Εθνικο Μετσοβιο Πολυτεχνειο
Τμημα Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Ηλεκτρικης Ισχυος
Εργαστηριο Ηλεκτρικων Μηχανων και Ηλεκτρονικων Ισχυος

Model Predictive Control Strategies for DC-DC Converters

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αικατερίνη Ι. Τσώνου

Επιβλέπων: Στέφανος Ν. Μανιάς
Καθηγητής Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2018

Εθνικο Μετσοβιο Πολυτεχνειο
Τμημα Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Ηλεκτρικης Ισχυος
Εργαστηριο Ηλεκτρικων Μηχανων και Ηλεκτρονικων Ισχυος

Model Predictive Control Strategies for DC-DC Converters

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αικατερίνη Ι. Τσώνου

Επιβλέπων: Στέφανος Ν. Μανιάς
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 21η Σεπτεμβρίου 2018

.....	.....	.....
Σ. Παπαθανασίου	Π. Γεωργιλάκης	Χ. Ψυλλάκης
Καθηγητής Ε.Μ.Π.	Αν. Καθηγητής Ε.Μ.Π.	Λέκτορας Ε.Μ.Π.

Αθήνα, Σεπτέμβριος 2018

.....
Αικατερίνη Ι. Τσώνου
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

You can kiss your family and friends good-bye and put miles between you, but at the same time you carry them with you in your heart, your mind, your stomach, because you do not just live in a world but a world lives in you.

Frederick Buechner

Abstract

The application of power electronics, including dc-dc converters, is expanding over the last years. However, the control of these devices still remains a challenge that the scientific community has to face. The main objective is to regulate the output voltage to the desired value, while neglecting any input voltage and load variations. Several current and voltage control methods have been developed through the years, but, due to the nonlinear switching characteristics of the dc-dc converters, these methods have still room for improvement.

A control algorithm that has been gaining popularity in the past years is model predictive control (MPC). Compared to other control methods, such as the proportional integral derivative (PID) controller with pulse width modulation (PWM), MPC has gained remarkable interest both in academia and industry over the past decades. This is true mainly because MPC can be applied in many different processes, can implement constraints and is easy to be understood, since its basic concept can be explained intuitively. The basic idea of MPC is to predict and optimize the future system behavior using the system model.

In this thesis the state-space representation modelling of the MPC is used for the control of an dc-dc buck converter operating in continuous conduction mode (CCM). The formulation of the objective function for the minimization of the voltage error is simple and is achieved by minimizing the output voltage error (difference between output voltage and a reference voltage). Two different implementations of the control problem are presented: a mixed-integer quadratic optimization problem solved with enumeration technique and a quadratic optimization problem solved using the gradient and Newton's descent methods. In addition, in order to provide offset-free tracking of the output voltage reference, a Kalman filter is used. A voltage MPC algorithm using these three methods (enumeration, gradient and Newton's) is applied in MATLAB/SIMULINK. Variations in the input voltage, the output voltage and the load are also applied in order to examine the response of the three methods to these.

The thesis is divided into two main chapters. In Chapter 1, the theoretical background with regards to the uses of dc-dc converters and their control methods is given. Since MPC is an optimal control scheme, the main aspects of the mathematical optimization theory are also introduced in this chapter. Furthermore, some important classes of optimization problems, namely convex optimization problems, as well as their solving techniques are presented. In Chapter 2, the system modelling for the dc-dc buck converter is presented, the objective function is formulated and the optimization problem is simulated. The results and their comparison for the three solving techniques mentioned above are also presented in Chapter 2.

Keywords dc-dc buck converter, buck converter, control methods, model predictive control, optimization, continuous conduction mode, enumeration method, gradient method, Newton's method

Περίληψη

Η εφαρμογή των ηλεκτρονικών ισχύος, συμπεριλαμβανομένων των dc-dc μετατροπέων, παρουσιάζει ανάπτυξη τα τελευταία χρόνια. Ωστόσο, ο έλεγχος αυτών των συσκευών παραμένει μια πρόκληση που καλείται να αντιμετωπίσει η επιστημονική κοινότητα. Ο κύριος στόχος είναι η ρύθμιση της τάσης εξόδου στην επιθυμητή τιμή, αμελώντας οποιεσδήποτε μεταβολές της τάσης εισόδου και του φορτίου. Κατά την διάρκεια των τελευταίων ετών, αρκετές μέθοδοι ελέγχου ρεύματος και τάσης έχουν αναπτυχθεί, αλλά λόγω των μη γραμμικών διακοπτικών χαρακτηριστικών των δς-δς μετατροπέων, αυτές οι μέθοδοι δεν έχουν ακόμη καταλήξει σε επαρκές επίπεδο ελέγχου.

Ένας αλγόριθμος ελέγχου που έχει αναδειχθεί τα τελευταία χρόνια είναι αυτός του μοντελοποιημένου προβλεπτικού ελέγχου (model predictive control - MPC). Σε σύγκριση με άλλες μεθόδους ελέγχου, όπως ο αναλογικός-ολοκληρωτικός-διαφορικός ελεγκτής (PID) με διαμόρφωση εύρους παλμού (PWM), ο προβλετικός έλεγχος έχει κερδίσει αξιοσημείωτο ενδιαφέρον τόσο στον ακαδημαϊκό όσο και στον βιομηχανικό κλάδο τις τελευταίες δεκαετίες. Αυτό ισχύει κυρίως επειδή ο προβλεπτικός έλεγχος μπορεί να εφαρμοστεί σε πολλές διαφορετικές διαδικασίες, μπορεί να υλοποιήσει περιορισμούς και είναι εύκολο να γίνει κατανοητό, καθώς η βασική του έννοια μπορεί να εξηγηθεί διαισθητικά. Η βασική ιδέα του προβλεπτικού ελέγχου είναι να προβλέψει και να βελτιστοποιήσει τη μελλοντική συμπεριφορά του συστήματος χρησιμοποιώντας το μοντέλο του συστήματος.

Στην παρούσα διπλωματική χρησιμοποιείται η μοντελοποίηση της αναπαράστασης του χώρου καταστάσεων (state-space representation) του MPC για τον έλεγχο ενός dc-dc buck μετατροπέα που λειτουργεί σε συνεχή λειτουργία αγωγιμότητας (CCM). Η διατύπωση της αντικειμενικής συνάρτησης για την ελαχιστοποίηση του σφάλματος τάσης είναι απλή και επιτυγχάνεται ελαχιστοποιώντας το σφάλμα τάσης εξόδου (διαφορά μεταξύ τάσης εξόδου και τάσης αναφοράς). Δύο διαφορετικές υλοποιήσεις του προβλήματος ελέγχου παρουσιάζονται: ένα μικτών ακεραίων τετραγωνικό πρόβλημα βελτιστοποίησης (mixed-integer quadratic optimization problem) που επιλύθηκε με τεχνική απαρίθμησης και ένα τετραγωνικό πρόβλημα βελτιστοποίησης που επιλύθηκε χρησιμοποιώντας τις gradient και Newton's descent μεθόδους. Επιπροσθέτως, προκειμένου να παρέχεται παρακολούθηση χωρίς το offset της αναφοράς τάσης εξόδου, χρησιμοποιείται ένα φίλτρο Kalman. Ένας αλγόριθμος τάσης MPC που χρησιμοποιεί αυτές τις τρεις μεθόδους (απαρίθμηση, gradient και Newton's) εφαρμόζεται σε MATLAB / SIMULINK. Μεταβολές στην τάση εισόδου, την τάση εξόδου και το φορτίο εφαρμόζονται επίσης για να εξεταστεί η απόκριση των τριών μεθόδων.

Η διπλωματική εργασία χωρίζεται σε δύο κύρια κεφάλαια. Στο Κεφάλαιο 1 παρουσιάζεται το θεωρητικό υπόβαθρο αναφορικά με τις χρήσεις των dc-dc μετατροπέων και των μεθόδων ελέγχου τους. Δεδομένου ότι το MPC είναι ένα βέλτιστο σχήμα ελέγχου, οι κύριες πτυχές της θεωρίας της μαθηματικής βελτιστοποίησης εισάγονται επίσης σε αυτό το κεφάλαιο. Επιπλέον, παρουσιάζονται μερικές σημαντικές κατηγορίες προβλημάτων βελτιστοποίησης, δηλαδή κυρτά προβλήματα βελτιστοποίησης (convex optimization problems), καθώς και οι τεχνικές επίλυσης τους. Στο κεφάλαιο 2 παρουσιάζεται η μοντελοποίηση του συστήματος για τον dc-dc buck μετατροπέα, καθώς επίσης διαμορφώνεται η

αντικειμενική συνάρτηση και προσομοιώνεται το πρόβλημα βελτιστοποίησης. Τα αποτελέσματα και η σύγκρισή τους για τις τρεις τεχνικές επίλυσης που αναφέρονται παραπάνω παρουσιάζονται στο Κεφάλαιο 2.

Λέξεις Κλειδιά dc-dc μετατροπέας, μέθοδοι ελέγχου, μοντέλο πρόβλεψης, βελτιστοποίηση, κατάσταση συνεχούς αγωγής, μέθοδος απαρίθμησης, μέθοδος gradient, μέθοδος Newton's

Acknowledgements

Throughout the writing of this diploma thesis, numerous persons supported me significantly in their personal way. The following lines aim to show them my deepest gratitude.

First of all, I would like to express my sincere and humble appreciation to Prof. Stefanos Manias for giving the opportunity to work in a research field I was interested in. His patience and assistance shaped this thesis.

Secondly, I would like to thank Prof. Ralph Kennel from Technical University of Munich for allowing me to work under his supervision and providing me with technical insights and innovative ideas.

Furthermore, I am most grateful to Petros Karamanakos for supporting me daily and giving me motivations to continue despite the difficulties. His contribution was critical for the completion of this thesis.

Last but not least, I wish to thank my family and my friends for being there for me throughout the conduction of this diploma thesis and my academic life in general. This diploma thesis is dedicated to them.

Contents

List of Algorithms

List of Figures

Chapter 1

Introduction

1.1 DC-DC Converters

1.1.1 Uses

Dc-dc converters are power electronic circuits that convert one level of electrical voltage into another level by switching action and this has received an increasing deal of interest in many areas. The main functions of the dc-dc converters, as these can be found in [25, 31], are summarized here:

- Convert a dc input voltage V_s into a dc output voltage V_0
- Regulate the dc output voltage against load and line variations;
- Reduce the ac voltage ripple on the dc output voltage below the required level;
- Provide isolation between the input source and the load;
- Protect the supplied system and the input source from the electromagnetic interference;
- Satisfy various international and national safety standards.

Modern applications of dc-dc converters can be found in electric vehicles [37, 8, 33, 26, 1], renewable energy systems [11, 29, 4, 30] and other [34] sectors.

An interesting example of the use of dc-dc converters in photovoltaic (PV) systems is [36] where a PV system with dc-dc converters is developed for water pumping in developing countries. This study is mentioned here for showing a basic application of dc-dc converters in PV systems and for emphasising the importance of control techniques in order to optimise the power output and therefore the performance of a whole PV system.

1.1.2 Control Methods

In their simplest form dc-dc converters comprise two semiconductor switches that are periodically switched on and off, and a low pass filter with an inductor and a capacitor. The filter is added to pass the dc component of the input, and to remove the switching harmonics from the output voltage. The detailed topology and main electrical circuit of the dc-dc converter is presented in Section 2.1.1

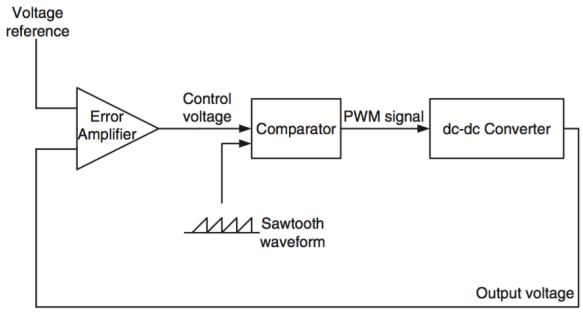
Despite the fact that the switch-mode dc-dc conversion is a well-established technology, the problems associated with these applications and their closed-loop controlled performance still pose theoretical and practical challenges. The main control objective for the dc-dc converters, is the regulation of the output voltage, while rejecting the impact of variations in the input voltage and the load. The difficulty in this, lies on the unregulated input voltage conditions, time-varying load, switched nonlinear characteristics, component-varying values due to temperature and pressure conditions, etc. Therefore, the control of the output voltage should be performed in a closed-loop manner using principles of negative feedback. The two most common closed-loop methods for dc-dc converters are the voltage-mode control and the current-mode control. These are schematically represented in Fig. 1.1.

In the voltage-mode scheme shown in Fig. 1.1a, the converter output voltage is measured and subtracted from an external reference voltage in an error amplifier. The error amplifier produces a control voltage that is used to determine the switching duty ratio by comparison with a constant frequency waveform. This duty ratio is used to maintain the average voltage across the inductor and is eventually setting the output voltage to its reference value without variations.

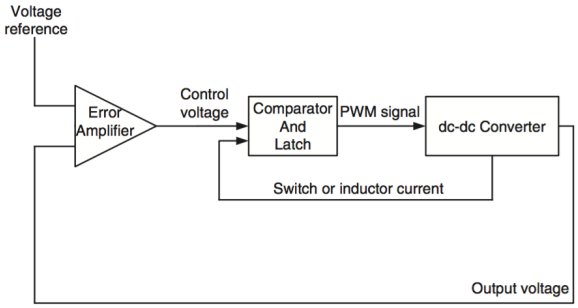
In the current-mode scheme shown in Fig. 1.1b, an additional loop feeds back an inductor current signal. This current signal, converted into its voltage analog, is compared to the control voltage and is used to control the duty cycle. An error signal is produced after comparing the output voltage to the reference voltage and this error signal is used to generate the reference current. The next step is to measure the inductor current and compare it to the reference current to generate the duty cycle and drive the switch of the converter. It is to be noted here that although the current-mode controllers have two loops, they are more often employed since the design procedure is simpler.

In general, pulse width modulation (PWM) techniques are used in the controller loop to turn on and off the controllable switch and maintain the output voltage equal to the reference. This means that by conforming the pulse width, i.e. by modifying the duty cycle d[1], the output voltage is regulated to the desired level. However, there are also strategies where a modulator is not required. According to these methods

[1] $d = t_{on}/T_s$, where t_{on} is the time the switch is closed and T_s is the switching period.



(a) Voltage-mode control



(b) Current-mode control

Figure 1.1: Main control schemes for dc-dc converters [31].

the switch is directly manipulated without the presence of an intermediate modulator. Therefore, the PMW used in Fig. 1.1 is optional and can be bypassed.

In literature many different approaches to the control problem can be found. These are mainly divided into two main groups: the linear and the nonlinear controllers. The majority of the controllers are based on the conventional proportional integral deviation (PID) controller and are tuned in the basis of the linear state-space average model of the converter. Throughout the years several nonlinear controllers based on the averaged or nonaveraged state-space model of the converter have been proposed as well. Although many of the approaches applied have been shown to be reasonably effective, several challenges such as the ease of the controller design and tuning as well as the robustness to the load variations have not been fully addressed yet. Moreover, the aim to improve the performance of the closed-loop system and enable a systematic design and implementation procedure still exists. Last but not least, the recent theoretical advances with regards to controlling hybrid systems, as well as the emergence of fast microprocessors that enabled the implementation of more computationally demanding algorithms, allow one to tackle these challenges in a novel way [17].

Model predictive control (MPC) is a control strategy that was developed an an alternative to the conventional PID control and has been gaining popularity in the field of power electronics the last years. Its success is based on the fact that it uses a mathematical model of the plant, which allows the controller to predict the impact of

its control actions. Furthermore, MPC is capable of handling complex and nonlinear dynamics, while several design criteria can be explicitly included in a simple and effective manner [17]. The basic idea of the MPC as well as its basic elements are presented Chapter 2.

1.1.3 Thesis Aim

The aim of this Thesis is to compare three different methods (enumeration, gradient descent and Newton's) to control the dc-dc buck converter using model predictive control (MPC) in terms of computational efficiency and complexity.

