NTNU – Trondheim
Norwegian University of
Science and Technology

Modelling and Control of High Performance Medium Voltage Drives

- Simulation and analysis of the Programmed Modulation strategy

Roger Enes

Master of Science in Electric Power Engineering
Submission date: June 2012
Supervisor: Tore Marvin Undeland, ELKRAFT

Norwegian University of Science and Technology
Department of Electric Power Engineering

Problem Description

Motor drives are continuously evolving towards becoming more energy efficient, getting a higher power density and to withstand more harsh environments. R & D engineers are working hard to reduce investment costs while maintaining a high product quality, to be competitive in the market. An attractive motor drive system should be capable of handling demanding dynamic operations in a controlled manner. Fast.

The switching losses are increasingly important as the trend points towards using higher dc-bus voltage levels in high power, medium voltage inverters. The losses caused by switching in IGBT devices increase as the operational voltage level increase. This necessitates the need for a modulation strategy that allows a low switching frequency without causing too much harmonic distortion in the phase currents. Lowering the switching frequency will increase the power density since the IGBTs then will generate less heat due to losses, thus be able to conduct a higher current.

Based on a literature study is Programmed Modulation, with Synchronous Optimal Modulation patterns, a modulation strategy that can operate at low switching frequencies, without distorting the load current to unacceptable levels. This makes it a very interesting modulation technique for medium voltage drives applications. Programmed Modulation applies optimized pulse width modulated patterns to determine the IGBT switching instants, this requires a novel approach to achieve dynamic control. Also undesirable current transients can occur during a pattern change in such a system if not presented countermeasures are done.

The concept of this modulation strategy is to be investigated and tested trough simulations in MATLAB simulink. To do this a simulation model must be built which operates by applying pre-calculated switching-patterns, the model should be thoroughly explained to make it easy for further model development and research.

Summary

In the master thesis the concept of Programmed Modulation is investigated for motor drives with Three-Level NPC inverter. Programmed Modulation operates with pre-calculated PWM switching-patterns, which enables the facility of off-line optimization of the converter switching-instants. The Optimization objectives are several, two commonly known are Selective Harmonic Elimination and Synchronous Optimal Modulation. The latter optimization focuses on generating a switching-pattern that will reduce the phase current harmonics. A reduction of harmonic components in the phase currents, means that a reduction in switching frequency is possible.

The switching loss component, compared to the total loss in IGBTs increases as the voltage level increase. A reduction in switching losses opens for an increase in the nominal current limit in IGBTs. Hence, reductions in switching frequency gives an increase in power density.

Conventional current control strategies cannot be used in a switching-pattern based drive system without sacrificing the optimality of the applied switching-pattern. A novel approach is therefore required to obtain dynamic control. For this has the stator flux trajectory control method been chosen and tested. A simulation model, specially built for Programmed Modulation, is proposed in this thesis. The model has a Stator Flux Trajectory Controller (SFTC) that calculates manipulation of the optimal switching-pattern, the manipulations are added to the original optimal switching-pattern to control the actual stator flux in the drive system. This SFTC is also used to eliminates, fast, deviations between the actual stator flux and a calculated optimal stator flux. This effectively eliminates the currents transient that otherwise could arise after a switching-pattern exchange, due to mismatch between optimal flux and actual flux trajectory.

The modulation strategy has been simulated, results shows that fast dynamic control is obtained by controlling the α- and β-components of the stator flux, in rotor field oriented coordinates. Combined with the use of Synchronous Optimal Modulation switching-patterns is this a very promising modulation strategy that have the required qualities for medium voltage drives. The simulation model needs further development, suggestions are given in the further work section.

Summary Given in Norwegian

Sammendrag

I denne mastergradavhandlingen er modulasjonsmetoden Programmerbar Modulasjon undersøkt for motorstyring med trenivå NPC omformer. Programmerbar Modulasjon anvender forhåndskalkulerte PWM svitsjemønster. Dette kan utnyttes ved å optimalisere svitsjemønstrene slik at de eliminerer utvalgte overharmoniske strømkomponenter, eller minimalisere en objektfunksjon. Til dette kan "Weighted Total Harmonic Distortion" (WTHD0) brukes, hvor en da oppnår et PWM svitsjemønster som genererer minimalt med overharmoniske komponenter i fasestrømmene. Den sistnevnte typen heter Synkront Optimalt Modulasjonsmønster. En reduksjon av de overharmoniske komponentene i fasestrømmen åpner for å redusere svitsjefrekvensen i omformeren.

En reduksjon i svitsjefrekvens vil gi en reduksjon i svitsjetap i svitsjeelementene (IGBTer), spesielt i mellomspenningsomformere hvor disse tapene er en større del av de totale tapene i svitsjeellementen. En reduksjon i svitsjetap muliggjør en økning i driftsstrøm. Altså, effekttettheten i omformeren øker.

Konvensjonelle styresystemer som er basert på å styre strømkomponentene i toaksesystemet kan ikke benyttes i et system som operer med forhåndskalkulerte svitsjemønster. Det ville ført til uakseptable forandringer i svitsjemønstrene og dermed miste optimaliseringen. En utradisjonell metode som kontrollere statorfluksen i den styrte maskinen er valgt for å oppnå hastighet og moment styring. En simuleringsmodell er laget, spesielt designet for å kunne teste ut Programmerbar Modulasjon. Den presenterte modellen benytter statorflukskontrollprinsippet via en kontroller kalt Stator Flux Trajectory Controller (SFTC). Denne kontroller har vist seg å være veldig effektiv til å eliminere uønskede strømtransienter som kan oppstå ved utbytting av svitsjemønster.

Modulasjonsstrategien har gjennom simuleringer i den presenterte modellen vist å gi en rask dynamisk respons. Rask dynamisk kontroll sammen med muligheter for om å senke svitsjefrekvensen ved a anvende Synkrone Optimal Modulasjonsmønster, er dette en veldig lovende modulasjonsmetode for motorstyringer i mellomspenningsapplikasjoner. Simuleringsmodellen trenger videre utvikling, forslag til dette er gitt i seksjon for videre arbeid.

Preface

This report is my final result after a semester consisting of literature searching, programming, Simulink modelling and a lot of challenges. Many solutions have been developed and almost an equal number has failed, it was "survival of the fittest" solution. The master thesis work has given me new fields of interest within Electric Power Engineering and i am very grateful that I was assigned with this thesis. Its work has given me valuable personal growth and knowledge within Programming, PWM modulation techniques and induction machine physics. This would not have been the same pleasant experience without the support of several people. Therefore would i like to thank those who have contributed in this project.

My office fellow students which i now consider as my friends have given their support through technical advice, profitable discussions, multiple view points and many memorable social times.

I must express my gratitude and admiration to my co-supervisor Dr.Eng. Roy Nilsen in Wärtsilä, who has helped me several times by solving dead-end problems. His ability to understand and solve problems through fundamental thinking, in no time, has just been fascinating.

In addition, I thank my family for moral support and encouragement throughout the project. I would also express my love to my girlfriend, who has given me several good advices on how to distribute time and also for her loving support.

At last, I would like to sincerely thank my supervisor Prof. Tore M. Undeland. who, with his lifelong experience within Power Electronics, provided inspiration and encouragement. He has given me valuable inputs both technical and administrative throughout the master thesis and it has been an honor to be his master student.

Trondheim 19.06.2011

Roger Enes

Roger Enes

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

Suppliers of offshore rigs and vessels systems are on an ever-ending struggle to offer the most energy efficient product with the lowest weight and volume. This highly includes the electric power system discipline and electric drives. An increase in Weight and volume equals an increase in cost. This correlation does not only apply for offshore installations. One improvement that is mutual when improving the electric systems efficiency, weight and size is to increase the voltage level.

In medium voltage drives, as the voltage level increases losses due to switching increasingly contribute to the total of the switching device(IGBTs)loss. In fact, switching loss is the major loss component in IGBTs for medium voltage applications [6]. Switching loss is emitted as heat from the IGBT and as a result, a current limiting factor will be imposed since heat build-ups can destroy the switching device, if upper limits are exceeded. Reference [6] states that a reduction in switching frequency from 800 to 200 Hz more than doubles the current capabilities of an EUPEC 6.5 kV IGBT. Clearly should modulation strategies that offers low operational switching frequencies be favourable in Medium voltage drives.

1.2 Motivation

Programmed Modulation is a modulation strategy that operates with pre-calculated switching instants defined by an objective function [6]. This objective function can be to eliminate selected harmonic components, or minimization of harmonic components that contribute to losses in an induction machine. Synchronous Optimal Modulation(SOM) is based in the latter objective, this means that switching-patterns are optimized to minimize current ripple in the phase currents. It is promised that Programmed Modulation with Synchronous Optimal Modulation will allow a very low switching frequency in the converter, due to the reduction in phase current harmonics that this strategy offer [2]. In medium voltage drives these-features will be of high interest. The allowance of a reduction in switching frequency will increase the power density of the converter.

The value of a modulation system that offers the above-mentioned qualities is just held back by the dynamic performance, robustness and complexity when considering the total system. The dynamic control of a drive system that use pre-calculated switching-patterns has previously been regarded as to slow [18], but due to recent development this has changed. Stator flux trajectory control is a technique used to achieve speed and torque control based on-line manipulation of the pre-calculated switching patters. This technique is promised to give a fast dynamic response in induction machine drives [4].There is another technique based on stator current trajectory control, but it depends on motor parameters and load conditions [2], due to that, is it not considered any further here in this master thesis. The focus is on the Stator flux trajectory control method which is explored by simulation in a presented simulation model, specially designed for switching-pattern based modulation. The model is thoroughly explained for further development and investigation.

1.3 Scope of work

Programmed Modulation is an unconventional modulation strategy, therefore has a theory chapter been included. It explains how to achieve pre-calculated pulse patterns that inherent the desirable harmonic content quality. Also it contains the stator flux trajectory control approach theory, that is used to eliminate unwanted current transients that can occur during switching-pattern exchanges, and achieve dynamic control.

The presented model can be used to test both SOM based pre-calculated patterns or the Selective Harmonic Elimination based patterns. Or any other pre-calculated patterns within some constraints explained in chapter 3. The pre-calculated switching-pattern based model is mainly built-up by the use of "user defined functions" in MATLAB, written in m-files. This is due to the requirement of processing pre-calculated PWM patterns which is not something regular modelling blocks can handle.

Some system assumptions, see below, has been made to limit the workload of constructing an operational model.

System Assumptions

I *Measuring induction machine rotor and stator flux*
Internal induction machine fluxes are assumed measured. Therefore is the machine parameters estimation model, normally used to estimate machine flux values, not included. This decision has given more time to focus on the modulation system.

II *Converter topology*
The Three-Level Neutral-Point-Clamped(NPC)topology has been chosen, which is one of the most commonly used converter topology [6].

III *DC-bus and neutral point balancing*
The DC bus voltage and the neutral point potential are assumed ideal in the model. In reality has the Three-Level NPC inverter an intrinsic neutral point balancing mechanism that tend the average neutral point voltage error to assume a zero value [8]. However during a transient, e.g. change in modulation index, excursions of the neutral point voltage can occur and cause over-voltages [9]. Reference [9] presents a technique for fast elimination of the neutral-point potential error by exploiting the existence of two redundant sub-bridges in a Three-Level NPC inverter, for a Synchronous Optimal Modulation drive system.

Chapter 2

Theory

The theory chapter focus on Programmed Modulation and how pre-calculation of switching-patterns can be utilized. Theory surrounding techniques that allow pre-calculated switching-patterns to be used in Motor Drives in an effective way is also presented. However, an performance indicator needed in pattern optimization is WTHD0, its features are explained first.

2.1 WTHD and Harmonic Losses in Induction Machines

Losses in an induction machine due to presence of harmonic components of higher order than the fundamental component, is not divided evenly over the harmonic spectra. The lower spectre of the harmonic components are contributing more to losses then the higher spectre of the harmonic components [1]. Higher order of harmonic components are cheaper to filter out because the required smoothing filter required is smaller in size, weight and therefore also cost [5].

2.1.1 WTHD

A performance indicator commonly known as Weighted Total Harmonic Distortion(WTHD), weights the lower frequency spectra of harmonic components more heavily than the high spectra of the harmonic components[1]. How WTHD can be derived to indicate the presence of the harmonic spectre that contribute the most to losses due to harmonic content in an induction machine, hence represent losses, now will be discussed in the following.

A simple load-model for an induction motor is shown in equation 2.1 [1].

$$I_n \cong \frac{U_n}{n\omega_1 L_\sigma} \tag{2.1}$$

Constant motor parameters are here assumed and the inductance L_σ, defined in equation 2.2, represents the induction motor inductances, where L_m is the mutual inductance, L_{ls} is the stator leakage inductance and L_{lr} is the rotor leakage inductance. WTHD can be derived by normalizing the current harmonic distortion HD_i with respect to $U_1/\omega_1 L_\sigma$, which corresponds to the maximum inrush current [1]. The derivation of WTHD for an induction motor is illustrated in equation 2.3. This WTHD can be further extended to be applicable in loss calculations [1].

$$L_\sigma = L_{ls} + \frac{L_m L_{lr}}{L_m + L_{lr}} \tag{2.2}$$

$$\begin{aligned} HD_i &= \frac{1}{\omega_1 L_\sigma}\sqrt{\sum_{n=2}^{\infty}\left(\frac{U_n}{n}\right)^2} \quad \Bigg| \quad \frac{\omega_1 L_\sigma}{U_1} \\ WTHD &= \frac{\sqrt{\sum\limits_{n=2}^{\infty}\left(\frac{U_n}{n}\right)^2}}{U_1} \end{aligned} \tag{2.3}$$

It is readily seen that WTHD weights lower harmonics by considering the $1/n^2$ factor, as mentioned earlier. To make the WTHD less sensitive to modulation index changes, e.g. avoid huge changes with the amplitude of the fundamental component, is U_1 replaced with the dc-link voltage. The new WTHD term which is now normalized with respect to the dc-link voltage is commonly termed WTHD0. The WTHD can also be derived to consider frequency dependent parameters [1]. A detailed derivation of the WTHD's for frequency dependent parameters is found in reference [1], page 82-93. The conclusions are represented below in equations 2.4 and 2.5.

$$\begin{aligned} WTHD0 &= \frac{\sqrt{\sum\limits_{n=2}^{\infty}\left(\frac{U_n}{n}\right)^2}}{U_{dc}} \\ WTHD01 &= \sqrt{\sum_{n=2}^{n_{0b}}\left(\frac{1}{n}\frac{U_n}{U_{dc}}\right)^2 + \frac{\omega_1}{\omega_{0b}}\sum_{n=n_{0b+1}}^{\infty}\frac{1}{n}\frac{4}{\left[\sqrt{n\frac{\omega_1}{\omega_{0b}}}+1.5\right]^2}\left(\frac{U_n}{U_{dc}}\right)^2} \\ WTHD02 &= \sqrt{W1 + W2 + W3} \end{aligned} \tag{2.4}$$

$$
\begin{aligned}
W1 &= \sum_{\substack{n=3k\pm1\\k=1,2,3,..}}^{n_{0b}} \frac{1}{n^2}(\frac{U_n}{U_{dc}})^2 \\
W2 &= \sum_{n_{0b}+1}^{n_{2b}} \frac{1}{n^{3/2}}\sqrt{\frac{\omega_1}{\omega_{0b}}}(\frac{U_n}{U_{dc}})^2 \\
W3 &= \sum_{n_{2b}+1}^{\infty} \frac{4}{\sqrt{n}} \frac{(\frac{\omega_1}{\omega_{0b}})^{3/2}(\frac{U_n}{U_{dc}})^2}{[\sqrt{n\frac{\omega_1}{\omega_{0b}}}+1.5]^2}
\end{aligned}
\tag{2.5}
$$

As mentioned, WTHD0 do not consider frequency dependent motor parameters, this is seen by the absence of the frequency dependence term ($\sqrt{\omega_1/\omega_{0b}}$) in equation 2.4. WTHD01 is the WHTD expression for stator parameters, with frequency dependency. In equation 2.4, if the last term under the square-root is neglected and the upper limit in the first summation sign is changed from n_{0b} to infinity, then the WTHD01 will be independent of frequency and equals the WTHD0. WTHD02 represents the WTHD for frequency dependent rotor resistance and rotor leakage inductance in an induction machine.

The "0" in for example WTHD01, indicates that it is normalized to the dc-bus voltage, which can also be seen in the equations above. Considering the WTHD02, n_{0b} is the highest harmonic order where $\omega_b < \omega_{0b}$ for the frequency dependent rotor resistance, and n_{2b} is the highest harmonic order where $\omega_b < \omega_{2b}$ for the frequency dependent rotor inductance.

Figure 2.1 illustrate how induction machine parameters typically change with frequency. Three different intervals of frequency dependence can be located. In WTHD02, W_1 represents the low frequency range where the frequency seen from the rotor is so low that AC parameters can be considered as DC parameters. W_2 represents the frequency range where the rotor resistance starts to increase due to the rotor bar effect. W_3 represents the frequency dependent range where both the rotor resistance and inductance are frequency dependent [1].

2.1.2 WTHD Relation to Harmonic Losses in an Induction Machine

How WTHD is related to losses due to harmonic distortion can be observed by isolating out load dependent quantities from an loss expression. First, loss calculations on an induction motor with constant machine parameters are considered. The stator and rotor loss can then be separately calculated as shown in equation 2.8 [1].

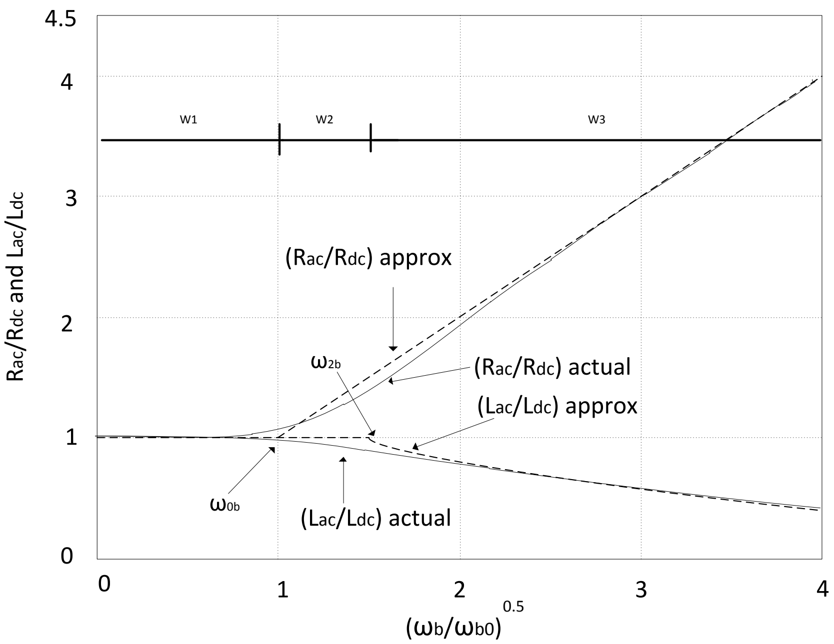


Figure 2.1: Sketch of normalized approximated and actual resistance and inductance of a solid rectangular bar in a slot as a function of the normalized bar frequency [1].

$$
\begin{aligned}
P_{1(cu)} &= 3I_{1n}^2 r_1 \\
&= 3(\frac{I_{1n}}{I_{1inrush}} I_{1inrush})^2 r_1 \\
&\cong 3\Big(\frac{\big[\frac{U_n^{'}}{\omega_n(L_1^{'}+L_{lr})}\big]^2}{\big[\frac{U_1^{'}}{\omega_1(L_1^{'}+L_{lr})}\big]^2}\Big) I_{1inrush}^2 r_1 \\
&= 3\Big(\frac{\sqrt{\sum\limits_{n=2}^{\infty}(\frac{U_n}{n})^2}}{U_1}\Big) I_{1inrush}^2 r_1 \\
&= 3(WTHD1)^2 I_{1,inrush}^2 r_1 \\
P_{2(cu)} &= 3I_{2n}^2 r_2 \cong 3\big[\frac{U_n^{'}}{\omega_n(L_1^{'}+L_{lr})}\big]^2 r_2 \\
&= 3(WTHD2)^2 I_{2,inrush}^2 r_2
\end{aligned}
\tag{2.6}
$$

Where $L_1^{'}$ is the resultant inductance in a Thevenin equivalent referred to the rotor side, e.g. parallel of L_{ls} and L_m. $U_n^{'}$ is the Thevenin equivalent voltage, e.g. the voltage over the mutual inductance. Load dependent quantities are her isolated inside square brackets, this shows that the source dependent part of the copper loss can be reduced by minimizing the WTHD factor.

When considering the the simplification of constant parameters are WTHD1 and WTHD2 are actually given by the same equation in an induction motor, as shown below [1]. Normalizing this mutual expression with respect to the DC-bus voltage gives the WTHD0.

Constant motor parameters

$$
\begin{aligned}
\sum_{n=2}^{\infty}\frac{P_{1,n}}{P_{1,inrush}} &= WTHD1^2 = \sum_{n=2}^{\infty}\frac{3\big[\frac{U_n'}{\omega_n(L_1'+L_2)}\big]^2 r_1}{3\big[\frac{U_1'}{\omega_1(L_1'+L_2)}\big]^2 r_1} = \sum_{n=2}^{\infty}\Big(\frac{\omega_1}{\omega_n}\Big)^2\Big(\frac{U_n'}{U_1'}\Big)^2 \\
\sum_{n=2}^{\infty}\frac{P_{2,n}}{P_{2,inrush}} &= WTHD2^2 = \sum_{n=2}^{\infty}\frac{3\big[\frac{U_n'}{\omega_n(L_1'+L_2)}\big]^2 r_2}{3\big[\frac{U_1'}{\omega_1(L_1'+L_2)}\big]^2 r_2} = \sum_{n=2}^{\infty}\Big(\frac{\omega_1}{\omega_n}\Big)^2\Big(\frac{U_n'}{U_1'}\Big)^2 \\
\frac{WTHD1^2}{U_{dc}} = \frac{WTHD2^2}{U_{dc}} &= WTHD0^2 = \sum_{n=2}^{\infty}\Big(\frac{\omega_1}{\omega_n}\Big)^2\Big(\frac{U_n'}{U_{dc}'}\Big)^2 = \underline{\sum_{n=2}^{\infty}\Big(\frac{1}{n}\Big)^2\Big(\frac{U_n'}{U_{dc}'}\Big)^2}
\end{aligned}
\tag{2.7}
$$

The total loss is calculated by implementing equation 2.7 into equation 2.8, see equation 2.8.

$$
\begin{aligned}
P_{1(cu)} &= 3I_1^2 r_1 \\
&= 3(WTHD0)^2 I_{1,inrush}^2 r_1 \\
P_{2(cu)} &= 3I_2^2 r_2 \cong 3\Big[\frac{U_n'}{\omega_n(L_1'+L_2)}\Big]^2 r_2 \\
&= 3(WTHD0)^2 I_{2,inrush}^2 r_2 \\
P_{loss,cu} &= P_{1(cu)} + P_{2(cu)} = 3(WTHD0)^2(I_{1,inrush}^2 r_1 + I_{2,inrush}^2 r_2)
\end{aligned}
\tag{2.8}
$$

The equation above makes it is clear that a reduction in WTHD0 corresponds with a reduction in losses due to harmonic distortion since the load dependent parts of the losses are isolated out. An example on how WTHD can be used in loss calculation is shown below [1].

Assuming that there are a frequency dependent rotor resistance and an independent stator resistance. The converter is operating in square-wave-mode.

Induction machine parameters are:

$$
\begin{array}{ll}
r_1 = 0.03\Omega & r_2 = 0.04\Omega \\
X_1 = 0.1234\Omega & \omega_1 = 377 rad/s \\
X_m = 2.5\Omega & \omega_{b0} = 300 rad/s \\
X_2 = 0.1176\Omega & WTHD2 = 0.11 \\
U_{ll} = 230V & WTHD1 = 0.046
\end{array}
$$

$$
\begin{aligned}
\frac{\sum_n P_{1n}}{P_{1,inrush}} &= (WTHD1)^2 \\
\sum_n P_{1n} &= (WTHD1)^2 \frac{(U_1^{'})^2}{[\omega_1(L_1^{'}+L2)]^2} r_1 = 18.68 W/phase \\
\frac{\sum_n P_{2n}}{P_{2,inrush}} &= (WTHD2)^2 \\
\sum_n P_{2n} &= (WTHD2)^2 \frac{(U_1^{'})^2}{[\omega_1(L_1^{'}+L2)]^2} r_2 = 140.0 W/phase
\end{aligned}
\tag{2.9}
$$

In higher frequencies, field weakening mode, copper losses due to harmonic components in the induction machine is reduced. In reference [1] the same example has been evaluated with the machine operating in field weakening mode with an operating speed of 2 pu. The induction machine losses are then $P_{1n} = 4.67\ W/phase$ and $P_{2n} = 52.64\ W/phase$.

2.2 Design of Programmed Modulation Pulse Width Patterns for Three-Level Neutrally-Point-Clamped Converters

The general term of the modulation strategy is Programmed Modulation, with the sub-terms Selective Harmonic Elimination Modulation and Synchronous Optimal Modulation, each with their own harmonic properties. Both are synchronous modulation techniques because both are based on employing pre-calculated PWM pattern for one fundamental period, and running this in a repetitively manner to modulate the voltage waveforms, in steady-state. In this section will the origin of these two aforementioned

Some papers and books also use the term "Harmonic elimination Pulse-Width modulation" for Selective Harmonic Elimination Modulation [10].

2.2.1 Selective Harmonic Elimination

In Programmed Modulation switching events can take place freely over the fundamental period, due to the fact that switching instants are not given by a carrier signal. This make it possible to calculate switching instants, normally addressed with an angular value, that have an eliminating effect on selected harmonic components [1]. With help from Fourier analyses, this is obtained by deriving equations that represents the amplitude of the harmonic components. By introducing variable switching angles (α_k) into these equations, switching events can be detected that will eliminate harmonic components. If the technique is used to eliminate harmonic components then it is termed Selective Harmonic Elimination(SHE) PWM or SHEPWM. Figure 2.2 illustrates the voltage waveform from a converter bridge-leg for a Three-Level NPC inverter where an n'th number of pulses have been introduced.

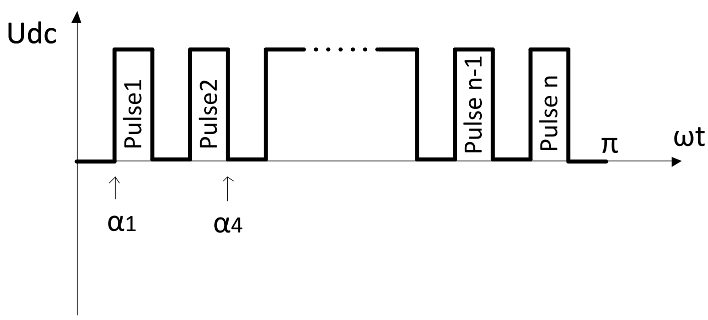


Figure 2.2: Illustration of voltage pulses given by pre-calculated switching-angles for a converter bridge-leg. Three-Level NPC converter.

Fourier analysis can be used to describe any kind of repeating infinite series in terms of sine and cosine terms [11]. A pulse width modulated converter-leg voltage in steady-state can therefore be expressed in terms of *sine* and *cosine* terms. In steady-state, ideally, the equilibrium line is zero which corresponds to a DC term equal to zero. When this is assumed, the general Fourier equation becomes

$$f(\omega t) = \sum_{h=1}^{\infty} a_h \sin(h\omega t) + b_h \cos(h\omega t) \tag{2.10}$$

If the converter-leg voltage waveform is chosen to have quarter-wave symmetry, and also include that it can have a positive, negative and zero voltage state, which it has in an Three-Level NPC inverter. Then the Fourier coefficients will be given by equation 2.11 [12], for a Three-Level NPC inverter leg voltage. There are several benefits with the use of quarter-wave symmetry which will be explained shortly.

$$\begin{aligned} a_h &= \frac{4U_{dc}}{h\pi}\left[\sum_{k=1}^{N}(-1)^{k+1}\cos(h\alpha_k)\right] \\ b_h &= 0 \end{aligned} \tag{2.11}$$

Now that an expression for the Fourier coefficients has been presented for the Nth harmonic component, can a harmonic eliminating switching instant be calculated for the selected harmonic components. By considering one particular over-harmonic component, can the switching-angle be simply detected by setting the specific harmonic component coefficient equal to zero and solve for (α_k). E.g. with N=2, only one harmonic component can be eliminated because one of the angles must be used to control the modulation index m, hence, the amplitude of the fundamental voltage component. Figure 2.3 indicates the SHE switching angles as a function of the modulation index m, with N=2. This was calculated by the use of an MATLAB algorithm that uses the f-solve function, developed by Roy Nilsen at Wärtsilä Norway AS.

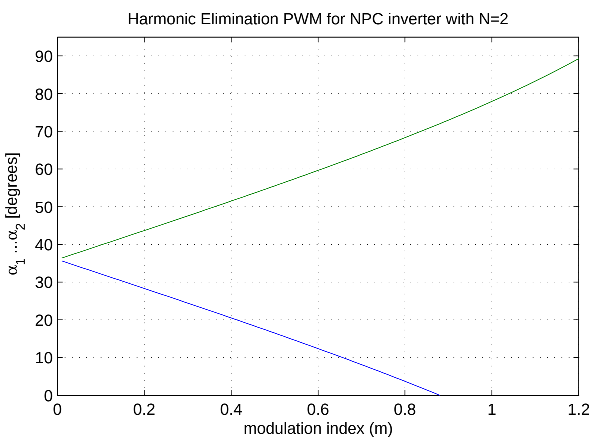


Figure 2.3: Selective Harmonic Elimination switching-angles with respect to the modulation index, $\alpha_1 < \alpha_2$.

A detailed derivation on how a similar equation to that in equation 2.11 can be obtained for a Two-Level inverter is illustrated in reference[1].

When selecting the harmonic components that are to be eliminated in a SHEPWM pattern, then only the $h = 6k \pm 1$ order harmonics (h) needs to be "dealt with" when solving the equations. Where k is $\in [1 \rightarrow \infty]$. The reason for this is that the third harmonic and its multiples will get cancelled out at the output of the converter, or simply, not seen by the wye-connected load. All even harmonics are cancelled out due to the fact that the positive and negative pulses are symmetrical [1]. The first harmonic components that should then be eliminated are therefore the 5^{th}, 7^{th}, 11^{th}, 13^{th}, 17^{th}, 19^{th}, 23^{th} and so on.

Control of the fundamental voltage amplitude is achieved by setting the a_1, that represent the fundamental inverter-leg voltage amplitude, equal the modulation index. See equation 2.12. This will be an constraint when solving the multiple equations for the harmonic eliminating angles. Hence, $N - 1$ harmonic components can be eliminated, where N is the number of switching events within each quarter wave or the pulse-number of each half wave. This also necessitate that a pulse pattern has to be pre-calculated for every modulation index that the drive system will encounter, when using pre-calculated SHEPWM-patterns.

$$a_h, (pu) = \frac{4U_{dc}}{h\pi}\left[\sum_{k=1}^{N}(-1)^{k+1}\cos(h\alpha_k)\right] = m \tag{2.12}$$

The basic formulas for generating SHEPWM patterns have now been presented here and equation 2.11 has N (α_1 to α_N) variables. In addition to the criteria in equation 2.12, the fact that the angles (α_1 to α_N) are in an ascending order is also added as a constraint in the calculation process, see equation 2.13 [12].

$$\alpha_1 < \alpha_2 < \alpha_3 < \alpha_4 < ....\alpha_N < \frac{2}{\pi} \tag{2.13}$$

So far only one inverter bridge-leg voltage (U_{a0}, U_{b0} or U_{c0}) waveforms have been discussed, these have the potential measured between the converter bride leg and the neutral point in the inverter. Fortunately are SHEPWM patterns generated for the converter bridge-leg voltages also eliminating the same harmonic components in the line-to-line voltages.

Equation 2.14 and 2.15 illustrate the voltage relations between the d- and q-voltage components, phase a voltage, line-to-line voltages and the converter bridge-leg voltages [13], for an induction machine. Also in a symmetrical three-phase load such as an induction machine is the zero voltage component U_0 equal to 0 [13].

$$U_{sd}^s(t) = U_a(t) = \frac{1}{3}\big(2U_{a0}(t) - U_{b0}(t) - U_{c0}(t)\big) = \frac{1}{3}\big(U_{ab}(t) - U_{ca}(t)\big) \tag{2.14}$$

$$U_{sq}^s(t) = \frac{1}{\sqrt{3}}\big(U_b(t) - U_c(t)\big) = \frac{1}{\sqrt{3}}(U_{b0}(t) - U_{co}(t)) = \frac{U_{bc}(t)}{\sqrt{3}} \tag{2.15}$$

From equation 2.14 is it shown that the phase voltage equals the difference between two line voltages. The harmonic components in the d- and q-components are given by equation 2.16 and 2.17 for a three phase symmetrical system [14].

$$U_{sa,h} = U^s_{sd,h} = \frac{2\hat{U}_{a0,h}}{3} \sin\left(\frac{h\pi}{3}\right)\left(2\sin\left(h\left(\frac{\pi}{3} - \omega t\right)\right) + \sin\left(h\left(\pi - \omega t\right)\right)\right) \tag{2.16}$$

$$U^s_{sq,h}(t) = \frac{2\hat{U}_{a0,h}}{\sqrt{3}} \sin\left(\frac{h\pi}{3}\right) \sin(h(\pi - \omega t)) \tag{2.17}$$

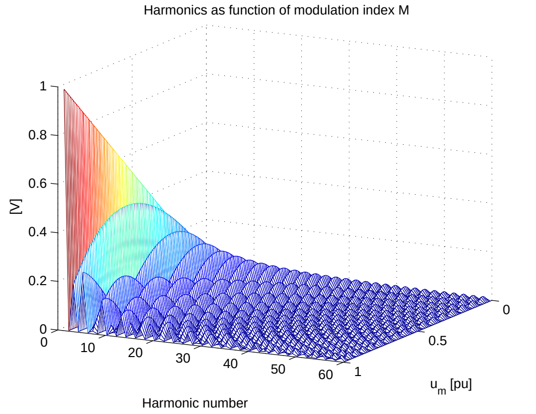
Note that the third harmonic and its multiples are eliminated by the first *sin* term when expanding from one phase leg to a three phase system, see equation 2.16 and 2.17. By considering the harmonic components of order $h = 6k \pm 1$ into the equations above gives [13]

$$U_{a,6k\pm1} = \hat{U}_{a0,6k\pm1} \cos((6k \pm 1)\omega t) \tag{2.18}$$

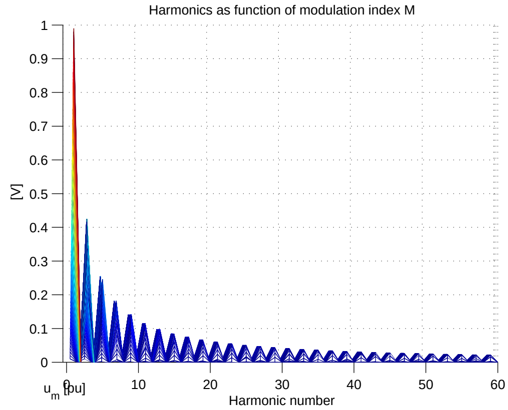
$$U^s_{sq,6k\pm1} = \pm\hat{U}_{a0,6k\pm1} \sin((6k \pm 1)\omega t) \tag{2.19}$$

Which reveals that the same order harmonic components exist in the d- and q components as in the bridge-leg voltages and for the phase voltages, hence, harmonic elimination properties in the three-phase system applies when the Programmed Modulation pattern is developed for converter bridge-leg voltages.

