame is used is when discussing estimation errors.

Equations (3.3a) and (3.3b) describes the current dynamics for the stator. The first two terms are caused by the properties of the stator and the rest of the terms are caused by the rotor. The effects caused by the rotor are generally called back electro motive force (back-EMF) and it is the information commonly used to estimate angular velocity and position. Equation (3.3c) is based on the torque equation (3.2) and Newton’s second law, effects of load torque and friction are also added.
Furthermore, $\omega$ denotes the electrical angular velocity and $\theta$ the electrical angular position. The mechanical value and the electrical value are related by the number of pole pairs, $N$, in the rotor. A movement of the rotor corresponding to the distance of a pair of magnets will provide a similar magnetic alignment. This distance is called an electric period, and there are $N$ electrical periods on a mechanical period. As all the calculations regarding current dynamics uses the electrical velocity and position they will be used throughout the thesis. When velocity or speed is mentioned it refers to the electrical angular velocity.

3.2 Motor used in this Thesis

The motor used in this project is a version of an SMPMSM developed by Stridsberg Powertrain and manufactured by SAAB Avitronics. Since its magnets are surface mounted $L_\delta = L_\gamma$ will generally be assumed throughout the rest of this thesis. Table 3.1 show the values of the motor parameters for this motor.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>0.081 $\Omega$</td>
</tr>
<tr>
<td>$L_\delta = L_\gamma$</td>
<td>$0.27 \cdot 10^{-3}$ H</td>
</tr>
<tr>
<td>$J$</td>
<td>0.0012 km$\cdot$m$^2$</td>
</tr>
<tr>
<td>$N$</td>
<td>10</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>0.0182 Wb</td>
</tr>
</tbody>
</table>

3.3 Simulink Model

![Simulink model of the PMSM](image)
3.3 Simulink Model

Figure 3.3 here illustrates the Simulink model. Since the motor is driven by voltages in the a-b-c frame and the model is in the d-q frame a frame transformation according to (2.3) has to be applied first. Following that, the motor equations, (3.3), are calculated and finally a reverse frame transformation (2.8) is applied to obtain the currents in a-b-c frame. The motor model is simulated in a continuous time frame, making simulations rather slow due to feedback of calculated variables.
Chapter 4

Estimators

Two kinds of estimators have been investigated, the extended Kalman filter and an observer based on back-EMF augmented by a low frequency injection method at low speeds. To better understand the problems connected with estimation of the motor states, a look at the possible convergence points of the estimators will be taken as well as a description on physical behaviors of the motor and sensors that can cause problems with observability.

4.1 Error Dynamics

The purpose of the observers to be presented is to estimate the position and velocity of the rotor as close to their true values as possible. However, due to the nature of the observers there exists false convergence points. With knowledge of these one can modify the observer so the area for which it converges toward the wanted convergence point is expanded so the false convergence points is excluded. The estimators will in a such sense be globally convergent, i.e. it will converge to the true position for any unknown starting position.

In a convergence point, the error in estimation of variables will be in a state of equilibrium. The difference between estimated and real states will be denoted as \( \epsilon_d = i_d - \hat{i}_d \), \( \epsilon_q = i_q - \hat{i}_q \) and \( \vartheta = \theta - \hat{\theta} \). In equilibrium we will have

\[
\begin{align*}
\frac{d}{dt} \epsilon_d &= 0 \\
\frac{d}{dt} \epsilon_q &= 0 \\
\frac{d}{dt} \vartheta &= \omega - \hat{\omega} = 0
\end{align*}
\]

If one regard the estimated rotor base d-q and the real rotor base \( \delta-\gamma \) separated by an angle \( \vartheta \), we can regard the estimated variables as functions of the real
variables by the use of equation (2.9).

\[ u_d = u_\gamma \cos(\vartheta) - u_\gamma \sin(\vartheta) \]  
\[ u_q = u_\gamma \cos(\vartheta) + u_\gamma \sin(\vartheta) \]  
\[ i_d = i_\delta \cos(\vartheta) - i_\gamma \sin(\vartheta) \]  
\[ i_q = i_\gamma \cos(\vartheta) + i_\delta \sin(\vartheta) \]

Then regard the equations of the PMSM (3.3) in the estimated base, one will have the following equations. Note that \( \Phi \) is directed along the \( \delta \)-axis and will be split into two components in the d-q base.

\[ \frac{d}{dt} i_d = \frac{u_d}{L} - \frac{R i_d}{L} + \omega \frac{i_q}{L} \]  
\[ \frac{d}{dt} i_q = \frac{u_q}{L} - \frac{R i_q}{L} - \omega \frac{i_d}{L} \]  
\[ \frac{d}{dt} \omega = \frac{N}{J} (\tau_{em} - \tau_L(\omega)) = \frac{N}{J} \left( \frac{3}{2} N \Phi i_\gamma - \tau_L(\omega) \right) \]  
\[ \frac{d}{dt} \theta = \omega \]

Here \( \tau_L(\omega) \) denotes the load torque as a function of angular velocity. It can consist of friction torque, block friction torque and various loads. Ignoring effects of the inverter, one can assume that the applied voltages are the same as the measured. The estimated variables can then be written as

\[ \frac{d}{dt} \hat{i}_d = \frac{u_d}{L} - \frac{R \hat{i}_d}{L} + \hat{\omega} \hat{i}_q \]  
\[ \frac{d}{dt} \hat{i}_q = \frac{u_q}{L} - \frac{R \hat{i}_q}{L} - \hat{\omega} \hat{i}_d - \frac{\Phi}{L} \hat{\omega} \]  
\[ \frac{d}{dt} \hat{\omega} = 0 \]  
\[ \frac{d}{dt} \hat{\theta} = \hat{\omega} \]

With the equations above we can examine how the equilibrium points will behave. Inserting (4.3) and (4.4) into (4.1) we obtain

\[ \frac{d}{dt} \epsilon_d = - \frac{R e_d}{L} + (\omega i_q - \hat{\omega} \hat{i}_q) + \frac{\Phi}{L} \omega \sin(\vartheta) \]  
\[ \frac{d}{dt} \epsilon_q = - \frac{R e_q}{L} - (\omega i_d - \hat{\omega} \hat{i}_d) - \frac{\Phi}{L} (\omega \cos(\vartheta) - \hat{\omega}) \]  
\[ \frac{d}{dt} \hat{\theta} = \omega - \hat{\omega} \]
Assuming the algorithm for calculating \( \hat{i}_d \) and \( \hat{i}_q \) is effective so the following is true: \( \epsilon_d = i_d - \hat{i}_d \to 0 \) and \( \epsilon_q = i_q - \hat{i}_q \to 0 \). Then we can identify the possible equilibrium points from the following equations:

\[
\begin{align*}
\frac{d}{dt} \epsilon_d &= (\omega i_q - \hat{\omega} \hat{i}_q) + \frac{\Phi}{L} \omega \sin(\vartheta) = 0 \quad (4.6a) \\
\frac{d}{dt} \epsilon_q &= (\omega i_d - \hat{\omega} \hat{i}_d) - \frac{\Phi}{L} (\omega \cos(\vartheta) - \hat{\omega}) = 0 \quad (4.6b) \\
\frac{d}{dt} \vartheta &= \omega - \hat{\omega} = 0 \quad (4.6c)
\end{align*}
\]

Two equilibrium points can be identified from the equation above as

\[
\begin{align*}
\hat{\omega} &= \omega \neq 0 \\
\vartheta &= 0
\end{align*}
\]

and

\[
\begin{align*}
\hat{\omega} &= \omega = 0 \\
\vartheta &= \pm 0.5 \pi \\
\tau_{em} &= \frac{3}{2} N \Phi (i_q \cos(\vartheta) - i_d \sin(\vartheta)) = \tau_L(\omega) = 0 \quad (4.7b)
\end{align*}
\]

The requirement on the electro magnetic torque is due to the derivative of the angular velocity has to be equal zero, according to (4.3c). The first equilibrium point is the one that is wanted. The second is obtained when the observer converges toward \( \hat{\omega} = \omega = 0; \vartheta = \pm 0.5\pi \). The reason for the angles \( \vartheta = \pm 0.5\pi \) in the second convergence point, are due to modeled load torque is zero when the angular velocity is zero. Also the current in the q-direction is increased toward the maximum from the torque reference, and the current in the d-axis is controlled toward zero in the controller. The following equation only holds true for \( \vartheta = \pm 0.5\pi, i_q \neq 0 \) and \( i_d = 0 \)

\[
\tau_{em} = N \Phi (i_q \cos(\vartheta) - i_d \sin(\vartheta)) = 0 \quad (4.8)
\]

Different methods for avoiding the unwanted convergence point will be described under each of the estimators.

### 4.2 Extended Kalman Filter

The linear Kalman filter is an optimal observer in the least square sense for a linear system with Gaussian noise. [20] This system, however, is nonlinear and the Kalman filter is extended to regard these non-linearities by linearizing the system to the first order around the working point of each time update.

The extended Kalman filter (EKF) is a commonly used observer for non-linear systems due to its relatively simplicity compared to other non-linear observers, although the EKF is not an optimal estimator as its linear counterpart is. [5] If initial states are false or the model is inaccurate the linearization will not be satisfactory and the estimator can diverge. The EKF is commonly used nowadays in navigational systems, GPS and in the space- and aviation industry. [5]

This algorithm is based on the work by M. Fadel et al. [2, 3], D. Simon [5], S. Bolognani et al. [1] and Z. Peroutka [4].
4.2.1 Theory

The EKF theory is similar to the linear Kalman filter, but due to the linearization in each time update the covariance of estimated states is not calculated by the Riccati equation as in the linear case. Instead the covariance of estimated states, $P_{k|k}$, is calculated with the aid of the Jacobian of the linearized state-equation and updated through the use of the calculated Kalman gain.