In the next paragraphs of this Chapter, a mathematical background is introduced in order for the reader to get familiar with the optimisation theory and how this is applied in the case of MPC. The formulation of the MPC problem is presented in the next Chapter, together with the simulation results and the conclusions.

1.2 Convexity

Definition 1.2.1. The set $C \subset \mathbb{R}^n$ is said to be convex if $\alpha x + (1-\alpha)y$ is in C whenever $\boldsymbol{x}$ and $\boldsymbol{y}$ are in C, and $\alpha \in [0,1]$.[16]

Geometrically this means that for every two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in the set C the straight line that connects the two points is also in the set. In Fig. 1.2 and 1.3 there are some examples of convex and nonconvex sets.

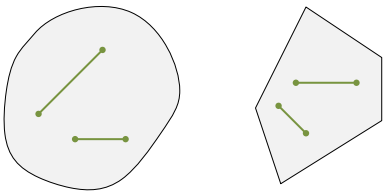


Figure 1.2: Convex sets

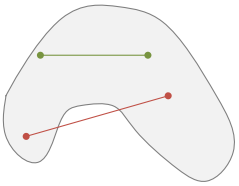


Figure 1.3: Nonconvex set

Definition 1.2.2. Let C be a nonempty convex set in $\mathbb{R}^n$. A function $f : C \to \mathbb{R}$ is said to be *convex* on C when, for all pairs $(\boldsymbol{x}, \boldsymbol{y}) \in C \times C$ and all $\alpha \in (0,1)$, there holds[16]

$$f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y}). \tag{1.1}$$

Geometrically this means that for every two points x and y in the set C the straight line that connects the points $(\boldsymbol{x}, f(\boldsymbol{x}))$ and $(\boldsymbol{y}, f(\boldsymbol{y}))$ lays above the function graph between these points, as show in Fig. 1.4.

When (1.1) holds as a strict inequality for $\boldsymbol{x} \neq \boldsymbol{y}$, the function f is said to be *strictly convex*. A very useful property of strictly convex functions is that they have at most one *global minimum.*

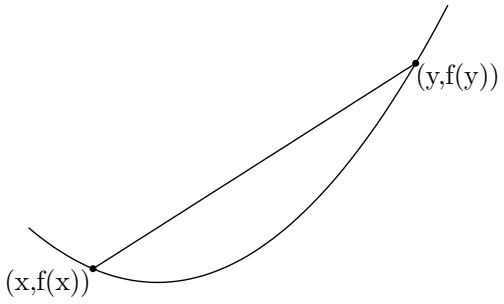


Figure 1.4: Convex function

1.3 Convex Optimization

Model predictive control is an optimal control scheme and therefore, an introduction to the basic terminology of the mathematical optimization theory here is necessary. In addition, some important classes of optimization problems, namely convex optimization problems, linear optimisation problems, quadratic optimization problems, mixed-integer linear optimization problems, and mixed-integer quadratic optimisation problems are presented. [17, 9]

According to [35], an optimisation problem is of the form:

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & g_i(\boldsymbol{x}) \leq 0 \quad i = 1, \ldots, m \\ & h_j(\boldsymbol{x}) = 0 \quad j = 1, \ldots, p \end{array} \tag{1.2}$$

The goal is to find the optimization variable $\boldsymbol{x} \in \mathbb{R}^n$ that minimizes the objective function f: $\mathbb{R}^n \to \mathbb{R}$, while satisfying the conditions $g_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m$ and $h_j(\boldsymbol{x}) = 0, j = 1, \ldots, p$. The inequalities $g_i(\boldsymbol{x}) \leq 0$ in (1.2) are called inequality constraints and the corresponding functions $g_i : \mathbb{R}^n \to \mathbb{R}$ inequality constraints functions. The equalities $h_j(\boldsymbol{x}) = 0$ in (1.2) are called equality constraints and the corresponding functions $h_j : \mathbb{R}^n \to \mathbb{R}$ equality constraints functions.

The domain $\mathcal{O}$ of the optimization problem (1.2) is the set of the points for which the objective function f and the constraint functions g and h are defined, thus

$$\mathcal{O} = \mathbf{dom} f \cap \bigcap_{i=1}^{m} \mathbf{dom} g_i \cap \bigcap_{j=1}^{p} \mathbf{dom} h_j. \tag{1.3}$$

A point $\boldsymbol{x} \in \mathcal{O}$ is said to be feasible if it satisfies all the constraints $g_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m$ and $h_j \boldsymbol{x} = 0, j = 1, \ldots, p$. The problem (1.2) is feasible if there exists at least

one feasible point, else it is infeasible.

The optimal value q^* of the problem (1.2) is defined as

$$q^* = \inf\{f(\boldsymbol{x}) \mid g_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m, h_j(\boldsymbol{x}) = 0, j = 1, \ldots, p\}. \tag{1.4}$$

The optimal value q^* may be equal to $\pm\infty$. If the problem is infeasible then $q^* = \infty$; if the problem is unbound below, i.e. there are points $\boldsymbol{x}_k$ such that $f(\boldsymbol{x}) \to -\infty$ as $k \to -\infty$, then $q^* = -\infty$. The solution $\boldsymbol{x}^*$ of the optimisation problem (1.2)is called optimal point, if $\boldsymbol{x}^*$ is feasible and $f(\boldsymbol{x}^*) = q^*$. The set of all optimal values

$$X_{opt} = \{\boldsymbol{x} \mid f(\boldsymbol{x}) = q^*, g_i(\boldsymbol{x}) \leq 0\, i = 1, \ldots, m, h_j(\boldsymbol{x}) = 0, j = 1, \ldots, p\} \tag{1.5}$$

is called optimal set. The optimal value is achieved, if there exists an optimal point $\boldsymbol{x}^*$ for the problem (1.2), otherwise the set X_{opt} is empty. If the optimal value is achieved, then the optimization problem is solvable. A feasible point $\boldsymbol{x}$ is locally optimal if it minimizes f in a subset of the feasible set, i.e. if there is an $R > 0$ such that

$$f(\boldsymbol{x}) = \inf\{f(\boldsymbol{z}) | g_i(\boldsymbol{z}) \leq 0, i = 1, \ldots, m, h_j(\boldsymbol{z}) = 0, j = 1, \ldots, p, ||\boldsymbol{z} - \boldsymbol{x}||_2 \leq R\}, \tag{1.6}$$

with $\boldsymbol{z} \in \mathbb{R}$ or, equivalently, if it is the solution to the optimization problem

$$\begin{array}{rll} \text{minimize} & f(\boldsymbol{z}) & \\ \text{subject to} & g_i(\boldsymbol{z}) \leq 0 & i = 1, \ldots, m \\ & h_j \boldsymbol{z} = 0 & j = 1, \ldots, p \\ & ||\boldsymbol{z} - \boldsymbol{x}||_2 \leq R & \end{array} \tag{1.7}$$

If a feasible point $\boldsymbol{x}$ minimizes f for the whole feasible set, then it is called globally optimal.

An important class of optimization problems are convex optimization problems. These are of the form

$$\begin{array}{rll} \text{minimize} & f(\boldsymbol{x}) & \\ \text{subject to} & g_i(\boldsymbol{x}) \leq 0 & i = 1, \ldots, m \\ & \boldsymbol{a}_j^T \boldsymbol{x} = \boldsymbol{b}_j & j = 1, \ldots, p \end{array} \tag{1.8}$$

where the objective function f and the inequality constraints functions $g_1, \ldots, g_m$ are convex, and the equality constraints functions are affine. Furthermore, the feasible set is convex; it is the intersection of the domain of the convex optimization problem (1.8), which is a convex set, with m convex sublevel sets $\{\boldsymbol{x} | g_i(\boldsymbol{x}) \leq 0, i = 1, \ldots, m\}$ and p hyperplanes $\{x | \boldsymbol{a}_j^T \boldsymbol{x} = \boldsymbol{b}_j, j = 1, \ldots, p\}$, i.e.

$$\mathcal{O} = \mathbf{dom} f \bigcap_{i=1}^{m} \mathbf{dom} g_i. \tag{1.9}$$

Based on the above a fundamental property of convex optimization problems is derived; any locally optimal point is also globally optimal.

1.3.1 Linear Programming

When the objective and constraint functions are all affine, the problem is called a linear program (LP). A general linear program has the form

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c}^T\boldsymbol{x} + \boldsymbol{d} \\ \text{subject to} \quad & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{1.10}$$

where $\boldsymbol{G} \in R^{m\times n}$ and $\boldsymbol{A} \in R^{p\times n}$. Linear programs are, of course, convex optimization problems.

It is common to omit the constant $\boldsymbol{d}$ in the objective function, since it does not affect the optimal (or feasible) set. Since we can maximize an affine objective $\boldsymbol{c}^T\boldsymbol{x}+\boldsymbol{d}$, by minimizing $-\boldsymbol{c}^T\boldsymbol{x} - \boldsymbol{d}$ (which is still convex), we also refer to a maximization problem with affine objective and constraint functions as an LP.

1.3.2 Quadratic Programming

The convex optimization problem (1.8) is called a *quadratic program* (QP) if the objective function is (convex) quadratic, and the constraint functions are affine. A quadratic program can be expressed in the form

$$\begin{aligned} \text{minimize} \quad & (1/2)\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{r} \\ \text{subject to} \quad & \boldsymbol{G}\boldsymbol{x} \preceq \boldsymbol{h} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \end{aligned} \tag{1.11}$$

where $\boldsymbol{P} \in S^n_+$, $\boldsymbol{G} \in R^{m\times n}$ and $\boldsymbol{A} \in R^{p\times n}$. [22, 6, 35]

1.3.3 Mixed-Integer Linear Programming

The optimization variable in some cases may contain a continuous component and a binary part. The optimization problem (1.10) in this case is called mixed-integer linear

program (MILP), and it is of the form

$$\begin{aligned} \text{minimize} \quad & \boldsymbol{c^T x} \\ \text{subject to} \quad & \boldsymbol{Gx} \preceq \boldsymbol{h} \\ & \boldsymbol{Ax} = \boldsymbol{b} \\ & \boldsymbol{x_b} \in \{0,1\}^{n_b} \end{aligned} \tag{1.12}$$

where $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_r^T & \boldsymbol{x}_b^T \end{bmatrix}^T$, with $\boldsymbol{x_r} \in \mathbb{R}^{n_r}$, $\boldsymbol{x_b} \in \{0,1\}^{n_b}$, and $n = n_r + n_b$. Furthermore, $\boldsymbol{c} \in \mathbb{R}^n$, $G \in \mathbb{R}^{m \times n}$, $h \in \mathbb{R}^m$, $A \in \mathbb{R}^{p \times n}$, and $\boldsymbol{b} \in \mathbb{R}^p$.

It should be noted that despite the fact that the objective function and the constraints functions are linear (or affine), the problem (1.12) is nonconvex because of the presence of the binary component. This means that the important property of convex optimization problems does not apply to MILPs; the locally optimal points may not be globally optimal. Finally, an MILP is NP-hard, i.e. the running time depends exponentially on the number of the binary components, as it is written in [17]

1.3.4 Mixed-Integer Quadratic Programming

If the optimization variable of the problem (1.11) contain both a real-valued part and a binary part, i.e. it is of the form $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_r^T & \boldsymbol{x}_b^T \end{bmatrix}^T$, with $\boldsymbol{x_r} \in \mathbb{R}^{n_r}$, $\boldsymbol{x_b} \in \{0,1\}^{n_b}$, and $n = n_r + n_b$, then the formulated optimization problem is called mixed-integer quadratic program (MIQP)

$$\begin{aligned} \text{minimize} \quad & (1/2)\boldsymbol{x^T Q x} + \boldsymbol{p^T x} \\ \text{subject to} \quad & \boldsymbol{Gx} \preceq \boldsymbol{h} \\ & \boldsymbol{Ax} = \boldsymbol{b} \\ & \boldsymbol{x_b} \in \{0,1\}^{n_b} \end{aligned} \tag{1.13}$$

with $\boldsymbol{Q} \in \boldsymbol{S_+^n}$, $\boldsymbol{p} \in \mathbb{R}^n$, $\boldsymbol{G} \in \mathbb{R}^{m \times n}$, $\boldsymbol{h} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{p \times n}$, and $\boldsymbol{b} \in \mathbb{R}^p$. As already mentioned in Section 1.3.3, problem (1.13) is non-convex because of the binary part $\boldsymbol{x_b}$, and it is NP-hard, as it is written in [17]

1.4 Solution Methods for Optimization Problems

1.4.1 Enumeration Method for Mixed Integer Programming

In general, solving the mixed-integer optimization problems (MIPs) presented in Sections 1.3.3 and 1.3.4 is a very challenging task. For determining the solution of an MIP, either in the form of 1.12), or in the form of (1.13), for an MILP or an MIQP, respectively, a straightforward option is to use an enumeration strategy.

According to the complete enumeration method at each integer variable are progressively assigned the different values of its domain. The procedure is repeated until no more free integer variables are left, and the complete solution for the integer variables $\tilde{\boldsymbol{x}}_{\boldsymbol{b}}$ is obtained. Therefore, the MILP is simplified to an LP (or to a QP if the problem is an MIQP). By solving the resulting LP (or QP) the optimal value $q^* = f(\boldsymbol{x}^*)$ of the real-valued variables is determined, as it is written in [17]

1.4.2 Descent Methods for Quadratic Programming

For these particular optimization problems one can take advantage of the convexity property they hold (meaning they have exactly one global minimum or maximum) and use descent iterative methods for the solution of the problem.

These methods produce a minimizing sequence $\boldsymbol{x}^{(k+1)}, k \in \mathbb{N}$ where

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + t^{(k)}\boldsymbol{\Delta x}^{(k)}.$$

$\boldsymbol{\Delta x}^{(k)}$is the *step* or *search direction*, is a vector in $\mathbb{R}^n$ and must be consider an entity. The scalar $t^{(k)} > 0$ is called *step size* or *step length* at iteration k

These methods are called *descent* in the sense that the value of the function to be minimized f is reduced in every iteration, or

$$f(\boldsymbol{x}^{(k+1)}) < f(\boldsymbol{x}^{(k)})[9].$$

In general the descent method is as follows.

Algorithm 1.1 General descent method

given a starting point $\boldsymbol{x}^{(k)} \in \mathbf{dom} f$
repeat

1. Determine a descent direction $\boldsymbol{\Delta x}^{(k)}$.
2. *Line search.* Choose a step size $t^{(k)} > 0$.
3. *Update.* $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + t^{(k)}\boldsymbol{\Delta x}^{(k)}$.

until stopping criterion is satisfied

One line search method is the *backtracking line search* and it depends on two constants $0 < \alpha < 0.5$ and $0 < \beta < 1$. The backtracking line search minimizes f along the ray $\{\boldsymbol{x}^{(k)} + \boldsymbol{\Delta x}^{(k)} | t^{(k)} \geq 0\}$ approximately.

Different descent methods exist depending on the way the descent direction $\boldsymbol{\Delta x}$ is chosen. Two of them are the *gradient descent method* and the *Newton's method*.

Algorithm 1.2 Backtracking line search [9].

given a descent direction $\boldsymbol{\Delta x}^{(k)}$ for f at $\boldsymbol{x}^{(k)} \in \mathbf{dom} f$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
t := 1.
while $f(\boldsymbol{x}^{(k)} + t^{(k)}\boldsymbol{\Delta x}^{(k)}) > f(x^{(k)}) + \alpha t^{(k)} \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{\Delta x}^{(k)}$, $t^{(k)} := \beta t^{(k)}$.

It should be noted that both methods are actually solution techniques for unconstrained optimization but can be used in certain cases of constrained optimization.

1.4.2.1 Gradient Method

For this method the negative gradient is chosen as the descent direction,

$$\boldsymbol{\Delta x}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)}).$$

This is a natural choice considering that the gradient points in the direction of the maximum increase (and thus the negative gradient points in direction of the maximum decrease.)

