CHALMERS

Computationally Efficient Model Predictive Direct Torque Control

Author: YASHAR ZEINALY
Examiner: Prof. BO EGARDT
Supervisor: Dr. TOBIAS GEYER

Automatic Cotnrol Group
Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2010

Acknowledgements

It is a pleasure to thank those who made this thesis possible. I owe my deepest gratitude to my supervisor Dr. Tobias Geyer for attracting me to New Zealand to work on such an interesting topic and for his efforts to make me feel welcome in Auckland. I appreciate his continuous support of my thesis, his patience in dealing with my mistakes and his immense knowledge. Most importantly, I would like to thank him for his kind advices that helped me to plan my future life.

Beside my supervisor, I would like to express my sincere gratitude to my examiner Prof. Bo Egardt who has made available his help and support in a number of ways. I appreciate his encouragement, enthusiasm and insightful comments.

I am indebted to Marie Iwanow in the student administration office who was so nice to me, and helped me in a number of ways with the administrative issues.

I thank my best friends Iman and Ramin for helping me out with 'LyX' issues and for the happy moments we had together.

Last but not the least, I would like to express my loving thanks to my mom, my dad, my sister Sahar and my brother Aidin for their loving support and encouragement throughout my life.

Yashar Zeinaly

November 2010, Göteborg

Abstract

Model predictive direct torque control (MPDTC) is a recent control scheme for three-phase ac electric drives based on direct torque control (DTC). Using a dynamic model of the drive, MPDTC predicts several future switching transitions, extends the output trajectories and chooses the inverter switch positions that minimize the switching frequency. The latest version of MPDTC allows for a long switching horizon, which is composed of multiple switching events (group of switching transitions) connected by extension segments. This enables us to achieve prediction horizons of 100 steps and more, which can significantly reduce the switching losses and the total harmonic distortions (THDs). The performance of MPDTC depends on the length of the switching horizon and the accuracy of the predictions. However, MPTDC schemes with long switching horizons and very accurate predictions are computationally demanding necessitating a very fast controller hardware. To reduce the associated computational burden, the following is proposed:

(i) Introducing methods for extending the output trajectories that yield fast yet accurate predictions.

(ii) Finding the promising switching transitions using the branch and bound technique and extending them, rather than enumerating all admissible switching transitions. This allows us to reduce the search space and thus the computation time.

In (i) we propose different extension methods and evaluate their performance in terms of the associated computational complexity, the accuracy of the predictions and closed-loop simulations. It is shown that these methods can achieve a good performance while reducing the computational burden of the MPDTC algorithm. In (ii) a branch and bound scheme for MPDTC is presented and its performance is compared with the original MPDTC algorithm. It is shown that using the proposed branch and bound algorithm, the number of iterations to obtain the optimal switch position can be reduced.

Keywords: power electronics, drive system, direct torque control, model predictive control, model predictive direct torque control.

Contents

Chapter 1

Introduction

Variable speed medium-voltage (MV) drives have found wide-spread applications in industry. They are used for wind turbines, pipeline pumps in the petrochemical industry, traction applications in the transportation industry, fans in the cement industry, pumps in water pumping stations and steel rolling mills in the metal industry [1]. In order to drive the machine at a variable speed, it must be fed with a variable frequency sinusoidal voltage. In power electronics, the inverter device synthesizes such a voltage by switching. As a consequence of switching, high-power inverters may generate a considerable amount of voltage and current harmonics. These harmonics cause additional power losses in the motor winding and the magnetic core, which also reduce the life-time of the motor. Distorted currents wave forms also give rise to torque pulsations, which are not desirable in most applications and may cause damage to the shaft, couplings and other mechanical components in the system. Increasing the switching frequency, reduces total harmonic distortions of the motor currents and the torque. However, the maximum switching frequency is limited due to constraints on the switching devices. Additionally, the switching loss in the inverter constitute a significant amount of total power loss in the MV drive. The switching loss minimization reduces the manufacturing and the maintenance costs of the MV drive. Therefore, in medium-voltage drives control, efforts are made to minimize the total harmonic distortions with limited switching frequency.

Direct torque control (DTC) is a common industrial scheme for controlling the speed and the torque of the medium voltage drives. DTC achieves a fast dynamic response with a simple controller structure. The control logic is stored on a pre-defined switching table. Model predictive direct torque control (MPDTC) inherits the core objectives of DTC and uses the elements of model predictive control to minimize the switching frequency of the inverter. In Chapter 2, the drive system and direct torque control are briefly presented. Then, the concept of model predictive direct torque control and its ingredients are briefly explained. It is shown in [2] that by using long *switching horizons*, the switching frequency and the total harmonic distortions can be reduced significantly.

The MPDTC algorithm enumerates the tree of all admissible switching sequences, and predicts the associated output trajectories to obtain the optimal switch position than minimizes the average switching frequency. However, for long switching horizons, the full enumeration is computationally very expensive due to the combinatorial explosion of the number of admissible switching sequences to be explored. Throughout this thesis, we were looking for methods to reduce the computational complexity of the MPDTC algorithm so that MPDTC with long switching horizons can be implemented in real-time using the currently available controller hardware. Our focus has been on (i) Developing methods for predicting the output trajectories that yield fast yet accurate predictions and (ii) Pruning the search tree using the branch and bound technique so that the number of switching sequences to be explored is reduced significantly.

In Chapter 3, *trajectory extension methods* are introduced which use the model-based predictions along with extrapolation and interpolation to obtain the predicted output trajectories. Additionally, using the simplified model of the drive, the state equations are solved and the resulting solutions are used to establish the analytical method for predicting the output trajectories.

In Chapter 4, the performance of the proposed trajectory extension methods is evaluated in the terms of their accuracy, computational complexity and their closed-loop performance.

In Chapter 5, the branch and bound technique tailored to the MPDTC problem [3] is introduced and its efficiency in reducing the computational burden is evaluated.

Chapter 2

Background

2.1 Variable Speed Medium-Voltage Drives

The development of medium-voltage drives started in mid-1950s when 4500-V gate turn off (GTO) thyristors became available on the market [1]. With rapid technology advancements in power semiconductor devices, insulated gate bipolar transistors (IGBT) and gate commutated thyristors (GCT) replaced the GTO in the inverter devices due to their superior switching characteristics, low switching losses, ease of gate control and snubberless operation. The medium-voltage drives have power ratings from 0.4 MW to 40 MW or more at voltage levels ranging between 2.3 kV to 13.8 kV. However, according to [1] the majority of installed medium-voltage drives have power ratings between 1 MW to 4 MW and voltage ratings between 3.3 kV to 6.6 kV. Fig. 2.1 shows the schematic of a medium-voltage drive.

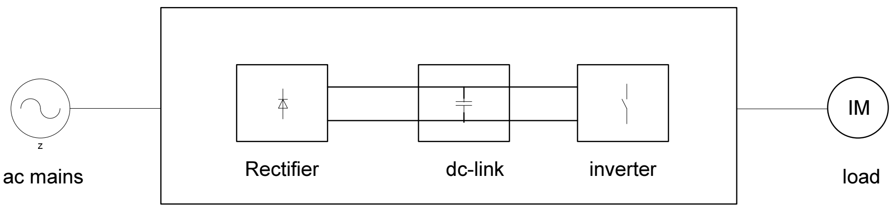
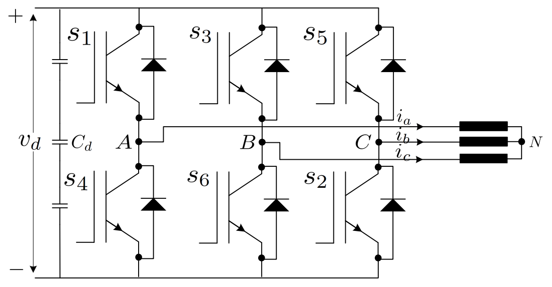
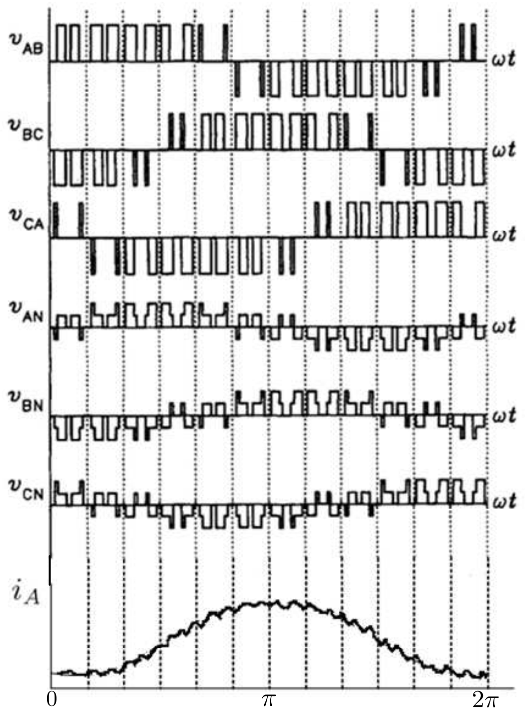


Figure 2.1: Simplified schematic of variable speed drive.

In these systems, DC/AC inverters are used to drive the induction motors as 3-phase variable frequency sources. The inverters are generally categorized into two main groups: i)Voltage source inverters (VSI) and ii)Current source inverters (CSI). Fig. 2.2a shows a simplified 2-level voltage source inverter, which is capable of producing two voltage levels 0 and v_d at its output terminals, where v_d is the dc-link voltage of the inverter. By manipulating the switch groups S_1-S_6, the inverter converts the DC voltage at its input to a 3-phase variable frequency sinusoidal voltage as shown in Fig. 2.2b.



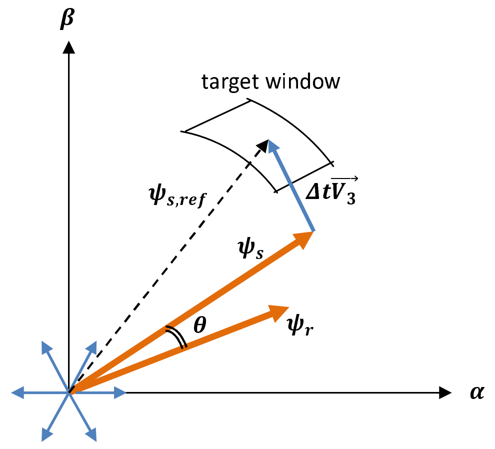


(a) **(b)**

Figure 2.2: 2-level inverter for high power applications. (a) Simplified representation (b) Output wave forms.

2.2 Direct Torque Control (DTC)

One of the methods that are used for controlling the induction machines' torque and speed is Direct Torque Control, which was introduced in 1985 and gradually became an industrial standard for induction motor drives [4]. DTC is based on the fast dynamic response of the stator flux vector when applying a voltage to the machine terminals. By choosing the right switch combinations on the inverter device, DTC drives the stator flux vector to the desired position such that the desired torque is achieved. The control logic is stored on a state of the art pre-designed switching table that is evaluated at every controller sampling-time $T_s = 25\mu s$. Depending on the application, the switching table may be designed to address a number of control objectives. In many applications the objectives are to keep the machine's electromagnetic torque and the stator flux magnitude within predefined hysteresis bounds. With 3-level neutral point clamped inverters [1], it is also necessary to balance the neutral point of the inverter. Fig. 2.3 introduces the principles of the DTC. For more information on DTC see [1,4,5].



$$T_e = K(\psi_{s\alpha}\psi_{r\beta} - \psi_{s\beta}\psi_{r\alpha}) = K|\psi_s||\psi_r|\sin\theta$$

$$|\psi_s| = \sqrt{\psi_{s\alpha}^2 + \psi_{s\beta}^2}$$

$$\frac{d\overrightarrow{\psi_s}}{dt} = \overrightarrow{V_s}(t) - R_s I_s$$

Figure 2.3: Principle of DTC. At each sampling-time DTC finds the right voltage vector to push the stator flux vector into the target window. The eight voltage vectors of a 2-level inverter are shown as well as the target window.

2.3 Model Predictive Direct Torque Control (MPDTC)

The idea underlying MPDTC [6, 7] is to replace the switching table in conventional DTC [4, 8] with a constrained optimal controller with a receding horizon policy [9–12]. The control objectives are to keep the machine's electromagnetic torque and the stator flux magnitude within predefined hysteresis bounds, which is referred to as the *feasible region*. With 3-level neutral point clamped inverters [1], it is also desired to balance the neutral point of the inverter. It is shown in [2, 6, 7, 13] that MPDTC significantly improves the performance of DTC by reducing the switching frequency of the inverter device.

2.3.1 Physical Model

The physical model is an essential ingredient of MPDTC, which enables us to predict the behavior of the drive.

Physical Model of the 3-Level Inverter

The simplified schematic of a 3-level voltage source inverter feeding an induction machine is shown in Fig. 2.4.

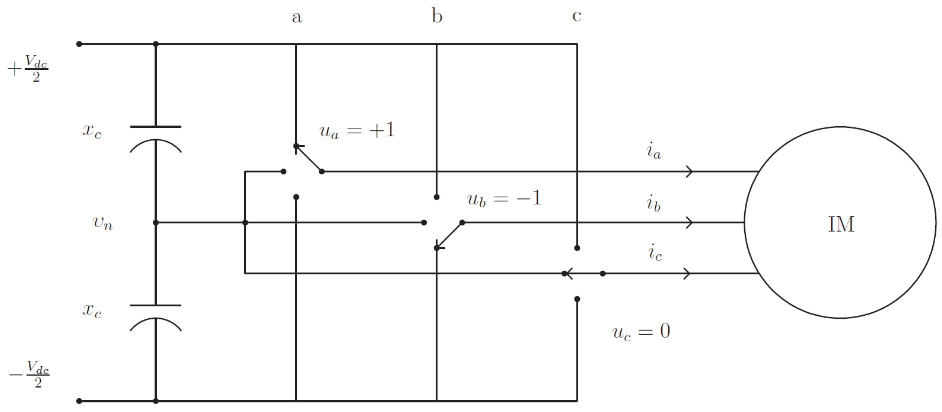


Figure 2.4: Equivalent representation of a 3-level inverter feeding an induction machine showing the switch positions. The picture belongs to T. Geyer [13]

The inverter can produce three different voltage levels, $-\frac{V_{dc}}{2}$, 0 and $\frac{V_{dc}}{2}$, at each of its three legs. The switch positions in each leg of the inverter are denoted by the integer values u_a, u_b, $u_c \in \{-1, 0, 1\}$, where the integers -1, 0, 1 yield the phase voltages $-\frac{V_{dc}}{2}$, 0 and $\frac{V_{dc}}{2}$, respectively. There are in total 27 switch positions of the form $u_{abc} \in \{-1, 0, 1\}^3$. The switch positions in 3-phase domain are transformed into the stator stationary reference frame using abc to $\alpha\beta0$ transformation matrix P.

$$P = \frac{2}{3}\begin{bmatrix} 1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{2.1}$$

The resulting 27 voltage vectors are shown in Fig. 2.5.

The voltage applied to the motor is obtained from

$$v_{\alpha\beta0} = \frac{V_{dc}}{2} P u_{abc} \tag{2.2}$$

The inverter dynamics include one state, which represents the inverter's neutral point potential v_n [6]

$$\frac{dv_n}{dt} = -\frac{1}{2x_C}\left((1-|u_a|)\, i_{sa} + (1-|u_b|)\, i_{sb} + (1-|u_c|)\, i_{sc}\right) \tag{2.3}$$

where i_{sa}, i_{sb} and i_{sc} are the stator phase currents and x_C is one of the two dc-link capacitors. Since for a three-phase symmetrical load $i_a + i_b + i_c = 0$, it can be shown that v_n can be equivalently represented by (2.4), where $i_{s,\alpha\beta0}$ is the stator current in the stator reference frame.

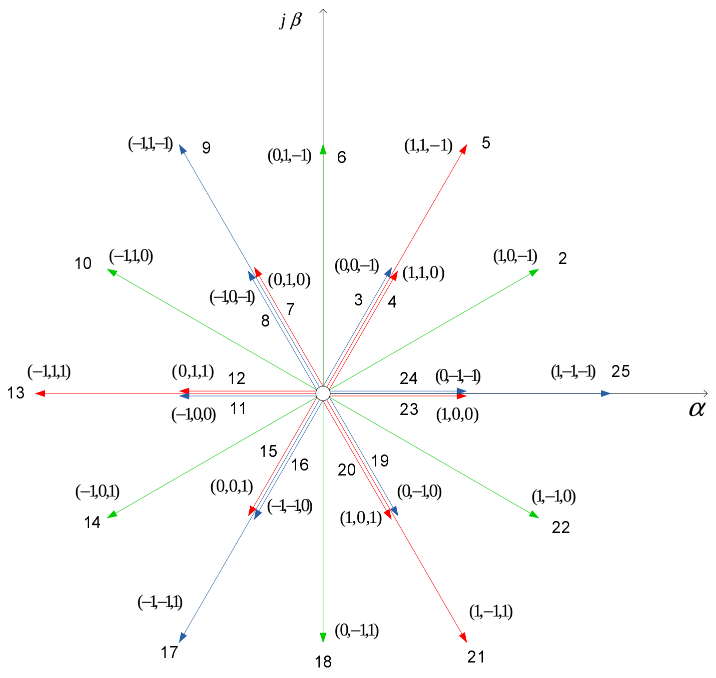


Figure 2.5: The voltage vectors of the 3-level inverter on $\alpha\beta$ plane with the corresponding 3-phase switch positions and the associated voltage vector numbers n. The three switch positions $(-1,-1,-1)$, $(0,0,0)$ and $(+1,+1,+1)$, with the corresponding voltage vector numbers $n = 26$, $n = 1$ and $n = 27$, respectively, are mapped into the origin of the $\alpha\beta$ plane

In (2.4), $|u_{abc}| = [|u_a|, |u_b|, |u_c|]^T$ is the component-wise absolute value of the inverter switch positions.

$$\frac{dv_n}{dt} = \frac{1}{2x_C} |u_{abc}|^T P^{-1} i_{s,\alpha\beta 0} \tag{2.4}$$

Physical Model of the Induction Machine

Assuming that the induction motor is three-phase symmetrical and its magnetic core is linear with a negligible core loss, the dynamics of induction machine are modeled in the stator $\alpha\beta 0$ reference frame[1] [1]. The state variables are the $\alpha-$and $\beta-$components of the stator and the rotor linkage fluxes $\psi_{s\alpha}$, $\psi_{s\beta}$, $\psi_{r\alpha}$ and $\psi_{r\beta}$, respectively, and the angular speed of the rotor (electrical) ω_r. The model parameters are the stator and the rotor winding resistances r_s and r_r, the stator, the rotor and the mutual reactances x_{ls}, x_{lr} and x_m. The inputs are v_α and v_β, which are the $\alpha-$and $\beta-$components of the voltage applied to the machine in $\alpha\beta 0$ reference frame. Please note that all the rotor variables and parameters are referred to the stator side. Throughout this report we assume that we are using the normalized quantities. The state equations are [6]

[1]This is called the stationary reference frame as well.

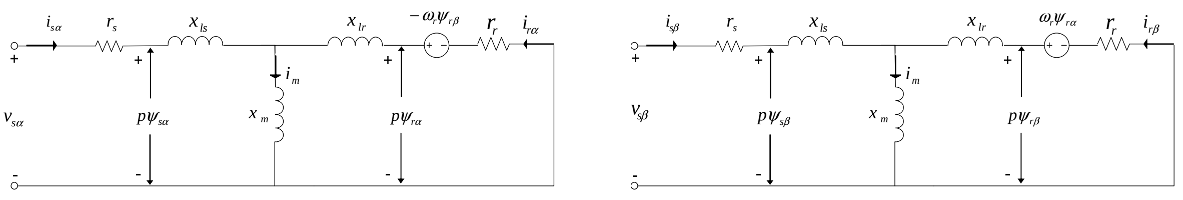


Figure 2.6: The equivalent circuits of the induction machine in α and β axis.

$$\begin{aligned}
\frac{d\psi_{s\alpha}}{dt} &= -r_s\frac{x_{rr}}{D}\psi_{s\alpha} + r_s\frac{x_m}{D}\psi_{r\alpha} + v_\alpha \\
\frac{d\psi_{s\beta}}{dt} &= -r_s\frac{x_{rr}}{D}\psi_{s\beta} + r_s\frac{x_m}{D}\psi_{r\beta} + v_\beta \\
\frac{d\psi_{r\alpha}}{dt} &= r_r\frac{x_m}{D}\psi_{s\alpha} - r_r\frac{x_{ss}}{D}\psi_{r\alpha} - \omega_r\psi_{r\beta} \\
\frac{d\psi_{r\alpha}}{dt} &= r_r\frac{x_m}{D}\psi_{s\beta} + \omega_r\psi_{r\alpha} - r_r\frac{x_{ss}}{D}\psi_{r\beta} \\
\frac{d\omega_r}{dt} &= \frac{1}{J}\left(T_e - T_l\right)
\end{aligned} \tag{2.5}$$

where x_{rr}, x_{ss} and D are defined as

$$\begin{aligned}
x_{ss} &= x_{ls} + x_m \\
x_{rr} &= x_{lr} + x_m \\
D &= x_{ss}x_{rr} - x_m^2
\end{aligned} \tag{2.6}$$

and the electromagnetic torque is defined as the cross product of the stator flux and rotor flux vectors.

$$T_e = \frac{x_m}{D}\left(\psi_{s\beta}\psi_{r\alpha} - \psi_{s\alpha}\psi_{r\beta}\right) \tag{2.7}$$

The magnitude of the stator flux is given by

$$\Psi_s = \sqrt{\psi_{s\beta}^2 + \psi_{s\alpha}^2} \tag{2.8}$$

The above equations constitute the model of the induction motor whose equivalent circuit is shown in Fig. 2.6.

The Simplified Model for Prediction

The time-constant of the rotor angular velocity, as captured by the last equation in (2.5), is larger than the prediction interval by several orders of magnitude. Therefore, we may omit the last equation in (2.5) assuming a constant ω_r during the prediction horizon.

