SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reference number: FCHPT-19990-26488

Algorithms for Process Modelling and Fast Model Predictive Control

DISSERTATION THESIS

Bratislava, 2014 **Ing. Alexander Szűcs**

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA
FACULTY OF CHEMICAL AND FOOD TECHNOLOGY

Reference number: FCHPT-19990-26488

Algorithms for Process Modelling and Fast Model Predictive Control

DISSERTATION THESIS

Study program: Process Control
Study field number: 2621
Study field: 5.2.14 Automation
Workplace: Department of Information Engineering and Process Control
Supervisor: Prof. Ing. Miroslav Fikar, DrSc.
Co-supervisor: Doc. Ing. Michal Kvasnica, PhD.

Bratislava, 2014 **Ing. Alexander Szűcs**

Slovak University of Technology in Bratislava
Institute of Information Engineering, Automation and Mathematics

Faculty of Chemical and Food Technology
Academic year: 2014/2015
Reg. No.: FCHPT-19990-26488

STU FCHPT

DISSERTATION THESIS TOPIC

Student:	**Ing. Alexander Szűcs**
Student's ID:	26488
Study programme:	Process Control
Study field:	5.2.14 automation
Thesis supervisor:	prof. Ing. Miroslav Fikar, DrSc.

Topic: **Algorithms for Process Modelling and Fast Model Predictive Control**

Assignment procedure from:	02. 09. 2010
Date of thesis submission:	02. 09. 2014

L. S.

Ing. Alexander Szűcs
Solver

prof. Ing. Miroslav Fikar, DrSc.
Head of department

prof. Ing. Miroslav Fikar, DrSc.
Study programme supervisor

Acknowledgements

I consider myself a lucky guy due to having a chance to absolve my PhD study in two different countries, supervised by two well known scientists in the field of automatic control, namely by Michal Kvasnica and Colin Jones. Even though the previous four years were rather difficult, I am heavily indebted to them. I have not just learned from them a lot, but they have introduced me two fundamentally different ways of approaching and solving problems, made my skills more versatile. Without a doubt, regardless what I will end up doing, the skills and knowledge acquired over the last four years will remain useful throughout my entire future career.

I would like to express my gratitude to prof. Miroslav Fikar too, who gave me a lot of freedom during my studies, intervening to it only when it was necessary. Nonetheless, his guidance was absolutely helpful and inevitable to succeed.

I have started this section by stating that I think I am a lucky person. Frankly, I have never thought that one day I might be somewhere where I am at the moment. For this and for many other things I want to further express my thanks to my friends, namely to Tomaš, Juro, Lukaš and to the group of "Gabos", who despite of my complex, many times annoying personality were keep supporting me and have contributed to my life in a form of unforgottable memories by means of different parties and discussions.

Special thanks go to my ex-girlfriend, Erika, who helped me a lot during my bachelor studies and pushed me towards studying. I think it was a good push. Thank you!

My biggest thanks go to my family for their mental support provided during my doctoral studies. Without their help, I would have already left the academia.

Alexander Szűcs
Bratislava, 2014

Abstract

This work aims to contribute to modelling and fast predictive control of processes. It can be divided into several topics.

Process modelling is investigated and an effective approximation technique is described. It can be used to approximate an original non-linear process model as a hybrid system with piecewise affine dynamics. We discuss three different cases, how one can obtain the approximation of an arbitrary nonlinear function. The most trivial case assumes that the analytic form of the nonlinear term is already known. On the other hand, if only some set of input-output measurements are given, we employ a two-stage procedure to obtain the final approximation. This method aims to select the appropriate subset of basis functions and consecutively finding a proper linear combination of them. Once we possess the analytic formula of our approximated function, we can obtain the final PWA approximation by solving standard nonlinear programs. We show, that under mild assumptions, the task can be transformed into a series of one-dimensional problems. Finally, we demonstrate the efficiency of our technique on an illustrative example, involving a highly nonlinear reactor.

The second part of the work deals with fast model predictive control. We investigate the problem of reduction of the amount of memory needed to describe explicit MPC solutions. The main idea of explicit MPC stems from pre-computation of the optimal control action for all possible initial conditions and subsequently storing them in a form of a look-up table. On one hand, this concept allows faster implementation, but on the other, requirements for memory storage increase too. In order to eliminate this drawback, we continue with a description of an effective, three-layer compression technique, allowing fast implementation on low-cost hardware platforms. This three-layer procedure first identifies similarities between polytopic regions in form of an affine transformation. If such a mapping exists, certain regions can be represented using less data. The second layer then applies data de-duplication to identify and remove repeating sequences of data. Regions are then described by integer pointers to such a unique set. Finally Huffman encoding is applied to compress such integer pointers using prefix-free variable-length bit encoding. The chapter ends with efficiency evaluation of the proposed technique on several, randomly generated feedback law examples.

The final chapter is devoted to the so-called operator splitting methods, by means of one can solve convex optimisation problems very efficiently by simply decomposing the original possibly complex problem into a series of simple operations well known from linear algebra. Several algorithms and their range of applicability are presented.

Abstrakt

Práca sa venuje modelovaniu a rýchlemu prediktívnemu riadeniu procesov. Skladá sa z viacerých tém.

V prvej časti sa zameriava na modelovanie procesov a aproximáciu pôvodne nelineárneho modelu za po častiach lineárny model, ktorý je vhodnejší pre použitie v rýchlom prediktívnom riadení. Navrhujeme efektívne aproximačné metódy, ktoré sú aplikovateľné vo viacerých prípadoch. Najjednoduchší prípad predpokladá existenciu analytického tvaru aproximovaného nelineárneho výrazu. Na druhej strane, pri existencii iba vstupno-výstupných dát, získanie finálnej aproximácie vyžaduje aplikovanie dvojkrokovej procedúry. Táto metóda sa vyznačuje hľadaním príslušnej podmnožiny bázických funkcií a následným nájdením koeficientov vhodnej lineárnej kombinácie. Keď máme k dispozícii analytický výraz aproximovanej funkcie, riešením štandardnej úlohy nelineárneho programovania ľahko získame výslednú transfomáciu, ktorá je po častiach afinná. Navyše, v práci ukazujeme, že pri splnení istých podmienok úloha môže byť pretransformovaná na sekvenciu jednorozmerných aproximácií. Efektívnosť metódy je demonštrovaná na vysoko nelineárnom modeli chemického reaktora.

Druhá časť práce sa zaoberá rýchlym prediktívnym riadením. Uvažujeme problém zníženia pamäťových nárokov explicitných prediktívnych regulátorov. Hlavná idea explicitného MPC spočíva v predpočítaní optimálneho akčného zásahu pre všetky možné počiatočné podmienky a ich následným uložením vo forme vyhľadávacej tabuľky. Na jednej strane, táto metóda umožňuje rýchlu implementáciu, avšak za cenu vyšších pamäťových nárokov. Za účelom eliminovania tohto nedostatku uvedieme opis efektívnej trojvrstvovej komprimačnej techniky, takto umožňujúc rýchlu implementáciu na výpočtových platformách s obmedzenou pamäťovou kapacitou. Trojvstrvová procedúra najprv identifikuje podobnosti medzi polytopickými regiónmi vo forme afinnej transfomácie. V prípade existencie takéhoto zobrazenia môžu byť určité regióny reprezentované úspornejším spôsobom. Druhá vrstva eliminuje opakované sekvencie dát pomocou de-duplikácie. Po tejto procedúre regióny sú charakterizované pomocou smerníkov. Získaná smerníková reprezentácia je v konečnej fáze nahradená bitovými sekvenciami, získaných pomocou Huffmanovho kódovania. Efektívnosť komprimačnej techniky je vyhodnotená na rôznych náhodne generovaných spätnoväzbových zákonov.

Posledná kapitola je venovaná k rôznym algoritmom slúžiacich na riešenie konvexných optimalizačných problémov. Hlavná idea týchto algoritmov spočíva v ich schopnosti pretransformovať originálny konvexný problém na sekvenciu jednoduchých operácií známych z oblasti lineárnej algebry. Porovnáme viaceré algoritmy a ich rozsahy použiteľnosti.

Contents

List of Abbreviations

ADMM	Alternating Direction Method of Multiplier
AMA	Alternating Minimisation Algorithm
CP I	Chambolle-Pock I
CP II	Chambolle-Pock II
ELCS	Extended Linear Complementary Systems
eMPC	explicit Model Predictive Control
FADMM	Fast Alternating Direction Method of Multiplier
FAMA	Fast Alternating Minimisation Algorithm
HYSDEL	Hybrid System Description Language
LCS	Linear Complementary Systems
MLD	Mixed Logical Dynamic
MPC	Model Predictive Control
PLC	Programmable Logic Controller
PWA	Piecewise Affine
PADMM	Proximal Alternating Direction Method of Multipliers
RHMPC	Receding Horizon Model Predictive Control

Chapter 1

Introduction

Mathematical models of physical plants play a crucial role in many areas connected to the field of control theory. The most challenging problem is usually represented by finding a compromise solution between the model's accuracy and its complexity. Naturally, the best possible simulation results can be achieved by nonlinear models, although control design techniques based on such models are difficult. The most common way of simplification, in order to avoid the usage of complex nonlinear models is represented by Taylor expansion, which allows to create a linearized model around one operating point. However, this concept solves our problem only partially, since it is not able to capture dynamical properties of the original nonlinear model in the entire domain of interest. This drawback becomes very significant with the increasing distance from the original operating point, resulting in discrepancies between the trajectories obtained from the original, nonlinear model and the approximated one. The most rational solution to the above addressed problem is to use more approximation points, creating several local models. This idea can be captured by the concept of hybrid systems (Morari et al. 2003), where the above mentioned multiple linearization technique is realizable through the interconnection of continuous variables with discrete ones. In other words, hybrid systems are capable to detect local models by means of boolean variables and according on their truth value switch over to the corresponding local model and pick up the respective expression associated with the particular region. There are several available frameworks to describe such systems like MLD, PWA models and max-min scaling functions. It was shown that under mild assumptions these models are equivalent to each other (Heemels et al. 2001) and transformation between these models is available too. In this work we propose an optimization-based procedure of deriving PWA functions, wrapped in a MATLAB toolbox called `AUTOPROX`. The toolbox is capable to approximate an arbitrary function as well as right-hand sides of differential

equations, exploiting the concept of separability. This technique allows us to extend the procedure to higher dimension by a simple transformation. Even functions, at first glance non-separable, can be "chunked" into several number of separable terms. Generally, the approach requires the analytic form of the function, but it can be effective even in case of input-output measurements. We propose an efficient method, how one can overcome the absence of analytic terms, where the coefficients multiplying the basis functions are calculated by optimization.

Model predictive control has become very popular control strategy in the recent years due to its ability to handle constraints. The main idea of this approach is based on solving optimization problems in each sampling time and implement only the first control action from the whole sequence, also known as receding horizon model predictive control (RHMPC) (Bemporad et al. 2000b). By solving an optimization problem in each step, we can push our states towards zero or track the reference, while guaranteeing optimality and satisfaction of constraints. This so-called online method is very effective in slow processes (e.g. chemical or petrochemical processes), but its implementation becomes cumbersome for fast mechatronics processes.

Fortunately, at the beginning of the 21th century a new concept appeared, called *explicit MPC* (Bemporad et al. 2000a). This concept enables to implement such controllers on fast systems, since the whole optimization procedure is performed off-line. Result of this computation is a look-up table, containing the control laws, each associated to its own region. The advantages are twofold. First, the control laws are affine functions of states, thus obtainment of the corresponding control actions simply reduces to finding the appropriate region, known as point location problem and pick up the corresponding affine function belonging to the region. The second reason is also quite obvious, since the regions over which the affine the control law is defined can be easily acquired by multi-parametric programming. Explicit MPC (eMPC) is still very attractive, but unfortunately it also posesses some drawbacks. First of all, it is confined to problems of small sizes. Furthermore,it is not able to adapt itself to control systems with varying parameters, because the parameters of the controller are computed in advance. Very challenging task is implementation of eMPC solutions on industrial control systems (e.g. PLCs), where the total amount of allowable memory is usually only 2 kB. For systems with many states and inputs the real-time implementation of eMPC solutions becomes difficult, since they usually contain more than 1000 regions. Moreover, if we take into account the number of affine functions, associated to the regions, we can easily run out of memory.

Therefore, second part of this work deals with an efficient compression technique to reduce the memory footprint required by an eMPC controller in order to implement it on

low-cost platforms. We present here a three-stage procedure, by means of a substantial reduction can be attained. The procedure contains three layers. The first one determines a subset of regions, denoted as basis regions, by means of one can reconstruct the remaining ones. We show how to formulate the search for such a mapping by solving a mixed-integer problem, which is done off-line. If the transformation exists, the corresponding regions can then be represented using less data. The second layer can either be applied on top of the first one, or independently. Here, memory is saved by identifying positive and negative duplicities in the half-space representation of several polytopic regions. The duplicate occurrences are then represented as mere integer pointers to the unique set of data. Compared to the first layer, the additional computation to be performed on-line is significantly smaller. Finally, in the last layer we propose to compress the integer pointers by Huffman encoding (Knuth 1985). Here, variable-length bit codewords are assigned to each integer, depending on its frequency of abundance. Main benefit of the proposed strategies is that they can be applied on top of all aforementioned complexity reduction schemes. Saving in terms of memory is achieved at the price of an increase of the implementation effort performed on-line. Therefore the approach is mainly suited for situations where the implementation device posesses enough computational power, but has severe memory limitations.

The significant progress that has been made in recent years both in hardware implementations and in numerical computing has rendered real-time optimization-based control a viable option when it comes to advanced industrial applications. More recently, the need for control of a process in the presence of a limited amount of hardware resources has triggered research in the direction of embedded optimization-based control. Many efficient high-speed solvers have been developed for both linear and nonlinear control, based on either *first order methods* (FiOrdOs (Ullmann 2011)), *interior point (IP) methods* (FORCES (Domahidi et al. 2012), CVXGEN (Mattingley and Boyd 2012)) or active sets (QPOASES (Ferreau et al. 2008b)).

In this work we focus on systems with linear dynamics, giving rise to convex control problems. We aim to explore a family of first order methods known as *decomposition schemes* or *operator splitting methods*. In the simplest case, the abstract form of the problem at hand is the minimization of the sum of two convex functions.

Formulations similar to the above have been studied extensively and we can look for their roots in the method of multipliers (Hestenes 1969), the Arrow-Hurwicz method (Arrow et al. 1958), Douglas-Rachford splitting (Douglas and Rachford 1956), and ADMM (Gabay and Mercier 1976b, Glowinski and Marrocco 1975b). More recent references that illustrate the applicability of such methods in modern engineering problems (signal and image processing, big data analysis, machine learning) are by Boyd et al. (2011b) and

Combettes and Pesquet (2011). The thesis Esser (2010) provides a nice and comprehensive description of the connection of several splitting algorithms under a common framework. Finally, the book (Bauschke and Combettes 2011) provides a mathematically rigorous introduction to operator splitting methods in general Hilbert spaces.

Therefore, in the third part of this work we present three popular splitting algorithms, the *Alternating direction method of multipliers (ADMM)*, the *Alternating minimization algorithm (AMA)* and the *primal-dual scheme from Chambolle and Pock I (CPI)* and their accelerated versions, *Fast alternating direction method of multipliers (FADMM)*, *Fast alternating minimisation algorithm (FAMA)*, and *primal-dual scheme from Chambolle and Pock II (CPII)*. Our choice is motivated by the fact that the methods have been analysed and extended by several communities, and their properties are well-understood.

Aims of the Thesis

The main goals of this thesis can be summarised as follows:

- The first field of contribution is modelling of hybrid systems, where we developed an effective approximation technique to obtain the corresponding PWA approximation of a nonlinear function either characterised by an analytic expression or by means of a set of input-output measurements. Furthermore in order to simplify the underlying procedure a software package called `AUTOPROX` was developed, by means of one can easily obtain the final approximation of the nonlinear function. The proposed approximation procedures were published in the following papers:

  – M. Kvasnica, A. Szűcs, and M. Fikar. Automatic derivation of optimal piecewise affine approximations of nonlinear systems. In *Preprints of the 18th IFAC World Congress Milano (Italy) August 28 - September 2, 2011*, pages 8675–8680, 2011c. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1166.

  – A. Szűcs, M. Kvasnica, and M. Fikar. Optimal piecewise affine approximations of nonlinear functions obtained from measurements. In *4th IFAC Conference on Analysis and Design of Hybrid Systems, Eindhoven, Netherlands*, pages 160–165, 2012. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1306.

  – A. Szűcs, M. Kvasnica, and M. Fikar. Matlab toolbox for automatic approximation of nonlinear functions. In M. Fikar and M. Kvasnica, editors, *Proceedings of the 18th International Conference on Process Control*, pages 119–124, Tatranská Lomnica, Slovakia, June 14-17, 2011 2011a. Slovak University of Technology in Bratislava. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id

- The second field of contribution is complexity reduction of explicit MPC solutions. We developed an effective three-layer compression technique, which can be applied to an arbitrary MPC solution. The underlying technique is able to significantly reduce memory requirements of explicit predictive controllers, thus enabling their implementation on industrial hardware platforms, e.g. PLCs. The proposed three-layer compression technique can be found in the following publication:

  – A. Szűcs, M. Kvasnica, and M. Fikar. A memory-efficient representation of explicit mpc solutions. In *Proceedings of the 50th CDC and ECC*, pages 1916–1921, Orlando, Florida, 2011b. URL http://www.kirp.chtf.stuba.sk/publication_i

- Finally, the last field of contribution are operator splitting methods. This concept refers to a set of algorithms capable of solving convex optimisation problems by transforming them into a series of simple operations, e.g. matrix-vector multiplication. Furthermore, opposed to explicit MPC, these algorithms could be very efficient for large-scale systems as well.

  – G. Stathopoulos, A. Szűcs, Y. Pu, and C Jones. Splitting methods in control. In *European control conference to appear*, 2014. URL http://www.kirp.chtf.stuba.sk/pu

Part I

MATHEMATICAL BACKGROUND

Chapter 2

Convex Sets and Functions

To understand the concepts of MPC, one has to take a deeper look on the basic building blocks constituting the basic pillars of convex optimisation, such as sets or functions. Therefore, in this chapter we will introduce some essential concepts from topology of sets, e.g. closedness, boundedness and characterise the basic properties of convex sets and functions.

2.1 Sets

Definitions in this section are due to Christophersen (2006), Grünbaum (2000), Weisstein (2010).

Definition 2.1 (ϵ-Ball) *The open n-dimensional ϵ-ball in $\mathbb{R}^n$ around a given point (center) $\boldsymbol{x}_c$ is the set*

$$\mathcal{B}_\epsilon(\boldsymbol{x}_c) := \{\boldsymbol{x} \in \mathbb{R}^n \mid \|\boldsymbol{x} - \boldsymbol{x}_c\| < \epsilon\},$$

where the radius $\epsilon > 0$ and $\|\cdot\|$ denotes any vector norm (usually the Euclidean vector norm $\|\cdot\|_2$).

Definition 2.2 (Neighborhood) *The neighborhood of a subset $\mathcal{S}$ of $\mathcal{X} \subseteq R^n$ is defined as a set $\mathcal{N}(\mathcal{S})$ with $\mathcal{S} \subset \mathcal{N}(\mathcal{S}) \subseteq \mathcal{X}$ such that for each $s \in \mathcal{S}$ there exist an n-dimensional ϵ-ball with $\mathcal{B}_\epsilon(s) \subseteq \mathcal{N}(\mathcal{S})$ and $\epsilon > 0$.*

Definition 2.3 (Convex set) *A set $\mathcal{S} \subseteq R^{n_s}$ is convex if the line segment connecting any pair of points of $\mathcal{S}$ lies entirely in $\mathcal{S}$, i.e. if for any $s_1, s_2 \in \mathcal{S}$ and any α with $0 \leq \alpha \leq 1$, we have*

$$\alpha s_1 + (1 - \alpha) s_2 \in \mathcal{S}.$$

See Figs. 2.1(a), 2.1(b).

Definition 2.4 (Convex hull) *The convex hull of a set $\mathcal{S} \subseteq \mathcal{R}^{n_s}$ is the smallest convex set containing $\mathcal{S}$, i.e.*

$$hull(\mathcal{S}) := \left\{ \sum_{i=1}^{k} \alpha_i s_i \;\middle|\; \forall s_i \in \mathcal{S}\ \exists \alpha_i,\ \alpha_i \geq 0,\ \sum_{i=1}^{k} \alpha_i = 1 \right\}.$$

Definition 2.5 (Closed set) *A set $\mathcal{S} \subseteq R^{n_s}$ is closed if every point not in $\mathcal{S}$ has a neighborhood disjoint from $\mathcal{S}$, i.e.*

$$\forall \boldsymbol{x} \notin \mathcal{S}, \exists \epsilon > 0 \quad \text{such that} \quad \mathcal{B}_\epsilon(\boldsymbol{x}) \cap \mathcal{S} = \emptyset.$$

Definition 2.6 (Bounded set) *A set $\mathcal{S} \subseteq \mathbb{R}^{n_s}$ is bounded if it is contained inside some ball $\mathcal{B}_r(\cdot)$ of finite radius r, i.e.*

$$\exists r < \infty,\ s \in \mathbb{R}^{n_s} \quad \text{such that} \quad \mathcal{S} \subseteq \mathcal{B}_r(s).$$

Definition 2.7 (Compact set) *A set $\mathcal{S}$ is compact if it is closed and bounded.*

Definition 2.8 (Set collection) *$\mathcal{S}$ is called a set collection (in $\mathbb{R}^{n_s}$) if it is a collection of finite number of n_s-dimensional sets $\mathcal{S}_i$, i.e.*

$$\mathcal{S} := \{\mathcal{S}_i\}_{i=1}^{N_\mathcal{S}},$$

where $dim(\mathcal{S}_i) = n_s$ and $\mathcal{S}_i \subseteq \mathbb{R}^{n_s}$ for $i = 1, \ldots, N_\mathcal{S}$ with $N_\mathcal{S} < \infty$. A set collection of sometimes also referred to as family of sets.

Definition 2.9 (Partition) *A collection of sets $\{\mathcal{S}_i\}_{i=1}^{N_\mathcal{S}}$ is a partition of a set $\mathcal{S}$ if $\mathcal{S} = \cup_{i=1}^{N_\mathcal{S}} \mathcal{S}_i$ and $\mathcal{S}_i \cap \mathcal{S}_j$ for all $i \neq j$, where $i, j \in \{1, \ldots, N_\mathcal{S}\}$.*

Definition 2.10 (Half-space)

$$S = \{x : p^T x \leq \alpha\}, \tag{2.1}$$

where p is nonzero vector in $\mathbb{R}^n$, and α is a scalar.

Definition 2.11 ($\mathcal{H}$-polyhedron) *A convex set $\mathcal{Q} \subseteq \mathbb{R}^n$ given as an intersection of finite number of half-spaces*

$$\mathcal{Q} = \{x \in \mathbb{R}^n \mid Q^x x \leq Q^c\} \tag{2.2}$$

Definition 2.12 ($\mathcal{H}$-polytope) *A bounded polyhedron $\mathcal{P} \subset \mathbb{R}^n$*

$$\mathcal{P} = \{x \in \mathbb{R}^n \mid \mathcal{P}^x x \leq \mathcal{P}^c\} \tag{2.3}$$

is called $\mathcal{H}$ - polytope

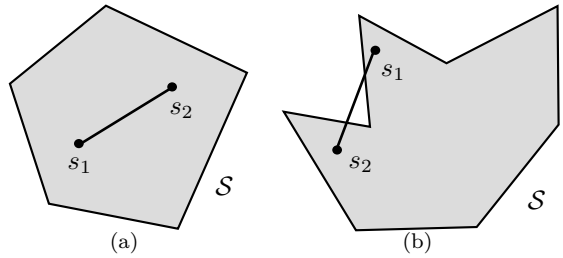


Figure 2.1: Convex (a) and non-convex (b) set.

Definition 2.13 ($\mathcal{V}$-polytope) *A polytope can also be represented by means of a convex combination of its vertices $\mathcal{V}_p$:*

$$\mathcal{P} = \left\{ x \in \mathbb{R}^n \mid x = \sum_{i=1}^{v_p} \alpha_i V_p^{(i)}, 0 \leq \alpha_i \leq 1, \sum_{i=1}^{v_p} \alpha_i = 1 \right\}, \tag{2.4}$$

where $\mathcal{V}_p^{(i)}$ denotes the i-th vertex of $\mathcal{P}$, and v_p is the total number of vertices of $\mathcal{P}$.

2.2 Functions

Definitions in this section are due to Christophersen (2006), Syau (1998).

Definition 2.14 (Vector 1-, ∞- , 2-norm) *The vector 1-, ∞-, and 2-norm of $x \in \mathbb{R}^n$ is defined as*

$$||x||_1 := \sum_{i=1}^{n} |x_i|, \quad ||x||_\infty := \max_{1 \leq i \leq n} |x_i|, \quad ||x||_2 := \sum_{i=1}^{n} x_i^2$$

respectively, where x_i is the i-th element of x.

Definition 2.15 (Convex/concave function) *A real-valued function $f : \mathcal{X} \mapsto \mathbb{R}^{n_f}$ is convex if its domain $\mathcal{X} \subseteq R^n$ is a convex set and*

$$\forall x_1, x_2 \in \mathcal{X}, \quad 0 \leq \alpha \leq 1 \quad \Longrightarrow \quad f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2)$$

where $\leq$ is to be considered component wise. $f(\cdot)$ is strictly convex if the last inequality above is replaced by strict inequality. $f(\cdot)$ is concave if $-1 f(\cdot)$ is convex.

Definition 2.16 (Affine function) *A real-valued function $f : \mathcal{X} \mapsto \mathbb{R}^{n_f}$ with $\mathcal{X} \subseteq \mathbb{R}^n$ is affine if it is of the form $f(x) := Fx + g$, where $F \in \mathbb{R}^{n_f \times n}$ and $g \in \mathbb{R}^{n_f}$.*

Definition 2.17 (Piecewise affine function) *A real-valued function* $f_{PWA} : \mathcal{X} \mapsto \mathbb{R}^{n_f}$ *with* $\mathcal{X} \subseteq \mathbb{R}^n$ *is piecewise affine (PWA), if* $\{\mathcal{X}_i\}_{i=1}^{N_\mathcal{X}}$ *is a partition of* $\mathcal{X}$ *and*

$$f_{PWA}(x) := F_i x + g_i \quad \forall x \in \mathcal{X}_i,$$

where $F_i \in \mathbb{R}^{n_f \times n}$, $g_i \in \mathbb{R}^{n_f}$, *and* $i = 1, \ldots, N_X$.

Definition 2.18 (Lower semi-continuous function) *A real-valued function* $f : \mathbb{R} \mapsto \mathbb{R}$ *on a set* $\mathcal{S}$ *is a lower semi-continuous at a point* $x \in \mathcal{S}$ *if, for each* $\lambda \in \mathcal{R}$, $\lambda \leq f(x)$, *there exists a neighbourhood* U *of* x *such that* $f(y) \geq \lambda$ *for all* $y \in \mathcal{U}$. *Function* $f : \mathcal{S} \mapsto [-\infty, +\infty]$ *is said to be lower semi-continuous if* f *is lower semi-continuous at each point of* $\mathcal{S}$.

2.3 Constrained Optimisation

A general constrained optimisation can be formulated as follows:

$$\begin{aligned} \min_u \quad & f(u) \\ \text{s.t.}: \quad & u \in \mathcal{S} \end{aligned} \tag{2.5}$$

Here the vector $u \in \mathbb{R}^N$ denotes the *optimisation variable*, $f : \mathbb{R}^n \to \mathbb{R}$ is the *objective function*, and $\mathcal{S}$ is the *constraint set*.

Definition 2.19 *Feasible and Optimal Solutions: Let* $f : \mathbb{R}^n \to \mathbb{R}$ *and consider the constrained optimisation problem* (2.5), *where* $\mathcal{S}$ *is a nonempty set in* $\mathbb{R}^n$.

- A point $u \in \mathcal{S}$ is called a feasible solution to problem (2.5)
- If $\bar{u} \in \mathcal{S}$ and $f(u) \geq f(\bar{u})$ for each $u \in \mathcal{S}$, then $\bar{u}$ is called an optimal solution, a global optimal solution, or simply a solution to the problem.
- If $\bar{u} \in \mathcal{S}$ and if there exists an $\epsilon-$ neighbourhood $\mathcal{N}_\epsilon(\bar{u})$ around u such that $f(u) \geq f(\bar{u})$, for each $u \in \mathcal{S} \cap \mathcal{N}_\epsilon(\bar{u})$, then $\bar{u}$ is called a local optimal solution.
- If $\bar{u} \in \mathcal{S}$ and $f(u) \geq f(\bar{u})$ for each $u \in \mathcal{S} \cap \mathcal{N}_\epsilon(\bar{u})$, $u \neq \bar{u}$, for some $\epsilon \geq 0$, then $\bar{u}$ is called strict local optimal solution.

