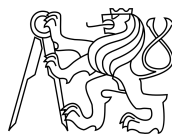


CZECH TECHNICAL UNIVERSITY IN PRAGUE  
FACULTY OF ELECTRICAL ENGINEERING  
DEPARTMENT OF CONTROL ENGINEERING



## DIPLOMA THESIS

Distributed Predictive Control

**Author:** Petr Endel

**Supervisor:** prof. Ing. Vladimír Havlena, CSc.

Prague, 2012

České vysoké učení technické v Praze  
Fakulta elektrotechnická

katedra řídicí techniky

## ZADÁNÍ DIPLOMOVÉ PRÁCE

Student: **Ing. Petr Endel**

Studijní program: Kybernetika a robotika  
Obor: Systémy a řízení

Název tématu: **Distribuované prediktivní řízení**

Pokyny pro vypracování:

1. Seznamte se s problematikou dekompozice optimalizačních problémů a s prediktivní regulací. Vypracujte řešerši algoritmů distribuovaných optimalizací vhodných pro aplikaci v oblasti prediktivní regulace.
2. Na základě řešerše vyberte nejpoužívanější/nejrozšířenější algoritmy a implementujte je v Matlabu.
3. Vybrané algoritmy distribuované optimalizace aplikované na vhodném příkladu prediktivní regulace porovnejte s klasickým centrálním přístupem.

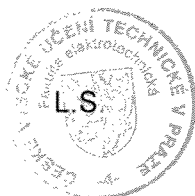
Seznam odborné literatury:

- [1] Přednášky prof. S. Boyda (zejména EE364a, EE364b), dostupné na <http://www.stanford.edu/~boyd/>
- [2] J. Nocedal, J. Wright, Numerical Optimization, Springer, 1999.
- [3] Rossiter J. A., Model-Based Predictive Control, CRC Press, 2003
- [4] Maciejowski J., Predictive control with constraints, Prentice Hall, 2001

Vedoucí: Prof. Ing. Vladimír Havlena, CSc.

Platnost zadání: do konce letního semestru 2011/2012

  
prof. Ing. Michael Šebek, DrSc.  
vedoucí katedry



  
prof. Ing. Boris Šimák, CSc.  
děkan

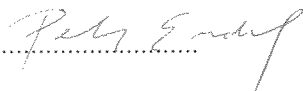
V Praze dne 31. 1. 2011

# Declaration / Prohlášení

I hereby formally declare that the work submitted is entirely my own and all sources (literature, projects, SW etc.) used are cited appropriately with the corresponding bibliographical references provided.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o do-  
držování etických principů při přípravě vysokoškolských závěrečných prací.

In Prague, date / V Praze dne .....11/1/2012

Signature / Podpis .....

# Acknowledgements

I would like to express my gratitude and thanks to professor Vladimír Havlena for his guidance and helpful comments. My sincere thanks go also to Mr. Jaroslav Pekař, PhD. from Honeywell Prague Laboratory, who was always willing to consult the topic and provide inspiring thoughts, explanations and advice. Last but not least I would like to thank to my parents for their support and patience.

# Abstract

In this thesis the framework of decomposition methods for convex optimization problems is investigated and applied, using the subgradient methods for solving the master algorithm. Potentially large engineering problems - distributed model predictive control and industrial energy network optimization - are solved in a distributed way.

Projected subgradient methods are applied to solve master problem of dual decomposition algorithm to obtain optimal solution of the problems. The comparison to centralized approach is provided, pointing out the main features of the distributed approach. Different step size rules of subgradient methods (together with advanced rules by Polyak and Nesterov with their modifications) are implemented, their convergence properties are compared and their implementation demands are also assessed. In the energy network optimization the enhanced method introducing so called net constraints is successfully applied and the influence of electricity price is investigated.

## Keywords

distributed model predictive control, decomposition methods, convex optimization, duality, subgradient methods

# Abstrakt

V této diplomové práci je prozkoumána problematika dekompozičních metod pro úlohy konvexní optimalizace a tyto metody jsou aplikovány s použitím subgradientních metod pro řešení nadřazeného algoritmu. Distribuovaným způsobem jsou řešeny potenciálně rozsáhlé inženýrské úlohy - distribuované prediktivní řízení a optimalizace průmyslové energetické sítě.

Subgradientní metoda s projekcí subgradientu je použita na vyřešení nadřazené úlohy duálního dekompozičního algoritmu pro získání optimálního řešení úloh. Je provedeno srovnání s centrálním přístupem, přičemž je poukázáno na hlavní znaky distribuovaného přístupu. Jsou implementována různá pravidla pro velikost kroku subgradientních metod (společně s pokročilými pravidly Polyaka a Nesterova a jejich modifikacemi), jsou porovnány jejich konvergenční vlastnosti a jsou posouzeny také jejich požadavky na implementaci. V případě energetické sítě je úspěšně aplikována rozšířená metoda řešení zavádějící takzvaná omezení na síť a je prozkoumán vliv ceny elektřiny na řešení úlohy.

## Klíčová slova

distribuované prediktivní řízení, dekompoziční metody, konvexní optimalizace, dualita, subgradientní metody



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context of decomposition techniques . . . . .	1
1.2	Motivation of the thesis . . . . .	2
1.3	Objectives of the thesis . . . . .	2
1.4	Historical overview and state-of-the-art . . . . .	2
1.4.1	Decomposition methods . . . . .	2
1.4.2	Distributed MPC . . . . .	3
1.5	Methods of the thesis . . . . .	4
1.6	Outline of the thesis . . . . .	4
<b>2</b>	<b>Model Predictive Control</b>	<b>6</b>
2.1	Introduction . . . . .	6
2.2	Formulation . . . . .	6
2.2.1	Predicted output and state . . . . .	7
2.2.2	Cost function . . . . .	8
2.2.3	Constraints . . . . .	9
2.2.4	Formulation of optimization problem . . . . .	10
2.2.5	Blocking . . . . .	10
2.3	Example . . . . .	11
<b>3</b>	<b>Optimization</b>	<b>13</b>
3.1	Introduction . . . . .	13
3.2	Standard form of an optimization problem . . . . .	13
3.3	Introduction to duality . . . . .	14
3.4	Optimality conditions . . . . .	16
3.5	Connection between optimization and mechanics . . . . .	17
<b>4</b>	<b>Subgradient methods</b>	<b>20</b>
4.1	Introduction . . . . .	20
4.2	Projected subgradient method for dual problems . . . . .	21



4.3	Basic step size rules . . . . .	23
4.3.1	Constant step size . . . . .	23
4.3.2	Constant step length . . . . .	24
4.3.3	Square summable, but not summable step sizes . . . .	24
4.3.4	Non-summable diminishing step sizes . . . . .	24
4.3.5	Non-summable diminishing step lengths . . . . .	24
4.4	Advanced step size rules . . . . .	25
4.4.1	Optimal step size by Polyak - with optimal value known	25
4.4.2	Optimal step size by Polyak - with optimal value un- known . . . . .	26
4.4.3	Algorithm by Nesterov . . . . .	27
5	Decomposition methods	29
5.1	Basic idea, separable problems . . . . .	29
5.2	Complicating variable . . . . .	30
5.2.1	Primal decomposition . . . . .	30
5.2.2	Dual decomposition . . . . .	31
5.3	Complicating constraint . . . . .	32
5.3.1	Primal decomposition . . . . .	32
5.3.2	Dual decomposition . . . . .	32
5.4	General form . . . . .	33
5.4.1	Primal decomposition . . . . .	34
5.4.2	Dual decomposition . . . . .	34
5.5	Example . . . . .	36
5.5.1	Problem formulation . . . . .	36
5.5.2	Solution using primal decomposition . . . . .	37
5.5.3	Solution using dual decomposition . . . . .	38
6	Applications to engineering problems	41
6.1	Dynamic control - MPC . . . . .	41
6.1.1	Problem formulation . . . . .	41
6.1.2	Distributed algorithm and its solution . . . . .	44
6.1.3	Comparison to centralized approach . . . . .	45
6.1.4	Illustration of possible deployment . . . . .	45
6.1.5	Analysis of different step size rules . . . . .	52
6.1.6	Comparison of different step size rules . . . . .	58
6.2	Static control - industrial energy network . . . . .	61
6.2.1	Problem formulation . . . . .	61
6.2.2	Distributed algorithm and its solution . . . . .	65

6.2.3	Comparison to centralized approach . . . . .	67
6.2.4	Comparison of selected step size rules . . . . .	68
6.2.5	Influence of electricity price . . . . .	69
7	Conclusion . . . . .	71
7.1	Investigative part . . . . .	71
7.2	Implementation part . . . . .	71
7.3	Innovative part . . . . .	72
A	Background research . . . . .	VI
A.1	Decomposition methods . . . . .	VI
A.1.1	Distributed Newton method for network optimization [JADB09] . . . . .	VI
A.1.2	A proximal center-based decomposition method for multi-agent convex optimization [NECO08] . . . . .	VII
A.1.3	A decomposition approach to distributed analysis of networked systems [LANG04] . . . . .	VII
A.1.4	Distributed consensus algorithm via LMI-based model predictive control and primal/dual decomposition methods [WAKA10] . . . . .	VIII
A.1.5	Dynamic dual decomposition for distributed control [RANT09] . . . . .	VIII
A.1.6	A tutorial on decomposition methods for network utility maximization [PALO06] . . . . .	VIII
A.2	Distributed MPC . . . . .	IX
A.2.1	Research and development trend of distributed MPC [ZENG08] . . . . .	IX
A.2.2	Distributed model predictive control: theory and applications [VENK06] . . . . .	IX
A.2.3	Decentralized model predictive control via dual decomposition [WAKA08] . . . . .	X
A.2.4	Stability analysis of decentralized RHC for decoupled systems [KEVI05] . . . . .	X
A.2.5	Distributed MPC based on a cooperative game [MAES09] . . . . .	XI
A.2.6	Distributed state estimation and model predictive control: application to fault tolerant control [MENI09] . . . . .	XI
A.2.7	Negotiation and learning in distributed MPC of large scale systems [JAVA10] . . . . .	XII

<i>CONTENTS</i>	ix
<b>A.2.8 Distributed hierarchical MPC for conflict resolution  in air traffic control [CHAL10] . . . . .</b>	<b>XIII</b>
<b>B Contents of the CD attached</b>	<b>XV</b>

# List of Figures

2.1	Unit step response of controlled system . . . . .	11
2.2	MPC solution of one optimization problem at time step 30 . . . . .	12
2.3	Simulation of system controlled by MPC (receding horizon) . . . . .	12
3.1	Mechanics example . . . . .	18
4.1	Illustration of Polyak's step size rule . . . . .	25
5.1	Graph representation of the Problem 5.10 . . . . .	37
5.2	Primal decomposition . . . . .	39
5.3	Dual decomposition . . . . .	40
6.1	MPC example situation . . . . .	43
6.2	Unit step response of subsystem I. . . . .	43
6.3	Unit step response of subsystem II. . . . .	43
6.4	Unit step response of subsystem III. . . . .	44
6.5	Subsystems' inputs and outputs, first iterate . . . . .	46
6.6	Subsystems' inputs and outputs, 1020 <sup>th</sup> iterate . . . . .	47
6.7	Centralized (red) and distributed (green) solution (part 1) . . . . .	48
6.8	Centralized (red) and distributed (green) solution (part 2) . . . . .	49
6.9	Central master algorithm . . . . .	50
6.10	Master algorithm distributed to nets . . . . .	50
6.11	Master algorithm distributed to node(s) . . . . .	51
6.12	Extension of the network (one subsystem added) . . . . .	51
6.13	Extension of the network (two subsystems added) . . . . .	51
6.14	Constant step size convergence - basic comparison . . . . .	52
6.15	Constant step size convergence - suboptimality of the solution . . . . .	53
6.16	Constant step length convergence . . . . .	54
6.17	Selected components of gradient for $\gamma = 0.001$ . . . . .	54
6.18	Square summable, but not summable step size convergence . . . . .	55
6.19	Square summable, but not summable step size convergence - fixed ratio $\frac{a}{b}$ . . . . .	56

6.20	Non-summable diminishing step size convergence . . . . .	56
6.21	Non-summable diminishing step length convergence . . . . .	57
6.22	Convergence for algorithm by Nesterov . . . . .	58
6.23	Convergence for algorithm by Polyak . . . . .	59
6.24	Distance to the optimum - Polyak, $f^*$ known . . . . .	59
6.25	Distance to the optimum - Polyak, $f^*$ unknown . . . . .	59
6.26	Comparison of convergences of all step size rules used . . . . .	60
6.27	Industrial energy network scheme . . . . .	61
6.28	Cost curves for boilers . . . . .	62
6.29	Two-section turbine scheme . . . . .	63
6.30	Revenue functions of turbines $T_1$ and $T_2$ . . . . .	64
6.31	Revenue curve for turbine $T_3$ . . . . .	64
6.32	Industrial energy network - block diagram . . . . .	65
6.33	Distributed solution obtained by constant step size rule . . . . .	67
6.34	Comparison of convergences of all step size rules used . . . . .	69
6.35	The solutions for different $C_E$ . . . . .	70
A.1	Block diagram of controller . . . . .	XII

# List of Algorithms

4.1	General optimization algorithm . . . . .	21
4.2	Nesterov's algorithm for minimizing unconstrained convex function $f(x)$	28
5.1	Primal decomposition, general form . . . . .	35
5.2	Dual decomposition, general form . . . . .	36

# List of Tables

6.1	Comparison of convergences of all step size rules used . . . . .	60
6.2	Distributed solution - $\alpha = 0.45$ , after 60 <sup>th</sup> iterate . . . . .	68
6.3	Centralized solution . . . . .	68
6.4	Comparison of convergences of all step size rules used . . . . .	69
6.5	Flows [ton/h] at selected subsystems for different $C_E$ [CZK/MW] . . . .	70
6.6	Profit for different $C_E$ [CZK/MW] . . . . .	70

# Typesetting <sup>1</sup>

$\mathbb{R}$	set of real numbers
$\mathbb{R}^n$	set of real vectors of $n$ components
$\mathbb{R}^{m \times n}$	set of real $m \times n$ matrices
$a, b, \dots, \alpha, \beta, \dots$	scalars
$\alpha(k), \alpha^{(k)}$	scalar at time step $k$
$w, x, \dots, \lambda, \nu, \dots$	column vectors
$x(k), x^{(k)}$	vector at time step $k$
$x_i$	$i$ -th component of vector $x$
$x_{m..n}$	subvector of $x$ , $[x_m, \dots, x_n]^T$
$(x)_i$	selected components of vector $x$ corresponding to subsystem $i$
$x_k^{k+N}$	vector $[x(k)^T, \dots, x(k+N)^T]^T$
$\ x\ _2$	quadratic norm of $x$
$\lambda \succeq 0$	generalized inequality (component-wise)
$A$	matrix
$I_N$	unity matrix of order $N$
$f(x)$	function, $f(x) : \mathbb{R}^n \rightarrow \mathbb{R}$
$\partial f(x)$	subdifferential of $f$ at $x$
$\nabla f(x)$	gradient of $f$ at $x$ , $\nabla f(x) = \left[ \frac{\partial f}{\partial x} \right]^T$
$f^*, f(x^*)$	optimal value of function $f(x)$
$\Pi_C(X)$	projection of set $X$ to set $C$

---

<sup>1</sup>The typesetting common in standard optimization literature (for example [BOYD09], [BERT99] or [NOCE06]) was preferred, because most of the thesis deals with the optimization theory. This literature implicitly take  $x$  as a (column) vector - the difference between vectors and scalars should be clear from the context (or it is explicitly stated to avoid confusion).