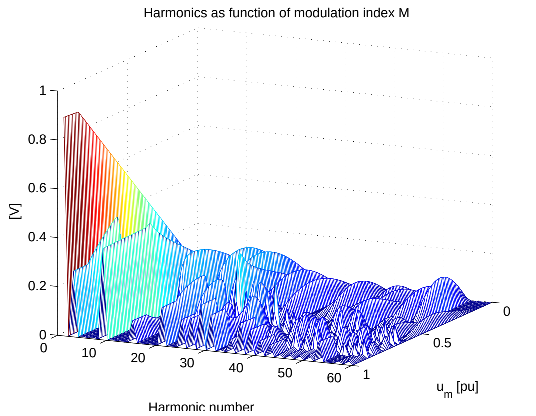
SHEPWM can eliminate the harmonic components, but only to a finite extent, limited by the number of pulses in the PWM pattern in question. The excess harmonic components that are not being eliminated by the SHEPWM pattern will actually increase in amplitude, this is shown in figure 2.4.



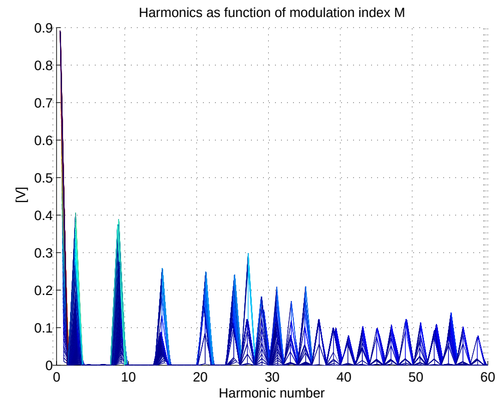
(a) Harmonic components without SHEPWM (N=1, sideview)



(b) Harmonic components without SHEPWM (N=1, frontview)



(c) Harmonic components with SHEPWM (N=8, sideview)



(d) Harmonic components with SHEPWM (N=8, frontview)

Figure 2.4: Harmonic amplitudes from 1^{st} to 60^{th} harmonic component, the modulation index m is indicated as u_m in the figures.

It is shown above that the $h = 6k \pm 1$ harmonic components, in figure 2.4 (c) and (d), are eliminated up to the 23^{th} order harmonic component. The harmonic coefficient with respect to modulation index is shown in figure 2.4 (a) and (c). It becomes clear when comparing figure 2.4 (b) and (d) that the SHEPWM has a "pushing" effect on the harmonic content to a higher harmonic order, e.g. compare from the 25^{th} and higher harmonic components in figure2.4 (b) and (d).

2.2.2 Synchronous Optimal Modulation

The equations for the harmonic component coefficients developed for SHEPWM can also be used in a modulation technique called Synchronous Optimal Modulation(SOM). The SOM switching patterns are pre-calculated with a different objective in terms of harmonic-content, compared to SHEPWM. SOM pattern switching events are determined in a way that reduce the harmonic content in the current, also reducing losses due to harmonic distortion in the controlled induction machine

[1]. This can be achieved by including the source dependent WTHD0 factor into the optimization process as a minimization criteria. How minimizing the WTHD0 will minimize losses due to harmonic distortion in the controlled induction machine is shown in section 2.1.

Due to skin effects, which gives frequency dependent motor parameters, deviations from the "optimal" pulse pattern will occur. A more sophisticated optimization that includes frequency dependency would be a complicated and comprehensive approach to find "the optimal" switching pattern. In reference [15] loss factors regarding harmonic copper are analysed, with and without skin effects. Also included here are the harmonic end-leakage losses and total harmonic stray load losses. However, the conclusion is that the optimal PWM pattern derived from WTHD0 was still the optimal pulse pattern when all loss factors where taken into consideration. Hence, it is sufficient to use simple WTHD0 as the minimum loss optimization criteria. Also [16] concludes that the deviations are very small when comparing a simple load model with a comprehensive one that includes skin effect, hence frequency dependent parameters.

The SOM pulse patterns for a Three-Level NPC converter is, of course, designed for a three-phase and the optimum voltage waveforms will be quarter-wave symmetric, for reasons explained in section 2.2.1. Then the third harmonic and it's multiples will not be seen by the load, and even harmonics will not exist. Hence the objective function in the optimization process is to minimize the following WTHD expression [1].

$$WTHD0 = \frac{1}{m}\sqrt{\sum_{k=1}^{\infty}\Big[\Big(\frac{U_{a0,6k-1(pu)}}{6k-1}\Big)^2 + \Big(\frac{U_{a0,6k+1(pu)}}{6k+1}\Big)^2\Big]} \tag{2.20}$$

Equation 2.22 will be used as the minimization criteria in this study since an algorithm has allredy been developed by Roy Nilsen at Wärtsilä Norway AS. But there are other objective functions that are minimizing the harmonic content in the phase currents. In reference [17] a loss factor d^2 is presented, which is represented in [18] as

$$d = \sqrt{\sum_{k\neq 1} h_i^2(kf_1)} \tag{2.21}$$

where h_i is the discreet current spectra given by

$$h_i(kf_1) = \frac{I_{h,rms}(kf_1)}{I_{h,rms,six-step}} \tag{2.22}$$

and k is the harmonic order[18].

2.2.3 Non-linear Transcendental Equation Set

When several harmonic components are to be eliminated or optimized, a set of non-linear transcendental equations has to be solved to obtain the discreet switching instants [12]. There are various ways of solving these equations to obtain the SHEPWM patterns, or the Synchronous Optimal Modulation patterns [19]. One of the most demanding challenges associated with Programmed Modulation is actually to find the "right" initial values close enough to an exact solution when generating patterns for a Three-Level NPC converter.

As the number of switching instants increase(N) the number of solutions also increase(due to many local minimum values), this make it difficult to be on the best initial "switching-angles" in each iteration, as the number of pulses increase. But if achieved, a fast convergence in the numerical calculations, and in the case of Optimal Pulse width Modulation, will it give a discrete switching pattern with the lowest WTHD0 values. E.g. Optimal.

In reference [20] different approaches are discussed for a two level inverter and the recommended solution is eventually to use different predicting schemes to generate initial switching values, and solve them with Newton's method. It do not exist any straight-forward techniques on how to find "the optimal" initial switching-angles when solving the equations for a Three-Level NPC converter. So in the case of finding the Programmed Modulation angles the "cut and try" method can used [13]. This is a tiresome method and one of the problems with this approach is that it is difficult to know if the solution is "the optimal" solution or just a local minima solution.

In reference [19] a technique called Chaotic Ant Colony Algorithm is used to solve the SHEPWM optimization to find the right switching angles for a Three-Level NPC inverter with 4 Pulses. The basis in that paper was to suggest a calculation method that could be used live during operation.

2.3 Dynamic and Transient Control in a Synchronous Optimal Modulation System

In high performance medium voltage drive systems, dynamic speed and torque control is a requirement for a solution to be able to compete on the commercial market. In this section the theory behind a control method that controls the stator flux-linkage to reduce current transient when going from one steady-state to a new one, will be presented. This is achieved trough a Stator Flux Trajectory Controller. Further on, is it described how this Stator Flux Trajectory Controller can be used to achieve torque and speed control in the drive system.

2.3.1 On-Line Manipulation of Optimal Switching pattern to Control the Stator Flux

When an adequate number of optimal pulse patterns have been pre-calculated for "every" steady state, for a given drive system, a control system to govern dynamic operations is needed. If the reference voltage change to a new system steady state operation, and the new pulse pattern that fits the new steady state is employed, unacceptable transient can occur if not controlled. Similar transients can occur even if the voltage amplitude between the old and the new pattern is neglect-able [4]. This is due to a displacement of the optimal stator flux trajectory which corresponds to the new voltage pattern and the old stator flux trajectory of the old switching pattern.

The stator flux-linkage cannot change instantaneously with a step change in voltage [4], this can lead to undesirable effects. E.g at the instant a new voltage pattern is put to use will the stator flux linkage vector still have its old position as the initial value, this displacement can results in over-currents with a slow time response. Simulation examples of such situations is shown in chapter 4.

This stator flux displacement is called the dynamic modulation error [4]. To reduce transient time and current amplitude when changing voltage pulse patterns, a method called Stator Flux Trajectory Control can be used. This technique reduce the dynamic modulation error in two steps as stated in reference [4] represented here:

> *(i) modify any new pulse pattern prior to its use, based on a prediction of the dynamic modulation error, and subsequently (ii) estimate and eliminate, to the extent possible, the dynamic modulation error during the subsequent sampling interval.*

Figure 2.5 gives an overview of how Joachim Holtz presents his Programmed Modulation system [4].

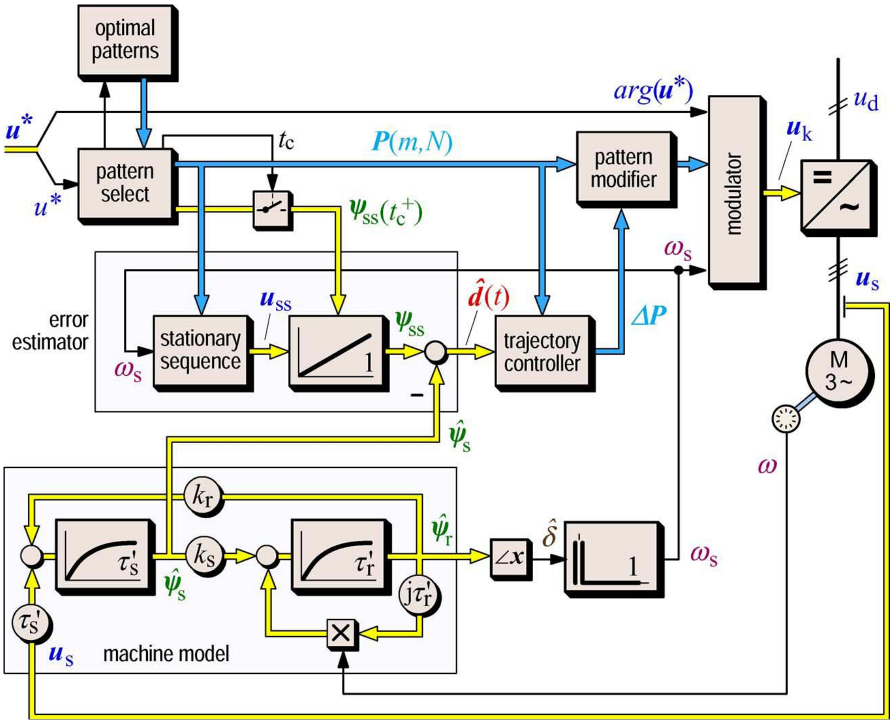


Figure 2.5: Stator flux trajectory tracking control system block diagram [2].

Note that a pulse pattern(P) is given by two variables, N and m. N, if the pulse number is given by $N = fs/(2f_1)$ then it also indicate the number of switching events in the quarter wave pattern. N is an integer number since the switching frequency is synchronous to the fundamental frequency. The modulation index(modulation amplitude) is abbreviated as m and is proportional to the fundamental voltage component a_1 [2]. In the block diagram 2.5 are the blue arrows indicating the main path for the discrete pulse pattern. The main blocks in the system are the machine model, error estimator, pattern selector and pattern modifier.

If the control system is operating in steady state with a pulse number of N equal 5 and a modulation index equal to 0.8 at the time t_1, then the voltage $(u_{ss}(t))$ and the optimal stator flux trajectory(ψ_{ss}) will be as illustrated in figure 2.6.

The stator flux is given by equation 2.23 [4].

$$\Psi_{ss}(t) = \int_{t_1}^{t} u_{ss}(t)dt + \Psi_{ss}(t_1) \tag{2.23}$$

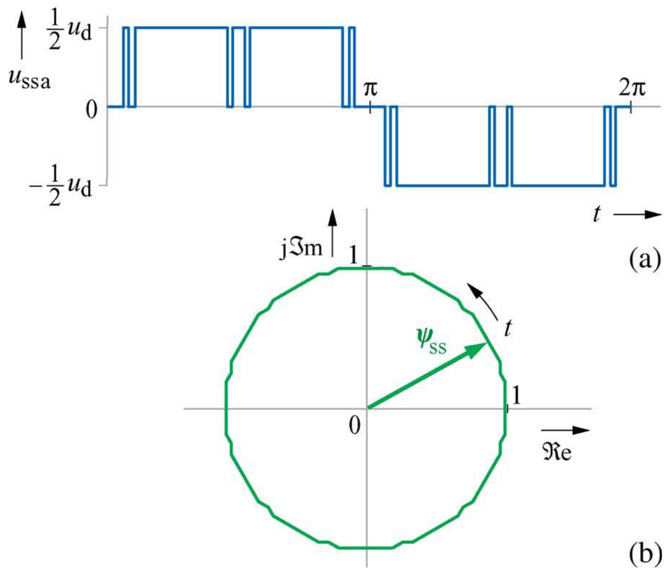


Figure 2.6: (a)Optimal stator voltage waveform and (b) corresponding optimal stator flux trajectory [2].

When a change in the steady-state voltage is required by the system, at the time instant t_c, a new pulse pattern $P(N, m)$ is chosen to control the system at $t > t_c$. The new pulse pattern is collected from the Optimal Patterns box in figure 2.5. For the coming interval the stator flux trajectory is given by [2].

$$\Psi_{ss}(t) = \int_{t_1}^{t} u_{ss}(t)dt + \Psi_{ss}(t_c) \tag{2.24}$$

Where $u_{ss}(t)$ is given by the pre-calculated voltage pattern. To determine $\Psi_{ss}(t_c)$, the initial stator flux Ψ_{ss} with respect to the newly selected pulse pattern $P(N, m)$ must be calculated.

$$\Psi_{ss}(\alpha = 0) = \int_{0}^{2\pi} u_{ss}(\alpha, m, n)d\alpha \tag{2.25}$$

Where α is the angle for the voltage waveform stretching from 0 to 2π, one fundamental period. Then the $\Psi_{ss}(t_c)$ is given by[2]

$$\Psi_{ss}(t_c) = -\Psi_{ss}(\alpha = 0) + \int_{0}^{t_c} u_{ss}(t)dt \tag{2.26}$$

The actual $\hat{\Psi}_s$ can be estimated in the machine model by measuring the terminal voltage u_s and mechanical speed ω, see figure 2.5. The subscript ^ indicates estimated variables. Now can the dynamic modulation error be estimated, and is given as a volt-second error, by subtracting the estimated actual stator flux-linkage vector from the new steady state stator flux-linkage vector with respect to the new voltage pulse pattern. Hence [2]

$$\hat{d}(t) = \Psi_{ss}(t) - \hat{\Psi}_s(t) \tag{2.27}$$

$\hat{d}(t)$ will be used to calculate the necessary modification of the pulse pattern $P(N, m)$ to get the stator flux-linkage on the new optimal trajectory. This operation is carried out in the Trajectory Controller block seen in the block diagram. The output signal with the angular changes is marked in figure 2.5 as ΔP.

Adding together the optimal $P(N, m)$ with the ΔP will alter the switching events in the originally optimum pulse pattern $P(N, m)$ and hold this change within a sampling interval T_k. How the volt-second error can be converted to modification of switching events is explained here. First, the following rules have been established regarding the altering process in reference [2] and are quoted below

- *Steps to a more positive potential are characterized by* $s = +1$*. These are those in which a phase potential changes from* $-u_d/2$ *to* 0 *or from* 0 *to* $+u_d/2$*, where* u_d *is the dc link voltage. Delaying such transition by a displacement* $\Delta t > 0$ *reduces the volt-second contribution of that phase; the contribution increases when the transition is advanced, i.e.,* $\Delta t < 0$*.*
- *Transitions steps to a more negative potential are characterized by* $s = -1$*. These are those in which a phase potential changes from* $+u_d/2$ *to* 0 *or from* 0 *to* $-u_d/2$*. Delaying such transition by a displacement* $\Delta t > 0$ *increases the volt-second contribution of that phase; the contribution decreases when the transition is advanced, i.e.,* $\Delta t < 0$*.*
- *The absence of transition steps in a sampling interval is marked by* $s = 0$*.*

The rules define how the time-altering of the transition leads to positive or negative volt-second contribution and makes it easer to grasp how a transition affect the volt-second and the stator flux-linkage vector. Transitions steps are altered within the sampling interval T_k to reduce the modulation error and there can be n transitions within each sampling interval, per phase, depending on the given pattern. An example on how these transition steps can be moved within a given switching-patten, and thus alter the waveforms of the converter bridge-leg voltages, within a sampling interval, is illustrated in figure 2.7.

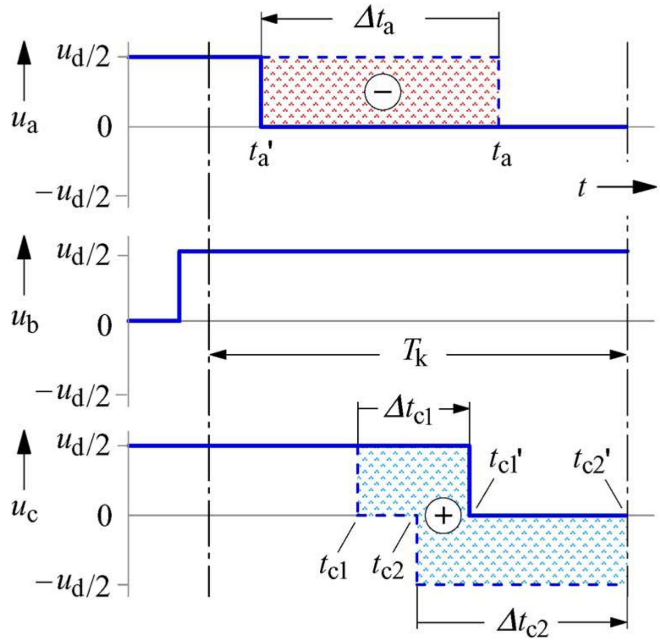


Figure 2.7: Illustrate the voltage of phase a, b and c where the sampling interval overlaps two transitions in phase c and one in phase a. The modifications reduce the dynamic modulation error $\hat{d}(t)$. [2]

The effect of Δt_a, which is the time displacement, on the dynamic modulation error is given by equation 2.28 [2].

$$\Delta d_a = -\frac{1}{3} u_d \sum_{i=1}^{n} s_{ai} \Delta t_{ai} \tag{2.28}$$

Δt_{ai} is a more generalized term due to the $i \in [1 \to n]$ term which indicates the existing number of transitions in the time interval T_k. The $\Delta d_a(k)$ will be in effect in the time interval $T_(k+1)$ and is estimated during $T_(k)$ by equation 2.29 [2]. During the time interval T_k the $\Delta d_a(k-1)$ will be in effect which was calculated in T_{k-1}.

$$\Delta d_a(k) = -(d_a(k) - \Delta d_a(k-1) \tag{2.29}$$

The minus signs in equation 2.29 comes from the fact that the modifications are inverse to the existing error. From equation 2.28 and 2.29 an expression for the time displacement in phase a can be obtained, see below. [2].

$$\Delta t_{ai} = \frac{3}{u_d} \frac{1}{s_{ai}} [d(k) - \Delta d(k-1)] \circ 1 \tag{2.30}$$

"1" is the unity vector, pointing in the phase direction of phase a. Similar expressions can be can be derived in the same manner for phase b and c where the unity vectors $a = 1\angle 120°$ and $a^2 = 1\angle -120°$ is used to point in the phase directions.

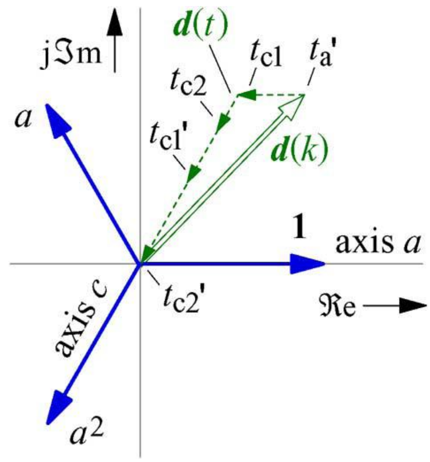


Figure 2.8: Vector diagram illustrating how transitions are reducing the dynamic modulation error $d(k)$ [2].

Figure 2.8 illustrates the dynamic modulation vector and compensating vectors for the given transitions illustrated in figure 2.7. A resulting expression for the three phases can now be derived to cover all phase manipulations, see equation 2.31 [2].

$$\Delta d = \frac{1}{6}u_d \sum_{i=1}^{n} \left[(2s_{ai}\Delta t_{ai} - 2s_{bi}\Delta t_{bi} - 2s_{ci}\Delta t_{ci} + j\sqrt{3}(2s_{bi}\Delta t_{bi} + 2s_{ci}\Delta t_{ci}\right] \tag{2.31}$$

2.3.2 Stator Flux Control - Speed and Torque Control

The conventional Direct Torque Control(DTC) technique do not need a dq-axis transformation, the electromagnetic torque is controlled by applying, constantly, appropriate stator voltage vectors [3]. The principles of how the torque is controlled can be used to achieve torque control in a Programmed Modulation drive system. This will be explained here. First consider the block diagram of an DTC system in figure 2.9

Illustrated in the block diagram, a voltage vector $\vec{u_s}$ is selected from estimated values of the stator flux-linkage vector $\vec{\Psi_s}$, its position θ_s, mechanical speed and the electromagnetic torque error. The stator flux-linkage and electromagnetic torque are estimated from the stator currents, the DC-bus voltage and converter gate signals.

Equation 2.32 can be used to estimate the stator flux-linkage space vector $\vec{\Psi_s}$ [3].

$$\vec{\Psi_s}(t) = \vec{\Psi_s}(t - \Delta T) + \int_{t-\Delta T}^{t} (\vec{u_s} - R_s\vec{i_s})d\tau = \hat{\Psi}_s e^{j\theta_s} \tag{2.32}$$

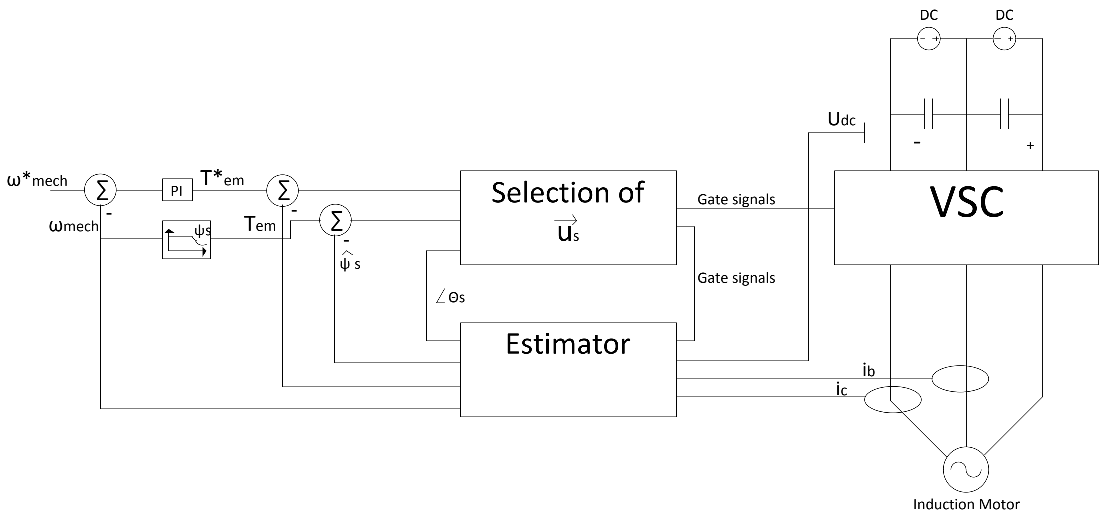


Figure 2.9: DTC block diagram overview [3].

Next, the rotor flux-linkage space vector, electromagnetic torque, and speed can be obtained by employing the equations listed under [3].

$$\begin{aligned}
\vec{\Psi}_r &= \frac{L_r}{L_m}(\vec{\Psi}_s - \sigma L_s \vec{i}_s) = \hat{\Psi}_r e^{j\theta_r} \\
\omega_r &= \frac{d}{dt}\theta_r \\
T_{em} &= \frac{p}{2} Im(\vec{\Psi}_s^{conj} \vec{i}_s)
\end{aligned} \tag{2.33}$$

It can be shown that electromagnetic torque in an induction machine can be expressed as the first equation in 2.34 [3]. It indicates that a change in the stator or rotor flux linkage vector, direction or amplitude, will influence the electromagnetic torque with respect to the change. In transient situations can the rotor flux-linkage be considered almost constant compared to the much faster stator flux-linkage. This can be seen by analysing the rate of change in the rotor flux-linkage vector referred to the rotor field orientation, given by equation 2.34. And comparing it with the rate of change in the stator flux linkage, which depends on the terminal voltage of the induction motor, see equation 2.32. It is therefore clear that it will be most efficient to use the stator flux-linkage vector compared to the rotor flux-linkage vector as an control parameter in a control system.

$$\begin{aligned}
T_e m &= \frac{p}{2}\frac{L_m}{L_\sigma^2}\hat{\Psi}_s \hat{\Psi}_r \sin\theta_{sr} \\
\vec{\Psi}_r^A(t) &= \vec{\Psi}_r^A(t-\Delta T) + \int_{t-\Delta T}^{t} (R_r \vec{i}_r^A) d\tau = \hat{\Psi}_r e^{j\theta_r^A}
\end{aligned} \tag{2.34}$$

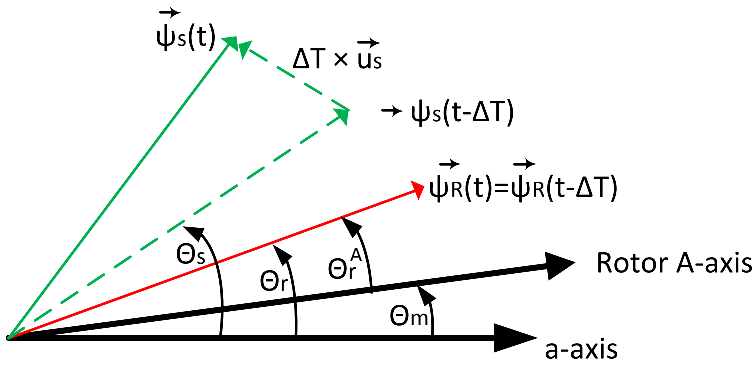


Figure 2.10: Flux-linkage vector diagram for DTC [3]

The basis in a DTC system is to choose a voltage space vector $\vec{u_s}$ form a table in the selector block, see figure 2.9, that will give a desired angular distance between the rotor and stator flux-linkage vector. Hence, give the desired electromagnetic torque. Figure 2.10 illustrate how the voltage vector $\vec{u_s}$ is used to increase the angular distance between the two flux-linkage vectors. This will in the illustrated case lead to a higher electromagnetic torque produced by the induction motor.

An overview of the DTC systems has been presented and it builds around controlling the stator flux-linkage space vector, indirectly, by controlling the terminal voltage of the induction machine. The Programmed Modulation control strategy use a Stator Flux Trajectory Controller(SFTC) to control the stator flux-linkage in the induction machine by changing an optimized pre-calculated voltage pattern. This method is promised to be very fast and effective in reference [4]. The functionalities of the SFTC is explained in section 2.3.1.

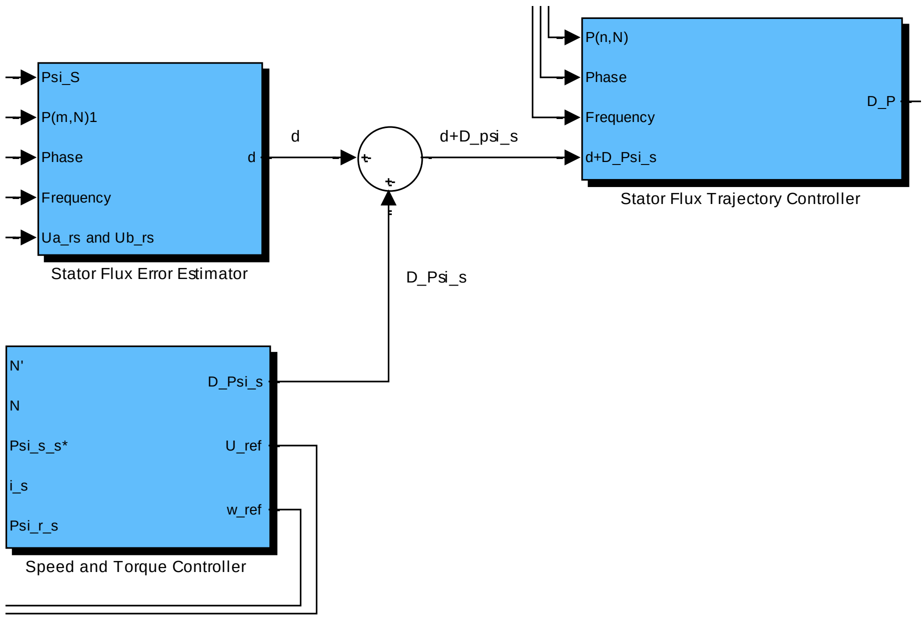
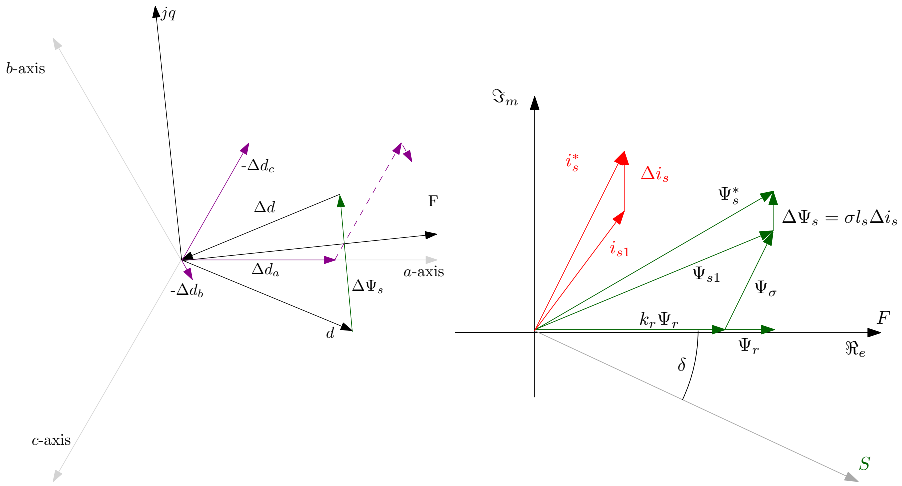


Figure 2.11: Injection of $\Delta\Psi_s$ shown in a block diagram extracted from the proposed simulation model.

The on-line manipulation of optimal pulse patterns is calculated in the SFTC and merged together with the original optimal SOM pattern in the modulator. The new modified switching pattern($P(m, N) + \Delta P$) will remove deviations between the optimal flux-linkage trajectory and the actual stator flux-linkage. This flux controller can also be used to control the angle between stator and rotor flux-linkage space vectors(θ_{sr}), if used correctly by a speed and torque controller.

In simple terms, can the desired change in stator flux-linkage($\Delta\vec{\Psi}_s$) be added to the dynamic modulation error($\vec{d}$) at the input of the SFTC. The controller will then process $\Delta\vec{\Psi}_s + \vec{d}$ as a modulation error between the optimal stator flux trajectory and the actual stator flux, and change the volt-second in the optimal pulse width pattern to implement $\Delta\vec{\Psi}_s + \vec{d}$ [4]. Figure 2.11 illustrates how the $\Delta\vec{\Psi}$ is added to the dynamic modulation error ($\vec{d}$) in a block diagram.

Figure 2.12(a) shows a vector diagram illustrating how the change in stator flux-linkage vector($\Delta\vec{\Psi}_s$) is injected into the dynamic modulation error($\vec{d}$). It also indicates how the resulting vector $\vec{d} + \Delta\vec{\Psi}_s$ error is divided onto each phase. Figure 2.12(b) Shows the different flux components in an induction machine, it also shows the relation between the$\Delta\vec{\Psi}_s$ and Δi_s[21].



(a) The dynamic modulation error d and the commanded change in stator flux $\Delta\Psi_s$ adds together as illustrated.

(b) Flux vectors which indicate the $\Delta\vec{\Psi}_s$ and Δi_s relation [21]

Figure 2.12: Stator and rotor flux vectors illustrated in vector diagrams. F indicates the rotor field oriented axis, also refereed to as the α-axis in this report. The two vector diagrams are not from the same operational state

Considering the stator flux vector in rotor field coordinates gives the $\Delta\Psi_s$ estimation as [21]

$$\Delta\Psi_s = \Psi_s^* - \Psi_{s1} \tag{2.35}$$

Where Ψ_s^* consists of a β-component that controls speed and a α-component (aligned with the rotor flux vector) that controls the machine magnetization, this is illustrated in figure 2.13. A PI regulator to each the d- and q-component is used to obtain a fast control response and to avoid that the control system will constantly feed a $\Delta\Psi_s$ signal to the SFTC [4].

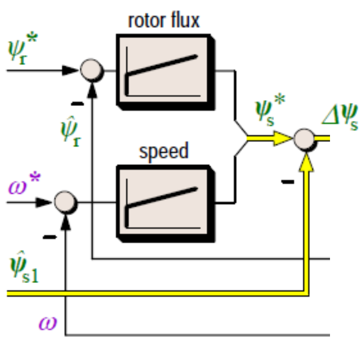


Figure 2.13: The composition of the stator flux reference and the resultant $\Delta\Psi_s$ [4], illustrated in a block diagram.

A conventional current-based controller in a Programmed Modulated system will not be a desirable solution because it will deteriorate the optimality of the pre-calculated pulse patterns [21]. Hence, the main purpose of the Programmed Modulation system, which is to be as close as possible to the pre-calculated voltage patterns to uphold the intention of the object function behind the patterns, would be partly lost. On the contrary can the optimality of the voltage pulse patterns be maintained by the use of the Stator Flux Trajectory Controller to achieve the desirable control [2].

2.3.3 Steady-State Voltage Reference

Unlike the DTC system that generates freely a $\vec{u_s}$ governed by the need to creating a desired torque, will a switching-pattern based drive system need a steady voltage reference[4], and modify its switching-pattern if a change in torque is required. If the speed change, due to a new change in applied torque, then the reference voltage also change to the new steady state.

The switching-pattern based control system needs a steady voltage reference signal without distortions from e.g. the converter switching. To have such a smooth voltage reference signal is very important to avoid perpetual stator flux-linkage deviations from its optimal trajectory. An unstable reference voltage would make the system constantly trying to adjust to new steady-state voltage pattern. Hence, the system will constantly be in a "transient operation mode". For this reason is it expedient to have a voltage reference that keeps the modulator in an approximate steady-state condition [4]. Proposed in references [4] is the principle of "*The self-controlled machine*".

The self controlled machine principle is based on perpetuated steady-state operations, unless the ΔP gives a change in the stator flux that will move the drive system to a new steady-state. A voltage reference estimation suggested in reference [4] is given in equation 2.36

$$u^{*'} = j\omega_{sss}\hat{\Psi}_{s1} + r_s \frac{\hat{\Psi}_{s1} - k_r\hat{\Psi}_r}{\sigma l_s} \tag{2.36}$$

Which generates a voltage reference from the rotor machine flux vectors and the rotor flux-linkage vectors and the angular speed(ω_{sss}).

The fundamental stator flux vector is given by

$$\tau_s' \frac{d\hat{\Psi}_{s1}}{d\tau} + \hat{\Psi}_{s1} = k_r\hat{\Psi}_r + G_s(\omega)(\hat{\Psi}_{s1} - \hat{\Psi}_s) + \tau_s' u* \tag{2.37}$$

where the $G_s(\omega)$ is a gain-tensor to account for model parameter mismatch [4], which is proposed to be $g_1 + j0$ in [22].

There is an alternative method of control of Switching-pattern based drive system in reference [12] that is based on U/f motor control. This method has not been analysed in this thesis.

2.4 Constant Switching Frequency in Programmed Modulation

Keeping the switching frequency low, and within a predefined range, is desirable since it will allow the converter to operate with a higher power density due to a reduction in switching-losses [6]. Programmed Modulation, where the switching frequency is synchronous to the fundamental frequency, requires that the switching frequency vary in proportion to the fundamental frequency in steps, to not exceed an upper switching frequency limit. This "step" can be considered as a way of "gearing" the modulation system by changing the pulse number N, as illustrated by the function $mf = f_s/f$ in figure 2.14.