The EKF-algorithm uses a slightly modified version of the state-equations for the motor. The modification is that the angular velocity is assumed to change slowly compared to the other states and it can be assumed to be constant in the prediction horizon. The method is often called the infinite inertia method due to this assumption equals that the moment of inertia of the motor is infinite. The generalization cuts down on computational time without affecting the performance of the observer noticeably. [4]

\[
\begin{align*}
\frac{di_d}{dt} &= \frac{u_d}{L} - \frac{R}{L} i_d + \omega i_q \\
\frac{di_q}{dt} &= \frac{u_q}{L} - \frac{R}{L} i_q - \omega i_d - \frac{\Phi}{L} \omega \\
\frac{d\omega}{dt} &= 0 \\
\frac{d\theta}{dt} &= \omega
\end{align*}
\]  

(4.9a)

(4.9b)

(4.9c)

(4.9d)

Written more condensely with added noise to states and measurements which are affected by errors in parameters, measurements and sample

\[
\begin{align*}
\dot{x} &= g(x, u) + w \\
y &= Cx + v
\end{align*}
\]  

(4.10a)

(4.10b)

where

\[
\begin{align*}
x &= \begin{bmatrix} i_d & i_q & \omega & \theta \end{bmatrix}^T \\
C &= \begin{bmatrix} 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \end{bmatrix} \\
u &= \begin{bmatrix} u_d & u_q \end{bmatrix}^T \\
y &= \begin{bmatrix} i_d & i_q \end{bmatrix}^T
\end{align*}
\]  

(4.10c)

(4.10d)

(4.10e)

(4.10f)

In the algorithm that follows the noises are handled as covariance matrices, that represents the mean square error of the quantities.

\[
\begin{align*}
Q &= \text{cov}(w) = E\{ww^T\} \\
Z &= \text{cov}(v) = E\{vv^T\}
\end{align*}
\]  

(4.11a)

(4.11b)

The EKF algorithm will be implemented on discrete form, and approximated in
the first order. The prediction step will then take the following form

\[
x(k+1) = x(k) + \dot{x}(k) \cdot T_e = \begin{pmatrix}
i_d(k) + \left( \frac{u_d(k)}{L} - \frac{R_{id}(k)}{L} + \omega(k) i_d(k) \right) \cdot T_e \\i_q(k) + \left( \frac{u_q(k)}{L} - \frac{R_{iq}(k)}{L} - \omega(k) i_q(k) - \frac{\Phi \omega(k)}{L} \right) \cdot T_e \\
\omega(k) \\
\theta(k) + \omega(k) \cdot T_e
\end{pmatrix}
\]  

(4.12)

To compute the Kalman gain, one must also know the covariance matrix of estimation, \(P\), and it is defined at the \(k\)-th time sample as

\[
P_k = E \left\{ [x_k - \hat{x}_k][x_k - \hat{x}_k]^T \right\}
\]  

(4.13)

In a Kalman filter for a linear system one would calculate the covariance matrix of the observations with the aid of the Riccati equation. For this non-linear system, \(P\) is not static, but has to be calculated at each linearization of the system. To calculate it we need the aid of the Jacobian of the predicted system states which is defined as follow

\[
F_k = \frac{\partial x(k+1)}{\partial x} = \begin{bmatrix}
1 - \frac{RF}{T} & T_e \omega_k & T_e i_q,k & 0 \\
-T_e \omega_k & 1 - \frac{RF}{T} & 0 & 0 \\
0 & T_e (i_d,k + \frac{\Phi}{L}) & 1 & 0 \\
0 & 0 & 0 & T_e
\end{bmatrix}
\]  

(4.14)

4.2.2 Algorithm

The basic algorithm the EKF operates by, is the following. The vectors and matrices are as defined in the above section.

Compute the state and error covariance ahead, called prediction step

\[
\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + \hat{x}_{k-1} T_e
\]  

(4.15a)

\[
P_{k|k-1} = F_{k-1} P_{k-1|k-1} F_{k-1}^T + Q_{k-1}
\]  

(4.15b)

Compute the Kalman gain

\[
K_k = P_{k|k-1} C^T (CP_{k|k-1} C^T + Z_{k-1})^{-1}
\]  

(4.15c)

Update estimation with measurement and error covariance matrix, called innovation step

\[
\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (y_k - C \hat{x}_{k|k-1})
\]  

(4.15d)

\[
P_{k|k} = [I - K_k C] P_{k|k-1}
\]  

(4.15e)
4.2.3 Filter Tuning

There are three covariance matrices for the EKF algorithm that needs tuning. $P_0$ - the initial choice of the covariance of estimation, $Q$ - the covariance of states and $Z$ - the covariance of measurements. $P_0$ represents the mean-square error of the initial estimations. $P_0$ does not have a major impact on the system, it will affect the initial transient but nothing more. $Q$ contains information about system noise and parameter errors. Large elements indicates larger uncertainties and will yield an increased Kalman gain. $Z$ represents the covariance of error in the measured currents, large elements in this matrix indicates much noise and uncertain measurements and will yield a decreased Kalman gain.

When choosing the covariances, it is common to assume the matrices to be diagonal because it is hard to obtain statistical knowledge about the off diagonal terms. Also, practice has shown that off diagonal terms have a small impact on the system compared to the diagonal terms. The choice of terms will have an effect on the convergence of the EKF, if the values of covariance is too far off it will lead to an improper Kalman gain resulting in a divergent algorithm.

The idea to design the covariance terms is rather simple but time consuming. To tune the filter terms, you run the motor controller from measured values on angular speed and angular position, and let the observer estimate the values without feeding them to the controller, this is referred to as running the observer offline. The different choices of covariance terms will yield different settling times in a step in reference speed combined with that the covariance terms are independent of each other makes it possible to tune the covariance term through iteration. Each covariance term can be tuned by iterative simulation or measurements until the lowest settling time is found for each term.

4.2.4 Convergence

The extended Kalman filter have the characteristic false convergence points that is represented in Section 4.1. Empirical studies in the form of Simulink simulations, shows for which initial position errors the observer converges toward the wanted. Figure 4.1 illustrates for which initial position errors the algorithm converges toward the wanted convergence point.
4.2 Extended Kalman Filter

A method proposed by Z. Zheng et al [2] is to induce a disturbance in the current dynamics equation in q-direction during the prediction step in the estimator, as the unwished equilibrium points is mostly dependent on this equation. The equation is modified according to:

\[
\frac{d\hat{i}_q}{dt} = \frac{\hat{u}_q}{L} - \frac{R\hat{i}_q}{L} - \hat{\omega}\hat{i}_d - \frac{\Phi}{L}\hat{\omega} + \frac{kRi_q}{L} \tag{4.16}
\]

where \(k\) is a positive coefficient. This can be seen as a perturbation of the stator resistance, the effects of this perturbation will be eliminated by the robustness of the system. Just a small value of \(k\) will be enough to push the area of convergence to envelope the unwanted equilibrium points with the effect that those equilibrium points will be unstable, and the wanted convergence point will remain stable. This modified convergence is illustrated in Figure 4.2.
4.2.5 Simulink Model

Figure 4.3. Simulink model of the EKF algorithm.

The blocks in this Simulink model, illustrated in Figure 4.3, represents the equations (4.12), (4.14) and (4.15). The values of the state vector, $x$, and the covariance matrix of estimation, $P$, are stored and used for calculations in the following time step. The algorithm is run in discrete time, matching the sample time of the hardware implementation.

4.3 Phase Locked Loop

The estimator of a Phase Locked Loop (PLL) type utilizes the fact that there is a difference in expected and real voltages in the d-direction of the rotor. This difference is due to the back-EMF differs from the expected, caused by a difference between real and estimated rotor position and speed. This algorithm is based on the work by L. Harnefors et al. [7] and O. Wallmark [6].

4.3.1 Estimating Position Error

The method of PLL needs information about how the current estimated rotor position is related to the true rotor position. In this section a way to obtain such information from measurable quantities is deducted originating from the equations of the PMSM in d-q frame.