# Chapter 1

## Introduction

### 1.1 Context of decomposition techniques

In modern control theory the requirements on the control are encapsulated into quality criterion and the control problem is translated into optimization problem [HAVL96]. Present industry processes are generally composed of different interconnected subsystems. Because of large number of sensors and actuators involved and the high performance requirements, modern control systems are becoming more and more complex. To control (in fact to optimize) the large scale systems, for example power systems, water networks or large traffic systems, various approaches are developed.

A centralized control solution, where all subsystems' interactions are considered, is a traditional option. The main drawbacks of this approach are high cost of installation, potential problems with computation times and maintenance and higher risk of failure due to their centralized nature [MENI09]. That is why the concept of decentralization was frequently used in last decades, when the system is divided into subsystems controlled by independent agents (regulators), without sharing any information between regulators. This research area is still active, for recent review of those methods see for example [BAKU08]. However, in case of strong interactions between subsystems the performance of the control can be deteriorated and the stability may not be achieved [SCAT09].

In another approach, called distributed, the controllers of different subsystems share some information in order to improve the performance, the robustness and the fault-tolerance of the closed-loop system [MAES09]. Comprehensive classification of those methods can be found in [SCAT09]. Possible enhancement of distributed approach is a hierarchical scheme, where the controllers are coordinated by an algorithm of higher level, called coordinator or master. The coordinator can be only one for the whole system or it can be spread within the system. That is the reason why the distinction between those two approaches, distributed and hierarchical, could be tricky. Decompo-

sition methods, which are used to decompose the original (large) optimization problem into smaller subproblems and the master problem, find its use in those two control approaches.

## 1.2 Motivation of the thesis

The main motivation of this thesis is to gain capability to handle large optimization problems, which are emerging in today's engineering practice (and definitely they will continue to emerge in the future). Solving those problems in a distributed/hierarchical way reduces the memory requirements and provides modularity. In this diploma thesis decomposition methods will be analysed and applied on the potentially large optimization problems - namely model predictive control (MPC) and industrial energy network optimization.

In next subsection, the objectives of the thesis will be stated. After that the background research will be summarized - the historical overview will be presented together with state-of-the-art of decomposition methods and distributed model predictive control. Then the choice of methods used in the thesis, based on the background research, will be commented. In last subsection of this introductory chapter the outline of the thesis will be presented.

## 1.3 Objectives of the thesis

The first objective of the thesis is to investigate decomposition methods of optimization problems and the predictive control. Different algorithms of decomposition techniques will be presented and those, that are suitable for predictive control, will be implemented. Then those algorithms will be applied to selected engineering problems, to solve them in a distributed way. Next objective is to compare those algorithms to centralized approach using selected problems, as well as provide comparison of the distributed algorithms applied.

## 1.4 Historical overview and state-of-the-art <sup>1</sup>

### 1.4.1 Decomposition methods

Decomposition methods of solving large-scale optimization problems first appear in the work of Dantzig [DANT60, TEBB01] concerning linear continuous problems (especially

---

<sup>1</sup>This section is based on the background research included in the Appendix A. It was not possible to avoid using specialized terms in those sections, the terms important in the following text will be explained in corresponding chapters.

when constraint matrix exhibits specific structure). According to [BERT99], other historically important papers were [EVER63] and [BEND62]. The former brings in generalized view of Lagrange multipliers method, not limiting the objective function to be differentiable or even continuous, and presents some extensions of the method (regarding the “stability” of the numerical solution). The latter article presents a method for mixed integer programming problems using iterates between subproblems and master problem. Another significant text was [LASD68]. There the dual problem is investigated thoroughly and new algorithm is proposed, using subproblem solutions to update prices. Adapted versions of those methods are reviewed for example in [BOYD07] and in the last chapter of textbook [BERT99] (with comprehensive list of references to the topic).

Recent papers in the area of decomposition methods were investigated in background research included in Appendix A.1. Some of the works focus on improving particular algorithms to get better performance - for example the paper [JADB09] (described in A.1.1) dealing with another representation of the dual Newton direction in order to obtain faster convergence in network optimization problems, or paper [NECO08] (excerpted in A.1.2), where new proximal center-based method is presented, with optimal selection of the step parameters. The subgradient methods are applied in [LANG04] for solving coupled linear matrix inequalities in a distributed algorithm obtained by primal decomposition method (the paper is summarized in A.1.3).

Decomposition methods are closely related to problems of interacting agents and game theory. The reduction of communication between agents while keeping the convergence to the consensus point is described in [WAKA10] (details are provided in A.1.4). Authors of other papers apply dual decomposition techniques to particular areas - for example to feedback control, together with performance validation of decentralized control laws, see [RANT09] and A.1.5, or to communication network utility maximization, in [PALO06] and A.1.6.

### 1.4.2 Distributed MPC

The roots of distributed control research can be tracked back into 1970s with frequently cited book [MESA70], where the theory of multilevel systems was set up. The paper [LAU72] introduces distributed scheme for controlling the commodity flow through the network and [WANG73] deals with stabilizing a linear time-invariant multivariable system by local feedback control laws. Predecessors of today’s model predictive control were studied about ten years later in [CHEN82] and [CUTL80] (although the principles of MPC had already been used in industry).

The topic of distributed model predictive control is extensively covered in [VENK06] together with broad literature review, discussing cooperative framework, distributed state estimation, vertical integration of MPCs and other related issues (the work is de-

scribed in more detail in A.2.2). The state-of-the-art of distributed MPC is presented in Appendix A.2, starting with the overview of current trends, which can be found in paper [ZENG08] (excerpted in A.2.1). In [WAKA08] (described in A.2.3) “standard” dual decomposition algorithm is applied to decentralized MPC for e.g. formation control.

The stability of decentralized MPC schemes for decoupled systems (with input and state constraints and common objective) is addressed in frequently cited [KEVI05] (see A.2.4). Authors of [MAES09] (description in A.2.5) provide sufficient conditions for practical stability of the closed-loop system consisting of two constrained linear systems coupled through inputs. Distributed estimation of states for MPC is tackled and applied to problem of Fault Tolerant Control in [MENI09] (summarized in A.2.6).

Multi agent MPC architecture presented in [JAVA10] (see A.2.7) uses reinforcement learning and two types of agents (MPCs and “negotiators”) - this approach is applied to water network model, while the real application in water network in Barcelona is in progress. Another recently published application of distributed MPC is the conflict resolution in air traffic management, in [CHAL10] (reviewed in A.2.8).

## 1.5 Methods of the thesis

Based on the background research, both the primal and dual decomposition methods will be investigated (as reviewed in [BOYD07]). For solving the master algorithm emerging from those methods the subgradient methods will be used, as presented in [WAKA08] or [LANG04]. Those methods present relatively simple approach how to solve convex optimization problem with non-differentiable objective function. Their advantage are the small memory requirements, which can play significant role in large optimization problems. Their drawback - slow convergence rate [JADB09] - might be made faster using advanced step size rules. The attention will be focused on convergence rate as well. The algorithms described in next chapters will be implemented in MATLAB computing environment (see [MATH11]).

## 1.6 Outline of the thesis

**Chapter 1** presents different approaches to deal with complex systems and problems. The topics of decomposition methods and distributed predictive control are introduced. The background research of those two topics (included in Appendix A) is summarized and the selection of methods used in the thesis is commented.

**Chapter 2** analyses model predictive control - formulation of its optimization problem, various cost functions, hard and soft constraints, together with blocking strategy. Those concepts are illustrated on the example.

**Chapter 3** provides the overview of basic optimization terms, such as Lagrangian

or dual function, and the explanation of Karush-Kuhn-Tucker optimality conditions. The connection between theoretical mechanics formulations and optimization concepts is depicted on the example, to illustrate the physical meaning of Lagrange multipliers.

**Chapter 4** covers in detail subgradient methods, which are used for solving non-differentiable convex optimization problems. The projected subgradient method is explained as well. The step size rules, which can be used for searching for the optimum, are listed.

**Chapter 5** provides compact description of decomposition methods. Both complicating variables and complicating constraints are considered, while both primal and dual approach are used to cope with them. After basic illustration the general approach is introduced in a form of primal decomposition algorithm and dual decomposition algorithm. Those concepts are verified on the example.

**Chapter 6** describes in detail the solution of two engineering problems using the methods presented in previous chapters. The first problem is dynamic (control of system consisting of three interconnected subsystems), while the second problem is static (determining the optimal steam flows in the industrial energy network). In both cases algorithms with different step size rules are compared, together with comparison to centralized solution.

**Chapter 7** summarizes the objectives achieved and the results of the thesis. The chapter is divided into three parts, describing the work done in investigative, implementation and innovative part.

## Chapter 2

# Model Predictive Control

### 2.1 Introduction

Model predictive control (MPC, also known as receding horizon control), is a technology used in industry since 1970s [HAVL05] to control multivariable constrained dynamical systems [DING10]. It is valued for its inherent ability to handle constraints while minimizing some cost (or maximizing some reward) [INST10] [ROSS03].

The crucial part of MPC controller is formed by a model of the system. The controller uses the model to predict the outputs, while its main task is to compute the inputs (manipulated variables) such that the response (controlled variables) of the system has desired properties. These properties are often expressed by so called cost function. Cost function penalizes the undesired behavior of the system and its value should be minimized (if the reward function is used, it is obvious that it should be maximized).

The prediction of inputs and outputs is made on the prediction horizon using the actual state of the system - so in this form it would be an open loop control. The feedback is introduced by applying only the first input and computing new prediction every sampling period. This feature is called receding horizon. In next subsections, the notation and basic concepts from [STECH10] will be used.

### 2.2 Formulation

Linear MPC, widely used in practice for its simplicity [MACE02], will be described. It is formed by linear model of the system and quadratic cost function. As a linear model, the impulse response, ARX or state space model could be used. The state space model was chosen in this thesis, because it gives the compact description of multivariable systems and all other formulations can be transformed into this form<sup>1</sup>,

---

<sup>1</sup>It is assumed that the reader is familiar with basics of linear systems theory, so the explanation of variables and matrices is not provided.

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k). \end{aligned} \tag{2.1}$$

Let's denote the number of inputs of this linear system by  $m$  and number of outputs by  $p$ .

### 2.2.1 Predicted output and state

If the vectors of outputs and inputs during the prediction horizon are formed as

$$\begin{aligned} y_k^{k+N-1} &= [y(k)^T \ y(k+1)^T \ \dots \ y(k+N-1)^T]^T, \\ u_k^{k+N-1} &= [u(k)^T \ u(k+1)^T \ \dots \ u(k+N-1)^T]^T \end{aligned}$$

and matrices  $P_y$  and  $H_y$  as

$$\begin{aligned} P_y &= \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{bmatrix}, \\ H_y &= \begin{bmatrix} D & & & \\ CB & D & & \\ \vdots & & \ddots & \\ CA^{N-2}B & \dots & CB & D \end{bmatrix}, \end{aligned}$$

where  $N$  is prediction horizon (scalar), the predicted output trajectories can be expressed as

$$y_k^{k+N-1} = P_y x(k) + H_y u_k^{k+N-1}, \tag{2.2}$$

where  $x(k)$  is in fact the initial state. In the same manner the expression for the system state can be formed. If

$$x_{k+1}^{k+N} = [x(k+1)^T \ x(k+2)^T \ \dots \ x(k+N)^T]^T,$$

$$P_x = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix},$$

$$H_x = \begin{bmatrix} B & & & \\ AB & B & & \\ \vdots & & \ddots & \\ A^{N-1}B & \dots & AB & B \end{bmatrix},$$

then

$$x_{k+1}^{k+N} = P_x x(k) + H_x u_k^{k+N-1} \quad (2.3)$$

is the prediction of system state during the prediction horizon of length  $N$ .

### 2.2.2 Cost function

Cost function usually has additive form, where each term represents different control requirement and is weighted according to the desired behavior. In most cases the requirements are related to reference tracking and actuator behavior (control effort or “energy”). There are many options for setting up the cost function, as well as different notations to express it - notation presented here is based on [STECH10], because it provides relatively efficient insight, although it does not correspond to all formal conventions.

The standard form of cost function is

$$J(u_k^{k+N-1}) = \frac{1}{2} \sum_{i=k}^{k+N-1} \left\{ q(i) \cdot \|e(i)\|_2^2 + r(i) \|u(i)\|_2^2 \mid x(k) \right\}, \quad (2.4)$$

where  $e(i) = y_{\text{ref}}(i) - y(i)$  is the tracking error (difference between reference signal and system output), scalar  $q(i)$  is weight of tracking error at time step  $i$  and  $r(i)$  is weight of control effort at  $i$ . Another cost function, which will be used in this thesis, penalizes movement of actuators instead of their positions (so called minimum movement controller),

$$J(u_k^{k+N-1}) = \frac{1}{2} \sum_{i=k}^{k+N-1} \left\{ q(i) \cdot \|e(i)\|_2^2 + r(i) \|\Delta u(i)\|_2^2 \mid x(k) \right\}, \quad (2.5)$$

where  $\Delta u(i) = u(i) - u(i-1)$ .