For this method the stopping criterion is usually of the form $\|\nabla f(\boldsymbol{x}^{(k)})\|_2 \leq \eta$, where η is a small positive number which expresses the accuracy of the calculation.

The gradient method is as outlined in algorithm 1.3.

Algorithm 1.3 Gradient descent method [9]

given a starting point $x^{(k)} \in \mathbf{dom} f$
repeat

1. $\boldsymbol{\Delta x}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)})$.
2. Choose a step size $t^{(k)} > 0$ using backtracking line search.
3. *Update.* $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + t^{(k)}\boldsymbol{\Delta x}^{(k)}$.

until stopping criterion is satisfied

1.4.2.2 Newton's Method

In Newton's method the *Newton step* is chosen as the descent direction. The Newton step is

$$\boldsymbol{\Delta x}^{(k)} = -\nabla^2 f(\boldsymbol{x}^{(k)})^{-1} \nabla f(\boldsymbol{x}^{(k)}).$$

The *Newton decrement*

$$\lambda = (\nabla f(\boldsymbol{x}^{(k)})^T \nabla^2 f(\boldsymbol{x}^{(k)})^{-1} \nabla f(\boldsymbol{x}^{(k)})^{1/2}$$

is used in the stopping criterion [9].

Algorithm 1.4 Newton's descent method

given a starting point $\boldsymbol{x}^{(k)} \in \mathbf{dom} f$, tolerance $\eta > 0$
repeat

1. *Compute the Newton step and decrement*
   $\boldsymbol{\Delta x}^{(k)} = -\nabla^2 f(\boldsymbol{x}^{(k)})^{-1} \nabla f(\boldsymbol{x}^{(k)})$, $\lambda^2 = \nabla f(\boldsymbol{x}^{(k)})^T \nabla^2 f(\boldsymbol{x}^{(k)})^{-1} \nabla f(\boldsymbol{x}^{(k)})$
2. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \eta$
3. Choose a step size $t^{(k)} > 0$ using backtracking line search.
4. *Update.* $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + t^{(k)} \boldsymbol{\Delta x}^{(k)}$.

In comparison to the gradient method, which takes into account only the slop of the function, the Newton's method uses the curvature of the function in the form of the Hessian matrix $\nabla^2 f$ as well, which makes it the fastest method of the two, meaning it needs less iterations to find the minimum.

Example 1.4.1. This example is designed to demonstrate the different paths the two methods discussed in sections 1.4.2.1 and 1.4.2.2 choose to minimize a function.

To demonstrate how these two methods work the function $f(x_1, x_2) = 3x_1^2 + x_2^2 - 2x_1x_2 - 6x_2 + 3$ will be minimized using them. In accordance with the minimization problem (1.11) the function f can be written in vector form as:

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \begin{bmatrix} 6 & -2 \\ -2 & 2 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 & -6 \end{bmatrix} \boldsymbol{x} + 3, \tag{1.14}$$

where $\boldsymbol{x} = [x_1 \quad x_2]^T$. In both cases the same starting point $(4, 7)$ was chosen, with $\eta = 10^{-6}$.

From Fig. 1.5 we can see that Newton's method follows a more direct path to the minimum than gradient method and thus is faster needing less iterations (78 iterations) than gradient method (123 iterations) to find the minimum.

For more information about the descent methods the reader may refer to [28] and [9].

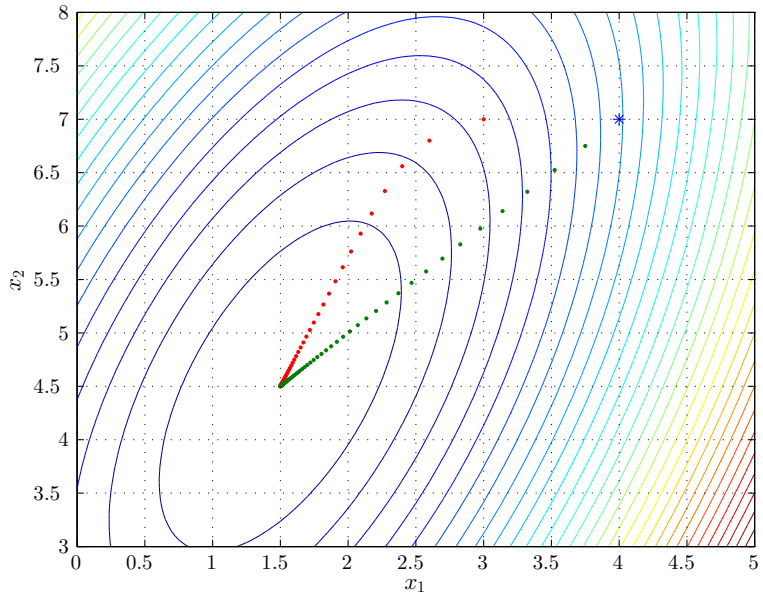


Figure 1.5: Minimization of the function $f(x_1, x_2) = 3x_1^2 + x_2^2 - 2x_1x_2 - 6x_2 + 3$ with the use of gradient descent (red dots) and Newton's descent (green dots) methods, starting from the same point (blue star) $(4, 7)$

1.5 Model Predictive Control

Model predictive control (MPC) has gained interest both in academia and industry over the past few decades compared to other control methods, such as PID or PWM control. This is true mainly because MPC can be applied in many different processes, can implement constraints and is easy to be understood, since its basic concepts can be explained intuitively.[10, 20]

The basic idea of model predictive control is to predict and optimize the future system behaviour using the system model. The basic elements of MPC are:

1. The *system model*, which describes the plant's behaviour over time.
2. The *control problem*, where an objective function is formulated with regard to the system model to calculate the optimal control sequence for a specific number of future instances (*prediction horizon*)
3. *Receding horizon policy*,

The following sections explain these elements in detail.

1.5.1 System Representation

Modelling is a very important part of the MPC algorithms, since different models produce algorithms with different complexity influencing the time needed to find the

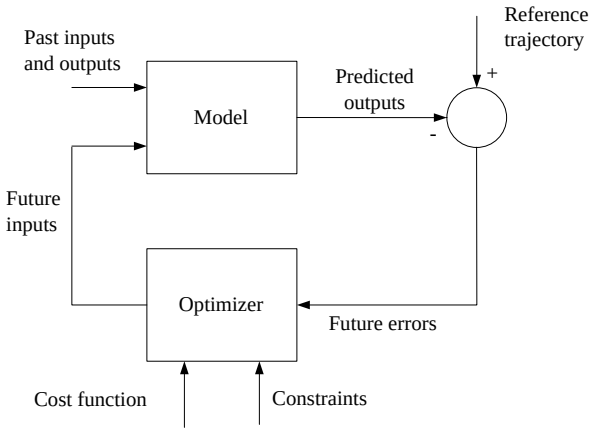


Figure 1.6: MPC structure,[10]

optimal solution to the control problem.[21] Many different forms of modelling can be used in MPC [10], but for the purposes of this thesis the *state space representation* will be used.

The use of microprocessors makes MPC a discrete-time controller, thus the state-space representation in discrete-time is in order for the system model:

$$\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \boldsymbol{u}(k)) \tag{1.15a}$$

$$\boldsymbol{y}(k) = g(\boldsymbol{x}(k)) \tag{1.15b}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^n$ is the state vector of the system at time instant kT_s, $\boldsymbol{u}(k) \in \mathbb{R}^m$ is the input vector at time instant kT_s, $\boldsymbol{y}(k) \in \mathbb{R}^p$ is the output vector at time instant kT_s, the functions f and g are the state-update and output functions, respectively, which can be linear or nonlinear, and T_s is the sampling interval.

Starting from the current state $\boldsymbol{x}(k)$, this model is used for the calculation of the state and the output future values over a finite number N of planed control actions $\{\boldsymbol{u}(k), \boldsymbol{u}(k+1), \ldots, \boldsymbol{u}(k+N)\}$.

Starting from the step $k+1$ for the state it holds:

$$\begin{aligned}
\boldsymbol{x}(k+1) &= f(\boldsymbol{x}(k), \boldsymbol{u}(k)) \\
\boldsymbol{x}(k+2) &= f(\boldsymbol{x}(k+1), \boldsymbol{u}(k+1)) = f(f(\boldsymbol{x}(k), \boldsymbol{u}(k)), \boldsymbol{u}(k+1)) \\
&\vdots \\
\boldsymbol{x}(k+N) &= f(\boldsymbol{x}(k+N-1), \boldsymbol{u}(k+N-1)) \\
&= f(f \ldots (f(\boldsymbol{x}(k), \boldsymbol{u}(k)), \boldsymbol{u}(k+1)), ...\boldsymbol{u}(k+N-1))
\end{aligned} \tag{1.16}$$

Thus the output over N steps is

$$\begin{aligned}
\boldsymbol{y}(k+1) &= g(f(\boldsymbol{x}(k), \boldsymbol{u}(k))) \\
\boldsymbol{y}(k+2) &= g(f(\boldsymbol{x}(k+1), \boldsymbol{u}(k+1))) = g(f(f(\boldsymbol{x}(k), \boldsymbol{u}(k)), \boldsymbol{u}(k+1))) \\
&\vdots \\
\boldsymbol{y}(k+N) &= g(f(\boldsymbol{x}(k+N-1), \boldsymbol{u}(k+N-1))) \\
&= g(f(f \ldots (f(\boldsymbol{x}(k), \boldsymbol{u}(k)), \boldsymbol{u}(k+1)), ...\boldsymbol{u}(k+N-1)))
\end{aligned} \tag{1.17}$$

Linear model For a linear model the discrete time state-space representation is:

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \tag{1.18a}$$

$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) \tag{1.18b}$$

where $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ are the system matrices.

Following the same procedure as in (1.16) and (1.17), the state over a finite number of planed actions N is:

$$\begin{aligned}
\boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\
\boldsymbol{x}(k+2) &= \boldsymbol{A}\boldsymbol{x}(k+1) + \boldsymbol{B}\boldsymbol{u}(k+1) \\
&= \boldsymbol{A}^2\boldsymbol{x}(k) + \boldsymbol{A}\boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{B}\boldsymbol{u}(k+1) \\
&\vdots \\
\boldsymbol{x}(k+N) &= \boldsymbol{A}\boldsymbol{x}(k+N-1) + \boldsymbol{B}\boldsymbol{u}(k+N-1) \\
&= \boldsymbol{A}^N\boldsymbol{x}(k) + \boldsymbol{A}^{N-1}\boldsymbol{B}\boldsymbol{u}(k) + \boldsymbol{A}^{N-2}\boldsymbol{B}\boldsymbol{u}(k+1) + \cdots + \boldsymbol{B}\boldsymbol{u}(k+N-1)
\end{aligned} \tag{1.19}$$

or in a matrix form:

$$\begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{x}(k+2) \\ \vdots \\ \boldsymbol{x}(k+N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} \boldsymbol{B} & 0 & \cdots & 0 \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}(k) \\ \boldsymbol{u}(k+1) \\ \vdots \\ \boldsymbol{u}(k+N-1) \end{bmatrix} \tag{1.20}$$

Using the *last applied* control action $\boldsymbol{u}(k-1)$ (which is known) it is possible to

write the future control actions $\boldsymbol{u}(k), \boldsymbol{u}(k+1), \ldots, \boldsymbol{u}(k+N-1)$ in the form:

$$\begin{aligned} \boldsymbol{u}(k) &= \boldsymbol{u}(k) - \boldsymbol{u}(k-1) + \boldsymbol{u}(k-1) \\ &= \Delta\boldsymbol{u}(k) + \boldsymbol{u}(k-1) \\ \boldsymbol{u}(k+1) &= \Delta\boldsymbol{u}(k+1) + \Delta\boldsymbol{u}(k) + \boldsymbol{u}(k-1) \\ &\vdots \\ \boldsymbol{u}(k+N-1) &= \Delta\boldsymbol{u}(k+N-1) + \cdots + \Delta\boldsymbol{u}(k) + \boldsymbol{u}(k-1) \end{aligned} \tag{1.21}$$

where $\Delta\boldsymbol{u}(k+l-1) = \boldsymbol{u}(k+l-1) - \boldsymbol{u}(k+l-2)$, $l = 1, \ldots, N$.

Combining (1.19) and (1.21) we get:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}(\Delta\boldsymbol{u}(k) + \boldsymbol{u}(k-1)) \\ &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k-1) + \boldsymbol{B}\Delta\boldsymbol{u}(k) \\ \boldsymbol{x}(k+2) &= \boldsymbol{A}^2\boldsymbol{x}(k) + \boldsymbol{A}\boldsymbol{B}(\Delta\boldsymbol{u}(k) + \boldsymbol{u}(k-1)) + \\ &+ \boldsymbol{B}(\Delta\boldsymbol{u}(k+1) + \Delta\boldsymbol{u}(k) + \boldsymbol{u}(k-1)) \\ &= \boldsymbol{A}^2\boldsymbol{x}(k) + (\boldsymbol{A}\boldsymbol{B} + \boldsymbol{B})\boldsymbol{u}(k-1) + (\boldsymbol{A}\boldsymbol{B} + \boldsymbol{B})\Delta\boldsymbol{u}(k) + \boldsymbol{B}\Delta\boldsymbol{u}(k+1) \\ &\vdots \\ \boldsymbol{x}(k+N) &= \boldsymbol{A}^N\boldsymbol{x}(k) + \boldsymbol{A}^{N-1}\boldsymbol{B}(\Delta\boldsymbol{u}(k) + \boldsymbol{u}(k-1)) + \\ &+ \cdots + \boldsymbol{B}(\Delta\boldsymbol{u}(k+N-1) + \cdots + \Delta\boldsymbol{u}(k) + \boldsymbol{u}(k-1)) \\ &= \boldsymbol{A}^N\boldsymbol{x}(k) + (\boldsymbol{A}^{N-1}\boldsymbol{B} + \cdots + \boldsymbol{B})\boldsymbol{u}(k-1) + \\ &+ (\boldsymbol{A}^{N-1}\boldsymbol{B} + \cdots + \boldsymbol{B})\Delta\boldsymbol{u}(k) + \cdots + \boldsymbol{B}\Delta\boldsymbol{u}(k+N-1) \end{aligned} \tag{1.22}$$

The equations 1.22 can be written in a matrix form as:

$$\begin{aligned} \begin{bmatrix} \boldsymbol{x}(k+1) \\ \vdots \\ \boldsymbol{x}(k+N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} \boldsymbol{B} \\ \vdots \\ \sum_{i=0}^{N-1} \boldsymbol{A}^i\boldsymbol{B} \end{bmatrix} \boldsymbol{u}(k-1) + \\ + \begin{bmatrix} \boldsymbol{B} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{N-1} \boldsymbol{A}^i\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{u}(k) \\ \vdots \\ \Delta\boldsymbol{u}(k+N-1) \end{bmatrix} \end{aligned} \tag{1.23}$$

The equations 1.20 and 1.23 describing the state evolution over a prediction horizon N are of course equivalent and for the system's output in both representations it holds

in a matrix form:

$$\begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ y(k+N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{C} & 0 & \cdots & 0 \\ 0 & \boldsymbol{C} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{C} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{x}(k+2) \\ \vdots \\ \boldsymbol{x}(k+N) \end{bmatrix} \tag{1.24}$$

1.5.2 Optimal Control Problem

In order to obtain the control law a cost function is needed. In general, the cost function is formulated in regard to the output difference to a specific reference signal and the required control effort. Such a function is of the form:

$$J(\boldsymbol{x}(k), \boldsymbol{U}(k)) = \sum_{l=k}^{k+N-1} P(\boldsymbol{x}(l+1|k), \boldsymbol{u}(l|k)) \tag{1.25}$$

where $\boldsymbol{U}(k) = \begin{bmatrix} \boldsymbol{u}^T(k) & \boldsymbol{u}^T(k+1) & \cdots & \boldsymbol{u}^T(k+N-1) \end{bmatrix}^T$ is the control input sequence and P is a function based on the norm:

$$\|\cdot\|_p := \left(\sum_{i=1}^{n} |\cdot|^p \right)^{1/p}, p \geq 1, p \in \mathbb{R}.$$

The most commonly used norms in MPC are $p = 1$, $p = \infty$, which produce a linear function, and $p = 2$, which produces a quadratic function.