The motor equations are as in (3.3), here written in a more condensed form.

$$L \frac{di}{dt} = u - Zi - \Phi_m$$ (4.17)
where the following are vectors $u = [u_d, u_q]^T$, $\Phi_m = [0, \omega \Phi]^T$, $i = [i_d, i_q]^T$. $L$ and $Z$ are matrices

$$L = \begin{bmatrix} L_d & 0 \\ 0 & L_q \end{bmatrix}, \quad Z = \begin{bmatrix} R & -\omega L_q \\ \omega L_d & R \end{bmatrix} \tag{4.18}$$

Since the bandwidth of the estimator is larger than the bandwidth of the controller the effects of current dynamics can be disregarded. Using the transformation matrix $e^{J\theta}$ that represents the difference between the base of estimation and the true motor position, the estimated voltages and real voltages can then be represented as functions of measured currents in the d-q base. This is a condensed way of representing the transformation matrix (2.5), where $J = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$ and $\vartheta = \theta - \hat{\theta}$.

$$u = \hat{Z}i - \hat{\Phi}_m \tag{4.19a}$$

$$u = e^{J\theta}Z e^{-J\theta}i - e^{J\theta} \Phi_m \tag{4.19b}$$

The $Z$ matrix and the $\Phi_m$ vector is transformed to the true position from the estimated position. The error in voltage estimation is then defined as

$$\epsilon = u - \hat{u} = [e^{J\theta} Z e^{-J\theta} - \hat{Z}]i - e^{J\theta} \Phi_m + \hat{\Phi}_m \tag{4.20}$$

We assume that the impedance matrix $Z$ and vector $\Phi_m$ are well estimated ($\hat{Z} \approx Z$, $\hat{\Phi}_m \approx \Phi_m$) and the rotor to be non salient ($L_q \approx L_d$). By those assumptions, $Z$ and $e^{-J\theta}$ will commute. Also $e^{J\theta}$ is an unitary operator so

$$\epsilon = u - \hat{u} = [e^{J\theta} e^{-J\theta} Z - \hat{Z}]i + [I - e^{J\theta}] \Phi_m \approx [I - e^{J\theta}] \Phi_m \tag{4.21}$$

And just looking at the d-axis voltage

$$\epsilon_d = u_d - \hat{u}_d \approx -\omega \Phi \sin(\vartheta) \tag{4.22}$$

By comparing (4.22) with the steady state voltage equation for the PMSM in d-axis

$$0 = u_d - Ri_d + \omega Li_q - \omega \Phi \sin(\vartheta) \tag{4.23}$$

one find that the error can be estimated by

$$\epsilon_d = u_d - \hat{u}_d \approx -\omega \Phi \sin(\vartheta) = -u_d + Ri_d - \omega Li_q \tag{4.24}$$

which gives information on how the estimated position is compared to the real position. Hence we can estimate the position error and the observer will reduce this error to zero according to the following algorithm.

### 4.3.2 Algorithm and Observer Gains

As seen in the section above, the error in estimation can be written in the form

$$\epsilon = K \sin[m(\theta - \hat{\theta})] \tag{4.25}$$
The algorithm then used to control the estimated speed and position to its true value is
\[ \dot{\hat{\omega}} = \gamma_1 \epsilon \] (4.26)
\[ \dot{\hat{\theta}} = \dot{\hat{\omega}} + \gamma_2 \epsilon \] (4.27)
The estimator in this form is non-linear, but to determine the gain parameters \( \gamma_1 \) and \( \gamma_2 \) by the help of a characteristic equation, the observer is linearized by a Taylor expansion in first order in the region \( \theta \approx \hat{\theta} \). By using \( \sin(\theta) \approx \theta \) for small \( \theta \) we obtain
\[ \dot{\hat{\omega}} = \gamma_1' (\theta - \hat{\theta}) \] where \( \gamma_1' = \gamma_1 m K \) (4.28)
\[ \dot{\hat{\theta}} = \dot{\hat{\omega}} + \gamma_2' (\theta - \hat{\theta}) \] where \( \gamma_2' = \gamma_2 m K \) (4.29)
The differential equations above have the following characteristic equation
\[ s^2 + \gamma_2' s + \gamma_1' = 0 \]. By properly choosing the gain parameters \( \gamma_1 \) and \( \gamma_2 \) one can place the poles in the system arbitrarily. In this case we choose to place both poles in \( s = -\rho \) where \( \rho \) is a positive constant. This placing yields the characteristic equation
\[ s^2 + 2 \rho s + \rho^2 = 0 \]. So the gains \( \gamma_1 \) and \( \gamma_2 \) can be chosen as
\[ \gamma_1 = \frac{\rho^2}{m K}, \quad \gamma_2 = \frac{2\rho}{m K} \] (4.30)
The choice of \( \rho \) will have an impact on the dynamics of the estimator. Larger \( \rho \) will make the observer faster but more sensitive to noise, while lower will make the observer slower but less sensitive to noise. For this specific application we have the following from (4.22) and (4.25)
\[ \epsilon = K \sin(m \hat{\theta}) \] and \( \epsilon = \omega \Phi \sin(\hat{\theta}) \) (4.31)
so we can deduce that \( K = \omega \Phi \) and \( m = 1 \).
The gains will also be modified for lower speeds, as the gain is inversely proportional to the estimated speed so low speeds will yield excessive gains. The modification is that the pole for the observer is modified for speed under a certain limit according to: \( \rho = \frac{\omega \Delta}{\omega_\Delta} \). This choice will remove excessive gains, but it will also lower the bandwidth and convergence of the estimator.
The gains in this observer is finally chosen as (sgn is the signum function)
\[ \text{for} |\omega| \geq \omega_\Delta : \quad \gamma_1 = \frac{\rho_\Delta^2}{\omega_\Phi}, \quad \gamma_2 = \frac{2\rho_\Delta}{\omega_\Phi} \] (4.32a)
\[ \text{for} |\omega| < \omega_\Delta : \quad \gamma_1 = \frac{\rho_\Delta^2 \hat{\omega}}{\omega_\Delta^2 \Phi}, \quad \gamma_2 = \frac{2\rho \text{sgn}(\hat{\omega})}{\omega_\Delta \Phi} \] (4.32b)

4.3.3 Convergence

With the use of Lyapunov’s direct method one can show that the PLL observer is globally asymptotically stable above the limit velocity \( \omega_\Delta \) [7]. By assuming the
true speed and position be given by \( \dot{\omega} = 0 \) and \( \dot{\theta} = \omega \) we can formulate how the estimation errors, \( \tilde{\omega} \) and \( \tilde{\theta} \), will behave in an equilibrium point.

\[
\begin{align*}
\dot{\omega} &= -\gamma_1' \sin(m\theta) \\
\dot{\theta} &= \tilde{\omega} - \gamma_2' \sin(m\theta)
\end{align*}
\]

(4.33a) (4.33b)

The stability of these two equations can be proved by the Lyapunov’s direct method. Lyapunov’s direct method states that [20]: An equilibrium point, \( x_0 \) to the system \( \dot{x} = f(x), x \in \mathbb{R}^n \) is globally asymptotically stable if one can find a function \( V \), that satisfies the following statements (\( V_x \) is the row vector \( \partial V/\partial x_1, ..., \partial V/\partial x_n \))

\[
V(x_0) = 0; V(x) > 0, x \neq x_0; V_x(x)f(x) < 0, x \neq x_0
\]

(4.34)

The stability will be proved by the following Lyapunov candidate for the PLL system, also illustrated in Figure 4.4

\[
V(\tilde{\omega}, \tilde{\theta}) = \frac{1}{2} \tilde{\omega}^2 + \gamma_1'(1 - \cos(m\theta))
\]

(4.35)

and its derivative

\[
\begin{align*}
\dot{V} &= \frac{\partial V}{\partial \tilde{\omega}} \dot{\tilde{\omega}} + \frac{\partial V}{\partial \tilde{\theta}} \dot{\tilde{\theta}} = \tilde{\omega} \dot{\tilde{\omega}} + \gamma_1' \dot{\tilde{\theta}} \sin(m\theta) \\
&= \tilde{\omega}(-\gamma_1' \sin(m\theta)) + \gamma_1' \sin(m\theta)(\tilde{\omega} - \gamma_2' \sin(m\theta)) \\
&= -\gamma_1' \gamma_2' \sin^2(m\theta)
\end{align*}
\]

(4.36)

We can see that for \( \{\gamma_1', \gamma_2'\} > 0 \) : \( V(\tilde{\omega}, \tilde{\theta}) > 0, \tilde{\omega} \neq 0, \tilde{\theta} \neq 2n\pi/m, V(0, 2n\pi/m) = 0 \) and \( -\gamma_1' \gamma_2' \sin^2(m\theta) \leq 0 \). This concludes that \( \{0, 2n\pi/m\} \) are local convergence points for this estimator. But since \( m = 1 \) and in the estimation \( \tilde{\theta} = 0 \) is regarded as the same point as \( \tilde{\theta} = 2n\pi, n \neq 0 \) we can conclude that the estimator is globally asymptotically stable for \( |\tilde{\omega}| > \omega_\Delta \).
4.3.4 Simulink Model

This algorithm is run in discrete time sampled with a time period matching that of the controller. Firstly in this model the error in d-axis voltage is calculated according to (4.24). The gains for the observer is then calculated by (4.32). The
magnitude of the estimated velocity decides which gains to use. The gain is multiplied with the error in voltage according to (4.26) and (4.27) and \( \dot{\omega}_k \) and \( \dot{\theta}_k \) are obtained. \( \dot{\omega}_k \) and \( \dot{\theta}_k \) are then obtained through a discrete summation as 
\[
\dot{\omega}_k = \dot{\omega}_{k-1} + T_e \cdot \dot{\omega}_k \quad \text{and} \quad \dot{\theta}_k = \dot{\theta}_{k-1} + T_e \cdot \dot{\theta}_k.
\]

4.4 Low Frequency Injection

Low frequency injection (LF-Inj) relies on the fact that if the real and the estimated position differ and a signal is injected along the estimated d-axis it will induce a change of torque that is detectable. If the estimated position is correct the injected signal will induce no disturbances. Since the change in velocity is relatively slow compared to the rest of the system, the injected signal will need to be of a low frequency, in order to induce detectable errors in the system. This algorithm is based on work by T. Kereszty et al. [8].