To obtain the form without the sum, weighting matrices  $Q$  and  $R$ , which contain values  $q(i)$  and  $r(i)$ ,  $i = k, \dots, k+N-1$  in diagonals, can be introduced, so that



$$J(u_k^{k+N-1}) = \frac{1}{2}[(y_{\text{ref } k}^{k+N-1} - y_k^{k+N-1})^T Q (y_{\text{ref } k}^{k+N-1} - y_k^{k+N-1}) + \dots \\ \dots + (\Delta u_k^{k+N-1})^T R (\Delta u_k^{k+N-1})], \quad (2.6)$$

where

$$\begin{aligned} y_{\text{ref } k}^{k+N-1} &= [y_{\text{ref}}(k)^T \ y_{\text{ref}}(k+1)^T \ \dots \ y_{\text{ref}}(k+N-1)^T]^T, \\ \Delta u_k^{k+N-1} &= [\Delta u(k)^T \ \Delta u(k+1)^T \ \dots \ \Delta u(k+N-1)^T]^T. \end{aligned}$$

It is obvious that all those forms use quadratic terms, which complies with the definition of linear MPC.

### 2.2.3 Constraints

As it was said, the ability to handle constraints of the process is one of the main advantages of MPC. The constraints are usually given by actuator properties, but also system outputs or state could be constrained. Two different kinds of constraints can be distinguished - hard constraints and soft constraints.

Hard constraints describe physical limitations and it is not possible to violate them. The form of hard constraints is clear,

$$\begin{aligned} u_{\min k}^{k+N-1} &\leq u_k^{k+N-1} \leq u_{\max k}^{k+N-1} \\ \Delta u_{\min k}^{k+N-1} &\leq \Delta u_k^{k+N-1} \leq \Delta u_{\max k}^{k+N-1} \\ y_{\min k}^{k+N-1} &\leq y_k^{k+N-1} \leq y_{\max k}^{k+N-1} \\ x_{\min k+1}^{k+N} &\leq x_{k+1}^{k+N} \leq x_{\max k+1}^{k+N}. \end{aligned} \quad (2.7)$$

The soft constraints can be violated, what implies a particular cost (new term is introduced into cost function). The formal description of soft constraint uses slack variable/vector  $\varepsilon(k)$ , for example

$$y_k^{k+N-1} \leq y_{\max k}^{k+N-1} + \varepsilon(k) \quad (2.8)$$

and the term added to the cost function is  $\frac{1}{2}S \|\varepsilon(k)\|_2^2$ , where  $S$  is the weight. Soft constraints are used when some disturbances are acting on the variable (system state or outputs), so they are important for practical implementation.

### 2.2.4 Formulation of optimization problem

To sum up, the basic MPC control problem, which is solved every sampling period, can be formulated as an optimization problem (the detailed explanation follows in next chapter, but the basic idea should be clear now)

$$\begin{aligned}
& \text{minimize} && J(u_k^{k+N-1}) \\
& \text{subject to} && \\
& u_{\min k}^{k+N-1} \leq && u_k^{k+N-1} \leq u_{\max k}^{k+N-1} \\
& \Delta u_{\min k}^{k+N-1} \leq && \Delta u_k^{k+N-1} \leq \Delta u_{\max k}^{k+N-1} \\
& y_{\min k}^{k+N-1} \leq && y_k^{k+N-1} \leq y_{\max k}^{k+N-1} \\
& x_{\min k+1}^{k+N} \leq && x_{k+1}^{k+N} \leq x_{\max k+1}^{k+N} \\
& && x(k+1) = Ax(k) + Bu(k) \\
& && y(k) = Cx(k) + Du(k),
\end{aligned} \tag{2.9}$$

where the cost function is

$$\begin{aligned}
J(u_k^{k+N-1}) = & \frac{1}{2}[(y_{\text{ref } k}^{k+N-1} - y_k^{k+N-1})^T Q (y_{\text{ref } k}^{k+N-1} - y_k^{k+N-1}) + \dots \\
& \dots + (\Delta u_k^{k+N-1})^T R (\Delta u_k^{k+N-1})].
\end{aligned}$$

The equations for prediction of system state 2.3 and system output 2.2 should be taken into account. Then, after some manipulations, the problem can be reformulated to a standard mathematical problem of quadratic programming

$$\begin{aligned}
& \text{minimize} && J(u_k^{k+N-1}) = \frac{1}{2}(u_k^{k+N-1})^T H (u_k^{k+N-1}) + \dots \\
& && \dots + (u_k^{k+N-1})^T F w \\
& \text{subject to} && Z u_k^{k+N-1} \leq W + V w,
\end{aligned} \tag{2.10}$$

where  $w$  is the vector of parameters, which contains for example initial state  $x(k)$ . This problem can be solved by common software tools.

### 2.2.5 Blocking

Solving optimization problem at each sampling period can be too demanding for the computation power. In the case of complex systems or systems with fast sampling period, it is necessary to reduce the on-line complexity of the problem. The most straight-forward way to do this is to reduce the number of optimization variables by so called blocking strategy.

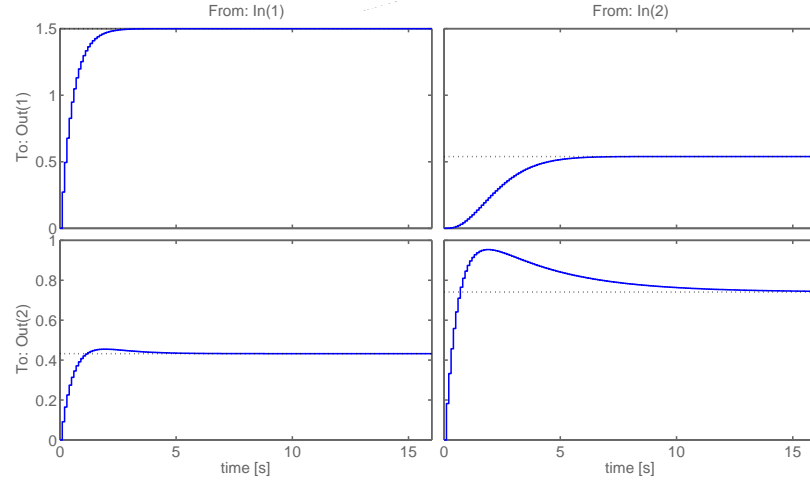


Figure 2.1: Unit step response of controlled system

Blocking fixes the manipulated variables for  $h$  periods by using blocking matrix  $B_{\text{block}} \in \mathbb{R}^{m \cdot N \times (m \cdot N/h)}$  (consisting of 0's and 1's) to define new vector of optimization variables

$$u_{\text{old } k}^{N+k-1} = B_{\text{block}} \cdot u_{\text{new } k}^{N+k-1}. \quad (2.11)$$

The term  $B_{\text{block}} \cdot u_{\text{new } k}^{N+k-1}$  is incorporated into the Problem 2.10. The number of optimization variables (components of  $u_{\text{old } k}^{N+k-1}$ , which is given by  $m \cdot N$ ) is then reduced by factor  $h$  to  $m \cdot N/h$  (number of components of  $u_{\text{new } k}^{N+k-1}$ ) as well as the original size of matrix  $H \in \mathbb{R}^{m \cdot N \times m \cdot N}$  is reduced to  $H \in \mathbb{R}^{(m \cdot N/h) \times (m \cdot N/h)}$ . It is also possible to define different lengths of blocks (it can be represented by vector  $\bar{h}$ , which contains lengths of blocks) to suit well the dynamic behavior of the system. For example at the beginning of the prediction horizon the manipulated variables are fixed for lower number of periods to deal better with the dynamics of the system.

### 2.3 Example

Consider the system with two inputs and two outputs with transfer function

$$G = \begin{bmatrix} \frac{3}{s+2} & \frac{1}{(s+1.6)(s^2+2s+1.16)} \\ \frac{s+0.7}{(s+1.8)(s+0.9)} & \frac{2(s+0.2)}{(s+1.8)(s+0.3)} \end{bmatrix}$$

sampled with period  $T = 0.1$  s. The unit step response is depicted on Figure 2.1.

The MPC for this system was implemented and then its behavior was simulated. The MPC had following properties - prediction horizon  $N = 30$  steps, the tracking weight  $q(i) = 1000, i = k, \dots, k+N-1$  (for all  $k$ ), the input weight  $r(i) = 10, i = k, \dots, k+N-1$  (for all  $k$ ), constraints on inputs  $u_{\min} = [-0.7 \ -0.4]^T$  and  $u_{\max} = [1 \ 2]^T$  for whole

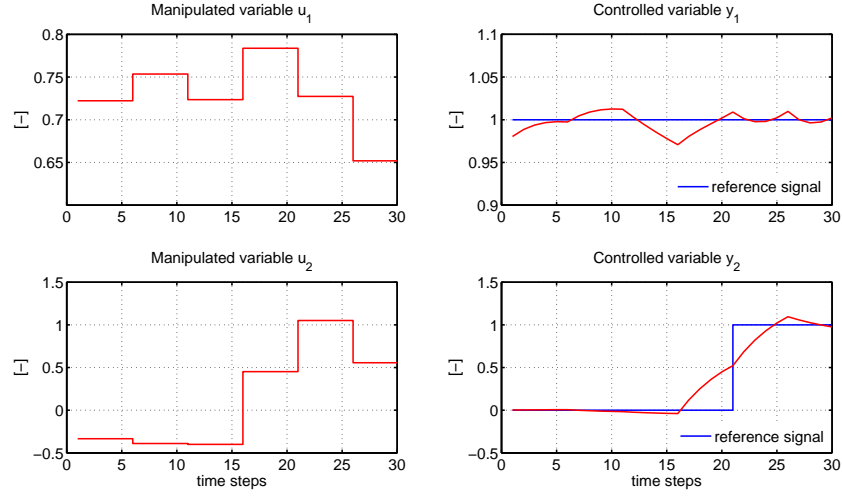


Figure 2.2: MPC solution of one optimization problem at time step 30

prediction horizon and all  $k$ . It used soft constraint on the output  $y_{\max} = 1.05$  with weight  $S = 1000$  (for all  $k$ ) in the cost function and blocking of inputs with constant value  $h = 5$ . The reference signals for the outputs of the system were unit steps at time step 25 for the first output and time step 50 for the second output.

On the Figure 2.2 the solution of one particular optimization problem at time step 30 can be seen. Here the effect of blocking can be noticed easily - the input is fixed for the given number of sampling periods. At the same time the prediction horizon of length  $N = 30$  time steps can be identified. Figure 2.3 depicts the whole simulation of the system controlled by MPC until step 100 (using receding horizon).

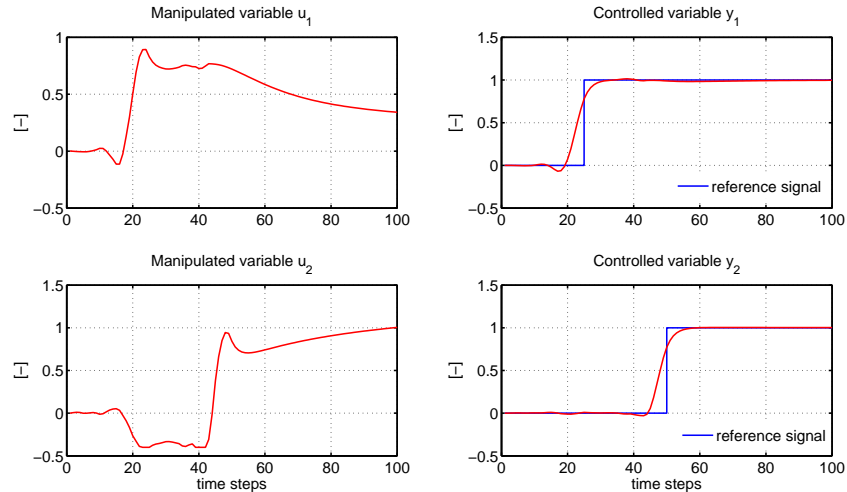


Figure 2.3: Simulation of system controlled by MPC (receding horizon)

## Chapter 3

# Optimization

### 3.1 Introduction

The optimization as a branch of mathematics finds its use in operations research and many engineering areas, including modern control theory. It is extensively studied since 1940s, when it was connected with names George Bernard Dantzig or John von Neumann.

The optimization problems basically can be divided into discrete and continuous. One of major sub-fields of continuous optimization is non-linear optimization, with its subset - convex optimization. Those areas are covered by standard textbooks, for example [POLY87], [BERT99] or [BOYD09]. This diploma thesis is focused on convex optimization and particularly on subgradient methods for solving optimization problems (discussed for example in [SCHO85]).

### 3.2 Standard form of an optimization problem

This chapter follows the notation of [BOYD09] and makes use of some parts of the optimization theory explanation from [STECH00] and [BOYD09].

The optimization problem in a standard form can be formulated as

$$\begin{aligned} & \text{minimize} && f_0(x) \\ & \text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m \\ & && h_i(x) = 0, \quad i = 1, \dots, p, \end{aligned} \tag{3.1}$$

where

$$\begin{aligned}
x \in \mathbb{R}^n & \quad \text{is the optimization variable,} \\
f_0 : \mathbb{R}^n \rightarrow \mathbb{R} & \quad \text{is the objective (cost) function,} \\
f_i : \mathbb{R}^n \rightarrow \mathbb{R} & \quad \text{are the inequality constraint functions,} \\
h_i : \mathbb{R}^n \rightarrow \mathbb{R} & \quad \text{are the equality constraint functions.}
\end{aligned}$$

In other words, solving optimization problem means searching for the value  $x$  which minimizes the objective (cost) function  $f_0(x)$  and satisfies the constraints defined by inequalities  $f_i(x) \leq 0$  and equalities  $h_i(x) = 0$ . The set of points that satisfy those constraints (the set of feasible points) is called the domain of the optimization problem.