The aim of the optimization problem is, using the formulated cost function J, to find the control sequence $\boldsymbol{U}(k)$ that results in the best performance of the system:

$$\begin{aligned} \text{minimize} \quad & J(k) \\ \text{subject to} \quad & \boldsymbol{x}(l+1) = f(\boldsymbol{x}(l), \boldsymbol{u}(l)) \\ & l = k, k+1, \ldots, N-1 \end{aligned} \tag{1.26}$$

The solution of this problem is the *optimal control sequence* at step k, $\boldsymbol{U}^*(k) = \begin{bmatrix} \boldsymbol{u}^{*T}(k) & \boldsymbol{u}^{*T}(k+1) & \cdots & \boldsymbol{u}^{*T}(k+N-1) \end{bmatrix}^T$

1.5.3 Receding Horizon Policy

The basic concept of *receding horizon policy* is as follows. Having obtained the optimal control sequence $\boldsymbol{U}^*(k)$ at step k, only the first term $\boldsymbol{u}^*(k)$ is used as the control input and the rest are discarded. Afterwards, the procedure is repeated, calculating a new optimal control sequence $\boldsymbol{U}^*(k+1) = \begin{bmatrix} \boldsymbol{u}^{*T}(k) & \boldsymbol{u}^{*T}(k+1) & \cdots & \boldsymbol{u}^{*T}(k+N-1) \end{bmatrix}^T$ for step $k+1$ using new state measurements. In this way, plant uncertainties and distur-

bances can be taken into account in the future control actions, providing feedback.[20]

Algorithm 1.5 MPC algorithm

1: Obtain state measurements $\boldsymbol{x}(k)$
2: Based on $\boldsymbol{x}(k)$, solve optimization problem (1.26)
3: Obtain optimal control sequence $\boldsymbol{U}^*(k) = \begin{bmatrix}\boldsymbol{u}^{*T}(k) & \boldsymbol{u}^{*T}(k+1) & \cdots & \boldsymbol{u}^{*T}(k+N-1)\end{bmatrix}^T$
4: Apply only $\boldsymbol{u}^*(k)$ to the plant
5: Update $k := k+1$
6: Go to step "1"

Chapter 2

DC-DC Buck Converter

2.1 Model of the DC-DC Buck Converter

2.1.1 Continuous-Time Model

The dc-dc buck converter, shown in fig. 2.1, reduces the dc input voltage $v_{in}(t)$ to a lower dc output voltage $v_o(t)$. The converter consists of an output load R, an inductor L with internal resistance R_l, which, depending on the conduction mode, stores and delivers energy to the load, and a capacitor C with equivalent series resistance R_c connected in parallel with the load in order to continuously provide voltage to the output. There are two power semiconductors; the switch S which is controllable and the diode D.

When the switch S is *on* ($S = 1$) energy is stored to the inductor and the inductor current $i_l(t)$ is increasing. When the switch is *off* ($S = 0$) the energy stored in the inductor flows to the output causing the inductor current $i_l(t)$ to decrease. If the inductor current becomes zero ($i_l(t) = 0$) then both the switch S and the diode D are *off* and the converter operates in discontinuous conduction mode (DCM). Otherwise (namely when the inductor current is positive, $i_l(t) > 0$), the converter operates in continuous conduction mode (CCM). In this thesis only the CCM is taken into account for the system's modelling. Figures 2.2, 2.3, 2.4 describe graphically the modes of operation of the buck converter. In these figures the parasitic resistances are not shown for simplicity reasons.

The continuous-time state-space equations, as they are defined in [14], are:

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = \boldsymbol{A_c}\boldsymbol{x}(t) + \boldsymbol{B_c}u(t) \tag{2.1a}$$

$$y(t) = \boldsymbol{C_c}\boldsymbol{x}(t) \tag{2.1b}$$

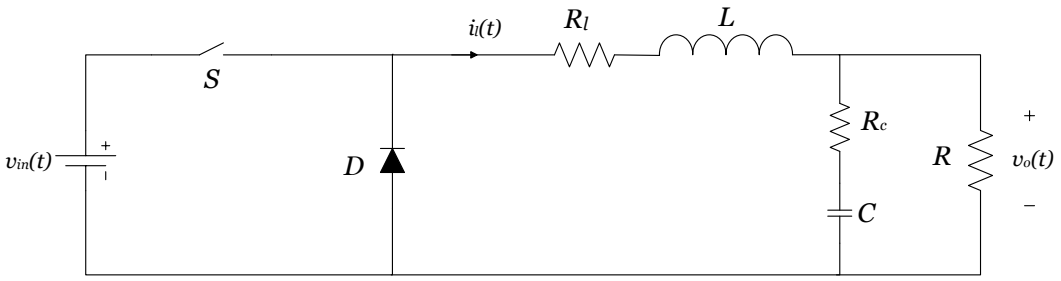


Figure 2.1: DC-DC Buck Converter

where

$$\boldsymbol{x}(t) = \begin{bmatrix} i_l(t) & v_o(t) \end{bmatrix}^T$$

is the state vector, $i_l(t)$ is the inductor current, $v_o(t)$ is the output voltage,

$$y(t) = v_o(t)$$

is the system output,

$$\boldsymbol{A_c} = \begin{bmatrix} -\frac{R_l}{L} & -\frac{1}{L} \\ R\frac{L-R_cR_lC}{(R+R_c)CL} & -\frac{L+R_cRC}{(R+R_c)CL} \end{bmatrix}, \boldsymbol{B_c} = \frac{v_{in}}{L}\begin{bmatrix} 1 \\ \frac{RR_c}{R+R_c} \end{bmatrix}, \boldsymbol{C_c} = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

are the system matrices and $u(t)$ is the control input.

There are two ways to define the variable u depending on the given physical meaning. The control input u can either denote the state of the switch S or the duty cycle. In the first case u is discrete, since the switch state is either *off* or *on*, with $u \in \{0, 1\}$. In the second case it is continuous since u is the duty cycle, namely $u = t_{on}/T$, where t_{on} is the interval the switch stays *on* over one switching period T. Here it holds $u \in [0, 1]$.

Based on these two different definitions of u, two different optimization problems are implemented for the buck converter in Section 2.2.

2.1.2 Discrete-Time Model

As already mentioned, MPC is a discrete-time controller and for this reason the continuous-time equations (2.1) need to be discretised. For the discretisation the Euler method is used

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} \approx \frac{\boldsymbol{x}(k+1) - \boldsymbol{x}(k)}{T_s}$$

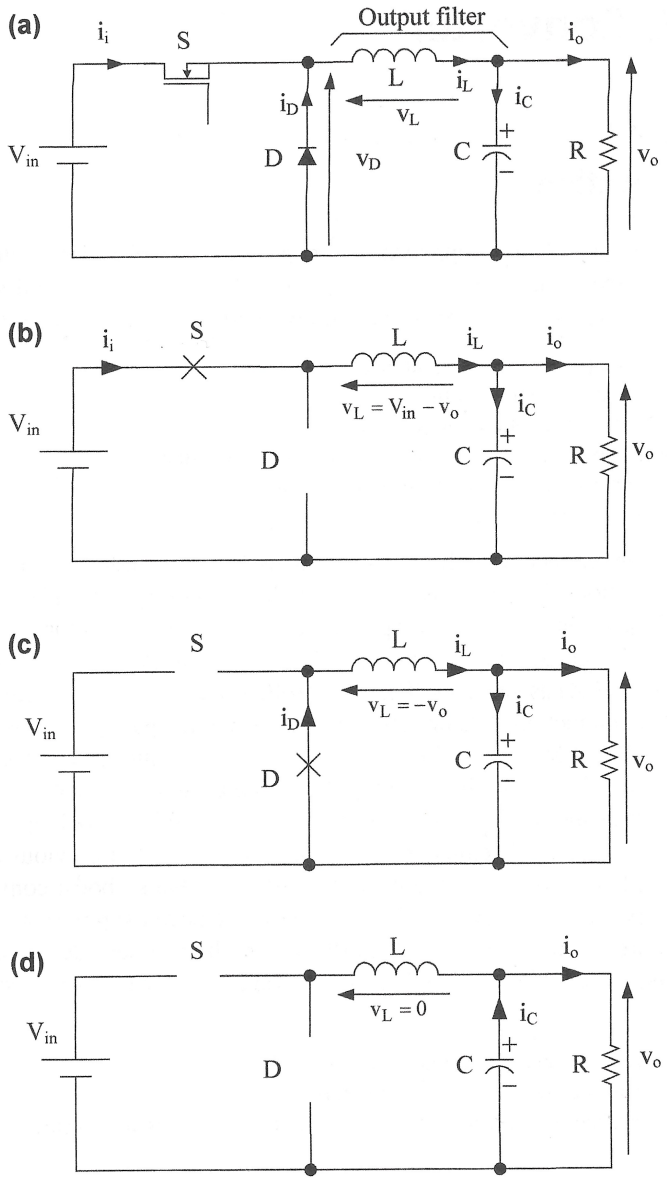


Figure 2.2: Buck converter and equivalent circuits of modes of operation. (a)Power (b) equivalent circuit when the switch is on (Mode I); (c) equivalent circuit when the diode is conducting (Mode II); (d) equivalent circuit when none of the semiconductors conduct (Mode III). [25]

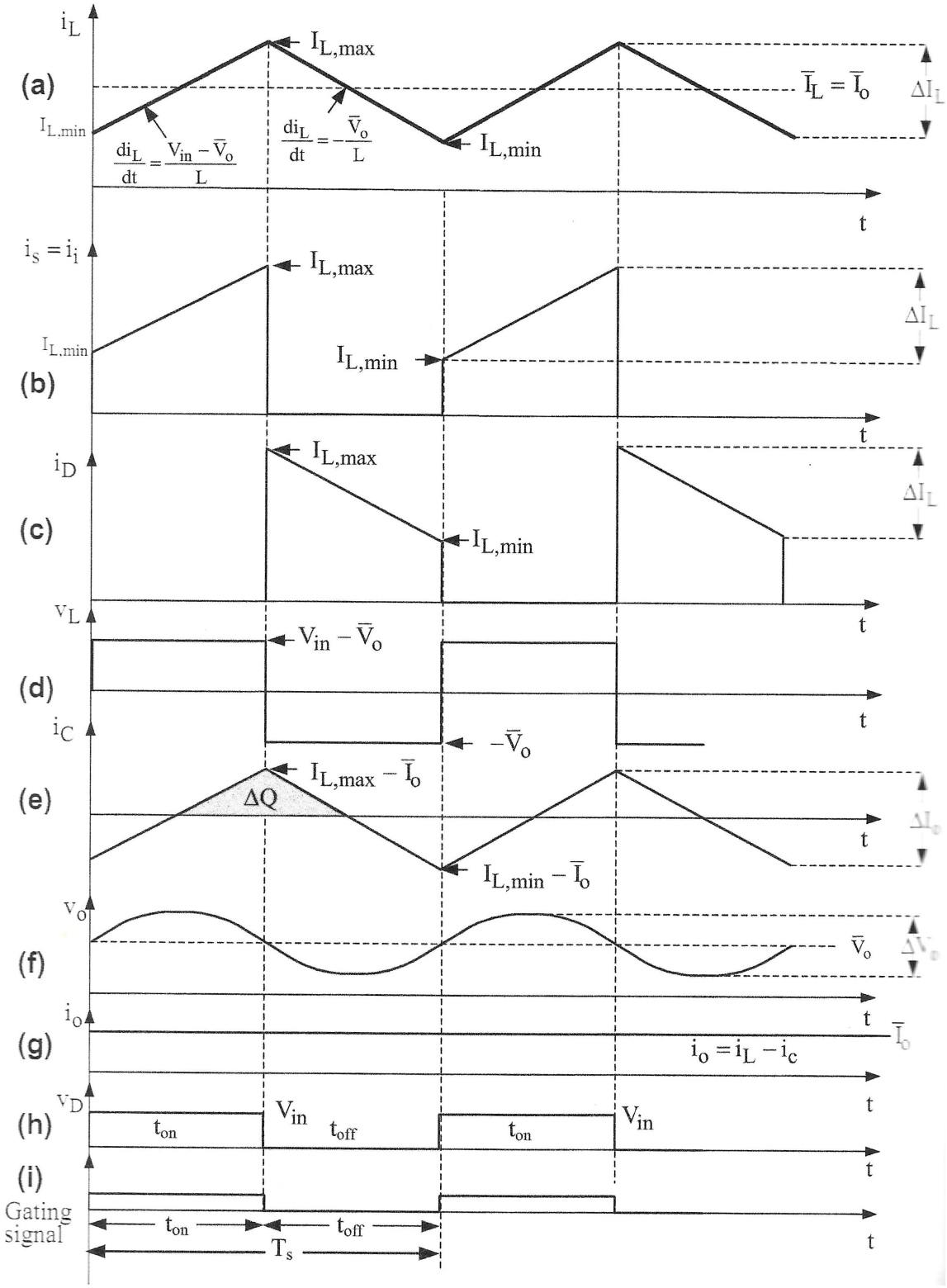


Figure 2.3: Buck converter waveforms when inductor current is continuous. (a) Inductor current; (b) switch S current; (c) diode D current; (d) inductor voltage; (e) capacitor current; (f) output voltage; (g) output current; (h) diode voltage; (i) gating signal.[25]

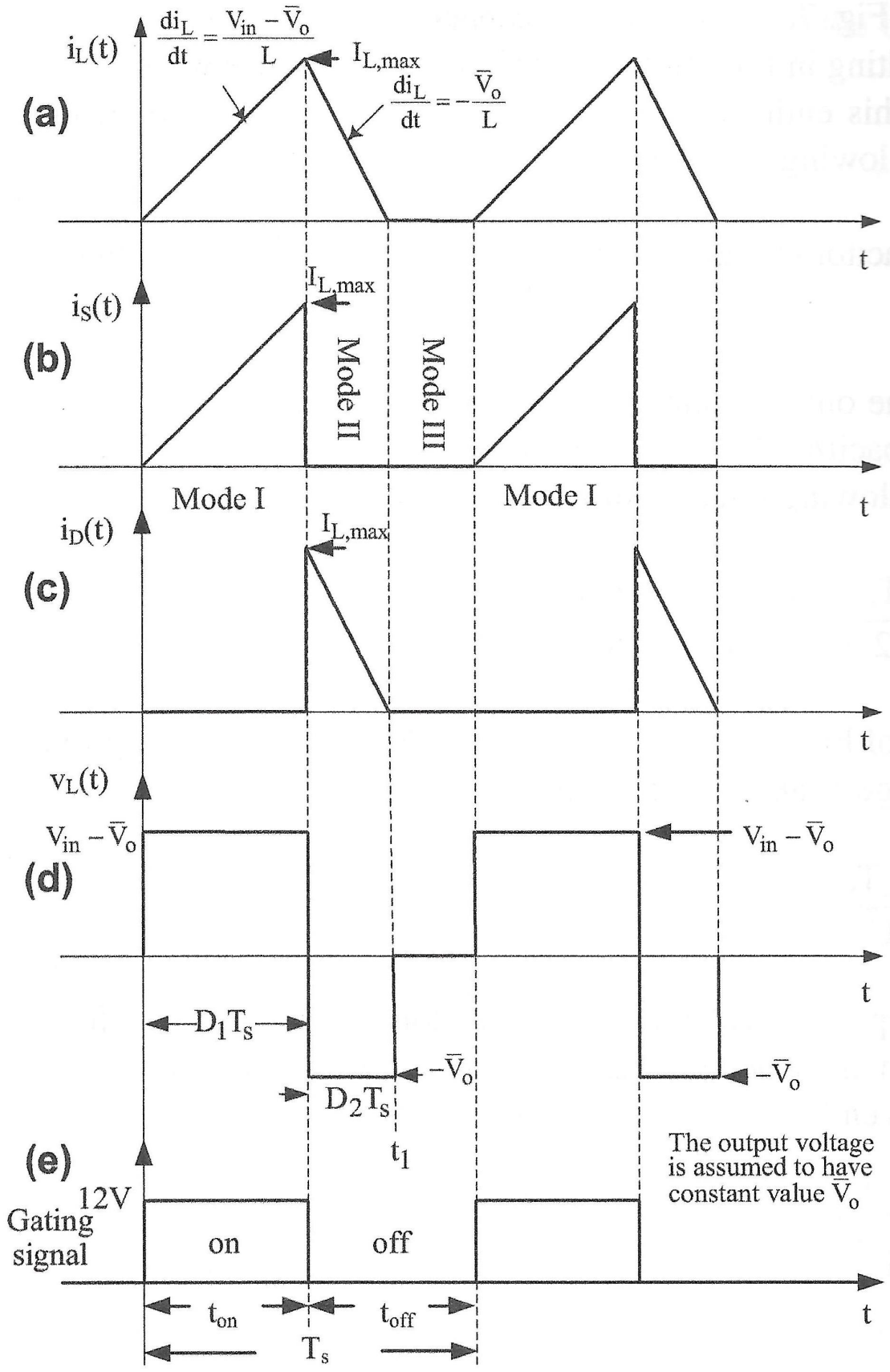


Figure 2.4: Buck converter waveforms when inductor current is discontinuous. (a) Inductor current; (b) switch S current; (c) diode D current; (d) voltage across diode D; (e) gating signal. [25]

where T_s is the sampling period. The discrete-time state-space equations are:

$$\boldsymbol{x}(k+1) = \boldsymbol{A_d}\boldsymbol{x}(k) + \boldsymbol{B_d}u(k) \tag{2.2a}$$

$$y(k) = \boldsymbol{C_d}\boldsymbol{x}(k) \tag{2.2b}$$

The matrices of the discrete time system are $\boldsymbol{A_d} = \boldsymbol{I} + \boldsymbol{A_c}T_s$, $\boldsymbol{B_d} = \boldsymbol{B_c}T_s$ and $\boldsymbol{C_d} = \boldsymbol{C_c}$ where $\boldsymbol{I}$ is the identity matrix of size two and $\boldsymbol{A_c}$, $\boldsymbol{B_c}$, $\boldsymbol{C_c}$ are the matrices described in Section 2.1.1