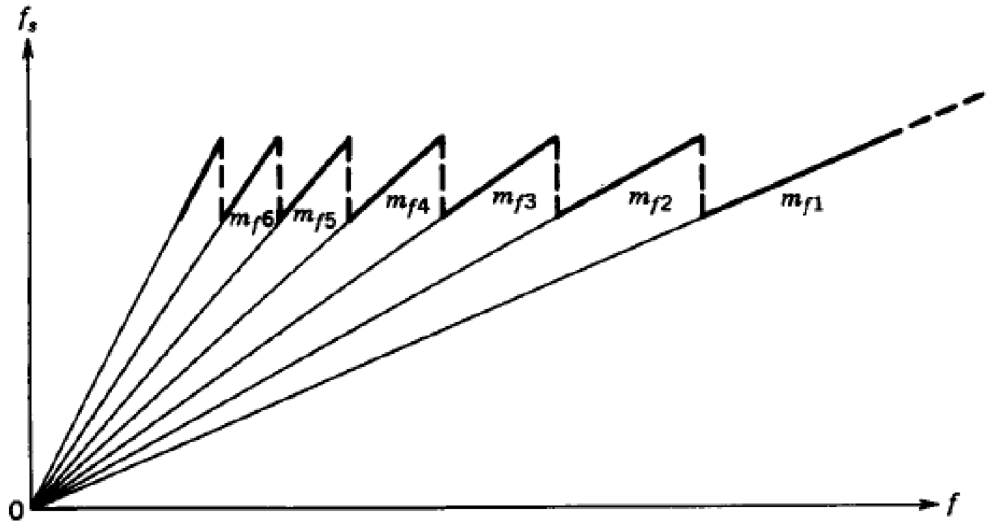


Figure 2.14: Switching frequency versus the fundamental frequency [5].

In variable frequency drive systems with Programmed Modulation will then the pulse number N vary as a function of f_1 to keep the switching frequency low. When the fundamental frequency is low then the pulse number is high, and vice versa.

If a switching period is considered as one $on - and - off$ switch, then the pulse number is given by equation 2.38 for a maximum switching frequency [23].

$$N = int\Big(\frac{f_{s,max}}{2f_1}\Big) \tag{2.38}$$

Which gives the following N with respect to the fundamental frequency with a $f_{s,max}$ equal to 400.

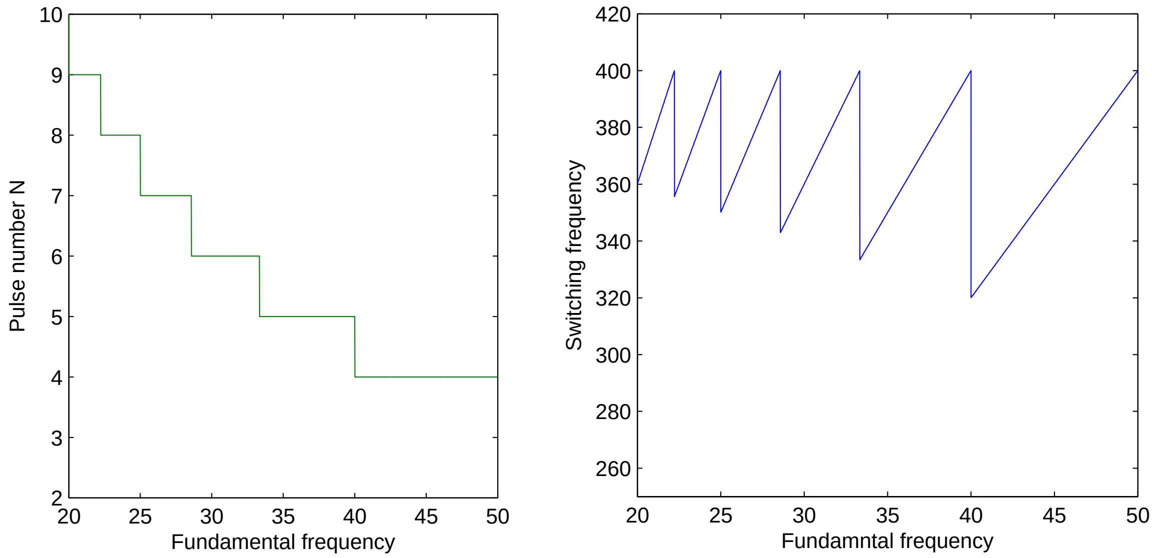


Figure 2.15: Switching frequency versus the fundamental frequency, with the pulse number N indicated in the left diagram. $f_{s,max}$=400.

At low fundamental frequencies will the number of pulses in the pre-calculated pattern get very high, this means a comprehensive optimization process is needed to generate the optimal PWM patterns. Also considering that other simpler modulation strategies are very good when operating with a high mf, is it clear that at low fundamental frequencies should a exchange between modulation strategies take place[1] [6].

Chapter 3

Modelling of A Switching-Pattern Based Drive System

To be able to analyse Programmed Modulation a simulation model has been built in MATLAB Simulink, with the additional simPowerSystem tool sett. Figure 3.1 shows an overview of the model illustrated as an block diagram, which is the interface in MATLAB Simulink. The Blue block is the modulation system which is the contribution in this master thesis. The green blocks are adopted from other models and represent the NPC converter and the induction motor which are used later in simulations.

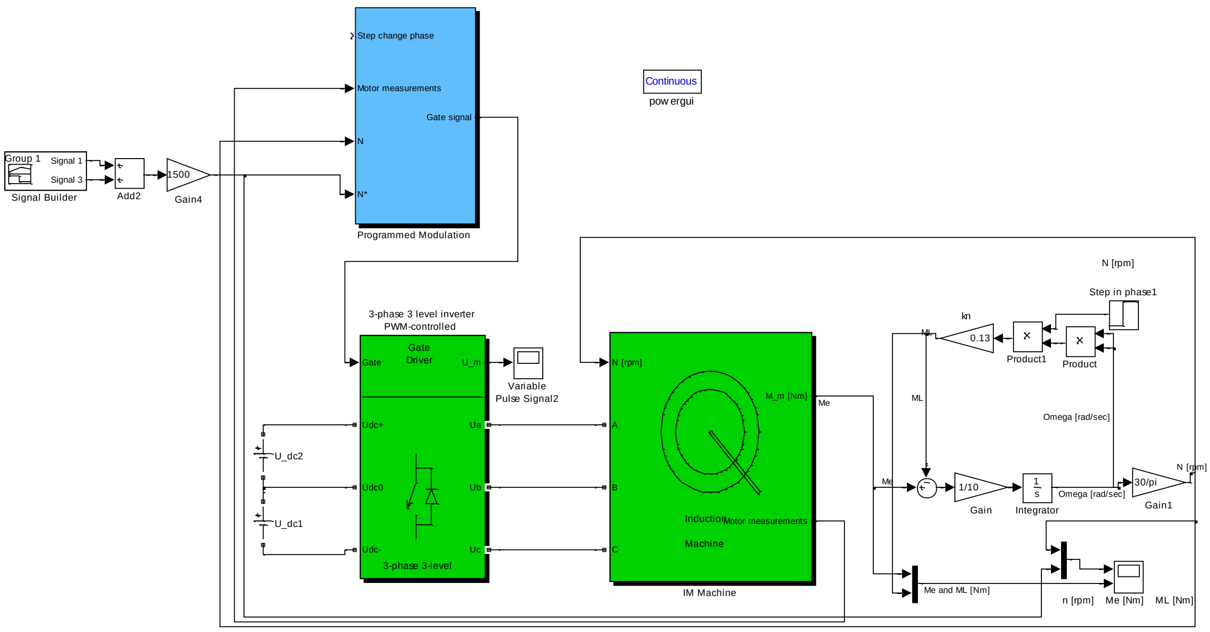


Figure 3.1: Overview of the modelled drive system

3.1 General Prospects of the Programmed Modulation Model

The subsystem of the Programmed Modulation block is shown in figure 3.2, it contains the main blocks of the modulation system, which is the Pattern Selector and the subsystems: Stator Flux Error Estimator, Speed and Torque controller, Stator Flux Trajectory Controller and Modulator.

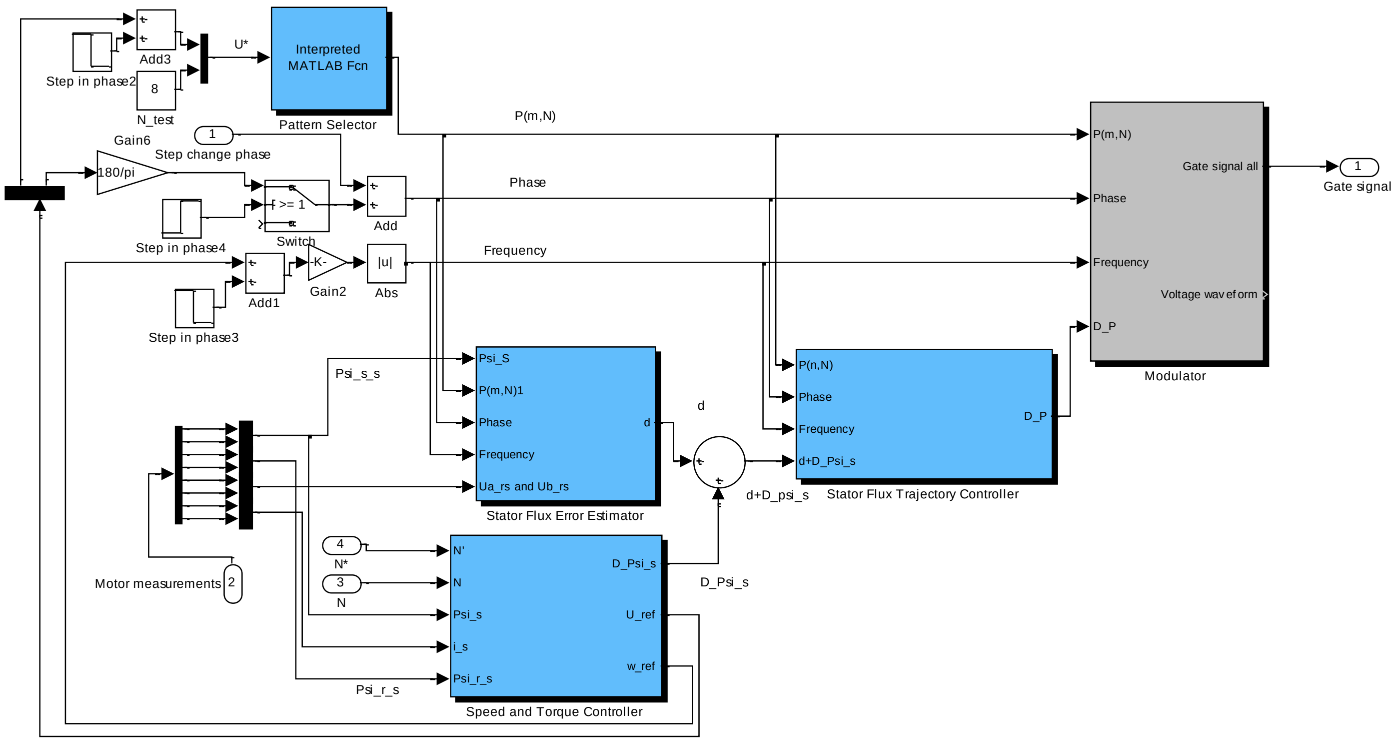


Figure 3.2: Programmed Modulation model overview

The Programmed Modulation system operates by processing sets of discreet switching instants representing a quarter wave of the fundamental converter bridge-leg voltage waveform. The Pattern Selector will choose a SOM pattern, which is implemented in this model, from a vector decided by the voltage reference m value and the pulse number N. Seen as the blue box in upper left corner in figure 3.2. This pattern will be distributed to the Stator Flux Error Estimator, Stator Flux Trajectory Controller and the Modulator.

The Stator Flux Error Estimator will compare a calculated optimal stator flux-linkage trajectory with the actual stator flux-linkage measured in the induction machine. The deviation will be the dynamic modulation error d. The optimal stator flux-linkage trajectory is calculated from the given optimal voltage pattern($P(m, N)$).

The Speed and Torque Controller block responsibility is to calculate a $\Delta\Psi$ that can increase or reduce the electromagnetic toque to reach a desired speed reference. Based on induction machine flux-linkages, speed reference and real speed. If the speed reference equals the actual speed, then the system should be considered "in a steady state" and the output from the controller ($\Delta\Psi$) should be zero. The controller block also estimates the reference electrical frequency(ω_{ref}) given by the angular speed of the rotor flux vector) and the voltage reference(U_ref) given by "the self controlled machine" principle.

When an estimation of the dynamic modulation error(d) and a desired change in stator flux-linkage($\Delta\Psi$) have been estimated and added together, will the result be the input of the Stator Flux Trajectory Controller block. The stator flux Trajectory Controller is to process this input and calculate the needed change in the switching pulse-pattern transactions, the change will correct the actual stator flux trajectory onto the desired trajectory. This may be an optimal trajectory, or an optimal trajectory with an extra $\Delta\Psi$ to make a change in the electromagnetic torque. The Flux controller presented in this thesis has two operational modes, the two modes have been named *Override* and *Synchronous*, discussed later.

When all measures have been counted for in the discreet part of the system is the original switching-pattern and the calculated changes sent to the Modulator. Here is the discreet switching values converted to converter gate signals. More detailed explanations is found in section 3.2.

As mentioned in the introduction chapter are the internal values of flux-linkages assumed measured inside the induction machine, due to this is the machine parameter estimation model, normally used to estimate machine parameters, neglected.

3.1.1 Global Understanding of Switching-Patterns in the Model

Global understanding in the model:

- A switching angle with an angular value equal to zero will be considered as a non-existent switching angle in the system. E.g a switching pattern with 4 switching-angles will be distributed in the system as a vector with 10 elements where 6 of them are zeros, example: $[\alpha_1, \alpha_2, \alpha_3, \alpha_4, 0, 0, 0, 0, 0, 0]$.
- The maximum number of discreet switching angles are 10 in a quarter-wave, hence the maximum number of pulses in a half fundamental period is 10, and the highest switching frequency is then 20 multiplied by the fundamental frequency. The lowest number of discreet switching-angles understandable by the model in each quarter-wave is 3.
- Programmed Modulation switching-patterns can be stored in the m-file "*PatternSelector*" belonging to the Pattern Select block. Initial stored switching-patterns are optimized towards minimizing the phase current with WTHD0 as the minimization criteria.

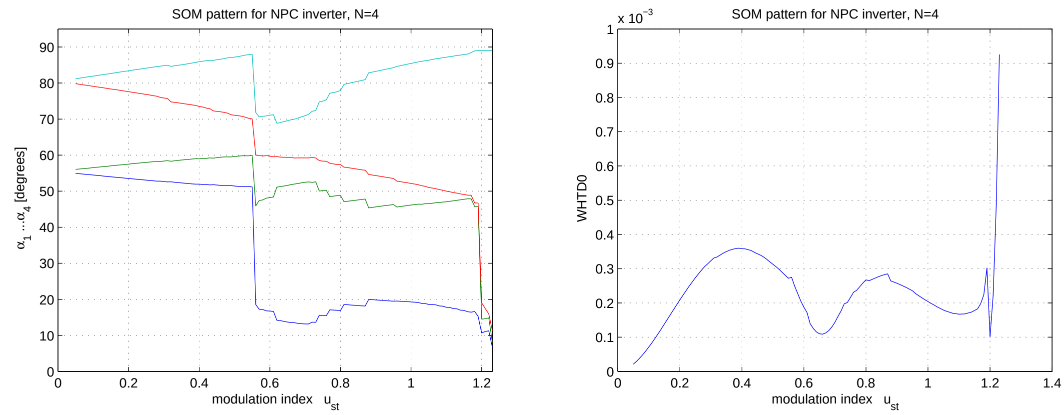
Maximum switching events within a quarter-wave is chosen to be 10, this corresponds to a switching frequency of 1000 Hz. A higher switching frequency than that will not be of any particular interest since other modulation strategies then will give good results with respect to power quality and drive control [24].

3.1.2 Calculation of Optimal Pulse Width Patterns

SOM pulse patterns and SHEPWM patterns have been calculated in MATLAB by the use of the *fmincon* function and *fsolv* function, respectively. Only SOM patterns was calculated to the extent necessary to test the drive system model. For this purpose, Pulse patterns with N equal 3 to 8 have been calculated over the modulation index range of 0.05 to approximately 1.2, with a resolution of 0.01. These have been implemented into the Pattern Selector block m-file(file name: *PatternSelector*) and are illustrated with WTHD0, harmonic spectre and switching-angles plots in appendix C.

To calculate the SOM patterns was an algorithm developed by Roy Nilsen in Wärtsilä used. The original algorithm was slightly expanded to calculate odd numbers of switching angles and to be able to implement several initial values. The use of several initial switching values makes it possible to "jump" between optimal trajectories, or in other words, local minimals.

There are several local minima in the optimization calculation of SOM patterns, as mentioned in section 2.2.3 does it not exist a straight-forward method to generate initial angular values that hits the most Optimal solution in the optimization process. The "cut and try" method was used to obtain the SOM patterns, and therefore is it not guaranteed that it is the most optimal solution.



(a) Angular Switching instants with respect to the modulation index

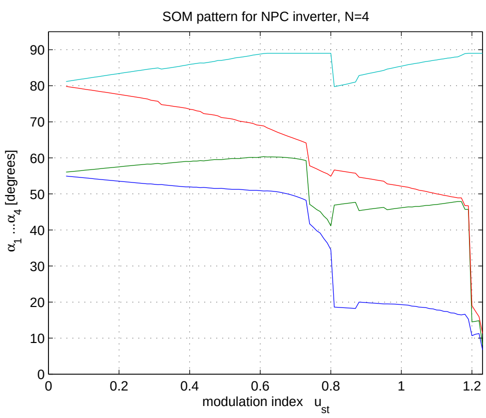
(b) WTHD0

Figure 3.3: SOM switching-pattern with corresponding WTHD0, N=4, u_{st} is the same as the modulation index m.

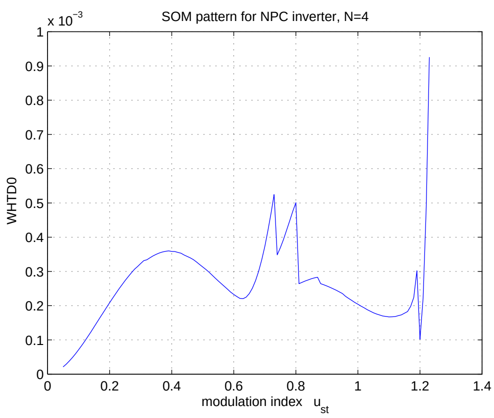
Figure 3.3 shows the plot of both the angular value of optimal switching events with respect to modulation index 3.3(a) and the corresponding WTHD0 response 3.3(b).

The optimization algorithm must be given an initial switching-angle guess to start the process. In figure 3.3(a) the values 55, 57, 80 and 82 was chosen as initial switching-angles for a m equal to 0.05, which is the first value for m. The algorithm will use the previous solution as the next initial guess for each iteration, e.g. the solution of m equal to 0.05 will be the initial guess when calculating the switching-angles for m equal to 0.06.

An observation from the "cut and try" process was that the solutions would often follow a "minimum" trajectory due to this, initial guess for the next iteration, method in the algorithm. In figure 3.3(a) a new set of initial values is forced into the optimization process at m equal 0.55. This is clearly visible in the switching angle plot where a "jump" can be seen. If a new set of initial values had not been forced into the optimization process at 0.55, a non optimal trajectory of solutions would be followed, this is illustrated in figure 3.4. The "jump" would not happen before m equals ca. 0.72 and then, to a new non-optimal trajectory.



(a) Angular Switching instants with respect to the modulation index [13]



(b) WTHD0 [13]

Figure 3.4: SOM angular switching pattern with corresponding WTHD0, N=4, non optimal. u_{st} is the same as the modulation index m.

3.2 Discreet Operator Blocks

MATLAB Simulink, including the simPowerSystems tool box, do not have any standard Programmed Modulation blocks that can be used to execute the required discreet operations in the

- Pattern Selector
- Stator Flux Error Estimator
- Stator Flux Trajectory Controller in override mode or synchronous mode
- Modulator

MATLAB Simulink offers an alternative, which is a "user defined function" called S-function. There are two types of S-functions available, an older type, called S-functions level 1, and a new type called S-function level 2. The difference is mainly that the level 2 type is easier to build/write and use. To realise the necessary model blocks listed by bullet points above in the SOM drive system model, S-functions level 2 was used for the Modulator, Stator Flux Error Estimator and the Stator Flux Trajectory Controller. The Pattern Selector was written in a *interpreted MATLAB function*.

The level 2 S-function can have separate sampling times for the inputs and outputs [25], variable sampling time and other beneficial features that made it the natural choice among different user-defined blocks in Simulink.

The different designed model blocks have been separately tested in a model called the Test Bench, especially the Stator Flux Trajectory Controller was thorough tested. The Test Bench is shown in figure 3.5, simulation results from the commissioning is shown in section 3.2.3.

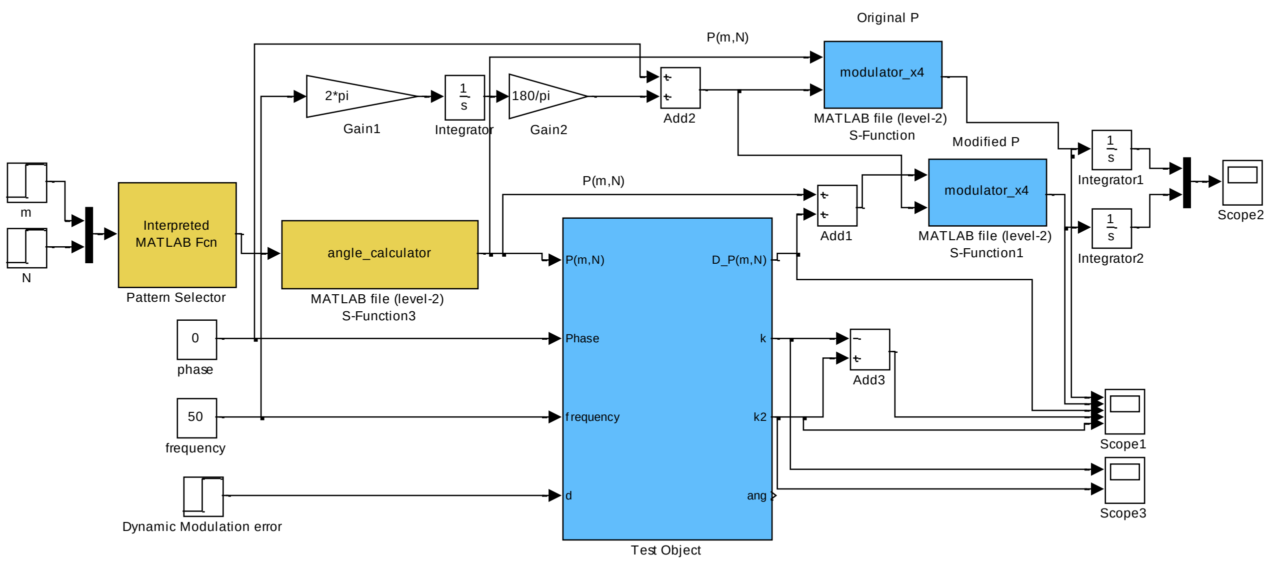


Figure 3.5: Block diagram of the Test Bench model.

3.2.1 Pattern Selector

The Pattern Selector is an *interpreted MATLAB function* which is a Simulink block where the inputs and outputs are processed by a user defined function [26]. The user defined function can be inserted directly into the "Function Block Parameter" window, or a m-file script can be written and addressed in the same window. The Pattern Selector block has an appurtenant m-file . All SOM pulse patterns, mentioned in section 3.1.2, are saved inside this m-file in matrices, one matrix for each N. Dependent on the input parameters m and N will this block give an output vector with 3 to 8 discreet switching angles, but always with 10 elements. There have not been generated switching patterns for N=9 and N=10, but the m-file script is prepared to have that.

The m-file script for this block is found in appendix D.3. In the appendix are the SOM pattern matrices removed since they would take a lot of space, the pattern matrices are found in the m-file *patternselector*.

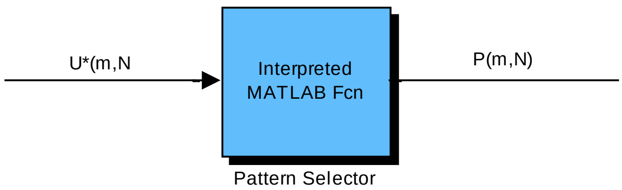


Figure 3.6: Pattern Selector, a user defined MATLAB function.

3.2.2 Optimal Flux Trajectory Estimator

The dynamic modulation error d is estimated in the Stator Flux Error Estimator subsystem, see figure 3.7. To find d an estimation of the real stator flux-linkage Ψ_s, and a calculation of the optimal stator flux-linkageΨ_{ss} is needed. In the model presented in this thesis the real stator flux-linkage is assumed measured, the optimal stator flux-linkage is calculated in the *Optimal_flux2* level 2 s-function which can be seen in the subsubsystem surrounded by a black square in figure 3.7. The level 2 S-function code for this block is found in appendix D.4.

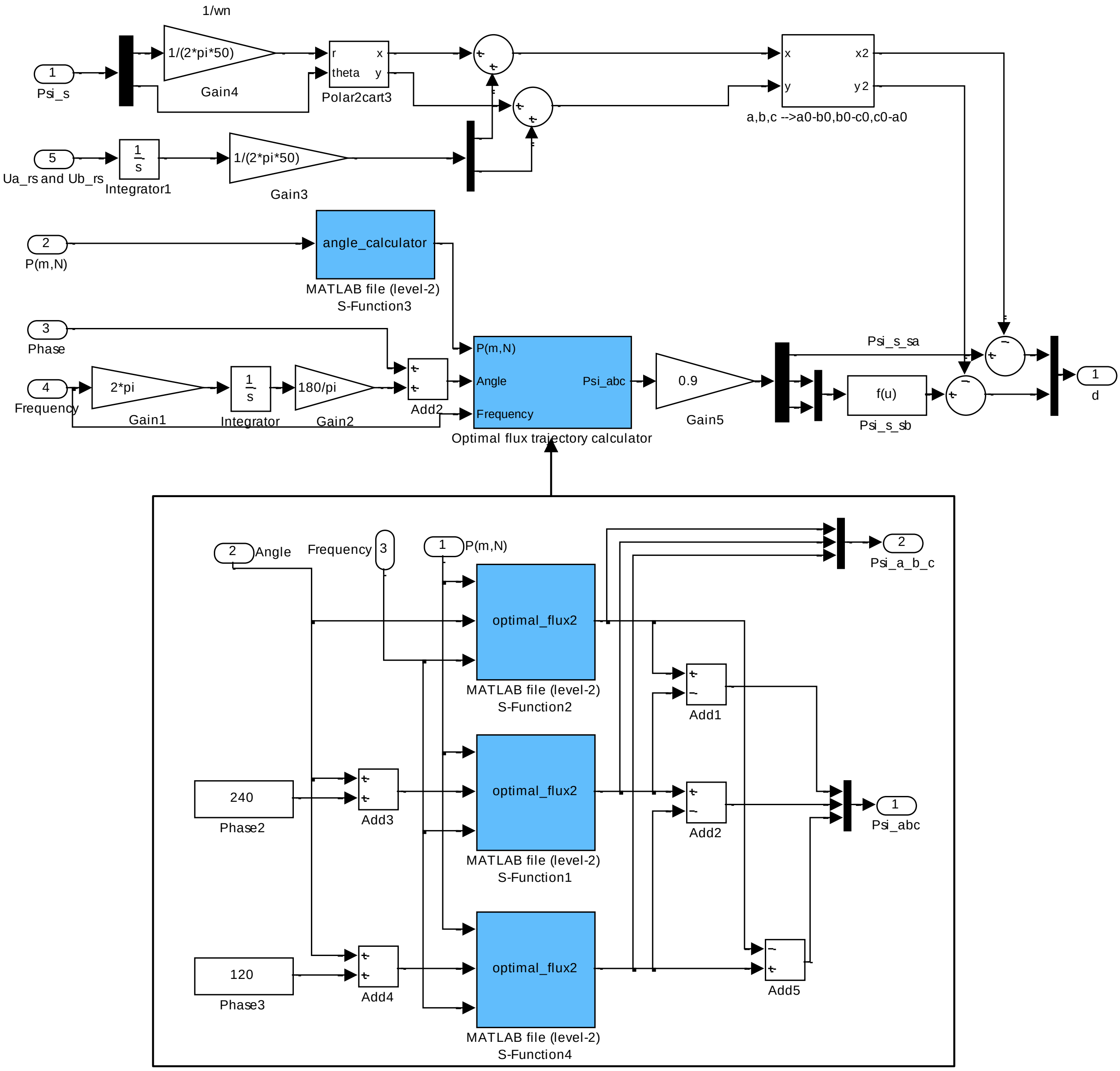


Figure 3.7: The subsystem illustrating the Stator flux trajectory error estimator and optimal flux trajectory calculator

In a transient situation where a new SOM pattern is required to operate, a new optimal stator flux-linkage trajectory also applies, given by the new voltage pattern. This stator flux-linkage develops according to equation 2.23 in the theory chapter. The *optimal_flux2* S-function written to generate the optimal stator flux-linkage trajectory, at all times, calculates the $\Psi_{ss}(m, N)$ according to equation 3.1. This is the same, just with other notations and it uses the angular position instead of time.

$$\begin{aligned} \Psi_{ss} &= \int_{0^o}^{Ad} u_{ss}(m, N) d(Ad) \\ Ad &= \int_0^t \omega_{ss} dt - 360 * pc + \theta \end{aligned} \tag{3.1}$$

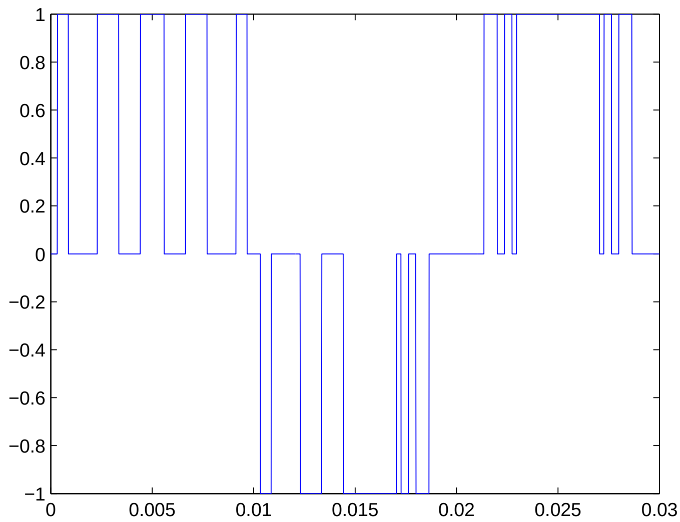
Where Ad is the angular driver, pc is an integer periodic counter that starts at 0, $u_{ss}(m, N)$ is the converter bridge-leg voltage,e.g. u_{a0}, and θ is the voltage reference phase angle. Equation 3.1 generates the flux-linkage trajectory for phase a. Phase b and c is calculated in the same manner but with a 240 and 120 phase-shift, respectively.

Figure 3.8(b) shows how the *optimal_flux2* S-function calculates the new optimal flux trajectory instantly when a patterns change is executed. In the simulation, the SOM pattern-change is executed at t=0.015 s, N is equal to 5 for both pattens and the modulation index m increase from 0.6 to 1. The voltage waveform for the same time interval is seen in figure 3.8(a).

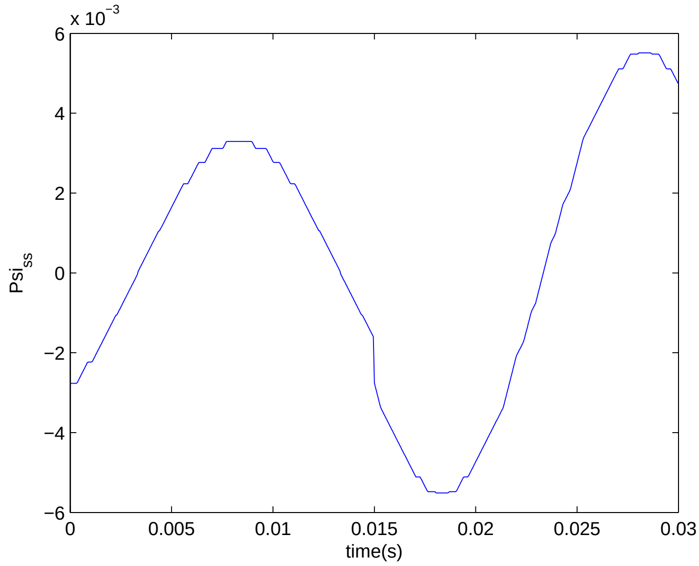
Shown in figure 3.7, the optimal stator flux trajectory in line-to-line values($a0 - b0$ and etc.) is given as the output from the Optimal Stator Flux Trajectory Calculator. This signal is then multiplied by a constant that adjusts the amplitude of the trajectory such that a steady-state deviation becomes, almost, equal to zero. The next step is to convert the instantaneous optimal line to line stator flux-linkage values to α and β components, stationary coordinates, the same as the "measured" stator flux in the induction machine.

Note that in the model block diagram in figure 3.7 the voltage-drop integral over the stator winding resistor is added to the measured stator flux. The reason for this is that if the modulation model gets expanded to include an induction machine model to estimate the rotor and stator flux vectors, then it do not need to consider the minor influence of the stator resistance. The stator flux trajectory is parameter independent [2].

The blue block in the upper left corner in figure 3.7 is the Angle Calculator, this generates the switching angles for the full fundamental period from quarter-wave switching angles. The S-function programming code for the *angle_ calculator* level 2 S-function is found in appendix D.1.



(a) PWM voltage waveform for converter bridge-leg a0. The modulation index m change from 0.6 to 1 at t=0.015 s.



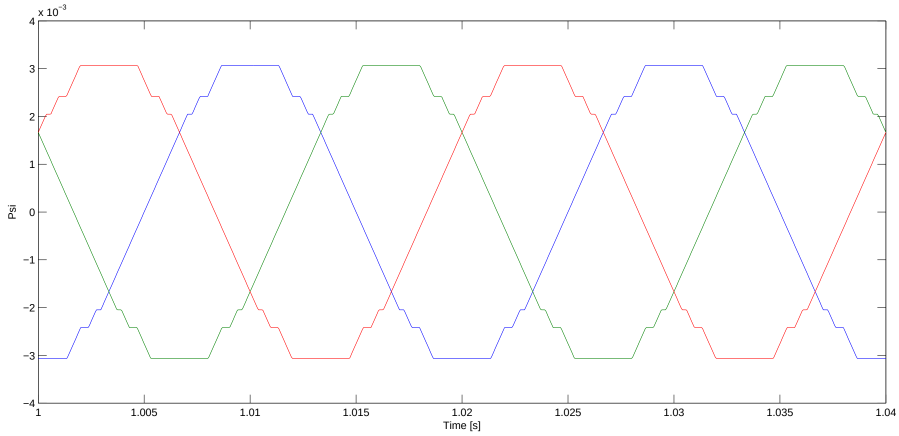
(b) Optimal stator flux trajectory $\Psi_{ss,a0-b0}$ calculated from the optimal voltage pattern in use. The optimal flux trajectory change instantly as the voltage pulse pattern change.

Figure 3.8: Virtual optimal converter flux-linkage and voltage pattern. A exchange of SOM pattern happens at t=0.015 s.

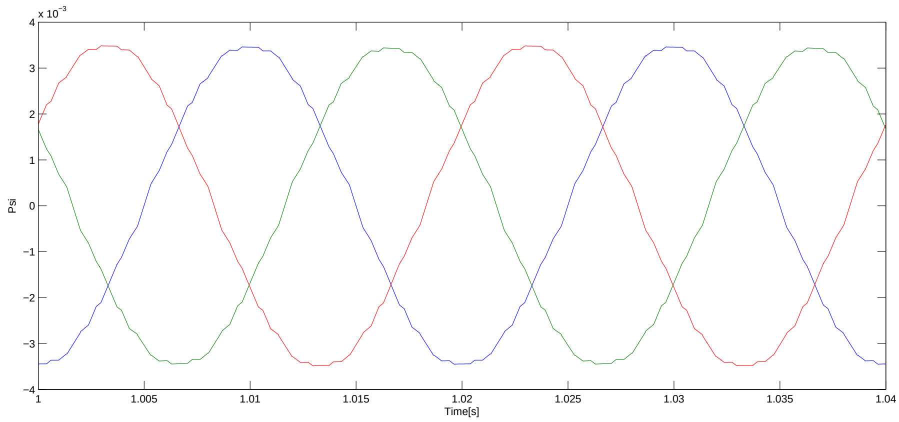
Voltage and flux relation between converter bridge-leg and load phase values

The optimal stator flux trajectory is derived from the optimal converter bridge-leg voltages(a0, b0 and c0), and the actual stator flux, assumed measured inside the motor, is given by the induction machine phase voltages(a, b and c). The converter bridge-leg voltages is not equal the induction machine phase voltages due to the fact that the converter can only control the line-to-line load voltages [13]. See appendix B for converter bridge-leg and phase voltage relations.