The optimal value  $p^*$  of the Problem 3.1 (the notation  $f_0^*$  can be used) is described as

$$p^* = \inf \{f_0(x) : f_i(x) \leq 0, i = 1, \dots, m, h_i(x) = 0, i = 1, \dots, p\}. \quad (3.2)$$

If the domain of the optimization problem is an empty set, then  $p^* = \infty$  and the problem is called infeasible. If there are feasible points for which  $f_0(x_k) \rightarrow -\infty$  as  $k \rightarrow \infty$ , then  $p^* = -\infty$  and the problem is called unbounded below.

### 3.3 Introduction to duality

So called Lagrangian is formed by introducing weighted sums of constraints to the objective function

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x), \quad (3.3)$$

where  $\lambda_i$  is a Lagrange multiplier associated with the  $i$ -th inequality constraint and  $\nu_i$  is a Lagrange multiplier associated with the  $i$ -th equality constraint. The vectors  $\lambda$  and  $\nu$  are called the dual variables associated with the Problem 3.1.

The Lagrange dual function is the minimal value of the Lagrangian over all feasible  $x$ ,

$$q(\lambda, \nu) = \inf_x L(x, \lambda, \nu) = \inf_x \left( f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{i=1}^p \nu_i h_i(x) \right). \quad (3.4)$$

Suppose  $\tilde{x}$  is a feasible point of the Problem 3.1, that means  $f_i(\tilde{x}) \leq 0$  and  $h_i(\tilde{x}) = 0$ , and  $\lambda \succeq 0$ . Then

$$\sum_{i=1}^m \lambda_i f_i(\tilde{x}) + \sum_{i=1}^p \nu_i h_i(\tilde{x}) \leq 0,$$

since the first sum is non-positive and the second sum is equal to zero. Then

$$L(\tilde{x}, \lambda, \nu) = f_0(\tilde{x}) + \sum_{i=1}^m \lambda_i f_i(\tilde{x}) + \sum_{i=1}^p \nu_i h_i(\tilde{x}) \leq f_0(\tilde{x})$$

and

$$q(\lambda, \nu) = \inf_x L(x, \lambda, \nu) \leq L(\tilde{x}, \lambda, \nu) \leq f_0(\tilde{x}).$$

Since  $q(\lambda, \nu) \leq f_0(\tilde{x})$  for every feasible point, it can be concluded that the dual function is the lower bound on the optimal value  $p^*$  (see 3.2),

$$q(\lambda, \nu) \leq p^* \tag{3.5}$$

for any  $\lambda \succeq 0$  and any  $\nu$ . It is natural to ask about the best lower bound on  $p^*$  which can be obtained from dual function. That best lower bound can be found by solving the problem

$$\begin{aligned} & \text{maximize} && q(\lambda, \nu) \\ & \text{subject to} && \lambda \succeq 0. \end{aligned} \tag{3.6}$$

This problem is called the Lagrange dual problem associated with the Problem 3.1 (which is called the primal problem). Important property of the dual problem is that it is always convex, since the objective dual function is always concave and the constraint is convex (for the proof see [BERT99], Chapter 5.1, where also geometrical interpretation of Lagrange multipliers is demonstrated clearly).

The optimal value of this problem  $d^* = q(\lambda^*, \nu^*)$  (the notation  $q^*$  can be used) is the best lower bound on  $p^*$  and it satisfies so called weak duality

$$d^* \leq p^*. \tag{3.7}$$

The difference  $p^* - d^*$  is the optimal duality gap of the problem. If the optimal duality gap is zero, then  $d^* = p^*$  and so called strong duality holds.

There exist various methods to identify problems where strong duality holds, called constraints qualifications - for example Slater's condition. Slater's condition states that when the problem is convex and there exists a strictly feasible point, then strong duality holds. Strictly feasible point is a point, where inequality constraints are satisfied with strict inequalities. For details see Chapter 5 of [BOYD09].

### 3.4 Optimality conditions

Let  $x^*$  be a solution of primal problem,  $(\lambda^*, \nu^*)$  be a solution of dual problem and suppose the strong duality holds. Then

$$\begin{aligned}
 f_0(x^*) &= q(\lambda^*, \nu^*) \\
 &= \inf_x \left( f_0(x) + \sum_{i=1}^m \lambda_i^* f_i(x) + \sum_{i=1}^p \nu_i^* h_i(x) \right) \\
 &\leq f_0(x^*) + \sum_{i=1}^m \lambda_i^* f_i(x^*) + \sum_{i=1}^p \nu_i^* h_i(x^*) \\
 &\leq f_0(x^*).
 \end{aligned}$$

It is obvious, that the last two lines hold even with equality, because  $h_i(x^*) = 0$  and  $\lambda_i^* > 0$ ,  $f_i(x^*) < 0$ . From this chain of equations it can be concluded that

$$\sum_{i=1}^m \lambda_i^* f_i(x^*) = 0,$$

which means that

$$\lambda_i^* f_i(x^*) = 0, \quad i = 1, \dots, m, \quad (3.8)$$

because each term in the sum is non-positive. This equation is called complementary slackness and it is satisfied when strong duality holds. It can be seen that the optimal Lagrange multiplier has to be zero, when  $f_i(x^*) < 0$  (the constraint is not active). For differentiable constraint functions of Problem 3.1 and strong duality, the gradient of  $L$  at  $x^*$  must equal to zero, since  $x^*$  minimizes  $L(x, \lambda^*, \nu^*)$  over  $x$ :

$$\nabla f_0(x^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^p \nu_i^* \nabla h_i(x^*) = 0. \quad (3.9)$$

All these important properties can be summarized into so called Karush-Kuhn-Tucker (KKT) optimality conditions,



$$\begin{aligned}
f_i(x^*) &\leq 0, & i = 1, \dots, m \\
h_i(x^*) &= 0, & i = 1, \dots, p \\
\lambda_i^* &\geq 0, & i = 1, \dots, m \\
\lambda_i^* f_i(x^*) &= 0, & i = 1, \dots, m \\
\nabla f_0(x^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^p \nu_i^* \nabla h_i(x^*) &= 0.
\end{aligned} \tag{3.10}$$

To sum up, for any optimization problem, where strong duality holds and the objective function is differentiable, any primal and dual optimal points satisfy those conditions. If the primal problem is convex, then the KKT conditions are also sufficient for the points to be primal and dual optimal (with zero duality gap).

KKT conditions present more general approach than the frequently used method of Lagrange multipliers, including not only equality constraints, but also inequality constraints. With no inequality constraints the conditions are the same as the Lagrange conditions.

### 3.5 Connection between optimization and mechanics

In theoretical physics (mechanics) the term Lagrangian is used to summarize the dynamic behavior of systems. Particular parameters, so called generalized coordinates  $x = x(t)$ , are chosen to describe the system, its configuration and behavior. Lagrangian in theoretical physics is the function of time and those generalized coordinates together with their time derivatives  $\dot{x} = \frac{\partial x}{\partial t}$ ,

$$L = L(t, x, \dot{x}), \tag{3.11}$$

for which the principle of least action applies - the nature realizes the trajectory for which the action functional

$$S(t_A, t_B) = \int_{t_A}^{t_B} L(t, x, \dot{x}) dt \tag{3.12}$$

has an extreme. The necessary condition for the extreme gives so called Lagrange equations for the coordinates and its time derivatives (for non-dissipative systems),

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{x}} - \frac{\partial L}{\partial x} = 0. \tag{3.13}$$

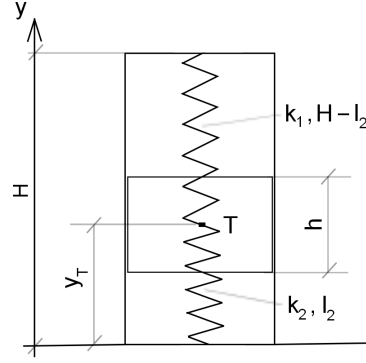


Figure 3.1: Mechanics example

The solution of those equations is realized trajectory of the system. More details can be found in [KULH11].

The choice of generalized coordinates can give the set of parameters which are dependent, then relations between them have to be incorporated - the constraints have to be introduced explicitly. That gives so called Lagrange equations of the first kind. If the parameters are independent (and there is no important parameter left), then the equations are called Lagrange equations of the second kind. The number of independent parameters is then the degree of freedom of the system. It is obvious that the methods of Lagrange multipliers and KKT conditions are used to solve the Lagrange equations of the first kind or to solve problems, where the constraints are explicit.

To give an example of mechanical interpretation of Lagrange multipliers, let's consider a system of a mass  $m$  (with center of gravity  $T$ ) connected to two springs in the shaft of height  $H$ , with the gravitation field, as depicted on Figure 3.1. The friction between the mass and the shaft is neglected. The springs have constants  $k_1$ ,  $k_2$  and lengths  $l_1 = H - l_2$  and  $l_2$ . To find the equilibrium (the static solution) it is necessary to search for the minimum of the potential energy

$$V(y_T) = \frac{1}{2}k_2(y_T - l_2)^2 + \frac{1}{2}k_1(H - l_2 - (H - y_T))^2 + mgy_T,$$

providing the constraints  $y_T \geq \frac{h}{2}$  and  $y_T \leq H - \frac{h}{2}$ . This results in the quadratic programming problem

$$\begin{aligned} \text{minimize} \quad & f_0(y_T) = \frac{1}{2}k_1(y_T - l_2)^2 + \frac{1}{2}k_2(y_T - l_2)^2 + mgy_T \\ \text{subject to} \quad & \frac{h}{2} - y_T \leq 0 \\ & y_T - H + \frac{h}{2} \leq 0. \end{aligned} \quad (3.14)$$

The solution to this problem can be obtained by forming the Lagrangian  $L(y_T, \lambda_1, \lambda_2)$

(in a optimization sense) and finding its minimum,

$$\begin{aligned}
L &= \frac{1}{2}(k_1 + k_2)(y_T - l_2)^2 + mgy_T + \lambda_1\left(\frac{h}{2} - y_T\right) + \lambda_2\left(y_T - H + \frac{h}{2}\right) \\
\frac{\partial L}{\partial y_T} &= (k_1 + k_2)(y_T - l_2) + mg - \lambda_1 + \lambda_2 = 0 \\
y_T &= \frac{-mg + (k_1 + k_2)l_2 + \lambda_1 - \lambda_2}{k_1 + k_2}.
\end{aligned} \tag{3.15}$$

The second derivative of the Lagrangian is  $k_1 + k_2$ , which is always positive, so the minimum of the Lagrangian was found.

The Problem 3.14 is convex with differentiable objective function and for sure there exists a strictly feasible point (see the situation on the Figure 3.1), so according to the Slater's condition the strong duality holds. This means that the KKT conditions can be formulated as

$$\frac{h}{2} - y_T^* \leq 0 \tag{3.16}$$

$$\begin{aligned}
y_T^* - H + \frac{h}{2} &\leq 0 \\
\lambda_1^* &\geq 0
\end{aligned} \tag{3.17}$$

$$\lambda_2^* \geq 0$$

$$\lambda_1^*\left(\frac{h}{2} - y_T^*\right) = 0 \tag{3.18}$$

$$\lambda_2^*\left(y_T^* - H + \frac{h}{2}\right) = 0$$

$$k_2(y_T^* - l_2) + k_1(y_T^* - l_2) + mg - \lambda_1^* + \lambda_2^* = 0. \tag{3.19}$$

The last equation is the zero gradient condition and it is crucial for solving the system of equations - it can be easily seen that the solution is the same as 3.15.

In fact, the Equation 3.19 is a force balance condition - the first two terms are the forces from the springs, the third term is the gravitation force. The Lagrange multipliers (last two terms) act as a contact forces in case the mass is situated in the marginal position. This can be seen from the complementary slackness conditions 3.18 - the multipliers are zero when the mass is between the marginal positions (constraints 3.16 are not active) and in the marginal position the particular multiplier becomes nonzero.

## Chapter 4

# Subgradient methods

### 4.1 Introduction

In this chapter basic concepts of subgradient methods and projected subgradient method for solving optimization problems is presented. The explanation starts with the unconstrained optimization problem. Usually it is not possible to find its analytical solution, so it is necessary to search for it iteratively. The algorithm of searching for the optimum produces a sequence of  $x^{(k)}$ ,  $k \in \mathbb{N}$  is an iteration number, where

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} \cdot \Delta x^{(k)}. \quad (4.1)$$

Vector  $\Delta x^{(k)}$  is called a step (or search direction), scalar value  $\alpha^{(k)} \geq 0$  is called a step size (step length, also scale factor). The general form of an algorithm is described in Algorithm 4.1 [BOYD09].

In case of differentiable convex objective functions the most widely used methods are the gradient method or Newton's method. They are deeply investigated in standard textbooks (for example in Chapter 9 of [BOYD09] or first chapters of [BERT99]). The subgradient methods are used for optimizing a non-differentiable convex functions - that kind of problems can arise in various applications areas (system identification, neural networks and many others) [NEDI02]. Their advantage is a simplicity and usually small memory requirements, which makes them suitable for large problems, as it was commented in Section 1.5.

In subgradient methods the search direction (for minimizing the objective function) is negative subgradient at the actual point, thus  $\Delta x^{(k)} = -g^{(k)}$  in Equation 4.1, thus

$$x^{(k+1)} = x^{(k)} - \alpha^{(k)} \cdot g^{(k)}. \quad (4.2)$$

Subgradient  $g$  of a function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  at a point  $x$  is defined as any vector  $g$  that satisfies

**Algorithm 4.1** General optimization algorithm

---

 Given an initial point  $x$  from feasible set

 Repeat ( $k$ -th iterate)

1. Determine a search direction  $\Delta x^{(k)}$
2. Choose a step size  $\alpha^{(k)}$
3. Update  $x^{(k+1)} = x^{(k)} + \alpha^{(k)} \Delta x^{(k)}$

 Until stopping criterion is satisfied
 

---

$$f(x) + g^T(y - x) \leq f(y) \quad \forall y \in \mathbb{R}^n \quad (4.3)$$

(thus the subgradient gives affine global under-estimator of the function). The set of all subgradients (the subdifferential) of  $f$  at  $x$  is denoted by  $\partial f(x)$  - any subgradient from this set can be chosen to determine the search direction. When  $f$  is differentiable, the subgradient reduces to the gradient.