The inverter is modeled with one state as in (2.4), which is described in terms of the stator currents in $\alpha\beta 0$ reference frame. The α-, β-components of $i_{s,\alpha\beta 0}$ are related to the stator and the rotor flux components according to (2.9) while the 0 component is always zero [6].

$$i_{s,\alpha\beta 0} = \frac{1}{D} \begin{bmatrix} x_{rr}\psi_{s\alpha} - x_m\psi_{r\alpha} & x_{rr}\psi_{s\beta} - x_m\psi_{r\beta} & 0 \end{bmatrix}^{\mathrm{T}} \tag{2.9}$$

The state vector of the drive is defined as

$$x = \begin{bmatrix} \psi_{s\alpha} & \psi_{s\beta} & \psi_{r\alpha} & \psi_{r\beta} & v_n \end{bmatrix}^{\mathrm{T}} \tag{2.10}$$

The input vector is the switch positions u_a, u_b, u_c

$$u = u_{abc} = \begin{bmatrix} u_a & u_b & u_c \end{bmatrix}^{\mathrm{T}} \tag{2.11}$$

The electromagnetic torque, the stator flux magnitude and the neutral point potential of the inverter constitute the output vector

$$y = \begin{bmatrix} T_e & \Psi_s & v_n \end{bmatrix}^{\mathrm{T}} \tag{2.12}$$

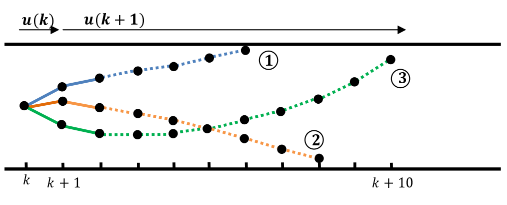
The last step for obtaining the discrete-time prediction model is to combine the simplified model equations (2.5), (2.7) and (2.8) with the inverter equations (2.4) and (2.9), and to use forward Euler method with sampling-time $T_s = 25\mu s$ to discretize the equations. For more details see [6, 13].

2.3.2 Switching Horizon

At time-step k, given the last control input $u(k-1)$ and current measurements, the MPDTC algorithm explores the tree of all admissible [2] *switching sequences* to find the *candidate sequences* $U(k) = \begin{bmatrix} u(k), & ..., & u(k+N_p-1) \end{bmatrix}$. Traveling through the tree of admissible switching sequences is controlled by the *switching horizon*, which is composed of *switching events* 'S' and *extension events.* During the 'S' event, new nodes are added to the tree by enumerating all admissible switching transitions and predicting the associated outputs (i.e. the electromagnetic torque, the stator flux magnitude and the inverter neutral point potential) using the nonlinear discrete-time model of the drive sampled at the sampling interval $T_s = 25\mu s$. During the 'E' event, the predicted output trajectories are extended[3] producing the extended nodes. At the end of the enumeration of the tree, the candidate sequences are the sequences whose associated

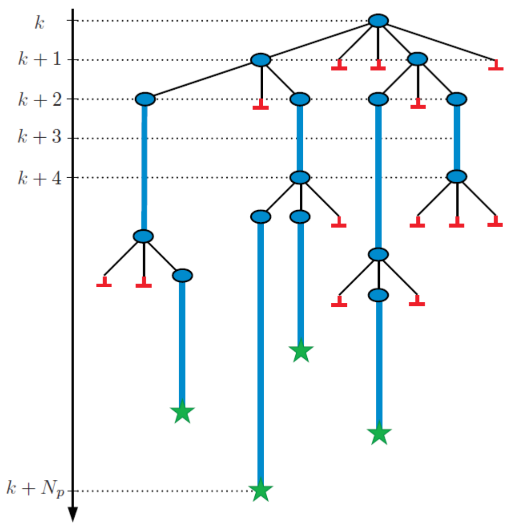
[2] Not all switching transitions are possible due to the physical constraints on the inverter.

[3] By 'extension' we mean to freeze the input and use one on the methods in Chapter 3 to obtain the future samples of the outputs.



	$u(k-1)$	$u(k)$	$u(k+1)$
①	[0, 0, 0]	[1, 0, 0]	[0,0,0]
②	[0, 0, 0]	[−1, 0, 0]	[0, −1, 0]
③	[0, 0, 0]	[−1, 0, −1]	[0, 1, 0]

(a) An example of the switching horizon 'SSE'. Three trajectories associated with three candidate sequences are shown within their bounds. The 'S' events happen at time-instants k and $k+1$ followed by an 'E' event. For each candidate sequence, the length of the prediction horizon is the number of time-steps for which the trajectories remain within the bounds.



(b) An example of the search tree for the switching horizon 'SSESE'. The ovals (blue), the stars (green) and the inverted (red) Ts denote the incomplete, candidate and non-candidate switching sequences, respectively. The 'S' events are shown by thin (black) lines while the 'E' events are shown by thick (blue) lines. This picture is taken from [3].

Figure 2.7: An example showing the switching horizon and the associated search tree

output trajectories are at every time-step either *feasible*, or *pointing in the proper direction*[4]. For each candidate sequence, the length of the prediction horizon N_p is given by the time duration for which the extended predicted outputs remain within their bounds. Fig. 2.7b shows the search tree for the switching horizon 'SSESE'. Fig. 2.7a provides an example of the switching horizon 'SSE'.

2.3.3 Minimizing the Cost Function

For each candidate sequence, the average switching frequency is calculated as the total number of switching transitions in the switching sequence divided by the prediction horizon length according to

$$c_i = \frac{1}{N_{p,i}} \sum_{l=k}^{k+N_p-1} \|u_i(l) - u_i(l-1)\|_1 \tag{2.13}$$

[4]A trajectory points in the proper direction if the magnitude of the bound violation is strictly decreasing over time.

The optimal switching sequence is obtained by minimizing the average switching frequency over all candidate switching sequences. According to the receding horizon policy, only the first input of the optimal sequence, $u(k)$, is applied to the drive and this procedure is repeated at the next controller sampling time $(k+1)T_s$ with new data. Minimizing the average switching frequency is an indirect way of minimizing the switching losses. Alternatively, the cost function may penalize the switching losses directly. The interested reader is referred to [2] for details.

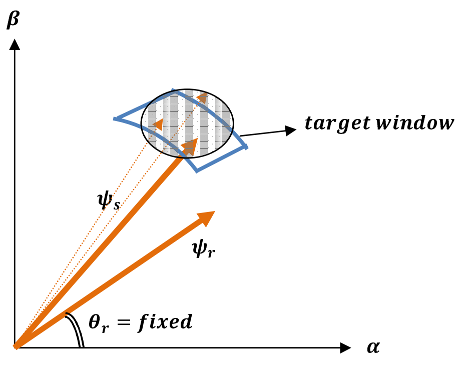
One important characteristic of MPDTC algorithm is that the cost function is evaluated over a variable prediction horizon N_p. Table 2.1 shows the optimization step for the example trajectories of Fig. 2.7a.

	$u(k-1)$	$u(k)$	$u(k+1)$	$\sum \lvert u \rvert$	N_p	c
1	$[0\,0\,0]$	$[1\,0\,0]$	$[0\,0\,0]$	2	6	0.33
2	$[0\,0\,0]$	$[-1\,0\,0]$	$[0\,-1\,0]$	3	8	0.375
3	$[0\,0\,0]$	$[-1\,0$ -1]	$[0\,1\,0]$	5	10	0.5

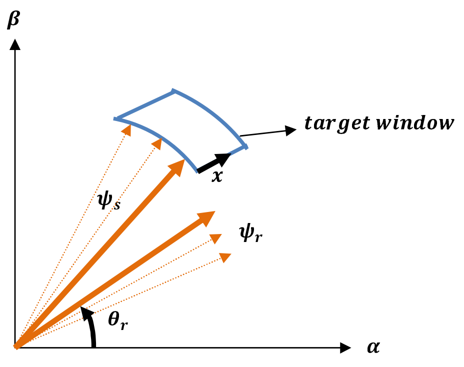
Table 2.1: An example of the optimization step for the switching horizon 'SSE'. The optimal input is the first step in switching sequence 1, i.e. $u(k) = [\;1\;\;0\;\;0\;]$.

Fig. 2.9 shows maps of the optimal cost and the associated switching input for the switching horizon 'SE' with $u(k-1) = [\;0\;\;0\;\;0\;]^{\mathrm{T}}$ when the switching frequency is penalized. The maps in Fig. 2.9 are calculated over a grid of $\psi_{s\alpha}$ and $\psi_{s\beta}$ including the target window for a fixed rotor angle as shown in Fig. 2.8a. It is also interesting to observe the behavior of the cost function along boundaries of the target window as the controller needs to make decisions when the outputs are about to hit their bounds.We can observe that when the output variables are within their bounds, the cost to go is zero as the controller does not need to switch. Moreover, when the upper bound of the output variables are violated, the MPDTC chooses $u(k) = [\;0\;\;0\;\;0\;]^{\mathrm{T}} = u(k-1)$ as, according to the model, the zero voltage vector drives the outputs to the feasible region with zero switching effort. The regions with non-zero cost are those where the lower bound on one of the output variables is violated requiring expensive switching by the controller to push the stator flux vector into the target window. Fig. 2.10 depicts maps of the optimal cost and the associated switching input for switching horizon 'SE' when the switching losses are minimized. The maps are obtained over a grid of x and θ_r as shown in Fig. 2.8b, where x is the distance traveled on an edge of the boundary and θ_r is the rotor angle.

In both cases, the 'cost to go' maps can be divided into regions. Similar regions are noticed on the optimal input maps, which shows the controller makes consistent decisions over each region.

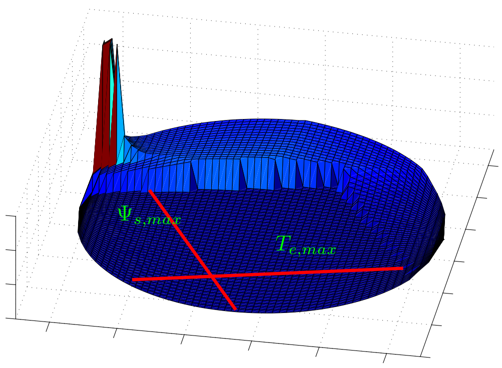


(a) The region associated with maps in Fig 2.9. The grid over which the maps are plotted (shaded oval) is shown along with the target window.

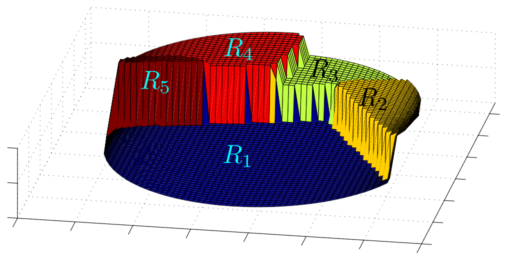


(b) The region associated with maps in Fig. 2.10. x and θ_r are shown along with the target window .

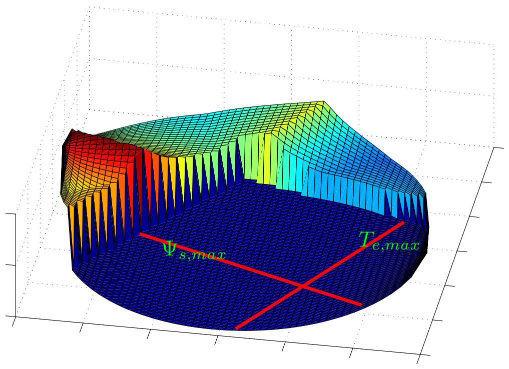
Figure 2.8: The regions used for obtaining the 'cost to go' maps



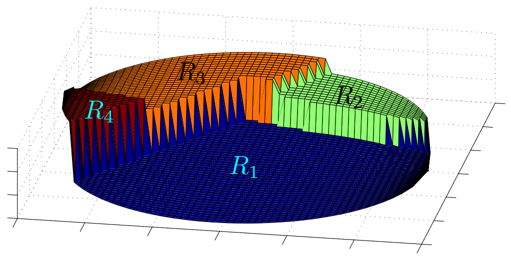
(a) The cost to go map for $\theta_r = 0.24$ rad along with the bounds on the torque and the stator flux magnitude.



(b) The optimal input map for $\theta_r = 0.24$ rad along with the switching regions. $R_1 : n = 1$, $R_2 : n = 7$, $R_3 : n = 6$, $R_4 : n = 9$, $R_5 : n = 10$

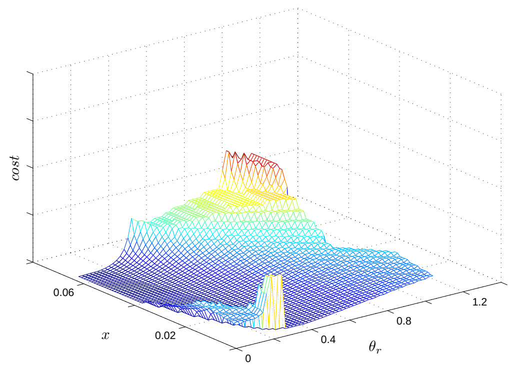


(c) The cost to go map for $\theta_r = 0.96$ rad along with the bounds on the torque and the stator flux magnitude.

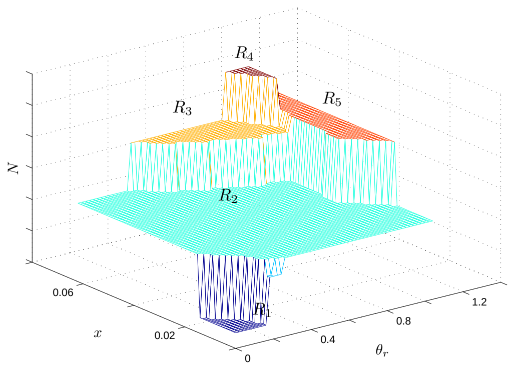


(d) The optimal input map for $\theta_r = 0.96$ rad along with the switching regions. $R_1 : n = 1$, $R_2 : n = 7$, $R_3 : n = 10$, $R_4 : n = 13$

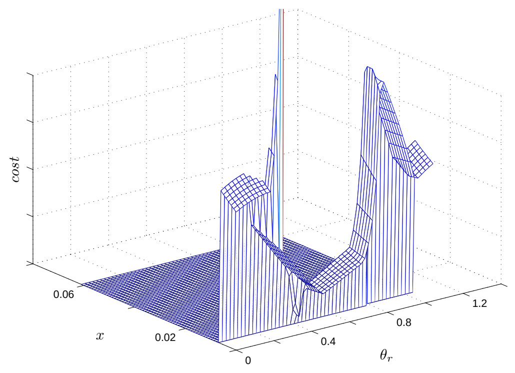
Figure 2.9: The 'cost to go' and optimal input maps when minimizing the switching frequency using the switching horizon 'SE' at $T_e = 1$pu and $\omega_e = 0.8$pu.



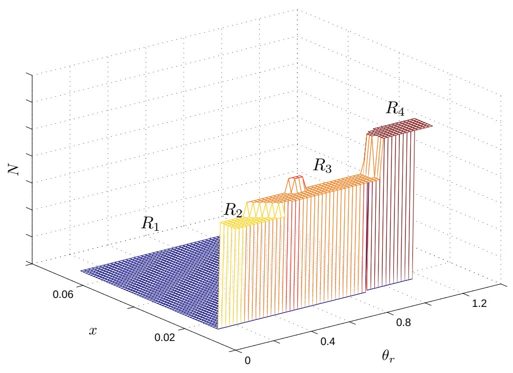
(a) The cost to go map along $T_{e,min}$ at $T_e = 1$pu and $\omega_e = 0.8$pu .



(b) The optimal input map along $T_{e,min}$ at $T_e = 1$pu and $\omega_e = 0.8$pu. $R_1 : n = 3$, $R_2 : n = 7$, $R_3 : n = 10$, $R_4 : n = 13$, $R_5 : n = 11$



(c) The cost to go map along $\Psi_{s,max}$ at $T_e = 1$pu and $\omega_e = 0.5$pu.



(d) The optimal input map along $\Psi_{s,max}$ at $T_e = 1$pu and $\omega_e = 0.5$pu. $R_1 : n = 1$, $R_2 : n = 9$, $R_3 : n = 10$, $R_4 : n = 13$

Figure 2.10: The 'cost to go' and optimal input maps when minimizing the switching frequency using the switching horizon 'SE'.

Chapter 3

Trajectory Extension Methods

In deriving the formula for the different extension methods (refer to Sect. 2.3.2), for the sake of simplicity, we limit ourselves to the switching horizon 'SE', which means a switching action at time k is followed by an extension event. It is straightforward to generalize the results to an arbitrary switching horizon. Before proceeding, some notations have to be introduced.

- $\boldsymbol{x} = [\ \psi_{s\alpha} \quad \psi_{s\beta} \quad \psi_{r\alpha} \quad \psi_{r\beta}\ ]^{\mathrm{T}}$, $\boldsymbol{y} = [\ T_e \quad \Psi_s \quad v_n\ ]^{\mathrm{T}}$ and $\boldsymbol{u} = [\ u_a \quad u_b \quad u_c\ ]^{\mathrm{T}}$ denote the state, output and input vectors, respectively. The i-th element in the state and output vectors are denoted by x_i and y_i, respectively.

- $\boldsymbol{y}_{min} = [\ T_{e,min} \quad \Psi_{s,min} \quad v_{n,min}\ ]^{\mathrm{T}}$ and $\boldsymbol{y}_{max} = [\ T_{e,max} \quad \Psi_{s,max} \quad v_{n,max}\ ]^{\mathrm{T}}$ are the lower and upper bounds on the output variables. The lower and upper bounds on the i-th output are denoted by $y_{i,max}$ and $y_{i,min}$, respectively.

- $f(.)$ and $g(.)$ are the drive's state and output equations discretized at the sampling time T_s, respectively.

- $f_d(.)$ and $g_d(.)$ are the drive's state and output equations discretized at the coarse sampling-time dT_s, respectively.

3.1 Open-loop Simulation (OL)

The most accurate way of extending the output trajectories is to use the internal model of the drive in an open-loop simulation [2]. Even though this requires excessive computational power making this approach impractical, this method is regarded as a benchmark to evaluate the performance of other extension methods against.

The switch position at time-step k, $\boldsymbol{u}(k)$, is applied to the nonlinear discrete-time model of the drive to compute the outputs from time-instant k on. The trajectories are extended until one of the outputs hits a bound, which determines the length of the prediction horizon N_p. The extended trajectories are described from time-step k to $k + N_p - 1$ according to

$$\begin{aligned} \boldsymbol{x}(k+n) &= f\left(\boldsymbol{x}(k+n-1), \boldsymbol{u}(k)\right), \quad 0 \le n \le N_p - 1 \\ \boldsymbol{y}(k+n) &= g\left(\boldsymbol{x}(k+n)\right), \quad 0 \le n \le N_p - 1 \end{aligned} \tag{3.1}$$

For an example, see Fig. 3.1a. More details about this method can be found in [2, 13].

3.2 Linear Extrapolation (LE)

The switch position from the last 'S' event, $\boldsymbol{u}(k)$, is applied to the nonlinear discrete-time model of the drive once to obtain the output sample at time-instant $k + 1$ according to

$$\begin{aligned} \boldsymbol{x}(k+1) &= f\left(\boldsymbol{x}(k), \boldsymbol{u}(k)\right) \\ \boldsymbol{y}(k+1) &= g\left(\boldsymbol{x}(k+1)\right) \end{aligned} \tag{3.2}$$

The output trajectories are then extrapolated from time-instant k on linearly using the samples at time-instances k and $k + 1$ until an output variable hits a bound. The extended trajectories are described from time-step k to $k + N_p - 1$ according to

$$\boldsymbol{y}(k+n) = \left(\boldsymbol{y}(k+1) - \boldsymbol{y}(k)\right) n + \boldsymbol{y}(k), \quad 0 \le n \le N_p - 1 \tag{3.3}$$

The points where the output bounds are crossed are calculated analytically according to (3.4).