4.4.1 Effects of the Carrier Signal

From the motor equations given in Chapter 3 we use the equation for the current dynamics in the q-direction on the rotor, as we detect the induced torque along that direction.

\[
\frac{d i_q}{dt} = \frac{u_q}{L} - \frac{R}{L} i_q - \frac{\omega L}{T} i_d - \frac{\Phi}{T} \omega
\]  

(4.37)

The equation above is then considered in steady state as the usage of the current derivative will only induce noise into the observer as there are uncertainties in the measurements of the current.

\[
u_q = R i_q + \omega L i_d + \omega \Phi
\]  

(4.38)

The back-EMF is identified as \( \omega \Phi \) in the above equation and it will be used to estimate the error in position in this algorithm. Further we use these two relations from the motor equation

\[
\tau = \frac{3N}{2} \Phi i_q
\]  

(4.39a)

\[
\frac{d}{dt} \omega = \frac{N}{J} (\tau - \tau_L)
\]  

(4.39b)

The signal used to induce measurable disturbances has the RMS current \( I_{cd}^c \) and it varies with the angular frequency \( \omega_c \). The injected signal can then be written as

\[
i_{cd}^c = \sqrt{2} I_{cd}^c \cos(\omega_c t)
\]  

(4.40)

Assuming there is a difference between estimated and real position of the rotor \( \theta = \dot{\theta} - \theta \) the injected signal will be divided into a q- and d-component

\[
i_{cd}(t) = i_{cd}^c(t) \cos(\theta)
\]  

(4.41)

\[
i_{cq}(t) = i_{cd}^c(t) \sin(\theta)
\]  

(4.42)
According to (4.39) above the component $i_{cq}(t)$ will induce a torque and change in velocity according to

$$
\tau_c(t) = \frac{3N}{2} \Phi i_{cd}(t) \sin(\vartheta)
$$

(4.43)

$$
\omega_c(t) = \sin(\vartheta) \frac{3N^2 \Phi}{2J} \int_0^t i_{cd}(t') dt'
$$

(4.44)

Inserting $i_{cd}$ and carrying out the integration we obtain

$$
\omega_c(t) = \sin(\vartheta) \frac{3N^2 \Phi^2 I_{cd}}{\sqrt{2J} \omega_c} \sin(\omega_c t)
$$

(4.45)

The back-EMF induced by the carrier signal in the true rotor frame is then obtained by

$$
e_{cq}(t) = \sin(\vartheta) \frac{3N^2 \Phi^2 I_{cd}}{\sqrt{2J} \omega_c} \sin(\omega_c t)
$$

(4.46)

Since the signal is measured in the estimated frame, one last frame transformation is applied to obtain the back-EMF used in the algorithm

$$
e_{cq}(t) = -e_{cq}(t) \cos(\vartheta)
$$

(4.47)

By assuming that the error is small and using a Taylor expansion in first order around $\vartheta \approx 0$, $(\cos(\vartheta) \approx 0, \sin(\vartheta) \approx \vartheta)$ we will then obtain

$$
e_{cq}(t) \approx -\vartheta \frac{3N^2 \Phi^2 I_{cd}}{\sqrt{2J} \omega_c} \sin(\omega_c t)
$$

(4.48)

**4.4.2 Algorithm**

From (4.38) in the former section one can extract information about the estimated back-EMF in steady state as

$$
\dot{e}_{cq}^e(t) = -u_{qf}^e(t) + \hat{R}_{i_q}(t) + \hat{\omega} \hat{L}_{i_d}(t)
$$

(4.49)

To obtain the needed information, $\dot{e}_{cq}^e(t)$, the estimated back-EMF is filtered to extract the components with the angular frequency $\omega_c$. This is achieved by removing the zero-average and the trend.

$$
\dot{e}_{cq}^e(t) = \dot{e}_q^e(t) - \frac{1}{T_e} \int_{t-T_e}^t \dot{e}_q^e(t') dt' - \frac{1}{2} \frac{d}{dt} \frac{1}{T_e} \int_{t-T_e}^t \dot{e}_q^e(t') dt'
$$

(4.50)

where $T_e = 2\pi/\omega_c$ is the period of the carry signal.

By demodulating the above error with a signal: $\sin(\omega_c t)$ and taking the moving average over half a period, one obtains a signal that is constant in steady state, which will be used to control the error in velocity to zero according to

$$
F_{\epsilon}(t) = \frac{2}{T_e} \int_{t-T_e/2}^t \dot{e}_{cq}^e(t') \sin(\omega_c t') dt'
$$

(4.51)
4.4 Low Frequency Injection

The method to control the error to zero is by a simple PI-controller.

\[ \omega_e(t) = k_p F_e(t) + k_i \int_0^t F_e(t')dt' \]  

(4.52)

where \( k_i \) and \( k_p \) are the integral and proportional observer gains.

The motivation for choosing the above signal to control the error to zero is the following. Assume that \( \dot{e}_{cq}(t) \approx e_{cq}(t) \) one will find the value \( F_e \) to be

\[ F_e(t) \approx \frac{2}{T_c} \int_{t-T_c/2}^t -\dot{\theta} \frac{3N^2\Phi^2 I_{cd}}{\sqrt{2}J\omega_c} \sin^2(\omega_c t')dt' = -\frac{3N^2\Phi^2 I_{cd}}{2\sqrt{2}J\omega_c} \]  

(4.53)

As seen when the signal \( F_e \) is controlled to zero, the error in position will also be driven close to zero. This algorithm works poorly during dynamic operation, so in order to improve the dynamic properties, the algorithm is augmented by a feed forward based on the steady state voltage equation. To remove the oscillating component induced by \( i_{cd} \), the moving average of the feed forward is used.

\[ \hat{\omega}_u(t) = \frac{1}{T_c} \int_{t-T_c}^t \frac{u_{eq}^{ef}(t') - \dot{R}_i(t')}{L_i(t') + \Phi} dt' \]  

(4.54)

The final estimated rotor velocity is then

\[ \hat{\omega}(t) = \hat{\omega}_u(t) + \omega_e(t) \]  

(4.55)

The rotor position is then estimated from a time integration of the estimated speed.

\[ \dot{\theta}(t) = \int_0^t \hat{\omega}(t')dt' \]  

(4.56)

4.4.3 Convergence

Unfortunately, the convergence of injection methods is not discussed in any of the articles regarding such methods. Simulations have shown that the convergence area for this estimator is rather narrow and follow the typical false convergence behavior in Section 4.1. One thing is slightly different though, in a false convergence point the velocity of the observer does not approach zero. This is due to the injected current in d-axis generating a torque.

One reason for the narrow convergence area is probably due to how the angular position is estimated. The angular position is obtained through a direct integration of the angular velocity without any correctional terms, and the angular velocity is not estimated correctly always due to filtering, measurement noises and errors in parameters which all will build up in the integrator until it will diverge. This, however, could likely be avoided by adding a correctional term to the integration similar to the PLL algorithm, but this has not been implemented into the estimator as of now.
First the error in q-axis voltage (4.49) is calculated followed by the extraction of the error caused by the injected signal using (4.50) and (4.51). The velocity is the estimated by (4.52) augmented by (4.54). The estimated position is integrated from the estimated velocity according to (4.56). The proposed algorithm in Section 4.4.2 is in a continuous time frame and has to be modified to run in the discrete time frame used in simulations.

The integrations are calculated by summations and the calculations of moving averages in (4.50),(4.51) and (4.54) are implemented by FIR filters. The FIR filters have $T_c/T_e$ coefficients for a full period of the carrier signal and the coefficients have a weight of $T_e/T_c$ each. If one denote the filtered signal at sample $k$ as $y(k)$ and the unfiltered signal as $x(k)$, the moving average will be implemented as

$$y(k) = \sum_{i=0}^{T_e/T_c} T_e \frac{T_c}{T_e} x(k-i) \quad (4.57)$$

The controller also has to be modified for this observer. In the controller the reference for current in the d-axis of the motor, $i_{ref}^d$, is changed from 0 to the injected carrier signal used for this algorithm $i_{ref}^{inj} = \sqrt{2}I_{e \omega} \cos(\omega t)$.

### 4.5 Combined PLL and LF Injection

Due to the nature of the PLL and the LF-Inj observers the thought of integrating them into a combined observer seems desirable as they perform well under different circumstances. The idea is to use PLL over a certain speed and below this speed LF-Inj will be used.
4.5 Combined PLL and LF Injection

4.5.1 Transitions

The main problem in general with combining observers is the transition between them, especially in this case as both observers are divergent outside of their operation speeds. The approach to deal with this situation is to use a hysteresis in the observed speed to switch between the modes of observation. The internal states will be reset with the states of the already used observer when the new one is engaged. The hysteresis is implemented by the aid of two switch variables. The first switch will control when the algorithm will be enabled, the second will control which estimator feeds information to the controller. The switches will work according to Table 4.1 and Table 4.2.

<table>
<thead>
<tr>
<th>Table 4.1. Operation of switch 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed:</td>
</tr>
<tr>
<td>$\omega &gt; \omega_{\Delta_{2}}$</td>
</tr>
<tr>
<td>Enable:</td>
</tr>
<tr>
<td>PLL: on</td>
</tr>
<tr>
<td>LF-Inj: off</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4.2. Operation of switch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed:</td>
</tr>
<tr>
<td>$\omega &gt; \omega_{\Delta_{2}}$</td>
</tr>
<tr>
<td>Feed to controller:</td>
</tr>
<tr>
<td>PLL: on</td>
</tr>
<tr>
<td>LF-Inj: off</td>
</tr>
</tbody>
</table>
4.5.2 Simulink Model

The PLL algorithm and the LF-Inj operates as in previous models. Switch 1 is implemented in the select-mode subsystem and switch 2 from the hysteresis block.
Chapter 5

Implementation

To test the theories and see how well their simulated results compare to a real motor an estimation method is chosen for implementation on hardware. The observer chosen for implementation is the EKF algorithm, with the following points of motivation and from the results of simulations as shown in Chapter 6.