2.2 Optimal Control of Buck Converter

The main control objective is to regulate the output voltage $v_o(k)$ to the desired voltage reference $v_{o,\text{ref}}$, minimizing the voltage error, by appropriately manipulating the switch S.

In this section two different implementations of the control problem are presented. In the first implementation a mixed-integer quadratic optimization problem is formulated, which is solved with enumeration technique, whereas in the second one a quadratic optimization problem.

2.2.1 Objective Function

The formulation of the *objective function* is achieved with respect to the prediction horizon N and the main control aim, which is to force the output voltage to track its reference $v_{o,\text{ref}}$ or, in other words, to minimize the output voltage error. Therefore, the *objective function* is:

$$J(k) = \sum_{l=k}^{k+N-1} \left(||v_o(l+1|k) - v_{o,\text{ref}}||_2^2 + \lambda ||\Delta u(l|k)||_2^2 \right) \tag{2.3}$$

The first term of the objective function expresses the minimization of the voltage error whereas the second term

$$\Delta u(l|k) = u(l|k) - u(l-1|k) \tag{2.4}$$

penalizes the difference between two consecutive control values. $\lambda > 0$ is a weighting factor that sets a trade-off between the two function terms.

The terms of the objective function (2.3) can be written in a vector form. As it is

described in Chapter 1, for an N–step prediction horizon it holds:

$$\begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{x}(k+2) \\ \vdots \\ \boldsymbol{x}(k+N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A_d} \\ \boldsymbol{A_d}^2 \\ \vdots \\ \boldsymbol{A_d}^N \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} \boldsymbol{B_d} \\ \boldsymbol{B_d} + \boldsymbol{A_d}\boldsymbol{B_d} \\ \vdots \\ \sum_{i=0}^{N-1} \boldsymbol{A_d}^i \boldsymbol{B_d} \end{bmatrix} u(k-1) + \\ + \begin{bmatrix} \boldsymbol{B_d} & 0 & \cdots & 0 \\ \boldsymbol{B_d} + \boldsymbol{A_d}\boldsymbol{B_d} & \boldsymbol{B_d} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{N-1} \boldsymbol{A_d}^i \boldsymbol{B_d} & \sum_{i=0}^{N-2} \boldsymbol{A_d}^i \boldsymbol{B_d} & \cdots & \boldsymbol{B_d} \end{bmatrix} \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N-1) \end{bmatrix} \tag{2.5}$$

$$\boldsymbol{Y} = \begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ y(k+N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{C_d} & 0 & \cdots & 0 \\ 0 & \boldsymbol{C_d} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{C_d} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{x}(k+2) \\ \vdots \\ \boldsymbol{x}(k+N) \end{bmatrix} \tag{2.6}$$

Combining equations (2.5) and (2.6), the output vector $\boldsymbol{Y}$ can be written in the form:

$$\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{Q}u(k-1) + \boldsymbol{S}\Delta\boldsymbol{U} \tag{2.7}$$

where

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{C_d}\boldsymbol{A_d} \\ \boldsymbol{C_d}\boldsymbol{A_d}^2 \\ \vdots \\ \boldsymbol{C_d}\boldsymbol{A_d}^N \end{bmatrix}, \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{C_d}\boldsymbol{B_d} \\ \boldsymbol{C_d}\boldsymbol{B_d} + \boldsymbol{C_d}\boldsymbol{A_d}\boldsymbol{B_d} \\ \vdots \\ \sum_{i=0}^{N-1} \boldsymbol{C_d}\boldsymbol{A_d}^i \boldsymbol{B_d} \end{bmatrix}, \tag{2.8}$$

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{C_d}\boldsymbol{B_d} & 0 & \cdots & 0 \\ \boldsymbol{C_d}\boldsymbol{B_d} + \boldsymbol{C_d}\boldsymbol{A_d}\boldsymbol{B_d} & \boldsymbol{C_d}\boldsymbol{B_d} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=0}^{N-1} \boldsymbol{C_d}\boldsymbol{A_d}^i \boldsymbol{B_d} & \sum_{i=0}^{N-2} \boldsymbol{C_d}\boldsymbol{A_d}^i \boldsymbol{B_d} & \cdots & \boldsymbol{C_d}\boldsymbol{B_d} \end{bmatrix} \tag{2.9}$$

$$\Delta\boldsymbol{U} = \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(k+N-1) \end{bmatrix}. \tag{2.10}$$

The sizes of the matrices $\boldsymbol{P}$, $\boldsymbol{Q}$, $\boldsymbol{S}$ and $\Delta\boldsymbol{U}$ are $N \times 2$, $N \times 1$, $N \times N$ and $N \times 1$, respectively.

Thus, the first term of the objective function (2.3) can be now rewritten as:

$$\sum_{l=k}^{k+N-1} ||v_o(l+1|k) - v_{o,\text{ref}}||_2^2 = ||\,\boldsymbol{Y} - \boldsymbol{V_{ref}}||_2^2 = ||\boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{Q}u(k-1) + \boldsymbol{S}\Delta\boldsymbol{U} - \boldsymbol{V_{ref}}||_2^2$$

with $\boldsymbol{V_{ref}} = \begin{bmatrix} v_{o,\text{ref}} & v_{o,\text{ref}} & \cdots & v_{o,\text{ref}} \end{bmatrix}^T$ a $N \times 1$ vector. For the second term, it can be easily perceived that

$$\sum_{l=k}^{k+N-1} ||\Delta u(l|k)||_2^2 = ||\Delta\,\boldsymbol{U}||_2^2.$$

The reformulated objective function, with its terms expressed in vector form, is

$$J(k) = ||\boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{Q}u(k-1) + \boldsymbol{S}\Delta\,\boldsymbol{U} - \boldsymbol{V_{ref}}||_2^2 + \lambda||\Delta\,\boldsymbol{U}||_2^2. \tag{2.11}$$

In order to obtain the control input at time instant kT_s, the objective function (2.11) needs to be minimized over the optimization variable, which is the control sequence $\boldsymbol{U}(k) = \begin{bmatrix} u(k) & u(k+1) & \cdots & u(k+N-1) \end{bmatrix}^T$. The fact that the elements of $\boldsymbol{U}(k)$ can denote either the state of the switch S, being binary, or the duty cycle, being continuous variables, leads to the formulation of two optimization problems, one for each representation, for the reasons stated in Chapter 1.

2.2.2 Mixed-Integer Quadratic Optimization Problem for Buck Converter

Considering the case where the control input is $u \in \{0, 1\}$, the control problem is of the form:

$$\begin{aligned} &\text{minimize} && J(k) \\ &\text{subject to} && (2.2). \end{aligned} \tag{2.12}$$

This is a *mixed-integer quadratic optimization problem* and in order to be solved an enumeration strategy is used. The objective function is evaluated for the 2^N different, possible control sequences $\boldsymbol{U}(k)$. Out of these 2^N control sequences, the one that results in the smallest (minimum) value for $J(k)$ is selected. This sequence $\boldsymbol{U}^*(k)$ is the optimal solution of the control problem, given by

$$\boldsymbol{U}^*(k) = \arg\min J(k) \tag{2.13}$$

The first element $u^*(k)$ of the optimal sequence $\boldsymbol{U}^*(k)$ is applied to the switch as control input and the rest are discarded. In the next time instant $(k+1)T_s$ new measurements are taken and the process to determine the control input is repeated.

Since the output of the controller is the state of the switch, no other stage between

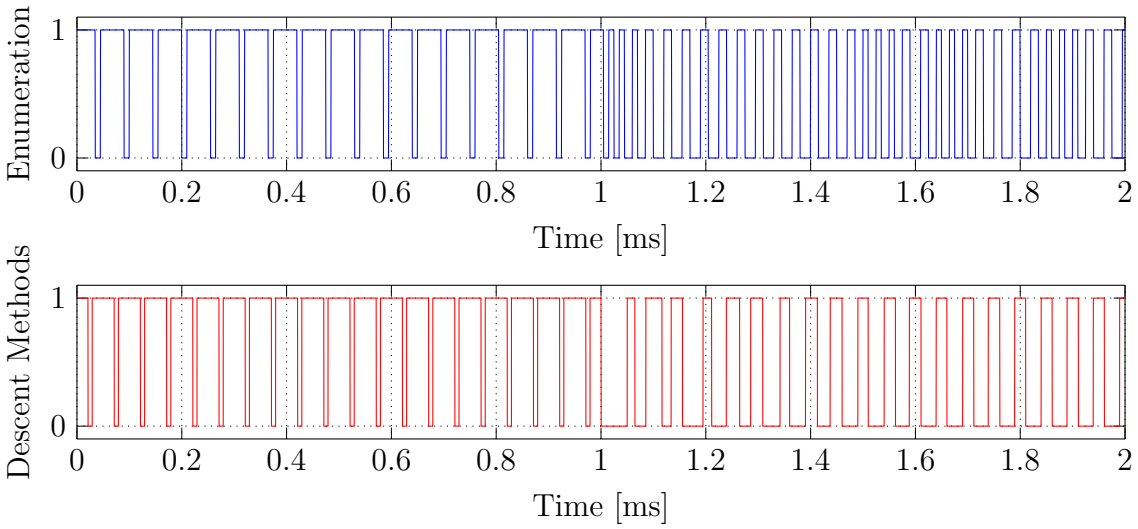


Figure 2.5: Switching frequency comparison of enumeration and descent methods. At time 1ms, where input voltage gets doubled and the operating point changes, the switching frequency for the enumeration technique increases whereas for the descent methods the switching frequency stays the same for the two operating points and only the duty cycle changes.

the controller and the switch is needed to manipulate the switch successfully. This results in a control sequence that causes the switch S to operate with variable frequency (in contrast to the method of Section 2.2.3), Nonetheless, the switching frequency f_{sw} has an upper bound that depends on the sampling time T_s,

$$f_{sw} \leq \frac{1}{2T_s}.$$

Looking at the objective function (2.11), it be easily perceived that the second term, which penalizes the difference between two consecutive control actions, can be used as a way to manipulate the switching frequency through the weighting factor λ.

2.2.2.1 Control Algorithm

The proposed enumeration strategy is summarised in the Algorithm 2.1.

2.2.3 Quadratic Optimization Problem for Buck Converter

Considering the case where the control input is $u \in [0, 1]$, the control problem is of the form:

$$\begin{aligned} &\text{minimize} && J(k) \\ &\text{subject to} && (2.2) \end{aligned} \tag{2.14}$$

This is a *quadratic optimization problem* and therefore the use of a descent method is appropriate of the minimization of $J(k)$. Descent minimization methods, starting

Algorithm 2.1 Voltage MPC algorithm with Enumeration Strategy for Buck Converter

function BUCKMPCENUM($\boldsymbol{x}(k)$,$u(k-1)$)
  $J^* = \infty$, $u^*(k) = 0$
  for all $\boldsymbol{U}(k)$ over N **do**
    $J = 0$
    for $l = k$ **to** $k + N - 1$ **do**
      $\Delta u(l) = u(l) - u(l-1)$
    end for
    $J = ||\boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{Q}u(k-1) + \boldsymbol{S}\Delta\boldsymbol{U} - \boldsymbol{V_{ref}}||_2^2 + \lambda||\Delta\boldsymbol{U}||_2^2$
    if $J < J^*$ **then**
      $J^* = J$
      $u^*(k) = \boldsymbol{U}(1)$
    end if
  end for
end function

from an initial point (control sequence) $\boldsymbol{U}(k)$, converge to the control sequence that minimizes the objective function (2.11), calculating the optimal solution

$$\boldsymbol{U}^*(k) = \arg\min J(k). \tag{2.15}$$

Out of the the N elements of the optimal solution $\boldsymbol{U}^*(k)$ only the first element $u^*(k)$ is applied on the switch as control input. New measurements are taken in the next time instant $(k+1)T_s$ and the process to determine the control input is repeated.

In order to increase the rate of convergence, the last optimal solution $\boldsymbol{U}^*(k)$ is used as the new starting point of the iterative method. This choice is made based on the fact that the control input $u(k-1)$ is actually the duty cycle. When the converter operates in steady state the duty cycle remains unchanged from time instance kT_s to $(k+1)T_s$. Consequently the controller's output (i.e. $u^*(k)$) does not change (or has only a small change) and the use of the last optimal solution as the starting point of the iterative method means that few (or even none) iterations are required for the method to converge.