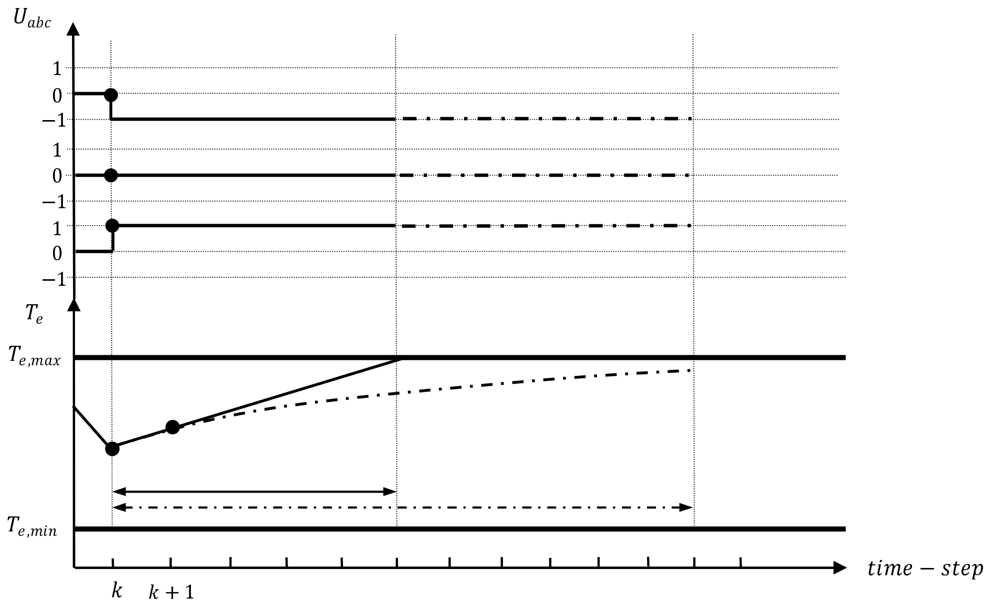
$$n_i = \min\left(\max\left(0, \frac{y_{i,max} - y_i\,(k)}{y_i\,(k+1) - y_i\,(k)}\right), \max\left(0, \frac{y_{i.min} - y_i\,(k)}{y_i\,(k+1) - y_i\,(k)}\right)\right) \tag{3.4}$$

The length of the prediction horizon, N_p, is given by

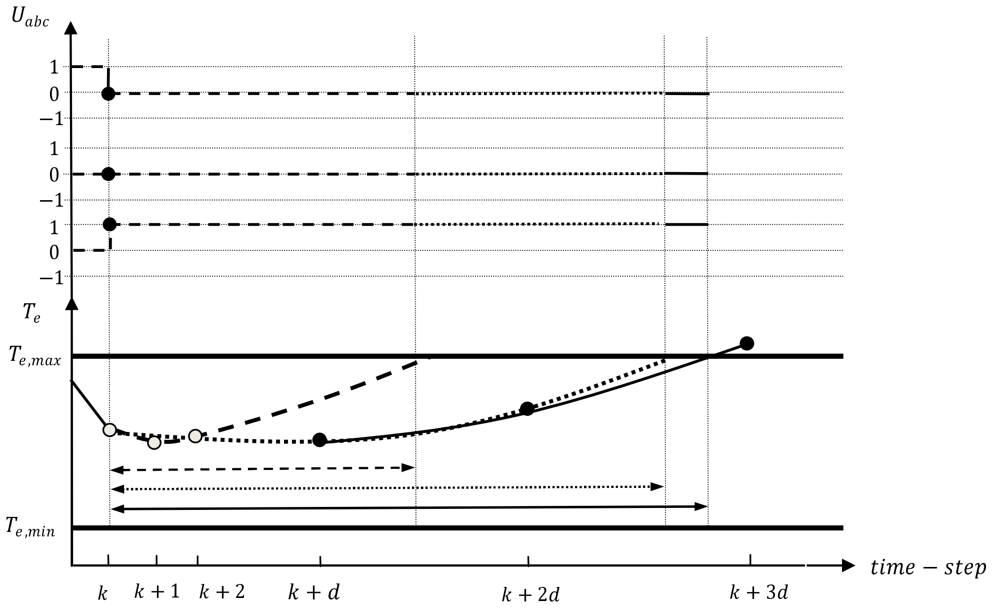
$$N_p = \min_i \{n_i\} \tag{3.5}$$

See Fig. 3.1a for an example.

Linear extrapolation is proposed in [6]. It is worth mentioning that MPDTC with the switching horizon 'SE' has been successfully implemented and tested on ABB's ACS6000 medium-voltage drive [7].

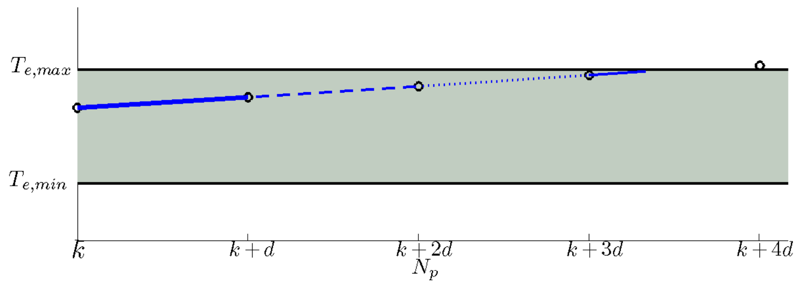


(a)

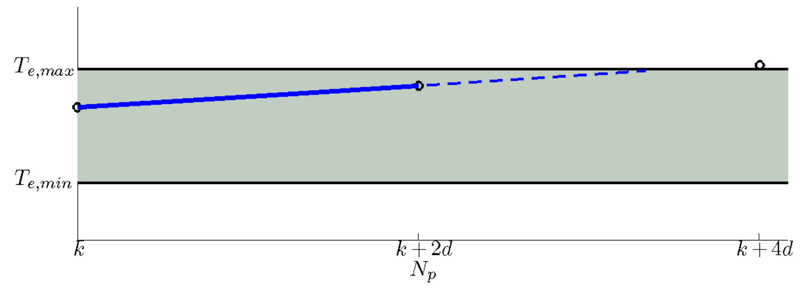


(b)

Figure 3.1: Examples of predicted torque trajectories between their bounds and the associated switch positions for the switching horizon ’SE’. The ’S’ event happens at time-step k. (a) Comparison between the OL (dash-dotted line) and LE (solid line) methods and the associated prediction horizon lengths (bi-directional arrows). The points (bold dots) used for the ’LE’ method are shown as well. (b) Comparison between the ’QE’ (dashed), ’QI’ (dotted) and ’QII’ (solid) methods and their prediction horizon lengths. The ’QE’ method uses the points at time-instants k, $k+1$ and $k+2$. The ’QI’ method uses the points at time-instants k, $k+d$ and $k+2d$. The ’QII’ method uses the additional point at time-step $k+3d$ to refine the predicted trajectory as the length of prediction horizon increases.



(a)



(b)

Figure 3.2: Examples of predicted torque trajectories using (a) QII and (b) QII2 extension methods for the switching horizon ’SE’. The ’S’ event happens at time-step k. The segments are shown with different lines styles. For this particular example, the trajectory extended using QII is composed of four segments whereas the trajectory extended using QII2 is composed of two segments.

3.3 Quadratic Extrapolation (QE)

Since the machine's exact output trajectories are of quadratic and trigonometric types, they can be better approximated by quadratic curves rather than lines. The switch position obtained at the last 'S' event is applied twice to the drive model to obtain the outputs at time-instants $k+1$ and $k+2$ according to

$$\begin{aligned} \boldsymbol{x}(k+n) &= f\left(\boldsymbol{x}(k+n-1), \boldsymbol{u}(k)\right), \quad n = 1,2 \\ \boldsymbol{y}(k+n) &= g\left(\boldsymbol{x}(k+n)\right), \quad n = 1,2 \end{aligned} \tag{3.6}$$

Using the output samples at time instants k, $k+1$, $k+2$ the output trajectories are extrapolated quadratically from time-step k on until the first of the three output variables leaves the feasible region. The extended trajectories are described by second order polynomials from time-step k until $k+N_p-1$ according to

$$\boldsymbol{y}(k+n) = \boldsymbol{a}n^2 + \boldsymbol{b}n + \boldsymbol{c}, \quad 0 \le n \le N_p - 1 \tag{3.7}$$

with the quadratic coefficient vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{c}$ given by

$$\begin{bmatrix} \boldsymbol{a}^{\mathrm{T}} \\ \boldsymbol{b}^{\mathrm{T}} \\ \boldsymbol{c}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\boldsymbol{y}(k) - \boldsymbol{y}(k+1) + \frac{1}{2}\boldsymbol{y}(k+2) \\ -\frac{3}{2}\boldsymbol{y}(k) + 2\boldsymbol{y}(k+1) - \frac{1}{2}\boldsymbol{y}(k+2) \\ \boldsymbol{y}(k) \end{bmatrix}^{\mathrm{T}} \tag{3.8}$$

The cross over points of the output bounds are given by (3.9) from which the length of the prediction horizon, N_p, is obtained according to (3.10).

$$\begin{aligned} n_i = \min & \left[\max\left(0, \frac{-b_i \pm \sqrt{b_i^2 - 4a_i\left(c_i - y_{i,max}\right)}}{2a_i}\right) \right. \\ , \quad & \left. \max\left(0, \frac{-b_i \pm \sqrt{b_i^2 - 4a_i\left(c_i - y_{i,min}\right)}}{2a_i}\right) \right] \end{aligned} \tag{3.9}$$

$$N_p = \min_i\{n_i\} \tag{3.10}$$

Fig. 3.1b provides an example of how the trajectories are extended.

3.4 Prediction with Quadratic Interpolation (QI)

Predicting the output samples far ahead in the future and then interpolating between them yields extended trajectories with better accuracy. The algorithm uses the switching input derived at the last 'S' event to compute the output samples at time-instants $k+d$ and $k+2d$ according to

$$\begin{aligned} \boldsymbol{x}(k+nd) &= f_d\left(\boldsymbol{x}(k+(n-1)d), \boldsymbol{u}(k)\right) \quad n=1,2 \\ \boldsymbol{y}(k+nd) &= g_d\left(\boldsymbol{x}(k+nd)\right) \quad n=1,2 \end{aligned} \tag{3.11}$$

Interpolating between outputs at time-instants k, $k+d$ and $k+2d$ quadratically yields the extended trajectories. The trajectories are extended until the first out of the three output variables becomes infeasible. The extended trajectories are

$$\boldsymbol{y}(k+n) = \boldsymbol{a}n^2 + \boldsymbol{b}n + \boldsymbol{c}, \quad 0 \le n \le N_p - 1 \tag{3.12}$$

with

$$\begin{aligned} \begin{bmatrix} \boldsymbol{a}^{\mathrm{T}} \\ \boldsymbol{b}^{\mathrm{T}} \end{bmatrix} &= \begin{bmatrix} d^2 & d \\ 4d^2 & 2d \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{y}(k+d) - \boldsymbol{y}(k) \\ \boldsymbol{y}(k+2d) - \boldsymbol{y}(k) \end{bmatrix}^{\mathrm{T}} \\ \boldsymbol{c}^{\mathrm{T}} &= \boldsymbol{y}(k)^{\mathrm{T}} \end{aligned} \tag{3.13}$$

$$\begin{bmatrix} \boldsymbol{a}^{\mathrm{T}} \\ \boldsymbol{b}^{\mathrm{T}} \\ \boldsymbol{c}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2d^2}\boldsymbol{y}(k) - \frac{1}{d^2}\boldsymbol{y}(k+d) + \frac{1}{2d^2}\boldsymbol{y}(k+2d) \\ -\frac{3}{2d}\boldsymbol{y}(k) + \frac{2}{d}\boldsymbol{y}(k+d) - \frac{1}{2d}\boldsymbol{y}(k+2d) \\ \boldsymbol{y}(k) \end{bmatrix}^{\mathrm{T}} \tag{3.14}$$

We can solve for N_p analytically, which is given by (3.9) and (3.10). The choice of the parameter d is crucial for this algorithm to yield good results. It should be chosen such that $2d$ covers the prediction horizon with a high probability. See Fig. 3.1b for more details.

3.5 Iterative Prediction with Quadratic Interpolation (QII)

Particularly for long prediction horizons, the extrapolation (interpolation) based on few predicted output samples can give rise to large errors with respect to the open-loop simulation method. If required, the QII method uses additional predicted output samples to refine the extrapolation as the length of the extended trajectories increases. As for prediction with quadratic interpolation, a quadratic curve is interpolated between the output samples at time-instants k, $k+d$ and $k+2d$. If no output bound is violated in the $[k\;k+2d]$ interval, everything is shifted by d time-steps and the outputs at time instant $k+3d$ are predicted using (3.11). A new quadratic curve is interpolated between the output samples at time-instants $k+d$, $k+2d$ and $k+3d$. This procedure is continued until a bound violation is detected.

The QII method yields extended trajectories that are composed of m segments. Each segment is characterized by a second order polynomial. The extended trajectories from time-step k to $k+N_p-1$ are

$$\mathbf{y}(k+n) = \begin{cases} \boldsymbol{a}^{\mathbf{1}}n^2 + \boldsymbol{b}^{\mathbf{1}}n + \mathbf{c}^{\mathbf{1}} & 0 \leqslant n \leqslant d \\ \boldsymbol{a}^{\mathbf{2}}(n-d)^2 + \boldsymbol{b}^{\mathbf{2}}(n-d) + \mathbf{c}^{\mathbf{2}} & d < n \leqslant 2d \\ \vdots & \vdots \\ \boldsymbol{a}^{\boldsymbol{m}}(n-(m-1)d)^2 + \boldsymbol{b}^{\boldsymbol{m}}(n-(m-1)d) + \boldsymbol{c}^{\boldsymbol{m}} & (m-1)d < n \leqslant md \end{cases} \tag{3.15}$$

with

$$\begin{aligned} \begin{bmatrix} \boldsymbol{a}^{\boldsymbol{i}\mathrm{T}} \\ \boldsymbol{b}^{\boldsymbol{i}\mathrm{T}} \end{bmatrix} &= \begin{bmatrix} d^2 & d \\ 4d^2 & 2d \end{bmatrix}^{-1} \begin{bmatrix} \boldsymbol{y}(k+id) - \boldsymbol{y}(k+(i-1)d) \\ \boldsymbol{y}(k+(i+1)d) - \boldsymbol{y}(k+(i-1)d) \end{bmatrix}^{\mathrm{T}} \\ \boldsymbol{c}^{\boldsymbol{i}\mathrm{T}} &= \boldsymbol{y}(k+(i-1)d)^{\mathrm{T}} \end{aligned} \tag{3.16}$$

$$\begin{bmatrix} \boldsymbol{a}^{\boldsymbol{i}\mathrm{T}} \\ \boldsymbol{b}^{\boldsymbol{i}\mathrm{T}} \\ \boldsymbol{c}^{\boldsymbol{i}\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2d^2}\boldsymbol{y}(k+(i-1)d) - \frac{1}{d^2}\boldsymbol{y}(k+id) + \frac{1}{2d^2}\boldsymbol{y}(k+(i+1)d) \\ -\frac{3}{2d}\boldsymbol{y}(k+(i-1)d) + \frac{2}{d}\boldsymbol{y}(k+id) - \frac{1}{2d}\boldsymbol{y}(k+(i+1)d) \\ \boldsymbol{y}(k+(i-1)d) \end{bmatrix}^{\mathrm{T}} \tag{3.17}$$

In contrast to the LE, QE and QI methods, the solution of N_p cannot be found analytically as m is not know a priori and depends on N_p according to

$$m = \begin{cases} \left\lceil \frac{N_p}{d} \right\rceil - 1 & N_p \geq 2d \\ 1 & N_p < 2d \end{cases} \tag{3.18}$$

3.6 Iterative Prediction with Quadratic Interpolation Non-overlapping (QII2)

This method, similarly to QII, interpolates between the output samples predicted using the model of the drive sampled at the coarse sampling time dT_s. The only difference is that if no bound violation is detected in the $[k\, k+2d]$ interval, the algorithm computes the outputs at time-instants $k+3d$ and $k+4d$ and the new quadratic curve will be interpolated using the output samples at $k+2d$, $k+3d$ and $k+4d$. This procedure is continued until one of the outputs leaves the feasible region. The extended trajectories consist of m segments of second order polynomials and are given by (3.19)

$$\mathbf{y}(k+n) = \begin{cases} \boldsymbol{a^1} n^2 + \boldsymbol{b^1} n + \boldsymbol{c^1} & 0 \leqslant n \leqslant 2d \\ \boldsymbol{a^2}(n-2d)^2 + \boldsymbol{b^2}(n-2d) + \boldsymbol{c^2} & 2d < n \leqslant 4d \\ \vdots & \vdots \\ \boldsymbol{a^m}(n-(m-1)2d)^2 + \boldsymbol{b^m}(n-(m-1)2d) + \boldsymbol{c^m} & (m-1)2d < n \leqslant 2md \end{cases} \tag{3.19}$$

where a_i, b_i and c_i are

$$\begin{bmatrix} \boldsymbol{a^i}^{\mathrm{T}} \\ \\ \boldsymbol{b^i}^{\mathrm{T}} \\ \\ \boldsymbol{c^i}^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2d^2}\boldsymbol{y}\left(k+(i-1)\,2d\right) - \frac{1}{d^2}\boldsymbol{y}\left(k+(2i-1)\,d\right) + \frac{1}{2d^2}\boldsymbol{y}\left(k+2id\right) \\ \\ -\frac{3}{2d}\boldsymbol{y}\left(k+(i-1)\,2d\right) + \frac{2}{d}\boldsymbol{y}\left(k+(2i-1)\,d\right) - \frac{1}{2d}\boldsymbol{y}\left(k+2id\right) \\ \\ \boldsymbol{y}\left(k+(i-1)\,2d\right) \end{bmatrix}^{\mathrm{T}}$$

As it was the case with QII method, There is no analytical solution for N_p. The number of segments in the extended trajectory, m, is a function of N_p according to (3.20)

$$m = \left\lceil \frac{N_p}{2d} \right\rceil \tag{3.20}$$

3.7 Analytical Approach (ANL)

The methods discussed so far are based on (iterative) extrapolation (interpolation). In this section we will introduce an analytical method to extend the output trajectories. We aim at expressing the outputs (i.e. the torque, the stator flux and the neutral point's potential) by polynomials of degree three or less, which enables us to solve for N_p analytically[1].

As the drive's model is represented by the linear set of state equations (2.5) and (2.3)[2], one can solve the set of linear differential equations to obtain the exact expressions describing the drive's states. For prediction purpose, these expressions are simplified by taking advantage of some important properties of the drive's model and using Taylor's series to approximate them about the nominal length of the prediction horizon. The output polynomials are obtained by using these expressions in the nonlinear output equations (2.8) and (2.7). Finally, the output polynomials are used to find N_p.

3.7.1 Analytic Solutions of the State Equations

Machine States The machine's state equations in (2.5) can be rewritten in matrix form

$$\dot{\boldsymbol{x}}(t) = A\boldsymbol{x}(t) + B\boldsymbol{u}(t) \tag{3.21}$$

with

$$\boldsymbol{x}(t) = \begin{bmatrix} \psi_{s\alpha} & \psi_{s\beta} & \psi_{r\alpha} & \psi_{r\beta} \end{bmatrix}^{\mathrm{T}} \tag{3.22}$$

and

$$\boldsymbol{u} = \begin{bmatrix} v_\alpha & v_\beta \end{bmatrix}^{\mathrm{T}} \tag{3.23}$$

The matrices A and B are

$$A = \begin{bmatrix} -a & 0 & b & 0 \\ 0 & -a & 0 & b \\ c & 0 & -f & -\omega_r \\ 0 & c & \omega_r & -f \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.24}$$

with $a = r_s \frac{x_{rr}}{D}$, $b = r_s \frac{x_m}{D}$, $c = r_r \frac{x_m}{D}$ and $f = r_r \frac{x_{ss}}{D}$, where r_s, r_r, x_{rr}, x_m, x_{ss} and D are the motor parameters in Chapter 2.

[1]Polynomials of higher degree or other nonlinear functions will necessitate the use of a numerical root finding algorithm.

[2]Note that the equation describing the neutral point voltage is not linear in its inputs. However, having the numerical values for the switch positions, the equation can be treated as linear.

The solution of the matrix differential equation (3.21) with the initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$ is [14]

$$\boldsymbol{x}(t) = \mathrm{e}^{At}\boldsymbol{x}_0 + \int_0^t \mathrm{e}^{A(t-\tau)}B\boldsymbol{u}(\tau)d\tau \tag{3.25}$$

with

$$\mathrm{e}^{At} = L^{-1}\left\{(sI - A)^{-1}\right\} \tag{3.26}$$

where I, s and $L^{-1}(.)$ are the identity matrix of appropriate size, the Laplace variable and the inverse Laplace operator, respectively.

$$sI - A = \left[\begin{array}{c|c} \overbrace{\begin{matrix} s+a & 0 \\ 0 & s+a \end{matrix}}^{W} & \overbrace{\begin{matrix} -b & 0 \\ 0 & -b \end{matrix}}^{X} \\ \hline \underbrace{\begin{matrix} -c & 0 \\ 0 & -c \end{matrix}}_{Y} & \underbrace{\begin{matrix} s+f & \omega_r \\ -\omega_r & s+f \end{matrix}}_{Z} \end{array}\right] \tag{3.27}$$

Defining the matrix blocks W, X, Y and Z as in (3.27) and using Sherman-Morrison inversion formula [15], we have

$$(sI - A)^{-1} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}^{-1} \tag{3.28}$$

$$= \left[\begin{array}{c|c} W^{-1} + W^{-1}X\left(Z - YW^{-1}X\right)^{-1}YW^{-1} & -W^{-1}X\left(Z - YW^{-1}X\right)^{-1} \\ \hline -(Z - YW^{-1}X)YW^{-1} & \left(Z - YW^{-1}X\right)^{-1} \end{array}\right] \tag{3.29}$$

When calculating the block matrices in (3.28), we approximate the expressions based on the fact that the prediction model is to be used for medium to high frequency range (i.e. $s >> 0$) so eliminating the terms that are too small within this frequency range[3]. Please see Appendix A for the details.