2.3.1 Linear Programming

If f in (2.5) is linear and $\mathcal{S}$ in (2.5) is defined by a set of linear inequalities $g_i(u) \leq 0, i = 1, \ldots, m$, problem (2.5) can be reformulated and solved as a linear programming (LP) problem.

$$\begin{aligned} \min_{u} \quad & c^T u \\ \text{s.t.} \quad & Au \leq b, \\ & A_{eq} u = b_{eq} \end{aligned} \tag{2.6}$$

There are three fundamentally different types of algorithms for solving LPs: simplex, interior-point (Hudzovič 2004) and active set methods (K.Tone 1993). Even though the former method has an exponential worst-case (Klee and Minty 2004), on average it solves the problem in polynomial time (Karmakar 1984). Due to the fact that the latter method has a polynomial worst case, both techniques are heavily utilised for solving linear programming problems.

2.3.2 Quadratic Programming

If f is convex and quadratic and $\mathcal{S}$ is defined by set of linear inequalities $g_i(u) \leq 0$, $i = 1, \ldots, m$, problem (2.5) is referred to as a quadratic programming (QP) problem:

$$\begin{aligned} \min_{u} \quad & 1/2 u^T H u + c^T u \\ \text{s.t.} \quad & Au \leq b, \\ & A_{eq} u = b_{eq} \end{aligned} \tag{2.7}$$

For positive definite matrix H, the interior point methods (Murty 2006) solve the problem in polynomial time. However, in case of H is indefinite, the problem becomes NP-hard (Sahni 1974). In fact, one negative eigenvalue of matrix H is sufficient to turn the underlying optimisation problem into a NP-hard problem. Furthermore, if the structure of the underlying problem can be exploited (Maes 2010), active set methods are applicable as well In chapter 5 we will introduce several algorithms by means of one can solve quadratic programming problems very quickly.

2.3.3 Mixed-Integer Linear Programming

If the vector u of optimisation variables is composed of real and a binary part, i.e. $u = [u_r^T, u_b^T]^T$ with $u_r \in \mathbb{R}^{n_r}$ and $u_b \in \{0,1\}^{n_b}$, and the objective function f is linear and constraint set $\mathcal{S}$ defined by linear inequalities $g_i(u) \leq 0, i = 1, \ldots, m$, problem (2.5) is referred to as a Mixed Integer Linear Program (MILP). Formally it can be stated as:

$$\begin{aligned} \min_{u} \quad & c_r^T u_r + c_b^T u_b \\ \text{s.t.} \quad & A_r u_r + A_b u_b \leq b \\ & A_{eq,r} u_r + A_{eq,b} u_b = b_{eq} \\ & u_b \in \{0,1\}^{n_b} \end{aligned} \tag{2.8}$$

The presence of binary variables makes solving these types of problems much harder to solve compared to above mentioned methods, namely to LP and QP. Obviously, the most straightforward approach to obtain the optimal solution is to enumerate all possible combinations of binary variables and pick up the optimiser corresponding to an optimisation problem with the lowest objective value. possible scenarios. Considering the fact that in general there are n_b binary variables, one would have to solve 2^{n_b} linear programming problems. However, this technique is very time consuming, and more advanced techniques have been developed, specifically Branch & Bound or Branch & Cut (Adjiman et al. 1996a, Soland 1971) which in order to find the optimal solution do not need to compute all the possible combinations.

2.3.4 Mixed-Integer Quadratic Programming

If the vector u of optimisation variables is composed of real and a binary part, i.e. $u = [u_r^T, u_b^T]^T$ with $u_r \in \mathbb{R}^{n_r}$ and $u_b \in \{0,1\}_b^n$, and the objective function f is quadratic and constraint set $\mathcal{S}$ defined by linear inequalities $g_i(u) \leq 0, \quad i = 1, \ldots, m$, problem (2.5) is referred to as a Mixed Integer Quadratic Program (MIQP). Formally it can be stated as follows:

$$\begin{aligned} \min_{u} \quad & u_r^T H_1 u_r + u_r^T H_2 u_b + u_b^T H_3 u_b + c_r^T u_r + c_b^T u_b \\ \text{s.t.} \quad & A_r u_r + A_b u_b \leq b \\ & A_{eq,r} u_r + A_{eq,b} u_b = b_{eq} \\ & u_b \in \{0,1\}^{n_b} \end{aligned} \tag{2.9}$$

Similarly to MILP problems, one has to choose between enumerating all the possible combinations or using more advanced techniques, like Branch & Bound to obtain the optimal solution.

Part II

HYBRID SYSTEMS

Chapter 3

Modeling of Hybrid Systems

3.1 Introduction to Hybrid Systems

Hybrid systems represent a compact framework which captures behavior of systems where continuous dynamics is coupled with discrete logic. Examples include, but are not limited to systems with discrete-valued actuators (such as on/off switches), piecewise linear nonlinearities, and finite state machines. Mathematically, hybrid systems can be described by the following frameworks:

- Piecewise affine systems (Sontag 1981)
- Mixed logical dynamical systems (Bemporad and Morari 1999b)
- Linear complementarity systems (Heemels et al. 2000)
- Extended linear complementarity systems (De Schutter 1999)
- Max-min-plus-scaling systems (De Schutter and Van den Boom 2001)

Under mild assumption all these frameworks are equivalent to each other (Heemels et al. 2001). In the sequel we will provide a comprehensive description of them.

This chapter is organized as follows. After formally stating the problem in Section 3.2, we give a detailed description of our approximation procedure in Section 3.3. Next in Section 3.4, we deal with the problem of obtaining the optimal PWA approximation from input-output measurements. Then, we illustrate the functionality of our software implementation in Section 3.5. Finally the proposed technique is demonstrated on a case study, including a model of a highly nonlinear reactor in Section 3.8. Material of this chapter is based on our results published in Kvasnica et al. (2010), Szűcs et al. (2011a), and Kvasnica et al. (2011c).

3.1.1 Piecewise Affine Systems

PWA systems are defined by partitioning the state-input space into polyhedral regions (2.12) and associating each region with a different linear (or affine) state-update equation:

$$x^+ = \begin{cases} A_1 x + B_1 u + c_1 & \text{if } \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in \mathcal{R}_1 \\ \quad \vdots & \quad \vdots \\ A_N x + B_N u + c_N & \text{if } \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in \mathcal{R}_N \end{cases} \tag{3.1}$$

Here $x \in \mathbb{R}^{n_x}$ is the state vector at time instance k, x^+ is the successor state at the next sampling instance, $x \in \mathbb{R}^{n_u}$ is the control action, $\mathcal{R}_i \subseteq \mathbb{R}^{n_x+n_u}$, $i = 1, \dots N$ are the polyhedral regions of the joint state-input space, and N is the number of individual affine dynamics. PWA systems arise naturally when nonlinear plants are approximated by the technique of multiple linearization (Sontag 1981).

3.1.2 Mixed Logical Dynamical Systems

MLD systems represent systems governed by discrete logic by a system of linear inequalities involving binary variables, which can be derived using so-called *big*–M formulation (Williams 1993). To illustrate the procedure, consider a logic statement of the following form

$$\delta = \begin{cases} 1 & \text{if } a^T x \leq b \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

which connects the truth value of a binary variable δ to satisfaction of the linear inequality $a^T x \leq b$ (which involves a real-valued variable $x \in \mathbb{R}^{n_x}$) via a logic equivalence relation. Let M and m denote, respectively, the maximum and minimum values which the linear expression $a^T x - b$ attains over the domain $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, i.e.

$$M = \max_{x \in \mathcal{X}} a^T x - b, \tag{3.3a}$$

$$m = \min_{x \in \mathcal{X}} a^T x - b. \tag{3.3b}$$

Then the `IF-THEN-ELSE` rule (3.2) is equivalent to satisfaction of the following system of linear inequalities:

$$a^T x - b \leq M(1 - \delta), \tag{3.4a}$$

$$a^T x - b \geq \epsilon + m\delta. \tag{3.4b}$$

Here, ϵ is a small constant, typically the machine precision, used to convert a strict inequality into a non-strict form. More complex logic expression involving e.g. one-way

implication ($\Leftarrow$ *or* $\Rightarrow$) and logic operations (and, or negation) can be translated in a similar fashion, see e.g. Bemporad and Morari (1999b), Williams (1993).

In the most general form, autonomous MLD systems are described by

$$x^+ = Ax + B_u u + B_\delta \delta + B_z z + B_0, \tag{3.5a}$$

$$E_x x + E_u u + E_\delta \delta + E_z z \leq E_0, \tag{3.5b}$$

where $x \in \mathbb{R}^{n_x}$ is the vector of states, $\delta \in \{0,1\}^{n_\delta}$ is the vector of binary variables, $z \in \mathbb{R}^{n_z}$ is the vector of auxiliary real variables, and $A, B_\delta, B_z, B_0, E_x, E_\delta, E_z, E_0$ are matrices or vectors of appropriate dimensions, Given a value of x and u, the state update x^+ can be computed by solving a feasibility problem, i.e. by finding a compatible combination of binary δ and real z variables satisfy constraints (3.5b). An illustrative example of transforming a PWA system into an MLD framework, as well as an extensive description of several fields of applications can be found in Bemporad and Morari (1999a).

3.1.3 Linear Complementarity Systems

In this section the linear complementary systems (LCS) are introduced. In general an LCS system can be described by the following equations

$$x^+ = Ax + Bu \tag{3.6a}$$

$$y = Cx + Du \tag{3.6b}$$

$$0 \leq y \perp x \geq 0 \tag{3.6c}$$

where A, B, C, D are matrices of appropriate dimension and variables u, x, y have dimensions R^m, R^n and R^p. The last equation denotes orthogonality between x and y, in other words, the dot product will be always zero, since the vectors are perpendicular. Modeling approach based on LCS systems is frequently used, for instance in electrical networks employing diodes. An exhaustive description of LCS systems, including, but not limited to their practical applications can be found in Heemels (1999).

3.1.4 Extended Linear Complementarity Systems

Extended linear complementarity systems (ELCS) are a special subclass of hybrid systems, which can be obtained from the above mentioned LCS systems. As it was described in the previous Section 3.1.3 the dynamical behavior of a general LCS system can be characterized by the equations (3.6a) (3.6b) subject to (3.6c), which implies that at each

time instant k we are capable to construct an index set $I \subseteq \{1, 2, \ldots, m\}$ such that

$$y_i = 0 \text{ for } i \in I \tag{3.7}$$

$$u_i = 0 \text{ for } i \notin I \tag{3.8}$$

Each index set represents a mode of the system. Therefore, in general there are 2^m different possible modes, but not all of them have to be necessarily feasible because of other constraints on u and y. By elimination of the variable y, we can obtain the set of equations,

$$Ax + Bu = 0 \tag{3.9a}$$

$$Cx + Du \geq 0 \tag{3.9b}$$

$$x \geq 0 \tag{3.9c}$$

$$(Cx + Du)^T x = 0 \tag{3.9d}$$

serving for description of such a model structure. Description and practical applicability of this framework can be found in De Schutter and Moor (1995; 1998).

3.1.5 Max-Min-Plus-Scaling Models

A max-min-plus-scaling (MMPS) function f of the variables $x_1, \ldots, x_n$ is defined by the recursive formula

$$f = x_i \vee \alpha \max(f_k, f_l) \vee \min(f_k, f_l) \vee f_k + f_l \vee \beta f_k, \tag{3.10}$$

with $i \in \{1, \ldots, n\}$, $\alpha, \beta \in \mathbb{R}$, and where f_k and f_l are again MMPS functions. The symbol "|" in (3.10) stands for "or". Consider a system that can be described by state space equations of the following form:

$$x^+ = M_x(x, u, v) \tag{3.11a}$$

$$y(k) = M_y(x, u, v), \tag{3.11b}$$

where M_x and M_y are MMPS functions, and where x is the state vector, $y(k)$ the output vector, and $u(k)$ and $v(k)$ are the input vectors. Systems, which behavior can be described by (3.11a) are also called extended *MMPS systems*. Typical examples of MMPS systems are digital circuits, computer networks and manufacturing plants. A gentle introduction to MMPS functions can be found in Heemels et al. (2001) and a more specific topic mainly focusing on utilising of the underlying mathematical framework in model predictive control can be found in De Schutter and van den Boom (2001).

3.2 Problem Statement

In this section we will show how to obtain the optimal PWA approximation of an arbitrary continous nonlinear function described by an analytical expression. We consider a generic dynamic system in discrete-time

$$x^+ = f(x,u), \tag{3.12}$$

where the vector field $f(\cdot,\cdot)$ is assumed to be continuous in the state variables $x \in \mathbb{R}^{n_x}$ and in the inputs $u \in \mathbb{R}^{n_u}$, and x^+ denotes the successor state. System states and inputs are assumed to be constrained to connected and closed domains $\mathcal{X} \subset \mathbb{R}^{n_x}$ and $\mathcal{U} \subset \mathbb{R}^{n_u}$, respectively.

The objective is to approximate (3.12) by a different dynamic system $x^+ = \tilde{f}(x,u)$ whose vector field $\tilde{f}(x,u)$ is a PWA function which consists of a pre-specified number N of local linear dynamics:

$$\tilde{f}(x,u) = \begin{cases} A_1 x + B_1 u + c_1 & \text{if } \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in \mathcal{R}_1 \\ \quad\vdots & \quad\vdots \\ A_N x + B_N u + c_N & \text{if } \left[\begin{smallmatrix} x \\ u \end{smallmatrix}\right] \in \mathcal{R}_N. \end{cases} \tag{3.13}$$

Here, $A_i \in \mathbb{R}^{n_x \times n_x}$, $B_i \in \mathbb{R}^{n_x \times n_u}$, $c_i \in \mathbb{R}^{n_x}$ are the state-update matrices of the i-th local linear approximation, and $\mathcal{R}_i \subset \mathbb{R}^{n_x \times n_u}$ is the region of validity of the i-th local model satisfying $\mathcal{R}_i \neq \emptyset$, $\text{int}(\mathcal{R}_i) \cap \text{int}(\mathcal{R}_j) = \emptyset$, $\forall i \neq j$, and $\cup_i \mathcal{R}_i = \mathcal{X} \times \mathcal{U}$.

Formally, the problem which we aim at solving can be stated as follows:

Problem 3.1 *Given a nonlinear vector field $f(x,u)$ of system (3.12), find the PWA approximation (3.13) of pre-specified complexity which minimizes the approximation error*

$$e_{aprx} := \int (f(x,u) - \tilde{f}(x,u))^2 \, dxdu, \tag{3.14}$$

where the integral is evaluated over the whole region of validity of (3.12), i.e. over $\mathcal{X} \times \mathcal{U}$.

Remark 3.1 *Since the approximation procedure discussed in the sequel considers only the vector field in the right-hand-side of (3.12), continuous-time systems $\dot{x} = f(x,u)$ can be treated as well.*

3.3 Optimal PWA Approximation

In this section we propose how to solve Problem 3.1, i.e. how to obtain an optimal PWA approximation $\tilde{f}$ of pre-specified complexity as in (3.13) which optimally approximates a given one-dimensional nonlinear function $f : \mathbb{R} \to \mathbb{R}$, provided that the analytical form of

f is known. Subsequently, we will show how to approximate nonlinear functions in higher dimensions. i.e. $f : \mathbb{R}^{n_z} \to \mathbb{R}$ by utilising the basic one-dimensional building block.

We distinguish between 3 special cases. The first one, described in Section 3.3.1, covers approximation of one-dimensional functions where $f : \mathbb{R} \to \mathbb{R}$ provided that the domain of f is connected and closed. Then, in Section 3.3.2 we show how to extend the procedure to approximation of multi-variable functions which satisfy a special property. Finally, in Section 3.3.3 we illustrate how to solve Problem 3.1 where f is an arbitrarily complex function, not satisfying any special properties.

3.3.1 Functions in One Variable

First, we consider the one-dimensional case, i.e. approximating a nonlinear function $f(z) : \mathbb{R} \to \mathbb{R}$, with a connected and closed domain $\mathcal{Z} \subset \mathbb{R}$, by a PWA function $\tilde{f}(z) = a_i z + c_i$ if $z \in \mathcal{R}_i$. Since $\mathcal{Z}$ is assumed to be connected and closed, it is a line segment $[\underline{z},\ \overline{z}]$. Regions $\mathcal{R}_i$ define the partition of such a line into N non-overlapping parts, i.e. $\mathcal{R}_1 = [\underline{z},\ r_1]$, $\mathcal{R}_2 = [r_1,\ r_2]$, …, $\mathcal{R}_{N-1} = [r_{N-2},\ r_{N-1}]$, $\mathcal{R}_N = [r_{N-1},\ \overline{z}]$ with $\cup_i \mathcal{R}_i = [\underline{z},\ \overline{z}]$. Solving Problem 3.1 then becomes to find the slopes a_i, offsets c_i and breakpoints r_i such that the approximation error is minimized, i.e.

$$\min_{a_i, c_i, r_i} \quad \int_{\underline{z}}^{\overline{z}} (f(z) - \tilde{f}(z))^2 \, dz \tag{3.15a}$$

$$\text{s.t.} \quad \tilde{f}(z) = \begin{cases} a_1 z + c_1 & \text{if } z \in [\underline{z},\ r_1] \\ \quad \vdots & \quad \vdots \\ a_N z + c_N & \text{if } z \in [r_{N-1},\ \overline{z}] \end{cases} \tag{3.15b}$$

$$\underline{z} \le r_1 \le \cdots \le r_{N-1} \le \overline{z}, \tag{3.15c}$$

$$a_i r_i + c_i = a_{i+1} r_i + c_{i+1},\ i = 1, \ldots, N-1, \tag{3.15d}$$

where (3.15d) enforces continuity of $\tilde{f}(z)$ along the breakpoints r_i. The IF-THEN based nonlinear constraint (3.15b) can be eliminated by observing that, by definition, regions $\mathcal{R}_i$ are non-overlapping, and the integral in (3.15a) can hence be written as

$$\int_{\underline{z}}^{\overline{z}} \big(f(z) - \tilde{f}(z)\big)^2 dz = \sum_{i=1}^{N} \Big(\int_{r_{i-1}}^{r_i} \big(f(z) - (a_i z + c_i)\big)^2 dz \Big), \tag{3.16}$$

with $r_0 = \underline{z}$ and $r_N = \overline{z}$. The nonlinear programming problem (NLP) (3.15) can therefore be written as

$$\min_{a_i, c_i, r_i} \quad \sum_{i=1}^{N} \Big(\int_{r_{i-1}}^{r_i} \big(f(z) - (a_i z + c_i)\big)^2 \, dz \Big) \tag{3.17a}$$

$$\text{s.t.} \quad \underline{z} \le r_1 \le \cdots \le r_{N-1} \le \overline{z}, \tag{3.17b}$$

$$a_i r_i + c_i = a_{i+1} r_i + c_{i+1},\ i = 1, \ldots, N-1. \tag{3.17c}$$

For simple functions $f(z)$, the integral in (3.17a) can be expressed in an analytical form in unknowns a_i, c_i, r_i, along with the corresponding gradients. For more complex expressions, the integrals can be evaluated numerically, e.g. by using the trapezoidal rule. In either case, problem (3.17) can be solved to a local optimality e.g. by using the `fmincon` solver of MATLAB. Alternatively, one can use global optimization methods (Adjiman et al. 1996b, Chachuat et al. 2006, Papamichail and Adjiman 2004) which guarantee that an ϵ-neighborhood of the global optimum can be found.

Example 3.1 *Consider the function $f(z) = z^3$ on domain $-1.5 \leq z \leq 1.5$. The analytic form of the integral (3.17a) is*

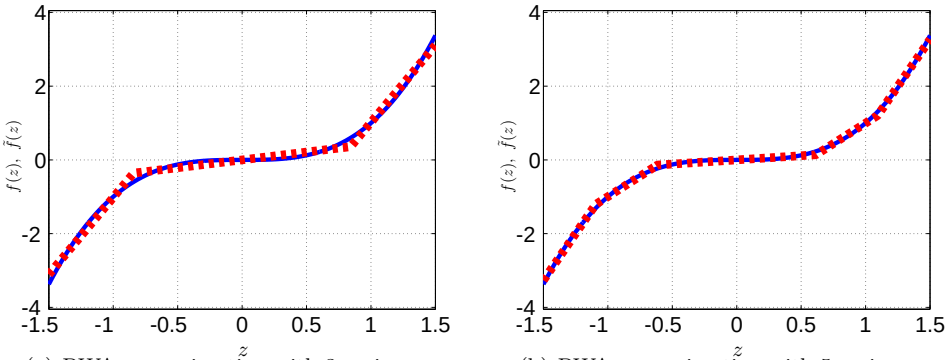
$$\sum_{i=1}^{N} \big(c_i^2(r_i + r_{i-1}) + a_i c_i(r_i^2 - r_i^2) + \frac{a_i^2}{3}(r_i^3 - r_{i-1}^3) - \\ -\frac{c_i}{2}(r_i^4 - r_{i-1}^4) - \frac{2a_i}{5}(r_i^5 - r_{i-1}^5) + \frac{1}{7}(r_i^7 - r_{i-1}^7)\big),$$

with $r_0 = -1.5$ and $r_N = 1.5$. The PWA approximation of $f(z)$ with $N = 3$ regions was obtained by solving the NLP (3.17) using `fmincon`, which took 0.12 s on a 3.4 GHz CPU running MATLAB 2012b. The obtained PWA approximation is then given by

$$\tilde{f}(z) = \begin{cases} 4.1797z + 3.1621 & \text{if } -1.5 \leq z \leq -0.8423 \\ 0.4257z & \text{if } -0.8423 \leq z \leq 0.8423 \\ 4.1797z - 3.1621 & \text{if } 0.8423 \leq z \leq 1.5 \end{cases}$$

The value of the integral error assuming only 3 approximation segments was 0.031. Naturally, quality of the approximation can be improved by increasing the complexity of the PWA function, i.e. by enlarging N. By simply increasing the number of approximation segments to 5 we were able to reduce the integral error to 0.005. Graphical representation of the corresponding PWA approximations with $N = 3$ and $N = 5$ are shown, respectively, in Figures 3.1(a) and 3.1(b).

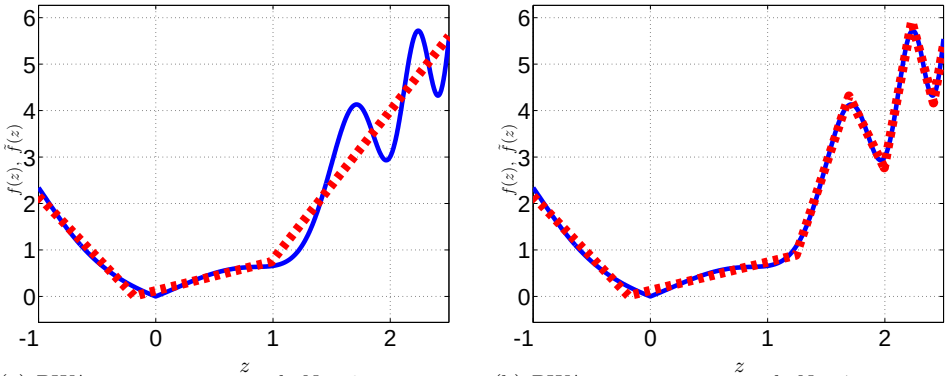
Example 3.2 *Consider the function $f(z) = |z| + 0.5z^2 - \sin(z^3)$ on domain $-1 \leq z \leq 2.5$, graph of which is shown in Figure 3.2(a). Since no analytic expression of the integral in (3.17a) could be obtained, we have opted for numeric integration of the cost while solving the NLP problem (3.17) by `fmincon`. The PWA approximations for $N = 3$ and $N = 7$ are shown in Figures 3.2(a) and 3.2(b). In case of 3 approximation segments the computation took 0.97 s, and the integral error was 0.65. By increasing the number of segments to 7, the computational procedure took approximately 3 s, but the approximation error was reduced by a factor of 30.*



(a) PWA approximation with 3 regions. (b) PWA approximation with 5 regions.

Figure 3.1: Graph of $f(z) = z^3$ (blue line) and the PWA approximations $\tilde{f}(z)$ (red dashed lines)

.



(a) PWA approximation with $N = 3$ regions. (b) PWA approximation with $N = 7$ regions.

Figure 3.2: Graph of $f(z) = |z| + 0.5z^2 - \sin(z^3)$ (blue line) and the PWA approximations $\tilde{f}(z)$ (red dashed lines).

3.3.2 Multivariable Separable Functions

The task is to approximate a given multivariable function $f(z_1, \dots, z_n) : \mathbb{R}^n \to \mathbb{R}$ with domain $\mathcal{Z} \subset \mathbb{R}^n$ by a PWA function $\tilde{f}(z_1, \dots, z_n)$, defined over the same domain, such that the approximation error (3.14) is minimized and the following assumption is fulfilled.

Assumption 3.1 *The function $f(z_1, \dots, z_n)$ can be written as $\sum_{i=1}^{n} \alpha_i \Big(\prod_{j=p_i}^{q_i} f_j(z_j) \Big)$.*

Definition 3.1 (Williams (1993)) *Function $f(z_1, \dots, z_n)$ is called* separable *if it can be expressed as the sum of functions of a single variable, i.e. $f(z_1, \dots, z_n) = f_1(z_1) + \cdots + f_n(z_n)$.*

If $f(z_1, \dots, z_n)$ is readily separable (e.g. when $f(z_1, z_2) = e^{z_1} + \sin(z_2)$), its optimal PWA approximation can be obtained by applying the 1D scenario of Section 3.3.1 to the individual components of the function, i.e. $\tilde{f}(z_1, \dots, z_n) = \tilde{f}_1(z_1) + \cdots + \tilde{f}_n(z_n)$. The total number of regions over which the PWA approximation $\tilde{f}(\cdot)$ is defined is hence given by $\sum_{j=1}^{n} N_j$, where N_j is the pre-specified complexity of the j-th approximation $\tilde{f}_j(z_j)$.

A surprisingly large number of non-separable functions can be converted into the separable form by applying a simple trick, elaborated in more details e.g. in Williams (1993). To introduce the procedure, consider a non-separable function $f(z_1, z_2) = z_1 z_2$ with domain $\mathcal{Z} := [\underline{z}_1,\ \overline{z}_1] \times [\underline{z}_2,\ \overline{z}_2]$. Define two new variables

$$y_1 = (z_1 + z_2), \quad y_2 = (z_1 - z_2). \tag{3.18}$$

Then it is easy to verify that $1/4(y_1^2 - y_2^2) = z_1 z_2$. The coordinate transformation therefore transforms the original function into a separable form, where both terms (y_1^2 and y_2^2) are now functions of a single variable. The procedure of Section 3.3.1 can thus be applied to compute PWA approximations of $f_{y_1}(y_1) := y_1^2$ and $f_{y_2}(y_2) := y_2^2$, where the function arguments relate to z_1 and z_2 via (3.18). Important to notice is that $f_{y_1}(\cdot)$ and $f_{y_2}(\cdot)$ have different domains, therefore their PWA approximations $\tilde{f}_{y_1}(y_1) \approx y_1^2$ and $\tilde{f}_{y_2}(y_2) \approx y_2^2$ will, in general, be different. Specifically, the domain of $f_{y_1}(\cdot)$ is $[\underline{y}_1,\ \overline{y}_1]$ with $\underline{y}_1 = \min\{z_1 + z_2 \mid \underline{z}_1 \le z_1 \le \overline{z}_1,\ \underline{z}_2 \le z_2 \le \overline{z}_2\}$ and $\overline{y}_1 = \max\{z_1 + z_2 \mid \underline{z}_1 \le z_1 \le \overline{z}_1,\ \underline{z}_2 \le z_2 \le \overline{z}_2\}$. Similarly, the domain of $f_{y_2}(\dot{)}$ is $[\underline{y}_2,\ \overline{y}_2]$, whose boundaries can be computed by respectively minimizing and maximizing $z_1 - z_2$ subject to the constraint $[z_1,\ z_2]^T \in \mathcal{Z}$. The overall PWA approximation $\tilde{f}(z_1, z_2) \approx z_1 z_2$ then becomes

$$\tilde{f}(z_1, z_2) = 1/4(\tilde{f}_{y_1}(z_1 + z_2) - \tilde{f}_{y_2}(z_1 - z_2)). \tag{3.19}$$

The value of $\tilde{f}(z_1, z_2)$ for any points z_1, z_2 is obtained by subtracting the value of the PWA function $\tilde{f}_{y_2}(\cdot)$ evaluated at the point $z_1 - z_2$ from the function value of $\tilde{f}_{y_1}(\cdot)$ evaluated at $z_1 + z_2$, followed by a linear scaling.