In the subgradient methods, the step lengths are often computed before the optimization begins, or there is particular (relatively simple) rule for updating the step during the optimization algorithm. There are different approaches how to choose the step length, which will be discussed in Sections 4.3 and 4.4.

There is no guarantee that the value of objective function will improve in every step (the subgradient method are not descent methods) - because of that the best point and the best value of objective function found so far,  $f_{\text{best}}^{(k)}$ , should be kept in the memory. In addition, stopping criteria don't work well in case of subgradient methods [BOYD08], so in practice the subgradient method is frequently used without any formal stopping criterion (the limit could be the number of iterations).

Let's point out that in case of solving dual problem, that means finding maximum of dual function, the problem can be easily reformulated. One simple modification is described by  $\max q(x) = -\min f(x)$ , if  $f(x) = -q(x)$  (then subgradients are  $g_f(x) = -g_q(x)$ ). This means that the algorithm can look for the minimum of negative dual function and obtain the same solution  $x^*$  as in the case of maximizing the dual function.

## 4.2 Projected subgradient method for dual problems

For constrained optimization problems, the projected subgradient method is used. When the feasible set  $C$  is convex, then the method is described by

$$x^{(k+1)} = \Pi_C(x^{(k)} - \alpha^{(k)} g^{(k)}), \quad (4.4)$$

where  $\Pi_C(X)$  is the projection of set  $X$  on  $C$ . Later (in the Subsection 5.4.2 and in implementations) the special case will be used, when  $C$  is affine,  $C = \{x | Ax = b\}$ , where  $A$  is full rank matrix. In that case the projection operator is also affine and defined by

$$\Pi_C(z) = z - A^T(AA^T)^{-1}(Az - b), \quad (4.5)$$

so the subgradient update is (using  $Ax^{(k)} = b$ , the property of  $C$ )

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - \alpha^{(k)}.g^{(k)} - A^T(AA^T)^{-1}(Ax^{(k)} - A\alpha^{(k)}.g^{(k)} - b), \\ x^{(k+1)} &= x^{(k)} - \alpha^{(k)}. (I - A^T(AA^T)^{-1}A).g^{(k)}. \end{aligned} \quad (4.6)$$

To highlight some parts of the explanation from [BOYD08], let's consider a problem

$$\begin{aligned} &\text{minimize} && f_0(x) \\ &\text{subject to} && f_i(x) \leq 0, \quad i = 1, \dots, m. \end{aligned} \quad (4.7)$$

The dual function of this problem is

$$q(\lambda) = \inf_x L(x, \lambda) = f_0(x^*(\lambda)) + \sum_{i=1}^m \lambda_i f_i(x^*(\lambda)), \quad (4.8)$$

where  $x^*(\lambda)$  is a unique minimizer of the Lagrangian over  $x$ . The dual problem has a form

$$\begin{aligned} &\text{maximize} && q(\lambda) \\ &\text{subject to} && \lambda \succeq 0. \end{aligned} \quad (4.9)$$

When Slater's condition holds, the primal problem can be solved by solving the dual problem - by finding optimal  $\lambda^*$  and then  $x^* = x^*(\lambda^*)$ . The projected subgradient method (searching for the minimum of negative dual function) has a form

$$\begin{aligned} \lambda^{(k+1)} &= (\lambda^{(k)} - \alpha^{(k)}.h^{(k)})_+ \\ h^{(k)} &\in \partial(-q)(\lambda^{(k)}), \end{aligned} \quad (4.10)$$

where  $(X)_+$  denotes the projection of  $X$  to non-negative values. From explanation in Section 3.3 can be concluded that  $(-q)$  is the supremum of a family of affine functions of  $\lambda$ . The supremum is achieved by

$$-f_0(x^*(\lambda)) - \sum_{i=1}^m \lambda_i f_i(x^*(\lambda)),$$

which has the gradient

$$h = -[f_1(x^*(\lambda)), \dots, f_m(x^*(\lambda))]^T \in \partial(-q)(\lambda).$$

Projected subgradient method for dual problem can be summarized as

$$\begin{aligned} x^{(k)} &= x^*(\lambda^{(k)}) \\ \lambda_i^{(k+1)} &= (\lambda_i^{(k)} + \alpha^{(k)} \cdot f_i(x^{(k)}))_+. \end{aligned} \tag{4.11}$$

The iterates  $x^{(k)}$  become feasible only in the limit [BOYD08].

Lagrange multiplier  $\lambda_i$  can be interpreted as the price of the resource  $i$  with usage quantified by  $f_i(x)$  in Problem 4.7. Then  $x^*(\lambda)$  in Equation 4.8 minimizes the total cost, consisting of objective function plus resource costs. The algorithm adjusts prices so that the usage of resources is within the limit ( $f_i(x) \leq 0$ ). If it is not ( $f_i(x) > 0$ ), then the price  $\lambda_i$  is increased by 4.11. The prices never become negative.

It is worth pointing out that if there are equality constraints in the primal Problem 4.7, in dual problem there are no additional constraints on  $\nu$  (see 3.6) - so it doesn't influence consequent procedure. For the same reason if there are only equality constraints (no inequality constraints) in primal problem, then the dual problem is unconstrained.

### 4.3 Basic step size rules

As was stated in two previous sections, the algorithm chooses a step size  $\alpha^{(k)}$  during the iterates. There are five basic ways how to choose it, so called step size rules. The main difference from standard descent methods is that some of those step size rules permit the steps size computation before the algorithm runs.

In [BOYD08] the convergence analysis of the algorithms using those rules can be found, here only the results of the analysis are presented. The basic assumption is  $\|g^{(k)}\|_2 \leq G$  (the norm of subgradients is bounded,  $G$  is a scalar value). We will comment in more detail the features of each step size rule in Chapter 6, where two particular engineering problems will be solved by subgradient methods presented in this chapter.

#### 4.3.1 Constant step size

The simplest case occurs when  $\alpha^{(k)} = \alpha$ , where  $\alpha > 0$ . It can be proven that in this case  $f_{\text{best}}^{(k)}$  converges to within  $G^2\alpha/2$  of optimal value.

### 4.3.2 Constant step length

This rule,

$$\alpha^{(k)} = \gamma / \|g^{(k)}\|_2, \quad (4.12)$$

where  $\gamma > 0$ , keeps the distance between the successive points constant,

$$\|x^{(k+1)} - x^{(k)}\|_2 = \gamma$$

(substitute  $\alpha^{(k)}$  definition of this rule to Equation 4.2 and compute absolute value). This rule makes the subgradient method to converge to within  $G\gamma/2$  of optimal value.

### 4.3.3 Square summable, but not summable step sizes

This rule chooses step sizes that satisfy conditions

$$\alpha^{(k)} \geq 0, \quad \sum_{k=1}^{\infty} (\alpha^{(k)})^2 < \infty, \quad \sum_{k=1}^{\infty} \alpha^{(k)} = \infty. \quad (4.13)$$

For example, one sequence satisfying those conditions is

$$\alpha^{(k)} = \frac{a}{b+k},$$

where  $a > 0$ ,  $b \geq 0$ . For this step size rule the subgradient algorithm converges to the optimal value, that means  $\lim_{k \rightarrow \infty} f_{\text{best}}^{(k)} = f^*$ .

### 4.3.4 Non-summable diminishing step sizes

The sequence of steps has to satisfy

$$\alpha^{(k)} \geq 0, \quad \lim_{k \rightarrow \infty} \alpha^{(k)} = 0, \quad \sum_{k=1}^{\infty} \alpha^{(k)} = \infty. \quad (4.14)$$

An example of a sequence with those properties is

$$\alpha^{(k)} = a/\sqrt{k},$$

where  $a > 0$ . This step size rule also makes the method to converge.

### 4.3.5 Non-summable diminishing step lengths

It is a combination of two rules mentioned,

$$\alpha^{(k)} = \gamma^{(k)} / \|g^{(k)}\|_2, \quad (4.15)$$



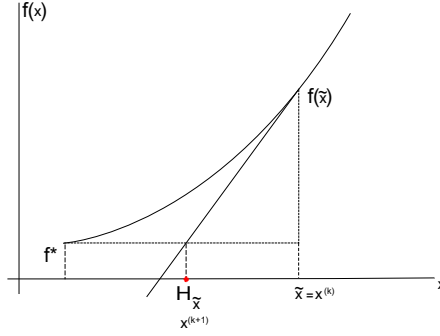


Figure 4.1: Illustration of Polyak's step size rule

where  $\gamma^{(k)}$  is non-summable diminishing,

$$\gamma^{(k)} \geq 0, \quad \lim_{k \rightarrow \infty} \gamma^{(k)} = 0, \quad \sum_{k=1}^{\infty} \gamma^{(k)} = \infty.$$

This method also converges to the optimum.

## 4.4 Advanced step size rules

### 4.4.1 Optimal step size by Polyak - with optimal value known

This dynamic step size rule was introduced by Polyak in 1980s. For minimizing the objective function it has a form

$$\alpha^{(k)} = \frac{f(x^{(k)}) - f^*}{\|g^{(k)}\|_2^2}, \quad (4.16)$$

where  $f^*$  is the optimal value. The step size chosen is optimal in a way that in every step it minimizes the upper bound of the distance to the optimum  $\|x^{(k+1)} - x^*\|_2^2$ .

To interpret the step size from slightly different point of view (based on [NEDI08]), let's define a hyperplane

$$H_{\tilde{x}} = \{y \in \mathbb{R}^n : f(\tilde{x}) + g^T(y - \tilde{x}) = f^*\},$$

where  $\tilde{x} = x^{(k)}$  is feasible non-optimal point of the problem. The example situation of one dimensional differentiable problem is depicted in the Figure 4.1, where the red dot represents  $H_{\tilde{x}}$ . Then Polyak's step size rule in order to obtain new iterate  $x^{(k+1)}$  projects the feasible point  $x^{(k)}$  on the hyperplane  $H_{\tilde{x}}$  by finding optimal  $\alpha^{(k)}$  such that

$$f(x^{(k)}) + g^T(x^{(k+1)} - x^{(k)}) = f^*.$$

Computing this optimal  $\alpha^{(k)}$  is now straightforward, because it can be substituted for

$x^{(k+1)}$  from Equation 4.2,

$$\begin{aligned} f(x^{(k)}) + g^T(x^{(k)} - \alpha^{(k)}g - x^{(k)}) &= f^* \\ f(x^{(k)}) - \alpha^{(k)}g^Tg &= f^* \\ \alpha^{(k)} &= \frac{f(x^{(k)}) - f^*}{\|g^{(k)}\|_2^2}. \end{aligned}$$

Thus the projection is  $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)}) - f^*}{\|g^{(k)}\|_2^2}g$ . It can be proven that this method converges to the optimum [BOYD08].

To solve the dual problem (thus maximizing the dual function), which will be in our main focus later, the equation for the step size is

$$\alpha^{(k)} = \frac{f^* - f(x^{(k)})}{\|g^{(k)}\|_2^2}$$

to keep the step size non-negative, or in enhanced form

$$\alpha^{(k)} = \kappa^{(k)} \cdot \frac{f^* - f(x^{(k)})}{\|g^{(k)}\|_2^2}, \quad (4.17)$$

where the factor  $\kappa^{(k)}$  is added such that  $0 < \kappa^{(k)} < 2$  [BERT99].

#### 4.4.2 Optimal step size by Polyak - with optimal value unknown

In most cases the optimal value  $f^*$  is not known, but it can be estimated. One possible estimate based on previous iterations of the algorithm is

$$\hat{f}_{(k)}^* = \min_{0 \leq i \leq k} f(x^{(i)}) - \delta, \quad (4.18)$$

where scalar  $\delta > 0$  [NEDI08]. In this estimate we use the minimum value found so far lowered by fixed value  $\delta$ , which we will call target parameter. This target parameter could be also variable, for example it can be lowered if the algorithm didn't meet the estimate from previous step. The algorithm taken from [NEDI02] (let's call it estimate by Nedic) defines the estimate as

$$\hat{f}_{(k)}^* = \min_{0 \leq i \leq k} f(x^{(i)}) - \delta_k, \quad (4.19)$$

where  $\delta_k$  is updated according to

$$\delta_{k+1} = \begin{cases} \sigma \delta_k & \text{if } f(x^{(k+1)}) \leq f_{(k)}^* \\ \max\{\beta \delta_k, \delta_{\min}\} & \text{if } f(x^{(k+1)}) > f_{(k)}^*, \end{cases}$$

where scalar  $\delta_0 > 0$  is the initial value of the target parameter,  $\delta_{\min} > 0$  is its minimal value,  $\sigma \geq 1$  and  $0 < \beta < 1$  are constant parameters for target parameter's update. To summarize this process, in step  $k$  we try to reach the estimate that is smaller by  $\delta_k$  than the best value achieved so far. If that level is reached ( $f(x^{(k+1)}) \leq f_{(k)}^*$ ), the target parameter is increased for next step (or kept at the same value if  $\sigma = 1$ ). If that level is not reached, the target parameter is reduced for next iteration (till the value of  $\delta_{\min}$ , which ensures that  $\alpha^{(k)} > 0$ ).

There are also various ways how to deal with the estimate of the optimal value while solving dual problem. For example it can be estimated from above by taking the function value of  $f(x_F)$  corresponding to primal feasible point  $x_F$ . Another estimate which can be used, taken from [BERT99] (for  $f_{\text{best}}^{(k)} = \max_{0 \leq i \leq k} f(x^{(i)}) > 0$ , the best value found so far positive), is

$$\hat{f}^* = (1 + \beta(k)) \cdot \max_{0 \leq i \leq k} f(x^{(i)}), \quad (4.20)$$

where  $\beta(k) > 0$  is a parameter which is increased if the previous iteration improved  $f_{\text{best}}^{(k)}$  and is decreased if it didn't. The combination of two estimates presented (again from [BERT99], let's call it estimate by Bertsekas) is

$$\hat{f}^* = \min \left\{ f(x_F), (1 + \beta(k)) \cdot \max_{0 \leq i \leq k} f(x^{(i)}) \right\}. \quad (4.21)$$

#### 4.4.3 Algorithm by Nesterov

Nesterov's algorithm for solving a convex problem was presented in [NEST83]. The aim of the algorithm was to reduce the amount of computation at each step as much as possible. The algorithm is shown in same form as it was presented in the original article, for minimizing the unconstrained function, in Algorithm 4.2.