In this method the use of a PWM stage between the controller and the switch S is necessary to successfully translate the controller output to the pulses that drive the switch, in contrast to the method analysed in Section 2.2.2, where no such stage is needed. In this stage the controller output is compared to a triangle wave to produce the needed pluses (fig. 2.6). This means that the switching frequency in this case is fixed and depends only on the frequency of the triangle wave. Choosing the period of the wave to be equal to the sampling period T_s, the switching frequency is

$$f_{sw} = \frac{1}{T_s}. \tag{2.16}$$

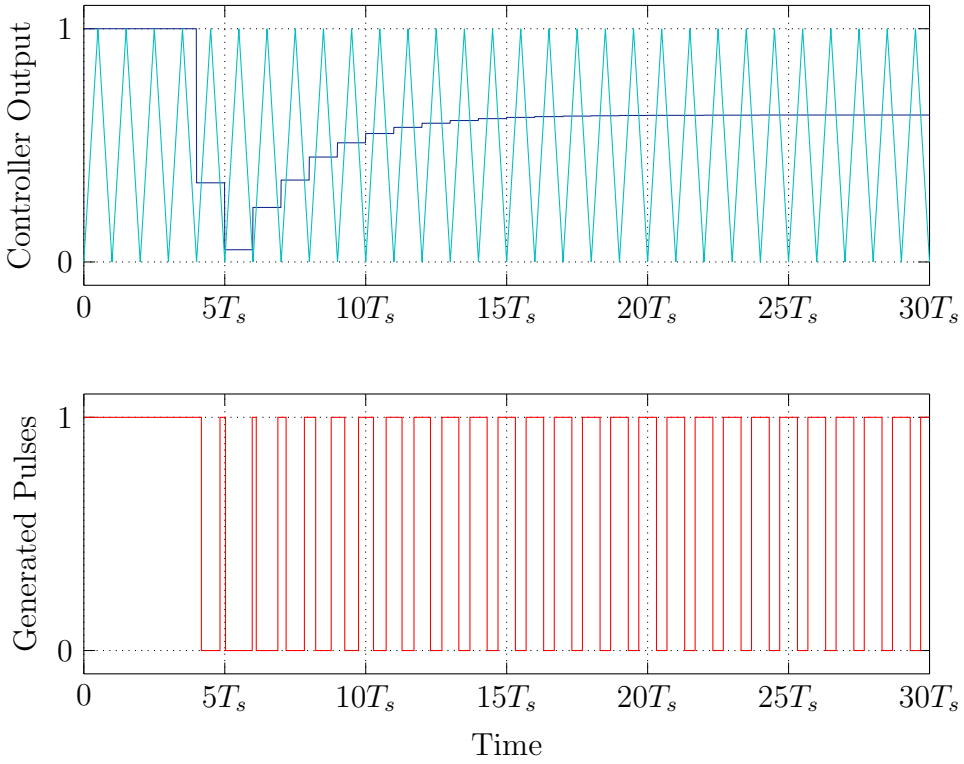


Figure 2.6: The controller's output(dark blue line) is compared to a triangular wave (light blue line) to create the pulses (red line) that drive the switch S.

To solve this optimization problem, two descent methods are presented; *gradient descent method* and *Newton's method*.

2.2.3.1 Gradient Descent Method

The objective function of the control problem in question is (2.11) and is repeated here for the reader's convenience:

$$J(k) = ||\boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{Q}u(k-1) + \boldsymbol{S}\Delta\boldsymbol{U} - \boldsymbol{V_{ref}}||_2^2 + \lambda||\Delta\boldsymbol{U}||_2^2.$$

As mentioned in Section 1.4.2.1, *gradient descent method* minimizes a convex function f using as search direction the function's gradient ∇f. This means that the calculation of ∇J is needed to proceed.

$J(k)$ is a function of $\boldsymbol{U}(k) = \begin{bmatrix} u(k) & u(k+1) & \cdots & u(k+N-1) \end{bmatrix}^T$, so its gradient ∇J is a $N \times 1$ vector, the entries of which are the partial derivative of $J(k)$ with respect to $\boldsymbol{U}(k)$ and is given by:

$$\nabla J = 2\boldsymbol{E}^T(\boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{Q}u(k-1) + \boldsymbol{S}\Delta\boldsymbol{U} - \boldsymbol{V_{ref}}) + 2\lambda\boldsymbol{F}^T\Delta\boldsymbol{U} \tag{2.17}$$

where

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{C_d B_d} & 0 & \cdots & 0 \\ \boldsymbol{C_d A_d B_d} & \boldsymbol{C_d B_d} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C_d {A_d}^{N-1} B_d} & \boldsymbol{C_d {A_d}^{N-2} B_d} & \cdots & \boldsymbol{C_d B_d} \end{bmatrix}, \boldsymbol{F} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}$$

with the matrices $\boldsymbol{P}$,$\boldsymbol{Q}$,$\boldsymbol{S}$ and $\boldsymbol{V_{ref}}$ been those described in Section 2.2.1. The matrices $\boldsymbol{E}$, $\boldsymbol{F}$ are both of $N \times N$ size. The gradient ∇J is calculated analytically in Appendix A.

The proposed gradient descent method is summarised in Algorithm 2.2 where $\alpha \in (0, 1)$ is the step size and *StopTolerence* is a small positive number which expresses the accuracy of the calculation.

Algorithm 2.2 Voltage MPC algorithm with gradient descent method for buck converter

function BUCKMPCGRAD($\boldsymbol{x}(k), \boldsymbol{U}(k-1)$)
  $\boldsymbol{U_{old}} = \boldsymbol{U}(k-1)$
  $\nabla \boldsymbol{J} = f(\boldsymbol{x}(k), \boldsymbol{U_{old}})$
  $\boldsymbol{U_{new}} = \boldsymbol{U_{old}} - \alpha \nabla \boldsymbol{J}$
  while $\| \boldsymbol{U_{new}} - \boldsymbol{U_{old}} \|_2 \geq StopTolerance$ & $\boldsymbol{U_{new}} \succeq 0$ & $\boldsymbol{U_{new}} \preceq 1$ **do**
    $\boldsymbol{U_{old}} = \boldsymbol{U_{new}}$
    $\nabla \boldsymbol{J} = f(\boldsymbol{x}(k), \boldsymbol{U_{old}})$
    $\boldsymbol{U_{new}} = \boldsymbol{U_{old}} - \alpha \nabla \boldsymbol{J}$
  end while
  $u^*(k) = \boldsymbol{U_{old}}(1)$
end function

2.2.3.2 Newton's Method

Newton's Method for minimization uses the first and the second order (partial) derivatives of the function which is minimized (see Section 1.4.2.2). The first order partial derivatives of the objective function $J(k)$ are given by equation (2.17) in Section 2.2.3.1. The second order partial derivatives of $J(k)$ are:

$$\begin{aligned} \nabla^2 J(k) &= \begin{bmatrix} \frac{\partial^2 J(k)}{\partial u^2(k)} & \frac{\partial^2 J(k)}{\partial u(k) \partial u(k+1)} & \cdots & \frac{\partial^2 J(k)}{\partial u(k) \partial u(k+N-1)} \\ \frac{\partial^2 J(k)}{\partial u(k+1) \partial u(k)} & \frac{\partial^2 J(k)}{\partial u^2(k+1)} & \cdots & \frac{\partial^2 J(k)}{\partial u(k+1) \partial u(k+N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J(k)}{\partial u(k+N-1) \partial u(k)} & \frac{\partial^2 J(k)}{\partial u(k+N-1) \partial u(k+1)} & \cdots & \frac{\partial^2 J(k)}{\partial u^2(k+N-1)} \end{bmatrix} \\ &= 2\boldsymbol{E}^T \boldsymbol{E} + 2\lambda \boldsymbol{F}^T \boldsymbol{F} \end{aligned} \tag{2.18}$$

From 2.18 can be perceived that $\nabla^2 J(k)$ depends on the values of the circuit's elements and the input voltage v_s and not on $\boldsymbol{U}^*(k)$ and thus changes only when v_s changes as well. Consequently, $[\nabla^2 \boldsymbol{J}]^{-1}$ needs to be calculated only when v_s changes.

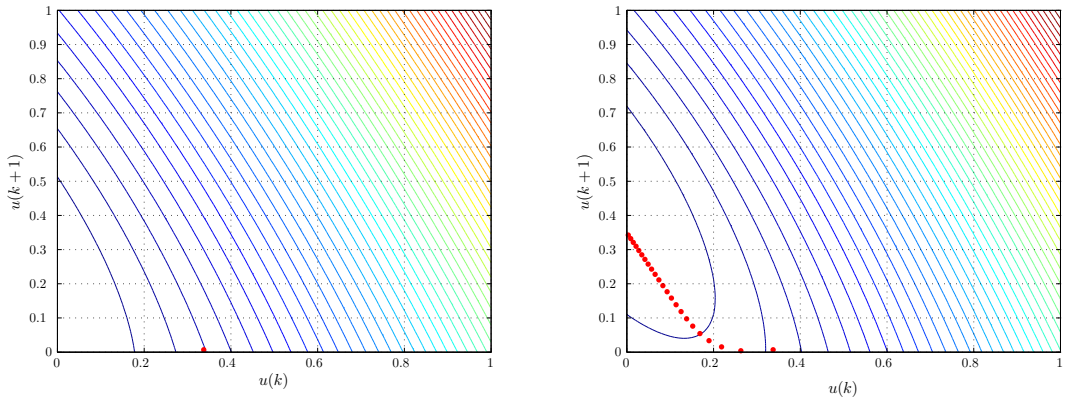
$\nabla^2 J$ as well as the gradient ∇J are calculated analytically in Appendix A.

The proposed Newton's method is summarised in Algorithm 2.3, where $\alpha \in (0, 1)$ is the step size and *StopTolerence* is a small positive number which expresses the accuracy of the calculation.

Algorithm 2.3 Voltage MPC algorithm with Newton's method for buck converter

function BUCKMPCNEWTON($\boldsymbol{x}(k), \boldsymbol{U}(k-1)$)
  $\nabla^2 \boldsymbol{J} = 2\boldsymbol{E}^T\boldsymbol{E} + 2\lambda \boldsymbol{F}^T\boldsymbol{F}$
  $\boldsymbol{U_{old}} = \boldsymbol{U}(k-1)$
  $\nabla \boldsymbol{J} = f(\boldsymbol{x}(k), \boldsymbol{U_{old}})$
  $\boldsymbol{U_{new}} = \boldsymbol{U_{old}} - \alpha[\nabla^2 \boldsymbol{J}]^{-1}\nabla \boldsymbol{J}$
  while $\frac{1}{2}[\nabla \boldsymbol{J}]^T[\nabla^2 \boldsymbol{J}]^{-1}\nabla \boldsymbol{J} \geq$ StopTolerance & $\boldsymbol{U_{new}} \succeq 0$ & $\boldsymbol{U_{new}} \preceq 1$ **do**
    $\boldsymbol{U_{old}} = \boldsymbol{U_{new}}$
    $\nabla \boldsymbol{J} = f(\boldsymbol{x}(k), \boldsymbol{U_{old}})$
    $\boldsymbol{U_{new}} = \boldsymbol{U_{old}} - \alpha[\nabla^2 \boldsymbol{J}]^{-1}\nabla \boldsymbol{J}$
  end while
  $u^*(k) = \boldsymbol{U_{old}}(1)$
end function

2.2.3.3 The Use of Descent Methods in Constrained Optimization



(a) At kT_s instance (b) At $(k+1)T_s$ instance

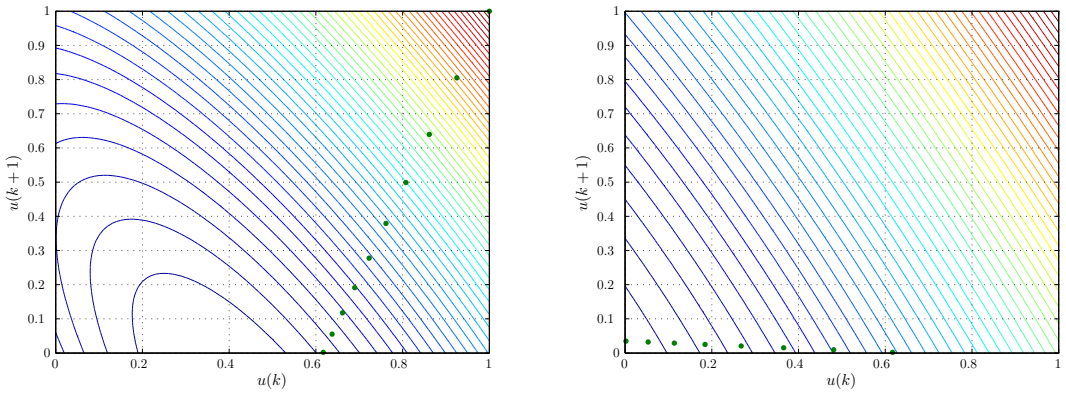
Figure 2.7: Gradient method during transient

In Section 1.4.2 it was stated that descent methods can be used for constrained minimization even though they are methods of unconstrained minimization. Figures 2.7 and 2.8 demonstrate why this is possible.

Figures 2.7 and 2.8 show the gradient's and Newton's convergence respectively at time instances kT_s and $(k+1)T_s$.

For both methods at time instance kT_s (Fig. 2.7a and 2.8a)the algorithm does not find the optimal solution, due to the use of gradient and Newton's methods in constrained optimization. At the next time instance $(k+1)T_s$ (Fig. 2.7b and 2.8b) the algorithm finds the optimal solution for both methods, due to the nature of MPC. Even though at kT_s the controller output is not correct, at the next time instant the controller manages to find the right one and correct the previous mistake.

This behaviour can be apparent only during transient. It may create an overshoot in output voltage, when the controller's output is greater than the optimal solution, or a delay in transient, when the controller's output is less than the optimal solution.



(a) At kT_s instance

(b) At $(k+1)T_s$ instance

Figure 2.8: Newton's method during transient

2.2.4 Load Uncertainties

Up to this point the analysis assumes that the load resistance is known and time-invariant, but in most applications this is not the case. Since the model of the converter depends on the load, its variations will cause a steady-state output voltage error. A Kalman filter [27] is used to provide offset-free tracking of the output voltage reference.

In order to model the effect the load variations have on the inductor current and the output voltage, the model of the converter is augmented by two integrating disturbance states i_e and v_e. The Kalman filter is used to estimate the augmented state vector

$$\boldsymbol{x_a} = \begin{bmatrix} i_l & v_o & i_e & v_e \end{bmatrix}^T. \tag{2.19}$$

The stochastic discrete-time state equation of the augmented model is

$$\boldsymbol{x_a}(k+1) = \boldsymbol{A_a}\boldsymbol{x_a}(k) + \boldsymbol{B_a}u(k) + \boldsymbol{w_1}(k) \tag{2.20}$$

and the measurement equation

$$\boldsymbol{x}(k) = \begin{bmatrix} i_l(k) \\ v_o(k) \end{bmatrix} = \boldsymbol{C_a x_a}(k) + \boldsymbol{w_2}(k). \tag{2.21}$$

The matrices are

$$\boldsymbol{A_a} = \begin{bmatrix} \boldsymbol{A_d} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix}, \boldsymbol{B_a} = \begin{bmatrix} \boldsymbol{B_d} \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{C_a} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} \end{bmatrix}$$

where $\boldsymbol{I}$ is the identity matrix of dimension two and $\boldsymbol{0}$ is a square zero matrix of dimension two. The variables $\boldsymbol{w_1} \in \mathbb{R}^4$, $\boldsymbol{w_2} \in \mathbb{R}^2$ express the process and the measurement noise respectively, with normal probability distributions. Their covariances are $E[\boldsymbol{w_1 w_1}^T] = \boldsymbol{W_1}$ and $E[\boldsymbol{w_2 w_2}^T] = \boldsymbol{W_2}$ and are positive semi-definite and positive definite respectively.

The equation of the estimated state $\hat{\boldsymbol{x}}_{\boldsymbol{a}}(k)$ is

$$\hat{\boldsymbol{x}}_{\boldsymbol{a}}(k+1) = \boldsymbol{A_a}\hat{\boldsymbol{x}}_{\boldsymbol{a}}(k) + \boldsymbol{K}\,\boldsymbol{C_a}(\boldsymbol{x_a}(k) - \hat{\boldsymbol{x}}_{\boldsymbol{a}}(k)) + \boldsymbol{B_a}u(k), \tag{2.22}$$

where $\boldsymbol{K}$ is the Kalman gain and is calculated based on the covariance matrices $\boldsymbol{W_1}$ and $\boldsymbol{W_2}$. These matrices are chosen in such a way to assign high credibility to the physical states (i.e. i_l and v_o) and low credibility to the disturbance states (i.e. i_e and v_e). Using the estimated states $\hat{i}_l$ and $\hat{v}_o$ as inputs to the controller, instead of the original measured states, and adjusting the output voltage reference to

$$\tilde{v}_{o,\text{ref}} = v_{o,\text{ref}} - \hat{v}_e, \tag{2.23}$$

it is possible to estimate the disturbances and remove their influence from the system.

2.3 Simulation Results

In this section simulation results with MATLAB/SIMULINK are presented to demonstrate how the controller works with each of the three methods, namely with enumeration, with gradient method and Newton's method.