The difference between the converter bridge-leg based optimal stator flux, and the actual phase stator flux measured in the induction machine is illustrated in figure 3.9 by comparing (a) and (b). Simulation results are from steady state operation, which was obtained after 1 s.



(a) Calculated optimal converter bridge-leg fluxes Ψ_{a0}, Ψ_{b0} and Ψ_{c0}, derived from converter leg voltages. The y-axis indicates the voltage integral in pu, x-axis indicates simulation time.



(b) Actual stator flux values Ψ_a, Ψ_b and Ψ_c. The y-axis indicates the voltage integral in pu, x-axis indicates simulation time.

Figure 3.9: Virtual optimal flux and induction machine phase flux waveforms

The block diagram of the Stator Flux Trajectory Error Estimator includes a transformation block. This transformation block, enlarged in figure 3.10, is used to transform the measured stator flux phase values (Ψ_a, Ψ_b and Ψ_c) from the induction machine to virtual converter bridge-leg line-to-line stator fluxes (Ψ_{a0-b0}, Ψ_{b0-c0} and Ψ_{c0-a0}). The voltage relations between the converter bridge-leg voltages and the load phase voltages is used in the transformation block, but as flux relations instead. The utilized relation is presented here in equation 3.2

$$\begin{aligned}\Psi_{a0b0}(t) &= \Psi_a(t) - \Psi_b(t)\\ \Psi_{b0c0}(t) &= \Psi_b(t) - \Psi_c(t)\\ \Psi_{c0a0}(t) &= \Psi_c(t) - \Psi_a(t)\end{aligned} \tag{3.2}$$

Inside fnc 1-3, in figure 3.10 below, are the above expressions implemented. This transformation is needed to achieve an accurate comparison of the actual and optimal stator flux, which leads to a correct calculation of the dynamic modulation error d.

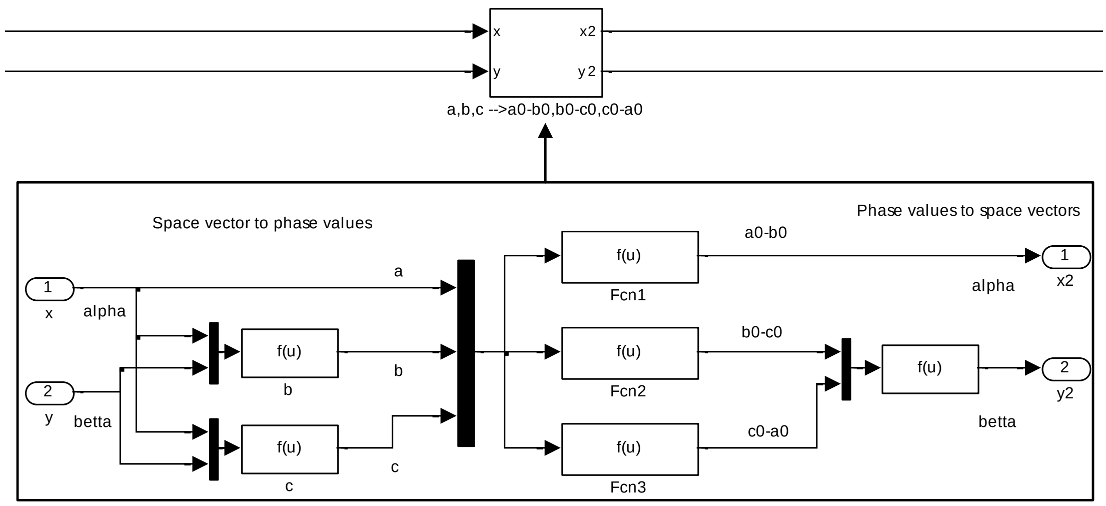


Figure 3.10: Phase(a,b and c) to converter(a0-b0, b0-c0 and c0-a0) transformation

Simulations verified that the transformer worked. Measured stator flux trajectory became equal to the calculated optimal stator flux trajectory, in steady-states, with the SFTC disconnected. Since the dynamic modulation error is calculated from line-to-line values will the error vector be phase-shifted by 30 degrees with respect to the phase values. Since the manipulations are done in the converter bridge-leg voltages, which are in phase with the phase voltages, should the dynamic modulation error be corrected. This is done by a multiplication of $1/\sqrt{3}$ and phase shift of -30 degrees.

3.2.3 Stator Flux Trajectory Controller

The Stator Flux Trajectory Controller is the "hart" of the switching-pattern based drive system due to its essential functions. That is to keep the drive system on an optimal trajectory and implement changes in the stator flux to control the electromagnetic torque. Without the SFTC can a pattern-exchange result in long transient over-currents, this effect is shown in the Simulation chapter later on.

The theory chapter explains how modifications of the optimal voltage pattern can be used to remove dynamic modulation error and how an implementation of $\Delta\Psi_s$ can change the electromagnetic torque and control the magnetization. In the simulation model presented in this thesis, two level 2 S-functions (Two operational modes) have been written to calculate the necessary modification on the optimal SOM pattern ΔP. Named override mode and synchronous mode. The names reflect how the mode operate when implementing large $d + \Delta\Psi_s$ vectors. Figure 3.11 shows the SFTC in the Synchronous optimal modulation system.

The Stator Flux Trajectory Controller subsystem is shown in figure 3.12. The $d{+}\Delta\Psi_s$ signal is converted from α and β coordinates, stator oriented, to instantaneous phase values. Hence, a wanted volt/sec change is achieved for each phase, which is easy to use and understand when altering the optimal switching pattens in seconds or angular value.

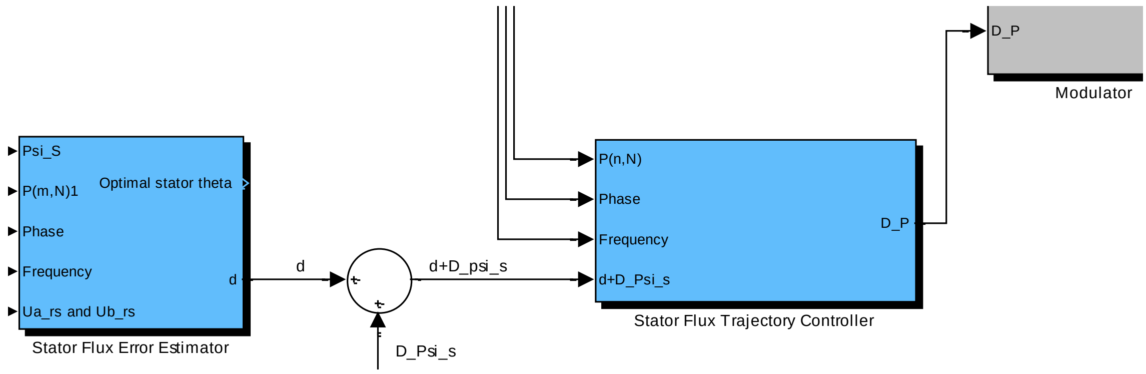


Figure 3.11: Stator Flux Trajectory Controller illustrated in the model block diagram

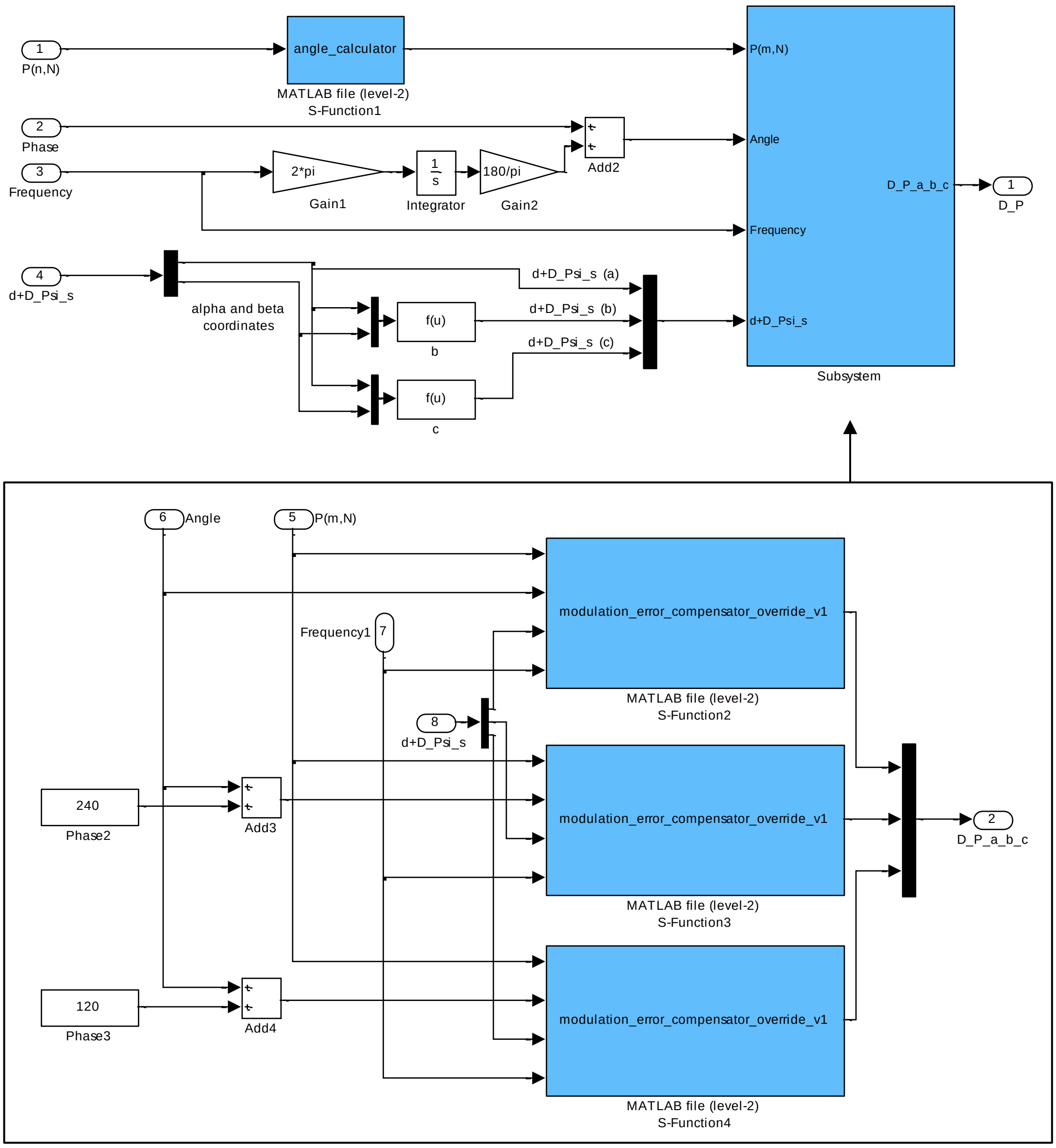


Figure 3.12: Stator Flux Trajectory Controller subsystem

Stator Flux Trajectory Controller - how it operates

This is an explanation on how the the S-functions *Stator_flux_error_compensator_override_v1* and the *Stator_flux_error_compensator_syncronous_v1* operates in general. First considering the operational part that is common for both modes. The difference between the two modes are discussed in later. The S-functions programming code is found in appendix D.5.

The S-functions receives the $d + \Delta\Psi_s$ signal, which is given in $[Vs]$ pu, for each phase. This $[Vs]$ pu signal is then transformed to a $[V \cdot deg]$ pu signal by equation 3.3. This transformation is done since the block is driven by a driving angle Ad, the same way as previous s-function blocks, and all the discreet switching angles are given in angular values.

$$d(V \cdot deg) = d(Vs) \cdot 360(deg) \cdot f(s^{-1}) \tag{3.3}$$

The S-function programming code operates according to the flowchart in figure 3.13.

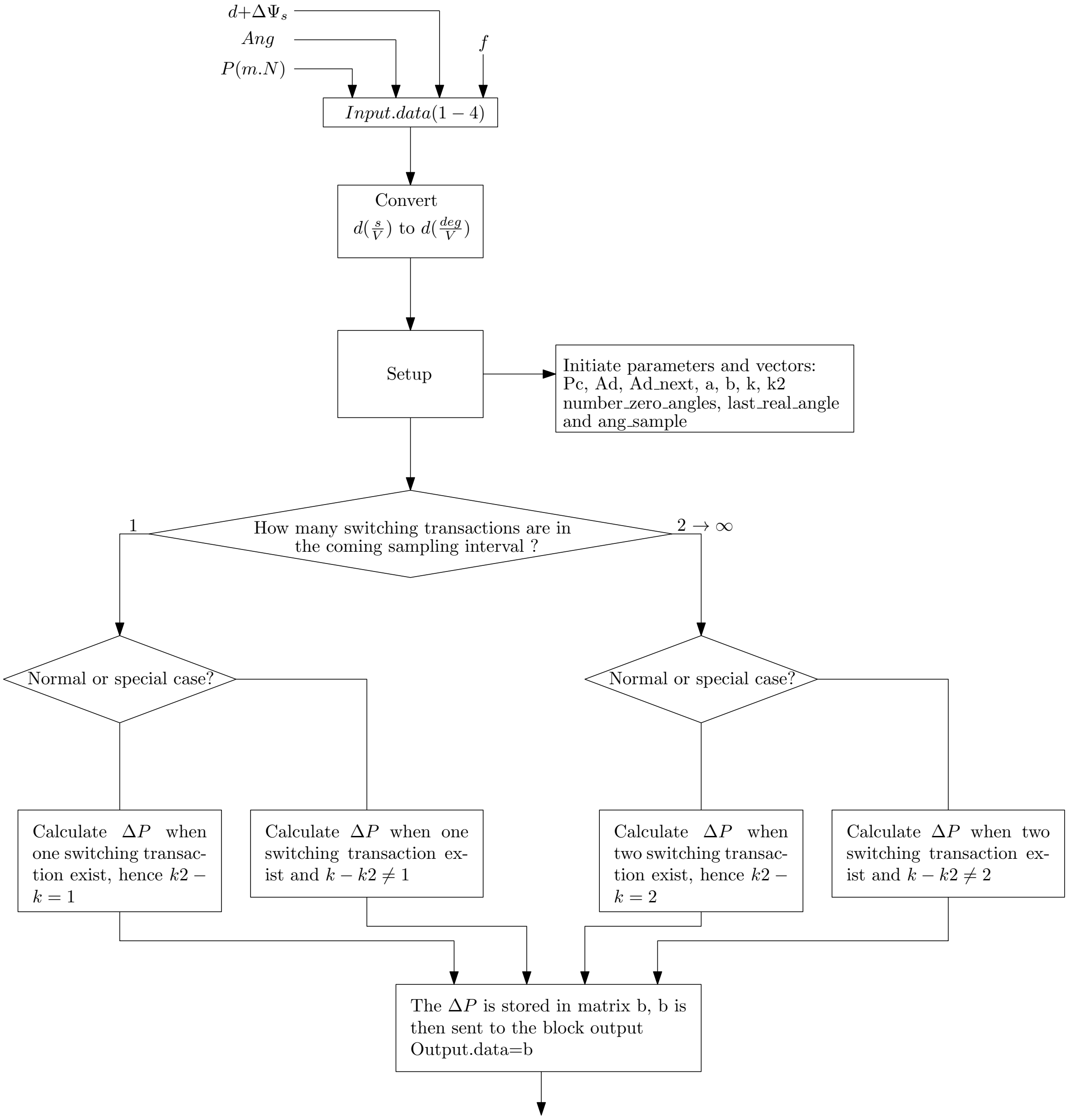


Figure 3.13: Algorithm flowchart for the S-functions in the Stator Flux Trajectory Controller

Parameter description:

Pc
Periodic counter, this parameter counts each time the *ang* input pass 360 degrees. Gives integer multiples of 360 degrees.

Ad
Angular driver given by *ang* and *Pc*. Ad is always between 0 and 360 degrees, and represent the angular starting point in a sampling period.

ang_sample
Is the angular sampling time of the S-function block.The sampling period can be adjusted by changing the $ang_s ample$ parameter in the S-function code, found as the first constant in the S-function $output\ function$ section. The sampling time is also given in degrees, this way will the width of the sampling time always be constant with respect to the length of a fundamental period. It is possible to chose a constant sampling time, this is explained in the programming code

Ad_next
Ad_next is given by *Ad*+*ang_sample*, and is therefore the last angular value of a sampling period. Hence, within a sampling period, the start and the end of a sampling period is given by *Ad* and *Ad_next*, respectively.

a
This is a 1×40 vector with all the optimal SOM switching instants, retrieved from input 1.

b
This is a 1×40 vector where all the calculated pattern changes are stored.

k
This is a counter which counts every time the *Ad* passes a switching angle, e.g. when *Ad* is 40 degrees and α_3 is 39, then the k will be equal to 3.

$k2$
Similar to k but use *Ad_next* angular position instead. The counters k and $k2$ are used to determine the number of switching transactions at the beginning of a sampling interval by $k2$-k=switching events during the coming sampling interval.

number_zero_angles
This parameter quantifies the number of switching angles, from 0 to 10, that have an angular value equal to zero. Since the system consider switching angles that are equal to zero to be "non-existent" is this quantity needed in some special cases. Special cases can occur at the end of a period or when the counters k and $k2$ "jumps" over zero angles.

last_real_angle This parameter indicates the number of actual switching angles in the pattern, in operation. This is needed for the same purpose as the *number_zero_angles*.

Figure 3.14 illustrates how discreet switching angles are moved in different directions to either increase or decrease the voltage integral of the converter bridge-leg voltage (thus indirectly changing the Stator flux). Some of the SFTC parameters are included in the figure to show how the S-function use them when deciding its next move. The arrows, red and blue, will be the output ΔP in each of the two examples in the drawing.

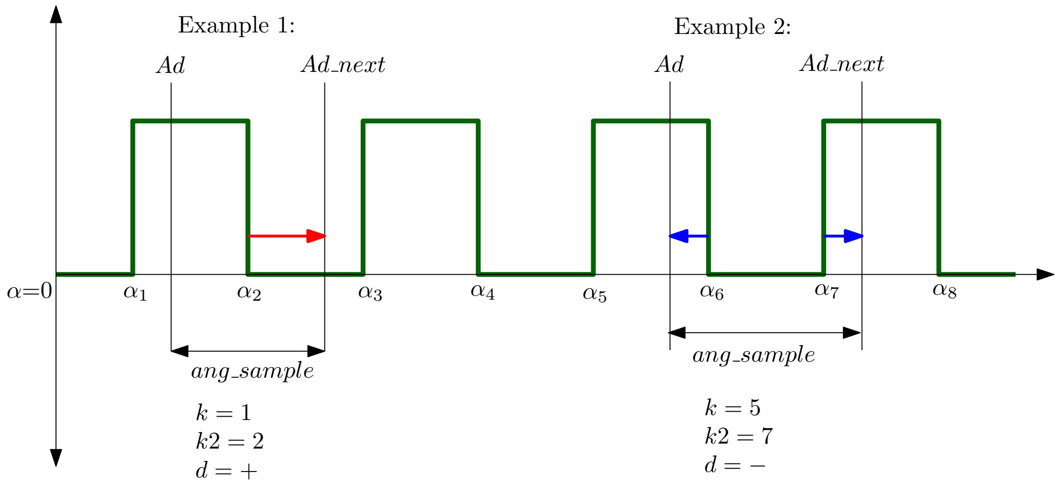


Figure 3.14: Illustrations on how switching angles are moved in phase *ao*. Example 1: the sampling interval overlaps one switching instant, example 5 overlaps two switching instants.

The maximum number of switching angles that can be modified within a sampling period is limited to 2 in this controller. This was decided to be sufficient after testing the Stator Flux Trajectory Controller with different sampling intervals in the Test Bench model. $P(1, 8)$ at f=50 Hz operation was chosen since this is a pattern with many switching instants and will therefore more easily have sampling intervals that overlaps several switching instants. The simulations data is presented in figure 3.15 and shows the converter bridge-leg voltage integral of altered voltage waveforms, where different sampling intervals(*ang_sample*) have been used.

In the simulations, a constant d value of 0.001 was initiated after 0.01 s as a dynamic modulation error, seen in the figure 3.15 as the voltage integral value increase after 0.01 s. How fast the SFTC can increase(which is the case in the simulation) the voltage integral give an indication on how fast the SFTC can change the stator flux in an induction machine.

Simulation results show that a larger sampling interval than 14 degrees will have a decreasingly effect on how efficient the SFTC is when operating with this particular Pattern. This is because the sampling interval overlaps three or more switching instants and losses its efficiency, since it can only move two of the discreet switching angles.

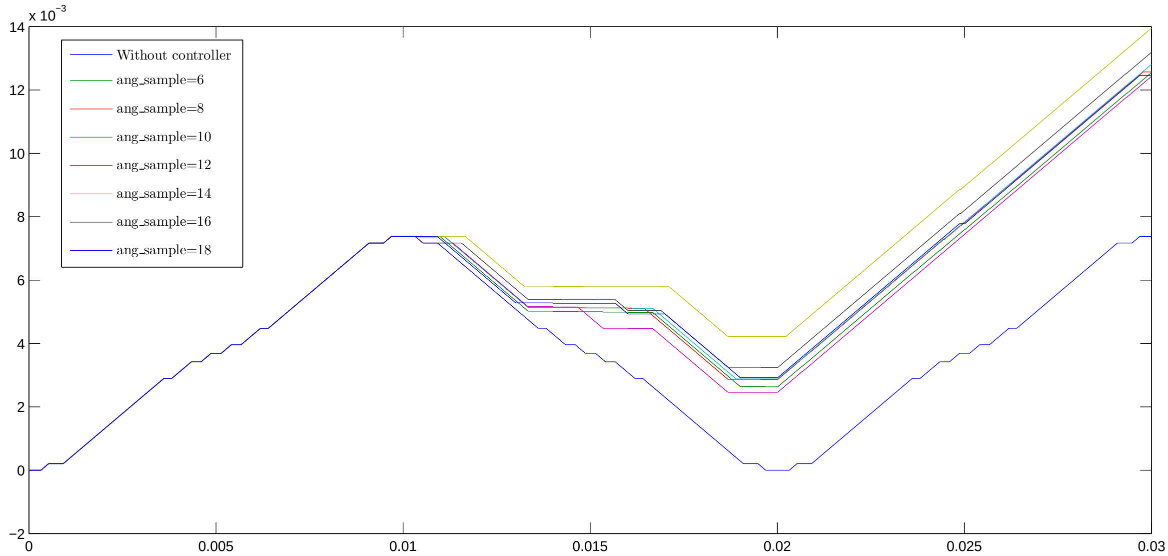


Figure 3.15: Comparison of different sampling interval durations

An ang_sample of 14 degrees corresponds to a sampling time of 0.77 ms at 50 Hz. The recommended sampling time interval is 0.5 ms in [2], and that is with a N equal to 5 with a 200 Hz switching frequency. Hence, it is sufficient to "only" be able to utilize 2 switching instants per sampling period.

As mentioned can the SFTC sampling be set as a constant angular value (frequency dependent) or as a constant sampling time. With a constant angle will the sampling interval be longer, in time, at low frequencies. This will increase the chance of covering a switching instant at lower frequencies, but at the same time slow down the reaction time. A large sampling interval will result in a slower SFTC in dynamic operations, e.g. when a dynamic change happens at Ad^+ will the SFTC not be able to implement any countermeasures until after Ad_next.

Sampling with constant time intervals will give short angular samplings at low frequencies and longer angular sampling intervals at high fundamental frequencies. This corresponds good with the fact the switching patterns at low fundamental frequencies will have a higher number of switching instants then at high fundamental frequencies. Se section 2.4

Override Mode

The difference between override mode and synchronous mode will not have any effect until sampling interval scenarios occur that can result in elimination or merging of converter bridge-leg voltage pulses.

The override mode will in such a scenario eliminate or merge voltage pulses to force the stator flux onto the optimal trajectory. This effect is clearly illustrated in figure 3.16 where a constant d is acting on the SFTC in override mode.

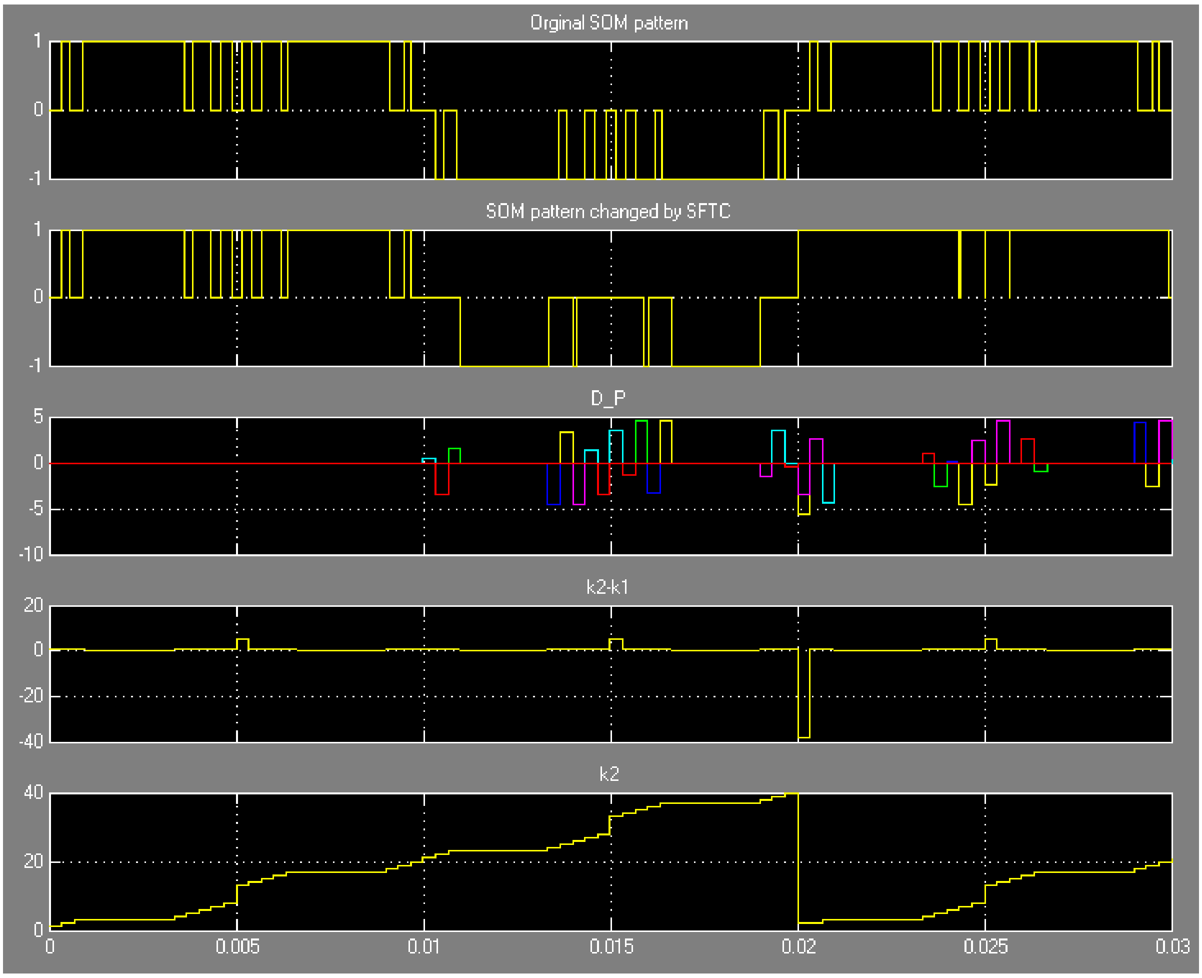


Figure 3.16: Simulation results from the SFTC testing in the Test Bench model, sampling only overlaps one switching instant.

The uppermost diagram illustrates the original $P(1,8)$ pattern, diagram number two shows the modified voltage waveform $(P(m,N)+\Delta_P)$ by the SFTC in override mode, for e.g. phase $a0$. The third diagram in figure 3.16 show the pattern modifications, where the y-axis indicates the angular move in degrees. In this test a d equal to 0.00025 Vs $[pu]$ was used which corresponds to a angular modification of 4.5 deg. The sampling interval ang_sample was sett to 6 deg.

The test indicates elimination of some PWM-voltage pulses. This is not due to an sampling interval that overlaps a whole pulse, the pulses get eliminated because two switching angles have been moved to the same value. E.g On and Off switching at the same time. There are not any sampling intervals that covers two switching instants at once in figure 3.16, that is illustrated in figure 3.17.

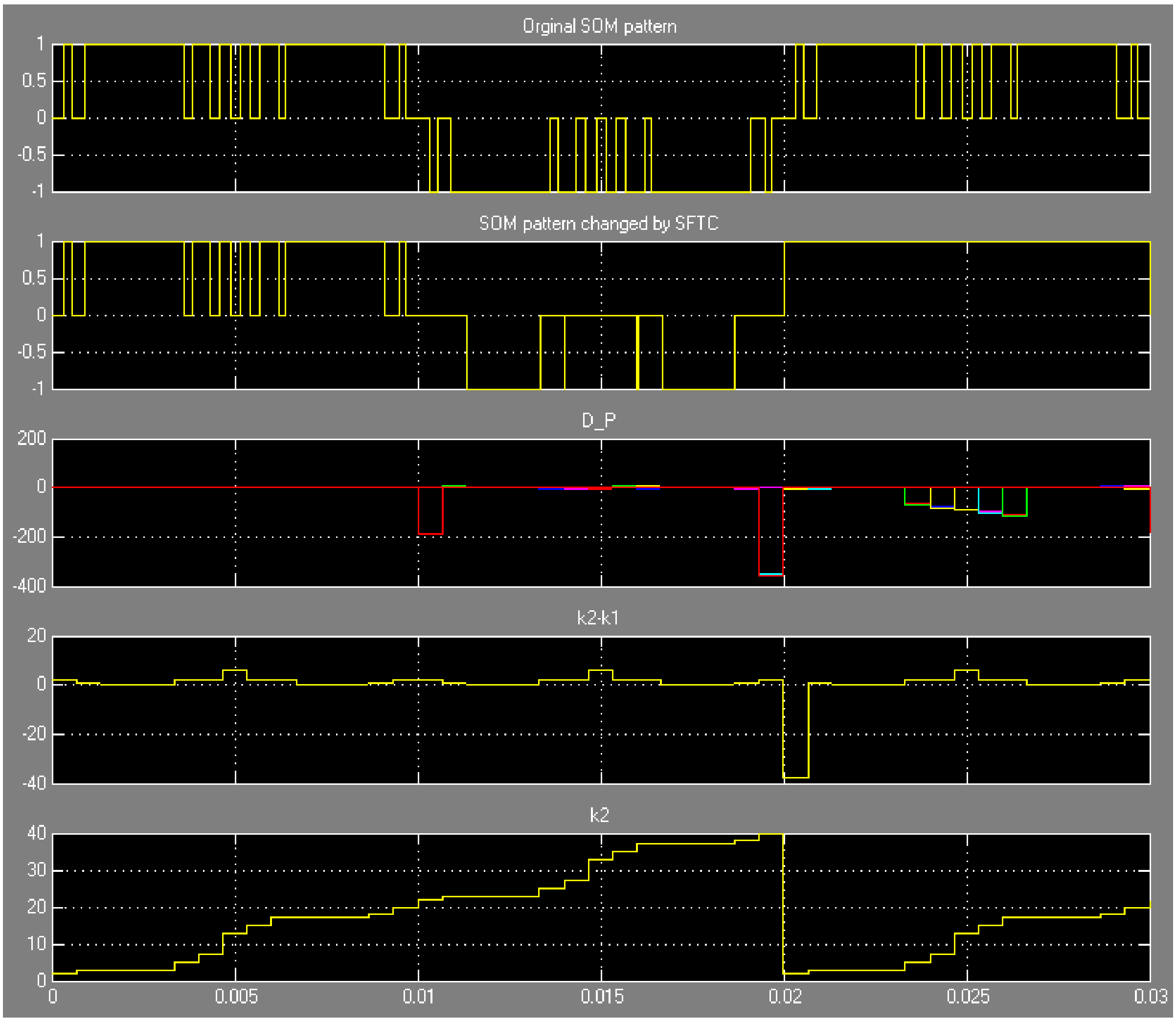


Figure 3.17: Simulation results from the SFTC testing in the Test Bench model, sampling interval covering two switching instants

By inspecting the D_P (ΔP), the middle scope window above, switching angles are being removed from the original $P(m, N)$ by setting the specific switching angle to zero in the final altered pattern. Hence the D_P has a equal angular value for the two specific switching angles, but negative and therefore gets cancelled out.

The advantage with the override mode is that it will give a fast change in the converter bridge-leg voltage integral, thus also be able to change the stator flux linkage fast. There have been thought of two disadvantages. The first is that the system can lose its synchronisation between the switching frequency and the fundamental frequency during dynamic operations due to elimination or merging of voltage pulses. Second, if during a sampling interval where the SFTC is sending an ΔP which holds two negative switching values(this angular value can be high if it is to set two of the last discreet switching angles to zero) to eliminate or merge a voltage pulse, and then the original $P(m, N)$ change. From the time instant that the $P(m, N)$ change, and until the end of the sampling period, can this result in distortions of the newly employed $P(m, N)$ that do not contribute to changing the converter bridge-leg voltage integral in the right direction.

Synchronous Mode

As the mode name indicate, will the SFTC in synchronous mode always have converter bridge-leg voltages that have a switching frequency that is synchronous to the fundamental frequency. This is done by removing the override function in the S-function programming code, and adding a small and adjustable "buffer" between the discreet switching angles in the modified SOM pattern $(P(m, N) + \Delta P)$. Se simulation results in figure 3.18. This buffer is found in the set-up section in the S-function algorithm.