If the Lipschitz constant  $L$  is known, then we can take constant step size  $\alpha^{(k)} = L^{-1}$ . Lipschitz constant of a function  $f(x)$  is the smallest  $L > 0$ , for which

$$\|g(y) - g(z)\|_2 \leq L \|y - z\|_2, \quad \forall y, z \in \text{dom } f(x). \quad (4.22)$$

With  $\alpha^{(k)} = L^{-1}$  the inequality 4.23 certainly holds (for  $i = 0$ ), so there is no need to evaluate it (again from [NEST83]) - the inner iterative part of the algorithm can be omitted. One possible modification of the original algorithm presented by Nesterov is

to omit the condition 4.23 as well and to use the basic step size rules for  $\alpha^{(k)}$  as they were defined in Section 4.3 [HONE11].

---

**Algorithm 4.2** Nesterov's algorithm for minimizing unconstrained convex function  $f(x)$

---

Select points  $y^{(0)}$  and  $z$  such that  $y^{(0)} \neq z$  and  $g(y^{(0)}) \neq g(z)$ .

Put  $k = 0$ ,  $a^{(0)} = 1$ ,  $x^{(-1)} = y^{(0)}$ ,  $\alpha^{(-1)} = \|y^{(0)} - z\|_2 / \|g(y^{(0)}) - g(z)\|_2$

$k$ -th iteration:

1. Calculate the smallest index  $i \geq 0$ , for which

$$f(y^{(k)}) - f(y^{(k)} - 2^{-i}\alpha^{(k-1)}g(y^{(k)})) \geq 2^{-i-1}\alpha^{(k-1)}\|g(y^{(k)})\|_2^2 \quad (4.23)$$

2. Put

$$\begin{aligned} \alpha^{(k)} &= 2^{-i}\alpha^{(k-1)} \\ x^{(k)} &= y^{(k)} - \alpha^{(k)}g(y^{(k)}) \\ a^{(k+1)} &= (1 + \sqrt{4(a^{(k)})^2 + 1})/2 \\ y^{(k+1)} &= x^{(k)} + (a^{(k)} - 1)(x^{(k)} - x^{(k-1)})/a^{(k+1)} \end{aligned}$$


---

The method uses local properties of objective function as well as global properties of convex functions [GONZ08]. It can be shown that the algorithm achieves optimal complexity with the smallest possible computational time per iteration - only one gradient computation and no function evaluation is needed per iterate (if the inner cycle is not considered). The proof of optimality of the algorithm is based on complexity theory and it is beyond the scope of this thesis to provide more explanation - the details can be found in [NEST04], [YAOL09] or [GONZ08].

## Chapter 5

# Decomposition methods

### 5.1 Basic idea, separable problems

The main idea of decomposition methods is to break the original optimization problem into smaller problems (let's call them subproblems). Those subproblems can be solved in parallel, which can get substantial savings. Even if the subproblems are solved sequentially, it is possible to get savings if the complexity of original problem is growing more than linearly. Another benefit of this approach is the possibility to introduce distributed solution of an engineering optimization problem - the search for optimal solution by actions of interconnected systems (agents). This chapter includes the explanation of basic concepts of decomposition methods, which follows [BOYD07].

A problem is separable (or trivially parallelizable), if it has a form of<sup>1</sup>

$$\begin{aligned} \text{minimize} \quad & f_0(x) = \sum_{j=1}^n f_j((x)_j) \\ \text{subject to} \quad & f_{i,j}((x)_j) \leq 0 \quad i = 1, \dots, m_j \\ & h_{i,j}((x)_j) = 0 \quad i = 1, \dots, p_j. \end{aligned} \tag{5.1}$$

The cost function has an additive form, where each term can be considered as a subproblem. The problem is formulated in a way where for each part of the vector  $x$  corresponding to the subproblem  $j$  (the notation  $(x)_j$ ,  $j = 1, \dots, n$  will be used) there is a separate component (term) in the cost function  $f_j((x)_j)$  and separate set of constraints. In this case the decomposition is trivial - it consists of solving optimization problems for each part  $(x)_j$  separately.

When the components of cost function are interconnected by at least one component of vector  $x$ , or the constraints involve variables from more than one subproblem, then the optimization problem is not separable. The components which interconnect the

---

<sup>1</sup>It is not an exact definition, but it is sufficient for the explanation purposes

cost function components are called complicating (coupling, public) variables and the constraints involving variables from more than one subproblem are called complicating (coupling) constraints. It is worth pointing out that when the complicating variables are fixed, then the problem is separable.

The decomposition methods allow us to solve this kind of problems - there are two approaches, primal and dual decomposition. First, the basic concepts of those methods will be shown, and then the general framework for decomposition of more complex problems will be introduced, showing particular algorithms in more detail.

## 5.2 Complicating variable

Consider the simple example, an unconstrained problem

$$\text{minimize } f_0(x) = f_1(x_{1..r}, y) + f_2(x_{r+1..n-s}, y), \quad (5.2)$$

where the vector of optimization variable  $x \in \mathbb{R}^n$  is formed as

$$x = [x_{1..r}^T, x_{r+1..n-s}^T, y^T]^T.$$

Here  $y \in \mathbb{R}^s$  is a vector of complicating (public) variables and  $x_{1..r}$ ,  $x_{r+1..n-s}$  are vectors of private (local) variables,  $r < n - s$  is an index of last private variable belonging to function  $f_1$ . Let's have a look how primal and dual techniques deal with this kind of problem.

### 5.2.1 Primal decomposition

The primal decomposition algorithm fixes the value of  $y$  - the problem is then separable - and solves subproblems (another notation is used to highlight the variables over which the optimization is done)

$$\begin{aligned} \min_{x_{1..r}} \quad & f_1(x_{1..r}, y) \\ \min_{x_{r+1..n-s}} \quad & f_2(x_{r+1..n-s}, y) \end{aligned}$$

with optimal values  $\phi_1(y)$  and  $\phi_2(y)$ . The original problem (called master problem or coordinator problem) is

$$\min_y f_0(x) = \phi_1(y) + \phi_2(y).$$

This problem can be solved by gradient method (for differentiable functions), subgradient method, bisection (when  $y$  is scalar) etc., where each iteration means solving of two subproblems. If the original problem is convex, then the master problem is also convex.

Primal decomposition method corresponds to a direct resource allocation, since the master problem allocates the existing resources by directly giving each subproblem the amount of resources that it can use [PALO06].

### 5.2.2 Dual decomposition

In dual decomposition new variables  $y_{\text{one}}$  and  $y_{\text{two}}$  (notation  $y_1$  is reserved for first component of  $y$ , so it cannot be used here) and their consistency constraint are introduced, so the original Problem 5.2 is reformulated as

$$\begin{aligned} & \text{minimize} && f_0(x) = f_1(x_{1..r}, y_{\text{one}}) + f_2(x_{r+1..n-s}, y_{\text{two}}) \\ & \text{subject to} && y_{\text{one}} = y_{\text{two}}. \end{aligned}$$

Now the dual problem is separable, because the Lagrangian is

$$\begin{aligned} L(x_{1..r}, y_{\text{one}}, x_{r+1..n-s}, y_{\text{two}}, \nu) &= f_1(x_{1..r}, y_{\text{one}}) + f_2(x_{r+1..n-s}, y_{\text{two}}) + \dots \\ &\quad \dots + \nu^T (y_{\text{one}} - y_{\text{two}}). \end{aligned}$$

Thus it is possible to solve subproblems

$$\begin{aligned} & \min_{x_{1..r}, y_{\text{one}}} && f_1(x_{1..r}, y_{\text{one}}) + \nu^T y_{\text{one}} \\ & \min_{x_{r+1..n-s}, y_{\text{two}}} && f_2(x_{r+1..n-s}, y_{\text{two}}) - \nu^T y_{\text{two}} \end{aligned}$$

with optimal values (Lagrange dual functions, see Section 3.3)  $q_1(\nu)$  and  $q_2(\nu)$ . The dual problem (master problem) is

$$\max_{\nu} \quad q(\nu) = q_1(\nu) + q_2(\nu).$$

Dual decomposition method corresponds to a resource allocation via pricing, since the master problem sets the price of the resources and each subproblem has to decide the amount of resources to be used depending on that price [PALO06]. Variable  $y_{\text{one}}$  is the amount of resources consumed by first subproblem,  $y_{\text{two}}$  is the amount of resources produced by second subproblem, while the consistency constraint  $y_{\text{one}} = y_{\text{two}}$  represents the economic equilibrium (supply equals demand).

### 5.3 Complicating constraint

Let's suppose the problem has a form

$$\begin{aligned} & \text{minimize} && f_0(x) = f_1(x_{1..r}) + f_2(x_{r+1..n}) \\ & \text{subject to} && h_1(x_{1..r}) + h_2(x_{r+1..n}) \leq 0. \end{aligned} \tag{5.3}$$

The constraint is not necessarily single - there could be a set of  $\gamma$  complicating constraints involving both  $x_{1..r}$  and  $x_{r+1..n}$ . The scalar  $r$  again just splits the vector  $x$  into subvectors.

#### 5.3.1 Primal decomposition

For the primal decomposition, a variable  $t \in \mathbb{R}^\gamma$  is introduced and the original problem is decomposed into subproblems

$$\begin{aligned} & \min_{x_{1..r}} && f_1(x_{1..r}) \\ & \text{subject to} && h_1(x_{1..r}) \leq t \end{aligned}$$

with optimal value  $\phi_1(t)$  and

$$\begin{aligned} & \min_{x_{r+1..n}} && f_2(x_{r+1..n}) \\ & \text{subject to} && h_2(x_{r+1..n}) \leq -t \end{aligned}$$

with optimal value  $\phi_2(t)$ . Variable  $t$  represents the amount of resources allocated to the first subproblem from the second subproblem. When  $t$  is fixed, the subproblems can be solved separately. The master problem is

$$\min_t \phi_1(t) + \phi_2(t).$$

Master can update the resource allocation using the (sub)gradients from the solution of subproblems.

#### 5.3.2 Dual decomposition

While using dual decomposition, the Lagrangian of the problem is



$$L(x_{1..r}, x_{r+1..n}, \lambda) = f_1(x_{1..r}) + f_2(x_{r+1..n}) + \lambda^T (h_1(x_{1..r}) + h_2(x_{r+1..n}))$$

and the Problem 5.3 can be (for fixed  $\lambda \succeq 0$ ) separated into subproblems

$$\begin{aligned} \min_{x_{1..r}} \quad & f_1(x_{1..r}) + \lambda^T h_1(x_{1..r}) \\ \min_{x_{r+1..n}} \quad & f_2(x_{r+1..n}) + \lambda^T h_2(x_{r+1..n}) \end{aligned}$$

with optimal values  $q_1(\lambda)$  and  $q_2(\lambda)$ . The master problem is

$$\max_{\lambda} q(\lambda) = q_1(\lambda) + q_2(\lambda)$$

and it is solved by the projected subgradient method presented in Section 4.2. As it was said there,  $\lambda$  can be interpreted as (non-negative) prices of resources, while master adjusts those prices in order to obtain optimality.

## 5.4 General form

The problem can be more complicated than the cases we have studied so far. In that situation, the structure of the problem could be represented by a hypergraph, where the nodes represent the subproblems - local variables, local objectives and local constraints - and the edges represent complicating (public) variables or constraints.

The graph can be expressed by a matrix  $E \in \mathbb{R}^{\theta \times \vartheta}$ , which describes its hyperedges in the following way ( $\theta$  is the total number of (scalar) public variables,  $\vartheta$  is the number of hyperedges):

$$E_{ij} = \begin{cases} 1 & y_i \text{ is in net } j \\ 0 & \text{otherwise.} \end{cases} \quad (5.4)$$

Here  $y_i$  is the  $i$ -th component of vector of all public variables  $y$ , which can be expressed as  $y = Ez$ , where  $z \in \mathbb{R}^{\vartheta}$  is the vector of common values of public variables on the hyperedges (vector of net variables, the number of elements of  $z$  is the same as the number of hyperedges  $\vartheta$ ). This equation represents the coupling constraints between subproblems. Using this representation of the hypergraph, the original optimization problem for  $n$  subproblems has a form

$$\begin{aligned}
& \text{minimize} && \sum_{i=1}^n f_i(x_{\text{local } i}, (y)_i) \\
& \text{subject to} && (y)_i = (E)_i z, \quad i = 1, \dots, n,
\end{aligned} \tag{5.5}$$

where  $(E)_i$  and  $(y)_i$  are parts of  $E$  and  $y$  corresponding to the subproblem  $i$ . In next subsections of this chapter we will suppose that subgradient method is used for solving master problem.

#### 5.4.1 Primal decomposition

In primal decomposition,  $\phi_i((y)_i)$  is the optimal value of subproblem

$$\min_{x_{\text{local } i}} f_i(x_{\text{local } i}, (y)_i).$$

One hyperedge (net) has one associated variable, an element of  $z$ , which is distributed to  $i$ -th subproblem by  $(y)_i = (E)_i z$ . Each subproblem optimizes its cost function using those values, which are fixed in each step (the problem is then separable). Then it computes particular gradient and sends it to the master. Master, solving the problem

$$\min_z \phi(z) = \sum_{i=1}^n \phi_i((y)_i), \tag{5.6}$$

where  $(y)_i = (E)_i z$ , sums up gradients from different subproblems and updates the public variables in order to obtain better global solution.

The process is described in Algorithm 5.1, where  $\alpha^{(k)}$  is the step size during iterates. It is important to emphasize that the algorithm is distributed - the only communication needed is between subproblems (in practical imlementations they can be called subsystems) and nets adjacent to them, there is no communication between different subsystems or different nets.