[3]The prediction model is not intended to be used for steady state

$$(sI-A)^{-1} \simeq \frac{1}{(s+f)^2+\omega_r^2}\begin{bmatrix} \frac{(s+f)^2+\omega_r^2}{s+a} & -bc\frac{\omega_r}{(s+a)^2} & b\frac{s+f}{s+a} & -b\frac{\omega_r}{s+a} \\ bc\frac{\omega_r}{(s+a)^2} & \frac{(s+f)^2+\omega_r^2}{s+a} & b\frac{\omega_r}{s+a} & b\frac{s+f}{s+a} \\ c\frac{s+f}{s+a} & -c\frac{\omega_r}{s+a} & s+f & -\omega_r \\ c\frac{\omega_r}{s+a} & c\frac{s+f}{s+a} & \omega_r & s+f \end{bmatrix} \tag{3.30}$$

Now we can rewrite (3.25) as

$$\boldsymbol{x}(t) = \left(L^{-1}\left\{(sI-A)^{-1}\right\}\right)\boldsymbol{x}_0 + L^{-1}\left\{G(s)\boldsymbol{U}(s)\right\} \tag{3.31}$$

where $G(s) = (sI-A)^{-1}B$ is the transfer function matrix given by

$$G(s) = (sI-A)^{-1}B = \frac{1}{(s+f)^2+\omega_r^2}\begin{bmatrix} \frac{(s+f)^2+\omega_r^2}{s+a} & -bc\frac{\omega_r}{(s+a)^2} \\ bc\frac{\omega_r}{(s+a)^2} & \frac{(s+f)^2+\omega_r^2}{s+a} \\ c\frac{s+f}{s+a} & -c\frac{\omega_r}{s+a} \\ c\frac{\omega_r}{s+a} & c\frac{s+f}{s+a} \end{bmatrix} \tag{3.32}$$

and $U(s) = L\{\boldsymbol{u}(t)\}$ is the Laplace transform of the switch positions in $\alpha\beta$ reference frame given by

$$\boldsymbol{U}(s) = L\{\boldsymbol{u}(t)\} = \frac{1}{s}\begin{bmatrix} v_\alpha & v_\beta \end{bmatrix}^{\mathrm{T}} \tag{3.33}$$

Finally, (3.31) becomes

$$\boldsymbol{x}(t) = \left(L^{-1}\left\{(sI-A)^{-1}\right\}\right)\begin{bmatrix} x_{01} & x_{02} & x_{03} & x_{04} \end{bmatrix}^{\mathrm{T}} + L^{-1}\left\{\frac{G(s)}{s}\begin{bmatrix} v_\alpha & v_\beta \end{bmatrix}^{\mathrm{T}}\right\} \tag{3.34}$$

Taking the inverse Laplace transform and expanding (3.34) yields

$$\begin{aligned} x_1(t) \;=\;& l_1v_\alpha + l_2v_\beta + (x_{01} + k_1x_{02} + k_4x_{03} - k_5x_{04} - l_1v_\alpha + l_3v_\beta)\,\mathrm{e}^{-at} \\ &+ (k_2x_{02} + l_4v_\beta)\,t\mathrm{e}^{-at} + (k_3x_{02} + k_5x_{03} + k_4x_{04} + l_5v_\beta)\,\mathrm{e}^{-ft}\sin(\omega_r t) \\ &+ (-k_1x_{02} - k_4x_{03} + k_5x_{04} + l_6v_\beta)\,\mathrm{e}^{-ft}\cos(\omega_r t) \end{aligned} \tag{3.35}$$

$$\begin{aligned} x_2(t) \;=\;& l_1v_\beta - l_2v_\alpha + (-k_1x_{01} + x_{02} + k_5x_{03} + k_4x_{04} - l_3v_\alpha - l_1v_\beta)\,\mathrm{e}^{-at} \\ &+ (-k_2x_{01} - l_4v_\alpha)\,t\mathrm{e}^{-at} + (-k_3x_{01} - k_4x_{03} + k_5x_{04} - l_5v_\alpha)\,\mathrm{e}^{-ft}\sin(\omega_r t) \\ &+ (k_1x_{01} - k_5x_{03} - k_4x_{04} - l_6v_\alpha)\,\mathrm{e}^{-ft}\cos(\omega_r t) \end{aligned} \tag{3.36}$$

$$\begin{aligned} x_3(t) \;=\;& p_1 v_\alpha + p_2 v_\beta + (m_1 x_{01} - m_2 x_{02} + p_3 v_\alpha + p_4 v_\beta)\,\mathrm{e}^{-at} \\ & + (m_2 x_{01} + m_1 x_{02} - x_{04} + p_5 v_\alpha + p_6 v_\beta)\,\mathrm{e}^{-ft}\sin(\omega_r t) \\ & + (-m_1 x_{01} + m_2 x_{02} + x_{03} - p_6 v_\alpha + p_5 v_\beta)\,\mathrm{e}^{-ft}\cos(\omega_r t) \end{aligned} \tag{3.37}$$

$$\begin{aligned} x_4(t) \;=\;& p_1 v_\beta - p_2 v_\alpha + (m_2 x_{01} + m_1 x_{02} + p_3 v_\beta - p_4 v_\alpha)\,\mathrm{e}^{-at} \\ & + (-m_1 x_{01} + m_2 x_{02} + x_{03} + p_5 v_\beta - p_6 v_\alpha)\,\mathrm{e}^{-ft}\sin(\omega_r t) \\ & + (-m_2 x_{01} - m_1 x_{02} + x_{04} - p_6 v_\beta - p_5 v_\alpha)\,\mathrm{e}^{-ft}\cos(\omega_r t) \end{aligned} \tag{3.38}$$

where the coefficients k_i, l_i, m_i and p_i are functions of a, b, c and f. See Appendix A for details on how to derive them.

Inverter State The inverter's neutral point in (2.4) is nonlinear in terms of the switch positions $u_{abc} = [\;u_a \quad u_b \quad u_c\;]^{\mathrm{T}}$. However, as the vector u_{abc} and, consequently, $|u_{abc}|$ is known in the beginning of the extension step, it can be treated as a constant vector. This renders (2.4) a linear differential equation.

$$\tfrac{dv_n}{dt} \;=\; \tfrac{1}{2x_C}\,|u_{abc}|^T\,P^{-1}\boldsymbol{i}_{s,\alpha\beta 0} \;= \gamma_1 i_{s,\alpha} + \gamma_2 i_{s,\beta} \tag{3.39}$$

where $\left[\;\gamma_1 \quad \gamma_2 \quad \gamma_3\;\right] = \frac{1}{2x_C}\,|u_{abc}|^T\,P^{-1}$ are constant coefficients.

The $\alpha-$and $\beta-$components of $i_{s,\alpha\beta 0}$ are expressed in terms of the stator flux and the rotor flux α- and $\beta-$components according to (2.9). Substituting for $i_{s,\alpha}$ and $i_{s,\beta}$ from (2.9), (3.39) becomes

$$\dot{v}_n(t) = \frac{\gamma_1}{D}\,(x_{rr}x_1(t) - x_m x_3(t)) + \frac{\gamma_2}{D}\,(x_{rr}x_2(t) - x_m x_4(t)) \tag{3.40}$$

Integrating (3.40) gives

$$v_n(t) = \int_0^t \Big(\frac{\gamma_1}{D}\,(x_{rr}x_1(\tau) - x_m x_3(\tau)) + \frac{\gamma_2}{D}\,(x_{rr}x_2(\tau) - x_m x_4(\tau))\Big)\,d\tau + v_n(0) \tag{3.41}$$

The expression for $v_n(t)$ is obtained by plugging in the expressions for $x_i(t)$, $i = 1, ..., 4$, from (3.35), (3.36), (3.37) and (3.38) into (3.41), and integrating.

So far we have obtained the complete solutions of the states. These expressions, however, are too complicated to be used in the analytical approach straight away. In the next section, a simple analytical solution will be developed based on some simplifying assumptions on the model of the drive and the complete solutions derived here.

3.7.2 Simplified Solutions of State Equations

Consider the machine model again.

$$\begin{bmatrix} \frac{d\psi_{s\alpha}}{dt} \\ \frac{d\psi_{s\beta}}{dt} \\ \frac{d\psi_{r\alpha}}{dt} \\ \frac{d\psi_{r\beta}}{dt} \end{bmatrix} = \begin{bmatrix} -r_s\frac{x_{rr}}{D} & 0 & r_s\frac{x_m}{D} & 0 \\ 0 & -r_s\frac{x_{rr}}{D} & 0 & r_s\frac{x_m}{D} \\ r_r\frac{x_m}{D} & 0 & -r_r\frac{x_{ss}}{D} & -\omega_r \\ 0 & r_r\frac{x_m}{D} & \omega_r & -r_r\frac{x_{ss}}{D} \end{bmatrix} \begin{bmatrix} \psi_{s\alpha} \\ \psi_{s\beta} \\ \psi_{r\alpha} \\ \psi_{r\beta} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} \tag{3.42}$$

Stator Flux The $-r_s\frac{x_{rr}}{D}$ terms account for the losses in the stator whereas the $r_s\frac{x_m}{D}$ terms are the positive contributions from the the rotor flux to the stator flux. Since $\psi_r \approx \psi_s$, the losses in the stator are almost offset by the positive contributions from the rotor flux. Moreover, as r_s is typically very small, these terms become negligible. We shall consider two cases. (i) neglecting the effect of r_s (i.e. $r_s = 0$) and (ii) taking into account the effect of r_s.

i) Stator flux trajectories when $r_s = 0$

In this case the stator flux is only moved by the voltage vectors and the stator flux dynamics are reduced to

$$\dot{\boldsymbol{\psi}}_s = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \boldsymbol{\psi}_s + \begin{bmatrix} v_\alpha \\ v_\beta \end{bmatrix} \tag{3.43}$$

which yields the following expression for the stator flux

$$\boldsymbol{\psi}_s(t) = \boldsymbol{\psi}_s(0) + t\boldsymbol{v} \tag{3.44}$$

where $\boldsymbol{\psi}_s(0)$ are the initial conditions and t is the time.

The same result can be obtained by setting $r_s = 0$ in (3.35) and (3.36), which makes all coefficients zero apart from $l_1(l_1 = \frac{1}{a})$. This reduces them to

$$\begin{aligned} x_1(t) &= x_{01} + \lim_{a\to 0} \frac{v_\alpha}{a}\left(1 - \mathrm{e}^{-at}\right) \\ &= x_{01} + tv_\alpha \end{aligned} \tag{3.45}$$

$$\begin{aligned} x_2(t) &= x_{02} + \lim_{a \to 0} \frac{v_\beta}{a}\left(1 - \mathrm{e}^{-at}\right) \\ &= x_{02} + t v_\beta \end{aligned} \tag{3.46}$$

ii) Taking into account the effect of r_s

Including the effect of the stator resistance, r_s, allows us to predict the stator flux trajectories more accurately. For this purpose, we shall consider the most dominant terms in (3.35) and (3.36) that account for the stator losses as well as the interaction between the rotor flux and the stator flux. The terms associated with $l_1$and k_5 are determined to be the most dominant.

$$x_1(t) = x_{01}\mathrm{e}^{-at} + k_5\mathrm{e}^{-ft}\left(x_{03}\sin(\omega_r t) + x_{04}\left(\cos(\omega_r t) - 1\right)\right) + l_1 v_\alpha\left(1 - \mathrm{e}^{-at}\right) \tag{3.47}$$

$$x_2(t) = x_{02}\mathrm{e}^{-at} + k_5\mathrm{e}^{-ft}\left(x_{04}\sin(\omega_r t) - x_{03}\left(\cos(\omega_r t) - 1\right)\right) + l_1 v_\beta\left(1 - \mathrm{e}^{-at}\right) \tag{3.48}$$

By using first order Taylor expansions of e^{-at}, $\mathrm{e}^{-ft}\sin(\omega_r t)$ and $\mathrm{e}^{-ft}\cos(\omega_r t)$ around the nominal time-duration of the prediction horizon t_p, (3.47) and (3.48) can be expressed, in the original variables $\begin{bmatrix} \psi_{s\alpha} & \psi_{s\beta} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^{\mathrm{T}}$, as

$$\psi_{s\alpha}(t) = s_1 t + r_1 \tag{3.49}$$

$$\psi_{s\beta}(t) = s_2 t + r_2 \tag{3.50}$$

where s_i and r_i are constants. See Appendix A for details.

Fig. 3.3 depicts the effect of r_s on approximating the stator flux components.

Rotor Flux The ω_r terms correspond to the rotation of the rotor flux vector with constant magnitude, whereas the $-r_r\frac{x_{ss}}{D}$ terms account for the losses due to r_r that shorten the length of the rotor flux. The $r_r\frac{x_m}{D}$ terms are the positive contributions from the stator flux due to the mutual inductance x_m. As the stator and the rotor flux almost lie on top of each other (i.e., $\psi_s \approx \psi_r$), the losses in the rotor flux are almost offset by the positive contribution from the stator flux. Moreover, r_r and consequently $-r_r\frac{x_{ss}}{D}$ and $r_r\frac{x_m}{D}$ are typically very small compared to ω_r. These observations allow us to omit the $-r_r\frac{x_{ss}}{D}$ and $r_r\frac{x_m}{D}$ terms in (3.42), which yields

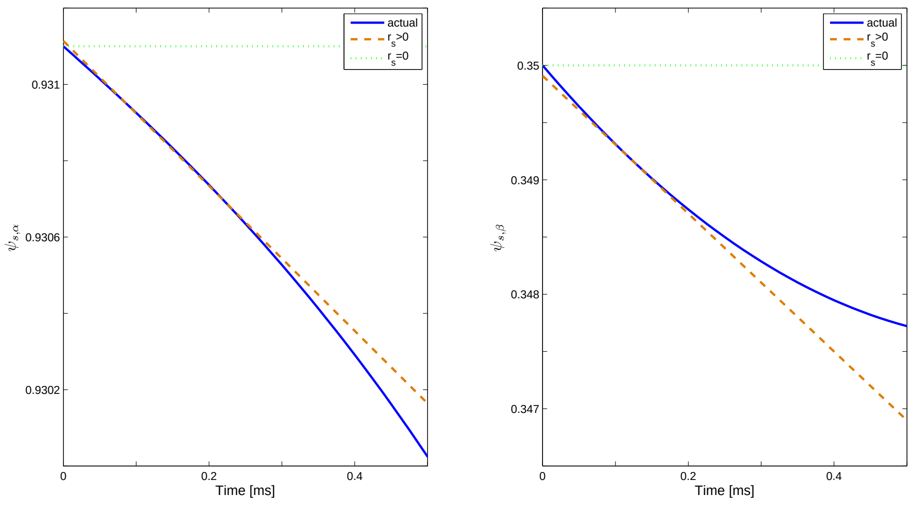


Figure 3.3: Trajectories of the stator flux components for some initial conditions and $u_k = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^{\mathrm{T}}$. For zero voltage vectors, the approximated flux components associated to $r_s = 0$ case (dotted line) are constant. The dashed line approximate the flux components more accurately in the vicinity of the nominal time-duration of the prediction horizon (i.e., 0.1 ms).

$$\dot{\psi}_r = \begin{bmatrix} 0 & -\omega_r \\ \omega_r & 0 \end{bmatrix} \psi_r \tag{3.51}$$

Equation (3.51) is a system rotating with a fixed speed and a constant magnitude. The solution of (3.51) is

$$\begin{bmatrix} \psi_{r\alpha} \\ \psi_{r\beta} \end{bmatrix} = \begin{bmatrix} \cos(\omega_r t) & \sin(\omega_r t) \\ -\sin(\omega_r t) & \cos(\omega_r t) \end{bmatrix} \begin{bmatrix} \psi_{r\alpha}(0) \\ \psi_{r\beta}(0) \end{bmatrix} \tag{3.52}$$

or equivalently

$$\psi_{r\alpha}(t) = \Psi_r \cos(\omega_r t + \Theta) \tag{3.53}$$

$$\psi_{r\beta}(t) = \Psi_r \sin(\omega_r t + \Theta) \tag{3.54}$$

where $\Psi_r = \sqrt{\psi_{r\alpha}^2(0) + \psi_{r\beta}^2(0)}$ and $\Theta = \arcsin\left(\frac{\psi_{r\beta}(0)}{\Psi_r}\right)$ are the length and angle of the rotor flux, respectively. Fig. 3.4 shows that the rotor flux can be seen as a rotating vector with constant speed and length over the prediction horizon.

Equations (3.53) and (3.54) can be verified by setting $r_r = 0$ in (A.14) and (3.38). We use second order Taylor expansions of $\Psi_r \sin(\omega_r t + \Theta)$ and $\Psi_r \cos(\omega_r t + \Theta)$ around the nominal

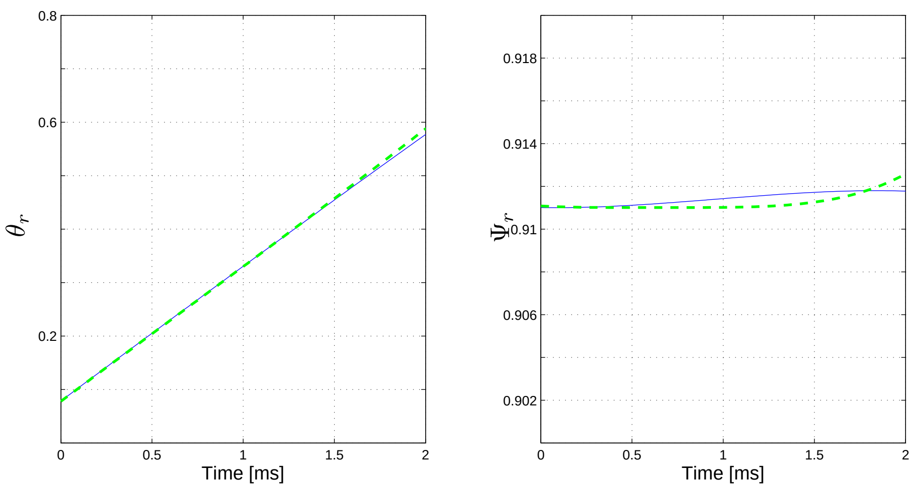


Figure 3.4: The actual (solid) and approximated (dashed) phase and angle of the rotor flux over a typical prediction horizon.

duration of the prediction horizon t_p, to express the rotor flux components in polynomials of degree two.

$$\psi_{r\alpha}(t) = a_1 t^2 + b_1 t + c_1 \tag{3.55}$$

$$\psi_{r\beta}(t) = a_2 t^2 + b_2 t + c_2 \tag{3.56}$$

Please refer to Appendix A for explicit expressions of a_i, b_i and c_i. The approximated rotor flux components are depicted in Fig 3.5b.

Inverter's Neutral Point Potential Using (3.49), (3.50), (3.55) and (3.56) in (3.41) yields

$$\begin{aligned} v_n &= \int_0^t \Big(\frac{\gamma_1}{D} \left(x_{rr} \left(s_1 \tau + r_1 \right) - x_m \Psi_r \cos\left(\omega_r t + \Theta \right) \right) + \\ &\quad + \frac{\gamma_2}{D} \left(x_{rr} \left(s_2 \tau + r_2 \right) - x_m \Psi_r \sin\left(\omega_r t + \Theta \right) \right) \Big) \, d\tau + v_n(0) \\ &= \frac{\gamma_1}{D} \left(x_{rr} \left(\frac{s_1}{2} t^2 + r_1 t \right) - x_m \frac{\Psi_r}{\omega_r} \left(\sin\left(\omega_r t + \Theta \right) - \sin\left(\Theta \right) \right) \right) \\ &\quad + \frac{\gamma_2}{D} \left(x_{rr} \left(\frac{s_2}{2} t^2 + r_2 t \right) + x_m \frac{\Psi_r}{\omega_r} \left(\cos\left(\omega_r t + \Theta \right) - \cos\left(\Theta \right) \right) \right) + v_n(0) \end{aligned} \tag{3.57}$$

By using the second order Taylor expansion of $\Psi_r \cos\left(\omega_r t + \Theta\right)$ and $\Psi_r \sin\left(\omega_r t + \Theta\right)$ around the nominal duration of the prediction horizon, equation (3.57) will be a polynomial of degree two.

$$v_n(t) = u_1 t^2 + u_2 t + u_3 \tag{3.58}$$

The coefficients u_i are derived in Appendix A.

3.7.3 Output Polynomials

The polynomial forms of the state trajectories, obtained in Sect. 3.7.2, are used along with the output equations in Chapter 2 to express the outputs in polynomials of degree 3 or less. This allows us to solve for the length of the prediction horizon analytically.

The stator flux polynomial is a second order polynomial obtained by using (3.49) and (3.50) in (2.8).

$$\begin{aligned} \Psi_s^2(t) &= (s_1 t + r_1)^2 + (s_2 t + r_2)^2 \\ &= \left(s_1^2 + s_2^2\right) t^2 + 2t\left(s_1 r_1 + s_2 r_2\right) + r_1^2 + r_2^2 \end{aligned} \tag{3.59}$$

The time duration for which the stator flux remains within its bounds is given by

$$\Psi_s(t) - \Psi_{s,min} = 0 \rightarrow t = t_1^* \geq 0 \tag{3.60}$$

$$\Psi_s(t) - \Psi_{s,max} = 0 \rightarrow t = t_2^* \geq 0 \tag{3.61}$$

$$t_{p1} = \min(t_1^*, t_2^*) \tag{3.62}$$

where $\Psi_{s,min}$ and $\Psi_{s,max}$ are the lower and upper bound on the stator flux.

The torque is a third degree polynomial which is obtained by using (3.49), (3.50), (3.55) and (3.56) in (2.7)

$$\begin{aligned} T_e(t) &= \frac{x_m}{D}\left(\left(s_2 t + r_2\right)\left(a_1 t^2 + b_1 t + c_1\right) - \left(s_1 t + r_1\right)\left(a_2 t^2 + b_2 t + c_2\right)\right) \\ &= \frac{x_m}{D}\left(\left(s_2 a_1 - s_1 a_2\right) t^3 + \left(s_2 b_1 + r_2 a_1 - s_1 b_2 - r_1 a_2\right) t^2 \right. \\ &\quad \left. + \left(s_2 c_1 + r_2 b_1 - s_1 c_2 - r_1 b_2\right) t + \left(r_2 c_1 - r_1 c_2\right)\right) \end{aligned} \tag{3.63}$$

It is always possible to find the real root(s) of a third order polynomial analytically[4]. See Appendix A for details. The time duration for which the torque remains feasible is given by

[4] A third order polynomial has at least one real root

$$T_e(t) - T_{e,min} = 0 \rightarrow t = t_1^* \tag{3.64}$$

$$T_e(t) - T_{e,max} = 0 \rightarrow t = t_2^* \tag{3.65}$$

$$t_{p2} = \min(t_1^*, t_2^*) \tag{3.66}$$

where $T_{e,min}$ and $T_{e,max}$ are the lower and upper bounds on the torque.

The polynomial form of the inverter's neutral point potential was obtained in Sect. 3.7.2.

$$v_n(t) = u_1 t^2 + u_2 t + u_3 \tag{3.67}$$

The time duration for which the neutral point potential remains within its bounds is given by

$$v_n(t) - v_{n,min} = 0 \rightarrow t = t_1^* \tag{3.68}$$

$$v_n(t) - v_{n,max} = 0 \rightarrow t = t_2^* \tag{3.69}$$

$$t_{p3} = \min(t_1^*, t_2^*) \tag{3.70}$$

where $v_{n,min}$ and $v_{n,max}$ are the associated lower and upper bounds .