- It is able to track both angular position and speed offline, thus making it possible to evaluate the algorithm offline before using it online.
- Being able to start up successfully from any unknown starting position.
- No need to induce any carrier voltages into the motor.
- The currents are not ideal, there is noise in the current that could make an injected signal hard to detect.
- No transitions between modes of observation.

5.1 Hardware

The major components of the hardware environment are

- The electric motor control unit (EMCU). The unit containing the electronics used to control the motor, including controller, observer, and other routines on a DSP. Also handling the power supply to the motor.
- A SMPMSM as described in Chapter 3.
- A brake. Used to actuate various loads on the motor.
- Two PCs. One used to command and handling the power supply to the EMCU and also to control the brake. The second computer is used to program the DSP with the aid of CodeWarrior and to retrieve data from the DSP into Matlab. The communication is done through a JTAG interface.
5.2 Physical Limitations on Motor and Sensors

There are a few limitations in the development hardware that will cause problems for the implementation. First, there is too little available performance on the DSP, forcing the control algorithm and the observer to run at half the intended frequency. Secondly there is no compensation for zero-crossing distortion.

The platform for implementation is a developing hardware installed in a test bench at SAAB Avitronics. Due to this, there are constraints on the available performance. The system is designed to operate with a fixed time period of 100 $\mu$s. The controller and other components utilize about 60 $\mu$s, leaving too little space for the proposed observer. The solution has been to run the system using a 200 $\mu$s time period instead, resulting in worsened dynamics in the controller due to unchanged time-dependent parameters and the actuated voltages from SVM and inverter bridge will be of poorer quality. Also by running the system at a lower frequency yields less measuring points per electrical period of the motor, which affects the performance of the observer.

Ideally the actuated voltages and measured currents should be pure sine functions, but there is noise and possible overtones. Also, the SVM and inverter bridge have problems at so called zero-crossings. Zero-crossings occurs when the space vector passes the a-b-c axes of the stator windings. The effects of zero-crossings can be seen in Figure 5.1. At low loads these interferences are rather large and will have a large impact, at higher loads the current measurements will be of better quality.

To gain some perspective on the problems with these current acquisitions, a measuring series was made on SAAB’s next generation of EMCU that compensates for zero-crossing distortion. The control algorithm was allowed to run at its intended 10 kHz and the velocity during measurements were the same as previously and the load was set to 3 Nm.

In Figure 5.2 it is apparent that the voltage actuation and in its turn the current measurements can perform a lot better than in the developing hardware used for the testing of the C coded EKF algorithm.
5.2 Physical Limitations on Motor and Sensors

Figure 5.1. Top picture shows the current acquisition with a load of 3 Nm and the bottom 6 Nm. The zero-crossing distortions can be seen at the angles $0, \pm 2\pi/3, \pm 4\pi/3$ and $\pi$ of an electrical period of the current.

Figure 5.2. Current measurement in the new EMCU at a load of 3 Nm.
5.3 Development Environment

The development of the algorithm in C was made with the aid of the Legacy Code Toolbox (LCT) for Matlab. [21] LCT allows you to compile C code into S-functions that can be incorporated into Simulink. This is particularly nice as you can test and evaluate the algorithm in a modeled environment before applying it to hardware. Further, changes in the code can be tested quickly without risk of harming the hardware. However, the fixed point integers compiled in the development PC is of 32-bits, compared to the 16-bits of the DSP. As discovered later, the lower resolution of the integers in the DSP led to numerical errors in the DSP that was not seen in the PC simulations.

When coded, the observer was incorporated with the motor control algorithm by SAAB and compiled in the CodeWarrior environment which also was used to download the application to the EMCU.

5.4 Implementation in C

To be able to run the algorithm on a DSP it has to be coded and there are a few considerations to be done and they are presented in this section.

5.4.1 Fixed Point Implementation

The implementation is in a fixed base of 16 bits, so there will only be $2^{16}$ different available values for all quantities used in the calculations. To assure that the available values are used to their full potential all variables and parameters will be scaled to values ranging from $[-1, 1 - 2^{-15}]$. Due to the 16-bit implementation, the values will be represented in the base of $2^{-15}$ so all available values are integers between $[-32768, 32767]$.

To scale all variables and parameters to this representation scaling functions are used, denoted $G(\cdot)$ in this thesis and scaled variables are obtained as $\lambda_{sc} = G(\lambda) \cdot \lambda$. The general idea when designing these functions is to divide by the the maximum value that the variable can obtain and then multiply it with the maximum desired value of the scaled variable. The scaled values will be dimensionless, they can be regarded as a percent value of the quantities’ maximum values. There will be three basic scaling functions, the rest of the scaling functions can be obtained by a combination of the basic three according to Table 5.1. To obtain the unscaled values one just multiplies the scaled value with the inverted scaling function, $\lambda = G^{-1}(\lambda) \cdot \lambda_{sc}$. The following example shows how to handle the scaling functions.
### Example 5.1: How to handle scaling functions

For example let us calculate the voltage from a known resistance and current, \( U = RI \).

Let the scaling functions be \( G(A) = \frac{1}{10 \Omega} \), \( G(V) = \frac{1}{200 \text{V}} \), \( G(\Omega) = \frac{G(V)}{G(A)} = \frac{10}{200} \) and the known quantities be \( R = 10 \Omega, I = 2 \text{A} \).

Unscaled the result will be: \( U = RI = 2 \cdot 10 = 20 \text{V} \).

Scaled: \( U_{sc} = RscI_{sc} = R \cdot G(\Omega) \cdot I \cdot G(A) = 0.5 \cdot 0.2 = 0.1 \)

and \( U = U_{sc} \cdot G^{-1}(V) = 0.1 \cdot 200 = 20 \text{V} \).

As seen the scaling functions will have no impact on the final result.

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Function</th>
<th>Combined Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current</td>
<td>( G(A) )</td>
<td>-</td>
</tr>
<tr>
<td>Voltage</td>
<td>( G(V) )</td>
<td>-</td>
</tr>
<tr>
<td>Angular Velocity</td>
<td>( G(\omega) )</td>
<td>-</td>
</tr>
<tr>
<td>Time</td>
<td>( G(t) )</td>
<td>( G^{-1}(\omega) )</td>
</tr>
<tr>
<td>Resistance</td>
<td>( G(\Omega) )</td>
<td>( G(V) \cdot G^{-1}(A) )</td>
</tr>
<tr>
<td>Inductance</td>
<td>( G(H) )</td>
<td>( G(V) \cdot G^{-1}(\omega) \cdot G^{-1}(A) )</td>
</tr>
<tr>
<td>Magnetic Flux</td>
<td>( G(Wb) )</td>
<td>( G(V) \cdot G^{-1}(\omega) )</td>
</tr>
<tr>
<td>Angular position</td>
<td>( G(\theta) = 1 )</td>
<td>( G(\omega) \cdot G^{-1}(\omega) )</td>
</tr>
</tbody>
</table>

The values of these scaling functions were chosen to match the ones used by SAAB Avitronics. Table 5.2 presents these values and also presents the resolution of the variables and Table 5.3 presents the scaled values. Worth noting is that since the scaling function of angular position is 1, it must be handled differently.

Angular position will be implemented as an 8-bit integer number varying from \([0, 255]\), representing an electrical period \([0, 2\pi]\). The choice of making the angular position an 8-bit integer is to match SAAB’s representation of angular position in the motor control algorithm.

<table>
<thead>
<tr>
<th>Function</th>
<th>Value</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>( G(A) )</td>
<td>( \frac{0.1}{10} )</td>
<td>0.0037 A</td>
</tr>
<tr>
<td>( G(V) )</td>
<td>( \frac{0.5}{200} )</td>
<td>0.0244 V</td>
</tr>
<tr>
<td>( G(\omega) )</td>
<td>( \frac{0.1 \cdot 2\pi}{2\pi} )</td>
<td>0.1 rad/s</td>
</tr>
<tr>
<td>( G(\theta) )</td>
<td>1</td>
<td>0.0245 rad</td>
</tr>
</tbody>
</table>
Table 5.3. Values of the scaled parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Scaled value (in base $2^{-15}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>398</td>
</tr>
<tr>
<td>L</td>
<td>4348</td>
</tr>
<tr>
<td>Φ</td>
<td>4884</td>
</tr>
<tr>
<td>$T_c$</td>
<td>21475</td>
</tr>
</tbody>
</table>

It is not only the motor parameters and the variables that needs to be scaled in the C-code implementation. The covariance matrices for the EKF also needs to be scaled. The elements representing the covariance of the states and measurements will be scaled by the corresponding scaling function squared. For example the covariance of current measurement in d-axis will have the scaled value:

$$i_{dsc} = i_d \cdot G^2(A)$$

The reason behind this scaling is made possible by examining the definition of the covariance matrices $Q$ (4.11a) and $Z$ (4.11b). The effect of this scaling is unfortunately that the scaled filter parameters will be very small and having just a short range of values to choose from that will not lead to divergence.

The scaling of the elements of the covariance matrices would be according to:

$$P_{ijsc} = P_{ij} \cdot G_i \cdot G_j$$ (5.1a)

$$Q_{ijsc} = Q_{ij} \cdot G_i \cdot G_j$$ (5.1b)

$$Z_{ijsc} = Z_{ij} \cdot G_i \cdot G_j$$ (5.1c)

Where $G = [G(A), G(A), G(\omega), G(\theta)]$.

The lack of resolution in the covariance matrix of estimation, $P$, has proved in tests to be of concern. The elements are rather low to begin with and after scaling most of them have ended up to have too low values to be somewhat correctly described by a 16-bit number, many of them only alternates between a value of 0 and 1.