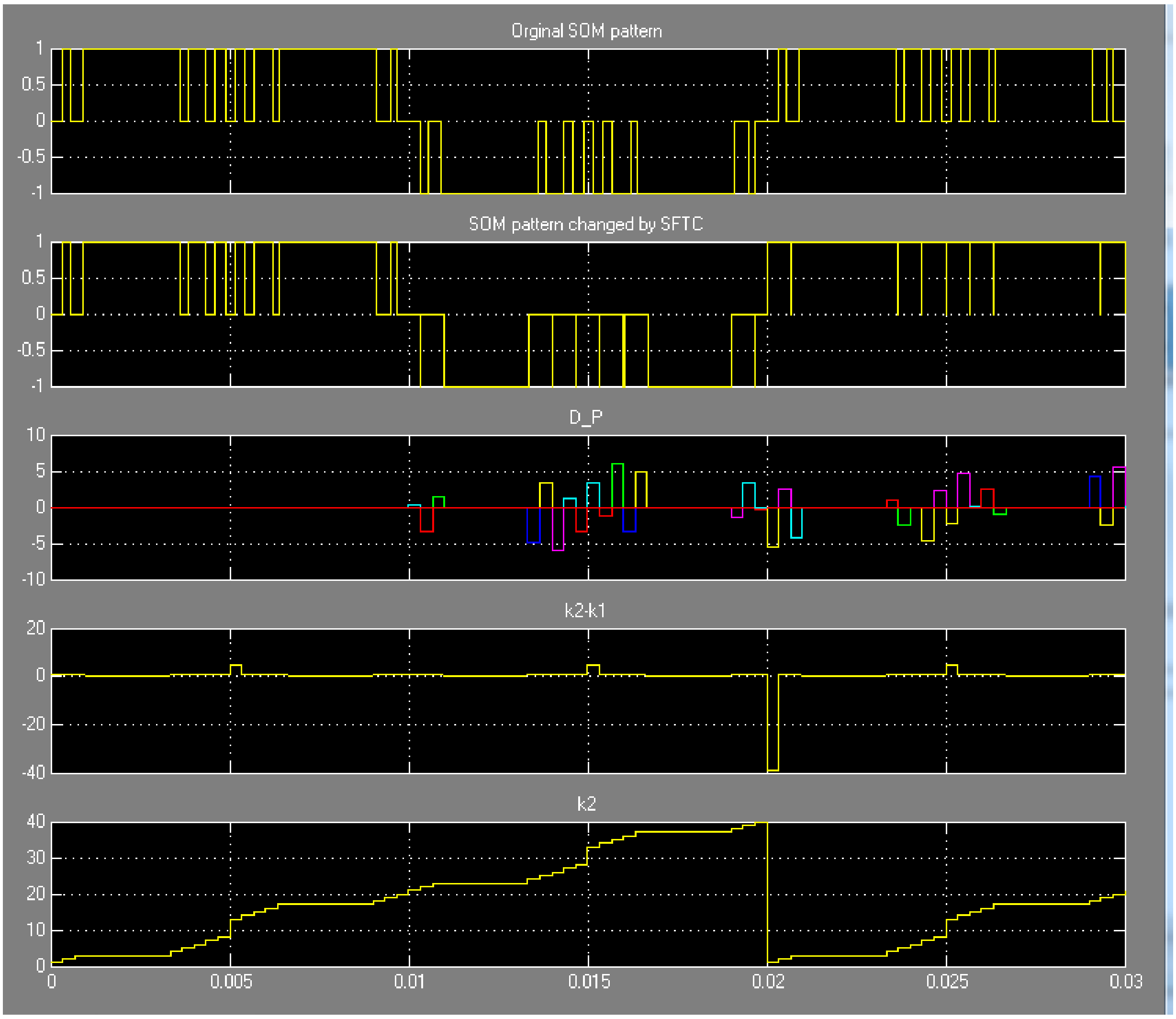


Figure 3.18: Simulation result of the SFTC in the Test Bench model, synchronous mode.

As the SFTC test indicates, is the converter leg voltage pulse pattern synchronous to the fundamental voltage in dynamic operations. The simulation was done with the same parameters and d value as in figure 3.17. The disadvantage that could occur when large negative(switching instant eliminating) angular values where used in the ΔP, in override mode, can not occur in the synchronous mode, since a switching instant is never removed, only moved.

3.2.4 Modulator

The interface between discreet switching-patterns and converter gate-signals is the *Modulator*. The optimal pattern $P(m, N)$ from the Pattern Selector block and ΔP from the Stator Flux Trajectory Controller is added together and forms the pulse pattern which is to be modulated. This can be seen in the upper part of figure 3.19. After the angular addition is the discreet pattern sent to the level 2 S-function $Modulator\_x4$ which generate the optimal voltage patten, one for each converter bridge-leg, this voltage pattern is then transformed into gate-signals before it is delivered to the output of the modulator subsystem. The Modulator_x4 MATLAB programming code is found in appendix D.2

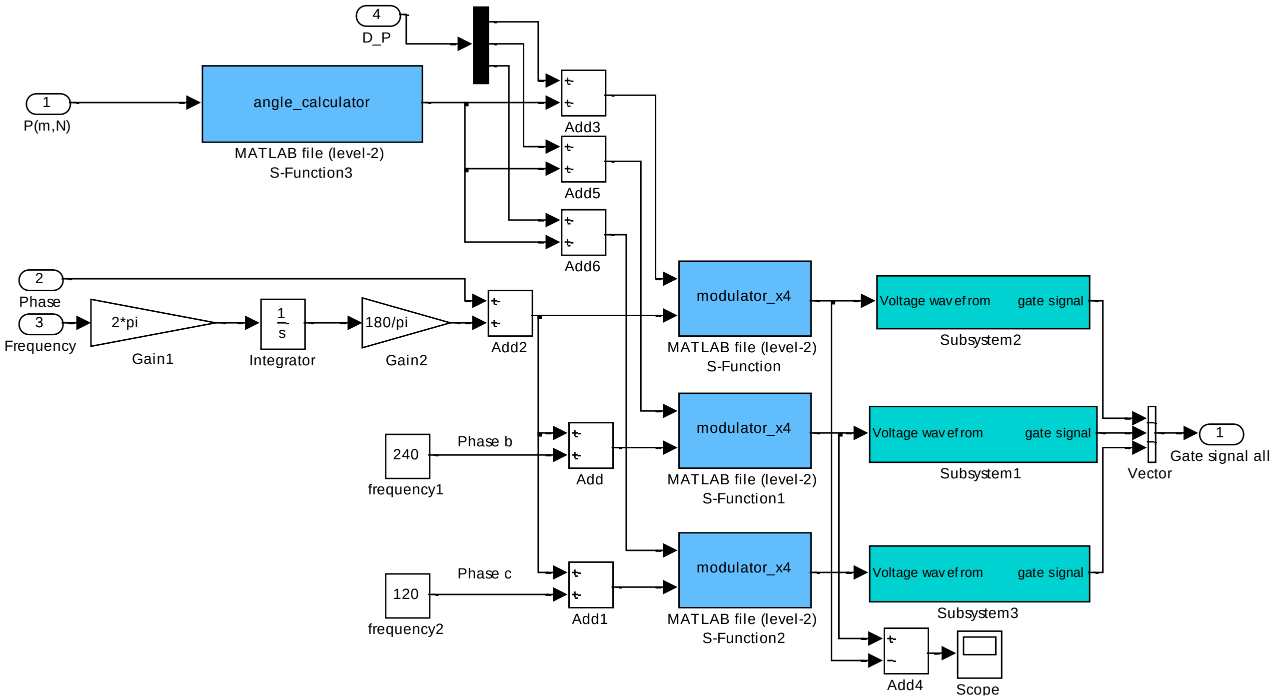
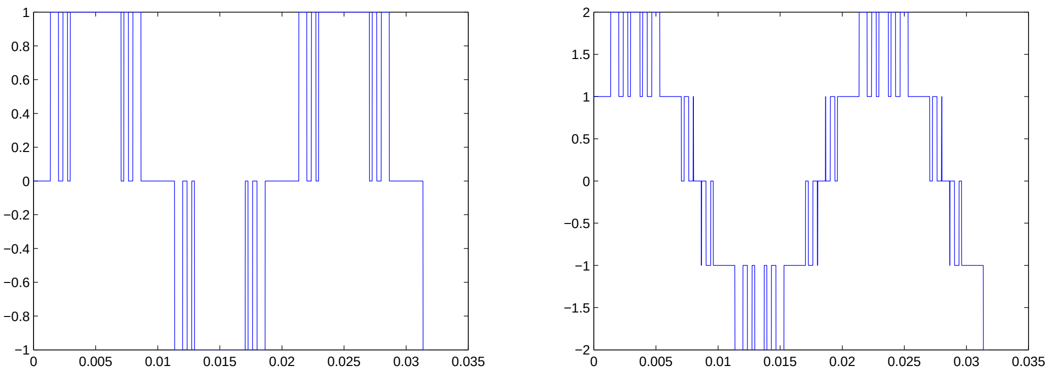


Figure 3.19: Modulator subsystem.

The Modulator block have two inputs, the discreet PWM pulse pattern values which is a 1×40 vector and the driving angular value. This driving angle is generated by integrating the reference electrical frequency $\omega_s s$, 240 and 120 electrical degrees are added to modulate phase b and c, respectively. Figure 3.20(a) illustrates the output waveforms from the Modulator and the line-to-line PWM signal when a SOM pattern with N equal to 5, and a modulation index m equal to 1 are used.

The PWM waveforms illustrates how the output voltages from the converter should be. Harmonic content in the PWM patterns can be measured in-between the modulator and the *PWM to gate-signals* blocks, in steady-state or dynamic operation.



(a) PWM waveform signal for converter leg, a0

(b) PWM line-to-line PWM waveform, a0-b0, which has the same waveform as the line-to-line voltage U_{ab}.

Figure 3.20: PWM waveforms generated by the Modulator_x4

3.3 Speed and Torque Control block - estimation of $\Delta\Psi_s$

Speed and torque control is achieved trough feeding the SFTC with the right signal. Also important in a pattern based drive system is to move from steady-state to steady-state, this requires a stable electric frequency signal and a stable voltage reference. The speed and torque controller block has three main objects in this model.

I Estimate $\Delta\Psi_s$

II Estimate ω_{ss}

III Estimate U_{ss}

$\Delta\Psi_s$ is obtained by comparing the actual stator flux with an estimated reference stator flux Ψ_s^*. This Ψ_s^* is a complex space vector put together by two separate components in rotor field coordinates, that are representing the rotor flux magnitude and speed references. Hence, the β components represent torque ($\Im$-axis) and the α component represent the rotor flux ($\Re$-axis), see figure 3.21

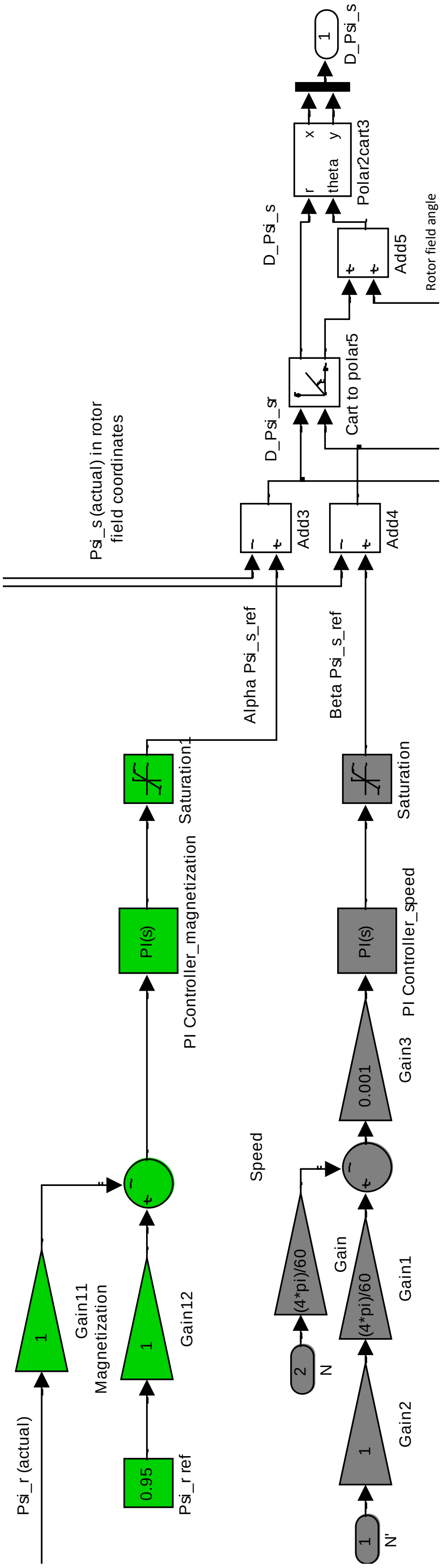


Figure 3.21: Estimation of Ψ_s^* from speed and torque controllers, the block diagram is retrieved from the Simulink model .

The control parameters in the model have not been tuned in a systematic way, this remains to be done. The control parameters in use have been found through trial and error testing.

The rotor flux has been set to a constant value (0.95 pu). If field-weakening operations of the induction machine is to be analysed, then, the constant must be replaced with an variable reference to achieve field weakening. Field weakening implies that the the magnetization has to be decreased in operational speeds higher than the nominal speed, to prevent damaging of the machine [27]. $\Delta\Psi_s$ is converter to stationary coordinates before the signal is given as an output, sent to the SFTC.

The rotor field rotational angle is extracted from the rotor flux-linkage, measured. Also ω_{ss} is given from the rotor field angle by derivation. The sampling of the derivation is set to 1 ms, this slow sampling gave a more stable ω_{ss} signal during simulations.

The Stationary voltage reference is calculated by equation 2.36. This means that the voltage reference U_{ref} is given by the induction machine flux values, and the equation is parameter dependent due to the use of r_s, k_r, σ and l_s.

An undistorted and stable reference voltage U_{ref} is obtainable trough the *self controlled machine* [4], the goal is to have a voltage reference that does not change, except when an external change occurs [4], e.g. from the speed controller. This has shown to be a challenging task. The voltage reference in the model is derived from the equations proposed in [4], but the system needs tuning to operate as it should. A noisy reference voltage has shown trough simulation to have an devastating effect on the whole control system. The system is shown in figure A.2, which is found in appendix A. Some gains have been set to zero to in the diagram to isolate out functions to get the voltage reference more stable. The saturation block on the output is to assure that the voltage reference does not exceed 1.2, the highest modulation index, thus the switching pattern with the highest modulation index.

An stable enough U_{ref}, to not disturb the control system, was obtained. But a focus on stabilizing the U_{ref} is recommended. This will improve the dynamic response and, most important, avoid that the system tries to constantly change switching-pattern in steady-state [4]. To estimate the fundamental component of the stator flux component $\Psi_s 1$ have a filter been used, this filter is located down in the left corner in figure A.2. The adjustable parameter is g1.

Chapter 4

Simulation Results and Data Analysis

This chapter contains simulation results with discussions. The induction motor model used in the simulations has the following data:

Un, line to line	690 Vrms
In,phase	478 Arms
fn	50 Hz
poles	2
rs, stator resistance[pu]	0.018
rR, rotor resistance [pu]	0.018
xH, main reactance [pu]	4
xsigma, total leakage reactance [pu]	0.2

The Three-Level NPC converter model used in simulations is found in the MATLAB simulink SimPowerSystems toolbox. The load is given by equation 4.1 which is a centrifugal load-model[27], typical for for pump and compressor systems. k_L is the load constant and ω is the angular velocity on the motor shaft. Note that the induction motor used in simulations is not a Medium Voltage induction machine, nevertheless will the simulation results represent the functionality of Programmed Modulation. The modulation system is also machine parameter independent, except for the calculation of the voltage reference. This makes the model easily adoptive to drive systems with other Three Level NPC converter models and induction machine models. The dc-bus voltage is constant.

$$T_L = k_L\omega^2 \tag{4.1}$$

4.1 Pattern Exchange Transients

In the following are simulation results from two different cases of pattern exchange presented. The first case illustrates an pattern exchange where an increase in modulation index is included. As discussed in the theory chapter, will current transient oscillations occur when a exchange between pulse patterns with different modulation index occur. The second case illustrate the effect of a pattern exchange where the modulation index is the same in both patterns. Both cases are tested with and without the SFTC. Block diagram over the system is located in appendix A.

4.1.1 Pattern Change With and Without an Active SFTC $m_1 \neq m_2$

In this simulation is the speed and magnetization controller disconnected ($\Delta\Psi_s$). Effects of a pulse pattern exchange without any interference from the dynamic control system is presented. Fist, is the effect of the SFTC illustrated as a step in voltage occur. The electric frequency is given by the angular velocity of the rotor flux linkage vector. Hence, the speed is determined by the steady-state voltage.

Torque (electromagnetic and load) responses is shown in figure 4.1, where the step-up in the modulation index is from 1.1 to 1.2. The induction machine operates in steady-state when the change occur, the pulse number remains the same $N = 5$. As a consequence to this step-up in voltage, high and long torque transient oscillations arise in the case where the SFTC is inactive (gray curve). The green curve is the electromagnetic torque during the pattern change with the SFTC active. The oscillation amplitude is then reduced in the sub-transient period and oscillations are almost completely removed after 10 ms. Very similar to the torque response is the response of the space vector current amplitude i_{s_s} during the same case. See figure 4.2.

If not considering the current oscillations, at t= 0.5 , i_{s_s} increase to an value of 1.5 pu, and slowly decays. This is due to the uncontrolled system which applies an high torque to reach a new steady-state speed corresponding to the new steady-steady voltage. This is ,barley, visible in figure 4.1 where the blue line indicates the rotational speed in RPM.

By comparing the i_{s_s} and the dynamic modulation error d, shown in figure 4.3, can a clear relationship be seen between the oscillations in the current and the dynamic modulation error. When the error has been eliminated, stops the oscillations. This indicates that the stator flux trajectory control is an effective way of handling unwanted transient currents.

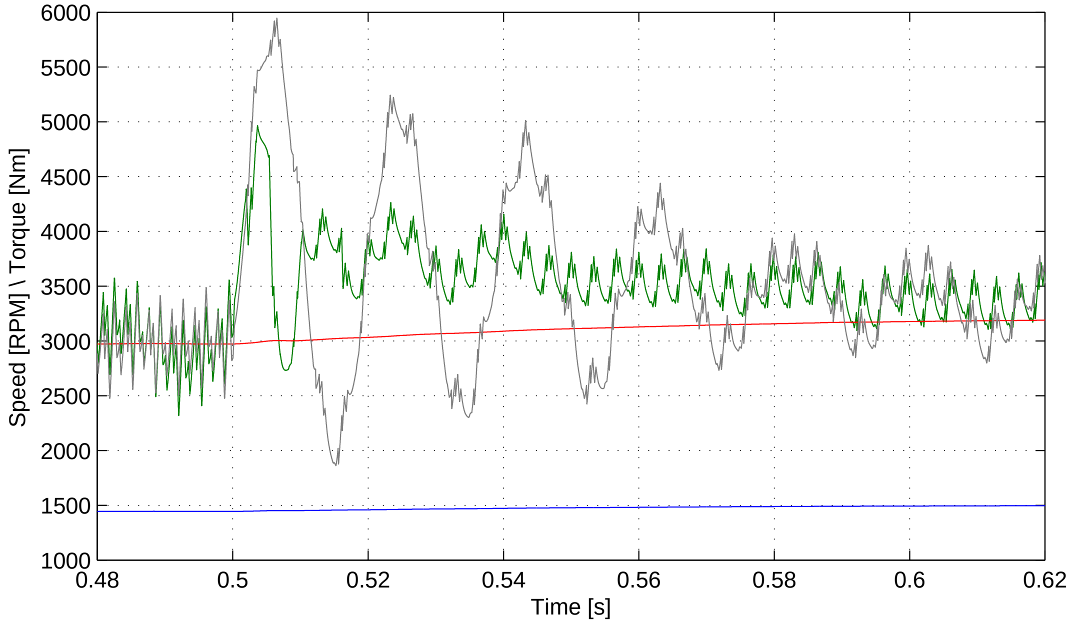


Figure 4.1: Electromagnetic torque during a pattern change from $P(1.1, 5)$ to $P(1.2, 5)$ at $t = 0.5$. Steady-state reached after 0.4 s. Torque response indicated in gray color is with the SFTC inactive.

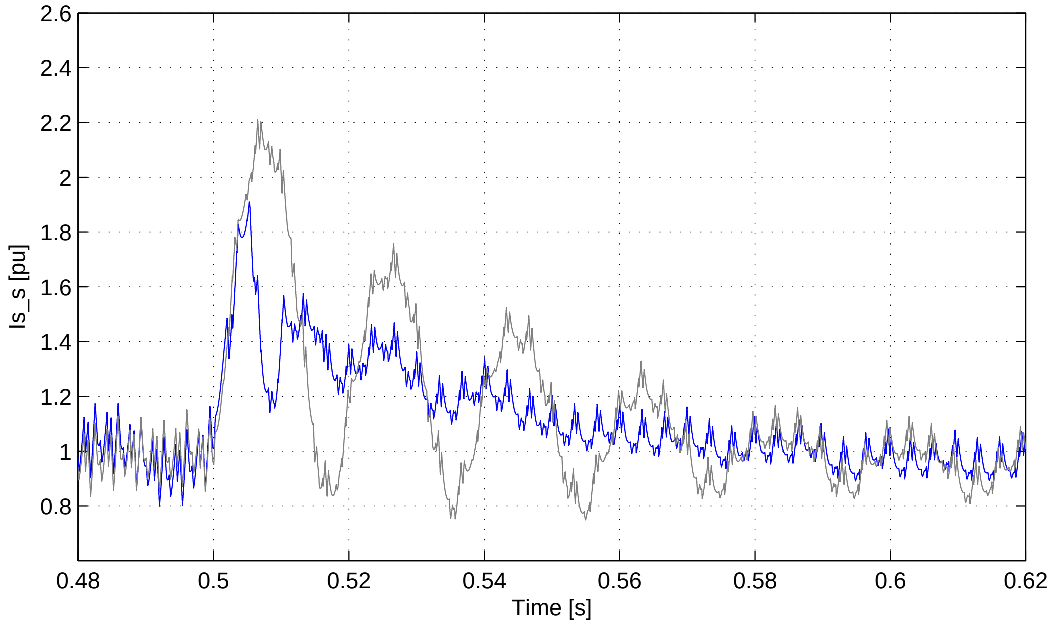


Figure 4.2: Stator current space-vector amplitude (i_{s_s}) during a pattern change from $P(1.1, 5)$ to $P(1.2, 5)$ at $t = 0.5$. Steady-state reached after 0.4 s. i_{s_s} response indicated in gray is from a simulation with the SFTC inactive.

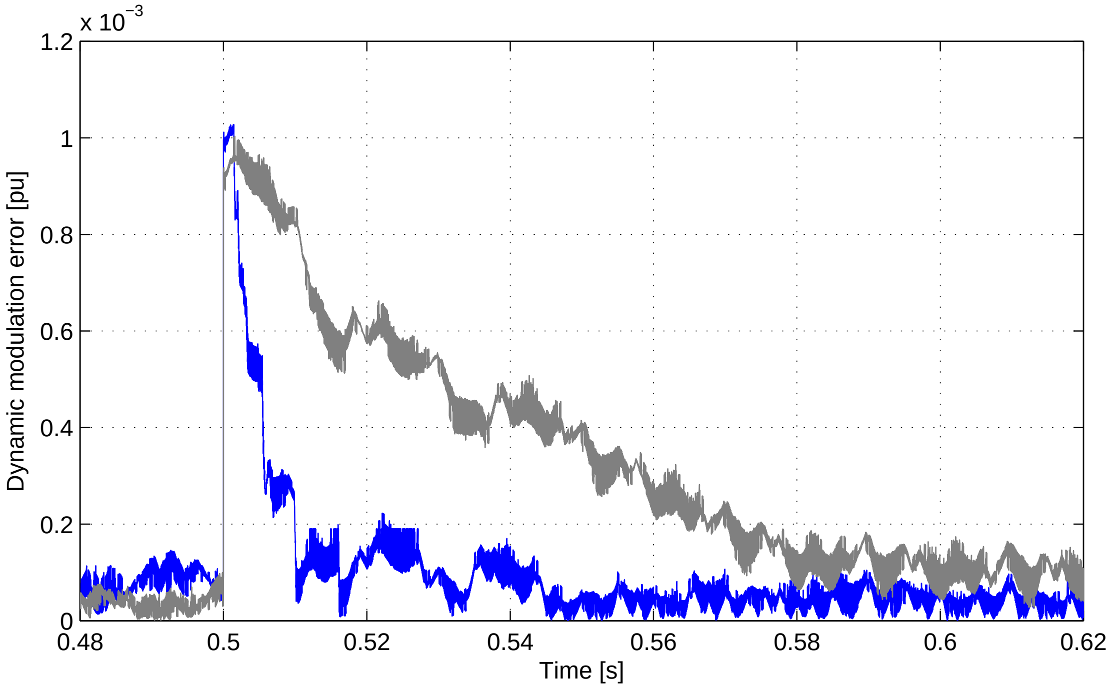


Figure 4.3: Dynamic modulation error (d) during a pattern change from $P(1.1, 5)$ to $P(1.2, 5)$ at $t = 0.5$. Steady-state reached after 0.4 s. d response indicated in gray is from a simulation with the SFTC inactive.

The oscillation effect in the phase currents is illustrated in figure 4.4, where the coloured responses are with the SFTC active and the gray curves are from the simulations with the SFTC inactive. Note that in figure 4.4 are the phase currents more distorted when m equals 1.1 compared to m equal 1.2. This corresponds good with the associated WTHD0 values of the two switching-patterns. See the WTHD response for the $N = 5$ patterns in appendix C. Figure 4.5 illustrates the actual stator flux during the switching-pattern exchange, with (coloured) and without (gray) the SFTC.

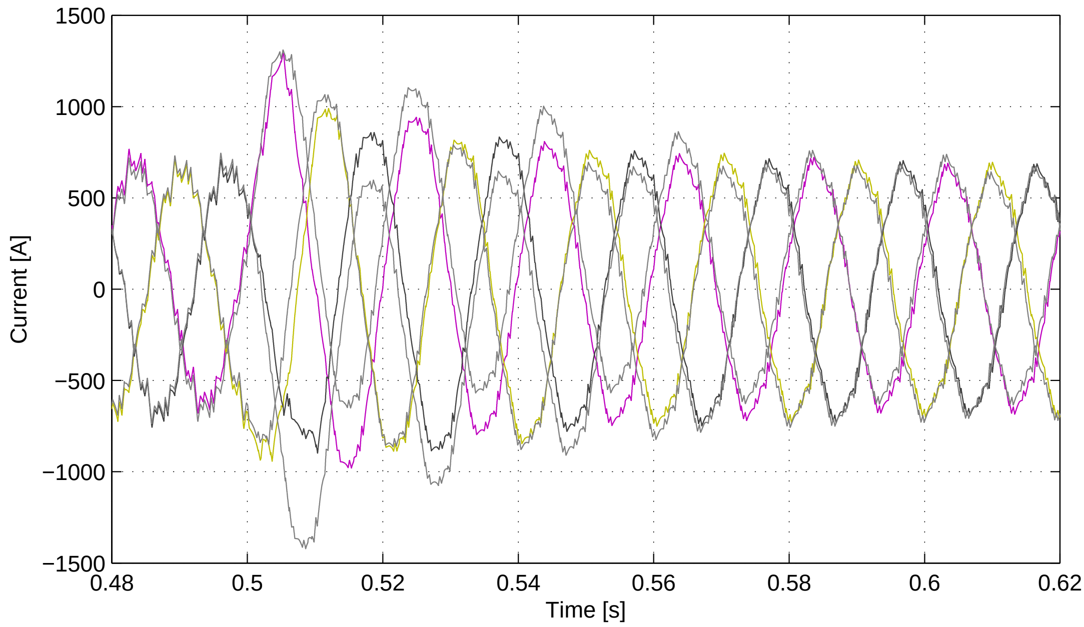


Figure 4.4: Phase currents during a pattern change from $P(1.1, 5)$ to $P(1.2, 5)$ at $t = 0.5$. Steady-state reached after 0.4 s. Phase currents curves indicated in gray are from a simulation with the SFTC inactive.

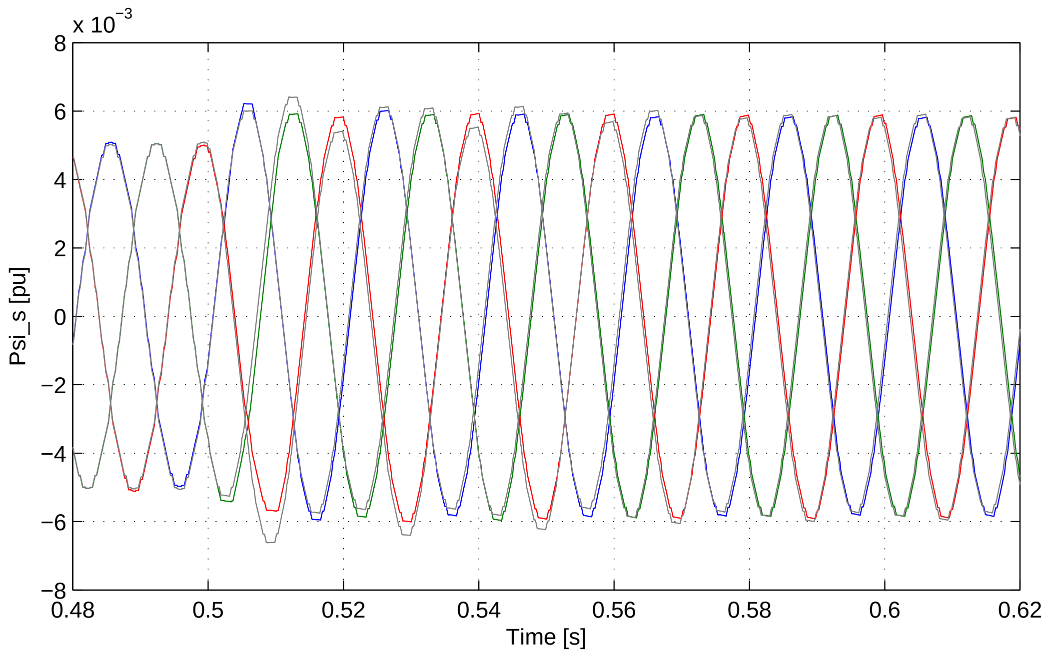


Figure 4.5: Ψ_{a0-b0}, Ψ_{b0-c0} and Ψ_{c0-a0} during a pattern change from $P(1.1, 5)$ to $P(1.2, 5)$ at $t = 0.5$. Steady-state reached after 0.4 s. stator flux responses indicated in gray is from a simulation with the SFTC inactive.

4.1.2 Pattern Change With and Without an Active SFTC $m_1 = m_2$

Even a change between two patterns that has the same modulation index and the same pulse number, but are from two different "optimal" solutions, can result in current transients. The current transients arise due to a mismatch between the two stator ripple trajectories in e.g. $P_1(0.77, 5)$ and $P_2(0.77, 5)$. The angular difference in switching events between the two particular patterns can be seen in appendix C. Where $P_1(0.77, 5)$ is a pattern from the switching-angle trajectory which can be seen from u_{st} equal 0.74 to 0.76. For the simulations presented here, has this trajectory been expanded to an m equal to 0.77. And $P_2(0.77, 5)$ is equal the pattern at m equal 0.77 shown in the appendix.

The switching-angles in the two patterns are representing the same modulation index m, hence the fundamental voltage component is equal. But since they both are representing a different "optimal" solutions of the $P(0.77, 5)$ combinations, will the optimal stator flux trajectory also have different ripple trajectories. Due to the different switching-patterns. This can cause current transients due to the difference in stator flux ripple at the instant when the pattern exchange occur. The difference between the optimal stator flux in P_1 and P_2 has been measured during simulations and is illustrated in figure 4.6. The measurement was between $P_2(0.77, 5)$ and the pattern in use, therefore is the deviation zero after the pattern change from P_1 to P_2.

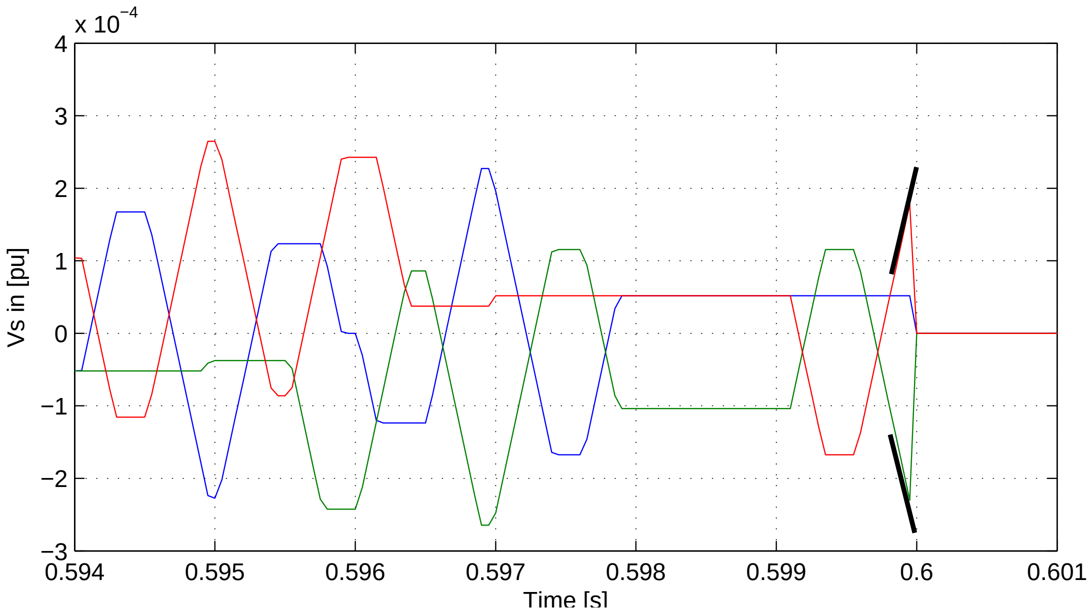


Figure 4.6: Δ flux between P_1 and P_2. Illustrates the deviation between the optimal flux ripples. A pattern change happens at t=0.6, the two gray lines indicates where the ripple is at time when the pattern-exchange takes place

From the deviations between the two optimal stator flux ripples was the switching-pattern exchange decided to occur at t=0.6 s. The black lines inserted into the figure illustrate the difference between the two trajectories at t=0.6. When this was tested, the pattern exchange caused current transient oscillations, the current response is illustrated in figure 4.7, the gray curve illustrate the response when the SFTC is inactive. To be sure of hitting the same ripple deviation in the three simulations a constant frequency of 30 Hz where used as the electrical frequency in this test.

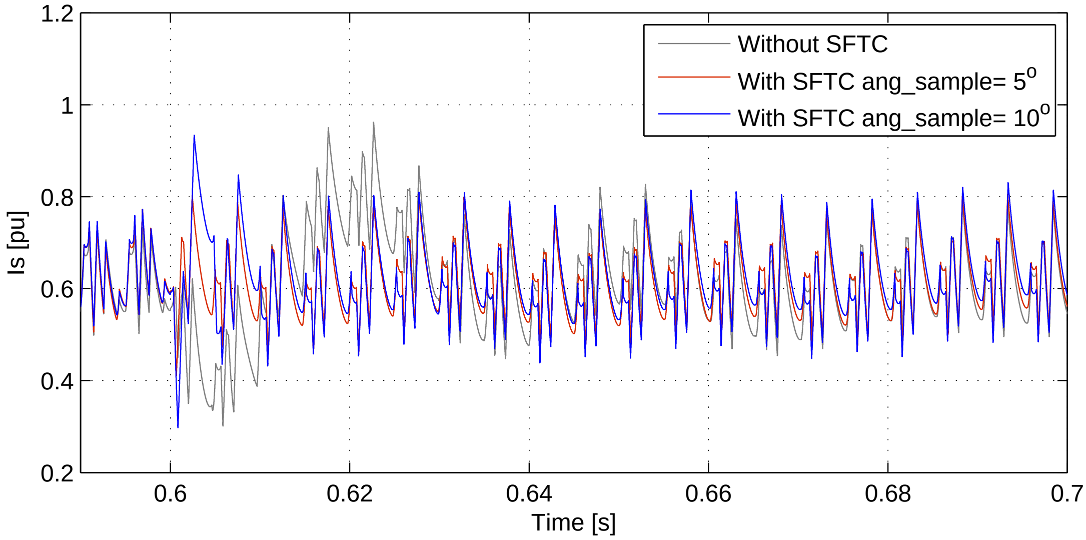


Figure 4.7: Stator current i_{s_s} during a pattern change from $P(0.77, 5)$ to $P(0.77, 5)$ at $t = 0.6$. Steady-state reached after 0.55 s. i_{s_s} response indicated in gray is from a simulation with the SFTC inactive, the red and blue i_{s_s} are from simulations with an active SFTC is used.

Comparing the three different i_{s_s} responses indicated in gray, blue and red in figure 4.7 shows that the SFTC effectively removes the current transient due to the switching-pattern exchange. Especially an ang_sample value of 5 degrees was very efficient, this corresponds to an sampling time in the SFTC of 0.46 ms. Which is close to the suggested sampling time in [2] which is 0.5 ms. The corresponding dynamic modulation errors are shown in figure 4.8, again can a strong correlation between the the dynamic modulation error and the current transient be seen.

The current transients in i_{s_s}reflects in the torque response shown in figure 4.9 and in the phase currents in figure 4.10.