The time-duration for which all outputs are feasible is given by

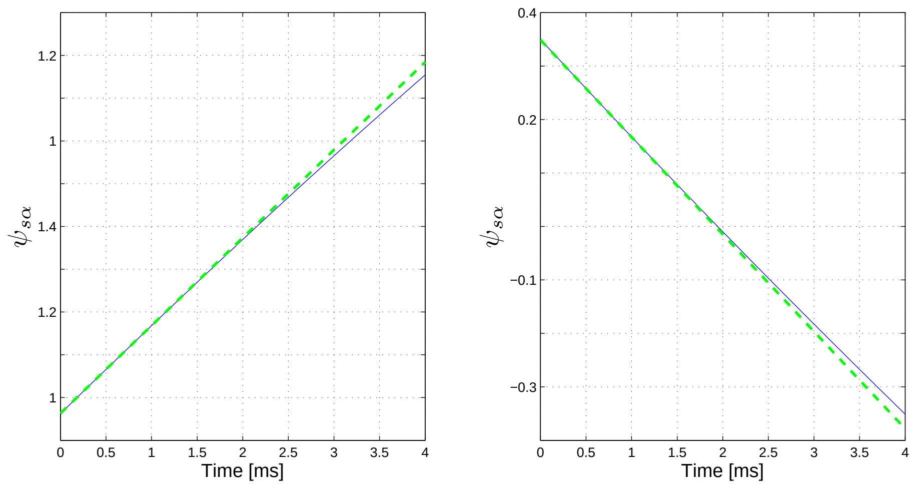
$$t_p = \min\left(t_{p1}, t_{p2}, t_{p3}\right) \tag{3.71}$$

The prediction horizon length, N_p, is the number of time-steps corresponding to t_p, which is given by

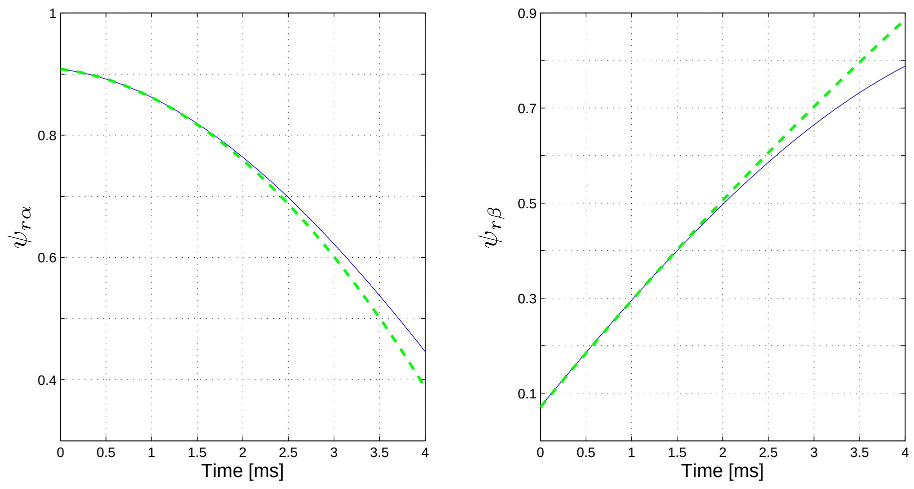
$$N_p = \left\lfloor \frac{t_p}{T_s} \right\rfloor \tag{3.72}$$

where T_s is the sampling time.

Fig. 3.5 compares the actual state trajectories with the approximate trajectories for given initial conditions and input. Fig. 3.6 shows the associated output trajectories.

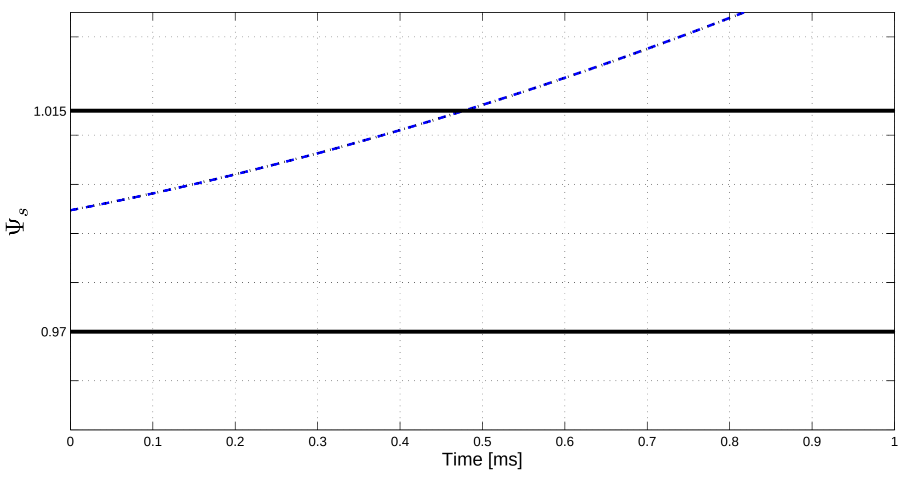


(a) Approximation of the stator flux components (solid line) by polynomials of degree one (dashed line) taking into account the effect of r_s.

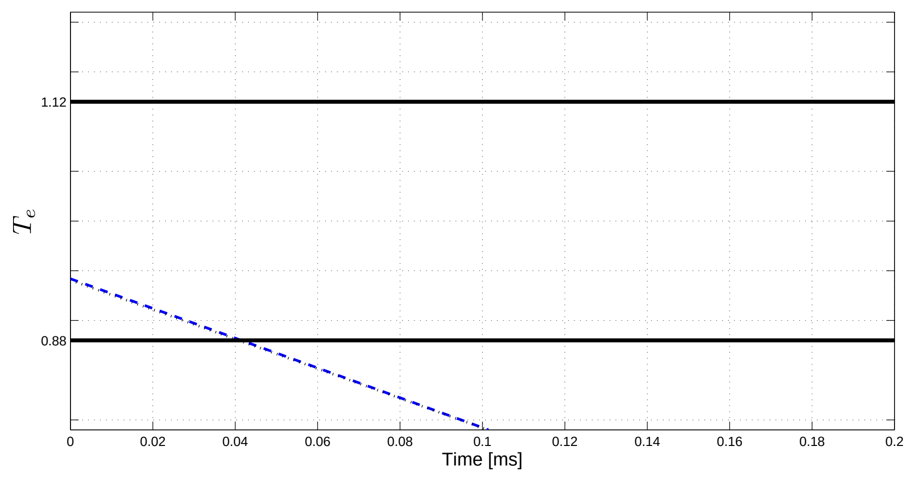


(b) Approximation of the rotor flux components (solid line) by polynomials of degree two (dashed line).

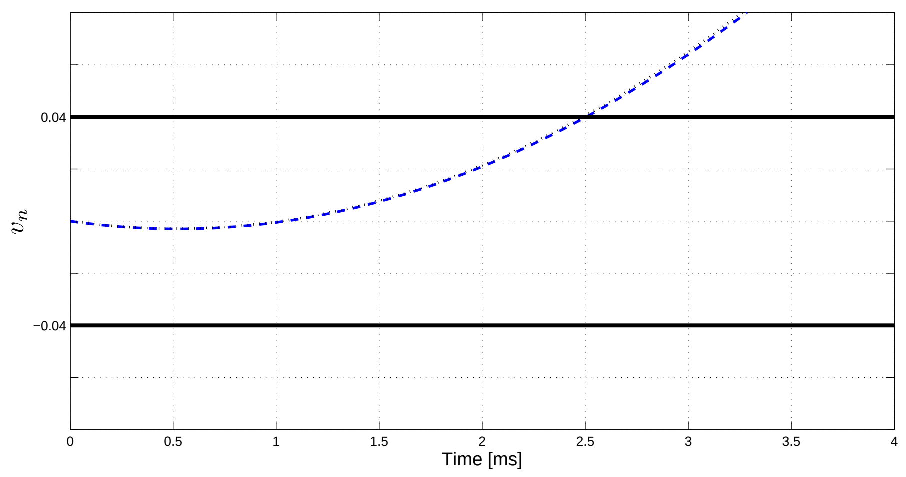
Figure 3.5: Comparison between simplified state solutions and exact state solutions for given initial conditions and input.



(a) Approximation of the length of the stator flux (dashed line) by its associated polynomial (dotted line).



(b) Approximation of the torque (dashed line) by its associated polynomial (dotted line).



(c) Approximation of the inverter's neutral point potential (dashed line) by its associated polynomial (dotted line).

Figure 3.6: Comparison between the exact output trajectories and their approximations, which are based on the simplified model. The trajectories are drawn between their bounds and almost lie on top of each other.

Chapter 4

Performance Evaluation of Extension Methods

4.1 Accuracy of Predictions

Using a 3-level voltage source inverter with a 2.5 MVA induction machine [5] as an example, the accuracy of the different extension methods is compared with the open-loop simulation method as it yields the most accurate predictions. For this purpose, we define the relative prediction horizon error as $N_p^{rel} = \frac{N_p^{ol} - N_p}{N_p^{ol}}$ where N_p and N_p^{ol} are the prediction horizons of the method under consideration and of the open-loop simulation method, respectively. The accuracy of predictions associated with each method is measured by the relative error histograms. The accuracy of predictions along the edges of the target window, as shown in Fig. 4.1, is depicted in Fig. 4.2. Since the performance of the QII and QII2 methods are almost identical, we have only shown the histograms for the QII method.

A few remarks are to be mentioned about the histograms of relative error of the prediction horizon length. They show how accurately the outputs can be predicted at the end of one extension step (i.e. for 'SE' switching horizon). We need the state vector in the switching step to build the output samples used for extending the trajectories. Thus, for long switching horizons like 'SESE', we have to obtain the state vector at the end of each 'E' event. For the 'OL' method, the state vector is automatically obtained as a part of the extension step. For other methods, the states are extended in the same way as outputs. In order to keep the correspondence between the extended state and output vectors, we need to predict the states accurately. Particularly the rotor states cannot be predicted accurately by linear extrapolation as discussed in Chapter 3. This is shown in Fig. 4.3 where the discontinuity in the predicted output trajectory happens because the correspondence between the extended output and state vector is lost. Therefore, for long

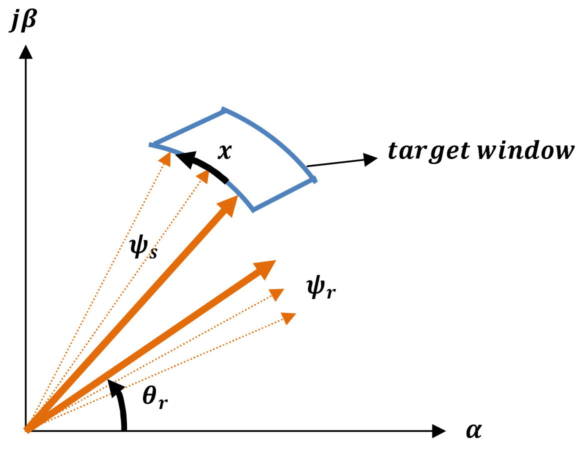


Figure 4.1: The region over which the histograms of relative error of the prediction horizon length are obtained. x is the distance traveled along the boundary of target window and θ_r is the rotor angle.

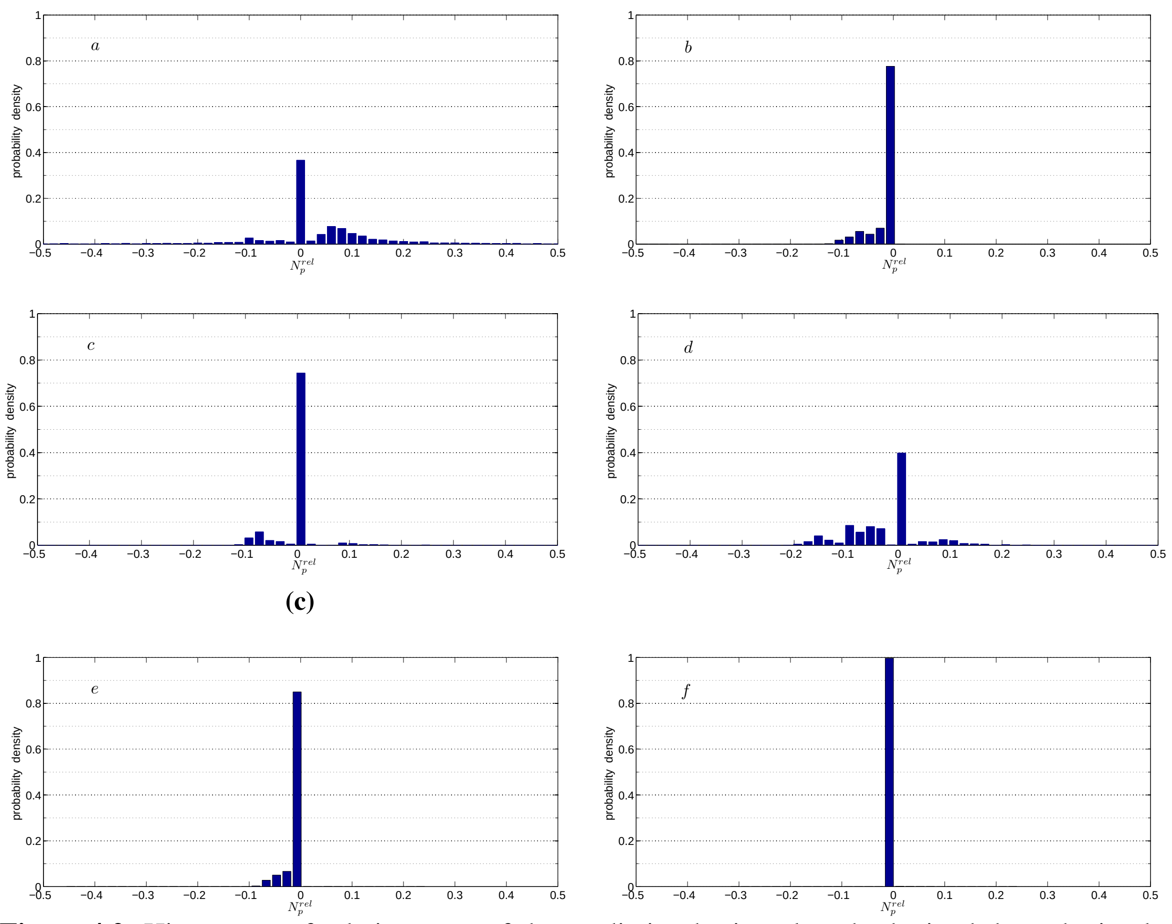


Figure 4.2: Histograms of relative error of the prediction horizon length obtained through simulation along the edges of the feasible region for the switching horizon 'SE', at $T_e = 1$pu and $\omega_e = 0.8$pu, where T_e and ω_e are the electromagnetic torque and speed references, respectively. (a) LE, (b) QE, (c) ANL with $t_p = 7T_s$, (d) ANL with $r_s = 0$, (e) QI with $d = 14$ and (f) QII with $d = 7$. The probability density of the relative error for the 'QII' method is highly concentrated about zero highlighting its accuracy.

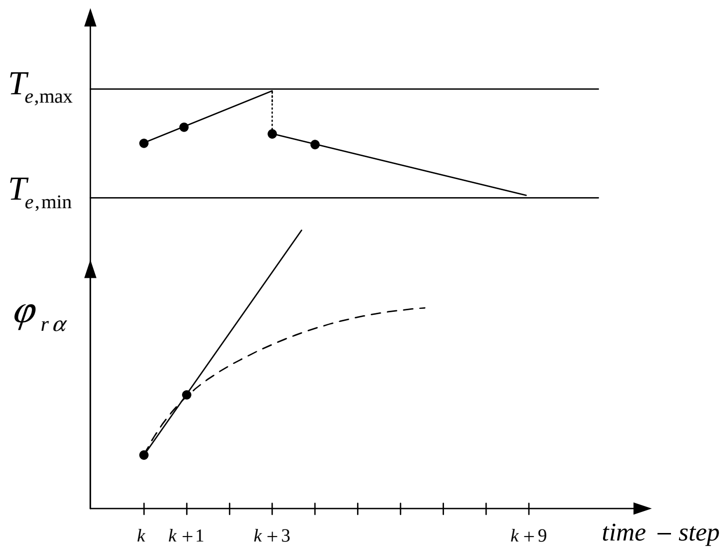


Figure 4.3: The extended torque and $\psi_{r\alpha}$ trajectories using 'LE' method. The first 'S' event takes place at time-step k. The torque and state trajectories are extend until at time-step $k+3$ the next 'S' action is triggered. However, a jump in the predicted output trajectory happens at time-step $k+3$ because the extended state (solid line) is an inaccurate estimation of the actual state (dashed-line).

Table 4.1: Number of operations in the sub-functions of the pseudo code

	Basic Operations			
function	$\pm$	$\times$	$\div$	sum
StateUpdate(Pred)	42	76	2	120
OutputUpdate	2	9	0	11
LinSolve	12	0	6	18
QuadSolve	42	39	30	111
AnlSolveStator	14	23	4	41
AnlSolveTorque	20	50	16	86
AnlSolveNeutral	27	69	23	119

switching horizons the 'LE' method does not work properly although it yields rather accurate predictions for the 'SE' switching horizon.

4.2 Computational Complexity

In this section the algorithms for the proposed extension methods are given as pseudo code and the associated computational burden is analyzed. As with modern Digital Signal Processors all basic operations such as additions, multiplications, divisions require one cycle, we only count the number of basic operations and do not distinguish between them. All other operations for loading and storing the variables, comparisons and execution of loops are neglected.

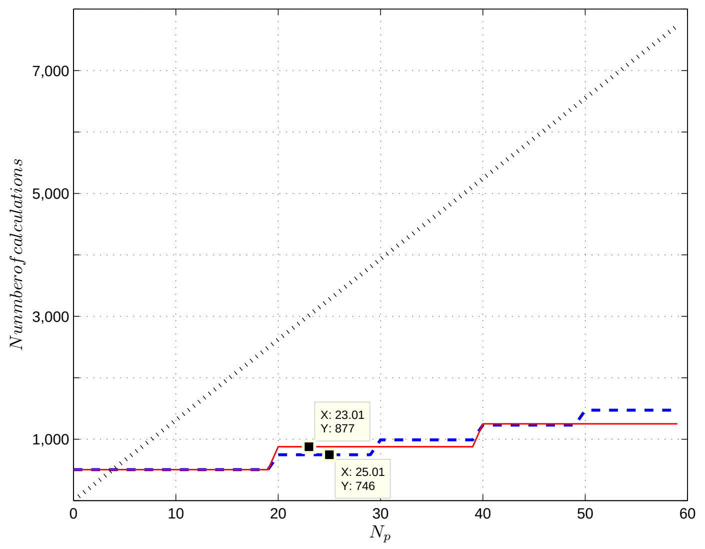


Figure 4.4: The number of calculations plotted as a function of the prediction horizon length for the OL (dotted line), QII (dashed line) and QII2 (solid line) methods for the switching horizon 'SE' and $d = 10$ steps. The computational burden for the QII and QII2 methods increases stepwise at every d and $2d$ steps, respectively. As shown in the graph, the number of the calculations for QII method is below QII2's for $d \leq N_p \leq 3d$, which is the most likely prediction horizon length if the parameter d is chosen properly.

4.2.1 Pseudo codes and Analysis

Table 4.1 shows the number of basic operations involved with each sub-function in the pseudo codes. The sub-function `StateUpdate` updates the drive's current state using the model discretized at T_s. `StateUpdatePred` updates the driver's current state using the model sampled at dT_s. `LinSolve` solves for the length of the linearly extrapolated trajectories. `QuadSolve` solves for length of the trajectories, which are extended by quadratic curves. The functions `AnlSolveStator`, `AnlSolveTorque` and `AnlSolveNeutral` are used in the analytical approach to solve the stator flux magnitude, the torque and the neutral point polynomials, respectively.

Let $X(k) = [x(k)]$, $Y(k) = [y(k)]$ denote the state and the associated output trajectories to be extended and $u(k)$ denote the last switching input in the switching sequence. The state and output samples from k on are computed to obtain $X(k) = [\ x(k),\ \ ...,\ \ x(k+N_p-1)\ ]$ and $Y(k) = [\ y(k),\ \ ...,\ \ y(k+N_p-1)\ ]$ until the output trajectory becomes infeasible. The results are stored in data structure `Pred`. Fig. 4.4 compares the computational complexity of the iterative methods (i.e., OL, QII and QII2).