5.4.2 Computational Errors

Computational errors can occur with the use of values in a fixed base, these errors are overflow and rounding errors. Overflow is when a result of the calculations is a value outside of the range of available values. Rounding errors occurs in divisions and multiplications, the result of these operations with 16-bit numbers is a 32-bit number. The 32-bit number has to be rounded into a 16-bit number to be usable in the rest of the algorithm and the rounding is toward the closest 16-bit number. In several consecutive multiplications, as in this algorithm, the errors will stack and could cause problems. The following table shows for which circumstances overflow will occur, to avoid overflows and rounding errors one have to evaluate the possible scenarios where they can occur, and designing the scaling functions so these scenarios are avoided.
Table 5.4. For which scenarios overflow will occur.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition: (a + b)</td>
<td>(</td>
</tr>
<tr>
<td>Subtraction: (a - b)</td>
<td>(</td>
</tr>
<tr>
<td>Division: (a/b)</td>
<td>(</td>
</tr>
</tbody>
</table>

Further more, so called intrinsic functions are used. They will assure that in case of overflow in addition, subtraction and division operations, the result will be saturated at \( \pm 1 \). Also the result of multiplication and division will be rounded toward the closest 16-bit value. The division operator will also only take positive arguments, this is due to that division with positive factors requires much less computational power than a general division, reassuring that factors are positive is handled by conditions enveloping divisions in the code.

5.4.3 Implementation of Matrices

Since there is no matrix equivalent in C, the matrices were initially coded as 2-dimensional arrays instead. The columns of the matrices will be regarded as an array within an array representing a row.

Example 5.2: Implementation of matrices

The following matrix: 
\[
A = \begin{bmatrix}
a_{11} & a_{12} & a_{13} & a_{14} \\
a_{21} & a_{22} & a_{23} & a_{24} \\
a_{31} & a_{32} & a_{33} & a_{34} \\
a_{41} & a_{42} & a_{43} & a_{44}
\end{bmatrix}
\]

will have an array-implementation:
\[
A[4][4] = [a_{11}, a_{21}, a_{31}, a_{41}, a_{12}, a_{22}, a_{32}, a_{42}, a_{13}, a_{23}, a_{33}, a_{43}, a_{14}, a_{24}, a_{34}, a_{44}]
\]

To avoid memory leaks and other problems, matrices and the functions of them will be coded statically, so there will be several functions of each mathematical operator due to the different sizes of the matrices. Multiplications, subtractions and additions will be handled by for-loops iterating through the rows and columns of the matrices. There is also one matrix inversion calculated in the algorithm, but due to it being a 2x2 matrix we can use the special case for 2x2 matrices rather than a time consuming general algorithm:

\[
\text{if } A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \text{ then } A^{-1} = \frac{1}{\text{det}(A)} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix} \tag{5.2}
\]

Since the EKF algorithm uses matrix operations and mostly 4x4 matrices to calculate the estimated variables it is very computational heavy. For example a multiplication between two 4x4 matrices needs 64 multiplications and 48 additions and in addition each value has to be retrieved by two pointers and data stored in a position designated by two pointers, resulting in a lot of extra time calculating
pointers. An algorithm on this form would take too much time to compute in a system like this where performance of the DSP is limited and the demands on sample time is high. Further, one can see that many of the matrices used for computations are sparse, resulting in many of the calculations performed always will yield zero as a result.

Due to the observations made, the algorithm was recoded to be more time efficient. Matrices were recoded as vectors to save pointer arithmetics and operations identified as always yielding zero as a result were excluded.

Example 5.3: New Implementation of matrices

The following matrix: \( A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \)

will have an array-implementation:

\[ A[16] = [a_{11}, a_{12}, a_{13}, a_{14}, a_{21}, a_{22}, a_{23}, a_{24}, a_{31}, a_{32}, a_{33}, a_{34}, a_{41}, a_{42}, a_{43}, a_{44}] \]

The effects of this reprogramming can be seen in Table 5.5. Not all operations are included, only those that have been changed in the recoding. As seen, there is a huge reduction in the operations and pointers needed. Due to the recoding the computational time of the algorithm was reduced from about 120 \( \mu s \) to about 60 \( \mu s \) on the processor running at 60 MHz. However, the code itself is longer and harder to read due to all the special coding of operations.

<table>
<thead>
<tr>
<th>Algorithm:</th>
<th>General</th>
<th>Special coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiplications:</td>
<td>336</td>
<td>136</td>
</tr>
<tr>
<td>Additions:</td>
<td>278</td>
<td>86</td>
</tr>
<tr>
<td>Substractions:</td>
<td>18</td>
<td>10</td>
</tr>
<tr>
<td>Pointers used:</td>
<td>40</td>
<td>14</td>
</tr>
</tbody>
</table>
Example 5.4: Matrix Multiplication.

Consider the following matrix multiplication:

\[
C = A \cdot B \quad \text{where} \quad A = \begin{bmatrix}
0 & a_{12} & a_{13} \\
a_{21} & a_{22} & 0 \\
a_{31} & 0 & a_{33}
\end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix}
b_{11} & 0 & b_{13} \\
b_{21} & b_{22} & b_{23} \\
0 & b_{32} & 0
\end{bmatrix}
\]

In the general algorithm it was coded according to:

```c
for(i=0;i<4;i++) {
    for(j=0;j<4;j++) {
        for(k=0;k<4;k++) {
            C[i][j] = itradd(C[i][j], itrmultr(A[i][k], B[k][j]));
        }
    }
}
```

In the special coded algorithm the same multiplication would be:

```c
C[0] = itrmultr(a_{12}, b_{21});
C[1] = itradd(itrmultr(a_{12}, b_{22}), itrmultr(a_{13}, b_{32}));
C[2] = itrmultr(a_{12}, b_{23});
C[3] = itradd(itrmultr(a_{21}, b_{11}), itrmultr(a_{22}, b_{21}));
C[4] = itrmultr(a_{22}, b_{22});
C[5] = itradd(itrmultr(a_{21}, b_{13}), itrmultr(a_{22}, b_{23}));
C[6] = itrmultr(a_{31}, b_{11});
C[7] = itrmultr(a_{33}, b_{32});
C[8] = itrmultr(a_{31}, b_{13});
```

One can see the difference between the two implementations quite clearly, the general algorithm is easier to read and it has more operations. The special coded operation is fast, but the code is harder to read.

The code itself will be separated into two source files with associated header files. One file will contain all the code for the algorithm, declarations of parameters and handling input and output. The second file will handle all the matrix operations.
Chapter 6

Results

Results of simulations and from the test bench are presented in this chapter as plots.

6.1 Results of Simulations

Different simulations have been run for each of the observers to evaluate their performance regarding convergence, robustness to parameter errors, robustness regarding uncertainties in current measurement and sample times.

The start-up behavior is tested for each observer. To test their global convergence, the start sequence is a step in reference angular velocity with an initial estimation error in angular position of $\pi$. Also a given sequence is run for each observer in simulations with change in speed reference from 1000 electric rad/s to 500 electric rad/s at 0.5 seconds. At 0.75 seconds a load is engaged and at 1 second the speed reference is changed from 500 electric rad/s to 1000 electric rad/s. This is to test different dynamic behaviors of the observers. Worth noting is that in the motor control algorithm developed by SAAB there are no steps in reference angular velocity, but they are ramped over a small period of time instead. It can be seen in the following plots that the controller have large gains, visible in steps in reference, which are of no problem in the hardware implementation due to reference in angular velocity being ramped.

A few parameters are fixed in order to evaluate the robustness in all the simulations: $\hat{R} = 0.8 \cdot R$, $\hat{L} = 1.1 \cdot L$, $\hat{\Phi} = 0.9 \cdot \Phi$, also an uniform distributed random noise is added to the currents in the a-b-c frames with a maximum amplitude of 1 A and a frequency of 200-600 Hz.

6.1.1 EKF, Simulink Model

Figures 6.1, 6.2, 6.3 and 6.4 shows the start up from a large error in estimated rotor position and Figures 6.5, 6.6, 6.7, 6.8, 6.9 and 6.10 illustrate the estimation under the running sequence. The covariance matrices used have the following values: $Q = \text{diag}[1, 1, 100, 1]$ and $Z = \text{diag}[1, 1]$. 
Figure 6.1. Angular velocity at start up with an initial error of $\pi$ in position.

Figure 6.2. Error in estimated angular velocity at start up with an initial error of $\pi$ in position.

Figure 6.3. Angular position at start up with an initial error of $\pi$ in position.
6.1 Results of Simulations

Figure 6.4. Error in estimated angular position at start up with an initial error of $\pi$ in position.

As seen in Figures 6.1 and 6.3 the EKF algorithm successfully starts up from an error of $\pi$ in initial position. Due to the initial error in estimated angular position, a reversal in speed occurs at the startup. The error in estimated angular velocity approaches 0 and the estimation in angular position have a small offset. This offset is due to the induced perturbation according to (4.16).

Figure 6.5. Angular velocity during the given sequence.
Figure 6.6. Error in estimated angular velocity during the given sequence.

Figure 6.7. Angular position at the step in velocity reference at 0.5 seconds.

Figure 6.8. Angular position when load is engaged at 0.75 seconds.
6.1 Results of Simulations

Figure 6.9. Angular position at the step in velocity reference at 1 second.

Figure 6.10. Error in estimation of angular position under the given sequence.