The procedure naturally extends to multivariable functions represented by the product of two nonlinear functions of a single variable, i.e. $f(z_1, z_2) = f_1(z_1)f_2(z_2)$. Here, the transformation (3.18) becomes

$$y_1 = f_1(z_1) + f_2(z_2), \quad y_2 = f_1(z_1) - f_2(z_2). \tag{3.20}$$

Therefore, $1/4(y_1^2 - y_2^2) = f(z_1, z_2)$ still holds. Let $f_{y_1}(y_1) := y_1^2$ and $f_{y_2}(y_2) := y_2^2$. The domain of $f_{y_1}(\cdot)$ is $[\underline{y}_1,\ \overline{y}_1]$ and $\mathrm{dom} f_{y_2}(\cdot) = [\underline{y}_2,\ \overline{y}_2]$ with

$$\underline{y}_1 = \min\{f_1(z_1) + f_2(z_2) \mid [z_1,\ z_2]^T \in \mathcal{Z}\}, \tag{3.21a}$$
$$\overline{y}_1 = \max\{f_1(z_1) + f_2(z_2) \mid [z_1,\ z_2]^T \in \mathcal{Z}\}, \tag{3.21b}$$
$$\underline{y}_2 = \min\{f_1(z_1) - f_2(z_2) \mid [z_1,\ z_2]^T \in \mathcal{Z}\}, \tag{3.21c}$$
$$\overline{y}_2 = \max\{f_1(z_1) - f_2(z_2) \mid [z_1,\ z_2]^T \in \mathcal{Z}\}, \tag{3.21d}$$

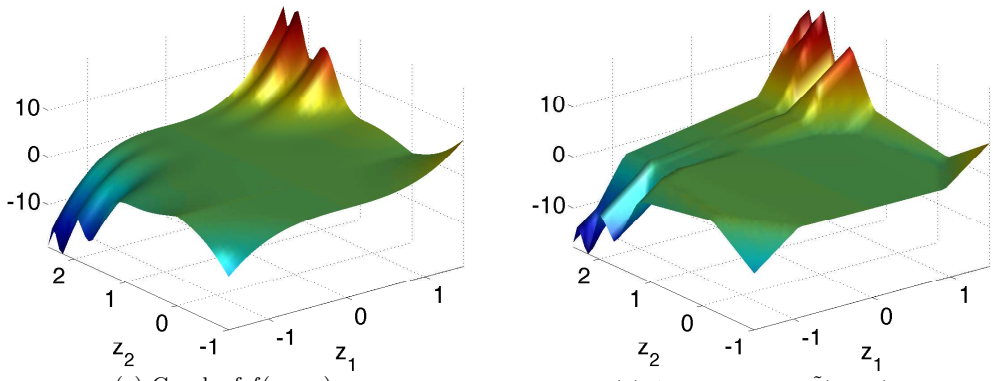
which can be computed by solving four NLP problems.

Finally, since all expressions are now functions of a single variable, the PWA approximations $\tilde{f}_1(z_1) \approx f_1(z_1)$, $\tilde{f}_2(z_2) \approx f_2(z_2)$, $\tilde{f}_{y_1}(y_1) \approx f_{y_1}(y_1)$, and $\tilde{f}_{y_2}(y_2) \approx f_{y_2}(y_2)$ can be computed by solving the NLP (3.17). The overall optimal PWA approximation $\tilde{f}(z_1, z_2) \approx f(z_1, z_2)$ then becomes

$$\tilde{f}(z_1, z_2) = 1/4\Big(\tilde{f}_{y_1}\big(\tilde{f}_1(z_1) + \tilde{f}_2(z_2)\big) - \tilde{f}_{y_2}\big(\tilde{f}_1(z_1) - \tilde{f}_2(z_2)\big)\Big). \tag{3.22}$$

The evaluation procedure is similar as above. I.e., given the arguments z_1 and z_2, one first evaluates $\tilde{z}_1 = \tilde{f}_1(z_1)$ and $\tilde{z}_2 = \tilde{f}_2(z_2)$. Subsequently, one evaluates $\tilde{y}_1 = \tilde{f}_{y_1}(\cdot)$ with the argument $\tilde{z}_1 + \tilde{z}_2$, then $\tilde{y}_2 = \tilde{f}_{y_2}(\cdot)$ at the point $\tilde{z}_1 - \tilde{z}_2$. Finally, $\tilde{f}(z_1, z_2) = 1/4(\tilde{y}_1 - \tilde{y}_2)$.

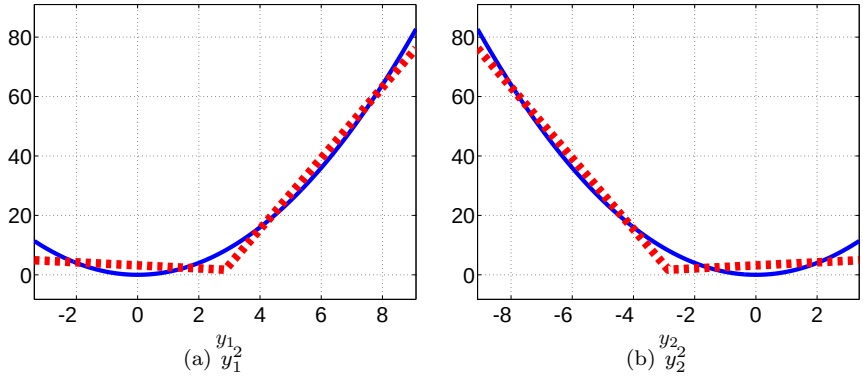
Example 3.3 *Consider a non-separable function given as the product of the two functions discussed in Examples 3.1 and 3.2, i.e.* $f(z_1, z_2) = f_1(z_1)f_2(z_2)$ *with* $f_1(z_1) = z_1^3$, $f_2(z_2) = |z_2| + 0.5z_2^2 - \sin(z_2)^3$ *on domain* $[-1.5,\ 1.5] \times [-1,\ 2.5]$. *Graph of the function is shown in Figure 3.3(a). In order to convert* $f(z_1, z_2)$ *into a separable form, we introduce variables* y_1 *and* y_2 *as per* (3.20). *The PWA approximation* $\tilde{f}(z_1, z_2) \approx f(z_1, z_2)$ *is then given by* (3.22). *Here,* $\tilde{f}_1(z_1)$ *was obtained by approximating* $f_1(z_1)$ *by a PWA function with* 3 *regions as shown in Figure 3.1(a), while* $\tilde{f}_2(z_2) \approx f_2(z_2)$ *was approximated by* 7 *regions as depicted in Figure 3.2(b). Subsequently, the domains* $[\underline{y}_1,\ \overline{y}_1]$ *and* $[\underline{y}_2,\ \overline{y}_2]$ *were computed via* (3.21), *which resulted into* $\mathrm{dom} y_1 = [-3.374,\ 9.095]$ *and* $\mathrm{dom} y_2 = [-9.095,\ 3.374]$. *Finally, the PWA approximations* $\tilde{f}_{y_1}(y_1) \approx y_1^2$ *and* $\tilde{f}_{y_2}(y_2) \approx y_2^2$ *were obtained by solving the NLP* (3.17) *with* $N = 2$. *Graphs of* y_1^2, y_2^2 *and their respective PWA approximations are presented in Figure 3.4. The overall approximation* $\tilde{f}(z_1, z_2)$ *therefore consists of* 14 *regions. Despite a rather crude approximation of the square functions, the combined PWA function* (3.22), *shown in Figure 3.3(b), features only a minor average approximation*



(a) Graph of $f(z_1, z_2)$. (b) Approximation $\tilde{f}(z_1, z_2)$

Figure 3.3: Graph of $f(z_1, z_2)$ and its PWA approximation (3.22) in Example 3.3.

error of 3% *and a worst-case error of* 15%. *By increasing the number of linearizations for* y_1^2 *and* y_2^2 *from* $N = 2$ *to* $N = 4$ *(hence increasing the complexity of* $\tilde{f}(z_1, z_2)$ *from* 14 *to* 18 *regions), the average and worst-case errors can be further reduced to* 1% *and* 8%, *respectively.*



(a) y_1^2 (b) y_2^2

Figure 3.4: Functions y_i^2 (blue) and their PWA approximation $\tilde{f}_{y_i}(y_i)$ (red dashed lines) in Example 3.3.

Separation of multivariable functions with more than two terms can be performed in an inductive manner. Consider $f(z_1, z_2, z_3) = f_1(z_1)f_2(z_2)f_3(z_3)$. First, approximate the product $f_1(z_1)f_2(z_2)$ by a PWA function of the form of (3.22), which requires four PWA approximations

$$\tilde{f}_1(\cdot) \approx f_1(\cdot),\ \tilde{f}_2(\cdot) \approx f_2(\cdot),\ \tilde{f}_{y_1}(\cdot) \approx y_1^2,\ \tilde{f}_{y_2}(\cdot) \approx y_2^2,$$

with y_1 and y_2 as in (3.20). Let $f_a(z_1, z_2) := f_1(z_1)f_2(z_2)$. Then $f(z_1, z_2, z_3) = f_a(z_1, z_2)f_3(z_3)$, which can again be approximated as a product of two functions. Specifically, define

$$y_3 = f_a(\cdot) + f_3(z_3), \quad y_4 = f_a(\cdot) - f_3(z_3), \tag{3.23}$$

and hence $f_a(z_1, z_2)f_3(z_3) = {}^1\!/_4(y_3^2 - y_4^2)$. The domains over which y_3^2 and y_4^2 need to be approximated are, respectively, $[\underline{y}_3,\ \overline{y}_3]$ and $[\underline{y}_4,\ \overline{y}_4]$ with

$$\underline{y}_3 = \min\{f_1(z_1)f_2(z_2) + f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{3.24a}$$
$$\overline{y}_3 = \max\{f_1(z_1)f_2(z_2) + f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{3.24b}$$
$$\underline{y}_4 = \min\{f_1(z_1)f_2(z_2) - f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{3.24c}$$
$$\overline{y}_4 = \max\{f_1(z_1)f_2(z_2) - f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{3.24d}$$

and $z = [z_1,\ z_2,\ z_3]^T$. Subsequently, three additional PWA approximations

$$\tilde{f}_{y_3}(y_3) \approx y_3^2,\ \tilde{f}_{y_4}(y_4) \approx y_4^2,\ \tilde{f}_3(z_3) \approx f_3(z_3)$$

need to be computed over the corresponding domains. The aggregated optimal PWA approximation $\tilde{f}(z_1, z_2, z_3) \approx f(z_1)f(z_2)f(z_3)$ consists of 7 individual approximations and is given by

$$\tilde{f}(\cdot) = {}^1\!/_4\Big(\underbrace{\tilde{f}_{y_3}\big(\hat{f}_a + \tilde{f}_3(z_3)\big)}_{\hat{y}_3} - \underbrace{\tilde{f}_{y_4}\big(\hat{f}_a - \tilde{f}_4(z_3)\big)}_{\hat{y}_4}\Big). \tag{3.25}$$

Here, $\hat{f}_a$ is the function value of $\tilde{f}_a(z_1, z_2) \approx f_1(z_1)f_2(z_2)$ at z_1 and z_2, where $\tilde{f}_a(\cdot)$ is obtained from (3.22), i.e.:

$$\hat{f}_a = {}^1\!/_4\Big(\underbrace{\tilde{f}_{y_1}\big(\tilde{f}_1(z_1) + \tilde{f}_2(z_2)\big)}_{\hat{y}_1} - \underbrace{\tilde{f}_{y_2}\big(\tilde{f}_1(z_1) - \tilde{f}_2(z_2)\big)}_{\hat{y}_2}\Big). \tag{3.26}$$

The overall PWA approximation $\tilde{f}(z_1, z_2, z_3)$ can then be evaluated, for any $z_1, z_2, z_3 \in \mathcal{Z}$, by computing the function values of the respective approximations in the following order:

Step 1: $\hat{y}_1 = \tilde{f}_{y_1}(\tilde{f}_1(z_1) + \tilde{f}_2(z_2))$,

Step 2: $\hat{y}_2 = \tilde{f}_{y_2}(\tilde{f}_1(z_1) - \tilde{f}_2(z_2)$,

Step 3: $\hat{y}_3 = \tilde{f}_{y_3}({}^1\!/_4(\hat{y}_1 - \hat{y}_2) + \tilde{f}_3(z_3))$,

Step 4: $\hat{y}_4 = \tilde{f}_{y_4}({}^1\!/_4(\hat{y}_1 - \hat{y}_2) - \tilde{f}_3(z_3))$,

Step 5: $\tilde{f}(z_1, z_2, z_3) = {}^1\!/_4(\hat{y}_3 - \hat{y}_4)$.

Such an inductive procedure can be repeated *ad-infimum* to derive PWA approximations of any multivariable function which satisfies Assumption 3.1. In general, the PWA

approximation will consists of $2p + n$ individual PWA functions, where n is the number of variables in $f(z_1, \ldots, z_n)$ and p is the number of products between individual subfunctions $f_j(z_j)$. As an example, for $f(\cdot) := \alpha_1 f_1(z_1) f_2(z_2) f_4(z_4) + \alpha_2 f_3(z_3) f_5(z_5)$ we have $p = 3$. We remark that inclusion of scalar multipliers α_j into the PWA description of the form (3.25)–(3.26) is straightforward and only requires linear scaling of the corresponding terms.

3.3.3 Multivariable Nonseparable Functions

When the nonlinear function $f : \mathbb{R}^{n_z} \to \mathbb{R}$ to be approximated does not satisfy Assumption 3.1, we propose to proceed as follows. As a rather general setup, consider that

$$f(z) = f_{\text{out},1}(f_{\text{out},2}(f_{\text{out},3}(\cdots(f_{\text{in}}(z))))) \tag{3.27}$$

with the inner function $f_{\text{in}} : \mathbb{R}^{n_z} \to \mathbb{R}$ satisfying Assumption 3.1 and arbitrary outer functions $f_{\text{out},i} : \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, m-1$. This relation can be further generalized to include sums and/or products of functions.

As a specific example, consider

$$f(z_1, z_2) = \exp(z_1 z_2), \tag{3.28}$$

where $f_{\text{in}}(z_1, z_2) = z_1 z_2$ and $f_{\text{out}}(w) = \exp(w)$. To derive an optimal PWA approximation $\tilde{f}$ of (3.28) , we introduce the substitution $w = f_{\text{in}}(z_1, z_2)$. Since f_{in} satisfies Assumption 3.1, the procedure of Section 3.3.2 can be applied to find its optimal PWA approximation $\tilde{w} = \tilde{f}_{\text{in}}(z_1, z_2) \approx z_1 z_2$. Define two new variables $y_1 = (z_1 + z_2)$ and $y_2 = (z_1 - z_2)$. Then $1/4(y_1^2 - y_2^2) = z_1 z_2$ trivially holds. Subsequently we can solve the NLP (3.17) to obtain optimal PWA approximations $\tilde{f}_{y_1}(y) \approx y^2$ on domain $[\underline{y}_1, \overline{y}_1]$ and $\tilde{f}_{y_2}(y) \approx y^2$ on domain $[\underline{y}_2, \overline{y}_2]$. We remark that although both functions to be approximated are the same (y^2), their respective domains will be different and are given by (3.21). Their PWA approximations will therefore differ as well. Next we derive a PWA approximation of $\tilde{f}_{\text{out}}(w) \approx \exp(w)$ again by solving (3.17). Value of the overall PWA approximation $\tilde{f}(z_1, z_2) \approx \exp(z_1 z_2)$ at a particular point (z_1, z_2) can then be obtained by evaluating the corresponding 1D approximations in the following order:

1. $\tilde{y}_1 = \tilde{f}_{y_1}(z_1 + z_2)$
2. $\tilde{y}_2 = \tilde{f}_{y_2}(z_1 - z_2)$
3. $\tilde{w} = 1/4(\tilde{y}_1 - \tilde{y}_2)$
4. $\tilde{f}(z_1, z_2) = \tilde{f}_{\text{out}}(\tilde{w})$

Such an substitution approach can be generalized to derive optimal PWA approximations of general nonlinear functions in the form of (3.27) by the following procedure:

1. Obtain optimal PWA approximation of the inner function $f_m(z)$ using the procedure in Section 3.3.2.

2. Define new variables w_i and approximate the 1D functions $f_i(w_i)$, $i = m-1, \dots, 1$, by solving (3.17).

If the multivariable inner function $f_{\text{in}} : \mathbb{R}^{n_z} \to \mathbb{R}$ with domain $\mathcal{Z}$ consists of more than two terms, its PWA approximation can be performed in an inductive manner. Consider $f_{\text{in}}(z_1, z_2, z_3) = f_1(z_1)f_2(z_2)f_3(z_3)$. First, approximate the product $f_1(z_1)f_2(z_2)$ by a PWA function of the form of (3.22), which requires four PWA approximations

$$\tilde{f}_1(\dot{)} \approx f_1(\cdot),\ \tilde{f}_2(\cdot) \approx f_2(\cdot),\ \tilde{f}_{y_1}(\cdot) \approx y_1^2,\ \tilde{f}_{y_2}(\cdot) \approx y_2^2,$$

with y_1 and y_2 as in (3.20). Let $f_a(z_1, z_2) := f_1(z_1)f_2(z_2)$. Then $f(z_1, z_2, z_3) = f_a(z_1, z_2)f_3(z_3)$, which can again be approximated as a product of two functions. Specifically, define

$$y_3 = f_a(\cdot) + f_3(z_3), \quad y_4 = f_a(\cdot) - f_3(z_3), \tag{3.29}$$

and hence $f_a(z_1, z_2)f_3(z_3) = 1/4(y_3^2 - y_4^2)$. The domains over which y_3^2 and y_4^2 need to be approximated are, respectively, $[\underline{y}_3,\ \overline{y}_3]$ and $[\underline{y}_4,\ \overline{y}_4]$ with

$$\underline{y}_3 = \min\{f_1(z_1)f_2(z_2) + f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{3.30a}$$
$$\overline{y}_3 = \max\{f_1(z_1)f_2(z_2) + f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{3.30b}$$
$$\underline{y}_4 = \min\{f_1(z_1)f_2(z_2) - f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{3.30c}$$
$$\overline{y}_4 = \max\{f_1(z_1)f_2(z_2) - f_3(z_3) \,|\, z \in \mathcal{Z}\}, \tag{3.30d}$$

and $z = [z_1,\ z_2,\ z_3]^T$. Subsequently, three additional PWA approximations

$$\tilde{f}_{y_3}(y_3) \approx y_3^2,\ \tilde{f}_{y_4}(y_4) \approx y_4^2,\ \tilde{f}_3(z_3) \approx f_3(z_3)$$

need to be computed over the corresponding domains. The aggregated optimal PWA approximation $\tilde{f}(z_1, z_2, z_3) \approx f(z_1)f(z_2)f(z_3)$ consists of 7 individual approximations and is given by

$$\tilde{f}_{\text{in}}(\cdot) = 1/4\Big(\underbrace{\tilde{f}_{y_3}\big(\hat{f}_a + \tilde{f}_3(z_3)\big)}_{\hat{y}_3} - \underbrace{\tilde{f}_{y_4}\big(\hat{f}_a - \tilde{f}_4(z_3)\big)}_{\hat{y}_4} \Big). \tag{3.31}$$

Here, $\hat{f}_a$ is the function value of $\tilde{f}_a(z_1, z_2) \approx f_1(z_1)f_2(z_2)$ at z_1 and z_2, where $\tilde{f}_a(\cdot)$ is obtained from (3.22), i.e.:

$$\hat{f}_a = 1/4\Big(\underbrace{\tilde{f}_{y_1}\big(\tilde{f}_1(z_1) + \tilde{f}_2(z_2)\big)}_{\hat{y}_1} - \underbrace{\tilde{f}_{y_2}\big(\tilde{f}_1(z_1) - \tilde{f}_2(z_2)\big)}_{\hat{y}_2} \Big). \tag{3.32}$$

The overall PWA approximation $\tilde{f}_{\text{in}}(z_1, z_2, z_3)$ can then be evaluated, for any z_1, z_2, $z_3 \in \mathcal{Z}$, by computing the function values of the respective approximations.

Such an inductive procedure can be repeated *ad-infimum* to derive PWA approximations of any multivariable inner function. In general, the PWA approximation will consists of $2p + n_z + m - 1$ individual PWA functions, where n_z is the number of variables, m is the number of functions in (3.27) and p is the number of products between individual subfunctions $f_j(z_j)$ in the inner function f_{in}. As an example, for $f_{\text{in}}(z) := \alpha_1 f_1(z_1) f_2(z_2) f_4(z_4) + \alpha_2 f_3(z_3) f_5(z_5)$ we have $p = 3$. We remark that inclusion of scalar multipliers α_j into the PWA description of the form (3.31)–(3.32) is straightforward and only requires linear scaling of the corresponding terms.

3.4 Approximation of Nonlinear Functions from Input-Output Data

This section deals with the problem of obtaining PWA approximation of arbitrary nonlinear function, when instead of an analytic expression only input-output measurements are given. Therefore, our first step is represented by seeking for an appropriate fitting function f. We propose to use a technique to compute an optimal linear combination of basis functions in the following form:

$$f(z) = \sum_{i=1}^{p} \alpha_i f_i(z), \tag{3.33}$$

where $\alpha_i \in \mathbb{R}$ are scalar multipliers and f_i are the basis functions. Therefore, in Section 3.4.2 we propose three different approaches to obtain the coefficients of the underlying fitting function. The most straightforward way to obtain the coefficients of the underlying linear combination is to solve a simple unconstrained optimisation problem. Further extension to the aforementioned approach is represented by a constrained quadratic program, which purpose is to minimise the cardinality of the vector containing the coefficients multiplying the respective basis functions. Alternatively, one can directly minimise the number of non-zero coefficients, however, this approach requires solving a mixed-integer linear programming problem.

After having obtained the desired analytic formula, we can proceed in our approximation procedure as it was described in Section 3.3.

3.4.1 Problem Definition

We are given a T samples of input data $z_i \in \mathcal{Z} \subset \mathbb{R}^{n_z}$ from some closed and bounded set $\mathcal{Z}$, and the corresponding measurements $y_i \in \mathbb{R}$, $i = 1, \ldots, T$. We want to fit the data

with a PWA function $\tilde{f} : \mathbb{R}^{n_z} \to \mathbb{R}$ with N regions which satisfies two design requirements:

R1: $\tilde{f}$ is well-posed (Bemporad and Morari 1999c) on $\mathcal{Z}$, i.e it satisfies $\text{int}(\mathcal{R}_i) \cap \text{int}(\mathcal{R}_j) = \emptyset$, $\forall i \neq j$ and $\cup_j \mathcal{R}_j = \mathcal{Z}$, $j = 1, \ldots, N$.

R2: $\tilde{f}$ is a good fit which achieves a low fitting error $e_{\text{fit}} = \sum_{i=1}^{T} (y_i - \tilde{f}(z_i))^2$.

Solving this problem (i.e. determining regions $\mathcal{R}_j \subseteq \mathbb{R}^{n_z}$ and parameters $a_j \in \mathbb{R}^{n_z}$, $c_j \in \mathbb{R}$, $j = 1, \ldots, N$), however, is not trivial (Kvasnica et al. 2011c) if the input samples z_i are vectors, i.e. when $n_z > 1$. The difficult part is how to divide the domain $\mathcal{Z}$ into non-overlapping regions $\mathcal{R}_j$ without creating "holes", i.e. guaranteeing that the union $\cup_j \mathcal{R}_j$ completely covers $\mathcal{Z}$ if dimension$(\mathcal{Z}) > 1$.

To overcome this difficulty, we propose to split the search for the PWA function $\tilde{f}$ into two steps. In the first stage we fit the input data, represented by the (z_i, y_i) pairs, with a nonlinear function $y = f(z)$:

Problem 3.2 *Given are T samples of input-output data (z_i, y_i), $i = 1, \ldots, T$. Fit the data with a multivariable function $f : \mathbb{R}^{n_z} \to \mathbb{R}$ such that the fitting error $e_{fit} = \sum_{i=1}^{T} (y_i - f(z_i))^2$ is minimized.*

Once the analytical form of the fitting function f is available, in the second step we search for its optimal PWA approximation, as it was described in the previous sections.

In order to fulfill the requirements stated in 3.4.1 in Section 3.4.2 we propose to solve Problem 3.2 by a simple unconstrained optimization problem to find an appropriate linear combination of the basis functions.

3.4.2 Function Fitting

To solve Problem 3.2 we need to determine the analytical form of the fitting function f which minimizes the fitting error $\sum_{i=1}^{T} (y_i - f(z_i))^2$. The usual approach is to select a subspace of *basis* functions $f_1, \ldots, f_n : \mathbb{R}^{n_z} \to \mathbb{R}$ such that

$$f(z) = \alpha_1 f_1(z) + \cdots + \alpha_n f_n(z). \tag{3.34}$$

The task then is to determine coefficients $\alpha_i \in \mathbb{R}$, $i = 1, \ldots, n$ which parametrize f and provide an optimal fit.

Needless to say, selection of the basis functions is crucial in obtaining a good fit. In many situations the basis is chosen by hand, employing prior knowledge about the analytical form of the nonlinearity from which the input-output data originated. One such an example was provided in Section 3.3.3.

If this prior information is not available, one can resort to a rather broad selection of basis functions (Boyd and Vandenberghe 2004b). One common subspace of functions on

$\mathbb{R}$ consists of polynomials of degree less than n. The simplest basis consists of the powers, i.e. $f_i(z) = z^{i-1}, i = 1, \ldots, n$.

We can also consider polynomials on $\mathbb{R}^{n_z}$, with a maximum total degree n

$$f_i(z) = \sum_{i_1+\cdots+i_n \leq n} z_1^{i_1} \cdots z_{n_z}^{i_n}, \tag{3.35}$$

or a maximum degree for each variable. An another common option is to use trigonometric polynomials of degree less than n with basis

$$\sin(kz),\ k = 1, \ldots, n-1, \quad \cos(kz), k = 0, \ldots, n-1. \tag{3.36}$$

Regardless of the choice of the basis functions, it is important to notice that f as in (3.34) is linear in the unknown coefficients $\alpha_1, \ldots, \alpha_n$. Therefore Problem 3.2 can be easily solved by solving a simple unconstrained optimization problem in the form

$$\min \sum_{i=1}^{T} (y_i - f(z_i))^2. \tag{3.37}$$

The unknown coefficients α_i can be obtained e.g. by taking derivative of (3.37) equal to zero.

The fitting problem (3.37) can be further extended to obtain a simple form of the fitting function f by minimizing the cardinality of the vector of parameters $\alpha = [\alpha_1, \ldots, \alpha_n]$ in (3.34). A simple heuristic approach would be to minimize the 1-norm of α (Boyd and Vandenberghe 2004b):

$$\min \sum_{i=1}^{T} (y_i - f(z_i))^2 + \gamma \|\alpha\|_1, \tag{3.38}$$

which can be cast as a constrained quadratic program. The tuning parameter $\gamma > 0$ here acts as a regularization coefficient.

A more rigorous approach is to directly minimize the number of non-zero components of α. This can be achieved by introducing a set of binary indicators $\delta_j \in \{0, 1\}$, $j = 1, \ldots, n$ which fulfill

$$(\alpha_j \neq 0) \Rightarrow (\delta_j = 1). \tag{3.39}$$

By employing the big-M technique (Bemporad and Morari 1999c, Williams 1993) we can rewrite (3.39) into a set of inequalities which are linear in δ_j and α_j:

$$-M\delta_j \leq \alpha_j \leq M\delta_j, \tag{3.40}$$

where M is a sufficiently large number. It is then easy to verify that minimization of the number of nonzero components amounts to minimizing the sum of corresponding binary

indicators, i.e.

$$\min \quad \sum_{i=1}^{T}(y_i - f(z_i))^2 + \gamma \sum_{j=1}^{n} \delta_j \tag{3.41a}$$

$$\text{s.t.} \quad -M\delta_j \leq \alpha_j \leq M\delta_j, \; j = 1, \ldots, n, \tag{3.41b}$$

which provides a good fit of minimal cardinality. Problem (3.41) is a mixed-integer quadratic program which can be solved to global optimality using state-of-the-art solvers (ILOG, Inc. 2003, Löfberg 2004). Complexity of (3.41) is primarily determined by the number of binary variables, i.e. by the number n of basis functions considered in (3.34).