Table 4.2: The computational burden associated with each extension method, where $\lceil . \rceil$ denotes the ceiling function.

extension method	LE	QE	QI	ANL
solution approach	analytic	analytic	analytic	analytic
total number of calc.	149	373	373	246

extension method	QII	QII2	OL
solution approach	sequential	sequential	sequential
total number of calc.	$\begin{cases} 242\left\lceil\frac{N_p}{d}\right\rceil - 111 & N_p > 2d \\ 373 & N_p \leq 2d \end{cases}$	$373\left\lceil\frac{N_p}{2d}\right\rceil$	$131N_p$

Open-loop Simulation (OL)

```
Pred = [];
Np=1;
X(k) = x(k);
Y(k) = y(k);
While y(k+Np-1) feasible
    x(k+Np)=StateUpdate(x(k+Np-1),u(k));
    y(k+Np)=OutputUpdate(x(k+Np));
    Np = Np + 1;
    X(k) = [X(k) x(k+Np-1)];
    Y(k) = [Y(k) y(k+Np-1)];
end
store X(k), Y(k) in Pred;
```

The code script in the `while` loop runs as many times as the length of the prediction horizon, N_p. The number of calculations for one 'E' event is give by

$$C_E = N_p\left(120 + 11\right) = 131N_p \tag{4.1}$$

Linear Extrapolation (LE)

Let $Y_{max}$and Y_{min} denote the upper and lower bounds on the output variables, respectively.

```
Pred = [];
X(k) = x(k);
Y(k) = y(k);
x(k+1)=StateUpdate(x(k),u(k));
y(k+1)=OutputUpdate(x(k+1));
[X(k), Y(k), Np] = LinSolve(y(k),y(k+1),Ymax,Ymin);
store X(K), y(K) in Pred;
```

For LE method, solution of the prediction horizon length can be found analytically so the computational burden is not a function of N_p and is given by

$$C_{LE} = 120 + 11 + 18 = 149 \tag{4.2}$$

Quadratic Extrapolation (QE)

```
Pred = [];
X(k) = x(k);
Y(k) = y(k);
x(k+1)=StateUpdate(x(k),u(k));
y(k+1)=OutputUpdate(x(k+1));
x(k+2)=StateUpdate(x(k+1),u(k));
y(k+2)=OutputUpdate(x(k+2));
[X(k), Y(k), Np] = QuadSolve(y(k),y(k+1),y(k+2),Ymax,Ymin);
store X(k), Y(k) in Pred;
```

Similar to LE, the solution for prediction horizon length is found analytically. The number of calculations for one 'E' event is given by

$$C_{QE} = 2 \times (120 + 11) + 111 = 373 \tag{4.3}$$

Prediction with Quadratic Interpolation (QI)

```
Pred = [];
X(k) = x(k);
Y(k) = y(k);
x(k+d)=StateUpdatePred(x(k),u(k));
x(k+2d)=StateUpdatePred(x(k+d),u(k);
y(k+d)=OutputUpdate(x(k+d));
y(k+2d)=OutputUpdate(x(k+2d));
[X(k), Y(k), Np] = QuadSolve(y(k+1),y(k+1+d),y(k+1+2d),Ymax,Ymin);
store X(k), Y(k) in Pred;
```

The solution of N_p is obtained analytically thus this approach is non-iterative. The number of required calculations is given by

$$C_{QI} = 2 \times (120 + 11) + 111 = 373 \tag{4.4}$$

Iterative Prediction with Quadratic Interpolation (QII)

```
Pred = [];
X(k) = x(k);
Y(k) = y(k);
Np = 2d;
x(k+d)=StateUpdatePred(x(k),u(k));
y(k+d)=OutputUpdate(x(k+d));
j = 1;
While Np>=2d
  x(k+(j+1)d)=StateUpdatePred(x(k+jd),u(k));
  y(k+(j+1)d)=OutputUpdate(x(k+(j+1)d));
  [X_j(k), Y_j(k), Np] = QuadSolve(y(k+(j-1)d),y(k+jd),y(k+(j+1)d),Ymax,Ymin);
  X(k) = [X(k) X_j(k)];
  Y(k) = [Y(k) Y_j(k)];
  j = j + 1;
end
store X(k), Y(k) in Pred;
```

The number of times for which the while loop is executed is given by

$$n = \begin{cases} \left\lceil \frac{N_p}{d} \right\rceil - 1 & N \geq 2d \\ 1 & N < 2d \end{cases} \tag{4.5}$$

The `Quadsolve` function is called n times while the `StateUpdate(Pred)` and `OutputUpdate` functions are called $n+1$ times. Therefore, the total number of basic operations is given by

$$\begin{aligned} C_{QII} &= \begin{cases} 131 \left\lceil \frac{N_p}{d} \right\rceil + 111 \left(\left\lceil \frac{N_p}{d} \right\rceil - 1 \right) & N_p > 2d \\ 131 \times 2 + 111 & N_p \leq 2d \end{cases} \\ &= \begin{cases} 242 \left\lceil \frac{N_p}{d} \right\rceil - 111 & N_p > 2d \\ 373 & N_p \leq 2d \end{cases} \end{aligned} \tag{4.6}$$

As in ’OL’ method, the computational burden depends on the length of the prediction horizon.

Prediction with Quadratic Interpolation non-overlapping (QII2)

```
Pred = [];
X(k) = x(k);
Y(k) = y(k);
N = 2d;
j = 1;
```

```
While Np>=2d
  x(k+(2j-1)d)=StateUpdatePred(x(k+2d(j-1)),u(k));
  y(k+(2j-1)d)=OutputUpdate(x(k+(2j-1)d));
  x(k+2jd)=StateUpdatePred(x(k+(2j-1)d),u(k));
  y(k+2jd)=OutputUpdate(x(k+2jd));
  [X_j(k), Y_j(k), Np] = QuadSolve(y(k+2d(j-1)),y(k+(2j-1)d),y(k+2jd),Ymax,Ymin);
  X(k) = [X(k) X_j(k)];
  Y(k) = [Y(k) Y_j(k)];
  j = j + 1;
end
store X(k), Y(k) in Pred;
```

The number of times for which the while loop is executed is given by

$$n = \left\lceil \frac{N_p}{2d} \right\rceil \tag{4.7}$$

The `Quadsolve` function is called n times while the `StateUpdate(Pred)` and `OutputUpdate` functions are called $2n$ times. Therefore, the total number of basic operations is given by

$$\begin{aligned} C_{QII2} &= 131\left(2\left\lceil \frac{N_p}{2d} \right\rceil\right) + 111\left(\left\lceil \frac{N_p}{2d} \right\rceil\right) \\ &= 373\left\lceil \frac{N_p}{2d} \right\rceil \end{aligned} \tag{4.8}$$

As in 'QII' method, the computational burden depends on the length of the prediction horizon.

Analytical Method

```
Pred = [];
X(k) = x(k);
Y(k) = y(k);
[ParamStator Np1] = AnlSolveStator(x(k),Ymax,Ymin);
[ParamRotor Np2] = AnlSolveTorque(x(k),ParamStator,Ymax,Ymin);
[ParamNeutral Np3 = AnlSolveNeutral(x(k),ParamStator,ParamRotor,Ymax,Ymin);
Np = min(Np1,Np2,Np3);
[X(k), Y(k)] = extendTraj(ParamStator,ParamRotor,ParamNeutral);
store X(k), Y(k) in Pred;
```

The number of calculations for each 'E' event is given by

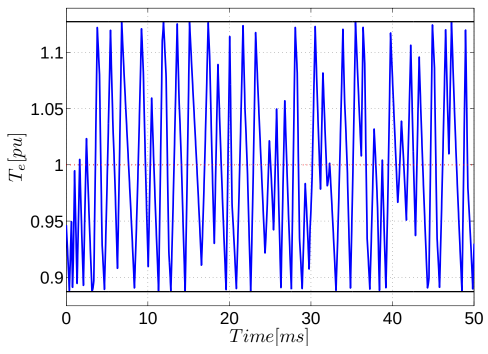
$$C_{ANL} = 246 \tag{4.9}$$

4.3 Closed-loop Performance

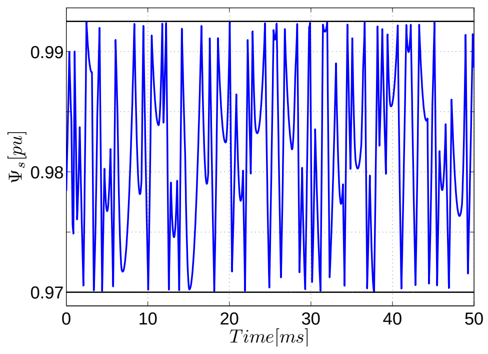
Using a 3-level voltage source inverter with a 2.5 MVA induction machine [5] as an example, the closed-loop performance of different methods is compared for different switching horizons when the switching frequency is minimized. Table 4.3 and Table 4.4 compare the extension methods in terms of the switching frequency, the torque THD and the switching losses. The results suggest that for the switching horizon 'SE' all methods exhibit fairly similar performance. As can be seen for the open-loop simulation method, with respect to MPDTC with the switching horizon 'SE', MPDTC with the switching horizon 'SSESE' reduces the switching frequency by about 20$\%$, while the switching losses are reduced by almost 30$\%$. At the same time, the torque THD almost remains constant as the switching horizon is increased. The QI and QII exhibit the same performance as the open-loop simulation method as the switching horizon is increased. For the analytical method, with respect to MPDTC with the switching horizon 'SE', MPDTC with the switching horizon 'SSESE' reduces the switching frequency by almost 14$\%$, while the switching losses are reduced almost by 20$\%$. This inferior performance as compared to the performance of the QI and QII methods, is partly due to the fact that the output equations are not approximated about the correct value of t_p. As mentioned before, the LE methods is not applicable for long switching horizons. The QE method can be used for long horizons only when the machine operates at $\omega_e < 0.7$pu. Fig. 4.5 shows the switch positions and the associated outputs.

Table 4.3: Closed-loop simulation results when minimizing the switching frequency. The table shows the average inverter switching frequency (Hz), the torque's total harmonic distortion (THD) and the switching losses for different extension methods for the machine running at 100$\%$ torque and 80$\%$ speed.

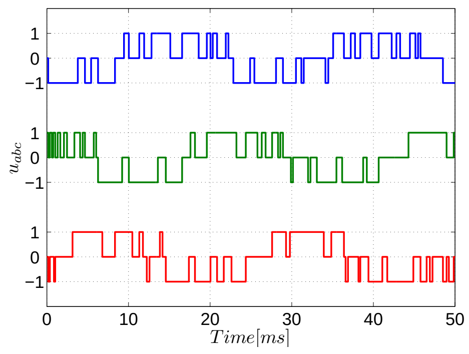
Switching Horizon	Extension Method	OL	LE	QE	QI	QII	ANL
	Freq. (Hz)	224	229	228	224	224	221
'SE'	$T_{e,THD}$	6.50	6.50	6.63	6.63	6.63	6.64
	Losses (kW)	6.15	6.26	6.29	6.14	6.14	6.07
	Freq. (Hz)	191	N/A	N/A	198	196	200
'SESE'	$T_{e,THD}$	6.44	N/A	N/A	6.43	6.53	6.63
	Losses (kW)	4.50	N/A	N/A	4.98	4.89	5.15
	Freq. (Hz)	182	N/A	N/A	184	182	192
'SSESE'	$T_{e,THD}$	6.40	N/A	N/A	6.22	6.45	6.53
	Losses (kW)	4.35	N/A	N/A	4.64	4.41	4.81



(a) The torque



(b) The length of the stator flux



(c) 3-phase switch positions

Figure 4.5: Closed-loop simulation for the switching horizon 'SE' using the 'OL' at $\omega_e = 0.8$pu, $T_e = 1$pu

Table 4.4: Closed-loop simulation results when minimizing the switching frequency. The table shows the average inverter switching frequency (Hz), the torque's total harmonic distortion (THD) and the switching losses for different extension methods for the machine running at 100% torque and 50% speed.

Switching Horizon	Extension Method	OL	LE	QE	QI	QII	ANL
'SE'	Freq. (Hz)	194	202	199	198	198	205
	$T_{e,THD}$	6.24	6.26	6.27	6.15	6.14	6.11
	Losses (kW)	7.21	7.21	7.22	7.27	7.27	7.14
'SESE'	Freq. (Hz)	170	N/A	171	168	172	175
	$T_{e,THD}$	6.28	N/A	6.13	6.14	6.16	6.26
	Losses (kW)	6.41	N/A	6.36	6.37	6.41	6.00

Chapter 5

Branch and Bound

5.1 Background

As mentioned in Chapter 2, the MPDTC algorithm explores the tree of all admissible switching sequences to find the candidate switching sequences. The number of allowed switching transitions at every time-step to the power of number of 'S' events in the switching horizon determines the worst case computational complexity of the MPDTC algorithm. The number of allowed switching transitions at every-time step is determined by the inverter's topology, whereas the switching horizon is a control parameter. Long switching horizons improve the control performance in terms of the switching losses and the torque/current THD[1] [2]. So far, we have investigated all admissible switching sequences to find the candidate sequences. This method is called *full enumeration*. However, this approach cannot be used for long switching horizons due to the combinatorial explosion in the number of admissible switching sequences. In [3], it is shown that, using *branch and bound* technique, certain parts of the search tree can be discarded without affecting the performance and thus reducing the computational complexity significantly.

Starting from the *root node*, child nodes are created from the parent nodes by applying the control events in the switching horizon. The child nodes whose associated output sequences are non-candidate, are called *non-candidate nodes*. These nodes will be removed from the tree and will not be branched on any further (i.e., they won't yield any child nodes.) The so far candidate child nodes with control events left are referred to as *bud nodes*, which are incomplete solutions of the optimization problem. These nodes are considered for further branching. The nodes with no control events left, whose associated output sequences meet the candidacy requirements at every time-step, are referred to as *complete candidate nodes*. They correspond to fully calculated candidate switching sequences. Fig. 5.1 provides an example of the search tree, where

[1]Total harmonic distortion

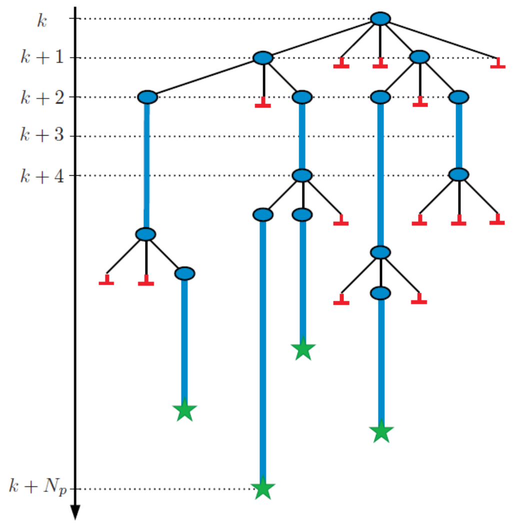


Figure 5.1: Examples of search tree for the switching horizon 'SSESE'. The ovals (blue), the stars (green) and the inverted (red) Ts denote the incomplete, candidate and non-candidate switching sequences (nods), respectively. The 'S' events are shown by thin (black) lines while the 'E' events are shown by thick (blue) lines. This picture is taken from [3].

the bud and root nodes, the non-candidate nodes and the complete candidate nodes are marked with blue (ovals), inverted (red) Ts and (green) starts, respectively.

Now we will define some terminology. The index i refers to the i-th switching sequence (node).

- Let E_i and $N_{p,i}$ represent the sum of switching transitions (switching losses) and the length of the switching sequence, respectively. The cost associated with a complete switching sequence is given by $c_i = \frac{E_i}{N_{p,i}T_s}$, where T_s is the controller sampling-time.
- c^* is the optimal (minimal) cost for all candidate switching sequences.
- $\overline{c}$ is the smallest cost found in the search tree so far. We have $\overline{c} \geq c^*$.
- $N_{p,max}$ is the maximal length of the prediction horizon and $N_{p,max} \geq N_{p,i}$ for all i.
- $\underline{c_i} = \frac{E_i}{N_{p,max}T_s}$ is the lowest cost achievable by the i-th bud node (incomplete switching sequence), where E_i is the sum of the switching losses (transitions) incurred so far for this sequence.

The formal definition of the branch and bound algorithm tailored to the MPDTC problem is described in detail in [3]. Here we will introduce the algorithm briefly by an example.

Example 1: Consider Fig 5.2. Assume that node 1 (solid-line) is an incumbent node found at some stage during the evolution of the search tree with the associated cost $c_1 = \frac{e_1+e_2}{N_{p,1}T_s}$, where

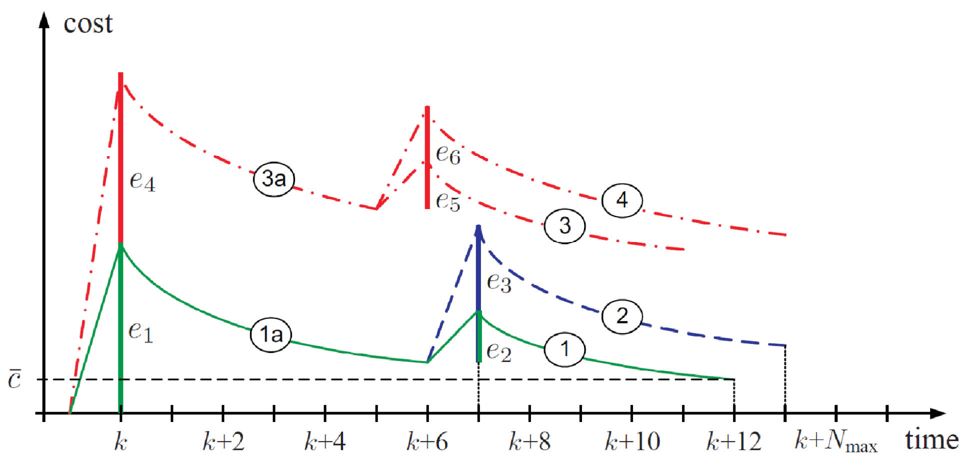


Figure 5.2: The cost evolution of four candidate sequences taken from [3].

$N_{p,1} = 12$ time-steps. As node 1 is the only candidate sequence found so far, the incumbent minimal cost becomes $\bar{c} = c_1$. Having computed the second switching transition at $k + 7$ with switching losses e_3, we can find a proof before extending sequence 2 (dashed-line) that this node and its descendants will not yield any better solution than what we already have, i.e. it will yield to solutions that are inferior to the incumbent minimal (node 1). To achieve this, the algorithm compares the lower bound on the cost of node 2, which is calculated as $\underline{c_2} = \frac{e_1+e_3}{N_{p,max}T_s}$, with the incumbent minimal cost $\bar{c}$. If $\underline{c_2} \geq \bar{c}$ this node will be pruned from the search tree and will not be considered for further branching. Otherwise, it will be considered for further branching, in the case extension, to produce child node(s). Similar argument holds for node 3 (dash-dotted-line). Having computed the first switching losses e_4, this node and its descendants are discarded if $\underline{c_4} = \frac{e_4}{N_{p,max}T_s} \geq \bar{c}$.

5.2 Ingredients of Branch and Bound Algorithm

There are two major ingredients to the branch and bound algorithm. (i) A function to calculate the lower bound on the cost of the bud nodes, which is referred to as the *bounding function.* (ii) *Branching Strategy*, which determines which bud node is first to be considered for further branching.

5.2.1 Bounding Function

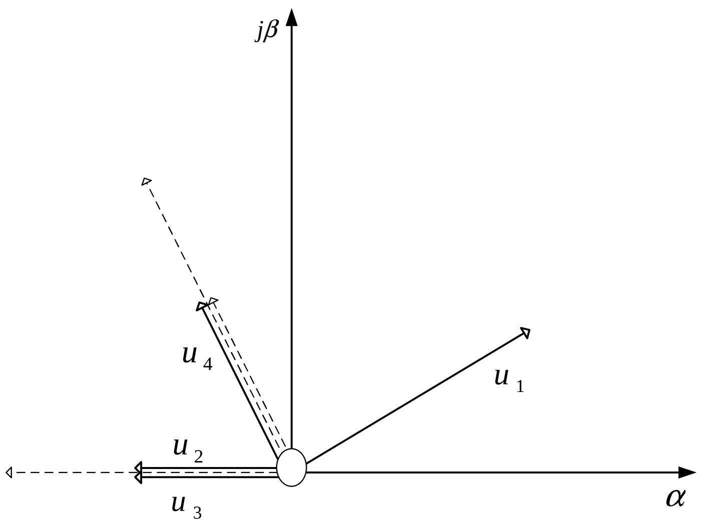
In the context of branch and bound, the bounding function estimates the minimum value of the cost obtainable by a bud node when it is further grown. It is important that the bounding function underestimates the actual minimum cost achievable by growing a bud node, yet be as accurate estimator as possible [16].

For the branch and bound algorithm tailored to the MPDTC problem, finding the lower bound on the cost associated to a bud node, $\underline{c_i}$, reduces to estimating the maximal length of the prediction horizon $N_{p,max}$. Unfortunately, it is not straightforward to estimate the maximal length of the prediction horizon for an incomplete switching sequence. One approach is to assign a fixed $N_{p,max}$ to all bud nodes in the search tree. The value of $N_{p,max}$ for the current optimization problem is set equal to the maximal length of the prediction horizon obtained during the last optimization problem. The other approach is to group the nodes and assign a $N_{p,max}$ to each group. We shall introduce this approach by an example.

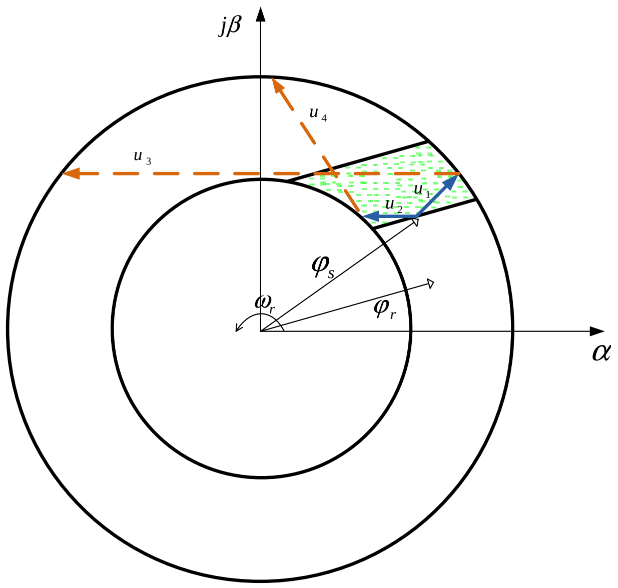
Example 2: For the search tree generated by the switching horizon 'SESE', the maximal length of the prediction horizon is estimated as follows. For the first 'S' event, the algorithm looks for all admissible switch position at time-step k and finds two of them, namely $u_1$and u_2 as shown in Fig. 5.3a. For the first 'E' event, the number of time-steps for which the output trajectories (i.e. the torque, the stator flux and inverter's neutral point potential) can be extended using u_1 and u_2 are analytically calculated as in Chapter 3 and referred to as $N^1_{p1}$and N^2_{p1} , respectively. In Fig. 5.3b, the evolution of the stator flux vector during this step is shown as (blue) solid vectors in $\alpha\beta$ plane. For the second 'S' event, the switch positions u_3 and u_4, whose corresponding voltage vectors have the shortest length among those pointing in the tangent direction to the lower bound on the stator flux magnitude, are chosen. For the second 'E' event, the number of time steps for which the stator flux magnitude can be extended using u_3 and u_4 before hitting its upper bound are calculated and referred to as N^1_{p2} and N^2_{p2}, respectively. Fig. 5.3b shows the evolution of the stator flux vector during this step as (orange) dashed vectors. The maximal length of the prediction horizon for all switching sequences starting with u_i is given by $N^i_{p,max} = N^i_{p1} + N^i_{p2}$, i $= 1, 2$.