#### 5.4.2 Dual decomposition

To obtain problem dual to the original one (Problem 5.5), the Lagrangian is formed as

$$\begin{aligned}
L(x_{\text{local } i}, y, z, \nu) &= \sum_{i=1}^n f_i(x_{\text{local } i}, (y)_i) + \nu^T (y - Ez) \\
&= \sum_{i=1}^n (f_i(x_{\text{local } i}, (y)_i) + (\nu)_i^T (y)_i) - \nu^T Ez.
\end{aligned} \tag{5.7}$$

where  $\nu \in \mathbb{R}^\theta$ , vector of Lagrange multipliers, has part  $(\nu)_i$  associated with  $i$ -th subproblem,  $i = 1, \dots, n$ . To find a dual function, let's first minimize over  $z$  - this gives

**Algorithm 5.1** Primal decomposition, general form

---

Given initial net variables vector  $z^{(0)}$

Repeat ( $k$ -th iterate)

1. Distribute net variables to subproblems:  $(y)_i^{(k)} = (E)_i z^{(k)}$ ,  $i = 1, \dots, n$
  2. Optimize subproblems  $f_i(x_{\text{local } i}^{(k)}, (y)_i^{(k)})$  - find optimal  $x_{\text{local } i}^{(k)*}$  and gradient  $g_i^{(k)}((y)_i^{(k)*})$
  3. Master sums up subgradients over each net:  $g^{(k)} = \sum_{i=1}^n (E)_i^T g_i^{(k)}$
  4. Master updates vector of net variables:  $z^{(k+1)} = z^{(k)} - \alpha^{(k)} g^{(k)}$
- 

the condition  $E^T \nu = 0$ , which means that the sum of multipliers over each net should be zero. Then the dual function is

$$q(\nu) = \sum_{i=1}^n q_i((\nu)_i),$$

where  $q_i((\nu)_i)$  is optimal value of the subproblem  $i$ :

$$\min_{x_{\text{local } i}, (y)_i} f_i(x_{\text{local } i}, (y)_i) + (\nu)_i^T (y)_i.$$

The subgradient of  $q_i((\nu)_i)$  at  $(\nu)_i$  is  $(y)_i$ , which is a part of the “global” subgradient  $y$ . This  $y$  will be used in next explanation for the sake of simplicity, but the distributive nature of the algorithm will be still preserved.

The dual problem has a form

$$\begin{aligned} & \text{maximize} && q(\nu) \\ & \text{subject to} && E^T \nu = 0. \end{aligned} \tag{5.8}$$

This problem can be solved by projected subgradient method, which was explained in Section 4.2. The projection into feasible set  $E^T \nu = 0$  is affine operator, so the Equation 4.6 can be used to compute the update of optimization variable

$$\nu^{(k+1)} = \nu^{(k)} + \alpha^{(k)} (I_\theta - E(E^T E)^{-1} E^T) y. \tag{5.9}$$

The update can be interpreted in the following way - the average values of public variables over each net  $\hat{z} = (E^T E)^{-1} E^T y$  are computed and then they are subtracted from corresponding public variable values,  $g = y - E\hat{z}$ , to form a projected subgradient - together it means that  $g = (I_\theta - E(E^T E)^{-1} E^T) y$ , where  $I_\theta$  is unit matrix of order  $\theta$ .

To summarize the algorithm from practical point of view, in dual decomposition each

**Algorithm 5.2** Dual decomposition, general form

Given initial price vector  $\nu^{(0)}$ , such that  $E^T \nu^{(0)} = 0$  (the sum of Lagrange multipliers is zero over each net)

Repeat ( $k$ -th iterate)

1. Optimize subproblems to obtain  $x_{\text{local } i}^{(k)*}$  and  $(y)_i^{(k)*}$
2. Master computes average value of public variable at each net:  
 $\hat{z}^{(k)} = (E^T E)^{-1} E^T y^{(k)*}$
3. Master updates prices of public variables:  $\nu^{(k+1)} = \nu^{(k)} + \alpha^{(k)}(y^{(k)} - E\hat{z}^{(k)})$

subproblem (subsystem) has its own copy of the public variables as well as Lagrange multipliers subvector (price vector). After the optimizations in subsystems the value of the public variables are compared and multipliers (prices) are updated. The goal is to obtain the consistency between local copies of the public variables. The procedure is presented in Algorithm 5.2, where  $\alpha^{(k)}$  is the step size during iterates. Again it is important to emphasize the distributive nature of the algorithm - the subsystems interact only with adjacent nets, there is no interaction between different subsystems or different nets.

## 5.5 Example

### 5.5.1 Problem formulation

Let's consider the problem (first formulated in standard optimization notation, the graph representation will follow)

$$\begin{aligned}
 &\text{minimize} && f_0 = f_1(x_1, y_1) + f_2(x_2, y_2, y_3) + f_3(x_3, y_4) \\
 &\text{subject to} && y_1 = y_2 \\
 &&& y_3 = y_4,
 \end{aligned} \tag{5.10}$$

where

$$\begin{aligned}
 f_1 &= [x_1 \quad y_1] \begin{bmatrix} 9 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} + [5 \quad 3] \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \\
 f_2 &= [x_2 \quad y_2 \quad y_3] \begin{bmatrix} 4 & 1 & 2 \\ 1 & 3 & 1 \\ 2 & 1 & 3 \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ y_3 \end{bmatrix} + [-2 \quad -4 \quad -2] \begin{bmatrix} x_2 \\ y_2 \\ y_3 \end{bmatrix}
 \end{aligned}$$

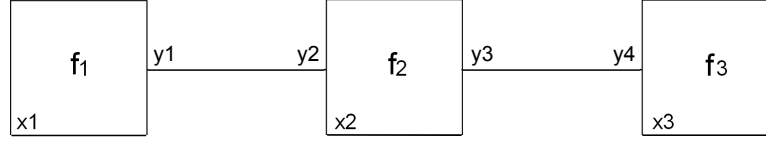


Figure 5.1: Graph representation of the Problem 5.10

and

$$f_3 = \begin{bmatrix} x_3 & y_3 \end{bmatrix} \begin{bmatrix} 4 & 1 \\ 1 & 5 \end{bmatrix} \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} + \begin{bmatrix} 1 & -2 \end{bmatrix} \begin{bmatrix} x_3 \\ y_3 \end{bmatrix}.$$

So there are three subproblems (subsystems) coupled by two complicating variables - the situation is depicted on Figure 5.1.

The graph is formed by three nodes and two edges. Matrix  $E$  describing the graph is

$$E = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix},$$

parts of public variables vector  $y$  corresponding to subproblems are  $(y)_1 = y_1$ ,  $(y)_2 = [y_2 \ y_3]^T$ ,  $(y)_3 = y_4$  and similarly the graph matrix can be split into

$$\begin{aligned} (E)_1 &= \begin{bmatrix} 1 & 0 \end{bmatrix}, \\ (E)_2 &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \\ (E)_3 &= \begin{bmatrix} 0 & 1 \end{bmatrix}. \end{aligned}$$

Slater's condition (see Section 3.3) is satisfied, since there are no inequality constraints and the problem is convex - so the strong duality holds.

### 5.5.2 Solution using primal decomposition

In primal decomposition, the implemented algorithm found the optimal value of

$$f_0 = -3.0163 \tag{5.11}$$

for

$$\begin{aligned}
 x_1 &= -0.6269 \\
 x_2 &= 0.1909 \\
 x_3 &= -0.3644 \\
 y_1 = y_2 &= 0.3210 \\
 y_3 = y_4 &= 0.4577.
 \end{aligned} \tag{5.12}$$

The values of  $f_0$  during iterations are shown in the Figure 5.2a, the absolute value of difference between  $f_0(x^{(k)})$  and optimal value  $p^*$  in Figure 5.2b. In this first example we use only constant step size rule and present the influence of different values of  $\alpha$  used by master algorithm. It is obvious that the value of step size influences the number of iterates needed to obtain the solution with particular accuracy (tolerance). The fastest convergence was obtained for  $\alpha = 0.18$ , where the error is reduced from about  $10^2$  to  $10^{-15}$  in 14 iterations, so it was reduced by factor  $10^{-17/14} \doteq 0.06$  each iteration.

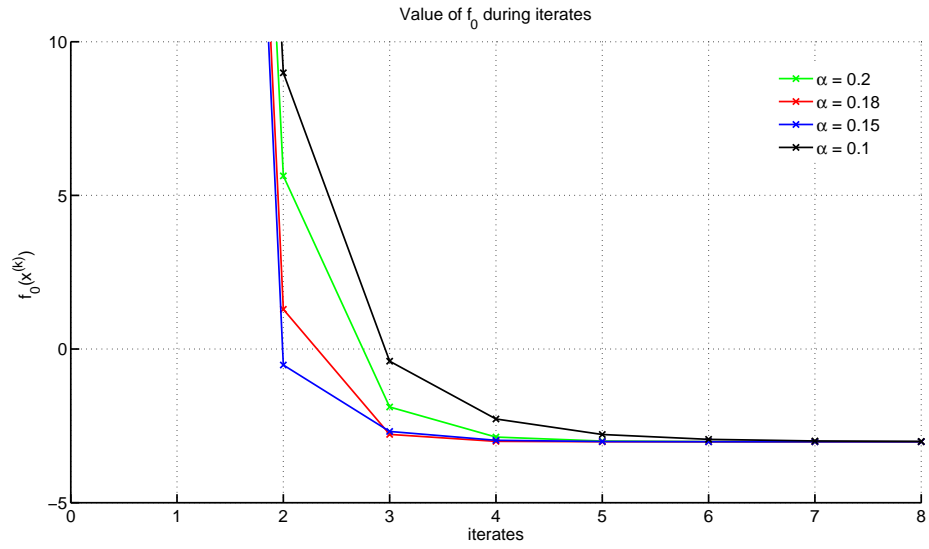
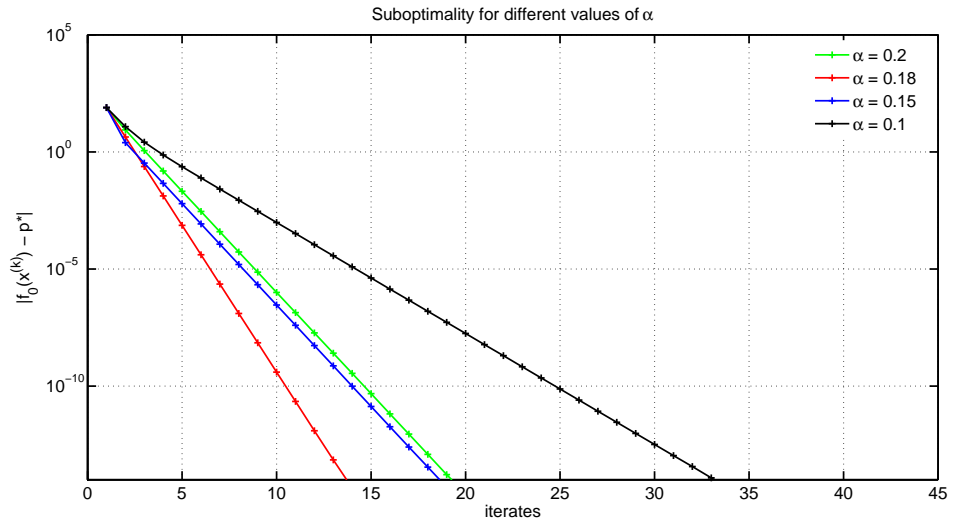
### 5.5.3 Solution using dual decomposition

In dual decomposition the initial price vector was chosen  $\nu = 0$ . The optimal value and values of variables found are exactly the same as in the case of primal decomposition (see 5.11 and 5.12), the prices vector converged to

$$\nu = \begin{bmatrix} -2.3883 \\ 2.3883 \\ -0.0759 \\ 0.0759 \end{bmatrix},$$

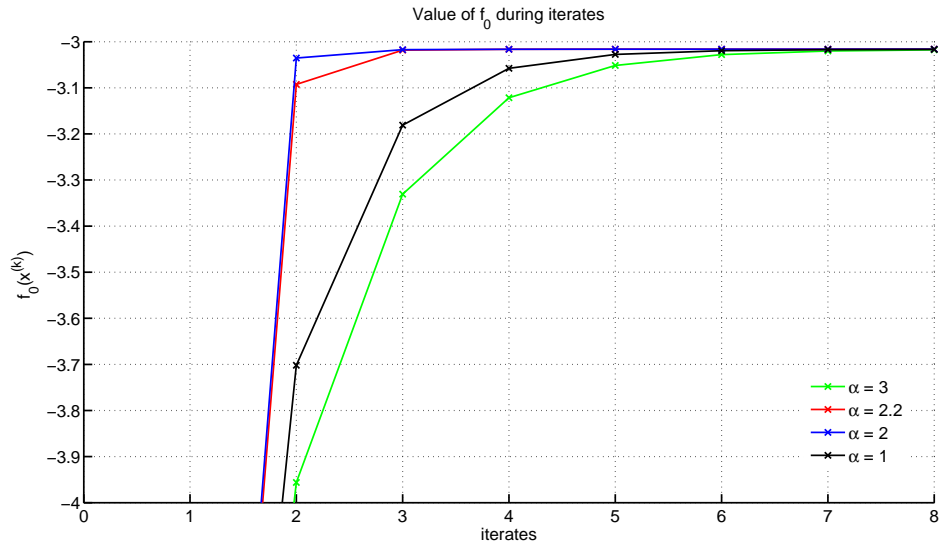
which satisfies the condition  $E^T \nu = 0$ .

As in the case of primal decomposition, the values of  $f_0(x^{(k)})$  during iterations for the different values of constant step size  $\alpha$  are shown in the Figure 5.3a and the absolute value of difference to the optimal value in the Figure 5.3b. It is worth noting that the dual problem solution gives lower bound of the original optimization problem solution, what is in accordance with the theoretical explanation. The fastest convergence occurs for  $\alpha = 2.2$ , where the error was reduced by factor  $10^{-15/11} \doteq 0.04$  each iteration.