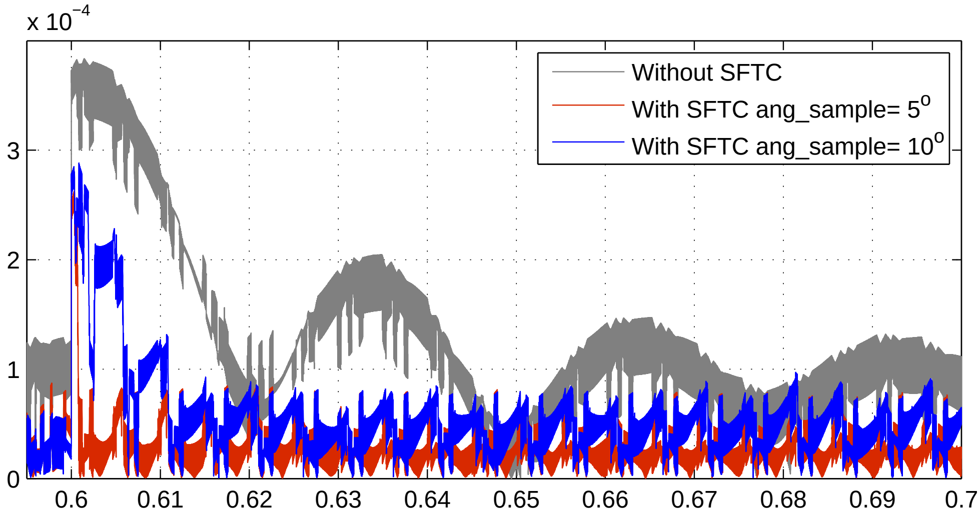


Figure 4.8: Dynamic modulation error (d) during a pattern change from $P(0.77, 5)$ to $P(0.77, 5)$ at $t = 0.6$. Steady-state reached after 0.55 s. d response indicated in gray is from simulation without the SFTC active, the red and blue d responses are from simulations with an active SFTC.

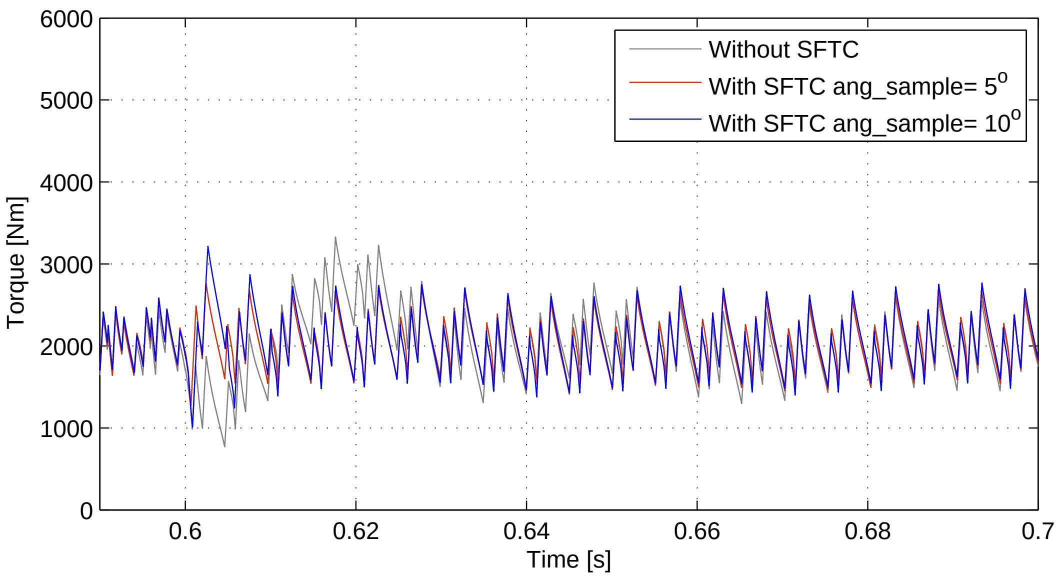


Figure 4.9: Electromagnetic torque(Te) during a pattern change from $P(0.77, 5)$ to $P(0.77, 5)$ at $t = 0.6$. Steady-state reached after 0.55 s. Te response indicated in gray is from simulation without the SFTC active, the red and blue responses are Te from simulations with an active SFTC

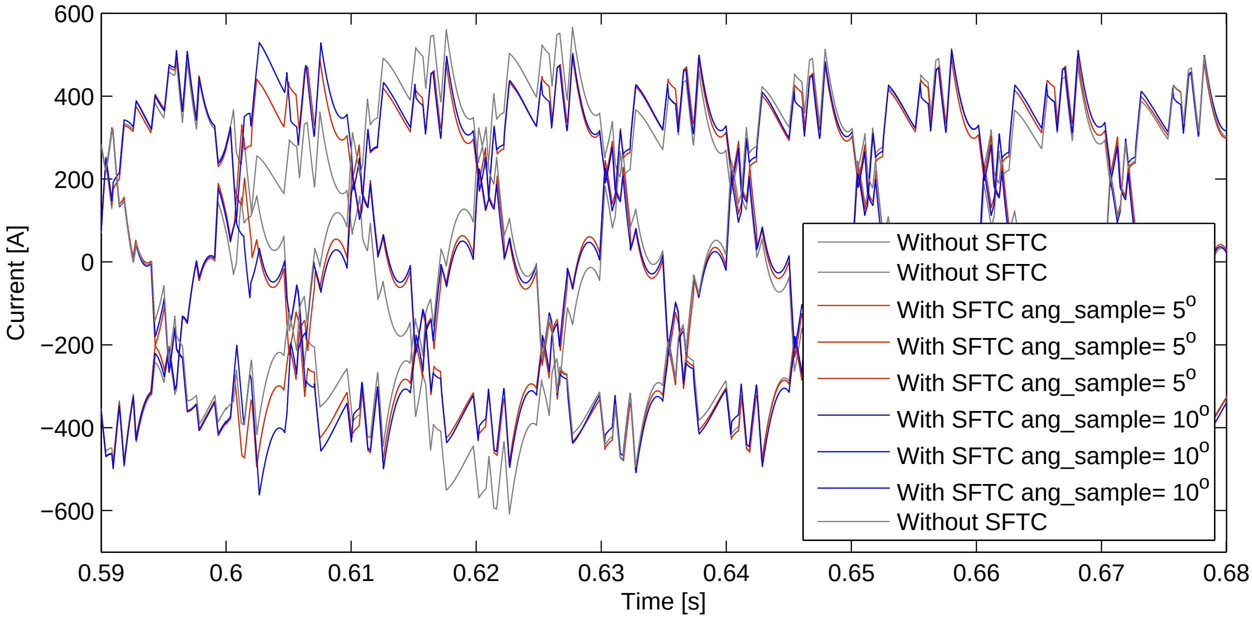


Figure 4.10: Phase currents during a pattern change from $P(0.77, 5)$ to $P(0.77, 5)$ at $t = 0.6$. Steady-state reached after 0.55 s. Phase currents response indicated in gray is from simulation without the SFTC active, the red and blue responses are phase currents from simulations with an active SFTC

4.2 Comparison of Phase Currents from Synchronous Optimal Pulse Width Modulation and Conventional Asynchronous Carrier Based PWM Modulation

A comparison of induction machine phase currents from Programmed Modulation with SOM pattern and a conventional carrier based PWM generator is shown in figure 4.11. The carrier based PWM generator is available in the MATLAB simulink SimPower toolset. The carrier based PWM system is operating in asynchronous modulation. Both PWM strategies are operating same induction motor with the same Three-Level NPC converter. Both systems operates with the same frequency and modulation index. Results presented are from steady-state operation where the operating frequency is 25 Hz an the switching frequency is 300.

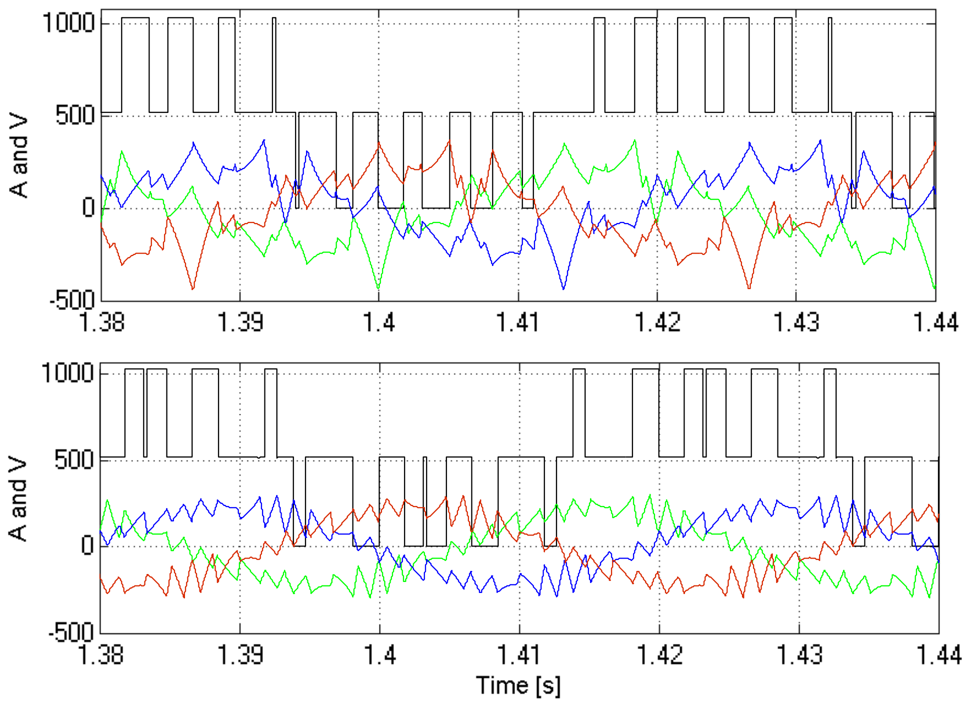


Figure 4.11: Phase current waveforms, the upper figure represent the carrier based PWM system, the lower figure represent the SOM system. Both modulation systems are in steady-state and have the same switching frequency.

By comparison is it clear that optimization of switching-patterns is an efficient way reduce the current ripple. It must be mentioned that it has not been appointed much work to the optimization process of SOM patterns, including the one used in figure 4.11. This means that the Pattern can, most likely, become more optimal, and produce less ripple than the presented example. Also the alternative optimization criteria mentioned in section 2.2.2 should be tested. The voltage waveform, illustrated in black, represent one of the converter bridge-leg voltages in each modulation strategy.

4.3 Dynamic Control with Synchronous Optimal Pulse Width Modulation

A speed reference with ramp, steady-state, step-up and step-down operation was randomly set, seen as the pink signal in figure 4.12. All systems are active in the following simulation. The pulse number N was set to 8 during the whole simulation. This means that the switching frequency was ca. 400 Hz at the point where the actual speed meet the reference ramp, and 560 Hz when the systems encounters the first step-up in speed reference. An pulse number of 16 corresponds to a ration between the fundamental frequency and the switching frequency of 16.

Analysing the start sequence of the electromagnetic torque. Control of the induction machine torque is achieved at approximately t=0.11 s, but does not stabilize until t=0.16 s. Seen as the "plateau" where the torque is 6 kNm in figure 4.12. The amplitude of the torque applied during step-changes in speed reference is limited by the maximum and minimum values of saturation block in the torque controller (β -component of the stator flux reference signal).

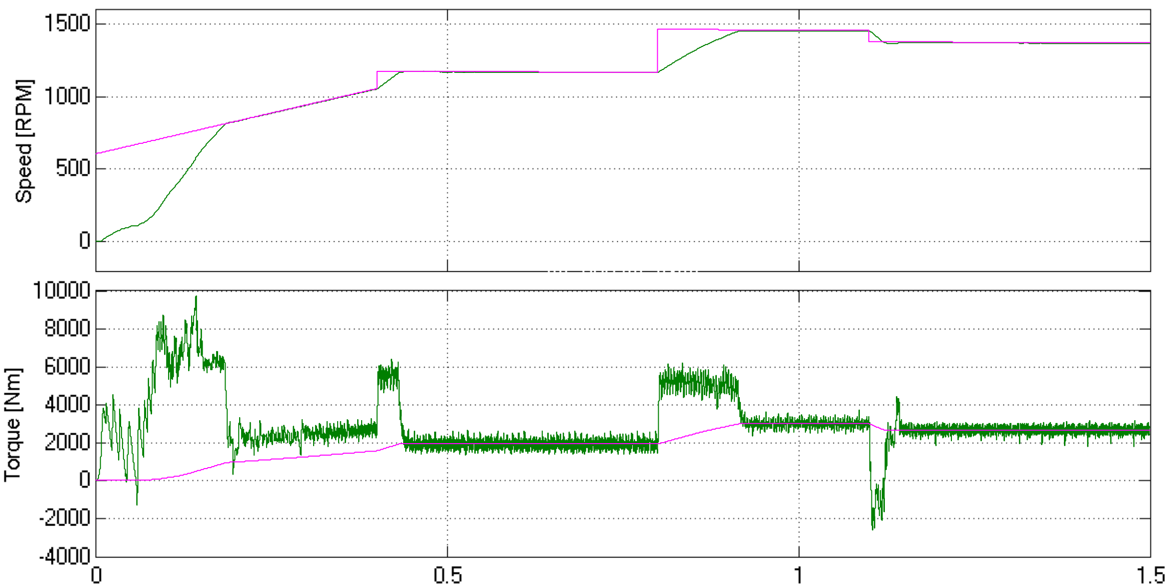


Figure 4.12: Torque and speed

The torque response initiated by a step change in speed reference is very fast, a closer look on the torque response is shown in figure 4.13. The step-up in speed reference occurs at t=0.401 s which means that the control system uses approximately 2 ms to execute the commanded change in the applied torque, according to the figure.

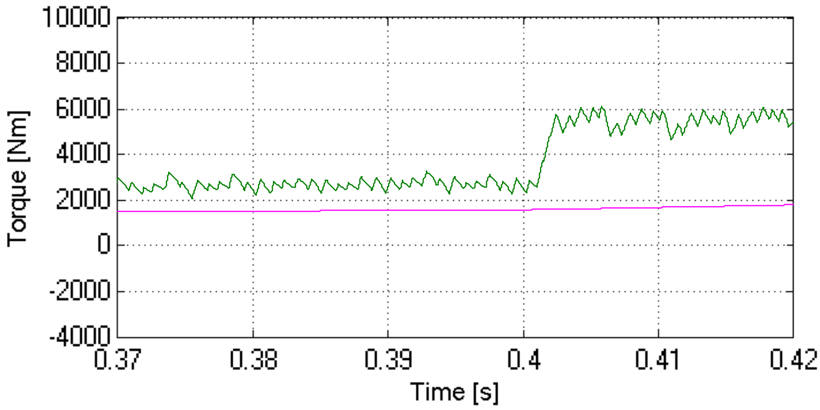


Figure 4.13: Closer look on the electromagnetic torque, indicated as the green curve, during the first step-up in speed reference.

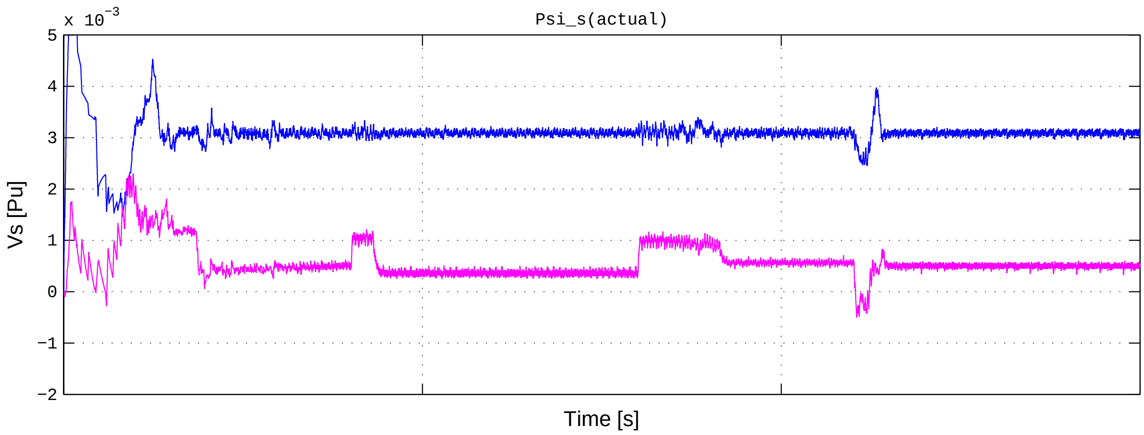


Figure 4.14: α and β component (blue and pink) of the measured stator flux Ψ_s, rotor field oriented.

Figure 4.15 illustrates Ψ_s in cartesian coordinates, rotor field oriented, which is the control parameter in this system. α- component represent the magnetization and β-component represent torque. The two positive step-up changes in reference speed results in maximum applied torque(t=0.41 and t=0.79). The α-component, representing the magnetization in the induction machine, is almost unaffected by the change in torque during this period. This same can be seen in the stator current in α- and β-components, rotor field oriented illustrated in figure 4.17. This implies that no coupling between the components exist, and control of the system is maintained.

On the contrary to the two step-ups in speed reference is the step-down in speed reference that occurs at ca. t=1.1 s not as stable. It reacts fast, but do not complete the dynamic operation as smoothly as the two previously steps. The problem with the unstable torque can be tracked back to the α-component of the actual stator flux, also seen in the α-component in the stator current which represent the magnetizing current in an induction machine [3]. They indicate coupling between the control parameters, and some control is lost. A drop in the rotor flux is also seen in figure 4.18. The source of this "loss of control" is most likely due to a drop in the reference voltage u_{ref}, e.g. the modulation index, illustrated in figure 4.19.

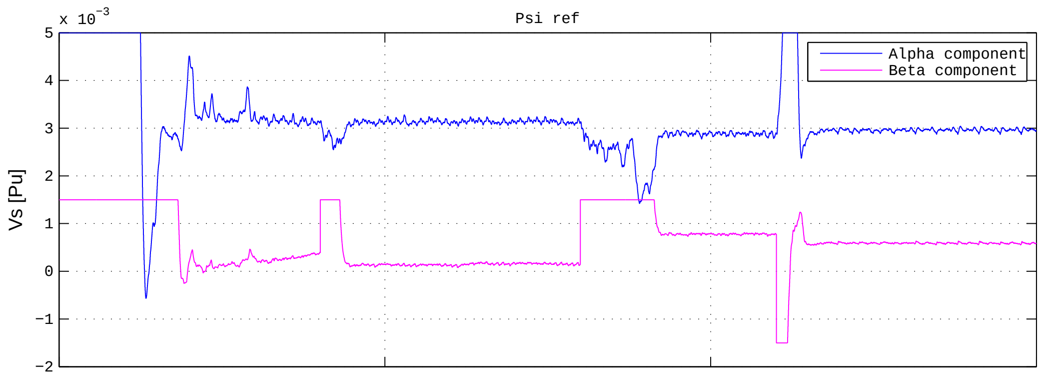


Figure 4.15: α- and β-component (blue and pink) of the estimated stator flux Ψ_s^*, rotor field oriented.

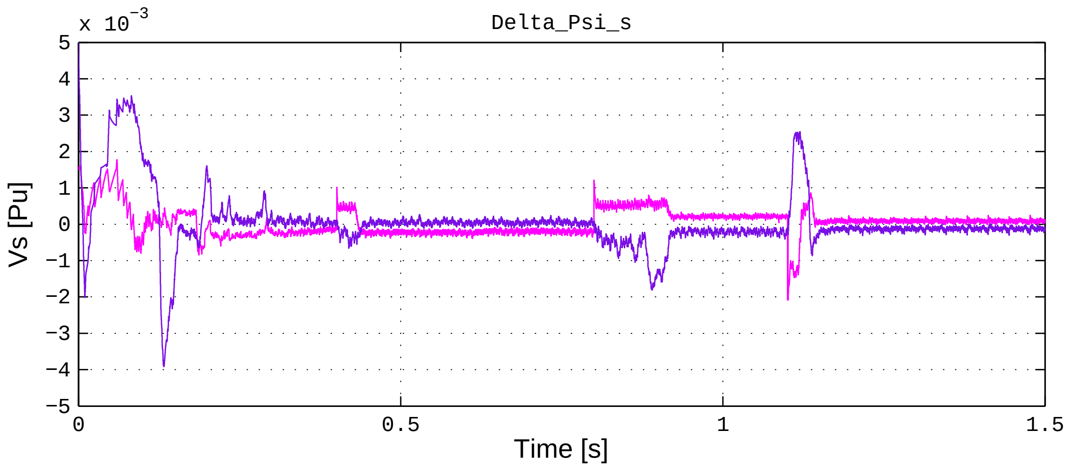


Figure 4.16: α- and β-component (blue and pink, respectively) of the estimated stator flux change $\Delta\Psi_s$, rotor field oriented.

Figure 4.15 illustrate the α- and β-component of the reference Ψ_s^*, which is separately estimated to generate an signal to control the actual stator flux Ψ_s. The resulting change in stator flux ($\Delta\Psi_s$) is shown in figure 4.16. Ideally should this signal be zero in stationary operations, the simulation result shows that it is close to zero but still needs tuning. Tuning the integral constants in the speed and rotor flux PI controllers should be done.

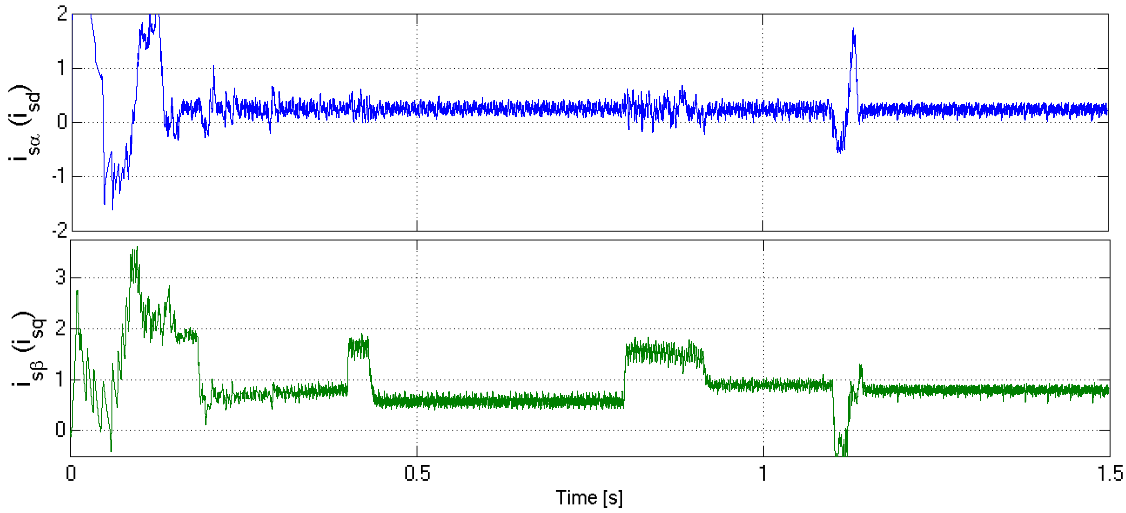


Figure 4.17: α- and β-component (blue and green) of the stator current i_s, rotor field oriented. Also commonly known as the d- and q-components.

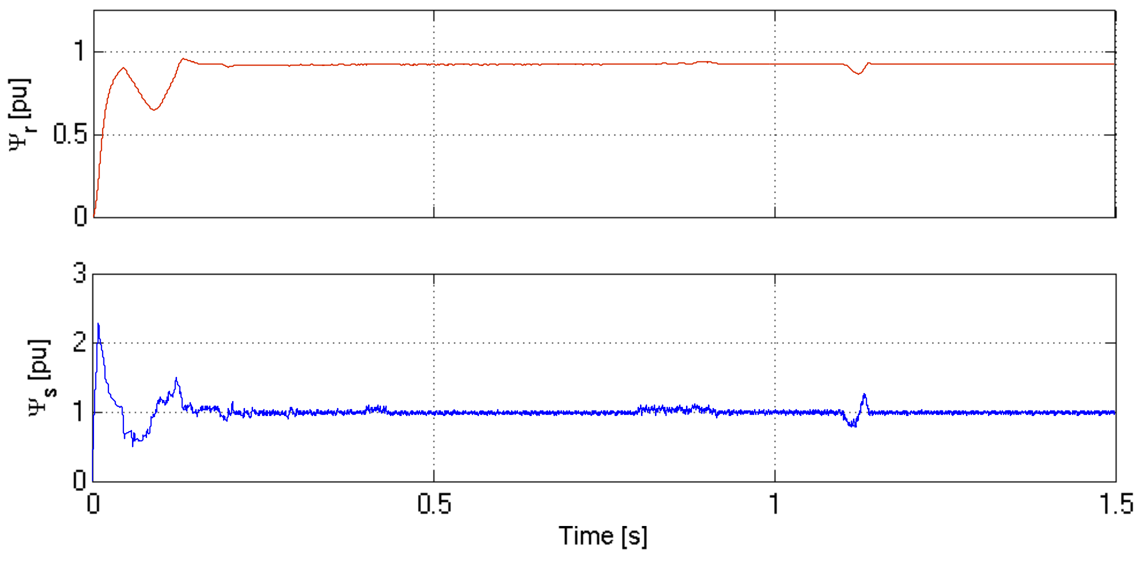


Figure 4.18: Ψ_r indicated in red, Ψ_s indicated in blue. Both measured and presented by the amplitude of the polar coordinates.

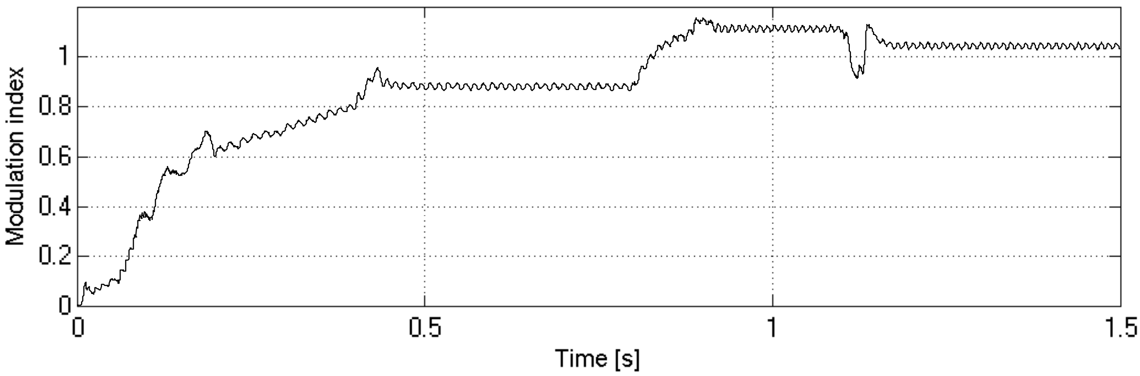


Figure 4.19: modulation index m

The small ripple that follows the modulation index signal trough the whole simulation should not exist. The signal will cause unnecessary exchanges of optimal switching-patterns during operations that should be steady-state e.g. using the same optimal pulse pattern.

Chapter 5

Conclusion and Further Work

5.1 Conclusion

In medium voltage drives, with the present motor and IGBT technology, should the perfect modulation strategy have qualities like fast dynamic control and allow a very low switching frequency without generating current harmonics.

The fundamental characteristics of Programmed Modulation allow the switching instants to be freely distributed over a fundamental period to generate the converter bridge-leg voltages. This feature is exploited in Synchronous Optimal Modulation pre-calculated patterns to achieve reduction of phase current harmonics. Optimization of switching instants are calculated with the objective of minimizing the harmonic components that contributes to losses in the induction machine. For this purpose, the Weighted Total Harmonic Distortion (WTHD0) can be used.

The Concept of Programmed Modulation has been tested in the presented MATLAB simulation model. Results show that fast dynamic control is achievable with programmed modulation trough manipulation of the switching-patterns, during operation, to control the torque. The modifications of the switching-patterns are calculated by the proposed Stator Flux Trajectory Controller (SFTC), this controller is also very efficient at eliminating unwanted current transients that can occur as a result of a pattern exchange. A Challenge with Programmed Modulation is to obtain a stable reference voltage. This is important, an unstable voltage reference has shown to disturb the control system which controls the α- and β-component of the stator flux, in rotor field coordinates. The presented MATLAB simulink model has been explained and further development is suggested below in the further work section.

Changing to Synchronous Optimal Modulation switching-patterns permits a reduction in switching frequency. Considering the fast dynamic control achievable in Programmed Modulation with the stator flux trajectory control technique is this modulation strategy well suited for medium voltage drives. Further research should definitely be carried out.

5.2 Further Work

The modulation looks promising and the most exciting research remains. Further development of the Programmed Modulation model is suggested to be:

I Investigate and implementation of the dc-bus balancing control system suggested in reference [9], this should not result in a higher switching frequency according to the paper.

II Add dead-time effects to improve the validity of simulation results.

III Implementation of a higher resolution of the PWM patterns, with respect to modulation index, into the model. Also the alternative optimization method-/criteria should be tested in the presented model.

IV The speed and torque control system that generate the $\Delta\Psi_s$ need both development and control parameter adjustments, also field weakening and power limitations should be included. The systems reference voltage estimation u^* is most likely the source of the interference in the control system. This reference voltage should be a signal without disturbance from the converter switchings so it works according to "*The Self controlled machine*" principle. E.g it is constant unless an external influence change the steady-state operation point.

V Include the induction machine estimation model to estimate the actual rotor and stator flux in the controlled induction machine.

VI A detailed comparison study should be done to benchmark the SOM drive system up against the Direct Torque- and Space Vector Modulation Strategies.

VII Laboratory set-up and testing of the Programmed Modulated Drive system to reveal new challenges and contrive new solutions.

Bibliography

[1] D. Grahame Holmes and Thomas A. Lipo. Pulse width modulation for power converters, 2003.

[2] J. Holtz and N. Oikonomou. Synchronous optimal pulsewidth modulation and stator flux trajectory control for medium-voltage drives. *Industry Applications, IEEE Transactions on*, 2007.

[3] Ned Mohan. *Advanced Electric Drives - Analysis, Control and Modeling using Simulink*. MNPERE, 2001.

[4] J. Holtz and N. Oikonomou. Fast dynamic control of medium voltage drives operating at very low switching frequency - an overview. *Industrial Electronics, IEEE Transactions on*, 2008.

[5] Ned Mohan, Tore M. Undeland, and William P. Robbins. *The Power Electronics: Converter, Applications and Design*. Ed. Wiley, 2003.

[6] H. Abu-Rub, J. Holtz, J. Rodriguez, and Ge Baoming. Medium-voltage multilevel converters - state of the art, challenges, and requirements in industrial applications. *Industrial Electronics, IEEE Transactions on*, 2010.

[7] Joachim Holtz and Bernd Beyer. The trajectory tracking approach-a new method for minimum distortion pwm in dynamic high-power drives. *Industrial Electronics, IEEE Transactions on*, 1994.

[8] H. du Toit Mouton. Natural balancing of three-level neutral-point-clamped pwm inverters. *Industrial Electronics, IEEE Transactions on*, 2002.

[9] J. Holtz and N. Oikonomou. Neutral point potential balancing algorithm at low modulation index for three-level inverter medium-voltage drives. *Industry Applications, IEEE Transactions on*, 2007.

[10] Yen-Shin Lai, Chun-Ming Li, and Chang-Huan Liu. Harmonic elimination pulse-width modulation techniques of real time inverter control for induction motor drives. In *Industrial Electronics Society, 1999. IECON '99 Proceedings. The 25th Annual Conference of the IEEE*, 1999.

[11] ErwinKreyszig. *Advanced Engineering Mathematics - 9'th Edition*. WILEY INTERNATIONAL EDITION, 2006.

[12] P.N. Enjeti and R. Jakkli. Optimal power control strategies for neutral point clamped (npc) inverter topology. *Industry Applications, IEEE Transactions on*, 1992.

[13] Roy Nilsen. Modulation methods for three level inverter. *Classification: confidential report*, 2010.

[14] Abdul Rahiman Beig, G. Narayanan, and V. T. Ranganathan. Modified svpwm algorithm for three level vsi with synchronized and symmetrical waveforms. *Industrial Electronics, IEEE Transactions on*, 2007.

[15] John M. D. Murphy and Michael G. Egan. A comparison of pwm strategies for inverter-fed induction motors. *Industry Applications, IEEE Transactions on*, 1983.

[16] Franz C. Zach and Hans Ertl. Efficiency optimal control for ac drives with pwm inverters. *Industry Applications, IEEE Transactions on*, 1985.

[17] Giuseppe S. Buja. Optimum output waveforms in pwm inverters. *Industry Applications, IEEE Transactions on*, 1980.

[18] J. Holtz. Pulsewidth modulation for electronic power conversion. *Proceedings of the IEEE*, 1994.

[19] Cungang Hu, Qunjing Wang, Weidong Jiang, Quan Chen, and Qiushi Xia. Optimization method for generating shepwm switching patterns using chaotic ant colony algorithm applied to three-level npc inverter. In *Electrical Machines and Systems, 2007. ICEMS. International Conference on*, 2007.

[20] J. Sun and H. Grotstollen. Solving nonlinear equations for selective harmonic eliminated pwm using predicted initial values. In *Industrial Electronics, Control, Instrumentation, and Automation, 1992. Power Electronics and Motion Control., Proceedings of the 1992 International Conference on*, 1992.

[21] N. Oikonomou and J. Holtz. Stator flux trajectory tracking control for high-performance drives. In *Industry Applications Conference, 2006. 41st IAS Annual Meeting. Conference Record of the 2006 IEEE*, 2006.

[22] J. Holtz and N. Oikonomou. Estimation of the fundamental current in low switching frequency high-dynamic medium voltage drives. In *Industry Applications Conference, 2007. 42nd IAS Annual Meeting. Conference Record of the 2007 IEEE*, 2007.

[23] A.K. Rathore, J. Holtz, and T. Boller. Optimal pulsewidth modulation of multilevel inverters for low switching frequency control of medium voltage high power industrial ac drives. In *Energy Conversion Congress and Exposition (ECCE), 2010 IEEE*, 2010.

[24] T. Bruckner and D.G. Holmes. Optimal pulse-width modulation for three-level inverters. *Power Electronics, IEEE Transactions on*, 2005.

[25] The MathWorks Inc. Documentation simulink, accessed may 11th, 2012. http://www.mathworks.se/help/toolbox/simulink/sfg/f7-67622.html#f7-68222.

[26] The MathWorks Inc. Documentation simulink, accessed may 11th, 2012. http://www.mathworks.se/help/toolbox/simulink/slref/interpretedmatlabfunction.html.

[27] Ned Mohan. *Electric Drives - An integrative approach.* MNPERE, 2003.

Appendix A

Model Block Diagrams

Most of the simulation system is displayed in the report, however, some need a whole page to be readable. Figure A.1 illustrate the modulation system when the electrical frequency is indirectly controlled by the voltage. The SFTC has been disconnected for the purpose of the simulation to demonstrate a uncontrolled system. Figure A.2 illustrate the block diagram of the *self controlled machine* system including the necessary filter to extract the fundamental component of the stator flux signal.