More formally, the algorithm for estimating $N_{p,max}$ works as following. Regroup consecutive 'S' events in the switching horizon into a single 'S' event. For instance, the switching horizon 'SSESE' becomes 'SESE'. For the first 'S' event followed by the 'E' event, calculate the number of time-steps for which the output trajectories can be extended using the i-th admissible switch position at time-step k and denote it as N^i_{p1}. For the j-th ($j \geq 2$) 'S' event followed by the 'E' event, choose the set of voltage vectors with the shortest length in $\alpha\beta$ plane that are tangent to the lower bound on the stator flux magnitude (i.e. the inner circle in Fig. 5.3b). If this set is empty, find the closest voltage vectors to the tangent vector. This ensures that the stator flux magnitude will hit its upper bound as late as possible.[2]. Ignore the bounds on the torque and calculate the number of time-steps for which the stator flux magnitude stays within its bounds using this switch position and denote it as N^i_{pj}. The maximal length of the prediction horizon for the switching sequences starting with the i-th switch position is given by

[2]In fact, it will not hit the lower bound at all.



(a) The voltage vectors of a 3-level inverter corresponding to switch positions in Example 2. Among voltage vectors pointing in the tangent direction to the lower bound on the length of the stator flux, the one with the shortest length is chosen.



(b) Evolution of the stator flux vector in $\alpha - \beta$ plane explained in Example 2. The feasible region (shaded) is bounded by the the upper (outer circle) and lower (inner circle) bounds on the length of the stator flux and the bounds on the torque (two parallel lines).

Figure 5.3: Estimating the maximal length of the prediction horizon

$$N^i_{p,max} = N^i_{p1} + \sum_{j=2}^{m} N^i_{pj} \tag{5.1}$$

where m denotes the number of 'S' events in the regrouped switching horizon.

A few remarks about the described algorithm are to be mentioned. Rather than having a fixed $N_{p,max}$ for all switching sequences, the switching sequences starting with identical switch position are assigned a certain $N_{p,max}$. Thus, there will be different $N_{p,max}$(s) to the number of the admissible switching transitions at time-step k. The first part of $N_{p,max}$ (i.e., N_{p1}) may underestimate the actual length of the prediction horizon whereas the remaining parts, overestimate the actual length of prediction horizon by ignoring some constraints. At the end, we hope that $N_{p,max}$ provides a rather tight upper bound on the length of the prediction horizon.

5.2.2 Branching Strategy

Any bud node with a $\underline{c}$ smaller than the incumbent cost, $\bar{c}$, is a possible candidate for branching. A policy is needed to choose among the candidates, which may be quite large in number. Two such policies are

- Best-first or global-best node: the bud node with the lowest cost anywhere on the search tree is selected for branching. For long switching horizons (e.g., 'SSESE') it may take many iterations (i.e., nodes to visit) to reach an incumbent node making us unable to prune the search tree during early iterations.
- Depth-first: the bud node with the lowest cost among the set of bud nodes that are just created is selected for branching. Depth-first branch and bound has the advantage that it takes us one level deeper into the search tree at each iteration, so it is more likely to reach an incumbent node more quickly. This allows us to start pruning at early iterations if the incumbent solution is good enough. If it is not possible to proceed any deeper into the tree, we will back up one level and choose another bud node from that level. However, it may take more nodes to visit (i.e. iterations) to find the optimal switching sequence with the associated cost c^*.

5.2.3 Warm Start

Getting a good incumbent solution early is extremely useful in pruning as many bud nodes will never be created if the lower bound on their costs is higher than the incumbent minimal cost. Particularly, we will consider generating a complete candidate switching sequence even before beginning the branch and bound process. A simple rule base was designed based on which an

Switch position	$u_1 = [1, 1, 0]$	$u_2 = [0, 0, -1]$	$u_3 = [1, 1, -1]$	$u_4 = [0, 1, -1]$
dU	2	1	3	3

Table 5.1: The number of switching transitions associated with admissible switch positions at point A, which can increase the toque and length of the stator flux. The last switch position is $u_0 = [0, 0, 0]$.

Switch position	$u_5 = [0, 1, 1]$	$u_6 = [-1, 0, 0]$
dU	2	1

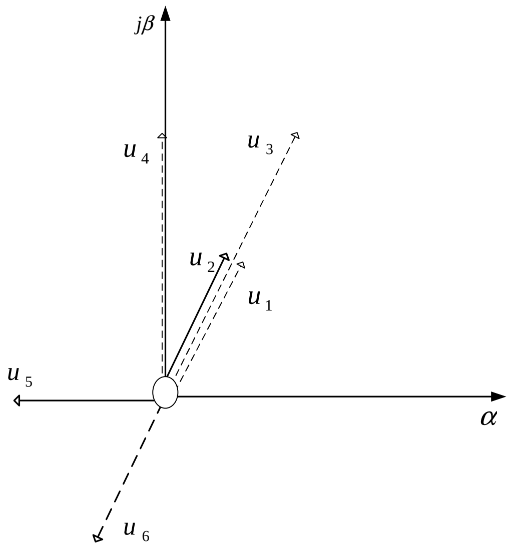
Table 5.2: The number of switching transitions associated with admissible switch positions at point B, which can decrease the toque and length of the stator flux. The last switch position is $u_0 = [0, 0, -1]$.

incumbent solution is be generated at the beginning of optimization process. Hopefully, this solution will be good enough to allow us to prune some nodes in very early iterations. This idea is introduced by an example.

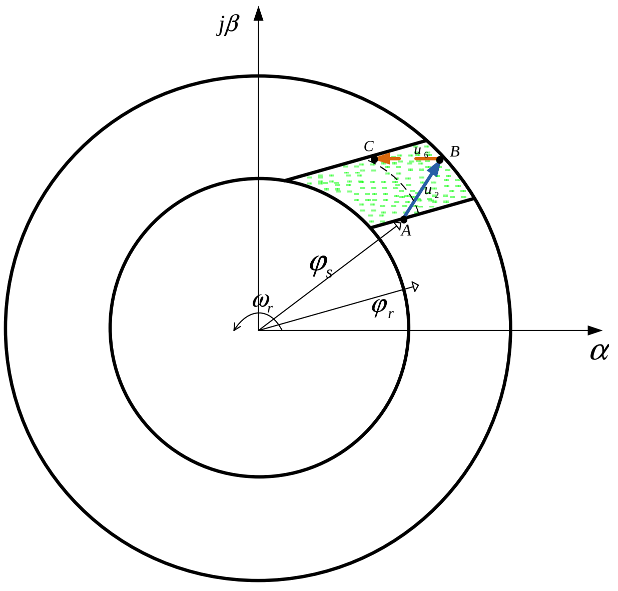
Example 3: Fig. 5.4b shows the state of the machine using a 3-level inverter and the switching horizon 'SESE' at some point during the simulations. Initially, we are at point A. With the last switch position being $u_0 = [\begin{array}{ccc} 0 & 0 & 0 \end{array}]^{\mathrm{T}}$, the torque has hit its lower bound and the stator flux magnitude is below it's mid-level. Thus, the set of admissible switch positions increasing the torque and length of the stator flux are identified as in Table 5.1. Out of this set, the switch position inducing the minimum number of switching transitions (i.e, u_2) is selected, which moves the stator flux as shown by (blue) solid line. The number of time-steps for which this switch position can be applied until we reach point B is calculated analytically and referred to as N_{p1}. As the length of the stator flux has hit its upper bound and the torque is above its mid-level in point B, the switch positions increasing the torque and decreasing the length of the stator flux will probably yield a long prediction horizon. Table 5.2 lists such switch positions and the associated switching transitions. Once again, the switch position with minimum number of switching transitions (i.e., u_6) is chosen, which moves the stator flux vector as shown by (orange) dashed line. The number of time-steps until we reach point C is calculated and referred to as N_{p2}. The complete switching sequence is given by

$$U(k) = \left[\underbrace{u_2, \ldots, u_2}_{N_{p1}\,\text{times}} \;, \underbrace{u_6, \ldots, u_6}_{N_{p2}\,\text{times}} \right] \tag{5.2}$$

More formally, the admissible switching positions of the inverter are divided into five categories, namely, increasing T_e, decreasing T_e, increasing Ψ_s, decreasing Ψ_s and constant Ψ_s. Depending on the state of the machine's outputs (i.e., the torque and length of the stator flux) a switch position is chosen for each 'S' event in the switching horizon according to Table 5.3 where the pair $(+, +)$ means switch positions that belong to the intersection of the sets 'increasing T_e' and 'increasing Ψ_s', and $(+, \star)$ denotes the switch positions that belong to the set 'increasing T_e'.



(a) The voltage vectors of a 3-level inverter corresponding to switch positions in Example 3.



(b) The incumbent switching sequence that moves the stator flux vector from point A to point C for switching horizon 'SESE'. The feasible region (shaded) is bounded by the upper (outer circle) and lower (inner circle) bounds on the length of the stator flux and the bounds on the torque (two parallel lines).

Figure 5.4: Finding an incumbent switching sequence for the switching horizon 'SESE'

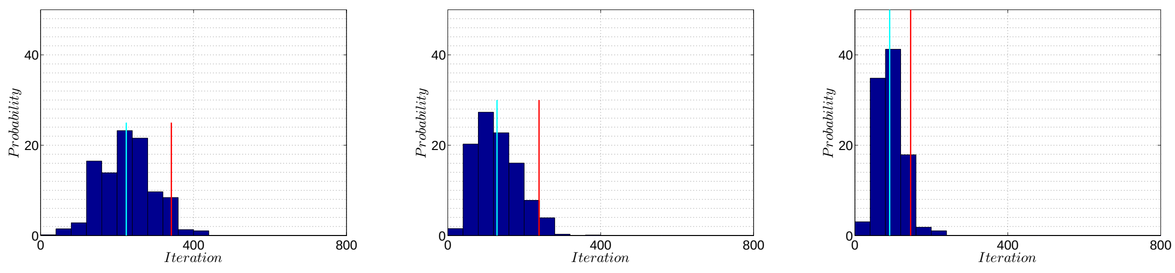
(T_e, Ψ_s)	$T_e = T_{e,max}$	$T_e = T_{e,min}$
$\Psi_s < \Psi_{s,ref}$	$(-,0), (-,+)$	$(+,+), (+,\star)$
$\Psi_s > \Psi_{s,ref}$	$(-,0), (-,-)$	$(+,-), (+,\star)$

(a) The set of switch positions when the torque is about to hit its bounds, where $\Psi_{s,ref} = \frac{\Psi_{s,max}+\Psi_{s,min}}{2}$.

(T_e, Ψ_s)	$\Psi_s = \Psi_{s,max}$	$\Psi_s = \Psi_{s,min}$
$T_e < T_{e,ref}$	$(+,-), (\star,-)$	$(+,+), (\star,+)$
$T_e > T_{e,ref}$	$(-,-), (\star,-)$	$(-,+), (\star,+)$

(b) The set of switch positions when the magnitude of the stator flux is about to hit its bounds, where $T_{e,ref} = \frac{T_{e,max}+T_{e,min}}{2}$.

Table 5.3: The logic to generate an incumbent solution. The first pair in each column is the first preference. The second pair is chosen if the first pair returns an empty set.



(a) No pruning **(b)** Fixed $N_{p,max}$ **(c)** Variable $N_{p,max}$

Figure 5.5: Histograms of nodes visited to obtain optimal $u(k)$ for the switching horizon 'SESE' at $T_e = 1$ pu, $\omega_e = 0.5$ pu where the mean and 95% probability points are specified.

5.3 Performance Evaluation

The performance of MPDTC algorithm with full enumeration is compared with MPDTC algorithm using branch and bound. Fig. 5.5 compare the histogram of iterations associated with full enumeration, fixed $N_{p,max}$ and variable $N_{p,max}$ methods. It is clear that the last method reduces the number of iterations effectively. Table. 5.4 summarizes the closed-loop simulation results.

Table 5.4: Closed-loop simulation results when minimizing the switching frequency. The table shows the average inverter switching frequency (Hz), the torque's total harmonic distortion (THD) at 100% torque.

Op. Point [pu]	Controller settings				Pred. horizon		Nodes explored		Performance		
ω_e	Sw. horizon	Node Sel.	$N_{p,max}$	Warm start	avg.	max.	avg.	max.	Sw. Freq	I_{THD}	$T_{e,THD}$
0.5	SESE	—	—	—	45.59	105	224	422	172	7.84	6.32
0.5	SESE	best-first	fixed	no	44.68	105	135	240	172	7.84	6.32
0.5	SESE	best-first	variable	no	39	103	91	145	172	7.63	6.12
0.5	SESE	best-first	variable	yes	38	103	89	284	173	7.53	6.06
0.5	SESE	depth-first	variable	no	38	103	88	302	168	7.73	6.18

Chapter 6

Conclusions and Future Work

6.1 Discussion

The trajectory extension methods were introduced in Chapter 3. The linear extrapolation method yields the least accurate predictions according to the histograms obtained in Chapter 4 but it is the cheapest one in terms of the computational complexity. For the switching horizon 'SE' at low speeds, the LE method seems to be a reasonable choice as the closed-loop performance of LE method is not significantly different from the performance of the open-loop simulation method while the associated computational burden is significantly reduced. For the switching horizon 'SE' at high speeds, the quadratic extrapolation, quadratic interpolation and analytical methods all exhibit an acceptable closed-loop performance with a reasonable number of basic calculations. With the same computational burden, the quadratic interpolation approach outperforms the quadratic extrapolation. The analytical method is computationally cheaper than quadratic interpolation at the cost of adding one more tuning parameter t_p.

As discussed in Chapter 4, the linear extrapolation and quadratic extrapolation methods are not applicable for long switching horizons. The analytical method can be used for long prediction horizons if the output polynomials are approximated about the right value of t_p, which depends on the operating point of the machine and the switching horizon. Based on the histograms of the relative error of the prediction horizon length, the prediction with quadratic interpolation outperforms the analytical method at the cost of a slightly increased computational burden. The iterative prediction with quadratic interpolation method yields the most accurate predictions as its histograms are highly concentrated about zero and its closed-loop performance is very similar to the performance of the open-loop simulation method. The disadvantage is that the associated computational burden increases with the length of the prediction horizon. The QII2 method performs equally well but its associated computational burden is higher than the one of the QII method for the nominal length of the prediction horizon. Therefore, for long switching horizons the QI and QII methods are preferred.

In Chapter 5, a method is proposed to estimate the maximal length of the prediction horizon which is required for obtaining the upper bound on the cost of each switching sequence. It is shown, via simulations, that the branch and bound algorithm employing such bounding function can significantly reduce the number of iterations required to find the optimal switching sequence. Two branching strategies, namely best-first and depth-first, were considered. No significant difference was observed between the associated number of iterations required to obtain the optimal switching sequence. The warm start concept was introduced in Chapter 5, where a simple switching logic was used to generate a complete candidate switching sequence before enumerating the branch and bound tree. However, as the proposed switching logic was very simple, the generated candidate switching sequences were far from the optimal. Consequently, the proposed approach was not effective in reducing the number of nodes required to obtain the optimal switching sequence.

6.2 Future work

The major disadvantage of the analytical method presented in Chapter 3 is that for a particular switching horizon and operating point of the machine, the correct value of t_p needs to be identified. Alternatively, instead of approximating the output equations about $t = t_p$ using the Taylor series, one can use iterative root finding algorithms such as the Newton-Raphson algorithm to solve for the length of the prediction horizon and compare. Then the accuracy of this method and its associated computational complexity should be compared with the ones of the open-loop simulation approach.

The proposed switching logic for initializing the branch and bound tree was too simple to make a significant difference. Alternatively, the DTC switching tables along with the analytical method can be used to find a 'near-optimal' incumbent switching sequence to initialize the branch and bound tree. If this initial switching sequence is good enough, we may start pruning the search tree at early iterations and thus reduce the total number of iterations required to obtain the optimal switching sequence.

In addition to using best-first or depth-first node selection strategies, one can use branching heuristics to branch on the most promising nodes first. One possible approach would be as following. At every switching event, the switch position which incurs the minimum cost according to the 'cost to go' maps of Fig. 2.10 is chosen for further branching among all possible switch positions. For this purpose, the algorithm must identify the output bound that is about to be violated[1], and determine the rotor angle θ_r and value of x as in Fig. 2.8b. Then the the plots of Fig. 2.10, can be used to find the switch position associated with the minimum cost. A few remarks are to be mentioned about the proposed method. The plots in Fig. 2.10 are obtained

[1]Note that we assume the switching event happens only when an output variable is about to hit its bounds

for $\theta_r \in [0, \frac{\pi}{3})$. If the rotor flux lies in any other region in $\alpha\beta$ plane, we can exploit the two important properties of the drive system, namely symmetry of voltage vectors and invariance of motor outputs under flux rotations [13], to map the flux vectors into $[0, \frac{\pi}{3})$ region, to obtain the switch position associated with the minimum cost using plots in Fig. 2.10, and to map the result back into the original region to obtain the actual switch position for further branching. Details on how to map the flux vectors can be found in [13].

Appendix A

Details on Derivations and Approximations

A.1 Approximating $(sI - A)^{-1}$

The A matrix is given by

$$A = \begin{bmatrix} -a & 0 & b & 0 \\ 0 & -a & 0 & b \\ c & 0 & -f & -\omega_r \\ 0 & c & \omega_r & -f \end{bmatrix} \tag{A.1}$$

with $a = r_s \frac{x_{rr}}{D}$, $b = r_s \frac{x_m}{D}$, $c = r_r \frac{x_m}{D}$ and $f = r_r \frac{x_{ss}}{D}$, where r_s, r_r, x_{rr}, x_m, x_{ss} and D are the motor parameters. Thus, we will have

$$sI - A = \left[\begin{array}{cc|cc} \multicolumn{2}{c|}{\overbrace{\hphantom{s+a \quad s+a}}^{W}} & \multicolumn{2}{c}{\overbrace{\hphantom{s+f \quad s+f}}^{X}} \\ s+a & 0 & -b & 0 \\ 0 & s+a & 0 & -b \\ \hline -c & 0 & s+f & \omega_r \\ 0 & -c & -\omega_r & s+f \\ \multicolumn{2}{c|}{\underbrace{\hphantom{s+a \quad s+a}}_{Y}} & \multicolumn{2}{c}{\underbrace{\hphantom{s+f \quad s+f}}_{Z}} \end{array}\right] \tag{A.2}$$

Using Sherman-Morrison inversion lemma [15], the inverse of (A.2) is

$$\begin{aligned}(sI-A)^{-1} &= \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}^{-1} \\ &= \left[\begin{array}{c|c} W^{-1}+W^{-1}X\left(Z-YW^{-1}X\right)^{-1}YW^{-1} & -W^{-1}X\left(Z-YW^{-1}X\right)^{-1} \\ \hline -(Z-YW^{-1}X)YW^{-1} & \left(Z-YW^{-1}X\right)^{-1} \end{array}\right]\end{aligned} \tag{A.3}$$

Let Δ denote the first block in (A.3) (i.e., $W^{-1}+W^{-1}X\left(Z-YW^{-1}X\right)^{-1}YW^{-1}$). We shall show the steps of simplifying Δ. The other blocks are simplified in the same manner.

$$\begin{aligned}\Delta &= \frac{1}{(s+f-\frac{bc}{s+a})^2+\omega_r^2}\begin{bmatrix} (s+f-\frac{bc}{s+a})\frac{bc}{(s+a)^2} & -\frac{bc\omega_r}{(s+a)^2} \\ \frac{bc\omega_r}{(s+a)^2} & (s+f-\frac{bc}{s+a})\frac{bc}{(s+a)^2} \end{bmatrix} \\ &+ \frac{1}{s+1}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\end{aligned} \tag{A.4}$$

With $s >> 0$ and f being an order of magnitude greater than bc , $\frac{bc}{s+a}$ becomes negligible when compared to $s+f$ and can be omitted. Thus, equation (A.4) becomes

$$\begin{aligned}\Delta &= \frac{1}{(s+f)^2+\omega_r^2}\begin{bmatrix} \frac{bc(s+f)}{(s+a)^2}+\frac{(s+f)^2+\omega_r^2}{s+a} & -\frac{bc\omega_r}{(s+a)^2} \\ \frac{bc\omega_r}{(s+a)^2} & \frac{bc(s+f)}{(s+a)^2}+\frac{(s+f)^2+\omega_r^2}{s+a} \end{bmatrix} \\ &\overset{a\approx f}{=} \frac{1}{(s+f)^2+\omega_r^2}\begin{bmatrix} \frac{bc}{(s+a)}+\frac{(s+f)^2+\omega_r^2}{s+a} & -\frac{bc\omega_r}{(s+a)^2} \\ \frac{bc\omega_r}{(s+a)^2} & \frac{bc}{(s+a)}+\frac{(s+f)^2+\omega_r^2}{s+a} \end{bmatrix}\end{aligned} \tag{A.5}$$

As $\frac{bc}{(s+a)^2}$ is orders of magnitude smaller than $\frac{(s+f)^2+\omega_r^2}{s+a}$, (A.5) becomes

$$\Delta = \frac{1}{(s+f)^2+\omega_r^2}\begin{bmatrix} \frac{(s+f)^2+\omega_r^2}{s+a} & -\frac{bc\omega_r}{(s+a)^2} \\ \frac{bc\omega_r}{(s+a)^2} & \frac{(s+f)^2+\omega_r^2}{s+a} \end{bmatrix}$$

A.2 Coefficients of Analytic Solutions of the State Equations in Chapter 3.