Apart from the technique, based on solving an unconstrained optimization problem, one can apply approaches seeking the optimal PWA affine approximation problem from input-output data by using an alternative method employing neural networks (Števek et al. 2012). In this work we have proposed to exploit the advantages provided by orthogonal activated function based neural networks (OAF–NN) to obtain the parameters of the linear combination of the final fitting function.

3.4.3 Complete Scheme

In order to solve Problem 3.2, one needs combine results of Section 3.4 with the approaches of Section 3.3 in the following manner:

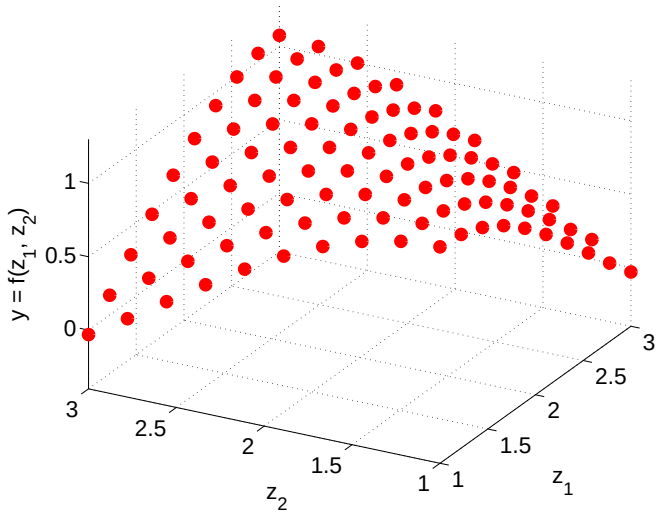
1. Obtain the analytic form of the function (3.34), which best fits the given set of data, applying the technique described in Section 3.4.2

2. Find the optimal PWA approximation of the obtained analytic fit by one of the methods described in Sections 3.3.1, 3.3.2, and 3.3.3.

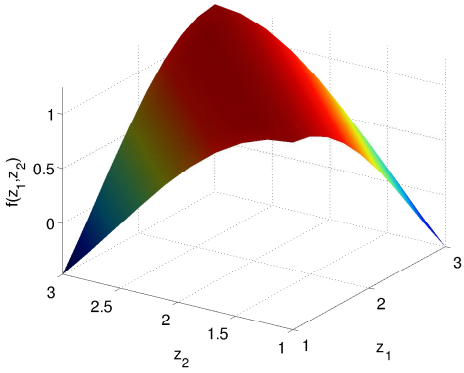
Example 3.4 *Consider a set of input-output data shown in Figure 3.5(a). To fit these data with a PWA function, we have first applied the procedure of Section 3.4.2 to obtain an optimal fit by the function* $f(z) = \sum_{i=1}^{3} \alpha_i f_i(z)$ *which consists of basis functions* $f_1 = 1$, $f_2 = \sin(z_1 z_2)$ *and* $f_3 = \cos(z_1 - z_2)$. *By solving (3.37) we have obtained*

$$f(z_1, z_2) = 0.02 + 0.08 \sin(z_1 z_2) + 1.2 \cos(z_1 - z_2), \tag{3.42}$$

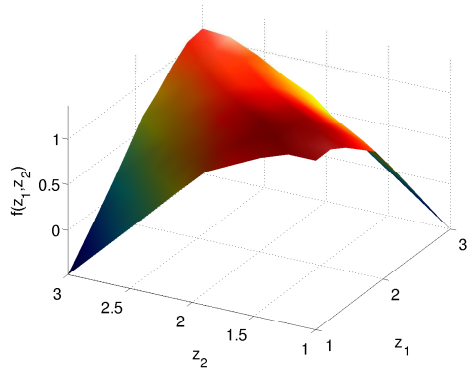
shown in Figure 3.5(b). To derive an optimal PWA approximation of f *in (3.42) we have applied the aforementioned procedure to approximate* $\sin(z_1 z_2)$ *by first approximating* $z_1 z_2$ *by a PWA function* $\tilde{f}_1(z_1, z_2)$ *and* $\sin(w)$ *by* $\tilde{f}_2(w)$. *Approximation of* $\cos(z_1 - z_2)$ *was performed in a similar manner. The resulting PWA approximation of (3.42), consisting of 15 regions, is depicted in Figure 3.5(c).*



(a) Input-output data (z_i, y_i) with $z_i \in \mathbb{R}^2$.



(b) Optimal fit with a trigonometric polynomial basis.

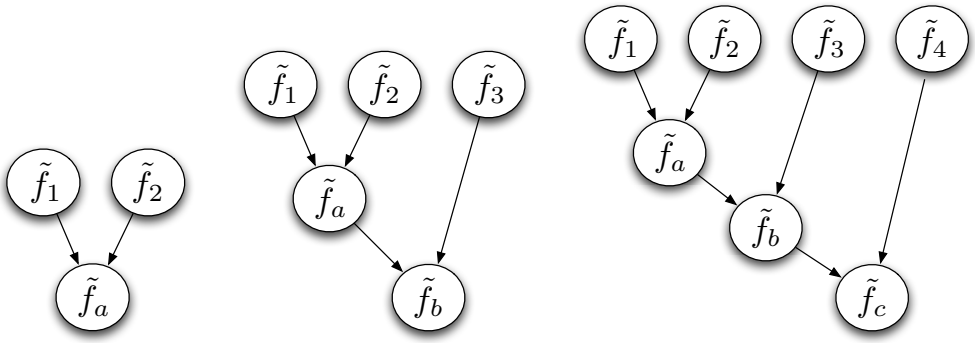


(c) Graph of optimal PWA approximation $\tilde{f}(z_1, z_2)$

Figure 3.5: Two-dimensional fit from Example 3.4.

3.5 Software Implementation

An algorithmic implementation of the inductive separation procedure of Section 3.3.2 is discussed next, provided that all functions are given in their symbolic representation. The procedure relies on two basic building blocks. The first one, represented by Algorithm 3.5.1, constructs the PWA approximation of a product of two functions, i.e. computes $\tilde{f}(z_i, z_j) \approx f_i(z_i) f_j(z_j)$. Strictly speaking, the algorithm differentiates between two scenarios. If either f_i or f_j are PWA functions which approximate the product of some other functions (say $f_i \approx f_p f_q$), then $\tilde{f} \approx f_i f_j$ is computed as shown in (3.29)–(3.32).



(a) Step 1. (b) Step 2. (c) Step 3.

Figure 3.6: Parsing tree $\mathcal{T}$ built by Algorithm 3.5.2.

Otherwise the procedure evidenced by (3.18)–(3.22) is followed.

Algorithm 3.5.2 then utilizes this block to construct a parse tree which defines the PWA approximation of the product of multiple functions, i.e. $\prod_{i=1}^{n} f_i(z_i)$. To illustrate the procedure, consider $f(z_1, z_2, z_3, z_4) = f_1(z_1)f_2(z_2)f_3(z_3)f_4(z_4)$. First, the stack of "unexplored" functions $\mathcal{S} = \{f_4, f_3, f_2, f_1\}$ is formed. In the first pass of the `while` cycle, f_1 and f_2 are popped from the stack and the PWA approximation $\tilde{f}_a \approx f_1 f_2$ is computed by Algorithm 3.5.1. Subsequently, $\tilde{f}_a$ is pushed back to $\mathcal{S}$ (which then becomes $\mathcal{S} = \{f_4, f_3, \tilde{f}_a\}$), and new nodes of the parse tree $\mathcal{T}$ are created as shown in Figure 3.6(a). The procedure then repeats from Step 4. I.e., f_3 and $\tilde{f}_a$ are popped from $\mathcal{S}$, $\tilde{f}_b \approx f_3 f_a$ is computed, and the parse tree is updated as illustrated in Figure 3.6(b). Due to Step 6, $\mathcal{S} = \{f_4, \tilde{f}_b\}$, and the algorithm therefore performs one more pass at which $\tilde{f}_c \approx f_4 f_b$ is created and inserted into the tree, which finally looks like in Figure 3.6(c). The algorithm thereupon terminates since $\mathcal{S} = \{\tilde{f}_c\}$ contains a single element.

If the function to be approximated contains sums of products, e.g. when $f(z_1, z_2, z_3, z_4) = \alpha_1 f_1(z_1)f_2(z_2) + \alpha_2 f_3(z_3)f_4(z_4)$, separate parsing trees have to be built by Algorithm 3.5.2 for each component of the summation. We remark that treating the scaling factors α_i only involves scaling the bottom-most node of the corresponding tree by the respective α_i.

The parsing tree generated by Algorithm 3.5.2 can be readily used to convert the PWA approximation $\tilde{f}(z_1, \ldots, z_n) \approx \sum_i \alpha_i \prod_j f_j(z_j)$ into a suitable mathematical model, which can subsequently be used for simulations, analysis, or control synthesis. Therefore we have created a software tool which takes a parsing tree $\mathcal{T}$ (or several such trees to accommodate for sums of products of functions), and *automatically* generates the corresponding HYSDEL representation of such a PWA approximation.

Next, we discuss software implementation of the approximation procedure described

Algorithm 3.5.1 PWA approximation of $f_i(z_i)f_j(z_j)$

REQUIRE: Functions $f_i(z_i)$, $f_j(z_j)$.

OUTPUT: Approximation $\tilde{f}(z_i, z_j) \approx f_i(z_i)f_j(z_j)$.

1: Obtain the PWA approximations $\tilde{f}_i(z_i) \approx f_i(z_i)$ and $\tilde{f}_j(z_j) \approx f_j(z_j)$ by solving two NLPs (3.17).

2: Get $\underline{y}_i$, $\overline{y}_i$, $\underline{y}_j$, and $\overline{y}_j$ from (3.21) or (3.30).

3: Compute the PWA approximations $\tilde{f}_{y_i}(y_i) \approx y_i^2$ and $\tilde{f}_{y_j}(y_j) \approx y_j^2$ on domains $[\underline{y}_i,\ \overline{y}_i]$ and $[\underline{y}_j,\ \overline{y}_j]$ by solving two NLPs (3.17).

4: **return** $\tilde{f}_i(z_i)$, $\tilde{f}_j(z_j)$, and the symbolic representation of $\tilde{f}(z_i, z_j)$.

Algorithm 3.5.2 PWA approximation of $\prod_{i=1}^n f_i(z_i)$

REQUIRE: Functions $f_i(z_i)$.

OUTPUT: $\tilde{f}(z_1, \ldots, z_n) \approx \prod_{i=1}^n f_i(z_i)$.

1: Create an empty last-in-first-out stack $\mathcal{S}$ and an empty tree $\mathcal{T}$.

2: Push $f_i(z_i)$, $i = n, \ldots, 1$ to the stack $\mathcal{S}$.

3: **while** $\mathcal{S}$ has more than one element **do**

4: Pop two elements $f_j(z_j)$ and $f_k(z_k)$ from $\mathcal{S}$.

5: Obtain $\tilde{f}_j(z_j)$, $\tilde{f}_j(z_j)$, and $\tilde{f}(z_j, z_k) \approx f_j(z_j)f_k(z_k)$ by calling Algorithm 3.5.1.

6: Push $\tilde{f}(z_j, z_k)$ to $\mathcal{S}$.

7: Create nodes $\tilde{f}_j(z_j)$, $\tilde{f}_k(z_k)$ and insert them to $\mathcal{T}$.

8: Create a node $\tilde{f}(z_j, z_k)$ and append it as a child of nodes $\tilde{f}_j(z_j)$ and $\tilde{f}_k(z_k)$.

9: **end while**

10: **return** Tree $\mathcal{T}$ representing $\tilde{f}(z_1, \ldots, z_n) \approx \prod_{i=1}^n f_i(z_i)$.

above. The implementation is provided in a form of an open-source MATLAB toolbox, called AUTOPROX, which is freely available from `http://www.kirp.chtf.stuba.sk/~sw/`. The toolbox provides two types of user interfaces. Input data can either be provided directly from the command line or, alternatively, entered using a graphical interface.

3.6 Command-Line Interface

The command-line interface is illustrated first by revisiting Example 3.1. To approximate the function $f(z) = z^3$, one proceeds as follows:

```
syms z
f = z^3
bounds = [-1.5, 1.5]
regions = 3
[aprx, data] = autoprox_1d(f, bounds, regions)
```

Here, AUTOPROX uses the Symbolic Toolbox to define symbolic representation of the function to be approximated on a given domain (represented by the `bounds` variable), with a given number of PWA segments (the `regions` variable). The first output argument (denotes as `aprx` here) is a function handle, which can be used e.g. to plot the approximation:

```
x = -1.5:0.001:1.5
plot(x, x.^3, x, aprx(x), '--')
```

which will generate a plot as seen in Figure 3.1(a). The second output (stored in the `data` variable) can be used to export the PWA approximation into the HYSDEL language:

```
hysdel_1d(data, 'filename.hys')
```

The generated HYSDEL model can be subsequently compiled by the HYSDEL compiler, which will provide a mathematical model suitable e.g. for control synthesis.

Approximation of 2D functions can be performed in a similar manner. Let us again consider Example 3.4, i.e. the task is to approximate the function $f(z_1, z_2) = z_1^3(|z_2| + 0.5z_2^2 - \sin(z_2)^3)$ on domain $[-1.5,\ 1.5] \times [-1,\ 2.5]$. Again, the first step is to define the function using symbolic variables:

```
syms z1 z2
f1 = z1^3
f2 = abs(z2) + 0.5*z2^2 - sin(z2^3)
```

Next, the function domain and number of approximation segments need to be provided:

```
f1_bounds = [-1.5, 1.5]
f2_bounds = [-1, 2.5]
f1_regions = 3
```

```
f2_regions = 7
y1_regions = 2
y2_regions = 2
```

Finally, the approximation $\tilde{f}(z_1, z_2)$ can be obtained by calling

```
[aprx,data] = autoprox_2d(f1,f2,f1_bounds,f2_bounds,...
 f1_regions, f2_regions,y1_regions, y2_regions)
```

Similarly as in the previous example, the `aprx` output is a function handle which can be used to directly evaluate the approximation at some given values of z_1 and z_2, e.g.

```
z1 = 0.5
z2 = -1
true_value = z1^3*(abs(z2) + 0.5*z2^2 - sin(z2^3))
aprx_value = aprx(z1, z2)
```

The second output (called `data`) again serves to generate the HYSDEL version of the approximation:

```
hysdel_2d(data, 'filename.hys')
```

Approximation of n-dimensional functions can be obtained by calling the `autoprox_nd` function. A detailed description of its calling syntax is omitted due to brevity, but is provided in the distribution package of AUTOPROX.

3.7 Graphical User Interface (GUI)

The GUI allows to perform the approximation in an easily accessible manner where all data can be entered conveniently without the need to remember the exact calling syntax of individual approximation functions.

The main window of the GUI is shown in Figure 3.7. The user starts by selecting the type of approximation using radio buttons. Then, he provides the symbolic representation of the function to approximate in the *FUNCTION* text box. The domain of the function, represented by its minimal and maximal bounds, has to be filled out next. After providing all necessary details, the user can select the number of approximation regions by a drop-down menu, as shown in Figure 3.8. Afterwards, the approximation is computed by clicking the *SPLIT* button. A concise statistical evaluation of the approximation will then appear in a corresponding section of the GUI. It informs the user about the approximation quality, represented by average and worst-case approximation errors. Finally, the approximation can be exported to a HYSDEL source by clicking the *EXPORT* button.

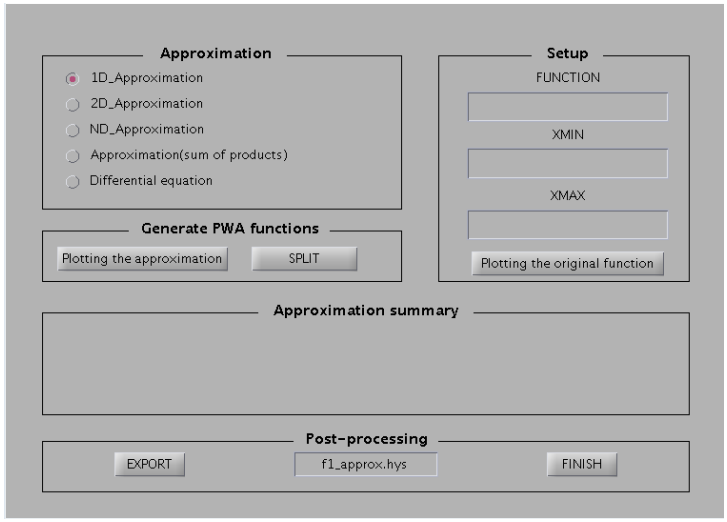


Figure 3.7: Basic GUI window.

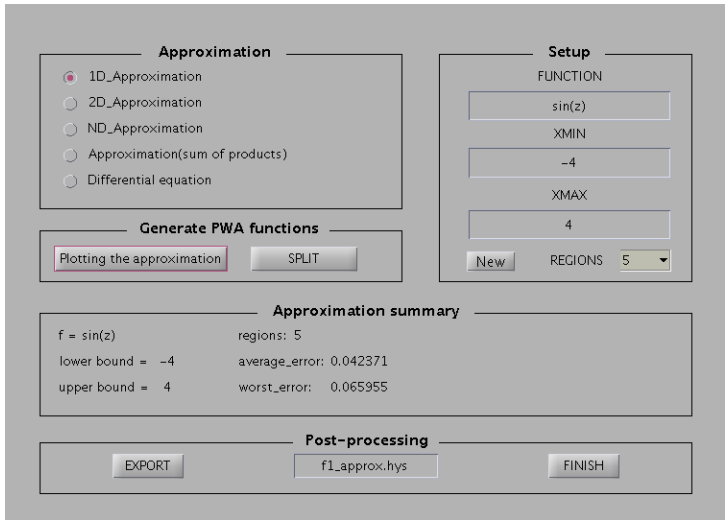


Figure 3.8: GUI windows after performing approximation.

3.8 Case Study

Consider a continuous stirred tank reactor (CSTR) where the reaction $A \rightarrow B$ takes place. The source compound is pumped into the reactor at a constant inflow with a constant concentration. The chemical reaction is exothermic and a coolant liquid is therefore pumped into the reactor's jacket to prevent overheating. The input temperature of the coolant is constant, while its flow rate q_c can be manipulated and is considered an

exogenous input. Concentration of the reactant c_A inside of the reactor, temperature of the reactor mixture ϑ, and temperature of the cooling liquid in the jacket ϑ_c are the state variables of the CSTR. The normalized material and energy balances of such a reactor are then given by

$$\begin{aligned} \dot{c}_A &= \alpha_1 - \alpha_2 c_A - \alpha_3 c_A e^{-\beta/\vartheta}, \\ \dot{\vartheta} &= \alpha_4 - \alpha_5 \alpha_2 c_A e^{-\beta/\vartheta} + \alpha_6 \vartheta + \alpha_7 \vartheta_c, \\ \dot{\vartheta}_c &= \alpha_8 q_c + \alpha_9(\vartheta - \vartheta_c) - \alpha_{10}\vartheta_c q_c, \end{aligned} \tag{3.43}$$

with constants α_i and β. The state and input variables are considered to belong to intervals $c_A \in [4,\ 4.2]\,\mathrm{mol\,m^{-3}}$, $\vartheta \in [300,\ 320]\,\mathrm{K}$, $\vartheta_c \in [290,\ 310]\,\mathrm{K}$, and $q_c \in [0.002,\ 0.02]\,\mathrm{m^3\,h^{-1}}$.

The model features two nonlinearities: $\vartheta_c q_c$ and $c_A e^{-\beta/\vartheta}$, both of which satisfy Assumption 3.1. Since the first one involves a direct product of two variables, its PWA approximation $\tilde{f}_a \approx \vartheta_c q_c$ can be obtained as in (3.19) by first defining $y_1 = \vartheta_c + q_c$, $y_2 = \vartheta_c - q_c$, followed by approximating the functions y_1^2 and y_2^2 by $\tilde{f}_{y_1}(y_1)$ and $\tilde{f}_{y_2}(y_2)$, respectively. Hence, the approximation $\tilde{f}_1(\vartheta_c, q_c) \approx \vartheta_c q_c$ is represented by

$$\tilde{f}_1(\vartheta_c, q_c) = {}^1\!/_4\big(\tilde{f}_{y_1}(\vartheta_c + q_c) - \tilde{f}_{y_2}(\vartheta_c - q_c)\big). \tag{3.44}$$

The second nonlinearity can be approximated as in (3.22). First, the PWA approximation $\tilde{g}(\vartheta) \approx e^{-\beta/\vartheta}$ is computed by solving (3.17). Then, $y_3 = c_A + e^{-\beta/\vartheta}$, $y_4 = c_A - e^{-\beta/\vartheta}$ are defined, followed by computing the respective PWA approximations $\tilde{f}_{y_3}(y_3) \approx y_3^2$ and $\tilde{f}_{y_4}(y_4) \approx y_4^2$. $\tilde{f}_2(c_A, \vartheta) \approx c_A e^{-\beta/\vartheta}$ is thus given by

$$\tilde{f}_2(c_A, \vartheta) = {}^1\!/_4\Big(\tilde{f}_{y_3}\big(c_A + \tilde{g}(\vartheta)\big) - \tilde{f}_{y_4}\big(c_A - \tilde{g}(\vartheta)\big)\Big) \tag{3.45}$$

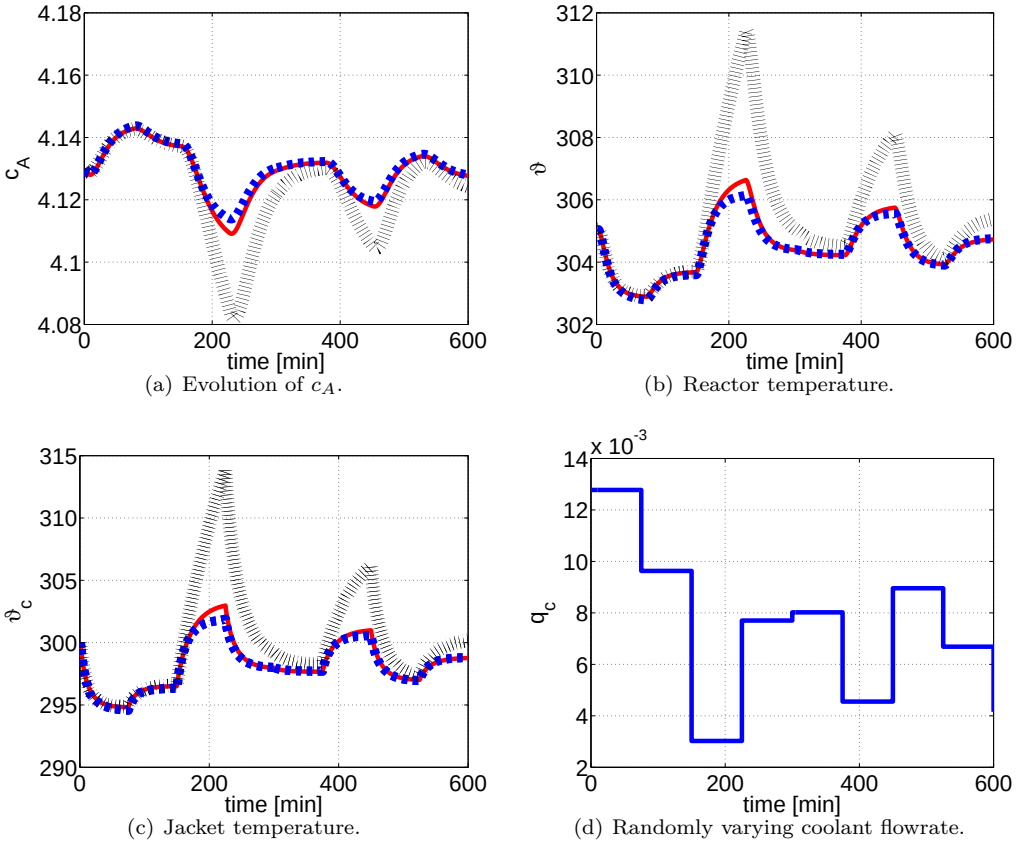
The overall PWA approximation of the original nonlinear system $\dot{x} = f(x, u)$ with $x = [c_A,\ \vartheta,\ \vartheta_c]^T$ and $u = q_c$ is thus

$$\begin{aligned} \dot{c}_A &\approx \alpha_1 - \alpha_2 c_A - \alpha_3 \tilde{f}_2(c_A, \vartheta)\big), \\ \dot{\vartheta} &\approx \alpha_4 - \alpha_5 \alpha_2 \tilde{f}_2(c_A, \vartheta) + \alpha_6 \vartheta + \alpha_7 \vartheta_c + \vartheta, \\ \dot{\vartheta}_c &\approx \alpha_8 q_c + \alpha_9(\vartheta - \vartheta_c) - \alpha_{10}\tilde{f}_1(\vartheta_c, q_c), \end{aligned} \tag{3.46}$$

which can be easily converted into the general PWA form (3.13) as described in Section 3.2.

To assess approximation accuracy, we have investigated the open-loop evolution of the original nonlinear model (3.43) and compared it to the behavior of its PWA approximation (3.46). To derive the PWA model, we have chosen 3 regions for $\tilde{f}_{y_1}(\cdot)$, $\tilde{f}_{y_2}(\cdot)$ in (3.44) and $\tilde{f}_{y_3}(\cdot)$, $\tilde{f}_{y_4}(\cdot)$ in (3.45), and $N = 2$ for $\tilde{g}(\theta) \approx e^{-\beta/\vartheta}$. The simulation results are shown in Figure 3.9. To better illustrate advantages of the PWA approximation, the simulation scenario also shows evolution of linearized version of (3.43) around the nominal steady

state $c_A^s = 4.13$, $\vartheta^s = 304$, $\vartheta_c^s = 297$, and $q_c^s = 0.006$. As can be seen from the results, the PWA approximation clearly outperforms the model based on a single linearization. Specifically, the model (3.46) provides a 15 times more accurate tracking of the nonlinear profile compared to the linear model. Important to notice is that the PWA model consists of 14 local linear models. By increasing N to 7 when approximating $\tilde{f}_{y_1}(\cdot)$, $\tilde{f}_{y_2}(\cdot)$ in (3.44) and $\tilde{f}_{y_3}(\cdot)$, $\tilde{f}_{y_4}(\cdot)$ in (3.45), the approximation accuracy is 60 times better compared to the linear model. The cost to be paid is the increased model complexity, which would then consist of 30 regions.



(a) Evolution of c_A.

(b) Reactor temperature.

(c) Jacket temperature.

(d) Randomly varying coolant flowrate.

Figure 3.9: Simulation results for the CSTR. Red line: nonlinear model (3.43), blue dashed line: PWA model (3.46), black dotted line: linear approximation.

3.9 Summary

In this chapter we proposed an optimisation-based approach to derive PWA approximation of nonlinear systems whose vector field is an a-priori known function of multiple variables. We showed that, under a certain assumption, the problem boils down to solving a series of one-dimensional problem. Subsequently, by utilising this basic one-dimensional building block we extended our procedure to multidimensional separable functions. Finally, we showed that by means of proper substitutions one can transform an arbitrary non-separable function into a separable one, hence allowing implementation of the underlying approximation procedure. In the most trivial case we assumed that the analytic expression of the approximated non-linear function is given. To overcome the difficulty stemmed from the absence of the analytical formula, we proposed to use an efficient two-stage optimisation-based technique to derive PWA approximations of *static* nonlinearities obtained from measured data. The first part of the procedure is focused on finding the best fit of measured data by a pre-specified set of basis functions. The result of this stage is an analytical formula of the fitting function which is subsequently used as an input to the second step. Once we have the analytical expression we can easily apply our procedure to obtain the final approximation. We also discussed the algorithmic and software implementation of the underlying approximation procedure. Specifically, we introduced a new software tool which is capable of exporting the obtained optimal PWA approximations into the HYSDEL language. This brings two crucial advantages. First, the HYSDEL compiler can be used to convert the PWA approximation into a mathematical form, which is then suitable e.g. for control design. Second, since the exported approximation is described in a human-readable format, it can be further fine-tuned by hand. We concluded this chapter by illustrating the approximation procedure involving a model of a highly non-linear chemical reactor.

Part III

Complexity Reduction in Explicit Model Predictive Control

Chapter 4

Explicit Model Predictive Control

MPC is a control strategy where based on the measurements $x(t)$ of plant's states at time t, a mathematical model of the plant is used to predict the evolution of the plant up to time $t+N$. Here, N is called the *prediction horizon*. A sequence of future control inputs is then calculated by optimizing the predicted plant behavior while taking constraints on states and inputs into account. MPC is usually implemented in the so-called receding horizon (RH) fashion. In this setup only the first element of the optimal control sequence is actually implemented to the plant and the rest is discarded. This repetitive optimization is then repeated every time new state measurements become available. This repetitive optimization is used to introduce feedback into the control scheme such that the effects of unpredicted disturbances can be mitigated.