Figures 6.5 and 6.6 show how the EKF algorithm estimates the velocity and its effects in transients. The algorithm follows the measured velocity well in transients and the effect of the engaged load on the velocity is eliminated quickly. The algorithm can handle large loads being enabled with 5 Nm used in this simulation. Figures 6.7, 6.8, 6.9 and 6.10 illustrates the effects of the transients in velocity on the estimated position. Figure 6.10 shows that the algorithm has some problems estimating the rotor position with a reduction in reference velocity, but the error in position is although eliminated quickly. There are no noticeable errors in the estimated position at the engage of load or an increase in reference velocity.

6.1.2 EKF, C-Code Model

In this Section, the scenarios are as in the former section, but here run through a fixed point S-function generated by C-code rather than a Simulink model of
the EKF algorithm. The C-coded algorithm should behave as the algorithm implemented in Simulink if it is coded correctly and have similar covariance values. Figures 6.11, 6.12, 6.13 and 6.14 illustrate the behavior at start-up from an incorrectly initial estimation of angular position and Figures 6.15, 6.16, 6.17, 6.18, 6.19 and 6.20 illustrates the result of the sequence described in Section 6.1. The following simulations shows that the behavior is indeed similar. The used covariance matrices have the following scaled values: $Q = \text{diag}[200, 200, 1, 1] \cdot 2^{-15}$ and $Z = \text{diag}[400, 400] \cdot 2^{-15}$.

**Figure 6.11.** Angular velocity at start up with an initial error of $\pi$ in position.

**Figure 6.12.** Error in estimated angular velocity at start up with an initial error of $\pi$ in position.
6.1 Results of Simulations

Figure 6.13. Angular position at start up with an initial error of π in position.

Figure 6.14. Error in estimated angular position at start up with an initial error of π in position.

The results of this start-up displayed in Figures 6.11, 6.12, 6.13 and 6.14, show a similar behavior for the fixed point algorithm compared to that of the result of simulation by the Simulink EKF model. There are some differences, which likely comes from the different choice of covariance matrices, different choice of perturbation parameter in the estimated q-axis current and fixed point arithmetics.
Figure 6.15. Angular velocity under the given sequence.

Figure 6.16. Error in estimated angular velocity under the given sequence.

Figure 6.17. Angular position at the step in velocity reference at 0.5 seconds.
6.1 Results of Simulations

Figure 6.18. Angular position when load is engaged at 0.75 seconds.

Figure 6.19. Angular position at the step in velocity reference at 1 second.

Figure 6.20. Error in estimated angular position under the given sequence.
The result of this sequence simulation illustrated in Figures 6.15, 6.16, 6.17, 6.18, 6.19 and 6.20 yet again displays a similar behavior to the EKF Simulink model. Worth noting is that the offset caused by the perturbation of the estimation of q-axis current is smaller in the C-coded observer, Figure 6.20, than in the Simulink modeled observer, Figure 6.10.

6.1.3 PLL

In these simulations, the poles of the observer was chosen as $\rho = 250$. As the PLL algorithm can not perform well at lower speeds it was allowed to start at 1000 electrical rad/s. Figures 6.21, 6.22, 6.23 and 6.24 shows the start up from a large error in position, and Figures 6.25, 6.26, 6.27, 6.28, 6.29 and 6.30 illustrates the sequence described in Section 6.1.

**Figure 6.21.** Angular velocity at start up with an initial error of $\pi$ in position.

**Figure 6.22.** Error in estimated angular velocity at start up with an initial error of $\pi$ in position.
6.1 Results of Simulations

Figure 6.23. Angular position at start up with an initial error of $\pi$ in position.

Figure 6.24. Error in estimated angular position at start up with an initial error of $\pi$ in position.

As seen in Figures 6.21, 6.22, 6.23, 6.24, the PLL algorithm can start up from an unknown starting position. However, the effects of the error in estimation is rather large. In Figure 6.24 one can see a so called cycle slip, the error in estimated angular position is drawn from its convergence area for $\theta = 0$ and it converges toward the next convergence point.
Figure 6.25. Angular velocity under the given sequence.

Figure 6.26. Error in estimated angular velocity under the given sequence.

Figure 6.27. Angular position at the step in velocity reference at 0.5 seconds.
6.1 Results of Simulations

Figure 6.28. Angular position when load is engaged at 0.75 seconds.

Figure 6.29. Angular position at the step in velocity reference at 1 second.

Figure 6.30. Error in estimated angular position under the given sequence.
Examining Figure 6.25 one can see that even if the algorithm is given the correct starting position and velocity it shows an oscillating behavior. Also the load that is applied at 0.75 seconds is of 3 Nm, which was the largest load the algorithm could handle at that speed. Figure 6.30 indicates that the algorithm has problems estimating the angular position during a decrease in reference velocity. However, no such issues in the estimation of position is shown for the engaging of load or an increase in reference velocity.

6.1.4 LF-Inj

The chosen injected carrier current reference was given the following parameters: $I_{cd} = 0.5 \text{A}$, $\omega_{cd} = 2\pi \cdot 200 \text{rad/s}$. The proportional observation constant was chosen as $k_p = 5$ and the integral observation constant is $k_i = 10$. As the algorithm is not globally convergent, one start up was performed at the correct position, displayed in Figures 6.33, 6.34, 6.35 and 6.36, and one with an initial error of $\pi$, displayed in Figures 6.31 and 6.32, to illustrate the different start ups. Also a sequence similar to the one used in previous algorithms is used, but the reference velocities are set to 50 to 25 to 50 electric rad/s and the engaged load is of magnitude 0.5 Nm with results displayed in Figures 6.37, 6.38, 6.39 and 6.40.

![Figure 6.31. Angular velocity at start up with an initial error of $\pi$ in position.](image)
6.1 Results of Simulations

Figure 6.32. Angular position at start up with an initial error of $\pi$ in position.

The position error, displayed in Figure 6.32, quickly approaches the wrong convergence point $0.5\pi$ as this algorithm has no form of correction to break that equilibrium point. As one can see in Figure 6.31 the velocity does not approach zero as predicted in Section 4.1. This is due to that $i^{ref}_d$ is not set to zero in this algorithm, instead the injected current in the d-axis will generate a torque with the frequency of the injection signal which is visible.

Figure 6.33. Angular velocity at start up from the correct position.
Figure 6.34. Error in estimated angular velocity at start up from the correct position.

Figure 6.35. Angular position at start up from the correct position.

Figure 6.36. Error in estimated angular position at start up from the correct position.
6.1 Results of Simulations

The algorithm can start up successfully from the known position, however it still has a rather poor dynamic operation and the settling time for the velocity is rather long. The estimated angular position suffers from a rather large offset, as can be seen in Figure 6.36. The reason for this offset has not yet been discovered, but it appears to be quite dependent on errors in estimated motor parameters.

Figure 6.37. Angular velocity under the given sequence.

Figure 6.38. Error in estimated angular velocity under the given sequence.
Figure 6.38 illustrates the poor dynamic properties of the LF-Inj algorithm, the error in estimated angular velocity is large compared to the reference in velocity. It is sensitive to the engagement of loads illustrated by the application of just a small load of 0.5 Nm. The offset in estimated angular position, as seen in Figure 6.40, is dependent on estimated velocity and applied load.

6.1.5 Combined PLL, LF-Inj
The two modes of observation in this combined observer utilizes the parameters from the above sections and the limit angular speeds are chosen as: $\omega_{\Delta_1} = 100$ rad/s and $\omega_{\Delta_2} = 150$ rad/s. Transition from LF-Inj to PLL was unfortunately never achieved with this combined observer. Reasons for this is a combined result of that the PLL observer cannot be run offline, as it has to do for a short while in
the transition and the oscillation behavior when the observer is engaged. However, it is possible to make the transition from the PLL mode of observation to the LF-Inj mode of observation. Figures 6.41, 6.42, 6.43 and 6.44 illustrate the transition from the PLL mode of observation to the LF-Inj mode of observation.

One can also see that the injected signal into the motor, that the LF-Inj method need, disturbs the PLL algorithm in the transition period. Outside this transition region, the combined observer operates as the original PLL or LF-Inj observer, in their respective region of operation.

Figure 6.41. The estimated angular velocity during the transient from PLL observer to the LF-Inj observer.

Figure 6.42. Error in estimated angular velocity during the transient from PLL observer to the LF-Inj observer.
Between the velocities 150 and 100 electric rad/s both observers are engaged, the PLL feeds information to the controller in this range and one can see the effects of the injected signal in the ripple of the estimated and measured velocity in Figures 6.41 and 6.42. When LF-Inj feeds the controller at below 100 rad/s the effect of the injected signal disappears. The effects of the injected signal can also be seen in the estimated position in Figures 6.43 and 6.44. There is a short jump in estimated position when the LF-Inj algorithm starts feeding the controller. The jump is due to the LF-Inj has been estimating offline and built up an error in estimated position due to errors in parameters, although when it starts feeding the controller the error in estimated angular position approaches an offset similar to earlier LF-Inj simulations.

Figure 6.43. The estimated angular position during the transient from PLL observer to the LF-Inj observer.

Figure 6.44. Error in estimated angular position during the transient from PLL observer to the LF-Inj observer.
6.1.6 Comparison of Simulation Results

As seen in Section 6.1.1 to 6.1.5, the Simulink implementation of the EKF algorithm has been the one performing with the best result of the investigated observers, showing good dynamic behavior, global convergence and good operations in all ranges of angular velocity. The EKF algorithm needs no transitions between modes of observation as the combined PLL and LF-Inj observer does and there is no need to inject any carrier currents into the motor. However, the computational time of the EKF algorithm is much longer than those of the PLL and LF-Inj algorithms, due to the former using matrices for computing the Kalman gains.