The solution of system equations in Laplace domain is given by

$$\boldsymbol{X}(s) = (sI-A)^{-1}\boldsymbol{x}_0 + G(s)\boldsymbol{U}(s) \tag{A.6}$$

with

$$(sI-A)^{-1} \simeq \frac{1}{(s+f)^2+\omega_r^2}\begin{bmatrix} \frac{(s+f)^2+\omega_r^2}{s+a} & -bc\frac{\omega_r}{(s+a)^2} & b\frac{s+f}{s+a} & -b\frac{\omega_r}{s+a} \\ bc\frac{\omega_r}{(s+a)^2} & \frac{(s+f)^2+\omega_r^2}{s+a} & b\frac{\omega_r}{s+a} & b\frac{s+f}{s+a} \\ c\frac{s+f}{s+a} & -c\frac{\omega_r}{s+a} & s+f & -\omega_r \\ c\frac{\omega_r}{s+a} & c\frac{s+f}{s+a} & \omega_r & s+f \end{bmatrix} \tag{A.7}$$

$$G(s) = \frac{1}{(s+f)^2+\omega_r^2}\begin{bmatrix} \frac{(s+f)^2+\omega_r^2}{s+a} & -bc\frac{\omega_r}{(s+a)^2} \\ bc\frac{\omega_r}{(s+a)^2} & \frac{(s+f)^2+\omega_r^2}{s+a} \\ c\frac{s+f}{s+a} & -c\frac{\omega_r}{s+a} \\ c\frac{\omega_r}{s+a} & c\frac{s+f}{s+a} \end{bmatrix} \tag{A.8}$$

$$\boldsymbol{x}(0) = \begin{bmatrix} x_{01} & x_{02} & x_{03} & x_{04} \end{bmatrix}^{\mathrm{T}} \tag{A.9}$$

$$\boldsymbol{U}(s) = \begin{bmatrix} v_\alpha & v_\beta \end{bmatrix}^{\mathrm{T}} \tag{A.10}$$

We shall present the steps for deriving the complete solutions of $X_1(s)$ and $X_3(s)$ in Chapter 3. The solutions of $X_2(s)$ and $X_4(s)$ are obtained in similar manner.

$$\begin{aligned} X_1(s) &= \frac{1}{(s+a)}x_{01} + \frac{-bc\omega_r}{[(s+f)^2+\omega_r^2](s+a)^2}x_{02} \\ &\quad + \frac{-b\omega_r}{[(s+f)^2+\omega_r^2](s+a)}x_{04} + \frac{b(s+f)}{[(s+f)^2+\omega_r^2](s+a)}x_{03} \\ &\quad + \frac{1}{s}\frac{1}{(s+a)}v_\alpha + \frac{1}{s}\frac{-bc\omega_r}{[(s+f)^2+\omega_r^2](s+a)^2}v_\beta \end{aligned} \tag{A.11}$$

$$\begin{aligned} X_3(s) &= \frac{c(s+f)}{[(s+f)^2+\omega_r^2](s+a)}x_{01} + \frac{-c\omega_r}{[(s+f)^2+\omega_r^2](s+a)}x_{02} \\ &\quad + \frac{s+f}{(s+f)^2+\omega_r^2}x_{03} + \frac{-\omega_r}{(s+f)^2+\omega_r^2}x_{04} \\ &\quad + \frac{1}{s}\frac{c(s+f)}{[(s+f)^2+\omega_r^2](s+a)}v_\alpha + \frac{1}{s}\frac{-c\omega_r}{[(s+f)^2+\omega_r^2](s+a)}v_\beta \end{aligned} \tag{A.12}$$

Using the partial fractions expansions of Table A.1, we can write $X_1(s)$ and $X_2(s)$ as

$$\begin{aligned} X_1(s) &= (l_1 v_\alpha + l_2 v_\beta)\,\frac{1}{s} + (k_2 x_{02} + l_4 v_\beta)\,\frac{1}{(s+a)^2} \\ &\quad (x_{01} + k_1 x_{02} + k_4 x_{03} - k_5 x_{04} - l_1 v_\alpha + l_3 v_\beta)\,\frac{1}{s+a} \\ &\quad + (k_3 x_{02} + k_5 x_{03} + k_4 x_{04} + l_5 v_\beta)\,\frac{\omega_r}{(s+f)^2 + \omega_r^2} \\ &\quad + (-k_1 x_{02} - k_4 x_{03} + k_5 x_{04} + l_6 v_\beta)\,\frac{s+f}{(s+f)^2 + \omega_r^2} \end{aligned} \tag{A.13}$$

$$\begin{aligned} X_3(s) &= (p_1 v_\alpha + p_2 v_\beta)\,\frac{1}{s} + (m_1 x_{01} - m_2 x_{02} + p_3 v_\alpha + p_4 v_\beta)\,\frac{1}{s+a} \\ &\quad + (m_2 x_{01} + m_1 x_{02} - x_{04} + p_5 v_\alpha + p_6 v_\beta)\,\frac{\omega_r}{(s+f)^2 + \omega_r^2} \\ &\quad + (-m_1 x_{01} + m_2 x_{02} + x_{03} - p_6 v_\alpha + p_5 v_\beta)\,\frac{s+f}{(s+f)^2 + \omega_r^2} \end{aligned} \tag{A.14}$$

where the coefficients p_i, m_i, k_i and l_i are

$$l_1 = \frac{1}{a}$$

$$l_2 = -bc\omega_r \left(\frac{\frac{1}{a^2}}{(f-a)^2+\omega_r^2} + \frac{\frac{3f^2-4af+a^2+\omega_r^2}{f^2+\omega_r^2} - \frac{2(f-a)}{a}}{[(f-a)^2+\omega_r^2]^2} \right) \overset{a\approx f}{\approx} -bc\left(\frac{\frac{1}{a^2}}{\omega_r} + \frac{1}{\omega_r^3}\right) \approx -\frac{bc}{a^2}\frac{1}{\omega_r}$$

$$l_3 = -bc\omega_r \left(\frac{\frac{2(f-a)}{a}}{[(f-a)^2+\omega_r^2]^2} - \frac{\frac{1}{a^2}}{(f-a)^2+\omega_r^2} \right) \approx \frac{bc}{a^2}\frac{1}{\omega_r}$$

$$l_4 = bc\omega_r \frac{\frac{1}{a}}{(f-a)^2+\omega_r^2} \approx \frac{bc}{a}\frac{1}{\omega_r}$$

$$l_5 = bc\frac{\frac{3f^2-4af+a^2+\omega_r^2}{f^2+\omega_r^2} - 2(f-a)}{[(f-a)^2+\omega_r^2]^2} \approx \frac{bc}{\omega_r^4} \approx 0$$

$$l_6 = bc\omega_r \frac{\frac{3f^2-4af+a^2+\omega_r^2}{f^2+\omega_r^2}}{[(f-a)^2+\omega_r^2]^2} \approx \frac{bc}{\omega_r^3} \approx 0$$

$$k_1 = bc\omega_r \frac{2(f-a)}{[(f-a)^2+\omega_r^2]^2} \approx 0$$

$$k_2 = -bc\omega_r \frac{1}{(f-a)^2+\omega_r^2} \approx -\frac{bc}{\omega_r} \approx 0$$

$$k_3 = -bc\frac{f^2-2af+a^2-\omega_r^2}{[(f-a)^2+\omega_r^2]^2} \approx \frac{bc}{\omega_r^2} \approx 0$$

$$k_4 = b\frac{f-a}{(f-a)^2+\omega_r^2} \approx 0$$

$$k_5 = b\omega_r \frac{1}{(f-a)^2+\omega_r^2} \approx \frac{b}{\omega_r}$$

$$p_1 = c\frac{\frac{f-a}{a} + \frac{\omega_r^2 - f(f-a)}{f^2+\omega_r^2}}{(f-a)^2+\omega_r^2} \approx \frac{c}{\omega_r^2}$$

$$p_2 = -c\omega_r \frac{\frac{1}{a} + \frac{a-2f}{f^2+\omega_r^2}}{(f-a)^2+\omega_r^2} \approx -\frac{c}{\omega_r}\left(\frac{1}{a} - \frac{a}{\omega_r^2}\right) \approx -\frac{1}{a}\frac{c}{\omega_r}$$

$$p_3 = c\frac{\frac{a-f}{a}}{(f-a)^2+\omega_r^2} \approx 0$$

$$p_4 = c\omega_r \frac{\frac{1}{a}}{(f-a)^2+\omega_r^2} \approx \frac{1}{a}\frac{c}{\omega_r}$$

$$p_5 = c\frac{\frac{a-f}{\omega_r} - \frac{\omega_r^2 - f(f-a)}{f^2+\omega_r^2}\frac{f}{\omega_r}}{(f-a)^2+\omega_r^2} \approx -\frac{cf}{\omega_r^3} \approx 0$$

$$p_6 = c\frac{1 + \frac{f(a-2f)}{f^2+\omega_r^2}}{(f-a)^2+\omega_r^2} \approx \frac{c}{\omega_r^2}$$

$$m_1 = c\frac{f-a}{(f-a)^2+\omega_r^2} \approx 0$$

$$m_2 = c\frac{\omega_r}{(f-a)^2+\omega_r^2} \approx \frac{c}{\omega_r}$$

The equations for $x_1(t)$ and $x_3(t)$ in Chapter 3 are simply obtained by taking the inverse Laplace transform of (A.13) and (A.14) using Table A.2.

Table A.1: Partial fraction expansions

fraction	partial expansion
$\frac{1}{[(s+f)^2+\omega_r^2](s+a)^2}$	$\frac{\frac{1}{(f-a)^2+\omega_r^2}}{(s+a)^2} + \frac{\frac{-2(f-a)}{[(f-a)^2+\omega_r^2]^2}}{s+a} + \frac{\frac{2(f-a)}{[(f-a)^2+\omega_r^2]^2}s+\frac{3f^2-4af+a^2-\omega_r^2}{[(f-a)^2+\omega_r^2]^2}}{(s+f)+\omega_r^2}$
$\frac{s+f}{[(s+f)^2+\omega_r^2](s+a)}$	$\frac{\frac{f-a}{(f-a)^2+\omega_r^2}}{s+a} + \frac{-\frac{f-a}{(f-a)^2+\omega_r^2}s+\frac{\omega_r^2-f(f-a)}{(f-a)^2+\omega_r^2}}{(s+f)^2+\omega_r^2}$
$\frac{1}{[(s+f)^2+\omega_r^2](s+a)}$	$\frac{\frac{1}{(f-a)^2+\omega_r^2}}{s+a} + \frac{-\frac{1}{(f-a)^2+\omega_r^2}s+\frac{a-2f}{(f-a)^2+\omega_r^2}}{(s+f)^2+\omega_r^2}$
$\frac{1}{s(s+a)}$	$\frac{\frac{1}{a}}{s} + \frac{-\frac{1}{a}}{s+a}$
$\frac{1}{s(s+a)^2}$	$\frac{\frac{1}{a^2}}{s} + \frac{-\frac{1}{a^2}}{s+a} + \frac{\frac{-1}{a}}{(s+a)^2}$
$\frac{1}{s[(s+f)^2+\omega_r^2]}$	$\frac{\frac{1}{f^2+\omega_r^2}}{s} + \frac{\frac{-1}{f^2+\omega_r^2}s+\frac{-2f}{f^2+\omega_r^2}}{(s+f)^2+\omega_r^2}$

Table A.2: Laplace transform pairs

Laplace domain	time domain ($t \geq 0$)
$\frac{1}{s}$	1
$\frac{1}{s+a}$	e^{-at}
$\frac{1}{(s+a)^2}$	$t\mathrm{e}^{-at}$
$\frac{\omega_r}{(s+f)^2+\omega_r^2}$	$\mathrm{e}^{-ft}\sin(\omega_r t)$
$\frac{s+f}{(s+f)^2+\omega_r^2}$	$\mathrm{e}^{-ft}\cos(\omega_r t)$

Table A.3: First order Taylor expansion of $F_i(t)$ about the nominal point $t = t_0$, $F_i(t) \approx p_i t + q_i$

$F_1(t)$	e^{-at}	p_1	$-a\mathrm{e}^{-at_0}$	q_1	$\mathrm{e}^{-at_0}(1+at_0)$
$F_2(t)$	e^{-ft}	p_2	$-f\mathrm{e}^{-ft_0}$	q_2	$\mathrm{e}^{-ft_0}(1+ft_0)$
$F_3(t)$	$\mathrm{e}^{-ft}\cos(\omega_r t)$	p_3	$-\mathrm{e}^{-ft_0}\left(f\cos(\omega_r t_0)+\omega_r\sin(\omega_r t_0)\right)$	q_3	$\mathrm{e}^{-ft_0}\left(\cos(\omega_r t_0)+ft_0\cos(\omega_r t_0)+\omega_r t_0\sin(\omega_r t_0)\right)$
$F_4(t)$	$\mathrm{e}^{-ft}\sin(\omega_r t)$	p_4	$\mathrm{e}^{-ft_0}\left(\omega_r\cos(\omega_r t_0)-f\sin(\omega_r t_0)\right)$	q_4	$\mathrm{e}^{-ft_0}\left(\sin(\omega_r t_0)+ft_0\sin(\omega_r t_0)-\omega_r t_0\cos(\omega_r t_0)\right)$

A.3 Affine Approximation of the Stator Flux Components in Chapter 3

s_i and r_i, are given by

$$\begin{aligned} s_1 &= p_1 x_{01} + k_5 p_3 x_{03} + k_5(p_4 - p_2)x_{04} - \frac{p_1}{a} v_\alpha \\ r_1 &= q_1 x_{01} + k_5 q_3 x_{03} + k_5(q_4 - q_2)x_{04} + \frac{1-q_1}{a} v_\alpha \end{aligned} \tag{A.15}$$

$$\begin{aligned} s_2 &= p_1 x_{02} + k_5 p_3 x_{04} - k_5(p_4 - p_2)x_{03} - \frac{p_1}{a} v_\beta \\ r_2 &= q_1 x_{02} + k_5 q_3 x_{04} - k_5(q_4 - q_2)x_{03} + \frac{1-q_1}{a} v_\beta \end{aligned} \tag{A.16}$$

where p_i and q_i are the first order Taylor expansions obtained about the nominal point $t = t_0$, which are given in Table A.3.

A.4 Quadratic Approximation of the Rotor Flux Components in Chapter 3

The coefficients a_i, b_i and c_i are obtained by second order Taylor expansion of $\Psi_r \cos(\omega_r t + \Theta)$ and $\Psi_r \sin(\omega_r t + \Theta)$ about $t = t_0$.

$$\begin{aligned} a_1 &= \Psi_r \left(-\frac{1}{2}\omega_r^2 \cos(\omega_r t_0 + \Theta) \right) \\ b_1 &= \Psi_r \left(t_0 \omega_r^2 \cos(\omega_r t_0 + \Theta) - \omega_r \sin(\omega_r t_0 + \Theta) \right) \\ c_1 &= \Psi_r \left(\cos(\omega_r t_0 + \Theta) + \omega_r t_0 \sin(\omega_r t_0 + \Theta) - \frac{1}{2} t_0^2 \omega_r^2 \cos(\omega_r t_0 + \Theta) \right) \end{aligned} \tag{A.17}$$

$$\begin{aligned} a_2 &= \Psi_r \left(-\frac{1}{2}\omega_r^2 \sin(\omega_r t_0 + \Theta) \right) \\ b_2 &= \Psi_r \left(\omega_r \cos(\omega_r t_0 + \Theta) + t_0 \omega_r^2 \sin(\omega_r t_0 + \Theta) \right) \\ c_2 &= \Psi_r \left(\sin(\omega_r t_0 + \Theta) - \omega_r t_0 \cos(\omega_r t_0 + \Theta) - \frac{1}{2} t_0^2 \omega_r^2 \sin(\omega_r t_0 + \Theta) \right) \end{aligned} \tag{A.18}$$

A.5 Quadratic Approximation of the Neutral Point Potential in Chapter 3

The coefficients of the second order polynomial describing the neutral point potential is given by

$$\begin{aligned} u_1 &= \frac{\gamma_1}{D}\left(\frac{x_{rr}}{2}s_1 - \frac{x_m}{\omega_r}a_2\right) + \frac{\gamma_2}{D}\left(\frac{x_{rr}}{2}s_2 + \frac{x_m}{\omega_r}a_1\right) \\ u_2 &= \frac{\gamma_1}{D}\left(x_{rr}r_1 - \frac{x_m}{\omega_r}b_2\right) + \frac{\gamma_2}{D}\left(x_{rr}r_2 + \frac{x_m}{\omega_r}b_1\right) \\ u_3 &= v_n(0) - \frac{\gamma_1}{D}\frac{x_m}{\omega_r}\left(c_2 - \sin(\Theta)\right) + \frac{\gamma_2}{D}\frac{x_m}{\omega_r}\left(c_1 - \cos(\Theta)\right) \end{aligned} \tag{A.19}$$

where a_i, b_i and c_i are given by (A.17) and (A.18).

A.6 Finding Real Roots of Polynomials of Degree 3

If we have

$$f(x) = ax^3 + bx^2 + cx + d \tag{A.20}$$

with a, b, c, $d \in R$ and $a \neq 0$, let

$$\begin{aligned} q &= \frac{3ac - b^2}{9a^2} \\ r &= \frac{9abc - 27a^2d - 2b^3}{54a^3} \end{aligned} \tag{A.21}$$

we define the discriminant

$$\Delta = q^3 + r^2 \tag{A.22}$$

There are two distinct cases

- $\Delta > 0$
  In this case there are one real root and two complex roots that are conjugates. We define

$$\begin{aligned} s &= \sqrt[3]{r + \sqrt{\triangle}} \\ t &= \sqrt[3]{r - \sqrt{\triangle}} \end{aligned} \tag{A.23}$$

- $\Delta \leq 0$
  In this case there are three real roots. For the sake of simplicity, we will express the complex number $r + i\sqrt{-\Delta}$ in polar form:

$$r + i\sqrt{\Delta} = \rho \mathrm{e}^{i\theta} = (\rho, \theta) \tag{A.24}$$

where

$$\begin{aligned} \rho &= \sqrt{-q^3} \\ \theta &= \arccos(\frac{r}{\rho}) \end{aligned}$$

We define

$$\begin{aligned} s &= \left(\sqrt[3]{\rho}, \frac{\theta}{3}\right) \\ t &= \left(\sqrt[3]{\rho}, -\frac{\theta}{3}\right) \end{aligned} \tag{A.25}$$

In both cases, the solutions are

$$x_1 = s + t - \frac{b}{3a} \tag{A.26}$$

$$x_2 = -\frac{1}{2}(s+t) - \frac{b}{3a} + i\frac{\sqrt{3}}{2}(s-t) \tag{A.27}$$

$$x_3 = -\frac{1}{2}(s+t) - \frac{b}{3a} - i\frac{\sqrt{3}}{2}(s-t) \tag{A.28}$$

References

[1] B. Wu, *High-Power Converters and AC Drives*, 1st ed. Wiley-IEEE Press, 2006.

[2] T. Geyer, “Generalized model predictive direct torque control: Long prediction horizons and minimization of switching losses,” in *Proceedings of the 48th IEEE Conference on Decision and Control*, 2009, pp. 6799 –6804.

[3] ——, “Computationally efficient model predictive direct torque control,” in *Proceedings of IEEE Energy Conversion Congress and Exposition*, Sept. 2010.

[4] I. Takahashi and T. Noguchi, “A new quick-response and high-efficiency control strategy of an induction motor,” *IEEE Transactions on Industry Applications*, vol. IA-22, no. 5, pp. 820 –827, Sept. 1986.

[5] P. Krause, O. Wasynczuk, and S. Sudhoff, *Analysis of Electric Machinery and Drive Systems*. Wiley-IEEE Press, Feb. 2002.

[6] T. Geyer, G. Papafotiou, and M. Morari, “Model predictive direct torque control - part I: Concept, algorithm, and analysis,” *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1894 –1905, Jun. 2009.

[7] G. Papafotiou, J. Kley, K. Papadopoulos, P. Bohren, and M. Morari, “Model predictive direct torque control - part II: Implementation and experimental evaluation,” *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 1906 –1915, Jun. 2009.

[8] G. Buja and M. Kazmierkowski, “Direct torque control of pwm inverter-fed ac motors - a survey,” *IEEE Transactions on Industrial Electronics,*, vol. 51, no. 4, pp. 744 – 757, Aug. 2004.

[9] D. Q. Mayne and J. B. Rawlings, “Constrained model predictive control: stability and optimality,” *Automatica*, vol. 36, no. 6, pp. 789 – 814, Jun. 2000.

[10] C. E. Garcia, D. M. Prett, and M. Morari, “Model predictive control: theory and practice—a survey,” *Automatica*, vol. 25, no. 3, pp. 335–348, May 1989.

[11] S. J. Qin and T. A. Badgwell, “A survey of industrial model predictive control technology,” *Control Engineering Practice*, vol. 11, no. 7, pp. 733 – 764, Jul. 2003.

[12] D. Q. Mayne and J. B. Rawlings, *Model Predictive Control: Theory and Design*, 1st ed. Nob Hill Publishing, LLC, Aug. 2009.

[13] T. Geyer, “Low complexity model predictive control in power electronics and power systems,” Ph.D. dissertation, ETH Zurich, 2005.

[14] W. J. Rugh, *Linear system theory*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[15] A. J. Laub, *Matrix Analysis For Scientists And Engineers*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2004.

[16] E. L. Lawler and D. E. Wood, “Branch-and-bound methods: A survey,” *OPERATIONS RESEARCH*, vol. 14, no. 4, pp. 699–719, 1966.