As shown in Bemporad et al. (2002b) the effort of implementing MPC in the Receding Horizon fashion (RHMPC) can be substantially reduced by pre-computing the optimal control action for all possible initial conditions as a function κ. For a large class of MPC problems, such a function can be shown to take a form of (PWA) function, which is composed of a set of polytopic regions and the associated affine feedback expressions. The main benefit is that obtaining the optimal control input at each sampling instance reduces to a mere function evaluation, which can be performed efficiently even on simple control devices in a matter of mili- and microseconds.

On the other hand, to achieve such a simple and fast implementation, all pre-computed data have to be stored in the memory of the target control hardware. Although this aspect is often neglected in the literature, in fact it plays a prominent role when implementing explicit MPC solutions on devices with low available memory storage. Typical examples include programmable logic controllers (PLCs) and embedded microchips, which are one of the most frequently used types of industrial control platforms. Such devices usually

only provide 2-8 kilobytes of memory capacity, a figure which represents a significant challenge in explicit MPC. Needless to say, unless all pre-computed data can be fit into memory, the controller cannot be implemented in practice. Therefore it is of imminent importance to keep the memory footprint $\mathcal{S}(\kappa)$ on an acceptable level.

The rest of this chapter is organised as follows. In Section 4.1 we briefly characterise explicit model predictive control. This description is followed by the problem statement in Section 4.2. Next, in Section 4.3 we provide a comprehensive literature overview regarding complexity reduction of explicit MPC solutions. Section 4.4 introduces our three-layer compression technique, by means of one can significantly reduce the memory requirements of explicit predictive controllers. The chapter ends with efficiency evaluation of our proposed methodology on randomly generated feedback laws, which is followed by a summary, where the main advantages and drawbacks of our proposed methodology will be discussed Material in this chapter is based on our results published in Szűcs et al. (2011b).

4.1 Properties of Explicit Model Predictive Control

We consider the class of constrained, discrete-time, linear time-invariant systems

$$x^+ = \Gamma x + \Xi u,\ x \in \mathcal{X},\ u \in \mathcal{U}, \tag{4.1}$$

where $x \in \mathbb{R}^{n_x}$ is the state vector, x^+ is the successor state, $u \in \mathbb{R}^{n_u}$ is the vector of control inputs, and $\mathcal{X} \subset \mathbb{R}^{n_x}$, $\mathcal{U} \subset \mathbb{R}^{n_u}$ are given polytopic sets. For system (4.1) we define the constrained finite-time optimal control problem:

$$\min_{U_N} \quad \sum_{k=0}^{N-1} \|Q_x x_{k+1}\|_p + \|Q_u u_k\|_p \tag{4.2a}$$

$$\text{s.t.} \quad x_{k+1} = \Gamma x_k + \Xi u_k,\ x_{k+1} \in \mathcal{X},\ u_k \in \mathcal{U} \tag{4.2b}$$

where x_k and u_k denote, respectively, the state and input predictions at time instance k, initialized by the measurements of the current state x_0. The prediction is carried out over a finite prediction horizon N. The explicit representation of the receding horizon MPC feedback $u^* = [I\ 0\ \cdots\ 0]U_N^*$ can be found as a PWA function of the initial condition x by solving (4.2) as a parametric program:

Theorem 4.1 (Bemporad et al. (2002b)) *The RHMPC feedback u^* for problem (4.2) with $p \in \{1, 2, \infty\}$ is given by*

$$u^* = \kappa(x) := \begin{cases} F_1 x + G_1 & \text{if } x \in \mathcal{R}_1 \\ \vdots & \\ F_R x + G_R & \text{if } x \in \mathcal{R}_R, \end{cases} \tag{4.3}$$

where:

- $\kappa : \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ *is a continuous PWA function;*
- $\mathcal{R}_i = \{x \mid H_i x \leq K_i\}$ *are polytopes with* $H_i \in \mathbb{R}^{c_i \times n_x}$, $K_i \in \mathbb{R}^{c_i}$, $i = 1, \ldots, R$;
- *the set of feasible initial conditions* $\Omega := \{x \mid \exists u_0, \ldots, u_{N-1}$ *s.t.* (4.2b) *holds*$\}$ *is a convex polytope;*
- $\{\mathcal{R}_i\}_{i=1}^{R}$ *is a partition of* Ω, *i.e.* $\cup_i \mathcal{R}_i = \Omega$ *and* $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$ *for all* $i \neq j$.

■

The advantage of such an explicit representation is obvious: obtaining the optimal control action for a given x reduces to a mere evaluation of the function κ, which is a two-stage process. In the first step, index i of the region which contains the state measurements is to be identified. This problem is referred to as the point location problem (Snoeyink 1997). Then, in the second step, the optimal control action is computed by evaluating $u^* = F_i x + G_i$. The point location problem can be solved e.g. by traversing the regions sequentially according to Algorithm 4.1.1 (its output is $\emptyset$ if $x \notin \cup_i \mathcal{R}_i$, in which case there is no feasible u which would guarantee satisfaction of constraints in (4.2b)).

Algorithm 4.1.1 Point location

```
for i = 1, ..., R do
    if H_i x ≤ K_i then
        return i
    end if
end for
```

4.2 Problem Definition

The crucial downside of the explicit MPC approach, however, is that the number of regions tends to be large, often above the limits of typical control hardware implementation platforms. Specifically, the amount of memory needed to execute Algorithm 4.1.1 on-line at each sampling instant, expressed as the number of floating-point numbers, is

$$\mathcal{S}(\mathcal{R}_i) = \sum_{i=1}^{R} c_i(n_x + 1), \tag{4.4}$$

where R is the number of regions and c_i is the number of defining half-spaces of the i-th region. Clearly, as R increases, and as the regions become more complex (i.e. with growing

c_i), the memory footprint of κ can easily exceed the provided memory capacity. Therefore, when targeting implementation devices with low memory storage, it is important to devise a more memory-efficient representation of the feedback law κ. In order to reduce the memory footprint of an arbitrary explicit model predictive controller we propose to apply a three-stage compression technique. In the first stage we obtain a set of unique half-spaces, representing the polytopic regions. Then, the expressions obtained from the previous stage are denoted by an integer subscript, hence allowing to represent the given controller by less data, since storing of an integer value requires only 2 bytes compared to floating-point numbers, which consume 4 or even more 8 bytes, depending on the processor architecture. In the final stage we exploit the concept of Huffman encoding, by means we acquire an efficient bit representation of the integer indexes, according to the frequency of their occurrences.

4.3 Overview of Methods for Complexity Reduction in Explicit Model Predictive Control

Complexity of the resulting eMPC controller can be decreased in several ways. One method is based on relaxation of optimality (Bemporad and Filippi 2003, Jones and Morari 2009, Ulbig et al. 2007), by means simpler, but only suboptimal solution can be achieved. Alternatively, one can supplant the original regions of the explicit solution with simpler objects e.g. hypercubes (Johansen and Grancharova 2003) or simplexes (Grieder et al. 2004, Scibilia et al. 2009) or interpolate the solution only from a small subset of regions (Rossiter and Grieder 2005). Although, all these methods in some cases can lead to a remarkable reduction of complexity, generally they do not guarantee substantial simplification. Moreover, they are characterized by suboptimality and stability issues.

The second main direction deals with simplification of already existing explicit solution and its replacement by a simpler functional dependence. This option involves optimal merging of regions (Geyer et al. 2008), elimination of regions, where the control action is saturated (Kvasnica and Fikar 2010) or elimination of saturated regions by separating functions (Kvasnica et al. 2011b). By these approaches substantial complexity reduction can be attained without loss of generality. Replacement of explicit control laws with smooth functions is also available either by the sum of wavelet curves (Summers et al. 2009), Laguerre polynomials (Valencia-Palomo and Rossiter 2010) or by ordinary multi-dimensional polynomials (Kvasnica et al. 2008; 2011a).

Third direction deals with the fastest evaluation of explicit solutions for a given value of initial condition. Standardly, this task is realized by sequential searching of all regions, which leads to linear complexity of this implementation stage. Number of operations,

required to perform such a policy can be decreased by creating appropriate search trees (Tøndel et al. 2003), where the complexity of the searching is only logarithmic in number of regions. Acceleration can be aimed by exploiting the convexity of the objective function (Baotic et al. 2008) or the continuity of the control law (Wen et al. 2009).

4.4 Main Results

In this section we show how to represent regions $\mathcal{R}_i$ more efficiently by exploiting their geometric properties. Each of the proposed three layers can be viewed at as a "compression" mechanism. Needless to say, additional computational effort needs then to be performed on-line to "decompress" the data. We provide quantification of such an additional effort as a function of the problem size. Decompression is performed on-the-fly on a region-by-region basis.

Only the polytopic nature of regions $\mathcal{R}_i$ is exploited by the proposed complexity reduction procedure. Continuity of κ and convexity of the feasible set Ω are not required. Therefore the approach is applicable to generic PWA function κ defined over polytopes. The scope of this work therefore extends to scenarios where tracking of a non-zero reference is achieved by a suitable augmentation of the state vector, or where linear hybrid systems are used as prediction models. For the same reason the procedure can be applied to post-process RHMPC feedback laws generated by other complexity reduction schemes, e.g. those reviewed in Kvasnica (2009).

To quantify achievable reduction in memory, we will assume that double-precision floating point numbers consume 8 bytes, while integers can be represented by 2 bytes. Each individual mathematical operation on a float or on an integer will be denoted as one FLOP.

4.4.1 Complexity Reduction via Affine Transformations

First we show how to represent some regions using less data by exploiting geometric similarities of such polytopes. We remind that the memory footprint of a region $\mathcal{R}_j = \{x \mid H_j x \leq K_j\}$ with $H_j \in \mathbb{R}^{c_j \times n_x}$ and $K_j \in \mathbb{R}^{c_j}$ is $c_j(n_x + 1)$ real numbers with $c_j \geq n_x + 1$. Here, we look for affine transformations $A_{i,j}x + b_{i,j}$ such that

$$\mathcal{R}_i = \{A_{i,j}x + b_{i,j} \mid x \in \mathcal{R}_j\}. \tag{4.5}$$

If there exist $A_{i,j} \in \mathbb{R}^{n_x \times n_x}$ and $b_{i,j} \in \mathbb{R}^{n_x}$ which map $\mathcal{R}_j$ onto $\mathcal{R}_i$, then the memory footprint of κ is reduced as follows: for each i, j for which the mapping exists, the half-space representation of the j-th region (i.e. matrices H_j, K_j with variable number of rows

c_j) can be replaced by matrices $A_{i,j}$, $b_{i,j}$ with fixed number of rows n_x. Then, once $x \in \mathcal{R}_j$ is to be verified in Step 2 of Alg. 4.1.1, it suffices to check whether $A_{i,j}x + b_{i,j} \in \mathcal{R}_i$, i.e.

$$x \in \mathcal{R}_j \;\Leftrightarrow\; A_{i,j}x + b_{i,j} \in \mathcal{R}_i. \tag{4.6}$$

It follows that memory footprint of region $\mathcal{R}_j$ is reduced by $(c_j - n_x)(n_x + 1)$ real numbers by only storing $A_{i,j}, b_{i,j}$ instead of H_j, K_j. Since $c_j \gg n_x + 1$ in practice, a significant reduction can be achieved.

Definition 4.1 *Let the polytopic partition $\{\mathcal{R}_i\}_{i=1}^R$ be given. The index set $\mathcal{I}_G \subseteq \{1, \ldots, R\}$ is called the index set of* generating regions *of the partition if for each $j \notin \mathcal{I}_G$ there exists an $i \in \mathcal{I}_G$ and the associated affine map $A_{i,j}x + b_{i,j}$ such that (4.5) holds.*

Lemma 4.1 *Let i, j be given and let $\mathcal{V}_i = [v_{i,1}, \ldots, v_{i,n_v}]$ and $\mathcal{V}_j = [v_{j,1}, \ldots, v_{j,n_v}]$ denote, respectively, the extremal vertices of $\mathcal{R}_i$ and $\mathcal{R}_j$. Then an affine transformation which guarantees (4.6) exists if there exist $A_{i,j} \in \mathbb{R}^{n_x \times n_x}$, $b_{i,j} \in \mathbb{R}^{n_x}$, and a binary permutation matrix $P \in \{0, 1\}^{n_v \times n_v}$ with $\sum_{m=1}^{n_v} P_{m,k} = 1$, $\forall k$, $\sum_{m=1}^{n_v} P_{k,m} = 1$, $\forall k$ such that*

$$\begin{bmatrix} A_{i,j} & b_{i,j} \end{bmatrix} \begin{bmatrix} \mathcal{V}_j \\ 1 \end{bmatrix} = \mathcal{V}_i P. \tag{4.7}$$

Proof of Lemma 4.1 *Since $\mathcal{V}_i$ and $\mathcal{V}_j$ are extremal vertices, (4.6) is equivalent to*

$$x \in \mathrm{convh}(\mathcal{V}_j) \;\Leftrightarrow\; (A_{i,j}x + b_{i,j}) \in \mathrm{convh}(\mathcal{V}_i) \tag{4.8}$$

where $\mathrm{convh}(\cdot)$ *denotes convex hull. By convexity of $\mathcal{R}_i$ and $\mathcal{R}_j$, the affine transformation exists if for each $s \in \{1, \ldots, n_v\}$, there exists a $t \in \{1, \ldots, n_v\}$ such that*

$$A_{i,j}v_{j,s} + b_{i,j} = v_{i,t}, \tag{4.9}$$

i.e. when there exists an appropriate permutation of vertices $\mathcal{V}_i$ such that (4.9) holds $\forall s$. But this is equivalent to existence of a binary permutation matrix P whose rows and columns sum up to 1. *Hence (4.7) follows.*

Problem (4.7) is a feasibility problem with real variables $A_{i,j}$, $b_{i,j}$ and the binary variables P, which can be solved by off-the-shelf software, like GLPK (Makhorin 2001) or CPLEX (ILOG, Inc. 2003). One can also look for a numerically scaled solution by, in addition, minimizing $\|A_{i,j}\|_1 + \|b_{i,j}\|_1$. Regions $\mathcal{R}_i$ are processed by Algorithm 4.4.1, which requires solving, at most, $1/2 n_x(n_x - 1)$ MIP problems (4.7). In practice, it will be less, since only regions with the same number of vertices need to be processed. The algorithm returns an auxiliary array $\mathcal{J}$ which denotes feasible i–j combinations. If $\mathcal{J}_j \neq \emptyset$

Algorithm 4.4.1

Initialize $\mathcal{J}_j = \emptyset$, $\mathcal{A}_j = \emptyset$, $\mathcal{B}_j = \emptyset$, $j = 2, \ldots, R$
for $i = 1, \ldots, R-1$ **do**
  for $j = i+1, \ldots, R$ **do**
    if $\mathcal{J}_j = \emptyset$ and (4.7) is feasible **then**
      $\mathcal{A}_j \leftarrow A_{i,j}$, $\mathcal{B}_j \leftarrow b_{i,j}$, $\mathcal{J}_j \leftarrow i$
    end if
  end for
end for

for some j, then $\mathcal{J}_j$ points to its associated generating region. If $\mathcal{J}_j = \emptyset$, then $\mathcal{R}_j$ is a generating region on its own, i.e. $\mathcal{I}_G = \{j \mid \mathcal{J}_j = \emptyset\}$.

Given the arrays of affine transformations $\mathcal{A}$ and $\mathcal{B}$, the point-location task can be implemented by Algorithm 4.4.2. The size of its input arguments is

$$\mathcal{S}(\mathcal{R}_i) = \sum_{i \in \mathcal{I}_G} c_i(n_x + 1) + \sum_{i \notin \mathcal{I}_G} n_x(n_x + 1), \tag{4.10}$$

a reduction by $\sum_{i \notin \mathcal{I}_G}(c_i - n_x)(n_x + 1)$ floating point numbers compared to the standard approach, cf. (4.4). The memory saving is hence proportional to the number of non-generating regions. The algorithm loops through regions sequentially. If a generating region is encountered, $x \in \mathcal{R}_i$ is checked directly. Otherwise, (4.6) is exploited and $\mathcal{A}_j x + \mathcal{B}_j \in \mathcal{R}_i$ is checked instead. Saving in terms of memory is traded for an increase in execution time. Here, compared to Algorithm 4.1.1, one needs to evaluate the affine transformations whenever a non-generating region is encountered, which requires $\sum_{i \notin \mathcal{I}_G}(2n_x^2)$ FLOPs in the worst case.

Algorithm 4.4.2

for $j = 1, \ldots, R$ **do**
  if $j \in \mathcal{J}$ **then**
    $i \leftarrow \mathcal{J}_j$, $x \leftarrow \mathcal{A}_i x + \mathcal{B}_i$
  else
    $i \leftarrow j$
  end if
  if $H_i x \leq K_i$ **then**
    return j
  end if
end for

Example 4.1 *Consider a double integrator sampled at 1 second, given by the following state-space representation:*

$$x^+ = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} u, \tag{4.11}$$

where the states and inputs are constrained, respectively, by $|x_i| \leq 5$, $i = 1, 2$, *and* $|u| \leq 1$. *With the choice of* $p = 1$, $Q_x = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$, $Q_u = 1$, *and* $N = 10$ *in* (4.2), *the explicit RHMPC feedback law consists of* 230 *regions, shown in Figure 4.1. Storing all regions would require* 2466 *floating point numbers, or* 19 *kilobytes. Algorithm 4.4.1 has found feasible affine transformations for* 198 *regions, representation of which can be simplified by only storing* $A_{i,j}$ *and* $b_{i,j}$. *Remaining* 32 *generating regions need to be represented using the full data, i.e. by matrices* H_i, K_i. *Here, the* 198 *affine transformations contribute by* 1188 *numbers, while the* 32 *regions require* 402 *floats. It follows that the total required memory is decreased from* 19 *to* 12 *kilobytes. The worst-case number of FLOPs*[1] *needed to perform point location via Algorithm 4.4.2 is* 6143 *compared to* 4110 *operations for Algorithm 4.1.1.*

4.4.2 Data De-Duplication

Instead of having to store all data (i.e. all matrices H_i and K_i), one can use de-duplication to first identify unique rows of $H = \left[\begin{smallmatrix} H_1^T & \cdots & H_R^T \end{smallmatrix}\right]^T \in \mathbb{R}^{m \times n_x}$ and $K = \left[\begin{smallmatrix} K_1^T & \cdots & K_R^T \end{smallmatrix}\right]^T \in \mathbb{R}^m$ with $m = \sum_i^R c_i$. Denote the unique rows by $\mathcal{H}$ and $\mathcal{K}$. If cardinality of $\mathcal{H}$ ($\mathcal{K}$) is smaller than number of rows in H (K), then the amount of memory can be significantly decreased by storing, for each region, only the pointers to $\mathcal{H}$ and $\mathcal{K}$. The saving is twofold. First, memory size of a pointer is smaller than for a floating point number. Second, since a single pointer is assigned to each row of $\mathcal{H}$ (which is n_x dimensional), the amount of memory is decreased n_x times for each entry.

As an example, consider three regions given in their respective half-space representation by

$$H_1 = \begin{bmatrix} 0 & 1 \\ 1 & -0.5 \\ 0 & -1 \\ -1 & 0.5 \end{bmatrix}, \; H_2 = \begin{bmatrix} 0 & 1 \\ 1 & -0.5 \\ -1 & 0.5 \\ 0 & -1 \end{bmatrix}, \; H_3 = \begin{bmatrix} -1 & 0.5 \\ 0 & -1 \\ 0 & 1 \\ 1 & -0.5 \end{bmatrix},$$
$$K_1 = \begin{bmatrix} 2.4 \\ 3.1 \\ -1.5 \\ 0 \end{bmatrix}, \; K_2 = \begin{bmatrix} 2.4 \\ 5.0 \\ -3.1 \\ 0 \end{bmatrix}, \; K_3 = \begin{bmatrix} 0 \\ 0 \\ 1.5 \\ 3.1 \end{bmatrix}.$$

The sets of unique rows are

$$\mathcal{H} = \{\left[\begin{smallmatrix} 0 & 1 \end{smallmatrix}\right], \left[\begin{smallmatrix} 1 & -0.5 \end{smallmatrix}\right], \left[\begin{smallmatrix} 0 & -1 \end{smallmatrix}\right], \left[\begin{smallmatrix} -1 & 0.5 \end{smallmatrix}\right]\},$$
$$\mathcal{K} = \{-3.1, -1.5, 0, 1.5, 2.4, 3.1, 5.0\}.$$

[1] It is worth noting that even slow CPUs typically found in industrial control hardware are able to perform tens of millions of FLOPs per second. With the reported computational figures the control algorithm could therefore be executed at the sampling range of hundreds of kilohertz.

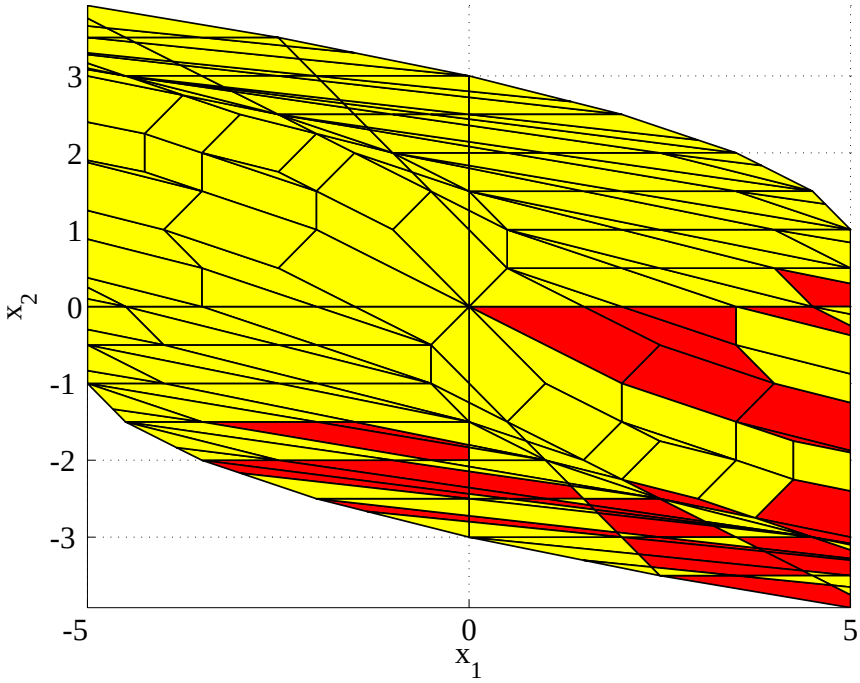


Figure 4.1: Regions of the explicit MPC solution for Example 4.1. Each of the 198 yellow regions can be obtained by applying a suitable affine transformation (4.6) to one of the 32 generating regions, shown in red.

The corresponding (unsigned) index set representation of the polytopic regions is then

$$\begin{aligned}\mathcal{I}_{H_1} &= \{1,\,2,\,3,\,4\},\ \mathcal{I}_{K_1} = \{5,\,6,\,2,\,3\},\\ \mathcal{I}_{H_2} &= \{1,\,2,\,4,\,3\},\ \mathcal{I}_{K_2} = \{5,\,7,\,1,\,3\},\\ \mathcal{I}_{H_3} &= \{4,\,3,\,1,\,2\},\ \mathcal{I}_{K_3} = \{3,\,3,\,4,\,6\},\end{aligned}$$

where each element of $\mathcal{I}_H$ and $\mathcal{I}_K$ points to the corresponding entry in $\mathcal{H}$ and $\mathcal{K}$.

Cardinality of $\mathcal{H}$ and $\mathcal{K}$, and hence the required storage space, can be further reduced by eliminating entries which are negations of others, i.e.

$$\mathcal{H} = \{[\,0\ 1\,],\,[\,1\ -0.5\,]\},\ \mathcal{K} = \{-3.1,\,-1.5,\,0,\,2.4,\,5.0\}.$$

Then the (signed) index set representation becomes

$$\begin{aligned}\mathcal{I}_{H_1} &= \{1,\,2,\,-1,\,-2\},\ \mathcal{I}_{K_1} = \{4,\,-1,\,2,\,3\},\\ \mathcal{I}_{H_2} &= \{1,\,2,\,-2,\,-1\},\ \mathcal{I}_{K_2} = \{4,\,5,\,1,\,3\},\\ \mathcal{I}_{H_3} &= \{-2,\,-1,\,1,\,2\},\ \mathcal{I}_{K_3} = \{3,\,3,\,-2,\,-1\},\end{aligned} \tag{4.12}$$

In this simple example, memory footprint of regions $\mathcal{R}_i$ was reduced from 36 floating point numbers representing matrices H_i, K_i to 9 floats for $\mathcal{H}$, $\mathcal{K}$, and 24 integer pointers. Assuming that one float is represented by 8 bytes and an integer by 2 bytes, de-duplication reduces required memory from 288 bytes to 120 bytes.

Remark 4.1 *Needless to say, the same de-duplication approach can be used to reduce memory footprint of affine transformations in* (4.6).

Algorithms 4.1.1 and 4.4.2 can be easily accommodated to exploit the signed index set representation. Whenever $H_i x \leq K_i$ needs to be checked, one constructs, on-the-fly, matrices H_i and K_i by $H_i = \{\text{sign}(j)\mathcal{H}_{|j|} \mid j \in \mathcal{I}_{H_i}\}$ and $K_i = \{\text{sign}(j)\mathcal{K}_{|j|} \mid j \in \mathcal{I}_{K_i}\}$. This involves negating the corresponding rows, depending on the sign of the index. Therefore execution of Algorithms 4.1.1 and 4.4.2 requires, at most, $\sum_{i=1}^{R} 2c_i$ additional FLOPs.

Example 4.2 *We revisit Example 4.1 and remind that the full set of* 230 *regions can be equivalently represented by* 32 *generating regions and by* 198 *associated affine transformations* (4.5). *The generating regions are described by* 134 *half-spaces, which require* $134(n_x+1)$ *floating point numbers* ($n_x = 2$ *in this example). Here, the sets of rows unique under unity scaling, i.e.* $\mathcal{H}$ *and* $\mathcal{K}$, *only contains* 17 *and* 47 *entries, respectively, which is equivalent to* $17n_x + 47$ *floating point numbers. The corresponding index sets* $\mathcal{I}_{H_i}$ *and* $\mathcal{I}_{K_i}$ *contribute by* 2×134 *integers. After de-duplication is applied to* $\mathcal{A}_j$ *and* $\mathcal{B}_j$ *as well, the*

Table 4.1: Frequencies of integers to encode.

Integer	−2	−1	1	2	3	4	5
Frequency	1	2	1	1	4	2	1

total memory footprint is reduced from 12 *kilobytes reported in Example 4.1 to just* 7.5 *kilobytes. This comes at the expense of performing additional* 1644 *FLOPs to reconstruct the regions on-the-fly using index set representations.*

4.4.3 Compression of Index Set Representations

Given are index set representations $\mathcal{I}_H = \cup_i \mathcal{I}_{H_i}$ and $\mathcal{I}_K = \cup_i \mathcal{I}_{K_i}$, whose entries point to corresponding rows in the set of unique elements $\mathcal{H}$ and $\mathcal{K}$. In traditional implementation, each element of $\mathcal{I}_H$ and $\mathcal{I}_K$ would need to be represented as a (signed) integer, i.e. by 16 bits (provided that cardinality of $\mathcal{H}$ and $\mathcal{K}$ does not exceed 2^{16}). A more efficient representation can be obtained by using a *prefix-free variable-length encoding* where bit-wise codewords are assigned to each element of the index sets. Length of a codeword is inverse-proportional to its abundance, such that size of $\mathcal{I}_K$ and $\mathcal{I}_H$ is compressed as much as possible.

Proposition 4.2 (Dasgupta et al. (2006)) *Given an index set $\mathcal{I}$ and an array of frequencies $\mathcal{F}$, Algorithm 4.4.3 generates an optimal coding tree $\mathcal{T}(\mathcal{I})$ as a full binary tree where the symbols to encode are at the leaves, and where each codeword is generated by a path from root to leaf, interpreting left traversal as* 0 *and right as* 1.