From these simulations results the EKF algorithm was chosen to be C-coded and then tested on hardware. The results of simulations of the C-coded EKF algorithm can be seen in Section 6.1.2 and results of the hardware implementation can be seen in Section 6.2.

6.2 Results of Implementation

The implemented EKF can track the angular position and angular speed of the rotor quite well under some modes of operation as shown in Figures 6.45, 6.46 and 6.47 where the algorithm has been tested offline.

Online, however, the observer has only worked under certain conditions. The angular speed must be over a certain angular velocity and also the load torque has to be rather large to provide better current acquisitions. The results of online operations are displayed in Figures 6.48, 6.49 and 6.50.

The measured velocity is differentiated from the measured angular position of the rotor provided by the resolver. The differentiated angular velocity is then filtered in a FIR filter. The measured angular position is acquired directly from the resolver.

The used covariance matrices have the following scaled values:

\[ Q = \text{diag}[200, 200, 1, 1] \cdot 2^{-15} \quad \text{and} \quad Z = \text{diag}[200, 200] \cdot 2^{-15}. \]

![Figure 6.45. The estimated angular velocity during offline estimation.](image)
Within the speed range where angular velocity and position can be estimated, illustrated in Figure 6.45, the EKF algorithm tracks the angular velocity and position rather well, although there is a small offset due to parameter and numerical errors. However, at speeds outside this range there will be numerical errors in the filter calculations and the algorithm will estimate angular velocity and position incorrectly.
6.2 Results of Implementation

Figure 6.48. The estimated angular velocity during sensorless control.

Figure 6.49. Error in estimated angular velocity during sensorless control.

Figure 6.50. The error in estimated position during sensorless control.
As seen in Figures 6.48, 6.49 and 6.50, the robustness of the closed-loop system eliminates the offsets in estimated position and velocity caused by parameter errors. The sensorless control was also only achieved in stationary or close to stationary conditions, the reason for this is yet again the numerical errors appearing in the filter calculations where the resolution of the integers is too low, yielding often 0 or 1 as a result of calculations, as described in Section 5.4.1.
Chapter 7

Discussion and Future Work

Two estimation methods have been studied in this thesis, and one has also been implemented in a test bench. A few conclusion can be drawn and discussed, also some proposals on future enhancements are presented.

7.1 Discussion

The implementation in Simulink of the combined PLL and LF-Inj observer works rather poorly in several aspects. The PLL has some flaws with poor operation at low velocities and oscillating behavior when engaged, making it a bad candidate for a combined observer as transitions are hard to make. The LF-Inj works well in its operating range, but lacks the desired global convergence. This design of an observer is not currently recommended.

As presented in the result chapter, estimation of angular position and angular speed of a PMSM by an EKF algorithm shows promise in the Simulink simulations. However, the algorithm implemented in test bench is by no means a finished working product, mainly due to the lack of resolution of the 16-bit integers, especially during filter calculations.

The EKF algorithm, augmented with the modification to guarantee global convergence have in simulations been a very robust observer regarding parameter errors and uncertain current measurements. The algorithm can operate in all the speed ranges of the motor and start up from any unknown starting position.

In the test bench, the EKF algorithm has been able to track angular position and angular speed successfully for speeds within the region of 500 to 2000 electrical rad/s, although it has only operated as intended in sensorless feed-back when the speed has been stationary, or close to stationary within this speed range. The reason for this is due to the implementation with 16-bit integers in the DSP, all the calculations involving covariance matrices are in the order of a scaling function lower than the scaled variables and parameters. The calculations involving these matrices suffers from relatively large rounding errors, that propagates through the rest of the algorithm.
The simulations where the EKF algorithm is implemented in 32-bit fixed point system displays much more stable operation than the 16-bit fixed point algorithm used in tests. A 32-bit fixed point implementation in hardware would probably perform better due to the 32-bit system having a $2^{16}$ times better resolution on integers, which would remove the large rounding errors in filter computations.

Also, the EKF algorithm is very computationally heavy, it forces the control algorithm to work at a sample frequency of 5kHz instead of its intended 10kHz, resulting in worsened dynamics of the controller, less ideal voltage actuation from the inverter bridge and larger dead times in the whole EMCU.

Another fact to mention is that the motor used for this thesis is quite different from the motors used by the authors of the articles used for this thesis. The motor used in this thesis is a large and powerful motor that operates with many permanent magnets, high torque, voltages and currents. The motors used in the scientific articles have been of smaller caliber so it is possible that observer algorithms developed for these motors do not suit the motor used in this thesis.

### 7.2 Future Work

There are a few things that can be done to enhance the performance of the EKF algorithm. The main issue with the fixed point algorithm is the low resolution of the matrix operations it suffers from at this point. Increasing the resolution of integers in the fixed point implementation from 16-bits to 32-bits would yield a $2^{16}$ better resolution and it would allow the matrix operations to perform without the large rounding errors that the 16-bit implementation currently has. By allowing a larger range of possible values in the algorithm from 16-bits to 32-bits would also allow a wider and better tuning of filter parameters as the scaling functions does not allow much room for filter tuning at this time. The 16 bit implementation could perhaps be improved as well by scaling the covariance matrices differently, yielding larger scaled values. The relatively large rounding errors might be decreased by such a solution.

The elements in the covariance matrices used in the implementation has not been tuned, values used in this thesis has been tested ad hoc until the observer estimates in an acceptable way. However, the 16-bit implementation does not allow much tuning, but in a 32-bit implementation one could most likely achieve better operation by properly tune these covariance matrices according to Section 4.2.

Other improvements would be a better control algorithm for the application of voltages in the stator, yielding a current with less problems in zero-crossovers. This improvement is already made in the new version of the EMCUs manufactured by SAAB, however, they are not installed in a test bench to be available for testing at this time. Further improvements would be to either fit another DSP into the EMCU dedicated to the observer, or install a faster DSP to achieve the intended sample frequency of 10 kHz.
Bibliography


## Appendix A

### Notations

#### A.1 Nomenclature

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u$</td>
<td>V</td>
<td>voltage</td>
</tr>
<tr>
<td>$i$</td>
<td>A</td>
<td>current</td>
</tr>
<tr>
<td>$L$</td>
<td>H</td>
<td>inductance</td>
</tr>
<tr>
<td>$R$</td>
<td>Ω</td>
<td>resistance</td>
</tr>
<tr>
<td>$\omega$</td>
<td>rad/s</td>
<td>angular velocity of the rotor</td>
</tr>
<tr>
<td>$\theta$</td>
<td>rad</td>
<td>angular position of the rotor</td>
</tr>
<tr>
<td>$\vartheta$</td>
<td>rad</td>
<td>difference between measured and estimated rotor position</td>
</tr>
<tr>
<td>$\Phi$</td>
<td>Wb</td>
<td>permanent magnet flux of the rotor</td>
</tr>
<tr>
<td>$N$</td>
<td>-</td>
<td>the amount of pole pairs of the rotor</td>
</tr>
<tr>
<td>$J$</td>
<td>kgm$^2$</td>
<td>moment of inertia on the rotor</td>
</tr>
<tr>
<td>$\tau$</td>
<td>N</td>
<td>torque</td>
</tr>
<tr>
<td>$D$</td>
<td>Ns/rad</td>
<td>constant of friction affecting the rotor</td>
</tr>
<tr>
<td>$Q$</td>
<td>-</td>
<td>covariance matrix regarding the uncertainty in states</td>
</tr>
<tr>
<td>$Z$</td>
<td>-</td>
<td>covariance matrix regarding errors in measurements</td>
</tr>
<tr>
<td>$P$</td>
<td>-</td>
<td>covariance matrix regarding errors is estimated variables</td>
</tr>
<tr>
<td>$\Box_{k</td>
<td>k-1}$</td>
<td>-</td>
</tr>
<tr>
<td>$\Box_{k</td>
<td>k}$</td>
<td>-</td>
</tr>
<tr>
<td>$\Box_k$</td>
<td>-</td>
<td>denotes the k-th time step</td>
</tr>
<tr>
<td>$\Box_d$</td>
<td>-</td>
<td>denotes a quantity in the d-axis</td>
</tr>
<tr>
<td>$\Box_q$</td>
<td>-</td>
<td>denotes a quantity in the q-axis</td>
</tr>
<tr>
<td>$\Box_\gamma$</td>
<td>-</td>
<td>denotes a quantity in the $\gamma$-axis</td>
</tr>
<tr>
<td>$\Box_\delta$</td>
<td>-</td>
<td>denotes a quantity in the $\delta$-axis</td>
</tr>
<tr>
<td>$\hat{\Box}$</td>
<td>-</td>
<td>denotes an estimated variable</td>
</tr>
<tr>
<td>$\hat{\Box}$</td>
<td>-</td>
<td>denotes difference between real and estimated quantity</td>
</tr>
<tr>
<td>$T$</td>
<td>s</td>
<td>time period</td>
</tr>
</tbody>
</table>
A.2 Abbreviations

PMSM  Permanent Magnet Synchronous Motor
SMPMSM  Surface Mounted Permanent Magnet Synchronous Motor
IMPMSM  Internal Mounted Permanent Magnet Synchronous Motor
EKF  Extended Kalman Filter
PLL  Phase Locked Loop
LF-Inj  Low Frequency Injection
EMCU  Electric Motor Control Unit
DSP  Digital Signal Processor
PWM  Pulse Width Modulation
SVM  Space Vector Modulation
EMF  Electro Motive Force
Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/

© Fredrik Petersson