The circuit parameters are $C = 220\mu F$, $R_C = 0.5\Omega$, $L = 250\mu H$ $R_L = 1\Omega$. The load resistance is $R = 10\Omega$. Initially the converter operates under nominal conditions, namely the input voltage is $v_{in} = 20V$ and the output voltage reference is $v_{o,\text{ref}} = 12V$. The prediction horizon is $N = 8$ and the sampling period for enumeration strategy is $T_{s,\text{enum}} = 5\mu s$.

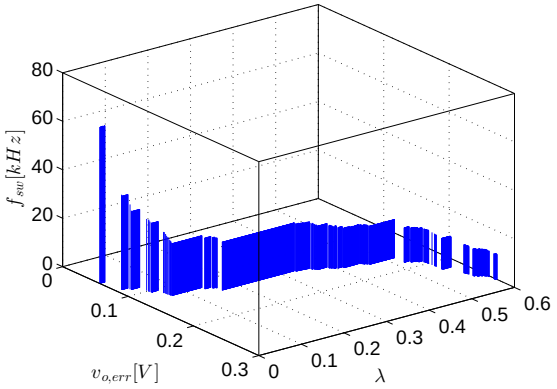


Figure 2.9: Weighting factor λ

As stated in Section 2.2.2, the switching frequency with the enumeration strategy is variable and changes for different operation points, while with the descent methods it is not (see Section 2.2.3). So, it is important that the converter operates with the same (or almost the same) switching frequency in all methods for the results to be comparable. Figure 2.9 shows how the weighting factor λ affects the switching frequency f_{sw} and the output voltage error $v_{o,\text{err}}$, where

$$v_{o,\text{err}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_{o,\text{ref}} - v_o(i))^2}, \tag{2.24}$$

where n is the number of samples needed to calculate $v_{o,\text{err}}$. An appropriate choice for the weighting factor is $\lambda \in (0.16, 0.33)$ where both the output voltage error and the switching frequency are constant ($f_{sw} \approx 20kHz$). The weighting factor is chosen $\lambda = 0.25$.

Thus, the sampling period for the iterative methods must be $T_{s,\text{desc}} = 50\mu s$, to correspond to the enumeration strategy switching frequency of $20kHz$.

The covariance matrices of the Kalman filter are chosen

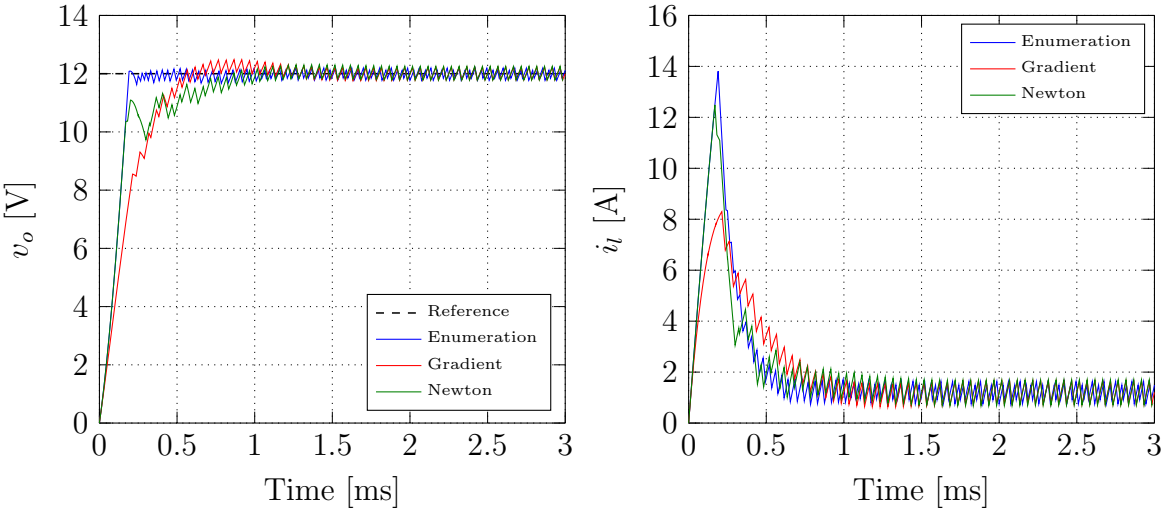
$$\boldsymbol{W_1} = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 50 \end{bmatrix}, \boldsymbol{W_2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.25}$$

2.3.1 Start up

Figure 2.10 shows the output voltage and the inductor current of the converter in nominal operation during start-up for the three methods. As can be seen, the controller increases the current to charge the capacitor to the reference voltage level as fast as possible and then the controller quickly restores the current to its nominal value. The

output voltage reaches its desired value faster with the enumeration technique (in about $t \approx 0.6$ms) than with gradient and Newton's methods (in about $t \approx 1.3$ms). This is due to the use of Kalman filter which, in the case of the iterative methods, causes the controller's output (the duty cycle) to oscillate delaying the system.

After the transient, the converter operates in steady state with a voltage ripple $v_{o,pp} \approx 0.5$V, for all methods.



(a) Output voltage during start up

(b) Induction current during start up

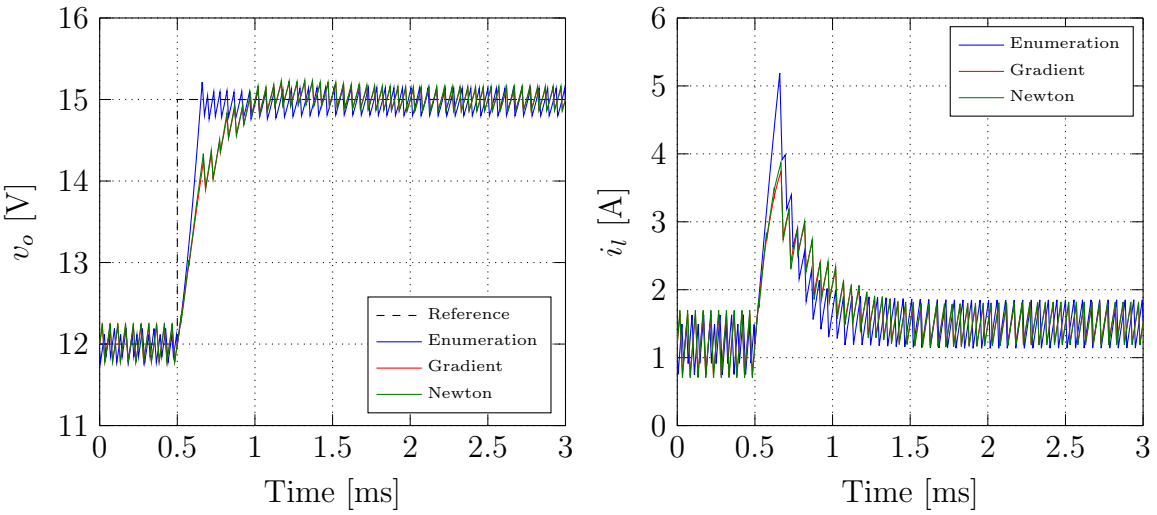
Figure 2.10: Start up

2.3.2 Step-up change in output voltage reference

With the converter operating in steady state in nominal conditions ($v_{in} = 20$V, $v_{o,\text{ref},1} = 12$V) a step-up change in the output voltage reference occurs at $t = 0.5$ms. The new voltage reference level is $v_{o,\text{ref},2} = 15$V. The system exhibits similar behaviour for all the methods (fig. 2.11); initially the current increases to bring the output voltage to the new reference level and then decreases to its normal value. Similarly to the start up results (fig. 2.10), the converter reaches the steady state faster with the enumeration technique than with the descent methods (0.5ms and 1ms respectively), a behaviour that is attributed to the Kalman filter.

2.3.3 Step-up change in input voltage

Next, with the converter operating in steady state at $v_{o,\text{ref},2} = 15$V, a step-up change in the input voltage takes place (at $t = 0.5$ms in fig. 2.12) from $v_{in} = 20$V to $v_{in} = 40$V. After the change, the output voltage follows the voltage reference for all methods, but the ripple is bigger. In the case of enumeration the switching frequency has also

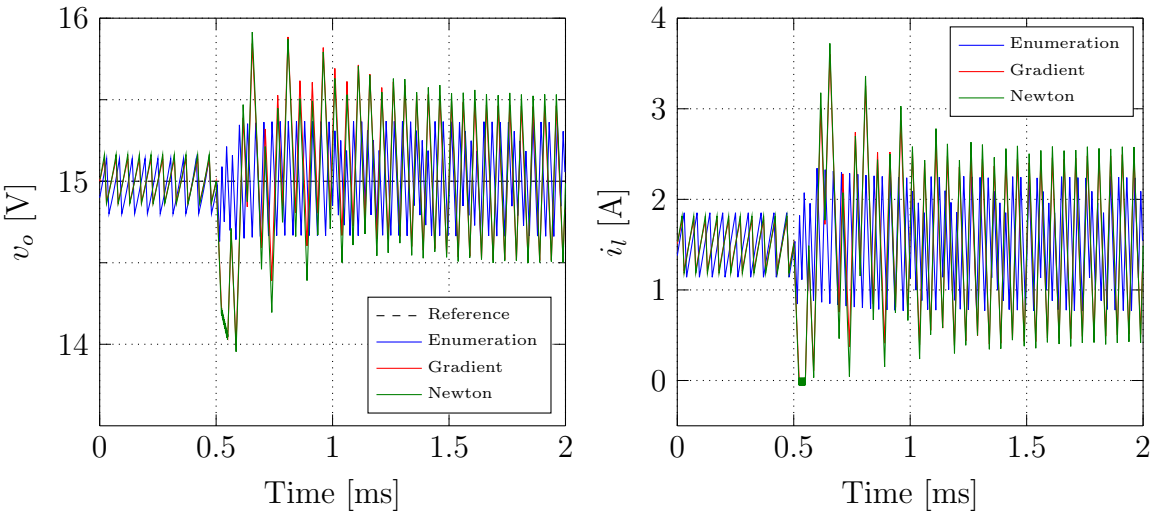


(a) Output voltage after step-up change in output voltage reference

(b) Induction current after a step-up change in output voltage reference

Figure 2.11: Step-up change in output voltage reference

changed from approximately 20kHz to 35kHz. This difference in switching frequency between enumeration and the descent methods explains why the ripple for gradient and Newton's methods is bigger.



(a) Output voltage after a step-up change in input voltage

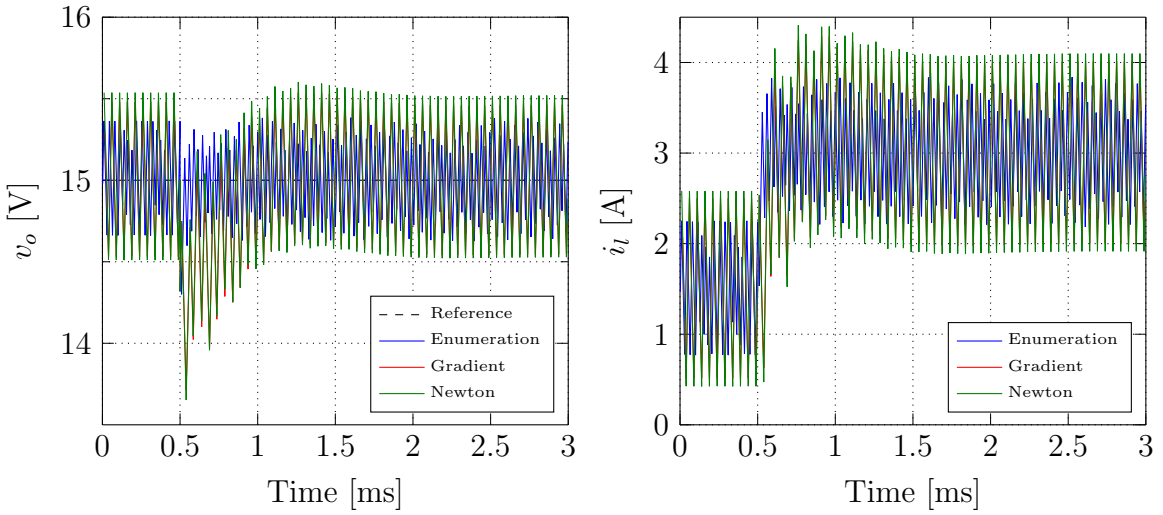
(b) Induction current after a step-up change in input voltage

Figure 2.12: Step-up change in input voltage

2.3.4 Step-down change in the load resistance

As a last test, a step-down change in the load resistance is examined from $R = 10\Omega$ to $R = 5\Omega$ (at $t = 0.5$ms in fig. 2.13). At the moment the change occurs there is a drop

in the output voltage for all methods. With enumeration technique the system settles faster (almost immediately) than with gradient and Newton's methods ($t \approx 1$ms). Moreover there is a difference in the output voltage ripple between the enumeration and the descent methods that is due to the switching frequency difference. In this operation mode the switch operates at about 35kHz for enumeration compared to 20kHz of the descent methods.



(a) Output voltage after a step-down change in the load

(b) Induction current after a step-down change in the load

Figure 2.13: Step-down change in load

2.3.5 Runtime comparison of enumeration, gradient and Newton's methods

In order to compare the three methods (enumeration, gradient and Newton's) in terms of efficiency, the mean time each of these algorithms need to find the optimal solution is calculated. For this purpose, a test with the following conditions took place. Starting up with nominal conditions, each simulation run for 0.5s with a step-up change in the input voltage at 0.25s, measuring the execution time of each iteration of the algorithm, i.e. the needed time to find the optimal solution. All simulations run in a Intel® Pentium® Dual-Core E5500 CPU.

Table 2.1 summarises the mean running time of each algorithm for various prediction horizons N. As can be seen, the running time of enumeration technique increases exponentially with prediction horizon, whereas the running time for gradient and Newton's methods remains relatively constant with that of Newton's method being the fastest of the three. Therefore, the enumeration technique is the least computationally efficient of the three, while Newton's method is the most efficient.

N	Enumeration	Gradient	Newton's
1	46.9	373.5	87.6
2	55.0	138.7	104.7
3	75.4	171.1	215.1
4	125.2	161.5	28.9
5	232.8	192.0	27.9
6	487.2	217.5	30.0
7	1100	217.2	33.9
8	2300	269.1	31.8
9	5000	283.6	43.3
10	10000	272.5	39.8
15	$0.5s$	268.2	52.4

Table 2.1: Running time in μs of enumeration, gradient and Newton's methods for the dc-dc buck converter

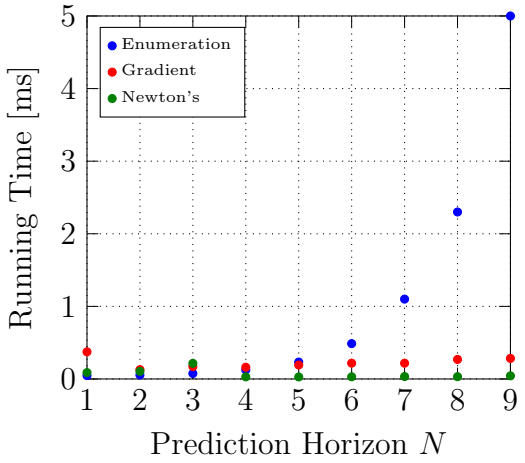


Figure 2.14: Scatter-plot of running time of enumeration, gradient and Newton's methods for the dc-dc buck converter

2.3.6 Conclusions

In this chapter, a buck converter voltage-mode controller formulated in the framework of MPC has been proposed. The discrete-time model of the converter is designed such that it accurately predicts the plant behavior in CCM. Two different implementations of the control problem have been presented and simulated: a mixed-integer quadratic optimization problem solved with enumeration technique and a quadratic optimization problem solved using the gradient and Newton's descent methods. In addition, a Kalman filter has been used as a load estimation scheme for providing offset-free tracking of the output voltage reference. The voltage MPC algorithm for these methods (enumeration, gradient and Newton's) has been applied in MATLAB/SIMULINK. In addition, a step-up change in the output and input voltage as well as a step-down in the load have been investigated.