As an illustration, consider the index sets in (4.12) and let $\mathcal{I}_K = \mathcal{I}_{K_1} \cup \mathcal{I}_{K_2} \cup \mathcal{I}_{K_3}$. Then the integers to encode appear with frequencies reported in Table 4.1. The corresponding Huffman tree is shown in Figure 4.2. Here, the optimal codewords are $\mathcal{C}(\mathcal{I}_K) = \{[-2 : 111], [-1 : 001], [1 : 110], [2 : 101], [3 : 01], [4 : 000], [5 : 100]\}$. It is easy to verify that such an encoding is prefix-free, i.e. that no codeword is a prefix of another codeword. Moreover, the most abundant integer 3 is encoded using fewest number of bits as to minimize the total length of binary representation of $\mathcal{I}_K$. Hence, instead of storing $\mathcal{I}_K$ as an array of 12 integers (i.e., 24 bytes), it suffices to store the tree (7 integers or 14 bytes) and 12 codewords of a total size 32 bits, or 4 bytes.

Size of the tree is proportional to the number of unique elements of the encoded set of integers $\mathcal{I}$. Decoding of a particular sequence of bits boils down to traversing the tree until a leaf is reached, whereupon the tree returns to its root. Decompression effort is therefore proportional, in the worst case, to the length m of the longest codeword. In

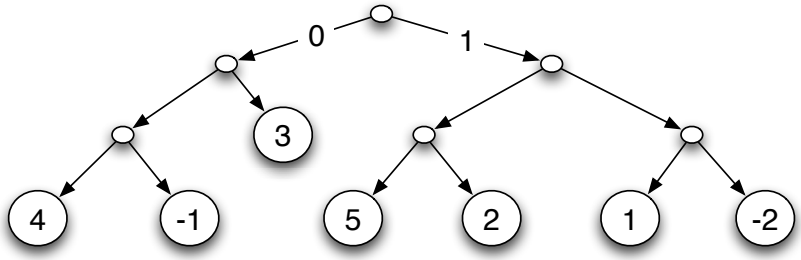


Figure 4.2: Huffman tree for
$\mathcal{I}_K = \{4, -1, 2, 3, 4, 5, 1, 3, 3, 3, -2, -1\}$.

Algorithm 4.4.3 Huffman encoding Dasgupta et al. (2006)

Let $\mathcal{Q}$ be a priority queue, ordered by positive frequencies $\mathcal{F} = [f_1, \ldots, f_n]$
for $k = n+1, \ldots, 2n-1$ **do**
  $i \leftarrow$ `deletemin`$(\mathcal{Q})$, $j \leftarrow$ `deletemin`$(\mathcal{Q})$
  Create a node k with children i, j
  $\mathcal{F}_k \leftarrow \mathcal{F}_i + \mathcal{F}_j$
  `insert`$(\mathcal{Q}, k)$
end for

total, $\sum_{i=1}^{R} 2c_i m$ operations are required to reconstruct regions $\mathcal{R}_i$ on-the-fly from their respective encoded index set representations.

Remark 4.2 *Traversing the tree only requires performing bit-wise operations, which are much cheaper than multiplications or additions on floating point numbers. Therefore a mere increase in FLOPs by a factor of n does not necessarily mean that evaluation speed would drop n times. In practice, it will be less.*

Example 4.3 *We continue with Example 4.2 where it was shown that* 134 *signed integers* $\mathcal{I}_H$ *pointing to one of the* 17 *unique rows of* $\mathcal{H}$*, and* 134 *signed integers* $\mathcal{I}_K$ *for the* 47 *unique elements of* $\mathcal{K}$ *are required for the index set representation of generating regions. The trees* $\mathcal{T}(\mathcal{I}_H)$ *and* $\mathcal{T}(\mathcal{I}_K)$ *were build by Algorithm 4.4.3 in* 0.05 *seconds. The trees had* 26 *and* 63 *leave nodes, respectively. Each element of* $\mathcal{I}_H$*,* $\mathcal{I}_K$ *was encoded as a prefix-free sequence of bits. For* $\mathcal{I}_H$*, the minimal codeword length was* 3*, the maximal was* 6*. For* $\mathcal{I}_K$*, the minimal and maximal code lengths were* 4 *and* 7*, respectively. It follows that the index sets* $\mathcal{I}_H$ *and* $\mathcal{I}_K$*, which originally required* 2×134 *integers, can be equivalently represented by the two trees (which need* 89 *integers) and* 2×134 *bit sequences, which in total attribute by* 527 *bits, or* 66 *bytes. Therefore memory footprint of the index set representations is reduced from* 536 *bytes to* 244 *bytes. Decompression of the bit codewords in a suitable*

Table 4.2: Average accumulated compression factors.

n_x	Sec. 4.4.1	Sec. 4.4.2	Sec. 4.4.3
2	1.5	4.3	8.2
3	1.3	5.9	13.7
4	1.7	8.1	24.6
5	1.5	10.4	43.2

modification of Algorithm 4.4.2 would require additional 3570 *FLOPs, in the worst case.*

4.5 Efficiency Evaluation

To asses efficiency of the proposed three-layer procedure on generic data, we have analyzed randomly-generated explicit RHMPC feedback laws for dimensions $2 \leq n_x \leq 5$. For each dimension, 20 random RHMPC feedback laws were generated by the MPT Toolbox (Kvasnica et al. 2004). Each controller was then processed by applying, consecutively, the similarity transformation of Section 4.4.1, then de-duplication of Section 4.4.2, followed by data compression of Section 4.4.3.

For various state dimensions, Figure 4.3 shows achieved memory reduction factors, i.e. the ratios between memory size of the original solution and the corresponding compression layer. Note that the figures show accumulated data, i.e. improvement of a particular layer upon a previous one. The unity basis corresponds to size of the original, uncompressed, RHMPC solution. As can be observed, reduction of memory size by a factor of 20 is not unusual. The average values are also summarized in Table 4.2. As expected, the compression factors increase with growing number of states. This trend is mostly notable for the de-duplication and compression methods.

Results in Figure 4.4 then quantify the factor by which the number of floating point operations increases in order to "decompress" a particular layer, with Algorithm 4.1.1 being the basis. However, as noted in Remark 4.2, this factor is not directly proportional to a slowdown in evaluation speed when the Huffman encoding layer is concerned. Although the evaluation effort is substantially increased, it is always out-weighted by a more substantial reduction in terms of memory. With growing problem dimension and number of regions, complexity of Algorithm 4.1.1 naturally increases. It is due to this fact that the relative factors in Figure 4.4 actually tend to improve when n_x is enlarged.

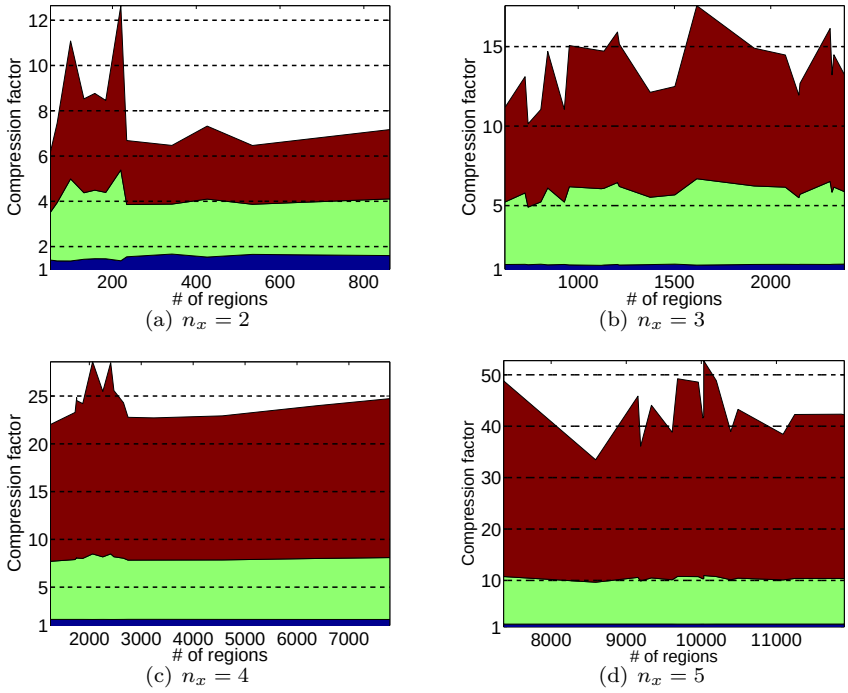


Figure 4.3: Accumulated reduction in memory storage achieved by individual layers (blue is for the similarity transformation, green for de-duplication, and red for compression). Note that individual figures have different scales on the y axis.

4.6 Summary

In this work, instead of decreasing $\mathcal{S}(\kappa)$ by reducing the number of regions, we look for a memory efficient representation of κ which requires less data. The procedure consists of three layers. The first one determines a subset of regions which can be obtained by applying affine transformation of the remaining regions. We show how to formulate the search for such a mapping by solving a mixed-integer problem, which is done off-line. If the transformation exists, the corresponding regions can then be represented using less data. The second layer can either be applied on top of the first one, or independently. Here, memory is saved by identifying positive and negative duplicities in half-space representation of several polytopic regions. The duplicate occurrences are then represented as mere integer pointers to the unique set of data. Compared to the first layer, the additional computation to be performed on-line is much smaller. Finally, in the last layer we propose to compress the integer pointers by Huffman encoding (Knuth 1985). Here, variable-length bit codewords are assigned to each integer, depending on its frequency of abundance. Main benefit of the proposed strategies is that they can be applied on top of

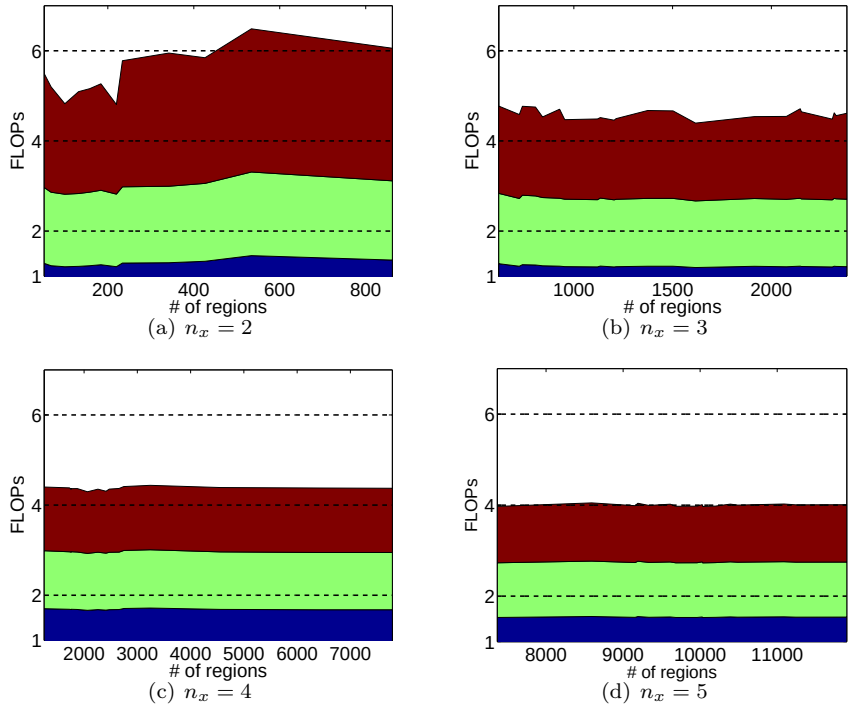


Figure 4.4: Accumulated increase in on-line computation needed to implement a particular layer (blue is for the similarity transformation, green for de-duplication, and red for compression).

all aforementioned complexity reduction schemes. Saving in terms of memory is achieved at the price of an increase of the implementation effort performed on-line. Therefore the approach is mainly suited for situations where the implementation device poses enough computational power, but has severe memory limitations.

Part IV

Fast Model Predictive Control

Chapter 5

Operator Splitting Methods in Control

In the previous chapter we have introduced explicit model predictive control as a method by means of one can solve MPC problems very quickly. In other words for systems with modest size, i.e. up to 4 state variables the solution can be obtained within microseconds. Besides this method is suitable for small-scale systems, we have also stated that this approach provides optimal solution only for fixed parameters. In order to eliminate the above mentioned downsides, in this chapter we will introduce a novel computational framework which can solve the underlying MPC problem very quickly. Furthermore, these algorithms do not posses any restrictions on the dimension of the corresponding problem. These methods, in general are denoted as alternating direction methods.

Alternating direction methods have attracted a lot of attention in many fields, e.g. signal processing, machine learning, and computer vision, where people need to address large-scale optimisation problems with non-differentiable objectives, and have already been shown that they are suitable for solving such problems. Recently it has been found out that MPC problems can often be transformed into a convex optimisation problem with a differentiable and a non-differentiable indicator function, thus enabling application of the aforementioned alternating methods.

Motivated by the interconnection between MPC and convex optimisation problems, in this section we describe several algorithms to solve convex optimal control problems quickly. The algorithms we present rely on an operator splitting technique. Such a technique breaks the problem into two parts, a quadratic optimal control problem (which can be solved very efficiently) and a set of single period optimisation problems (which can be solved in parallel, often analytically). An iteration that alternates these two steps then converges to a solution. We demonstrate that the proposed algorithms can solve optimal control problems to an acceptable accuracy very rapidly, indicating that it is suitable

for use in e.g. high-frequency control applications. Another advantage of our methods is that in many cases, after some off-line pre-computation, the algorithm requires no division operations. In these cases it can be implemented in fixed-point arithmetic, for example on a field-programmable gate array (FPGA) for high-speed embedded control.

The rest of the chapter is organised as follows. In Secion 5.1 we describe the main computational techniques and fields related to fast model predictive control, which is followed by the problem statement in Section 5.2. Next, in Section 5.3 we characterise a set of algorithms serving to solve convex optimisation problems in a fast and efficient manner. Subsequenly, we discuss several possibilities of the improvement in a convergence rate in Section 5.4. The chapter concludes by a case study presented in Section 5.5. Material of this chapter is based on our results published in the accepted paper (Stathopolous et al. 2014).

5.1 Prior and Related work

In this section we give a brief overview of some important prior work in several related areas.

Interior-point methods. A generic interior-point solver which does not exploit the problem structure would scale in complexity with cube of the time-horizon (Betts 2001). If the structure of the underlying problem is exploited, however, the complexity only grows linearly. In Wang and Boyd (2008) the authors developed a custom interior-point method that can solve quadratic optimal control problems with box constraints very rapidly by exploiting problem structure. A similar approach was taken by Rao et al. (2004). For work detailing efficient primal-dual interior-point methods to solve the quadratic programs (QPs) that arise in optimal control see for example Åkerblad and Hansson (2004), Hansson (2000), Hansson and Boyd (1998).

Automatic code generation. Typically creating a custom interior-point solver is a very labor-expensive exercise. In Mattingley and Boyd (2012) the authors describe the automatic generation of high speed custom solvers directly from high level descriptions of the problem. These automatically generated custom solvers are tailored to the problem at hand, providing dramatic speedups over generic solvers.

Explicit MPC. Explicit model predictive control is a technique for solving quadratic optimal control problems with polyhedral constrains (Bemporad et al. 2002a, Tøndel et al. 2001), with all data fixed except of the initial state. In this case the solution is a piecewise

affine function of the initial state. The polyhedra that define the regions, and the associated coefficients in the affine function, can be computed off-line. Solving the problem then reduces to searching in a lookup table, and then evaluating the affine function (which is division free). Due to the exponential growth in the number of regions, explicit MPC can realistically only be applied to system with very modest numbers of states and constraints. For an extension that can handle larger problems by using partial enumeration see Pannocchia et al. (2006). A more thorough description related to basic properties of explicit model predictive control, including but not limited to its mathematical description can be found in chapter 4

Active set methods. Active set methods are a set of techniques for solving QPs that are closely related to the simplex method for linear programming. They rely on identifying the set of constraints that are active at the optimum and then solving a simpler problem using just these constraints. The use of active set methods to solve the QPs that arise in conrol has been explored by Ferreau et al. (2008a).

Fast gradient methods. Fast gradient methods inspired by Nesterov's accelerated first order methods (Nesterov 1983a), have been applied to the optimal control problem (Kögel and Findeisen 2011, Richter et al. 2009; 2010). These techniques typically require only the evaluation of a gradient and a projection at each iteration. Thus, they generally require less computation than, say, interior-point methods, at the expense of high accuracy.

Embedded control. There has been much recent interest in using MPC in an embedded control setting, for example in autonomous or miniature devices. The challenge is to develop algorithms that can solve convex optimisation problems quickly, robustly and within the limitations of on-board chip architectures. Many techniques have been investigated, including interior-point methods, active set methods and others; see e.g. Jerez et al. (2011), Ling et al. (2008), Longo et al. (2011).

Operator splitting. The technique we employ in this thesis relies on the work done on monotone operators and operator splitting methods. The history of operator splitting goes back to the 1950s; ADMM itself was introduced in the mid-1970s by Glowinski and Marrocco (1975a) and Gabay and Mercier (1976a). It was shown in Gabay (1983) that ADMM is a special case of splitting technique known as Douglas-Rachford splitting, and Eckstein and Bertsekas (1992) showed in turn that Douglas-Rachford splitting is a special case of the proximal point algorithm. For convergence results for operator splitting algorithms and example applications see Boyd et al. (2011a) and the references therein. Operator splitting has seen use in many application areas, see, e.g. Annergren et al.

(2012), Tøndel and Johansen, Wahlberg et al. (2012). In Lin et al. (2012) the authors use operator splitting to develop sparse feedback gain matrices for linear-quadratic control problems.

5.2 Problem Formulation

In this work we focus on systems with linear dynamics, giving rise to convex control problems. The purpose of this work is to explore a family of first order methods known as *decomposition schemes* or *operator splitting methods*. In the simplest case, the abstract form of the problem at hand is the minimization of the sum of two convex functions and can be written as

$$\text{minimize} \quad f(x) + g(Ax) \ , \tag{5.1}$$

with variables $x \in \mathbb{R}^n$, where $f : \mathbb{R}^n \to (-\infty, \infty]$ and $g : \mathbb{R}^m \to (-\infty, \infty]$ are proper, lower semi-continuous (lsc) convex functions and $A : \mathbb{R}^n \to \mathbb{R}^m$ is a linear map. A splitting method can be applied to the above problem after rewriting it as

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax = z \ , \end{array} \tag{5.2}$$

by alternatively (or simultaneously) minimizing over f and g. A dual variable update for the equality constraint ensures that the solutions of problems (5.2) and (5.1) are identical. Inequality constraints are already present in the formulation in the form of indicator functions, *i.e.*, a membership function for a set C

$$\delta_C(x) = \begin{cases} 0 & x \in C \\ \infty & \text{otherwise.} \end{cases} \tag{5.3}$$

Although it is established that splitting methods are quite beneficial when applied to large-scale problems (Guler 1992), their potential in solving small to medium scale embedded optimization problems has not been studied so extensively. Our purpose is to study the behavior of such algorithms as solvers of control-related problems of that scale. Our effort focuses on identifying special characteristics of these problems and how they can be exploited by some popular splitting methods. Some of the questions that we attempt to answer are:

1. It is very common in practice that optimal control problems come with a quadratic objective, since in this way stability can be proven for regulation or tracking purposes. What is the best way to exploit this smooth term?

2. Given that a control problem has to be solved repeatedly (*e.g.*, MPC), how can warm-starting affect the speed?

3. Given the structure of the problem at hand, which algorithms will converge more quickly?

We narrow the general formulation to our problems of interest which can, without loss of generality, be written as

$$\begin{aligned} &\text{minimize} && (1/2)z^TQz + c^Tz + \sum_{i=1}^{M} l_i(T_iz + t_i) \\ &\text{subject to} && Az = b \ , \end{aligned} \tag{5.4}$$

with variable $z \in \mathbb{R}^n$, where $Q \in \mathbb{S}^n_+$, $T_i \in \mathbb{R}^{p_i \times n}, c \in \mathbb{R}^{1\times n}, t \in \mathbb{R}^{1\times n}, A \in \mathbb{R}^{m\times n}$, and $b \in \mathbb{R}^{1\times m}$ Finally, parameter M denotes the number of functions l_i The following assumption holds:

Assumption 5.1 *The functions $l_i : \mathbb{R}^{p_i} \to (-\infty, \infty]$ are closed, lsc convex functions.*

Formulation (5.4) is quite general and can describe a wide range of convex optimisation problems. The choice of the quadratic part $(1/2)z^TQz + c^Tz$ and the equality constraints $Az = b$ being represented in an explicit way is motivated by the standard form that control problems take.

For lighter notation, we define $f(z) := \big\{(1/2)z^TQz + c^Tz \mid Az = b\big\}$. We also denote the concatenated vectors and matrices associated with the affine term in the l_i's as $T = (T_1, \ldots, T_M)$ and $t = (t_1, \ldots, t_M)$. Using slack variables $y_i = T_iz + t_i,\ i = 1, \ldots, M$, the *Lagrangian* for (5.4) is written as

$$L = f(z) + \sum_{i=1}^{M} l_i(y_i) + \sum_{i=1}^{M} \langle \lambda_i, -t_i - T_iz + y_i \rangle \ , \tag{5.5}$$

where $\lambda_i \in \mathbb{R}^{p_i}$ are dual variables associated with the equality constraints introduced above. We can recover the optimum by solving

$$(\lambda^\star, z^\star, y^\star) = \operatorname*{argmax}_{\lambda} \operatorname*{argmin}_{z,y} L(\lambda, z, y) \ , \tag{5.6}$$

where $\lambda = (\lambda_1, \ldots, \lambda_M) \in \mathbb{R}^p$, $y = (y_1, \ldots, y_M) \in \mathbb{R}^p$, $p = \sum_{i=1}^M p_i$. For solving problem (5.4) we consider three approaches, namely solving a saddle point problem either on the Lagrangian, the augmented Lagrangian function or a generic saddle-point formulation that involves taking the Legendre-Fenchel dual of the functions $l_i(\cdot)$.

The *augmented Lagrangian* (Boyd et al. 2011b) for problem (5.4) is defined by

$$L_\rho = f(z) + \sum_{i=1}^{M} l_i(y_i) + \sum_{i=1}^{M} \langle \lambda_i, -t_i - T_iz + y_i \rangle + \frac{\rho}{2} \sum_{i=1}^{M} \| - t_i - T_iz + y_i\|^2 \ , \tag{5.7}$$

for $\rho > 0$ and the problem to solve becomes

$$(\lambda^\star, z^\star, y^\star) = \operatorname*{argmax}_{\lambda} \min_{z,y} L_\rho(\lambda, z, y) \ . \tag{5.8}$$

Another option is to apply some partial dualization to the Lagrangian formulation, resulting in a primal-dual equivalent that is easier to solve. Making use of the Legendre-Fenchel conjugate,

$$l_i^\star(\lambda_i) = \sup_z \langle T_i z + t_i, \lambda_i \rangle - l_i(T_i z + t_i) \ , \tag{5.9}$$

, where pair of angle brackets in the above mentioned expression for (5.9) stands for dot product. Functions $l_i(T_i z + t_i)$ can now be expressed as

$$l_i(T_i z + t_i) = \sup_{\lambda_i} \langle T_i z + t_i, \lambda_i \rangle - l_i^\star(\lambda_i) \ . \tag{5.10}$$

In this way the affine argument of $l_i(\cdot)$ appears in a bilinear term and $l_i^\star(\cdot)$ becomes a function of a simple argument. Consequently we can solve the saddle-point formulation

$$(z^\star, \lambda^\star, \nu^\star) = \min_{z \in Z} \operatorname*{argmax}_{\lambda, \nu} S(z, \lambda, \nu) \ , \tag{5.11}$$

where

$$S = \langle Tz + t, \lambda \rangle + \langle Az - b, \nu \rangle + (1/2) z^T Q z + c^T z - \sum_{i=1}^{M} l_i^\star(\lambda_i) \ . \tag{5.12}$$

Note that the equality constraints $Az = b$ are now treated explicitly by means of the multiplier ν. It is interesting that for indicator functions of convex cones, the Legendre-Fenchel dual (Bauschke 2006) is the indicator function of the polar cone, rendering the evaluation of $l_i^\star$ easy, especially for the standard self-dual cones.

5.3 The Algorithms

The three approaches for solving (5.4), *i.e.*, (5.6), (5.8), and (5.11) originate from Rockafellar's *Proximal method of multipliers* (Rockafellar 1956). When applying decomposition to this method, we obtain a unified framework for the three algorithms, known as the *Proximal alternating direction method of multipliers (PADMM)* which is written as:

Remark 5.1 *Termination criteria for all methods have been derived in the spirit of (Goldstein et al. 2012; Section 1). We define primal and dual residuals for ADMM (FADMM) as*

$$r^k = -t - Tz + y, \quad s^k = -\rho T^T (y^k - y^{k-1}) \ ,$$

for AMA (FAMA) the primal residual

$$r^k = -t - Tz + y \ ,$$

while for CPI and CPII we have accordingly

$$s^k = -\begin{bmatrix} T \\ A \end{bmatrix}(\bar{z}^k - z^k) + \frac{1}{\rho}\begin{bmatrix} \lambda^{k+1} - \lambda^k \\ \nu^{k+1} - \nu^k \end{bmatrix}$$
$$r^k = \frac{1}{\tau}(z^{k+1} - z^k) \ .$$

Termination holds whenever $\|r^k\|_2 \leq \epsilon$ *and* $\|s^k\|_2 \leq \epsilon$.

Remark 5.2 *Besides the parameters defined in the corresponding section related to the specific algorithm, matrices given in problem 5.4 are required as well. Output of the underlying algorithms is a one-dimensional array containing the optimiser.*

Algorithm 5.3.1 Proximal Alternating Direction Method of Multipliers (PADMM)

REQUIRE: Initialize $z^0 \in \mathbb{R}^n$, $y_i^0 \in \mathbb{R}^{p_i}$, $\lambda^0 \in \mathbb{R}^{p_i}$, and $\rho > 0$

loop

1: $z^{k+1} = \underset{z}{\text{argmin}} \quad f(z) + \sum_{i=1}^{M} \langle \lambda_i^k, -T_i z \rangle +$
$$(\rho/2)\sum_{i=1}^{M} \| - t_i - T_i z + y_i^k\|^2 + (1/2)\|z - z^k\|_{P_1}^2$$

2: $y_i^{k+1} = \underset{y_i}{\text{argmin}} \quad l_i(y_i) + \langle \lambda_i^k, y_i \rangle + (\rho/2)\| - t_i -$
$$T_i z^{k+1} + y_i\|^2 + (1/2)\|y_i - y_i^k\|_{P_{2i}}^2, \ i = 1, \ldots, M$$

3: $\lambda_i^{k+1} = \lambda_i^k + \rho(-t_i - T_i z^{k+1} + y_i^{k+1}), \ i = 1, \ldots, M$

end loop

Algorithm 5.3.1 comes with many names, *e.g.*, *Linearized proximal method of multipliers* (L-PMM) (Shefi and Teboulle 2014), *Split Inexact Uzawa* (SIU) (Zhang et al. 2011), *Generalized Alternating Direction Method of Multipliers* (GADMM) (Deng and Yin 2012). The matrices P_1, P_{2i} are positive semidefinite and offer some flexibility in preconditioning the proximal term. The second step of the algorithm is a *proximal minimization step* and can be written via the *prox* operator of a function, defined as

$$\mathbf{prox}_{\rho f}(x) := \inf_{y \in Y}\left\{ f(y) + \frac{1}{2\rho}\|y - x\|^2 \right\} \ . \tag{5.13}$$

From this scheme we can recover:

- Alternating direction method of multiplier (ADMM) Glowinski and Marrocco (1975b), Gabay and Mercier (1976b): We set $P_1 = 0$ and $P_{2i} = 0$. ADMM converges in function values $f(z^k) + \sum_{i=1}^{M} l_i(y_i^k) \to p^\star$, in the residual $y^k - Tz^k - t \to 0$, as well

as to the dual optimum $\lambda^\star$ for an arbitrarily large stepsize ρ and with no extra assumptions.

Algorithm 5.3.2 Alternating direction method of multiplier (ADMM)

REQUIRE: Initialize $z^0 \in \mathbb{R}^p$, $\lambda^0 \in \mathbb{R}^p$, and $\rho > 0$

loop

1: $z^{k+1} = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^{M} \langle \lambda_i^k, -T_i z \rangle + (\rho/2) \sum_{i=1}^{M} \| - t_i - T_i z + y_i^k \|^2$

2: $y_i^{k+1} = \mathbf{prox}_{\frac{1}{\rho} l_i} \left(T_i z^{k+1} + t_i - \lambda_i^k / \rho \right), \ i = 1, \ldots, M$

3: $\lambda_i^{k+1} = \lambda_i^k + \rho(-t_i - T_i z^{k+1} + y_i^{k+1}), \ i = 1, \ldots, M$

end loop

- Alternating minimization algorithm (AMA) (Tseng 1991): The algorithm is a hybrid scheme, consisting of minimizing the original Lagrangian (5.5) in Step 2, and the augmented one (5.7) in Step 3 (drop all colored terms in Algorithm 5.3.1). In this way, the quadratic coupling that comes from the augmented Lagrangian term in the first step vanishes, allowing for further decomposition if the structure of f permits to do so. In order to ensure convergence, the stepsize ρ has to be taken as $\epsilon \leq \rho \leq \frac{4\sigma_f}{\|T\|^2} - \epsilon$, where $\epsilon \in (0, \frac{2\sigma_f}{\|T\|^2})$ and f has to be strongly convex, with convexity modulus σ_f. Under these assumptions, convergence of the primal sequence $z^k \to z^\star$, the dual sequence $\lambda^k \to \lambda^\star$ and the residual sequence $y^k - Tz^k - t \to 0$ can be proven (Tseng 1991).