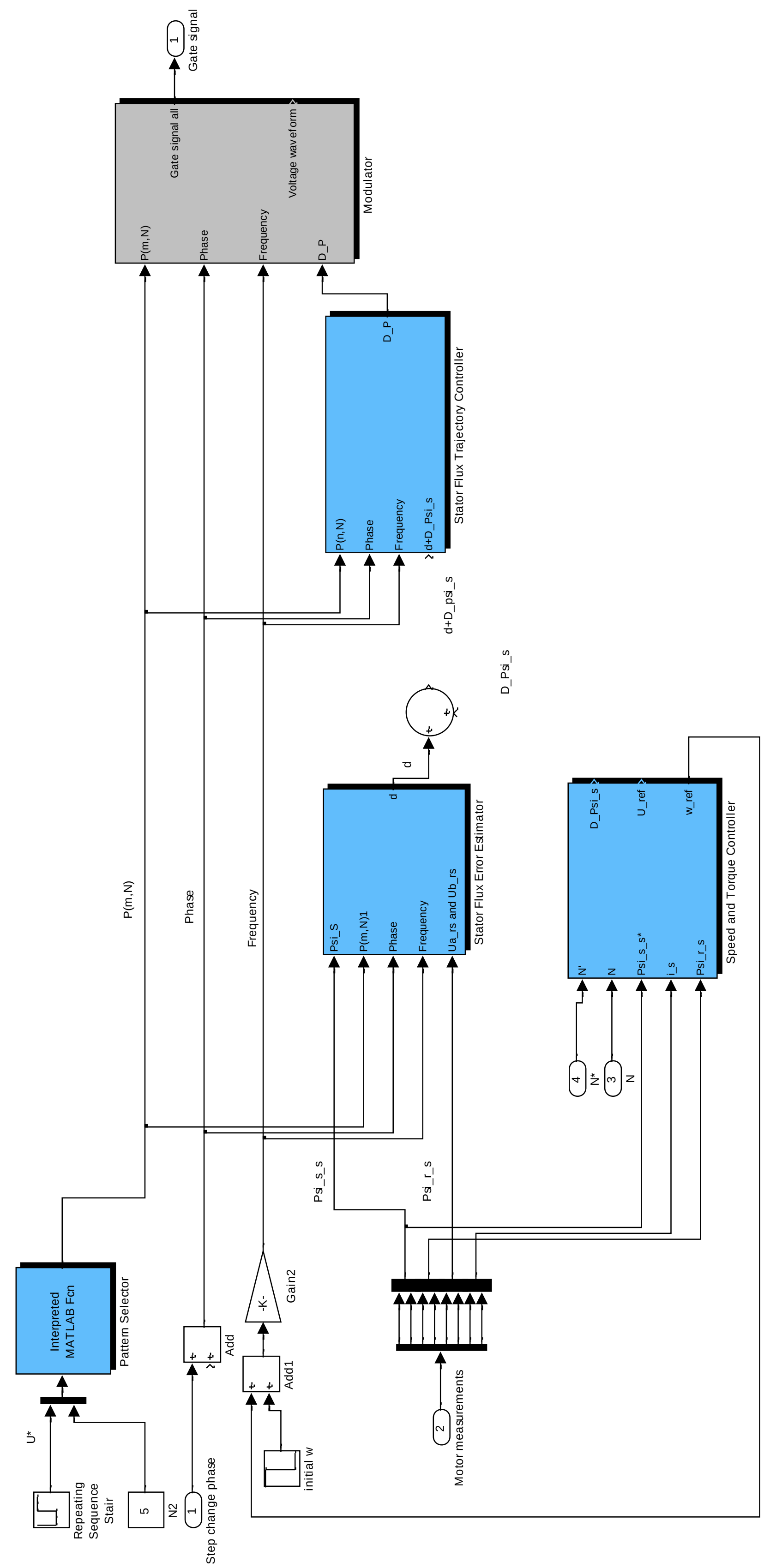


Figure A.1: Switching-pattern based Programmed Modulation system

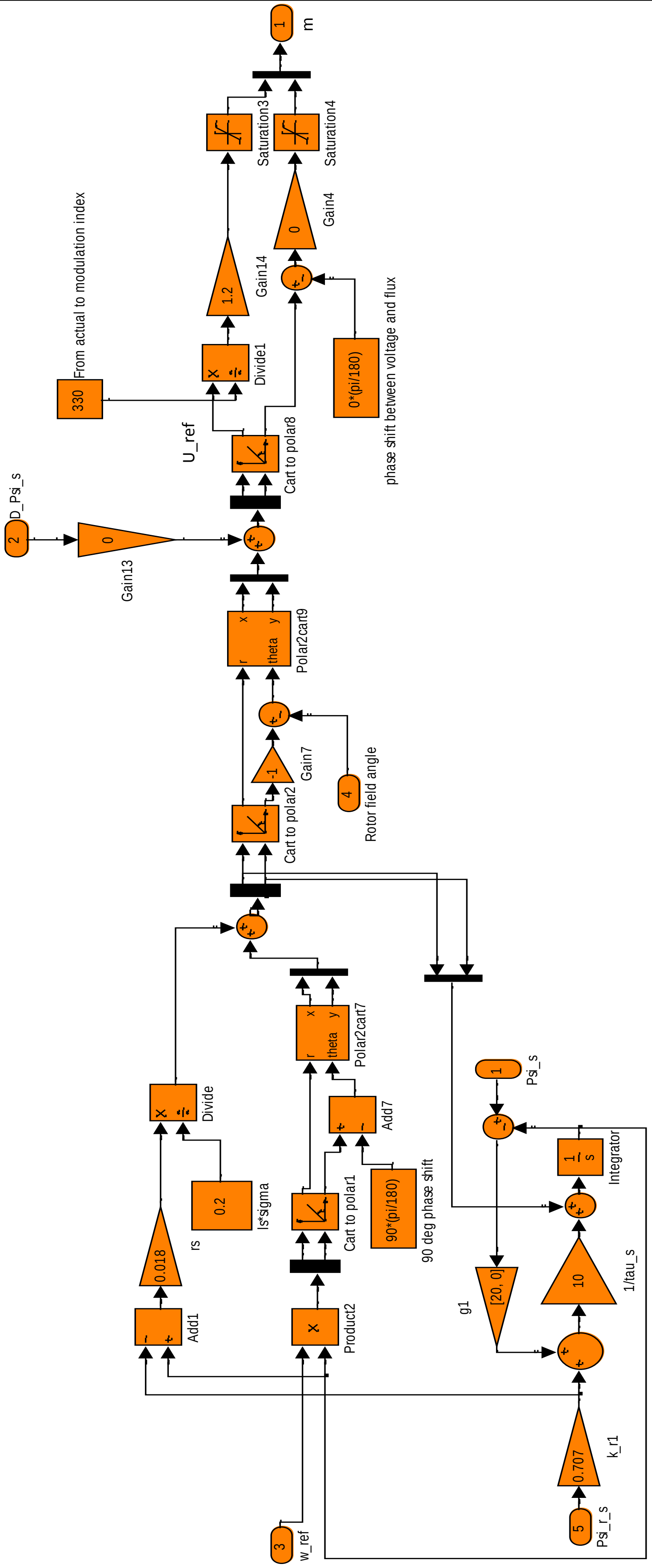


Figure A.2: The self controlled machine subsystem illustrated in a block diagram.

Appendix B

Relationship Between Converter Bridge Leg Voltages and Induction Machine Phase Voltages

Active converters can only control the line-to-line voltage, and have the following voltage relations [13].

$$\begin{aligned} U_{ab}(t) &= U_{a0}(t) - U_{b0}(t) \\ U_{bc}(t) &= U_{b0}(t) - U_{c0}(t) \\ U_{ca}(t) &= U_{c0}(t) - U_{a0}(t) \end{aligned} \tag{B.1}$$

U_{a0}, U_{b0} and U_{c0} are the converter bridge leg voltages. The phase voltages of the induction machine can now be expressed as [13]

$$\begin{aligned} U_{a0}(t) - U_{b0}(t) &= U_a(t) - U_b(t) \\ U_{b0}(t) - U_{c0}(t) &= U_b(t) - U_c(t) \\ U_{c0}(t) - U_{a0}(t) &= U_c(t) - U_a(t) \end{aligned} \tag{B.2}$$

And the zero system voltage $U_0(t)$ can be expressed by phase voltages as [13]

$$U_0(t) = \frac{U_a(t) + U_b(t) + U_c(t)}{3} \tag{B.3}$$

From the above equation is the motor phase voltage given as shown in equation B.4.

$$
\begin{aligned}
U_a(t) &= \frac{1}{3}\big(2U_{a0}(t) - U_{b0}(t) - U_{c0}(t)\big) + U_0 \\
U_b(t) &= \frac{1}{3}\big(2U_{b0}(t) - U_{c0}(t) - U_{a0}(t)\big) + U_0 \\
U_c(t) &= \frac{1}{3}\big(2U_{c0}(t) - U_{a0}(t) - U_{b0}(t)\big) + U_0
\end{aligned} \tag{B.4}
$$

and the stator flux phase components are

$$
\begin{aligned}
\Psi_a(t) &= \frac{1}{3}\big(2\Psi_{a0}(t) - \Psi_{b0}(t) - \Psi_{c0}(t)\big) + \Psi_0 \\
\Psi_b(t) &= \frac{1}{3}\big(2\Psi_{b0}(t) - \Psi_{c0}(t) - \Psi_{a0}(t)\big) + \Psi_0 \\
\Psi_c(t) &= \frac{1}{3}\big(2\Psi_{c0}(t) - \Psi_{a0}(t) - \Psi_{b0}(t)\big) + \Psi_0
\end{aligned} \tag{B.5}
$$

where Ψ_{a0}, Ψ_{b0} and Ψ_{c0}, is the virtual flux given by the converter bridge-leg voltages integral.

By the use of equation B.2 an expression to transform induction machine stator phase flux values to converter bridge leg line-to-line stator flux values given as.

$$
\begin{aligned}
\Psi_{a0b0}(t) &= \Psi_a(t) - \Psi_b(t) \\
\Psi_{b0c0}(t) &= \Psi_b(t) - \Psi_c(t) \\
\Psi_{c0a0}(t) &= \Psi_c(t) - \Psi_a(t)
\end{aligned} \tag{B.6}
$$

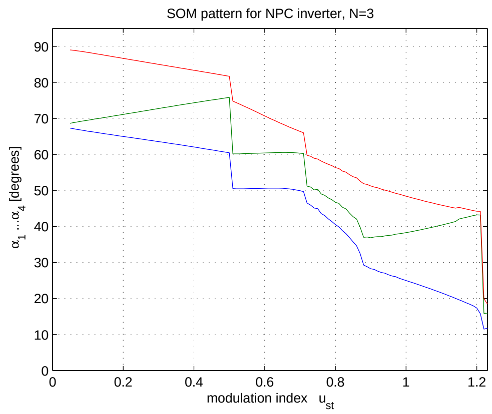
These are needed to correctly compare the optimal stator flux trajectory with the actual stator flux trajectory.

Appendix C

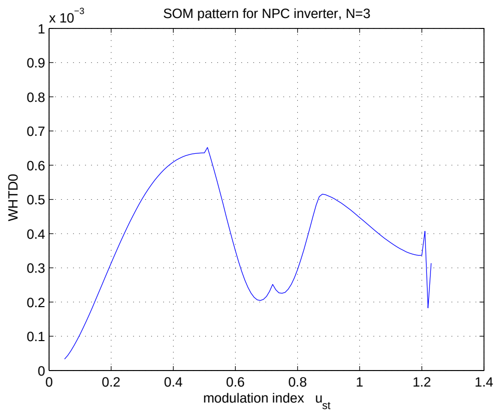
Optimal Pulse Width Patterns

In the following figures are the modulation index m indicated as u_{st}.

C.1 N=3



(a) Angular Switching instants with respect to modulation index



(b) WTHD0

Figure C.1: SOM angular switching pattern with corresponding WTHD0, N=3

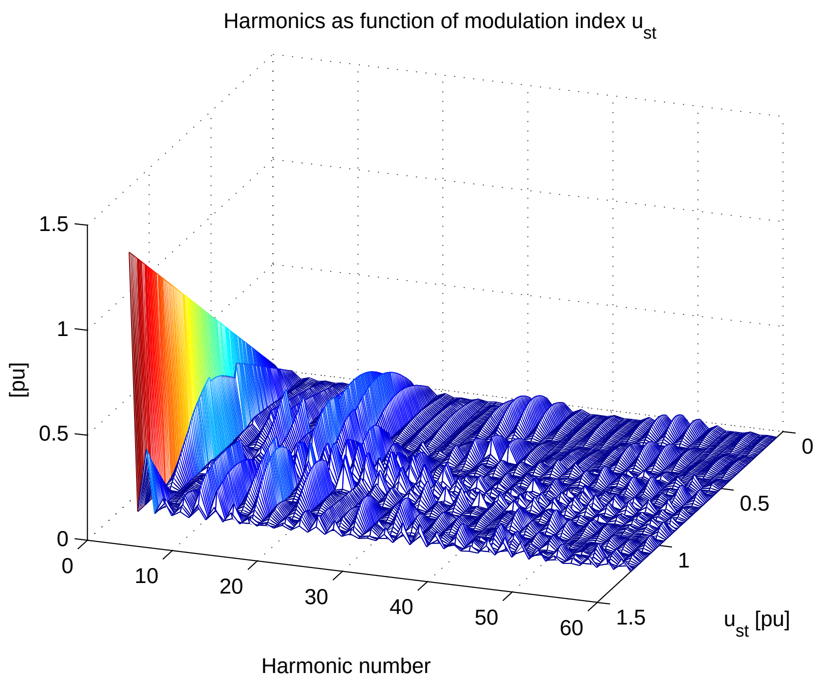
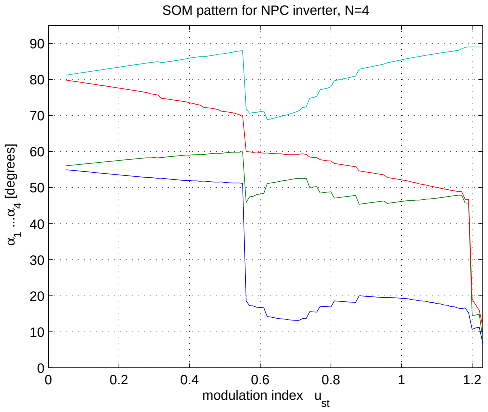
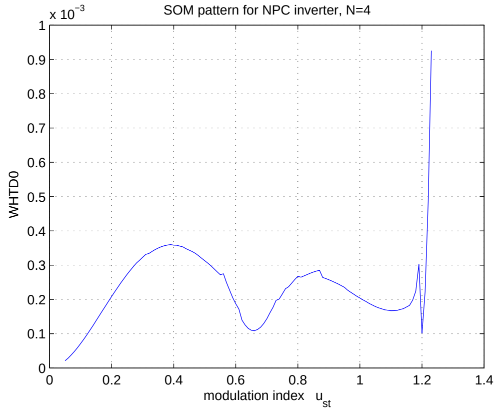


Figure C.2: Harmonic amplitudes, N=3

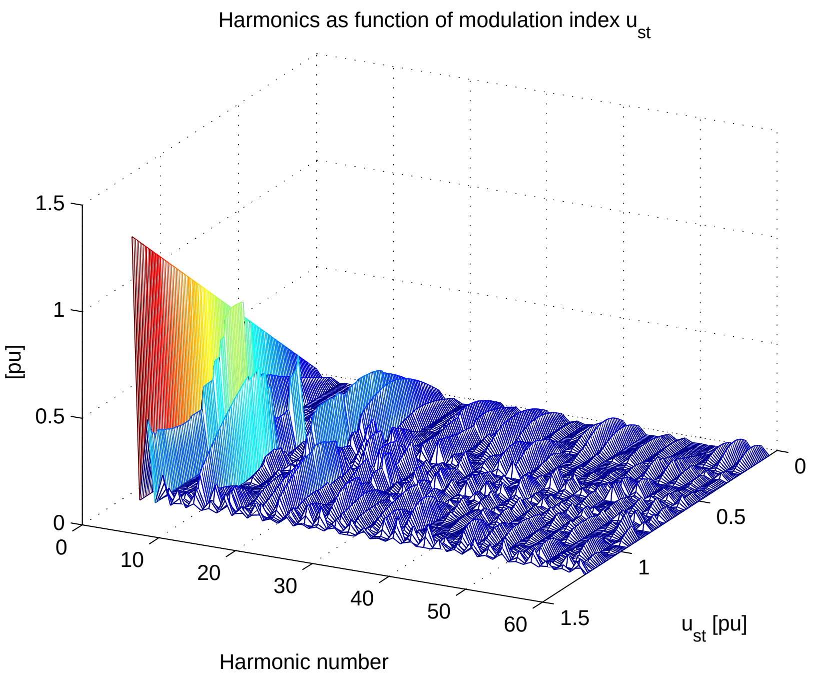
C.2 N=4



(a) Angular Switching instants with respect to modulation index



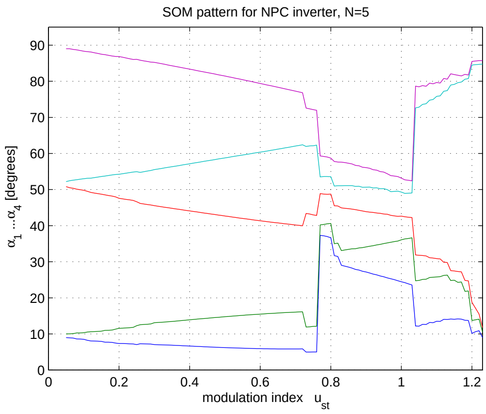
(b) WTHD0



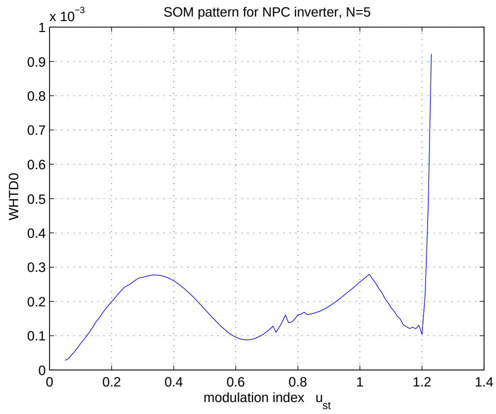
(c) harmonics

Figure C.3: SOM angular switching-pattern with corresponding WTHD0 and harmonic amplitudes, N=4

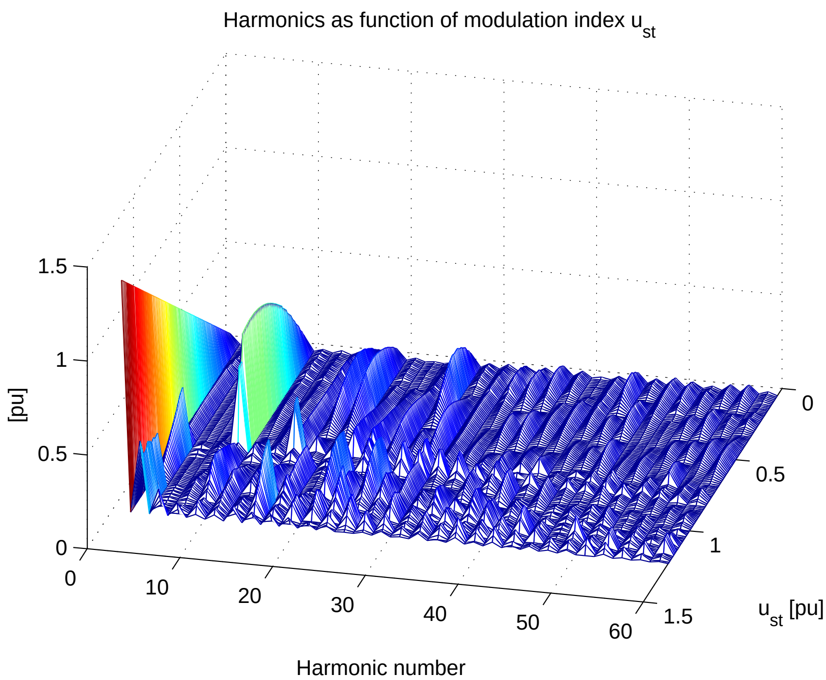
C.3 N=5



(a) Angular Switching instants with respect to modulation index



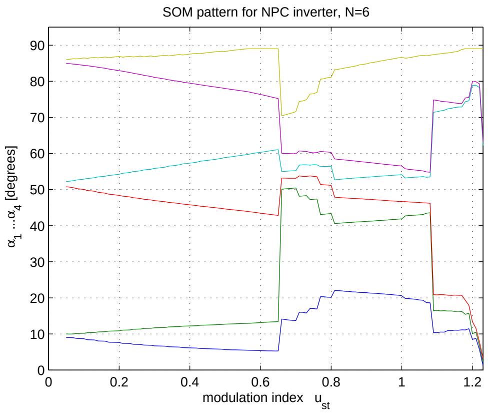
(b) WTHD0



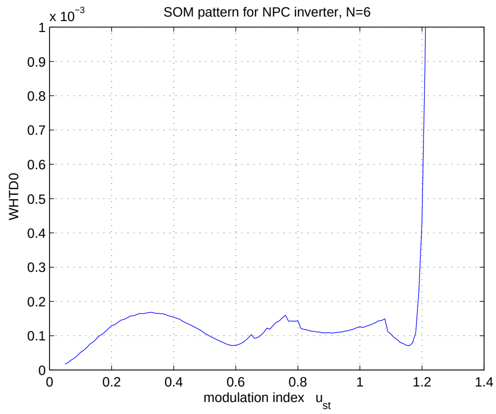
(c) harmonics

Figure C.4: SOM angular switching-pattern with corresponding WTHD0 and harmonic amplitudes, N=5

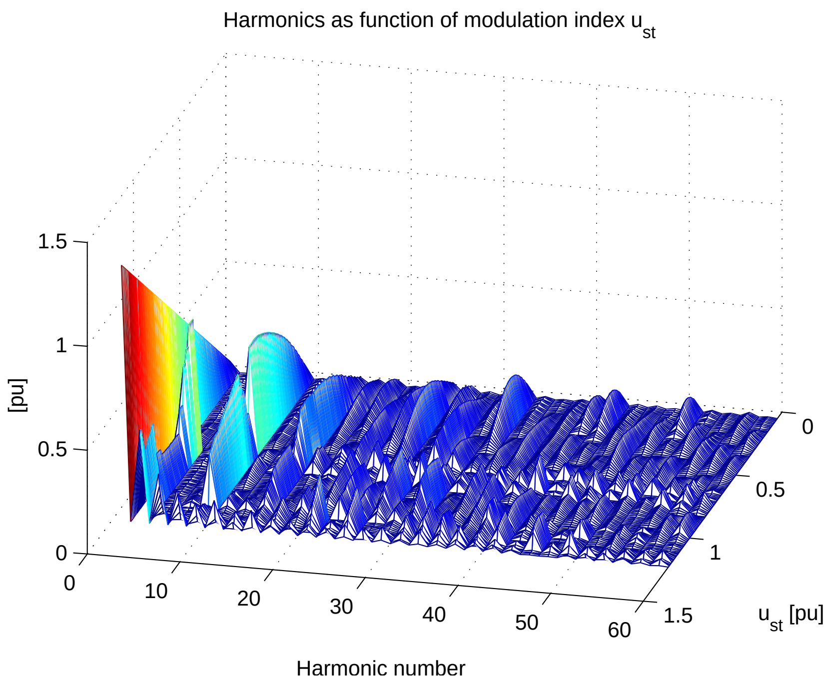
C.4 N=6



(a) Angular Switching instants with respect to modulation index



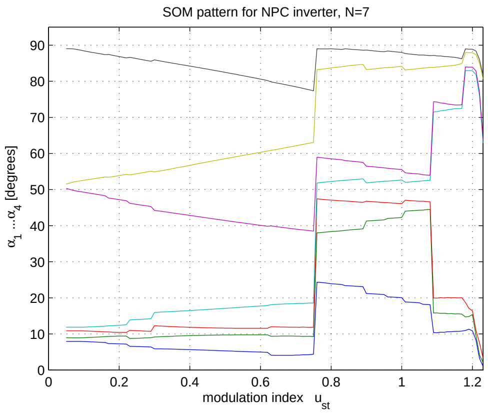
(b) WTHD0



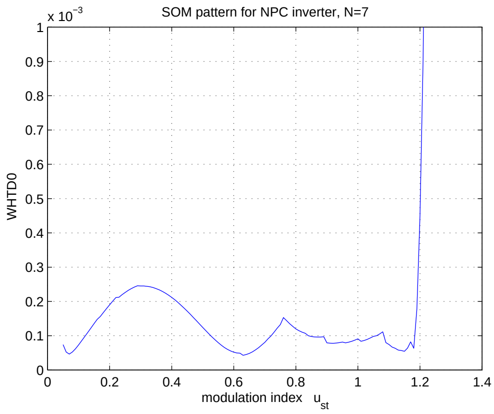
(c) harmonics

Figure C.5: SOM angular switching-pattern with corresponding WTHD0 and harmonic amplitudes, N=6

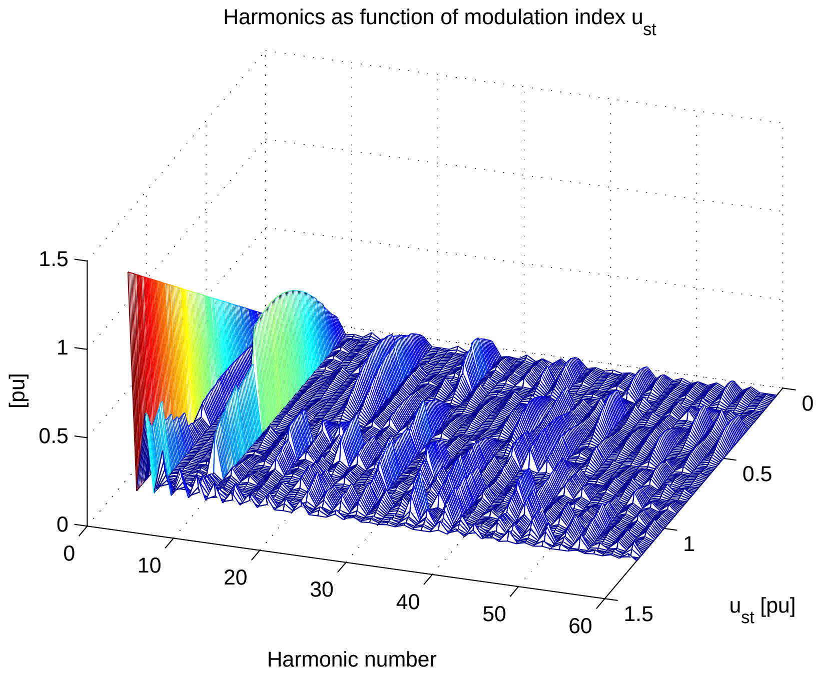
C.5 N=7



(a) Angular Switching instants with respect to modulation index



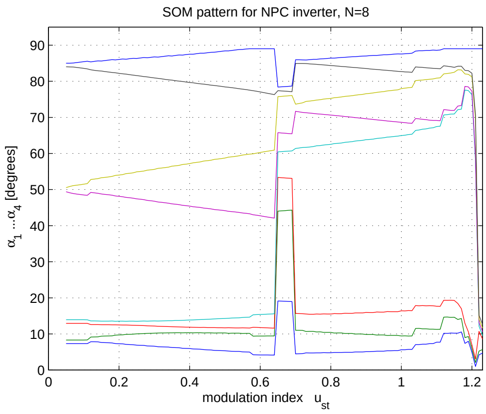
(b) WTHD0



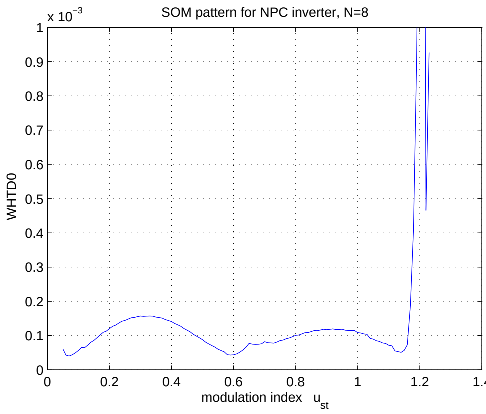
(c) harmonics

Figure C.6: SOM angular switching-pattern with corresponding WTHD0 and harmonic amplitudes, N=7

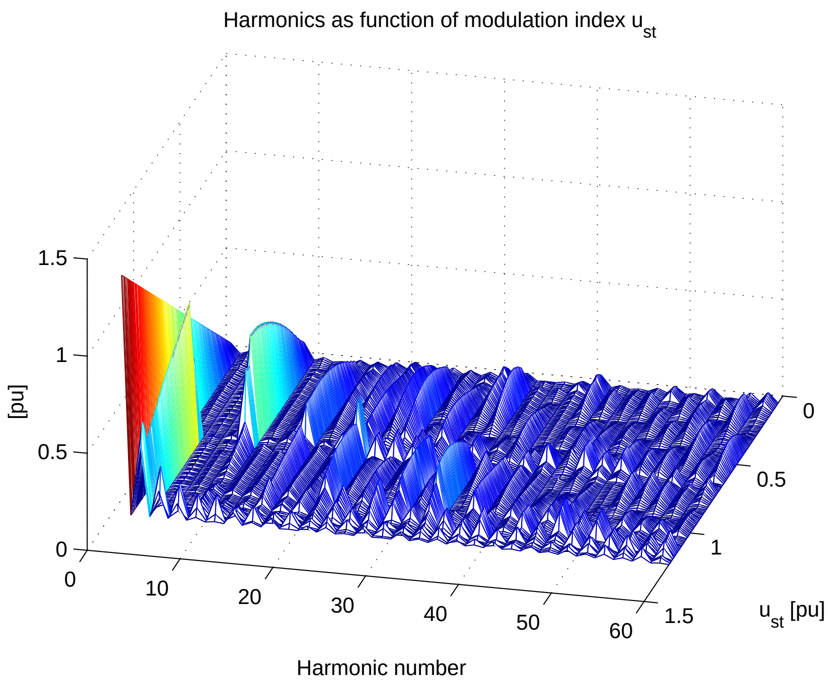
C.6 N=8



(a) Angular Switching instants with respect to modulation index



(b) WTHD0



(c) harmonics

Figure C.7: SOM angular switching-pattern with corresponding WTHD0 and harmonic amplitudes, N=8

Appendix D

Discreet Block Codes

It is advisable to read about Level 2 S-function in the MATLAB help menu before making any modifications in the codes.

D.1 Switching Angle Calculator

This code is used in the Switching Angle Calculator. The function of this block was to generate the switching angles for a full fundamental period from the quarter wave angle set given by the Pattern Selector.

```
function angle_calculator(block)
%Level 2 S-function created to calculate switching angles from
    quarter-wave symmetric switcing angle sets for a Programmed
    Modulation model.

%Written by Roger Enes
  setup(block);
%endfunction

function setup(block)

  %% Register number of input and output ports
  block.NumInputPorts  = 1;
  block.NumOutputPorts = 1;

  %% Setup functional port properties to dynamically
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

  block.InputPort(1).Dimensions  = -1;


  block.OutputPort(1).Dimensions  = 40;



  %% Set block sample time
```

```
                         %Sample time
  block.SampleTimes = [0.000005 0];

  %% Set the block simStateCompliance to default (i.e., same as a built-in block)
 block.SimStateCompliance = 'DefaultSimState';

  %% Run accelerator on TLC
  block.SetAccelRunOnTLC(true);

  %% Register methods

block.RegBlockMethod('SetInputPortDimensions',      @SetInpPortDims);
block.RegBlockMethod('SetOutputPortDimensions',     @SetOutPortDims);
block.RegBlockMethod('Outputs',                     @Output);
%endfunction

function SetInpPortDims(block, idx, di)
  block.InputPort(idx).Dimensions = di;

  %endfunction
  function SetOutPortDims(block, idx, di)
  block.OutputPort(idx).Dimensions = di;
   %endfunction

function Output(block)

% ---------------Switching angles from quarter to full wave ------------
a=zeros(1,40);

for i=1:10
    a(i)=block.InputPort(1).data(i);
end

for i=11:40
if i>=11 && i<=20                   %second quarter
   a(i)=180-a(21-i);
    elseif i>=21 && i<=30           %third quarter
       a(i)=180+a(i-20);
    elseif i>=31 && i<=40           %fourth quarter
       a(i)=360-a(41-i);
end
end

b=a;
% ----------Angle update, unused angles(0) are set to zero ------------
% A switching-angle with an angular value of 0 deg is treated as an non-exixtent switching instant.

for i=1:10

    if a(i)==0
```

```
        b(i:(21-i))=0;
        b((20+i):(41-i))=0;
    end

end
block.OutputPort(1).data=b;
```

D.2 Modulator

```
function modulator_x4(block)
% This is the s-function code for the Modulator user-defined block in the
% Programming Modulation model.

%Written by Roger Enes as a part of his master thesis.
  setup(block);

%endfunction

function setup(block)

  %% Register number of input and output ports
  block.NumInputPorts  = 2;
  block.NumOutputPorts = 1;

  %% Setup functional port properties to dynamically
  %% inherited.
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

  block.InputPort(1).Dimensions   = -1;
  block.InputPort(1).DirectFeedthrough = true;

  block.InputPort(2).Dimensions   = 1;
  block.InputPort(2).DirectFeedthrough = true;


  block.OutputPort(1).Dimensions   = 1;
  %block.OutputPort(2).Dimensions   = 1;

  %% Set block sample time to inherited

  block.SampleTimes = [0.000005 0];




  %% Set the block simStateCompliance to default (i.e., same as a built-in block)
 block.SimStateCompliance = 'DefaultSimState';

  %% Run accelerator on TLC
  block.SetAccelRunOnTLC(true);

  %% Register methods


block.RegBlockMethod('SetInputPortDimensions',      @SetInpPortDims);
block.RegBlockMethod('SetOutputPortDimensions',     @SetOutPortDims);
block.RegBlockMethod('Outputs',                 @Output);
```

```
%endfunction

function SetInpPortDims(block, idx, di)
  block.InputPort(idx).Dimensions = di;
  %endfunction
  function SetOutPortDims(block, idx, di)
  block.OutputPort(idx).Dimensions = di;
   %endfunction


function Output(block)


% Angle driver (Ad) is a periodically running from 0 to 360 degrees
    .

ang=block.InputPort(2).data(1);
Pc=floor(ang/360);
Ad=ang-360*Pc;
% ---------------Sets all switching angles into a matrix
    ------------
a=zeros(1,40);

for i=1:40
    a(i)=block.InputPort(1).data(i);
end


% -----------------------Angle counter(Ac) setup
    ----------------------
% This k counts everytime Ad passes an angle value.

k=0;
%block.Dwork(1).Data=0;

for i=1:40
    if a(i)~=0
      if Ad > a(i)
       k=i;
      end
    end
end

%--------------------------Gate signal generation
    --------------------

if k<=20
    if mod(k,2)~=0
    U_ss=1;
    else U_ss=0;
    end
else
    if mod(k,2)~=0
    U_ss=-1;
```

```
    else U_ss=0;
    end
end

block.OutputPort(1).data=U_ss;
```

D.3 Pattern Selector

```
function [pwm] = PatternSelector(u1)
%function y = fcn(u1)
%#codegen

pwm=zeros(1,10);

N=u1(2);
m=u1(1);

%Choice of modulation methode: 1 is Optimal Pulse Width Modulation(
    OPWM) and 0
%is Harmonic Elimination Method (HEM)


OPWM_or_HEM=1;


%_______________________OPWM_____________________

if OPWM_or_HEM==1
    if N==2

    elseif N==3


        for i=2:length(m_data)
            if  m<=m_data(1)
            pwm(1:N)=P_m_N3(:,1);
            break
            elseif m<m_data(i+1) && m>m_data(i-1)
            pwm(1:N)=P_m_N3(:,i)';
            break
            end
        end

    elseif N==4


        for i=2:length(m_data)
            if  m<=m_data(1)
            pwm(1:N)=P_m_N4(:,1);
            break
            elseif m<m_data(i+1) && m>m_data(i-1)
            pwm(1:N)=P_m_N4(:,i)';
            break
            end
        end

    elseif N==5


        for i=2:length(m_data)
            if  m<=m_data(1)
```

```
                pwm(1:N)=P_m_N5(:,1);
                break
                elseif m<m_data(i+1) && m>m_data(i-1)
                pwm(1:N)=P_m_N5(:,i)';
                break
                end
            end

        elseif N==6

            for i=2:length(m_data)
            if  m<=m_data(1)
                pwm(1:N)=P_m_N6(:,1);
                break
            elseif m<m_data(i+1) && m>m_data(i-1)
                pwm(1:N)=P_m_N6(:,i)';
                break
            end
            end


        elseif N==7
                    for i=2:length(m_data)
                if  m<=m_data(1)
                pwm(1:N)=P_m_N7(:,1);
                break
                elseif m<m_data(i+1) && m>m_data(i-1)
                pwm(1:N)=P_m_N7(:,i)';
                break
                end
            end

        elseif N==8
            P_m_N8=[
            m_data=[
                for i=2:length(m_data)
                if  m<=m_data(1)
                pwm(1:N)=P_m_N8(:,1);
                break
                elseif  m>=m_data(118)+(m_data(119)-m_data(118))/2
                pwm(1:N)=P_m_N8(:,119);
                break
                elseif m<m_data(i+1) && m>m_data(i-1)
                pwm(1:N)=P_m_N8(:,i)';
                break
                end
            end
        elseif N==9

        elseif N==10
        end
end
```

```
%________________________HEM________________________
if OPWM_or_HEM==0
    if N==2

    elseif N==3

    elseif N==4

    elseif N==5

    elseif N==6
        % HEM P_m_N6 and m_data are both derived from the N=8
           routine. E.g N=8-6.
        P_m_N6=[
        m_data=[
        for i=2:length(m_data)
            if  m<m_data(1)
            pwm(1:N)=P_m_N6(:,1);
            break
            elseif m<m_data(i+1) && m>m_data(i-1)
            pwm(1:N)=P_m_N6(:,i)';
            break
            end
        end


    elseif N==7

    elseif N==8

    elseif N==9

    elseif N==10
    end
end


%if u1(1)==1 && u1(2)==8

    %opwm=[9.077,14.97,19.67,62.47,64.92,73.61,78.19,86.45,0 0];
%elseif u1(1)==0.8 && u1(2)==8

    %opwm=[20.09 27.9 33.45 42.37 53.99 58.27 66.99 73.17 0 0 ];
%elseif  u1(1)==0.8 || u1(1)==1 && u1(2)==7

    %opwm=[10 11 20 21 30 31 47 0 0 0 ];


%end