The simulation results show the fast dynamics achieved by the controller and demonstrate potential advantages of the discussed algorithms. In addition, the Kalman filter effectively rejects the variations in the load and quickly adjusts the voltage reference following any such change.

With regards to the three solving methods applied, the switching frequency with the enumeration strategy is variable and changes for different operation points, while with the descent methods it is not. This is a clear drawback of the enumeration method on the absence of a modulator and the direct manipulation of the converter switches.

During start-up, the converter immediately increases the current to charge the capacitor to the reference voltage level and then the controller quickly restores the current to its nominal value for all three methods. The output voltage reaches its

desired value faster for the enumeration technique than for gradient and Newton's methods due to the Kalman filter. After the step-up of the output voltage, the system exhibits again similar behavior for all the methods. Initially the current increases to bring the output voltage to the new reference level and then decreases to its normal value. Similarly to the start-up results, the converter reaches the steady state faster with the enumeration technique than with the descent methods, a behavior that is again attributed to the Kalman filter. With the step-up change in the input voltage, the output voltage remains at the same level showing the ability of the controller to successfully reject the input voltage variations. In the case of the descent methods, a higher ripple is identified. With the step down of the load resistance, a drop in the output voltage is observed for all three methods. With the enumeration technique, the system settles faster to its final operating point.

Finally, concerning the running time, this increases exponentially for enumeration technique with the prediction horizon, whereas the running time for gradient and Newton's methods remains relatively constant with that of Newton's method being the fastest of the three. Therefore, the enumeration technique is the least computationally efficient of the three, while Newton's method is the most efficient.

2.3.7 Future work

As far as the buck converter is concerned in this thesis only the continuous conduction mode (CCM) was taken into account. This decision was made to simplify the optimization problem, but in the general case a buck converter can operate in discontinuous conduction mode (DCM). This is something that could be explored in future work following the steps shown for the CCM.

The methodology used in this thesis for the dc-dc buck converter could be also used in other types of dc-dc converters, such as the boost converter. Contrary to the buck converter, the state space representation of the boost converter is nonlinear which by extension means that the objective function will not be convex. Nonetheless, the move-blocking technique can be potentially helpful to overcome this problem. This method has been already used in [18, 19, 17] to control dc-dc boost converter using the enumeration technique. The application of gradient descent and Newton's method in a similar system can be the subject of future work.

Appendix A

Partial derivatives of the objective function

As it was described in Section 1.5.1, there are two equivalent ways to express the state evolution of a linear system over a N-step prediction horizon. In Section 2.2.1 the expression in respect to $\Delta \boldsymbol{U}$ (eq. 1.23) was used for the formulation of the objective function J, whereas here the expression in respect to $\boldsymbol{U} = \begin{bmatrix} u(k) & u(k+1) & \cdots & u(k+N-1) \end{bmatrix}^T$ (eq. 1.20) will be used to calculate the first and second order partial derivatives of the objective function J used in Sections 2.2.3.1 and 2.2.3.2.

According to the analysis in Section 1.5.1, the state evolution over a N-step prediction horizon for the dc-dc buck converter is

$$\begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{x}(k+2) \\ \vdots \\ \boldsymbol{x}(k+N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{A_d} \\ \boldsymbol{A_d}^2 \\ \vdots \\ \boldsymbol{A_d}^N \end{bmatrix} \boldsymbol{x}(k) + \begin{bmatrix} \boldsymbol{B_d} & 0 & \cdots & 0 \\ \boldsymbol{A_d B_d} & \boldsymbol{B_d} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A_d}^{N-1}\boldsymbol{B_d} & \boldsymbol{A_d}^{N-2}\boldsymbol{B_d} & \cdots & \boldsymbol{B_d} \end{bmatrix} \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+N-1) \end{bmatrix}. \tag{A.1}$$

The corresponding output in matrix and vector form respectively is

$$\boldsymbol{Y} = \begin{bmatrix} y(k+1) \\ y(k+2) \\ \vdots \\ y(k+N) \end{bmatrix} = \begin{bmatrix} \boldsymbol{C_d} & 0 & \cdots & 0 \\ 0 & \boldsymbol{C_d} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \boldsymbol{C_d} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}(k+1) \\ \boldsymbol{x}(k+2) \\ \vdots \\ \boldsymbol{x}(k+N) \end{bmatrix} \tag{A.2}$$

$$\boldsymbol{Y} = \boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{E}\,\boldsymbol{U}(k) \tag{A.3}$$

where

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{C_d A_d} \\ \boldsymbol{C_d A_d}^2 \\ \vdots \\ \boldsymbol{C_d A_d}^N \end{bmatrix}, \boldsymbol{E} = \begin{bmatrix} \boldsymbol{C_d B_d} & 0 & \cdots & 0 \\ \boldsymbol{C_d A_d B_d} & \boldsymbol{C_d B_d} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{C_d A_d}^{N-1} \boldsymbol{B_d} & \boldsymbol{C_d A_d}^{N-2} \boldsymbol{B_d} & \cdots & \boldsymbol{C_d B_d} \end{bmatrix}, \tag{A.4}$$

$$\boldsymbol{U}(k) = \begin{bmatrix} u(k) & u(k+1) & \cdots & u(k+N-1) \end{bmatrix}^T. \tag{A.5}$$

and $\boldsymbol{A_d}$, $\boldsymbol{B_d}$, $\boldsymbol{C_d}$ are the matrices in Section 2.1.2.

Comparing (2.7) to (A.3) it is easy to see that

$$\boldsymbol{E}\,\boldsymbol{U}(k) = \boldsymbol{Q}u(k-1) + \boldsymbol{S}\Delta\boldsymbol{U}. \tag{A.6}$$

where the vector $\Delta\boldsymbol{U}$ can be written as

$$\Delta\boldsymbol{U} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & -1 & 1 & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix} \boldsymbol{U}(k) - \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} u(k-1) = \boldsymbol{F}\,\boldsymbol{U}(k) - \boldsymbol{G}u(k-1) \tag{A.7}$$

Using (A.3) and (A.7), the objective function can be formulated as:

$$\begin{aligned} J(k, \boldsymbol{U}(k)) &= ||\,\boldsymbol{Y} - \boldsymbol{V}_{\text{ref}}||_2^2 - \lambda||\Delta\boldsymbol{U}||_2^2 \\ &= ||\boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{E}\,\boldsymbol{U}(k) - \boldsymbol{V}_{\text{ref}}||_2^2 - \lambda||\boldsymbol{F}\,\boldsymbol{U}(k) - \boldsymbol{G}u(k-1)||_2^2 \\ &= \boldsymbol{U}^T(k)\boldsymbol{E}^T\boldsymbol{E}\,\boldsymbol{U}(k) - 2(\,\boldsymbol{V}_{\text{ref}} - \boldsymbol{P}\boldsymbol{x}(k))^T\boldsymbol{E}\,\boldsymbol{U}(k) + (\,\boldsymbol{V}_{\text{ref}} - \boldsymbol{P}\boldsymbol{x}(k))^T(\,\boldsymbol{V}_{\text{ref}} - \boldsymbol{P}\boldsymbol{x}(k)) \\ &+ \lambda(\,\boldsymbol{U}^T(k)\boldsymbol{F}^T\boldsymbol{F}\,\boldsymbol{U}(k) - 2\boldsymbol{G}^T\boldsymbol{F}\,\boldsymbol{U}(k)u(k-1) + \boldsymbol{G}^T\boldsymbol{G}u^2(k-1)) \end{aligned} \tag{A.8}$$

The first order partial derivatives of $J(k, \boldsymbol{U}(k))$, combined with (A.6) and (A.7), are[1]:

$$\begin{aligned} \nabla J(k, \boldsymbol{U}(k)) &= 2\boldsymbol{E}^T\boldsymbol{E}\,\boldsymbol{U}(k) - 2\boldsymbol{E}^T(\,\boldsymbol{V}_{\text{ref}} - \boldsymbol{P}\boldsymbol{x}(k)) + \lambda(2\boldsymbol{F}^T\boldsymbol{F}\,\boldsymbol{U}(k) - 2\boldsymbol{F}^T\boldsymbol{G}u(k-1)) \\ &= 2\boldsymbol{E}^T(\boldsymbol{E}\,\boldsymbol{U}(k) - \boldsymbol{V}_{\text{ref}} + \boldsymbol{P}\boldsymbol{x}(k)) + 2\lambda\boldsymbol{F}^T(\boldsymbol{F}\,\boldsymbol{U}(k) - \boldsymbol{G}u(k-1)) \\ &= 2\boldsymbol{E}^T(\boldsymbol{P}\boldsymbol{x}(k) + \boldsymbol{Q}u(k-1) + \boldsymbol{S}\Delta\boldsymbol{U} - \boldsymbol{V}_{\text{ref}}) + 2\lambda\boldsymbol{F}^T\Delta\boldsymbol{U} \end{aligned} \tag{A.9}$$

[1] In [12] it is stated that the derivatives of the function $y = \boldsymbol{A}x$, where y, x are $m-$ and $n-$ column vectors respectively and $\boldsymbol{A}$ is a $m \times n$ matrix are $\frac{\partial y}{\partial x} = \boldsymbol{A}^T$, whereas those of the function $y = x^T\boldsymbol{B}x$, with $\boldsymbol{B}$ a $m \times m$ symmetric matrix, are $\frac{\partial y}{\partial x} = 2\boldsymbol{B}^T x$

The second order partial derivatives of $J(k, \boldsymbol{U}(k))$ are:

$$\nabla^2 J(k, \boldsymbol{U}(k)) = 2\boldsymbol{E}^T\boldsymbol{E} + 2\lambda\boldsymbol{F}^T\boldsymbol{F} \tag{A.10}$$

Bibliography

[1] F. Akar, Y. Tavlasoglu, E. Ugur, B. Vural, and I. Aksoy. A bidirectional non-isolated multi-input dc–dc converter for hybrid energy storage systems in electric vehicles. *IEEE Transactions on Vehicular Technology*, 65(10):7944–7955, Oct 2016.

[2] M. Arnold. Hybrid modeling and optimal control of a step-up dc-dc converter. Semester project, Eidgenössische Technische Hochschule Zürich, 2005.

[3] A. Auslender and M. Teboulle. *Asymptotic cones and functions in optimization and variational inequalities*. Springer Science & Business Media, 2003.

[4] E. Babaei, O. Abbasi, and S. Sakhavati. An overview of different topologies of multi-port dc/dc converters for dc renewable energy source applications. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2016 13th International Conference on*, pages 1–6. IEEE, 2016.

[5] A. Barvinok. *A course in convexity*, volume 54. American Mathematical Society, Providence, RI, 2002.

[6] D. Bertsekas. *Convex analysis and optimization*. Athena Scientific, Belmont, Massachusetts, 2003.

[7] D. Bertsimas and J. N. Tsitsiklis. *Introduction to linear optimization*. Athena Scientific, Belmont, Massachusetts, 1997.

[8] S. Bolte, N. Fröhleke, and J. Böcker. Dc-dc converter design for power distribution systems in electric vehicles using calorimetric loss measurements. In *2016 18th European Conference on Power Electronics and Applications (EPE'16 ECCE Europe)*, pages 1–7, Sept 2016.

[9] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[10] E. F. Camacho and C. Bordons. *Model predictive control*. Springer Science & Business Media, 2nd edition, 2007.

[11] M. Das and V. Agarwal. Design and analysis of a high-efficiency dc–dc converter with soft switching capability for renewable energy applications requiring high voltage gain. *IEEE Transactions on Industrial Electronics*, 63(5):2936–2944, May 2016.

[12] P. J. Dhrymes. *Mathematics for Econometrics*. Springer, New York, 2013.

[13] R. Erickson. *Fundamentals of power electronics*. Kluwer Academic, Norwell, Massachusetts, 2001.

[14] T. Geyer, G. Papafotiou, R. Frasca, and M. Morari. Constrained optimal control of the step-down dc-dc converter. *IEEE TRANSACTIONS ON POWER ELECTRONICS*, 23(5):2454–2464, SEPTEMBER 2008.

[15] B. Grünbaum. *Convex polytopes*. Springer-Verlag, New York, second edition, 2003.

[16] J. B. Hiriart-Urruty and C. Lemaréchal. *Fundamentals of convex analysis*. Springer-Verlag, Berlin, Heidelberg, Germany, 2001.

[17] P. Karamanakos. *Model Predictive Control Strategies for Power Electronics Converters and AC Drives*. PhD thesis, National Technical University of Athens, 2013.

[18] P. Karamanakos, T. Geyer, and S. Manias. Direct model predictive current control strategy of dc-dc boost converters. *IEEE JOURNAL OF EMERGING AND SELECTED TOPICS IN POWER ELECTRONICS*, 1(4):337–346, December 2013.

[19] P. Karamanakos, T. Geyer, and S. Manias. Direct voltage control of dc-dc boost converters using enumeration-based model predictive control. *IEEE TRANSACTIONS ON POWER ELECTRONICS*, 29(2):968–978, February 2014.

[20] W. H. Kwon and S. Han. *Receding horizon control: model predictive control for state models*. Advanced Textbooks in Control and Signal Processing. Springer-Verlag London, 2005.

[21] M. Ławryńczuk. *Computationally efficient model predictive control algorithms*, volume 3. Springer, 2014.

[22] G. M. Lee, N. N. Tam, and N. D. Yen. *Quadratic programming and affine variational inequalities: a qualitative study*, volume 78 of *Nonconvex Optimization and Its Applications*. Springer US, 2005.

[23] R. Lucchetti. *Convexity and Well-Posed Problems*. Springer Science & Business Media, 2006.

[24] D. Luenberger. *Linear and nonlinear programming*. Addison-Wesley, Reading, Massachusetts, 1984.

[25] S. N. Manias. *Power Electronics and Motor Drive Systems*. Academic Press, 1st edition, 2017.

[26] R. R. Melo, F. L. M. Antunes, and S. Daher. Bidirectional interleaved dc-dc converter for supercapacitor-based energy storage systems applied to microgrids and electric vehicles. In *2014 16th European Conference on Power Electronics and Applications*, pages 1–10, Aug 2014.

[27] G. Pannocchia and J. B. Rawlings. Disturbance models for offset-free model-predictive control. *AIChE Journal*, 49(2):426–437, 2 2003.

[28] E. Polak. *Optimization*, volume 3. Springer-Verlag, New York, 1997.

[29] P. Poovarasan, M. Saraswathi, and R. Nandhini. Analysis of high voltage gain dc-dc boost converter for renewable energy applications. In *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, pages 0320–0324, April 2015.

[30] P. Poure, S. Weber, B. Vidales, M. Madrigal, and D. Torres. High step-up dc-dc converter for renewable energy harvesting applications. In *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, pages 1–6, June 2016.

[31] M. Rashid. *Power electronics handbook*. Academic Press, San Diego, 2001.

[32] J. A. Rossiter. *Model-based predictive control: a practical approach*. CRC Press, Boca Raton, 2003.

[33] G. Sarriegui, S. Beushausen, and R. W. D. Doncker. Comparison of high frequency inductors for bidirectional dc-dc converters for electric vehicles. In *2016 18th European Conference on Power Electronics and Applications (EPE'16 ECCE Europe)*, pages 1–8, Sept 2016.

[34] F. Shang, G. Niu, and M. Krishnamurthy. Design and analysis of a high-voltage-gain step-up resonant dc–dc converter for transportation applications. *IEEE Transactions on Transportation Electrification*, 3(1):157–167, March 2017.

[35] W. Sun and Y.-X. Yuan. *Optimization theory and methods: nonlinear programming*. Springer, New York, 2006.

[36] A. Tomar and S. Mishra. Multi-input single-output dc-dc converter based pv water pumping system. In *2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES)*, pages 1–5, July 2016.

[37] F. Velandia, W. Martinez, C. A. Cortes, M. Noah, and M. Yamamoto. Power loss analysis of multi-phase and modular interleaved boost dc-dc converters with coupled inductor for electric vehicles. In *2016 18th European Conference on Power Electronics and Applications (EPE'16 ECCE Europe)*, pages 1–10, Sept 2016.