Algorithm 5.3.3 Alternating minimization algorithm (AMA)

REQUIRE: Initialize $\lambda^0 \in \mathbb{R}^p$, and ρ within permitted range

loop

1: $z^{k+1} = \underset{z}{\operatorname{argmin}} \quad f(z) + \sum_{i=1}^{M} \langle \lambda_i^k, -T_i z \rangle$

2: $y_i^{k+1} = \mathbf{prox}_{\frac{1}{\rho} l_i} \left(T_i z^{k+1} + t_i - \lambda_i^k / \rho \right), \ i = 1, \ldots, M$

3: $\lambda_i^{k+1} = \lambda_i^k + \rho(-t_i - T_i z^{k+1} + y_i^{k+1}), \ i = 1, \ldots, M$

end loop

- Chambolle-Pock primal-dual scheme, basic version (CPI) (Chambolle and Pock 2011): Chambolle and Pock's scheme solves problem (5.11) by means of the alternation procedure (presented in Algorithm 5.3.4) which is seemingly different from Algorithm 5.3.1.

Algorithm 5.3.4 Chambolle-Pock I (CPI)

REQUIRE: Initialize $\lambda^0 \in \mathbb{R}^p$, $\nu^0 \in \mathbb{R}^m$ $z^0 \in \mathbb{R}^n$. Choose $\tau, \rho > 0$ and $\tau\rho\|(T,\ A)\|^2 < 1$, $\theta \in [0, 1]$.

loop

1: $\lambda_i^{k+1} = \mathbf{prox}_{\rho l_i^\star}\left(\lambda_i^k + \rho(T_i\bar{z}^{k+1} + t_i)\right),\ i = 1, \ldots, M$

2: $\nu^{k+1} = \nu^k + \rho(A\bar{z}^k - b)$

3: $z^{k+1} = \underset{z\in Z}{\operatorname{argmin}} \quad (1/2)z^TQz + c^Tz + \sum_{i=1}^{M} T_i^T\left\langle z, \lambda_i^{k+1}\right\rangle +$
$\left\langle z, A^T\nu^{k+1}\right\rangle + (1/2\tau)\|z - z^k\|^2$

4: $\bar{z}^{k+1} = z^{k+1} + \theta(z^{k+1} - z^k)$

end loop

As is proven in Shefi and Teboulle (2014), Algorithm 5.3.4 is equivalent to Algorithm 5.3.1, for the special choices $P_{2i} = 0$ and $P_1 = (1/\tau)I - \rho\sum_{i=1}^{M} T_i^TT_i$, with $\theta = 1$. In this way, Algorithm 5.3.4 linearizes the quadratic term that appears in Step 1 of Algorithm 5.3.1 and hence decouples the minimization problem. Note that AMA achieves the same decoupling, but in a different way. The cost of simplifying the optimization problem comes, as in AMA, with restrictions to the stepsizes, since the condition $\tau\rho\|(T,\ A)\|^2 < 1$ has to hold.

5.4 Accelerated Convergence

There are various extensions of the three methods we presented that can significantly improve their performance in practical applications. In general there are two ways to improve timings:

1. Improving the theoretical convergence rates, which is done by exploiting properties of the functions in (5.4).

2. Speeding up the computations, which can be done is several ways, *e.g.*, fast numerical linear algebra, preconditioning of the data.

In many cases the two approaches are competing. For example, one can precondition the problem so that an accelerated variant of a method can be used, but at the same time some favorable sparsity pattern of the original problem is lost. In our experience, there is no "golden rule" when it comes to choosing a particular method and applying the various extensions for speeding it up. The choice of the method should be motivated from the problem's structure and vice-versa. In the subsections that follow we aim at providing

the reader with a wide overview of several variants of the methods that improve the convergence rates. Computational speedup is not explored in the current version of the article due to space limitations.

5.4.1 How to Split

The first question that comes to mind when using a splitting method is how to perform the splitting. This choice can heavily affect the speed of the algorithm. Choosing a splitting pattern is equivalent to formulating the two subproblems that have to be solved in the algorithmic schemes 5.3.2, 5.3.3 or 5.3.4. Consequently, the choice will also restrict the options for acceleration. A general guideline would be the following:

1. Both subproblems should have a closed form solution if possible; if not, they should be cheap to solve. The whole purpose of using splitting on (5.4) is to end up with simpler subproblems.

2. More precisely, the proximal step should be simple to solve. The step constitutes often of projections onto simple constraint sets, or proximal minimizations with respect to norms.

3. Expensive operations, like matrix inversions, should be avoided. If there are quantities that do not change during the execution, they should be prefactored.

4. If an accelerated version of an algorithm can be used without heavily altering a well-structured problem, then it should be used.

ADMM In this case, most of the flexibility comes in Step 2, since Step 3 is either a simple projection or a proximal minimization operation, provided l_i is simple. The augmented Lagrangian term will contribute with a quadratic term of the form $(\rho/2)z^T \left(\sum_{i=1}^{M} T_i^T T_i\right) z$ to the objective, hence even if Q is a diagonal matrix, the resulting quadratic term is most probably dense. In this sense, one can either minimize the resulting quadratic function restricted to the subspace $Az = b$, *i.e.*, solve a KKT system (see O'Donoghue et al. (2012)), or by eliminating the equality constraint. Note that this is equivalent to taking a Newton step on a quadratic perturbation of $f(z)$, which explains why this approach needs relatively few iterations for convergence. The bottleneck is the matrix inversion that has to be performed at each iteration. If ρ is constant, one can use either a sparse LDL factorization on the KKT system, or a Cholesky factorization in the second case and consequently solve by means of forward-backward substitution (Boyd and Vandenberghe 2004a; Appendix C).

AMA The method is applicable under the assumption that f is strongly convex. On the other hand, if the assumption holds and f has some structure (*e.g.*, diagonal, block diagonal), the method should be preferred since the matrix inversion can be very cheap. In several MPC applications this is not the case though, since, in order to ensure strong convexity, f becomes a dense quadratic form for the condensed problem. Note that the spectral radius (Rota and Strang 1960) of T and the minimum eigenvalue of the quadratic term will affect the choice of the stepsize, many times leading to a very small one.

CPI This method combines properties of the other two, in the sense that the first step is still decoupled but there is no strong convexity assumption. In order to avoid densification of the quadratic term, we choose to treat the equality constraints in a Lagrangian fashion (Step 2), a choice that, along with the stepsizes' limitations, can render the algorithm slow to converge in iterations' number. Keeping Step 3 simple allows for moving some (simple) constraints directly in the objective ($z \in Z$), if the resulting optimization problem has a closed form solution. The algorithm is built such that it favors simple computations in the expense of more iterations.

5.4.2 Improvements in the Convergence Rate

All three schemes have benefited from *Nesterov's optimal relaxation sequence* as introduced in Nesterov (1983b). Nesterov's method is a variant of gradient descent, where, instead of a gradient descent update $\{x^k\}$ sequence one uses the over-relaxed sequence $\{\hat{x}^k\}$:

$$\alpha^{k+1} = \left(1 + \sqrt{4(\alpha^k)^2 + 1}\right)/2, \tag{5.14a}$$

$$\hat{x}^{k+1} = x^k + \frac{\alpha^k - 1}{\alpha^{k+1}}(x^k - x^{k-1}) \tag{5.14b}$$

with $\alpha^0 = 1$. Application of the scheme results in an $O(1/k^2)$ global rate of convergence in function values; a rate that is optimal for first order methods. Convergence in terms of the sequences is trickier to prove. Roughly speaking, when the optimal $O(1/k^2)$ rate in terms of the primal (dual) function values is achieved, the primal (dual) sequences converge with rate $O(1/k)$ (Chambolle and Pock 2011, Goldstein et al. 2012, Shefi and Teboulle 2014).

Linear convergence rates have also been proven for ADMM and CP methods under specific assumptions on the structure of problem (5.4). Due to space limitations we only present the extensions of the methods that are based on Nesterov's acceleration or similar techniques, and we collect all other special cases in a table in the end of the section.

ADMM For ADMM, convergence of the sequences $\{z^k\}, \{y^k\}, \{\lambda^k\}$ with rate $O(1/\sqrt{k})$ is proven in the recent work of Shefi and Teboulle (2014). These rates are *global* and come *with no further assumptions on the structure of the problem.*

A fast version of the method (FADMM), based on Nesterov's acceleration, was first presented in Goldstein et al. (2012). The algorithm is presented below.

Algorithm 5.4.1 Fast alternating direction method of multiplier (FADMM)

REQUIRE: Initialize $\alpha^0 = 1$, $\hat{y}^0 = y^{-1} \in \mathbb{R}^p$, $\hat{\lambda}^0 = \lambda^{-1} \in \mathbb{R}^p$, and $\rho > 0$

loop

1, 2, 3: Same as in ADMM using the over-relaxed sequences $\hat{\lambda}^k$ and $\hat{y}^k$.

4: if $E^k > 0$[1] then

5: Apply Nesterov's scheme (5.14a) to λ^k and y^k.

6: else

7: $\alpha^{k+1} = 1$, $\hat{y}^{k+1} = y^k$ and $\hat{\lambda}^{k+1} = \lambda^k$

8: end if

end loop

Nesterov's optimal relaxation is applied on the sequences $\{y^k\}$ and $\{\lambda^k\}$. The authors use an *adaptive restarting scheme* based on the residuals' error (O'Donoghue and Candes 2012). Since the accelerated sequences often exhibit an oscillatory behavior and might over(under)shoot the optimal value, a check is performed, and if the residuals increase in two subsequent iterations, the acceleration scheme is reset.

FADMM can be shown to have a global $O(1/k^2)$ convergence rate in the dual function's values under the assumption that f and l_i are strongly convex and furthermore l_i are quadratic. In the absence of these limiting assumptions, we can have an empirically fast convergence with unproved rate. All details are given in Goldstein et al. (2012). Note that FADMM can be applied to the same family of problems as ADMM with no extra assumptions and small additional computational cost.

FAMA The accelerated version of AMA makes use of Nesterov's acceleration scheme on the dual sequence $\{\lambda^k\}$ (Goldstein et al. 2012). Under the same stepsize restriction as in the basic version, *convergence of the dual objective value* at rate $O(1/k^2)$ has been proven, inspired from the convergence proof of the FISTA algorithm (Beck and Teboulle 2009). Same as with FADMM, FAMA can practically be applied to every problem that AMA can solve.

CPII For the basic version of CP (CPI), a partial primal-dual gap is shown to shrink with rate $O(1/k)$ in an ergodic sense for the sequences $\{z^k\}$, $\{\lambda^k\}$ and $\{\nu^k\}$. CP algorithm

Algorithm 5.4.2 Fast alternating minimization algorithm (FAMA)

REQUIRE: Initialize $\alpha^0 = 1$, $\hat{\lambda}^0 = \lambda^{-1} \in \mathbb{R}^p$, and $\rho \leq \frac{\sigma_f}{\|T\|^2}$

loop

1, 2, 3: Same as in ADMM using the over-relaxed dual sequence $\hat{\lambda}^k$.

4: Apply Nesterov's scheme (5.14a) to λ^k.

end loop

comes with an accelerated variant, under the assumption that f is uniformly convex, denoted here as *the second method* of Chambolle and Pock (CPII). The acceleration is achieved by means of adaptive changes of the primal and dual stepsizes τ and ρ, as well as of the relaxation parameter θ, which are updated according to the scheme:

$$\theta^k = 1/\sqrt{1 + 2\gamma\tau^k},\ \tau^{k+1} = \theta^k\tau^k,\ \rho^{k+1} = \rho^k/\theta^k\ ,$$

where $\gamma \leq \sigma_f$, assuming knowledge of the convexity modulus of f. The variant results in a *global* $O(1/k)$ *convergence rate for the primal sequence* $\{z^k\}$, (Chambolle and Pock 2011; Theorem 2).

Algorithm 5.4.3 Chambolle-Pock II (CPII)

REQUIRE: Initialize $\lambda^0 = 0 \in \mathbb{R}^p$, $\nu^0 = 0 \in \mathbb{R}^m$ $z^0 \in \mathbb{R}^n$ and $\rho^0, \tau^0 > 0$, $\tau^0\rho^0\|(T, A)\|^2 < 1$, $\theta \in [0, 1]$.

loop

1, 2, 3: Same as in CPI.

4: $\theta^k = 1/\sqrt{1 + 2\sigma_f\tau^k}$, $\tau^{k+1} = \theta^k\tau^k$, $\rho^{k+1} = \rho^k/\theta^k$

5: Same as Step 5 of CPI.

end loop

In case that Q is diagonal, the extra computational cost that comes from the acceleration is insignificant.

5.5 Case Study

We demonstrate some of the methods presented in the previous sections with an optimal control problem that involves MPC for tracking of a reference signal. We focus on explaining how to rewrite our problems so that we maximally exploit the ideas presented in Section 5.4.

In this example the linearized model of a Boeing 747-200 (B747) is considered (Hartley et al. 2013). The model has $n = 12$ states and $m = 17$ inputs and the aim is tracking

of a reference signal $r(k)$ for three of the states. We discretize with sampling period $T_s = 0.2s$ and consider in total a signal of 115 setpoints. Firstly, a *steady state target calculator* computes a pair of setpoints $(\delta x_s(k), \delta u_s(k))$ for the aircraft, according to a desired reference signal. Subsequently, an MPC controller is tracking the delivered setpoint. The steady-states are generated by solving a strongly convex dense QP with $n+m = 29$ variables and bound constraints on the inputs (Hartley et al. 2013; Section II,B). The affine term in the objective is a function of $r(k)$, hence the optimization has to be performed as many times as is the length of the reference signal. The MPC problem is a simple quadratic one, with $Q \succeq 0$ and the same bound constraints on the inputs. The affine term is also time-varying since it is a function of the generated setpoints.

Steady state calculator The problem to solve is

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\theta_s^T H_s \theta_s - h_s(k)^T \theta_s \\ \text{subject to} & \theta_{min} \leq \theta_s \leq \theta_{max} \ , \end{array} \tag{5.15}$$

with variables $\theta_s \in \mathbb{R}^{n+m}$ and $H_s \succ 0$. Since the objective is strongly convex, we can use accelerated versions of the methods. To this end, FAMA and CPII are valid options, however, the dense structure of H_s would require a forward backward substitution at each iteration, something that can be avoided. We thus take the Cholesky factorization of H_s, *i.e.*, $H_s = LL^T$, L is lower triangular and invertible and perform a change of basis, $\tilde{\theta}_s = L^T \theta_s$. Now the problem can be reformulated as

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\tilde{\theta}_s^T \tilde{\theta}_s - \tilde{h}_s(k)^T \tilde{\theta}_s \\ \text{subject to} & C\tilde{\theta}_s \leq d \ , \end{array} \tag{5.16}$$

with variables $\tilde{\theta}_s \in \mathbb{R}^{n+m}$, $\tilde{h}_s(k) = L^{-1} h_s(k)$. The matrix-vector pair (C, d) describes the polytopic constraints that are now imposed in the place of the simple bound constraints that we had in (5.15). This is the price paid for eliminating the dense Hessian in the objective. By introducing a slack variable $y = C\tilde{\theta}_s - d$, $y \leq 0$, we can apply FAMA to the modified problem with $f(\tilde{\theta}_s) = \frac{1}{2}\tilde{\theta}_s^T \tilde{\theta}_s - \tilde{h}_s(k)^T \tilde{\theta}_s$, $l(y) = \delta_-(y)$, $T = C$, $t = -d$. For the stepsize we choose $\rho = 1/\lambda_{\max}(C^T C)$.

As a second option, we use ADMM with the parameters tuned as in Ghadimi et al. (2013) in the same setting. This version achieves linear convergence rate by means of the optimal stepsize selection $\rho = 1/\sqrt{\lambda_{\min}(CC^T)\lambda_{\max}(CC^T)}$. In our case C is singular and so we consider the smallest nonzero eigenvalue.

Accordingly we can use CPII. Problem 5.16 can be written in a saddle point form as

$$\min_{\tilde{\theta}_s} \max_{\lambda} \left\{ \langle C\tilde{\theta}_s - d, \lambda \rangle + \frac{1}{2}\tilde{\theta}_s^T \tilde{\theta}_s - \tilde{h}_s(k)^T \tilde{\theta}_s - \delta_+(\lambda) \right\} \ ,$$

so we can use CPII with $Z = \mathbb{R}^{n+m}$, $l_i^\star(\lambda) = \delta_+(\lambda)$, T, t as defined above. Note that there are no equality constraints, hence there is no ν-update. We initialize the primal stepsize $\tau^0 = 100$ according to (Chambolle and Pock 2011; Theorem 2).

We solve the problem 115 times with the affine term varying slightly from one iteration to the other. We terminate based on the residual decrease, with the accuracy threshold set to 10^{-3} for FAMA and CPII and 10^{-4} for ADMM (see Remark 5.1). FAMA needs 495 iterations on average, with average time 0.85 ms per solve, ADMM 194 iterations at 0.56 ms per solve and CPII 1100 iterations at 4.9 ms per solve. The solutions achieved are quite accurate, with a normed relative error ($\|\theta_s - \theta_s^\star\|/\|\theta_s^\star\|$) of $\approx 10^{-5}$ for all the methods, sumed over all 115 instances. The optimal stepsize selection renders ADMM clearly superior in this case.

Main advantage of the presented methods stem from the fact, that while eMPC controllers are able to compute the corresponding control action in range of micro-seconds they are limited by the dimension of the given problem. In other words, for systems with relatively small amount of state variables the aforementioned approach produces satisfactory results. For more complex problems containing more optimisation variables or "non-trivial" constraints, one has to find a way to make the problem much simpler. On the other hand, the presented algorithms do not have any restrictions regarding the complexity of the underlying problem.

MPC for tracking The MPC problem described in (Hartley et al. 2013; Section II) can be written in the condensed form

$$\begin{array}{ll} \text{minimize} & \delta_{u_s}^T \delta_{u_s}^T + h(k)^T \delta_{u_s} \\ \text{subject to} & C\delta_{u_s} \leq d \ , \end{array} \tag{5.17}$$

with variables $\delta_{u_s} \in \mathbb{R}^{Nm}$, after having changed the basis in the same way as before. We solve the problem for the following scenarios: $N = 5$, cold start, warm started at the primal and dual optima of the previous solve. The outputs are reported in Table 5.1. ADMM behaves significantly better than the other two methods in terms of iterations, but FAMA is faster overall in timings. With the number of variables increasing, the cost per iteration starts being more evident when using ADMM. We observe that warm starting makes a big difference in terms of iteration counts.

5.6 Summary

In this chapter we have introduced a novel method serving to solve wide range of convex optimisation problems quickly. We have shown that under mild assumptions a control

Table 5.1: Efficiency evaluation of the presented algorithms in terms of number iterations and runtime

		ADMM	FAMA	CPII
$N = 5$	Av. No. Iters. Cold\Warm	1362 \548	2279 \778	1544\825
	Min.\Max. No. Iters. Warm	72\1504	83\5947	1\2111
	Av. Time Cold \Warm	46.90\19.82	42.74 \14.82	75.16 \40.53
	Relative error $\|(x,u) - (x^\star,u^\star)\|/\|((x^\star,u^\star))\|$	1.61×10^{-4}	1.62×10^{-4}	1.61×10^{-4}

related problem, for instance an MPC formulation can be transformed into a convex optimisation problem, where the objective function will be given as a sum of a differentiable and non-differentiable function, where the latter one is represented by an indicator function of a constraint set. The main advantage of the aforementioned transformation stems from the ability to solve the modified problem by means of simple operations, known from linear algebra, mainly multiplications between matrices and vectors. Next, we have introduced a common computational framework from which we have subsequently derived three popular splitting methods. After having introduced the basic versions of the underlying algorithms, we have derived their accelerated versions, mostly based on Nesterov's relaxation sequence. We have concluded the chapter by a non-trivial example of a Boeing 747 aircraft, where have demonstrated the functionality of the presented algorithms assuming different scenarios and assessed the efficacy of the given framework in terms of number of iterations and runtimes.

Chapter 6

Conclusions and Contributions of the Thesis

This work deals with combined topics of modelling and model predictive control of processes. We try to introduce several techniques and approaches that make MPC fast and reliable method usable also for embedded devices. These include (i) a novel approximation technique for modelling of nonlinear processes, (ii) a new compression technique, serving for memory efficient representation of eMPC solutions, and (iii) several on-line algorithms by means of one can solve convex optimisation problems very efficiently.

Approximation of nonlinear process behaviour by piece-wise affine models helps to solve MPC problems more efficiently. In case of input-output data we proposed to solve a simple, unconstrained optimization problems. Next, we discussed approximation of single-variable functions, which then serve as the basic building blocks to perform the task in higher dimensions, by transforming it into a series of one-dimensional problems. Procedures and algorithms reported in this work are implemented in our AUTOPROX toolbox, which is freely available for download from http://www.kirp.chtf.stuba.sk/~sw/. The toolbox provides an easy-to-use interface to derivation of optimal PWA approximations and is also capable to exporting the resulting models into the HYSDEL language. The proposed technique is suitable to obtain an approximation of an arbitrary non-linear function if the analytic form of the underlying function is already given. Furthermore, obtainment of the parameters of the corresponding PWA function always boils down to a sequence of approximations of single variable functions, which mathematically can be expressed as a non-linear programming problem. Next, since the proposed procedure focuses on static nonlinearities, it can be applied to a right hand side of an arbitrary system of differential equations as well. On the other hand, in higher dimensions, during

the transformation of the original problem into series of simpler ones, one has to introduce auxiliary functions, which has two main downsides. Firstly, for functions described by complex formula containing many variables, the analytic form of the final approximation function can be quite complex. Moreover, due to the already mentioned transformation, evaluation of the final approximation error can be cumbersome as well, because the propagation of the approximation error in case of multivariable functions is not straightforward at all. Therefore, in the future, besides improvement of the existing software package, we would like to focus on finding a more rigorous approach characterising the propagation of the corresponding approximation error.

In the second part of the work we have proposed to decrease memory requirements for implementation of explicit MPC solutions by applying three compression-like approaches. First, mixed-integer programming was used to derive suitable affine transformations which allow certain regions to be represented using fixed amount of data. Then, de-duplication was utilised to identify a unique subset of data and converting the regions into index set representations. Finally, the integer indices were compressed by Huffman encoding. By means of a large case study we have demonstrated that a significant memory saving can be achieved. This reduction comes at the price of having to perform additional computation on-the-fly, amount of which was quantified for each level. Efficiency of de-duplication and compression increases with growing problem dimension, which is due to the fact that regions become more complex. The proposed methodology is suitable to reduce the memory footprint of an arbitrary explicit MPC solution, even in the case of a discontinuous solution. Furthermore, it can be applied on top of other complexity reduction schemes, and many times the memory footprint can be reduced by a factor of 50. On the downside, this reduction comes at the price in a form of an increased on-line computation during the decompression phase. Unfortunately, efficacy of our proposed compression algorithm heavily depends on the geometric structure of the underlying explicit MPC solution. Nonetheless, the biggest drawback stems from the property of explicit MPC itself, since this approach is mainly applicable for small-scale systems, and the prediction model has to be either a linear or a hybrid one, in a form of a PWA or MLD model. Therefore, one possible future direction regarding this drawback should be a development of methods ensuring control laws in a closed form for non-linear systems too. However, the "curse of dimensionality" remains a severe disadvantage of this methodology. One way to overcome this problem is to apply the recently becoming popular combinatorial approach, which, however, has its own drawbacks too. Another option could be usage of fast on-line algorithms, briefly summarised in the next section.

The final part deals with operator splitting methods which under some assumptions allow to convert the original convex optimisation problem into a sequence of single oper-

ations, more specifically operations between vectors and matrices. We described a couple of algorithms, and their accelerated versions. We concluded the chapter with an example, where we showed the applicability of the aforementioned algorithms. The described numerical framework is capable to solve wide range of convex optimisation problems within mili- and microseconds, depending on the complexity of the given problem. The algorithms are mainly composed of elementary operations performed on matrices and vectors, hence they are easily implementable in mid-level languages, like C, or in embedded systems with specific hardware architecture. Nonetheless, for small-scale systems these methods can not compete with explicit model predictive control, where obtainment of the respective control action reduces to a mere function evaluation. Furthermore, in this framework we always assume linear models and in case of MLD or PWA models, one has to apply some non-convex splitting methods, where are still a lot of open questions currently under active research. Bottom line, the presented algorithms are more universal compared to explicit MPC in terms of scalability, but the main bottleneck in this setup is the requirement of a linear dynamics. In the future we would like to focus on non-convex problems, extending the range of applicability of operator splitting methods. Moreover, since the presented algorithms are both applicable on small- and large scale systems, we are planning to investigate efficacy of these methods on distributed systems containing a huge number of optimisation variables.