```

```
end
```

D.4 Optimal Stator Flux Trajectory Calculator

```
function optimal_flux2(block)
%% Level-2 MATLAB file S-Function for times two demo.
%This level-2 S-function code is used to operate the optimal_flux2
    block in
%the Programming Modulation model. The code is used to find the
    optimal
%stator flux trajectory, at any instant.

%Written by Roger Enes as a part of his master thesis.
  setup(block);

%endfunction

function setup(block)

  %% Register number of input and output ports
  block.NumInputPorts  = 3;
  block.NumOutputPorts = 1;

  %% Setup functional port properties to dynamically
  %% inherited.
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

  block.InputPort(1).Dimensions  = -1;
  block.InputPort(1).DirectFeedthrough = true;

  block.InputPort(2).Dimensions  = 1;
  block.InputPort(2).DirectFeedthrough = true;

  block.InputPort(3).Dimensions  = 1;
  block.InputPort(3).DirectFeedthrough = true;

  block.OutputPort(1).Dimensions  = 1;

  %block.OutputPort(2).Dimensions  = 1;

  %% Set block sample time to inherited

  block.SampleTimes = [0.00005 0];



  %% Set the block simStateCompliance to default (i.e., same as a
      built-in block)
 block.SimStateCompliance = 'DefaultSimState';

  %% Run accelerator on TLC
  block.SetAccelRunOnTLC(true);

  % Register methods

```

```
block.RegBlockMethod('SetInputPortDimensions',       @SetInpPortDims
    );
block.RegBlockMethod('SetOutputPortDimensions',      @SetOutPortDims
    );

block.RegBlockMethod('Outputs',                  @Output);


%endfunction


 %
    function SetInpPortDims(block, idx, di)
  block.InputPort(idx).Dimensions = di;
  %endfunction
  function SetOutPortDims(block, idx, di)
  block.OutputPort(idx).Dimensions = di;
%endfunction


function Output(block)



%frequency
f=block.InputPort(3).data;

%Half wave
h=180;

% Angle driver (Ad) is a periodically running from 0 to 360 degrees
    .

ang=block.InputPort(2).data(1);
Pc=floor(ang/360);
Ad=ang-360*Pc;



% ---------------Sets all switching-angles into a vector
    ------------
a=zeros(1,40);

for i=1:40
    a(i)=block.InputPort(1).data(i);
end

% -----------------------Angle counter(k) setup
    ----------------------
% k counts everytime Ad passes a switching-angle.

k=0;
for i=1:40
    if a(i)~=0
      if Ad > a(i)
```

```
            k=i;
          end
      end
end

% Make even and odd counters which will be used to calculate the optimal
% Psi trajectory.

ke=0;       %counter even
ko=0;       %counter odd

if k==1 || k==0
    ke=1;
    ko=1;
elseif mod(k,2)==0
    ke=k;
    ko=k-1;
else
    ke=k-1;
    ko=k-2;
end

% -------------------- Optimal Flux linkage ---------------------------

%Discreet integration of U_ss

b=a*(1/(360*f));
Psi=0;
Ad_t=Ad*(1/(360*f));

if Ad<=h

    if mod(k,2)==0
     Psi=sum(b(2:2:ke)-b(1:2:ko));
    else Psi=sum(b(2:2:ke)-b(1:2:ko))+(Ad_t-b(k));
    end

elseif Ad>h && Ad<=a(21)
    Psi=sum(b(2:2:ke)-b(1:2:ko));

elseif Ad>a(21)

     if mod(k,2)==0
     Psi=sum(b(2:2:20)-b(1:2:19))-sum(b(22:2:ke)-b(21:2:ko));
     else Psi=sum(b(2:2:20)-b(1:2:19))-sum(b(22:2:ke)-b(21:2:ko))-(Ad_t-b(k));
     end

end

%Offset compencation
Psi_ofs=sum(b(2:2:40)-b(1:2:39))*0.25;
Psi_ss=Psi-Psi_ofs;

```

```
block.OutputPort(1).data=Psi_ss;
%endfunction
```

D.5 Stator Flux Trajectory Controller - Override mode

```
function modulation_error_compensator_override_v1(block)
%% Level-2 MATLAB file S-Function
%  This MATLAB s-Function is used to control the Stator Flux
   Trajectory
%  Controller user-defined MATLAB function, in override mode.

%Hint: To understand how the algorithm operates or "thinks", draw a
    PWM pattern and put
%in Ad, Ad_next, k and k2 values, then go through the "if"
   functions.

%Written by Roger Enes as a part of his master thesis.


  setup(block);

%endfunction

function setup(block)

  %% Register number of input and output ports
  block.NumInputPorts  = 4;
  block.NumOutputPorts = 1;

  %% Setup functional port properties to dynamically
  %% inherited.
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

  block.InputPort(1).Dimensions  = -1;
  block.InputPort(1).DirectFeedthrough = true;

  block.InputPort(2).Dimensions  = 1;
  block.InputPort(2).DirectFeedthrough = true;

  block.InputPort(3).Dimensions  = 1;
  block.InputPort(3).DirectFeedthrough = true;

  block.InputPort(4).Dimensions  = 1;
  block.InputPort(4).DirectFeedthrough = true;

  block.OutputPort(1).Dimensions  = 40;


  %% Set block sample time to inherited [-2 0]

  block.SampleTimes = [-2 0];

  %% Set the block simStateCompliance to default (i.e., same as a
     built-in block)
```

```
 block.SimStateCompliance = 'DefaultSimState';

  %% Run accelerator on TLC
  block.SetAccelRunOnTLC(true);
  % Register methods
block.RegBlockMethod('SetInputPortSamplingMode',
   @SetInputPortSamplingMode);
block.RegBlockMethod('PostPropagationSetup',    @DoPostPropSetup);
block.RegBlockMethod('SetInputPortDimensions',      @SetInpPortDims
   );
block.RegBlockMethod('SetOutputPortDimensions',     @SetOutPortDims
   );
block.RegBlockMethod('InitializeConditions',    @InitConditions);
block.RegBlockMethod('Outputs',                 @Output);
block.RegBlockMethod('Update',                 @Update);
%endfunction



    function SetInpPortDims(block, idx, di)
  block.InputPort(idx).Dimensions = di;
  %endfunction
  function SetOutPortDims(block, idx, di)
  block.OutputPort(idx).Dimensions = di;
%endfunction


function SetInputPortSamplingMode(block, idx, fd)
block.InputPort(1).SamplingMode = 0.0001;
block.InputPort(idx).SamplingMode = fd;

block.OutputPort(1).SamplingMode = fd;
%endfunction


function DoPostPropSetup(block)

  %% Setup Dwork
  block.NumDworks = 1;

  block.Dwork(1).Name = 'k';
  block.Dwork(1).Dimensions      = 40;
  block.Dwork(1).DatatypeID      = 0;
  block.Dwork(1).Complexity      = 'Real';
  block.Dwork(1).UsedAsDiscState = true;


%endfunction
    function InitConditions(block)

%% Initialize Dwork
  block.Dwork(1).Data = zeros(1,40);
%endfunction
```

```
    function Output(block)
%-------------------------------------------------Trajectory Controller(pattern altering)----------------------------------------------------------

% ---------------------OUTPUT PRE-CALCULATION SETUP ----------------------

% Frequency
f=block.InputPort(4).data;

% ______Specify sampling time in degrees or time _____


sampling_in_time=0.0005;
sampling_in_degrees=7;

time_or_angle=1; %1 is constant time based sampling. Any other value will result in an angular(frequency dependent) samling-interval
if time_or_angle==1
    sample=sampling_in_time*360*f;
else sample=sampling_in_degrees;
end

ang_sample=sample;
%____________________________________________________________

%Half wave
h=180;

% Angle driver (Ad) is the periodic angle with respect to
% frequency (ang). Pc is a periodic conter.

ang=block.InputPort(2).data(1);
Pc=floor(ang/360);
Ad=ang-360*Pc;

% To avoid setting a(1) to zero during the first global simulation sample.
if Ad==0
    Ad=0.001;
end

% Converting seconds to degrees
d_t=block.InputPort(3).data;                    %Flux deviation given in Vs
d=d_t*360*f;                                    %Flux deviation given in Vs

%Defining the vector b where calculated pattern changes are stored during a
%sample.
b=zeros(1,40);

%Ad_next is the angular value for the next sample hit.
```

```
Ad_next=Ad+ang_sample;
if Ad_next>360
    Ad_next=Ad_next-360;
end


%Sets all descreet switching angles into a vector
a=zeros(1,40);
for i=1:40
    a(i)=block.InputPort(1).data(i);
end

%Determining how many zero angles there are in the first 10 angles to
%determine the number of "real" swiching angles in the operating pattern.
for i=1:10
    if a(i)==0
        number_of_zero_angles=11-i;
        last_real_angle=i-1;
        break
    end
end

% -----------------------Angle counters(k and k2)setup ----------------------
% k and k2 counts everytime Ad and Ad_next passes a switching-angle
% , and it jumps over inactive switching-angles(zero angles). k is
% the counter for the momentary angles, k2 is a counter for the next
% sample hit, e.g. given by Ad_next.
k=0;
for i=1:40
    if a(i)~=0
      if Ad > a(i)
       k=i;
      end
    end
end


k2=0;
for i=1:40
    if a(i)~=0
      if Ad_next > a(i)
       k2=i;
      end
    end
end



% Maximum switching events within one samplin period is 2, this "if -function"
% is therefore limiting the number of switching events considered within a
```

```
% sampling period.

if k2-k>2 || k>30 && k2<10
    if k==last_real_angle && k2>k+2+number_of_zero_angles*2 || k==last_real_angle+20 && k2>k+2+number_of_zero_angles*2
        k2=k+2+number_of_zero_angles*2;
    elseif k==last_real_angle-1 && k2>=k+2+number_of_zero_angles*2 || k==last_real_angle+19 && k2>k+2+number_of_zero_angles*2
        k2=k+2+number_of_zero_angles*2;
    elseif k>=39 && k2<10 && k2>0
        k2=k-38;
    elseif k2-k>2 && k~=last_real_angle && k~=last_real_angle+20 && k~=last_real_angle-1 && k~=last_real_angle+19 || k>30 && k<39 && k2<k
        k2=k+2;
    end
end

%When k is larger than 20 are we standing in the negative half-period. By
%changing the polarity of the dynamic modulation error can the
%pattern-modifying operation used in the positive half period be used in
%the negative half period, of the fundamentl frequency.
if k>=20
    d=-d;
end

% ----------------------END OF SETUP -------------------------------------




%%       ------------------BLOCK OPERATIONS --------------------    %%


%If the switching inherit two switching transactions within one sampling period this code will be active %%

%When a sampling period overlaps the positive and the negative half-period and contains two switching instants---

%-------------SPECIAL CASES---------------

%The sampling period overlaps end-beginning of a fundamental period

% Case I
if k==39 && k2 ==1
    d=-d;                          %This is to reverse the effect in comand line 212 - 214
    if d>0
```

```
        if d>a(40)-Ad+a(1)-0.1    % Must have a smal buffer(0.1) since a switching angle with 0 degrees is regarded as a non-existent switching angle
            b(40)=Ad-a(40);
            b(1)=-a(1)+0.1;
        elseif d<a(40)-Ad+a(1)-0.1
            if d<a(40)-Ad
                b(40)=-d;
            elseif d>a(40)-Ad
                b(40)=Ad-a(40);
                b(1)=-d-b(40);
            end
        end
    elseif d<0
        if d<a(40)-360+a(1)-Ad_next
            b(40)=(360-a(40));
            b(1)=Ad_next-a(1);
        elseif d>a(40)-360+a(1)-Ad_next
            if d>a(40)-360
                b(40)=-d;
            elseif d<a(40)-360
                b(40)=(360-a(40));
                b(1)=-d-b(40);
            end
        end
    end
end


%Case II
if k==40 && k2==2
    d=-d;                                                   %This is to reverse the effect in comand line 212 - 214
        if d>0
            if d>(a(k2-1)+Ad_next-a(k2))-0.1     % the use of 0.1 is done because setting the angle to zero means that it dose not exist.
                b(k2-1)=-a(1)+0.1;                % the use of 0.1 is done because setting the angle to zero means that it dose not exist.
                b(k2)=Ad_next-a(k2);
            elseif d<(a(k2-1)+Ad_next-a(k2))
                if d<(a(k2-1))-0.1
                    b(k2-1)=d*(-1);
                elseif d>(a(k2-1))-0.1
                    b(k2-1)=-a(k2-1)+0.1;
                    b(k2)=(d+b(k2-1));
                end
            end

        elseif d<0
            if d<(a(k2-1)-a(k2))
                 b(k2-1)=-a(k2-1);
                 b(k2)=-a(k2);
            elseif d>(a(k2-1)-a(k2))
                 b(k2-1)=-d;
```

```
            end
        end
end



% The following code is used if the sampling period overlaps a "
    jump" over zero angles

if k==last_real_angle && k2==k+number_of_zero_angles*2+2 || k==
    last_real_angle+20 && k2==k+number_of_zero_angles*2+2

    if mod(k,2)~=0
        d=-d; %This will make the controller compatible with odd k'
            s when "jumping" over zero angles.
    end

    if d>0
            if d>(a(k2-1)-Ad+Ad_next-a(k2))
                b(k2-1)=Ad-a(k2-1);
                b(k2)=Ad_next-a(k2);
            elseif d<(a(k2-1)-Ad+Ad_next-a(k2))
                if d<(a(k2-1)-Ad)
                    b(k2-1)=d*(-1);
                elseif d>(a(k2-1)-Ad)
                    b(k2-1)=(Ad-a(k2-1));
                    b(k2)=(d+b(k2-1));
                end
            end

        elseif d<0
            if d<(a(k2-1)-a(k2))
                 b(k2-1)=-a(k2-1);
                 b(k2)=-a(k2);
            elseif d>(a(k2-1)-a(k2))
                 b(k2-1)=-d;
            end
     end
end

% The case where k==last_real_angle-1 is dealt with in the general
    case
% below.



% When the sampling time is crossing the end/beginning interface
    but all
% modifications are going to be executed in the foregoing period,
    Ad_next has to be
% moved as shown under. If not, problems with the last switching
    angles in a period will arrise.

```

```
if k<k2 && Ad_next<Ad
    Ad_next=360;
end

%-----The general case-----------

%When the sampling period contains TWO switching instants and it is NOT
%crossing any special areas. hence, the general case.


if k2-k==2 && k~=19 && k~= 39 && k~=40 && k~=last_real_angle && k~=last_real_angle+20  ||  k==last_real_angle-1 && k2==k+number_of_zero_angles*2+2 ||  k==last_real_angle+19 && k2==k+number_of_zero_angles*2+2 % The creteria and the exeptions when special cases is evaluated.

    if mod(k,2)~=0                                %This impies that "we are standing" in a pulse and looking towards Ad_next. Tips! Draw a PWM pattern :)
        if d>0
            if d>(a(k2)-a(k+1))                   %Checking if the modulation error is arger than posible correction in this samplig period
                b(k+1)=-a(k+1);                   %This will remove the switching angle in the modulation block.
                b(k2)=-a(k2);                     %This will remove the switching angle in the modulation block. Together with the foregoing code-line, a whole pulse is removed and maximum contribution in Vs is obtained.
            elseif d<(a(k2)-a(k+1))               %if the modulation error is so small that it will be eliminated during this sampling period.
                b(k+1)=d;
            end

        elseif d<0                                %Same as abowe, but the modulation error is negative.
            if d<(Ad-a(k+1)+a(k2)-Ad_next)
                b(k+1)=Ad-a(k+1);
                b(k2)=Ad_next-a(k2);
            elseif d>(Ad-a(k+1)+a(k2)-Ad_next)
                if d>(Ad-a(k+1))
                    b(k+1)=d;
                elseif d<(Ad-a(k+1))
                    b(k+1)=(Ad-a(k+1));
                    b(k2)=(d-b(k+1))*(-1);        %multiplie with (-1) because the switching k+2 must be delayed to give a negative Vs contribution.
                end
            end
        end

    elseif mod(k,2)==0                            %This implies that "we are standing" in-between two pulses if the d originally
```

```
        is positive and not negative due to command line 211 - 214.
        if d>0
            if d>(a(k+1)-Ad+Ad_next-a(k2))
                b(k+1)=Ad-a(k+1)+0.01;
                b(k2)=Ad_next-a(k2);
            elseif d<(a(k+1)-Ad+Ad_next-a(k2))
                if d<(a(k+1)-Ad)
                    b(k+1)=d*(-1);
                elseif d>(a(k+1)-Ad)
                    b(k+1)=(Ad-a(k+1));
                    b(k2)=(d+b(k+1));
                end
            end

        elseif d<0
            if d<(a(k+1)-a(k2))                  %Checking if the
               modulation error is arger than posible correction in
                this samplig period
                b(k+1)=-a(k+1);                   %This will remove
                    the switching angle in the modulation block.
                b(k2)=-a(k2);                     %This will remove
                    the switching-angle in the modulation block.
            elseif d>(a(k+1)-a(k2))               %The modulation
               error is so small that it will be eliminated in this
                sampling period
                b(k+1)=-d;
            end
        end
    end
end



%%    If the switching interval only inherit one transaction %%


Ad_next=Ad+ang_sample;
if Ad_next>360
    Ad_next=Ad_next-360;
end

%General cases                          Special case I
                                        special case II
if k2-k==1       || k==last_real_angle && k2==k+1+
   number_of_zero_angles*2 || k==last_real_angle+20 && k2==k+1+
   number_of_zero_angles*2   % common: only one transaction

    if mod(k,2)==0 && d<0          %We are standing in between two
       pulses and looking forward to Ad_next.
        if d<a(k2)-Ad_next
            b(k2)=Ad_next-a(k2);
        else b(k2)=-d;
        end


    elseif mod(k,2)==0 && d>0
```

```
        if d>a(k2)-Ad
            b(k2)=Ad-a(k2);
        else b(k2)=-d;
        end


    elseif mod(k,2)~=0 && d<0       %We are standing in a pulse and looking forward to Ad_next.
        if d<Ad-a(k2)
            b(k2)=Ad-a(k2);
        else b(k2)=d;
        end


    elseif mod(k,2)~=0 && d>0
        if d>Ad_next-a(k2)
            b(k2)=Ad_next-a(k2);
        else b(k2)=d;
        end

    end
end

% ----End of period effect-----

if k==40 && k2==1
    d=-d;                            %this is to revers the previous d=-d, see comand line 211 - 214.
    if d<0
        if d<a(k2)-Ad_next
            b(k2)=Ad_next-a(k2);
        else b(k2)=d;
        end

    elseif d>0
        if d>a(k2)-0.01
            b(k2)=-a(k2)+0.01;
        else b(k2)=-d;
        end
    end
end



%%


D_P=b;

block.OutputPort(1).data=D_P;

% ----------------Set time for next sample period -------------------------
%By activating Alt II will the block force a new sample after 0.1 ms when a
```

```
%pattern change is done. This will give a faster dynamic modulation
    error
%compansation since the block do not need to wait until the end of
    a
%sampling period to execute countermeasures.

%Alt I
block.NextTimeHit = block.CurrentTime+ang_sample/(360*f)+0/(360*f);

%Alt II

%   if block.Dwork(1).Data==block.InputPort(1).Data
%   block.NextTimeHit = block.CurrentTime+ang_sample/(360*f)+0/(360*
    f);
%   else block.NextTimeHit = block.CurrentTime+0.0001;
%   end

        function Update(block)
block.Dwork(1).Data=block.InputPort(1).Data;

%endfunction
```

D.6 Stator Flux Trajectory Controller - Synchronous mode

```
function modulation_error_compensator_synchronous_v1(block)
%% Level-2 MATLAB file S-Function
%  This MATLAB s-Function is used to control the Stator Flux
   Trajectory
%  Controller user-defined MATLAB function, in SYNCHRONOUS mode.

%Hint: To understand how the algorithm operates or "thinks", draw a
    PWM pattern and put
%in Ad, Ad_next, k and k2 values, then go through the "if"
   functions.

%Written by Roger Enes as a part of his master thesis.

  setup(block);

%endfunction

function setup(block)

  %% Register number of input and output ports
  block.NumInputPorts  = 4;
  block.NumOutputPorts = 1;

  %% Setup functional port properties to dynamically
  %% inherited.
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;

  block.InputPort(1).Dimensions  = -1;
  block.InputPort(1).DirectFeedthrough = true;

  block.InputPort(2).Dimensions  = 1;
  block.InputPort(2).DirectFeedthrough = true;

  block.InputPort(3).Dimensions  = 1;
  block.InputPort(3).DirectFeedthrough = true;

  block.InputPort(4).Dimensions  = 1;
  block.InputPort(4).DirectFeedthrough = true;


  block.OutputPort(1).Dimensions  = 40;


  %% Set block sample time to inherited

  block.SampleTimes = [-2 0];

  %% Set the block simStateCompliance to default (i.e., same as a
     built-in block)
```

```
 block.SimStateCompliance = 'DefaultSimState';

  %% Run accelerator on TLC
  block.SetAccelRunOnTLC(true);
  % Register methods
block.RegBlockMethod('SetInputPortSamplingMode',
   @SetInputPortSamplingMode);
block.RegBlockMethod('PostPropagationSetup',    @DoPostPropSetup);
block.RegBlockMethod('SetInputPortDimensions',      @SetInpPortDims
   );
block.RegBlockMethod('SetOutputPortDimensions',     @SetOutPortDims
   );
block.RegBlockMethod('InitializeConditions',    @InitConditions);
block.RegBlockMethod('Outputs',                 @Output);
block.RegBlockMethod('Update',                 @Update);
%endfunction

    function SetInpPortDims(block, idx, di)
  block.InputPort(idx).Dimensions = di;
  %endfunction
  function SetOutPortDims(block, idx, di)
  block.OutputPort(idx).Dimensions = di;
%endfunction

function SetInputPortSamplingMode(block, idx, fd)
block.InputPort(1).SamplingMode = 0.0001;
block.InputPort(idx).SamplingMode = fd;

block.OutputPort(1).SamplingMode = fd;

function DoPostPropSetup(block)

  %% Setup Dwork
  block.NumDworks = 1;

  block.Dwork(1).Name = 'k';
  block.Dwork(1).Dimensions      = 40;
  block.Dwork(1).DatatypeID      = 0;
  block.Dwork(1).Complexity      = 'Real';
  block.Dwork(1).UsedAsDiscState = true;

%endfunction
    function InitConditions(block)

%% Initialize Dwork
  block.Dwork(1).Data = zeros(1,40);
%endfunction

    function Output(block)
%-------------------------------------------------Trajectory
   Controller(pattern altering)
   ----------------------------------------------------------

% ---------------------OUTPUT PRE-CALCULATION SETUP
   ----------------------

```

```
% Frequency
f=block.InputPort(4).data;


% ____________________Specify sampling time in degrees________________

sampling_in_time=0.0005;
sampling_in_degrees=7;

time_or_angle=1;        %1 is constant time based sampling. Any other value will result in an angular(frequency dependent) samling-interval
if time_or_angle==1
    sample=sampling_in_time*360*f;
else sample=sampling_in_degrees;
end

ang_sample=sample;

% ------------------------------------------------------------------

%Buffer between switching intervals. This is special for the synchronous mode of the Stator
%Flux Trajectory controller.

% change "buf" to chage the buffer between switching instatns.
buf=0.5;

%Half wave
h=180;

% Angle driver (Ad) is the periodic angle with respect to
% frequency (ang). Pc is a periodic conter.

ang=block.InputPort(2).data(1);
Pc=floor(ang/360);
Ad=ang-360*Pc;

% To avoid setting a(1) to zero.
if Ad==0
    Ad=0.001;
end

% Converting time to angle
d_t=block.InputPort(3).data;                    %Flux deviation given in Vs
d=d_t*360*f;                                    %Flux deviation given in Vs

%Defining the vector b where calculated pattern changes are stored during a
%sample.
b=zeros(1,40);
```

```
%Ad_next is the angular value for the next sample hit.
Ad_next=Ad+ang_sample;
if Ad_next>360
    Ad_next=Ad_next-360;
end


%Sets all descreet switching angles into a vector
a=zeros(1,40);
for i=1:40
    a(i)=block.InputPort(1).data(i);
end

%Determining how many zero angles there are in the first 10 angles to
%determine the number of swiching angles in the system.
for i=1:10
    if a(i)==0
        number_of_zero_angles=11-i;
        last_real_angle=i-1;
        break
    end
end

% ----------------------Angle counters(k and k2)setup ----------------------
% k and k2 counts everytime Ad and Ad_next passes a switching-angle
% , and it jumps over inactive switching-angles(zero angles). k is
% the counter for the momentary angles, k2 is a counter for the next
% sample hit, e.g. given by Ad_next.

k=0;
for i=1:40
    if a(i)~=0
      if Ad > a(i)
       k=i;
      end
    end
end


k2=0;
for i=1:40
    if a(i)~=0
      if Ad_next > a(i)
       k2=i;
      end
    end
end


% Maximum switching events within one samplin period are 2, this "if-function"
```

```
% is therefore limiting the number of switching events considered within a
% sampling period.

if k2-k>2 || k>30 && k2<10
    if k==last_real_angle && k2>k+2+number_of_zero_angles*2 || k==last_real_angle+20 && k2>k+2+number_of_zero_angles*2
        k2=k+2+number_of_zero_angles*2;
    elseif k==last_real_angle-1 && k2>=k+2+number_of_zero_angles*2 || k==last_real_angle+19 && k2>k+2+number_of_zero_angles*2
        k2=k+2+number_of_zero_angles*2;
    elseif k>=39 && k2<10 && k2>1
        k2=k-38;
    elseif k2-k>2 && k~=last_real_angle && k~=last_real_angle+20 && k~=last_real_angle-1 && k~=last_real_angle+19 || k>30 && k<39 && k2<k
        k2=k+2;
    end
end



%When k is larger than 20 are we standing in the negative half-period. By
%changing the polarity of the dynamic modulation error can the
%pattern-modifying operation used in the positive half period be used in
%the negative half period, of the fundamentl frequency. This effecet is reversed in some of the special cases.
if k>=20
    d=-d;
end

% ----------------------END OF SETUP -------------------------------------






%%       ------------------BLOCK OPERATIONS --------------------      %%



%If the switching inherit two switching transactions within one sampling period this code will be active %%

%When a sampling period overlaps the positive and the negative hvlf period and contains two switching instants---

%-------------SPECIAL CASES---------------

%The sampling period overlaps end-beginning of a fundamental period
```

```
% Case I
if k==39 && k2 ==1
    d=-d;                              %this is to revers the previous d=-d, see comand line 217-219.
    if d>0
        if d>a(40)-Ad+a(1)-0.1-buf     % Must have a smal buffer (0.1) since a switching angle with 0 degrees is regarded as a non-existent switching angle
            b(40)=Ad-a(40)+buf;
            b(1)=-a(1)+0.1;
        elseif d<a(40)-Ad+a(1)-0.1-buf
            if d<a(40)-Ad-buf
                b(40)=-d;
            elseif d>a(40)-Ad-buf
                b(40)=Ad-a(40)+buf;
                b(1)=-d-b(40);
            end
        end
    elseif d<0
        if d<a(40)-360+a(1)-Ad_next+buf
            b(40)=(360-a(40));
            b(1)=Ad_next-a(1)-buf;
        elseif d>a(40)-360+a(1)-Ad_next+buf
            if d>a(40)-360
                b(40)=-d;
            elseif d<a(40)-360
                b(40)=(360-a(40));
                b(1)=-d-b(40);
            end
        end
    end
end


%Case II
if k==40 && k2==2
    d=-d;                                                   %this is to revers the previous d=-d, see comand line 217-219.
        if d>0
            if d>(a(k2-1)+Ad_next-a(k2))-0.1-buf
                b(k2-1)=-a(1)+0.1;                          % the use of 0.1 is done because setting the angle to zero means that it dose not exist.
                b(k2)=Ad_next-a(k2)-buf;
            elseif d<(a(k2-1)+Ad_next-a(k2)-buf)
                if d<(a(k2-1))-0.1
                    b(k2-1)=d*(-1);
                elseif d>(a(k2-1))-0.1
                    b(k2-1)=-a(k2-1)+0.1;
                    b(k2)=(d+b(k2-1));
                end
            end

        elseif d<0
            if d<(a(k2-1)-a(k2)+buf)
                 b(k2-1)=a(k2)-a(k2-1)-buf;
```

```
            elseif d>(a(k2-1)-a(k2)+buf)
                 b(k2-1)=-d;
            end
        end
end





% The sampling period overlaps the "jump" over zero angles

if k==last_real_angle && k2==k+number_of_zero_angles*2+2 || k==
   last_real_angle+20 && k2==k+number_of_zero_angles*2+2

    if mod(k,2)~=0
        d=-d; %This will make the controller compatible with odd k'
           s when "jumping" over zero angles.
    end

    if d>0
            if d>(a(k2-1)-Ad+Ad_next-a(k2)-2*buf)
                b(k2-1)=Ad-a(k2-1)+buf;
                b(k2)=Ad_next-a(k2)-buf;
            elseif d<(a(k2-1)-Ad+Ad_next-a(k2)-2*buf)
                if d<(a(k2-1)-Ad-buf)
                    b(k2-1)=d*(-1);
                elseif d>(a(k2-1)-Ad-buf)
                    b(k2-1)=(Ad-a(k2-1)+buf);
                    b(k2)=(d+b(k2-1));
                end
            end

        elseif d<0
            if d<(a(k2-1)-a(k2)+buf)
                 b(k2-1)=a(k2)-a(k2-1)-buf;
            elseif d>(a(k2-1)-a(k2)+buf)
                 b(k2-1)=-d;
            end
     end
end

% The case where k==last_real_angle-1 is dealt with in the general
   case
% below.


% When the sampling time is crossing the end/beginning interface
   but all
% modifications are going to be executed in the foregoing period,
   Ad_next has to be
% moved as shown under. If not, problems with the last switching
   angles in a period will arrise.
if k<k2 && Ad_next<Ad
    Ad_next=360;
```

```
end

%-----The general case-----------
%When the sampling period contains two switching instantsand and it is NOT
%crossing any special areas. hence, the general case.


if k2-k==2 && k~= 39 && k~=40 && k~=last_real_angle && k~=last_real_angle+20  ||  k==last_real_angle-1 && k2==k+number_of_zero_angles*2+2 || k==last_real_angle+19 && k2==k+number_of_zero_angles*2+2 % The creteria and the exeptions when special cases is evaluated.

    if mod(k,2)~=0                              %This impies that "we are standing" in a pulse and looking towards Ad_next. Tips! Draw a PWM pattern :)
        if d>0
            if d>(a(k2)-a(k+1)-buf)                     %Checking if the modulation error is larger than posible correction in this samplig period
                b(k+1)=a(k2)-a(k+1)-buf;
            elseif d<(a(k2)-a(k+1)-buf)                 %The modulation error is so smal that it will be eliminated in this sampling period
                b(k+1)=d;
            end

        elseif d<0                                  %Same as abowe, but the modulation error is negative
            if d<(Ad-a(k+1)+a(k2)-Ad_next+2*buf)
                b(k+1)=Ad-a(k+1)+buf;
                b(k2)=Ad_next-a(k2)-buf;
            elseif d>(Ad-a(k+1)+a(k2)-Ad_next+2*buf)
                if d>(Ad-a(k+1)+buf)
                    b(k+1)=d;
                elseif d<(Ad-a(k+1)+buf)
                    b(k+1)=(Ad-a(k+1)+buf);
                    b(k2)=(d-b(k+1))*(-1);      %multiplie with (-1) because the switching k+2 must be delayed to give a negative volt/sec contribution.
                end
            end
        end



    elseif mod(k,2)==0      %This implies that "we are standing" in-between two pulses if the d originally is positive and not negative due to command line 217 - 219.
        if d>0
            if d>(a(k+1)-Ad+Ad_next-a(k2)-2*buf)
                b(k+1)=Ad-a(k+1)+buf;
                b(k2)=Ad_next-a(k2)-buf;
```

```
                elseif d<(a(k+1)-Ad+Ad_next-a(k2)-2*buf)
                    if d<(a(k+1)-Ad-buf)
                        b(k+1)=d*(-1);
                    elseif d>(a(k+1)-Ad-buf)
                        b(k+1)=(Ad-a(k+1)+buf);
                        b(k2)=(d+b(k+1));
                    end
                end

            elseif d<0
                if d<(a(k+1)-a(k2)+buf)                        %Checking if
                    the modulation error is arger than posible
                    correction in this samplig period
                     b(k+1)=a(k2)-a(k+1)-buf;
                elseif d>(a(k+1)-a(k2)+buf)                    %The modulation
                     error is so smal that it will be eliminated after
                    this sampling period
                     b(k+1)=-d;
                end
            end
        end
end


%%    If the switching interval only inherit one transaction %%


Ad_next=Ad+ang_sample;
if Ad_next>360
    Ad_next=Ad_next-360;
end

%General cases                           Special case I
                                          special case II
if k2-k==1        || k==last_real_angle && k2==k+1+
    number_of_zero_angles*2 || k==last_real_angle+20 && k2==k+1+
    number_of_zero_angles*2   % common: only one transaction

    if mod(k,2)==0  && d<0          %We are standing in between two
       pulses and looking forward to Ad_next.
        if d<a(k2)-Ad_next+buf
            b(k2)=Ad_next-a(k2)-buf;
        else b(k2)=-d;
        end


    elseif mod(k,2)==0 && d>0
        if d>a(k2)-Ad -buf
            b(k2)=Ad-a(k2)+buf;
        else b(k2)=-d;
        end


    elseif mod(k,2)~=0 && d<0       %We are standing in a pulse and
       looking forward to Ad_next.
```

```
        if d<Ad-a(k2)+buf
            b(k2)=Ad-a(k2)+buf;
        else b(k2)=d;
        end


    elseif mod(k,2)~=0 && d>0
        if d>Ad_next-a(k2)-buf
            b(k2)=Ad_next-a(k2)-buf;
        else b(k2)=d;
        end

    end
end

% ----End of period effect-----

if k==40 && k2==1
    d=-d;                              %this is to revers the previous d =-d, see comand line 217-219.
    if d<0
        if d<a(k2)-Ad_next+buf
            b(k2)=Ad_next-a(k2)-buf;
        else b(k2)=d;
        end

    elseif d>0
        if d>a(k2)-0.01
            b(k2)=-a(k2)+0.01;
        else b(k2)=-d;
        end
    end
end

%%


D_P=b;

block.OutputPort(1).data=D_P;

% ----------------Set time for next sample period -------------------------
%By activating Alt II will the block force a new sample after 0.1 ms when a
%pattern change is done. This will give a faster dynamic modulation error
%compansation since the block do not need to wait until the end of a
%sampling period to exicute countermeasures.

%Alt I
block.NextTimeHit = block.CurrentTime+ang_sample/(360*f)+0/(360*f);

%Alt II
%if block.Dwork(1).Data==block.InputPort(1).Data
```

```
%block.NextTimeHit = block.CurrentTime+ang_sample/(360*f)+0/(360*f)
   ;
%else block.NextTimeHit = block.CurrentTime+0.0001;
%end

        function Update(block)
block.Dwork(1).Data=block.InputPort(1).Data;
```