Bibliography

C. S. Adjiman, I. P. Androulakis, Maranas C. D., and C. A. Floudas. A global optimisation method, α bb for process design. *Computers and Chemical Engineering*, 20:419–424, 1996a. 14

C. S. Adjiman, I. P. Androulakis, C. D. Maranas, and C. A. Floudas. A global optimization method, αBB for process design. *Computers and Chemical Engineering*, 20:419–424, 1996b. 23

M. Åkerblad and A. Hansson. Efficient solution of second order cone program for model predictive control. *International Journal of Control*, 77(1):55–77, January 2004. 66

M. Annergren, A. Hansson, and B. Wahlberg. An ADMM algorithm for solving ℓ_1 regularized MPC, 2012. Manuscript. 67

K. J. Arrow, L. Hurwicz, and H. Uzawa. Studies in linear and non-linear programming. *Stanford University Press*, 1958. 3

M. Baotic, F. Borrelli, A. Bemporad, and M. Morari. Efficient On-Line Computation of Constrained Optimal Control. *SIAM Journal on Control and Optimization*, 47(5): 2470–2489, September 2008. 51

Heinz H. Bauschke. Symbolic computation of fenchel conjugates. *ACM SIGSAM Bulletin*, 2006. 70

Heinz H Bauschke and Patrick L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer Science+ Business Media, 2011. 4

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2009. 76

A. Bemporad and C. Filippi. Suboptimal explicit RHC via approximate multiparametric quadratic programming. *Journal of Optimization Theory and Applications*, 117(1): 9–38, April 2003. 50

A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999a. 19

A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999b. 17, 19

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. In *Proc. American Contr. Conf.*, 2000a. 2

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The Explicit Linear Quadratic Regulator for Constrained Systems. Technical Report AUT99-16, Automatic Control Laboratory, ETH Zurich, October 2000b. 2

A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, January 2002a. 66

A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, January 2002b. 47, 48

Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, March 1999c. ISSN 00051098. 32, 33

J. T. Betts. *Practical Methods for Optimal Control using Nonlinear Programming*. SIAM, 2001. 66

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating di rection method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011a. 67

S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004a. 74

Stephen Boyd and Lieven Vandenberghe. *Approximation and Fitting*. Cambridge University Press, fourth edit edition, 2004b. 32, 33

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 2011b. 3, 69

B. Chachuat, A. B. Singer, and P. I. Barton. Global methods for dynamic optimization and mixed-integer dynamic optimization. *Ind. Eng. Chem. Res.*, 45(25):8373–8392, 2006. 23

A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 2011. 72, 75, 77, 79

F. J. Christophersen. *Optimal Control and Analysis for Constrained Piecewise Affine Systems*. PhD thesis, ETH Zurich, Switzerland, August 2006. 9, 11

Patrick L Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer New York, 2011. 4

S. Dasgupta, Ch. Papadimitriou, and U. Vazirani. *Algorithms*. McGraw-Hill Science/Engineering/Math, 1 edition, September 2006. ISBN 0073523402. 57, 58

B. De Schutter. The extended linear complementarity problem and linear complementary-slackness systems. In *Proceedings of the European Control Conference 1999 (ECC'99)*, Karlsruhe, Germany, August–September 1999. Paper 1006-3 / CM-9.3. 17

B De Schutter and B. De Moor. The extended linear complementarity problem. *Mathematical Programming*, 1995. 20

B De Schutter and B. De Moor. Optimal traffic light control for a single intersection. *European Journal of Control*, 4:260–276, 1998. 20

B. De Schutter and T. Van den Boom. On model predictive control for max-min-plus-scaling discrete event systems. *Automatica*, 37(7):1049–1056, 2001. 17

B De Schutter and T. van den Boom. Model predictive control for max-plus-linear discrete event systems. *Automatica*, 37:1049–1056, 2001. 20

Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Rice CAAM technical report TR12-14*, 2012. 71

Alexander Domahidi, Aldo U. Zgraggen, Melanie Nicole Zeilinger, Manfred Morari, and Colin Neil Jones. Efficient interior point methods for multistage problems arising in receding horizon control. In *CDC*, 2012. 3

J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Comp. Math. Appl.*, 1956. 3

J. Eckstein and D. Bertsekas. On the Douglas—-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55: 293–318, 1992. 67

J.E. Esser. *Primal Dual Algorithms for Convex Models and Applications to Image Restoration, Registration and Nonlocal Inpainting*. 2010. URL http://books.google.ch/books?id=EVkHcgAACAAJ. 4

H. Ferreau, H. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8): 816–830, 2008a. 67

H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 2008b. 3

D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary-Value Problems*. North-Holland: Amsterdam, 1983. 67

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximations. *Computers and Mathematics with Applications*, 2:17–40, 1976a. 67

D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. *Comp. Math. Appl.*, 1976b. 3, 71

T. Geyer, F.D. Torrisi, and M. Morari. Optimal complexity reduction of polyhedral piecewise affine systems. *Automatica*, 44(7):1728–1740, July 2008. 50

Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *arXiv preprint arXiv:1306.2454*, 2013. 78

R. Glowinski and A. Marrocco. Sur l'approximation, par elements finis d'ordre un, et la resolution, par penalisation-dualité, d'une classe de problems de Dirichlet non lineares. *Revue Française d'Automatique, Informatique, et Recherche Opérationelle*, 9: 41–76, 1975a. 67

R. Glowinski and A. Marrocco. A Modification of the Arrow-Hurwicz Method for Search of Saddle Points. 1975b. 3, 71

T. Goldstein, B. O'Donoghue, and S. Setzer. Fast Alternating Direction Optimization Methods. *arXiv.org*, 2012. 70, 75, 76

P. Grieder, Z. Wan, M. Kothare, and M. Morari. Two level model predictive control for the maximum control invariant set. In *American Control Conference*, Boston, Massachusetts, June 2004. 50

B. Grünbaum. *Convex Polytopes*. Springer-Verlag, second edition, 2000. 9

O. Guler. Applications of splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Optimization*, 1992. 68

A. Hansson. A primal-dual interior-point method for robust optimal control of linear discrete-time systems. *IEEE Transactions on Automatic Control*, 45(9):1639–1655, 2000. 66

A. Hansson and S. Boyd. Robust optimal control of linear discrete-time systems using primal-dual interior-point methods. In *Proceedings of the American Control Conference*, volume 1, pages 183–187, 1998. 66

E.N. Hartley, J.L. Jerez, A. Suardi, Jan M. Maciejowski, E.C. Kerrigan, and G. Constantinides. Predictive Control using an FPGA with Application to Aircraft Control. *IEEE Transactions on Control Systems Technology*, 2013. 77, 78, 79

W. P. M. Heemels, B. De Schutter, and A. Bemporad. Equivalence of hybrid dynamical models. *Automatica*, 37(7):1085–1091, 2001. 1, 17, 20

Wilhelmus Petrus Maria Hubertina Heemels. *Linear Complementarity Systems: A Study in Hybrid Dynamics*. PhD thesis, Technische Universiteit Eindhoven, 1999. 19

W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland. Linear complementarity systems. *SIAM Journal on Applied Mathematics*, 60(4):1234–1269, 2000. 17

R.M. Hestenes. Multiplier and gradient methods. *Journal of Optimization Theory and Applications*, 1969. 3

Peter Hudzovič. *Optimalizácia*. Slovak University of Technology, 1 edition, 2004. 13

ILOG, Inc. *CPLEX User Manual*. Gentilly Cedex, France, 2003. http://www.ilog.fr/products/cplex/. 34, 52

J. Jerez, E. Kerrigan, and G. Constantinides. A condensed and sparse QP formulation for predictive control. In *CDC-ECE*, pages 5217–5222, 2011. 67

T.A. Johansen and A. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Trans. on Automatic Control*, 48: 810–815, May 2003. 50

C.N. Jones and M. Morari. Approximate Explicit MPC using Bilevel Optimization. In *European Control Conference*, Budapest, Hungary, August 2009. 50

N. Karmakar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984. 13

V. Klee and G. J. Minty. *How Good is the Simplex Algorithm*. Academic Press, 2004. 13

Donald E. Knuth. Dynamic huffman coding. *J. Algorithms*, 6(2):163–180, 1985. 3, 60

M. Kögel and R. Findeisen. Fast predictive control of linear, time-invariant systems using an algorithm based on the fast gradient method and augmented lagrange multipliers. In *Proceedings of the 2011 IEEE International Conference on Control Applications*, pages 780–785, September 2011. 67

K.Tone. An active-set strategy in an interior point method for linear programming. pages 345–360, 1993. 13

M. Kvasnica. *Real-Time Model Predictive Control via Multi-Parametric Programming: Theory and Tools*. VDM Verlag, Saarbruecken, January 2009. 51

M. Kvasnica and M. Fikar. Performance-lossless complexity reduction in explicit mpc. In *Proceedings of the 49th IEEE Conference on Decision and Control 2010*, pages 5270–5275, 2010. 50

M. Kvasnica, P. Grieder, M. Baotic, and M. Morari. Multi-Parametric Toolbox (MPT). In *Hybrid Systems: Computation and Control*, pages 448–462, March 2004. http://control.ee.ethz.ch/~mpt. 59

M. Kvasnica, F. J. Christophersen, M. Herceg, and M. Fikar. Polynomial approximation of closed-form MPC for piecewise affine systems. In *Proceedings of the 17th IFAC World Congress*, pages 3877–3882, Seoul, Korea, July 6-11 2008. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=709. 50

M. Kvasnica, A. Szűcs, and M. Fikar. Optimization-based automatic derivation of hybrid approximations. In *VOCAL 2010, Program and Abstracts*, pages 55–55, 2010. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1051. 17

M. Kvasnica, J. Löfberg, and M. Fikar. Stabilizing polynomial approximation of explicit MPC. *Automatica*, 47(10):2292–2297, 2011a. 50

M. Kvasnica, I. Rauová, and M. Fikar. Simplification of explicit MPC feedback laws via separation functions. In *Proceedings of the 18th IFAC World Congress*, pages 5383–5388, Milano, Italy, aug 2011b. 50

M. Kvasnica, A. Szűcs, and M. Fikar. Automatic derivation of optimal piecewise affine approximations of nonlinear systems. In *Preprints of the 18th IFAC World Congress Milano (Italy) August 28 - September 2, 2011*, pages 8675–8680, 2011c. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1166. 17, 32

F. Lin, M. Fardad, and M. Jovanovic. Design of optimal sparse feedback gains via the alternating direction method of multipliers. In *Proceedings of the 2012*, pages 4765–4770, June 2012. 68

K. Ling, B. Wu, and J. Maciejowski. Embedded model predictive control (MPC) using a FPGA. In *Proceedings of the 17th IFAC World Congress*, pages 15250–15255, July 2008. 67

J. Löfberg. YALMIP, 2004. Available from http://users.isy.liu.se/johanl/yalmip/. 34

S. Longo, E. Kerrigan, K. Ling, and G. Constantinides. Parallel move blocking model predictive control. In *CDC-ECE*, pages 1239–1244, 2011. 67

Christopher M. Maes. *A regularised active-set method for sparse convex quadratic programming*. PhD thesis, Institute for computational and mathematical engineering, 2010. 13

A. Makhorin. *GLPK - GNU Linear Programming Kit*, 2001. http://www.gnu.org/directory/libs/glpk.html. 52

Jacob Mattingley and Stephen Boyd. CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering*, 2012. 3, 66

M. Morari, M. Baotic, and F. Borrelli. Hybrid Systems Modeling and Control. *European Journal of Control*, 9(2-3):177–189, 2003. 1

Katta G. Murty. A new practically efficient interior point method for quadratic programming. 2006. 13

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983a. 67

Yu. Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Dokl. Akad. Nouk SSSR*, 1983b. 75

B. O'Donoghue and E.J. Candes. Adaptive Restart for Accelerated Gradient Schemes. *arXiv.org*, 2012. URL http://arxiv.org/abs/1204.3982v1. 76

B. O'Donoghue, G. Stathopoulos, and S. Boyd. A splitting method for optimal control. *IEEE Transactions on Control Systems Technology*, 2012. 74

G. Pannocchia, J. Rawlings, and S. Wright. Fast, large-scale model predictive control by partial enumeration. *Automatica*, 43(5):852–860, May 2006. 67

I. Papamichail and C. S. Adjiman. Global optimization of dynamic systems. *Computers and Chemical Engineering*, 28:403–415, 2004. 23

C. Rao, S. Wright, and J. Rawlings. Application of interior point methods to model predictive control. *Journal of optimization theory and applications*, 99(3):723–757, November 2004. 66

S. Richter, C. Jones, and M. Morari. Real-time input-constrained MPC using fast gradient methods. In *Proceedings of the 48th IEEE Conference on Decision and Control*, pages 7387 –7393, December 2009. 67

S. Richter, S. Marièthoz, and M. Morari. High-speed online MPC based on a fast gradient method applied to power converter control. In *Proceedings of the 2010 American Control Conference*, pages 4737 –4743, July 2010. 67

R.T. Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of operations research*, 1956. 70

J. A. Rossiter and P. Grieder. Using interpolation to improve efficiency of multiparametric predictive control. *Automatica*, 41:637–643, 2005. 50

G. Rota and G Strang. A note on the joint spectral radius. *Indag. Math*, 1960. 75

S. Sahni. Computationally related problems. *SIAM Journal on Computing*, 3(4):262–279, 1974. 13

F. Scibilia, S. Olaru, and M. Hovd. Approximate explicit linear MPC via delaunay tessellation. In *Proceedings of the 10th European Control Conference*, Budapest, Hungary, 2009. 50

Ron Shefi and Marc Teboulle. Rate of Convergence Analysis of Decomposition Methods Based on the Proximal Method of Multipliers for Convex Minimization. *SIAM Journal on Optimization*, 2014. 71, 73, 75, 76

J. Snoeyink. Point Location. In J. E. Goodman and J. O'Rouke, editors, *Handbook of Discrete and Computational Geometry*, chapter 30, pages 558–574. CRC Press, Boca Raton, New York, 1997. 49

R. M. Soland. An algorithm for separable non convex programming problems ii: Nonconvex constraints. *Management Science*, 17:759–773, 1971. 14

E. D. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Trans. on Automatic Control*, 26(2):346–358, April 1981. 17, 18

G. Stathopolous, A. Szűcs, and N. C. Pu, Y. Jones. Splitting methods in control. In *Proceedings of ECC, France, Strasbourg*, 2014. 66

G. Stathopoulos, A. Szűcs, Y. Pu, and C Jones. Splitting methods in control. In *European control conference to appear*, 2014. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1484.

J. Števek, A. Szűcs, M. Kvasnica, Š. Kozák, and M. Fikar. Smart technique for identifying hybrid systems. In Anikó Szakál, editor, *Proceedings of IEEE 10th Jubilee International Symposium on Applied Machine Intelligence and Informatics*, volume 10, pages 383–388. Óbuda University, Hungary, IEEE, 26-28, January 2012. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1259. 34

S. Summers, C.N. Jones, J. Lygeros, and M. Morari. A multiscale approximation scheme for explicit model predictive control with stability, feasibility, and performance guarantees. In *IEEE Conference on Decision and Control*, Shanghai, China, December 2009. 50

Yu-Ru Syau. A note on convex functions. *International J. Math*, 22:525–534, 1998. 11

A. Szűcs, M. Kvasnica, and M. Fikar. Matlab toolbox for automatic approximation of nonlinear functions. In M. Fikar and M. Kvasnica, editors, *Proceedings of the 18th International Conference on Process Control*, pages 119–124, Tatranská Lomnica, Slovakia, June 14-17, 2011 2011a. Slovak University of Technology in Bratislava. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1138. 17

A. Szűcs, M. Kvasnica, and M. Fikar. A memory-efficient representation of explicit mpc solutions. In *Proceedings of the 50th CDC and ECC*, pages 1916–1921, Orlando, Florida, 2011b. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1210. 48

A. Szűcs, M. Kvasnica, and M. Fikar. Optimal piecewise affine approximations of nonlinear functions obtained from measurements. In *4th IFAC Conference on Analysis and Design of Hybrid Systems, Eindhoven, Netherlands*, pages 160–165, 2012. URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1306.

P. Tøndel, T. Johansen, and A. Bemporad. An algorithm for multi-parametric quadratic programming and explicit MPC solutions. In *IEEE Conference on Decision and Control*, pages 1199–1204, 2001. 66

P. Tøndel and Johansen.

P. Tøndel, T. A. Johansen, and A. Bemporad. Evaluation of Piecewise Affine Control via Binary Search Tree. *Automatica*, 39(5):945–950, May 2003. 51

P. Tseng. Applications of splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM J. Control Optim.*, 1991. 72

A. Ulbig, S. Olaru, D. Dumur, and P. Boucher. Explicit solutions for nonlinear model predictive control: A linear mapping approach. In S. G. Tzafestas and P. J. Antsaklis, editors, *Proc. of of the European Control Conference 2007*, pages 3295–3302, 2007. 50

E. Ullmann. A Matlab toolbox for C-code generation for first order methods. Master's thesis, ETH Zurich, 2011. 3

G. Valencia-Palomo and J.A. Rossiter. Using Laguerre functions to improve efficiency of multi-parametric predictive control. In *Proceedings of the American Control Conference*, pages 4731–4736, Baltimore, USA, 2010. 50

B. Wahlberg, S. Boyd, M. Annergren, and Y. Wang. An admm algorithm for a class of total variation regularized estimation problems. In *To appear, Proceedings 16th IFAC Symposium on System Identification*, July 2012. 68

Y. Wang and S. Boyd. Fast model predictive control using online optimization. In *Proceedings IFAC World Congress*, pages 6974–6997, July 2008. 66

Eric W. Weisstein. The web's most extensive mathematics resource. 2010. 9

Ch. Wen, X. Ma, and B. E. Ydstie. Analytical expression of explicit MPC solution via lattice piecewise-affine function. *Automatica*, 45(4):910 – 917, 2009. ISSN 0005-1098. doi: DOI:10.1016/j.automatica.2008.11.023. 51

H.P. Williams. *Model Building in Mathematical Programming*. John Wiley & Sons, Third Edition, 1993. 18, 19, 25, 33

X. Zhang, M. Burger, and S. Osher. A Unified Primal-Dual Algorithm Framework Based on Bregman Iteration. *Journal of Scientific Computing*, 2011. 71

List of Publications

M. Kvasnica, A. Szűcs, M. Fikar, J. Drgoňa Explicit MPC of LPV Systems in the Controllable Canonical Form *In Proceedings of European Control Conference* URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1427

M. Kvasnica, R. Gondhalekar, A. Szűcs, M. Fikar Stabilizing Refinement of Low-Complexity MPC Controllers *Preprints of 4th IFAC Nonlinear Model Predictive Control Conference* URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1332

A. Szűcs, M. Kvasnica and M. Fikar. Data Compression Techniques for Complexity Reduction in Explicit MPC. *Selected Topics In Modeling and Control - University of Technology Press Bratislava*, URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1205.

J. Števek A. Szűcs, M. Kvasnica, Š Kozák and M. Fikar. Smart technique for identifying hybrid systems. *Proceedings of IEEE 10th Jubilee International Symposium on Applied Machine Intelligence and Informatics 2012*, IEEE URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1259.

A. Szűcs, M. Kvasnica and M. Fikar. MATLAB Toolbox for Automatic Approximation of Nonlinear Functions, *Proceedings of the 18th International Conference on Process Control, Slovak University of Technology in Bratislava 2011*, Tatranska Lomnica, Slovakia URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1138.

A. Szűcs, M. Kvasnica and M. Fikar. A Memory-Efficient Representation of Explicit MPC Solutions *In Proceedings of the 50th CDC and ECC 2011*, Orlando, Florida URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1210.

M. Kvasnica, A. Szűcs and M. Fikar. Automated Piecewise Affine Approximation of Nonlinear Systems 2011, *Selected Topics on Constrained and Nonlinear Control. Preprints*, STU Bratislava - NTNU Trondheim URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1063.

M. Kvasnica, A. Szűcs and M. Fikar. Automatic Derivation of Optimal Piecewise Affine Approximations of Nonlinear Systems, *Preprints of the 18th IFAC World Congress Milano (Italy) 2011*, URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1166.

M. Paulovič, M. Kvasnica, A. Szűcs and M. Fikar. Safety Verification of Rule-Based Controllers, *Proceedings of the 18th International Conference on Process Control*, URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1137.

M. Kvasnica, A. Szűcs and M. Fikar. Optimization-Based Automatic Derivation of Hybrid Approximations 2011, *Program and Abstracts*, VOCAL, Veszprém, Hungary URL http://www.kirp.chtf.stuba.sk/publication_info.php?id_pub=1051.

Curriculum Vitae

Meno a priezvisko: Alexander Szűcs
Trvalý pobyt: Ibrányiho 1242/15, Kráľovský Chlmec 077 01, Slovensko
Phone: +421 915 864 631
Email: alexander.szucs@stuba.sk
Web: http://www.kirp.chtf.stuba.sk/~szucs
Dátum narodenia: 22.12.1985
Štátna príslušnosť: Slovensko

Vzdelanie

- 2010 – súčasnosť: Ph.D štúdium, študijný program: Riadenie procesov
- 2008 – 2010: inžinierské štúdium, študijný program: Automatizácia a informatizácia v chémii a potravinárstve, Fakulta chemickej a potravinárskej technológie, Slovenská technická univerzita v Bratislave
- 2005 – 2008: Bakalárske štúdium, študijný program: Automatizácia, informatizácia a manažment v chémii a potravinárstve, Fakulta chemickej a potravinárskej technológie, Slovenská technická univerzita v Bratislave

Pracovné skúsenosti

- 9.2010 – súčasnosť: interný doktorand FCHPT STU v Bratislave, Slovensko
- 10.2012 – 5.2014: vedecko-výskumný pracovník na EPFL Lausanne, Švajčiarsko

Výskumná činnosť

- Prediktívne riadenie
- Hybridné systémy
- Softvérové balíky pre riadenie

Počítačové zručnosti

- programovacie jazyky C/C++
- skriptovacie jazyky: Bash, Perl
- web technológie: HTML, XHTML, XML, DTD, XML SCHEMA, XPATH, XSLT
- operačné systémy: Linux – Ubuntu, Mandriva, Windows XP, 7, Mac OS X
- verzovacie systémy: git, mercurial
- textové editory: Vim, Openoffice, MS Office, Pages, LaTeX, Emacs
- rôzne: Matlab/Simulink

Jazykové znalosti

- slovenčina, maďarčina – materinské jazyky
- angličtina, čeština – plynule
- francúzština, nemčina – pokročilý

Resumé

Prekladaná dizertačná práca sa zaoberá s kombinovanou tematikou modelovania a prediktívneho riadenia procesov. Práca sa skladá z troch, vzájomne sa prelínajúcich oblastí, ktoré sú nasledovné:

- nová aproximačná metóda, slúžiaca na opis dynamických vlastností nelineárnych procesov
- trojvrstvová komprimačná technika, pomocou ktorej dokážeme radikálne znížiť pamäťové nároky explicitných prediktívnych regulátorov
- množina on-line algoritmov, ktoré dokážu efektívne riešiť širokú škálu optimalizačných problémov

Matematické modely reálnych zariadení hrajú veľmi dôležitú úlohu vo viacerých oblastiach spojených s procesným riadením. Najväčšou výzvou je nájsť matematický model, ktorý je dostatočne presný a zároveň nie príliš zložitý. Pochopiteľne, najpresnejšie simulačné výsledky sú dosiahnuteľné pomocou nelineárnych modelov, avšak teória návrhu riadenia, založená na báze už spomenutých modelov nie je rozpracovaná v dostatočnej miere. Najbežnejším spôsobom zjednodušenia je rozvoj do Taylorovho radu, ktorý umožňuje tvorbu linearizovaného modelu v okolí jedného operačného bodu. Modely získané touto procedúrou dokážu exaktne opísať dynamické vlastností pôvodného nelineárneho modelu v okolí zvoleného operačného bodu, ale ich presnosť klesá so vzrastajúcou vzdialenosťou od vybraného linearizačného bodu. Najracionálnejším riešením je použiť viacero aproximačných bodov, takto vytvárajúc rôzny počet lokálnych modelov. Matematicky môže byť táto idea reprezentovaná pomocou hybridných systémov. Hybridné modely dokážu prepojiť spojitú dynamiku s logickými premennými a na základe pravdivostnej hodnoty binárnych premenných máme možnosť si vybrať príslušný lokálny model. V prvej časti tejto práce sme si predstavili novú aproximačnú techniku, pomocou ktorej hľadanie

parametrov výslednej aproximovanej funkcie dokážeme naformulovať ako optimalizačný problém. Ukázali sme, že za istých podmienok pôvodný, dosť často zložitý problém dokážeme pretransformovať na sekvenciu jednorozmerných aproximačných problémov. Využitím základného, jednorozmerného stavebného kameňa, našu metódu sme rozšírili na dvojrozmerné funkcie. Navyše sme ukázali, že pomocou vhodných substitúcií, dokážeme pretransformovať ľubovolnú neseparovateľnú funkciu do separovateľnej podoby. Pri najtriviálnejšom scenári sme predpokladali existenciu analytického tvaru nelineárnej aproximovanej funkcie, na druhej strane pri absencii nelineárneho výrazu sme navrhovali aplikovať dvojkrokovú procedúru na získanie aproximačnej funkcie zo vstupno-výstupných dát, kde v prvej fáze sme sa snažili nájsť koeficienty aproximačnej funkcie. Keď sme už mali k dispozícii analytický tvar aproximovanej funkcie, aplikovaním našej procedúry sme ľahko získali finálnu aproximáciu. Predstavená aproximačná procedúra je zabalená do kompaktného softvérového balíka, ktorý zároveň umožňuje export parametrov optimálnej PWA aproximácie do jazyka HYSDEL.

V druhej polovici 20.storočia prediktívne riadenie sa stalo veľmi populárnou riadiacou stratégiou, hlavne kvôli jeho vlastnosti efektívne sa zaobchádzať s ohraničeniami. Hlavná idea tohoto prístupu spočíva v riešení optimalizačného problému v každej perióde vzorkovania, takto získajúc sekvenciu optimálnych akčných zásahov. Z tejto sekvencie sa vyextrahuje prvý iba člen a aplikuje sa do reálneho zariadenia. Táto metodológia sa veľmi často označuje ako riadenie s pohyblivým horizontom. Riešením optimalizačného problému v každom kroku dokážeme tlačiť stavové veličiny do nuly alebo sledovať nejakú referenčnú trajektóriu. Takýto on-line prístup je veľmi efektívny v prípade systémov s pomalou dynamikou, na druhej strane, jeho implementácia je ťažkopádna pre rýchle mechatronické procesy. Našťastie, na začiatku 21. storočia bolo ukázané, že namiesto riešenia optimalizačnej metódy on-line, optimálne riešenie sa dá predpočítať pre všetky možné počiatočné podmienky off-line, čo sa označuje ako explicitné prediktívne riadenie. Takto získané riešenie sa dá uložiť vo forme vyhľadávacej tabuľky, ktorá obsahuje samotné zákony riadenia a k nim prislúchajúce regióny. Výhody sú dvojaké. Všetky zákony riadenia sú affinnou funkciou stavovej veličiny, čo znamená, že získanie príslušného zákona riadenia sa redukuje na nájdenie konkrétneho regiónu a následnú evaluáciu k nemu prislúchajúcej affinnej funkcie. Druhú výhodu predstavuje fakt, že riešenie sa dá získať pomocou multiparametrického programovania. Explicitné MPC stále predstavuje veľmi atraktívny smer, avšak disponuje aj s veľkým nedostatkom. Táto metóda je predovšetkým aplikovateľná pre systémy s menším počtom stavov. Ďalej, nedokáže sa adaptovať pre riadenie systémov s časovo premenlivými parametrami, pretože parametre takéhoto regulátora sú rátané pre zafixovaných parametroch. Obrovskú výzvu predstavuje aj implementácia takejto riadiacej stratégie na výpočtových platformách s obmedzeným množstvom pamäte.

Preto, v druhej časti tejto práce sme navrhovali efektívnu komprimačnú techniku, slúžiacu na zníženie pamäťových nárokov explicitných prediktívnych regulátorov. Prvá vrstva nájde podmnožinu regiónov, pomocou ktorých sa dajú bez problémov zrekonštruovať tie zvyšné. Ukázali sme, že problém hľadania bázických regiónov sa dá naformulovať ako problém celočíselného programovania. Druhá vrstva sa môže aplikovať na dáta pochádzajúce z tej prvej alebo nezávisle. V tomto prípade, pamäť sa šetrí identifikáciou pozitívnych a negatívnych duplicít v príslušnej polpriestorovej reprezentácii polytopických regiónov. Nájdené duplicity sú reprezentované ako prosté smerníky na množinu unikátnych dát. Na poslednej vrstve, smerníky získané z druhej vrstvy sa zakódujú pomocou Huffmanovho kódovania. Smerníky, reprezentované pomocou celých čísiel sú asociované bitovými vzormi, a to v závislosti od frekvencie ich výskytu. Hlavnými výhodami tejto metódy sú, že je aplikovateľná na ľubovoľné explicitné riešenie a v niektorých prípadoch pamäťový otlačok sa môže redukovať až 50-násobne. Ne druhej strane, efektívnosť metódy je vysoko závislá na geometrickej štruktúre daného riešenia. Navyše, pri riešení príslušného optimalizačného problému vždy sa uvažuje lineárny alebo hybridný predikčný model. Preto, v budúcnosti by sme sa chceli zameriavať na hľadanie riešení zákonov riadenia v uzavretej forme uvažovaním nelineárnych predikčných modelov. Avšak hlavný nedostatok, týkajúci sa škálovateľnosti explicitného prediktívneho riadenia ostáva naďalej veľkým problémom, ktorý by sa dalo možno odstrániť aplikovaním rýchlych on-line algoritmov.

Metóda alternatívnych smerov upútala veľkú pozornosť v predovšetkým v oblastiach spracovania signálov, strojového učenia, kde ľudia potrebujú riešiť optimalizačné problémy s veľkým počtom premenných a nediferencovateľnou účelovou funkciou. Navyše bolo ukázané, že aj MPC formulácie môžu byť pretransformované do podoby konvexných optimalizačných metód, obsahujúcich diferencovateľnú a nediferencovateľnú časť vo forme indikátorovej funkcie, takto umožňujúc implementáciu metód alternatívnych smerov. Práve to prepojenie medzi prediktívnym riadením a konvexnou optimalizáciou nás viedlo k takzvaným operátorovým metódam, pomocou ktorých pri splnení istých predpokladov dokážeme prekonvertovať pôvodný konvexný optimalizačný problém na sekvenciu jednoduchých operácií známych z lineárnej algebry. V tretej časti práce sme opisovali sadu on-line algoritmov, slúžiacich na riešenie širokej škály konvexných optimalizačných algoritmov. Ukázali sme, ako sa dajú odvodiť už spomenuté algoritmy z jednotnej štruktúry, charakterizujúcej spoločné črty každého algoritmu. Najprv sme charakterizovali základné algoritmy a následne ich akcelerované verzie. Ako to už bolo spomenuté, prezentované algoritmy obsahujú jednoduché operácie vykonaných pomocou matíc alebo vektorov, preto sú jednoducho implementovateľné napríklad v programovacom jazyku C alebo v zabudovaných systémoch so špeciálnou hardvérovou architektúrou. Musí sa však skonštatovať, že pre dynamické systémy s menším počtom stavov odprezentované algoritmy nemôžu kon-

kurovať explicitnému prediktívnemu riadeniu, kde získanie akčných zásahov sa redukuje na jednoduchú evaluáciu affinnej funkcie. Veľkou reštrikciou týchto algoritmov ďalej je v tom, že uvažuje sa iba lineárna dynamika. V prípade hybridných predikčných modelov by sme museli siahať po niektorej z metód nekonvexnej operátorových metód, ktoré sú stále predmetom aktívneho výskumu.