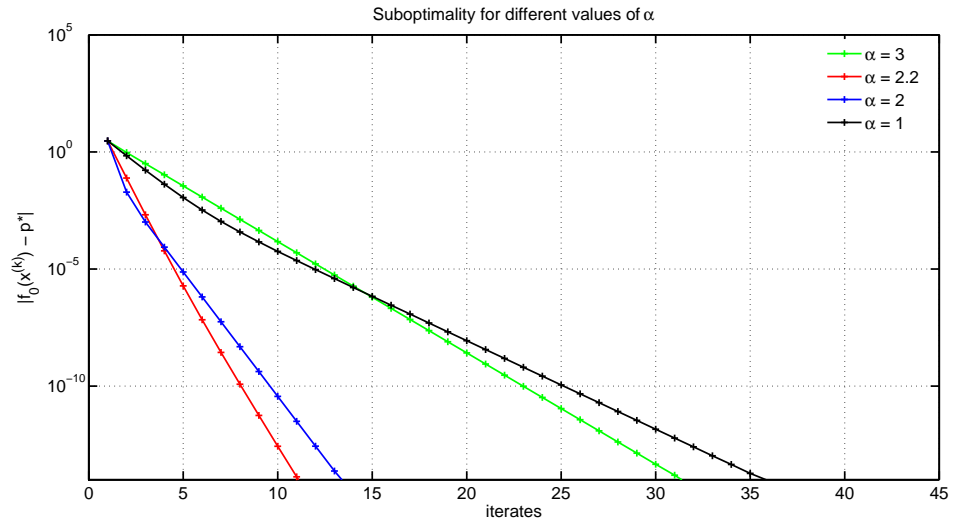
(a) Values of  $f_0$  during iterates

(b) Suboptimality during iterates

Figure 5.2: Primal decomposition



(a) Values of  $f_0$  during iterates



(b) Suboptimality during iterates

Figure 5.3: Dual decomposition



## Chapter 6

# Applications to engineering problems

This chapter forms the main implementation part of the thesis. Here we will apply the methods described in previous chapters to two selected engineering problems. The first one is dynamic - distributed MPC (described in Section 6.1), while the second one is static - industrial energy network optimization (described in Section 6.2).

### 6.1 Dynamic control - MPC

In this section the dual decomposition method for control of three interconnected dynamic subsystems will be presented. After the problem formulation, the algorithm described in Subsection 5.4.2 will be used to obtain the solution of master problem. Then this solution will be compared to the solution of centralized approach, while different deployment options of the decentralized algorithm will be examined as well. In addition, different step size rules used to obtain the solution will be analysed and convergence of those step size rules will be compared.

#### 6.1.1 Problem formulation

The inputs, outputs and interconnections of subsystems are depicted on Figure 6.1. The dynamics of the subsystems is shown in the Figures 6.2, 6.3 and 6.4 (subsystems were discretized with sampling period  $T = 0.1$  s). Each subsystem has one manipulated input and one controlled output (the reference signal  $y_{\text{ref}}$  for that output is given as a unit step at time step 20 for all subsystems).

For each subsystem we use MPC controller with following parameters: prediction horizon  $N = 50$  steps, the tracking weight  $q(i) = 1, i = k, \dots, k + N - 1$  (for all  $k$  for tracked outputs), the input weight  $r(i) = 1, i = k, \dots, k + N - 1$  (for all  $k$  for controlled inputs), with no constraints on inputs, no soft constraint on the output

and no blocking. The constraints are omitted, because for the demonstration of the decomposition method they lack significance, but it would be easy to deal with them. The blocking is not used for easier comparison to centralized approach.

We want to find the solution of optimization problem

$$\begin{aligned}
& \text{minimize} && J^I((u^I)_k^{k+N-1}) + J^{II}((u^{II})_k^{k+N-1}) + J^{III}((u^{III})_k^{k+N-1}) \\
& \text{subject to} && (y_1^I)_k^{k+N-1} = (u_1^{II})_k^{k+N-1} \\
& && (y_2^I)_k^{k+N-1} = (u_1^{III})_k^{k+N-1} \\
& && (y_1^{III})_k^{k+N-1} = (u_1^I)_k^{k+N-1} \\
& && x(k+1)^M = A^M x(k)^M + B^M u(k)^M \\
& && y(k)^M = C^M x(k)^M + D^M u(k)^M
\end{aligned} \tag{6.1}$$

for all  $k$ , where the superscripts indicate the subsystems ( $I$ .,  $II$ ., or  $III$ .), while  $M$  takes all values  $I$ .,  $II$ ., and  $III$ .),  $u^M = [u_1^M \ u_2^M]^T$  and

$$\begin{aligned}
J^M &= \frac{1}{2} \{ [(y_{\text{ref}}^M)_k^{k+N-1} - (y^M)_k^{k+N-1}]^T Q^M [(y_{\text{ref}}^M)_k^{k+N-1} - (y^M)_k^{k+N-1}] + \dots \\
&\dots + [(\Delta u^M)_k^{k+N-1}]^T R^M [(\Delta u^M)_k^{k+N-1}] \},
\end{aligned}$$

where matrices  $Q^M$  and  $R^M$  are formed with respect to weights  $q(i)$ ,  $r(i)$  and tracked outputs and controlled inputs, for example diagonal matrix

$$\begin{aligned}
Q^I &\in \mathbb{R}^{3N \times 3N} = \mathbb{R}^{150 \times 150} \\
Q^I &= \begin{bmatrix} 0 & & & & & \\ & 0 & & & & \\ & & 1 & & & \\ & & & \ddots & & \\ & & & & 0 & \\ & & & & & 0 \\ & & & & & & 1 \end{bmatrix},
\end{aligned}$$

because we track only the last output from three outputs of subsystem  $I$ .

To rephrase the problem formulation, we are searching for the minimum of additive cost function of three controllers, that are controlling three interconnected subsystems. Those interconnections represent complicating constraints, which have to be met in order to obtain physically realizable solution. It is important to keep in mind that this solution is the solution of one (!) sampling period of MPC (as is depicted in the Figure 2.2), without implementing the receding horizon (this would need to apply only the first input of the solution and compute solution of new problem in next step).

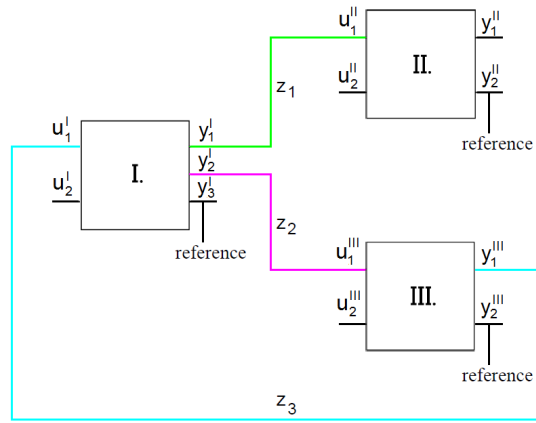


Figure 6.1: MPC example situation

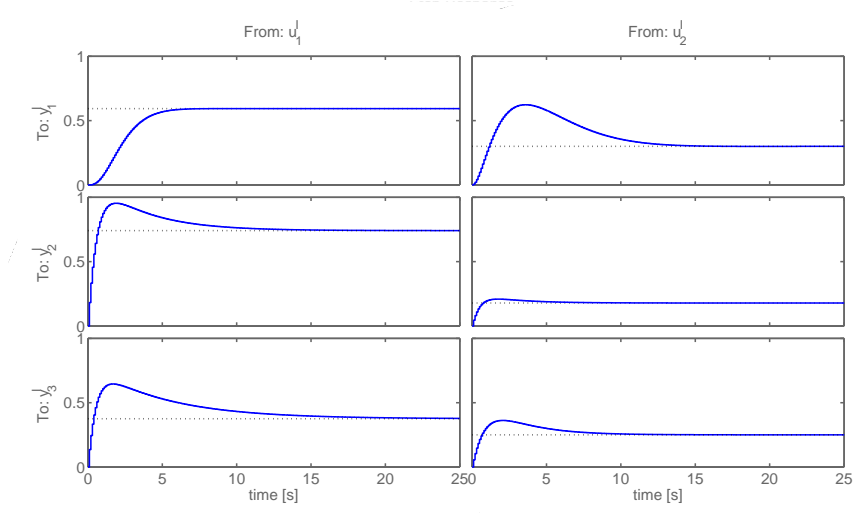


Figure 6.2: Unit step response of subsystem I.

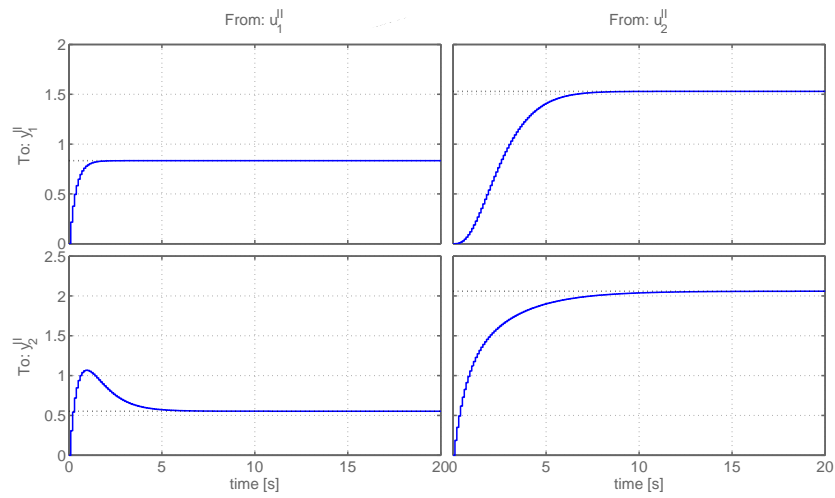


Figure 6.3: Unit step response of subsystem II.

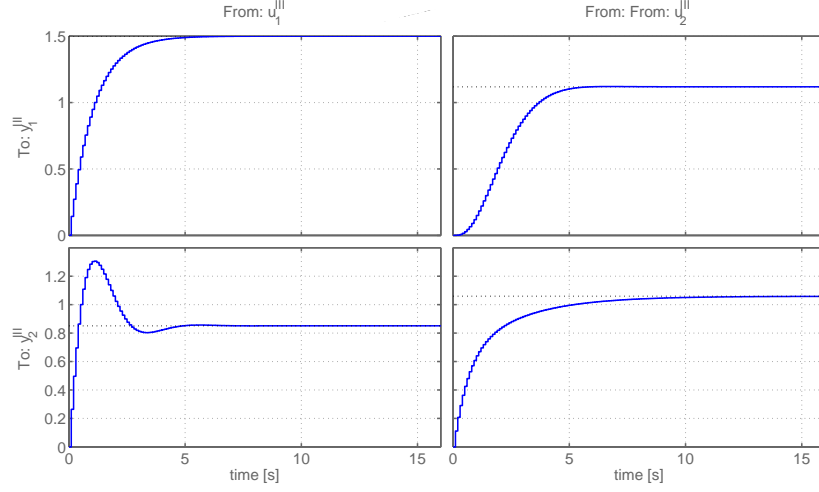


Figure 6.4: Unit step response of subsystem III.

### 6.1.2 Distributed algorithm and its solution

Algorithm 5.2 was implemented in order to get solution of the optimization Problem 6.1. The hypergraph is formed by three nodes and three edges and is described by matrix

$$E = \begin{bmatrix} I_N & 0 & 0 \\ I_N & 0 & 0 \\ 0 & I_N & 0 \\ 0 & I_N & 0 \\ 0 & 0 & I_N \\ 0 & 0 & I_N \end{bmatrix},$$

where  $I_N$  is the unit matrix of order  $N$  (in this particular case  $N = 50$ ). The step was chosen constant with value  $\alpha = 0.00395$  (we will focus on various step size rules in Subsection 6.1.5) and initial price vector was set to a zero vector.

The situation at the beginning of the algorithm is shown in the Figure 6.5, where the colors (same as in Figure 6.1) represent pairs of signals which connect subsystems (except of blue color representing reference signals and red color showing non-connecting signals). In the beginning of the algorithm each MPC solves its particular optimization problem, without any information about the interconnections (we can say that the complicating constraints are neglected). Those subsystems' solutions are not realizable, because the interconnections imply the need for consistency of interconnecting signals, which is not satisfied.

During the algorithm (see Subsection 5.4.2 for rigorous explanation) the subsystems receive updates of Lagrange multipliers of shared signals (in other words “prices” of

public variables, which are stored locally in subsystems) from master algorithm and they change their optimal solutions accordingly. Subsystems return their subgradients to master, who computes next update of Lagrange multipliers in order to obtain the consistency between shared signals. For measuring the inconsistency of signals during iterates we will use a norm  $\|y - Ez\|_2$  (the meaning of this term should be clear after reading the Chapter 5). This norm can be used as a stopping criterion, together with limited number of iterations.

After 1020 iterations the algorithm reaches the value of the norm of inconsistency  $\|y - Ez\|_2 = 0.01$ , which ensures the reasonable precision of the solution - the results for all three subsystems are depicted in the Figure 6.6. There the consistency between interconnecting signals can be easily verified. In next subsection we compare this solution to the solution obtained by the centralized approach.

### 6.1.3 Comparison to centralized approach

As we can see from the Figures 6.7 and 6.8, which show the situation during the iterations (centralized solution in red, distributed solution in green, reference signals in blue), the solution obtained by the distributed approach converges to the centralized solution. The constant step size with  $\alpha = 0.00395$  was used, with no blocking (for details of the solution see previous subsection). For the illustration the number of iterates was chosen as  $k \in \{1, 250, 500, 1020\}$ , which corresponds to the values of inconsistency  $\|y - Ez\|_2 \in \{6.850, 0.316, 0.088, 0.010\}$  (in the same order).

It would be beyond the scope of this thesis to provide some quantitative analyses, which could be for example the memory requirements comparison or communication demands. On the other hand in next subsection we will provide qualitative description of the possible practical implementation and deployment of the algorithm used, to make an overview of the distributed approach and to emphasize its benefits.

### 6.1.4 Illustration of possible deployment

In the Subsections 5.4.1 and 5.4.2 we pointed out the distributive nature of the algorithms presented. Now at this particular example we can demonstrate it more specifically. In the case of “ordinary” step size rules (mostly basic step size rules, we will specify this term at the end of this subsection) there is no need for master algorithm to be situated at one location, as “central” master algorithm collecting information from all nets, see red circle on a Figure 6.9. It could be distributed on the nets, as is depicted on the Figure 6.10, or situated in the selected node(s) - the situation on the Figure 6.11. On the last figure the master could be spread into all nodes, coordinating the particular nets.

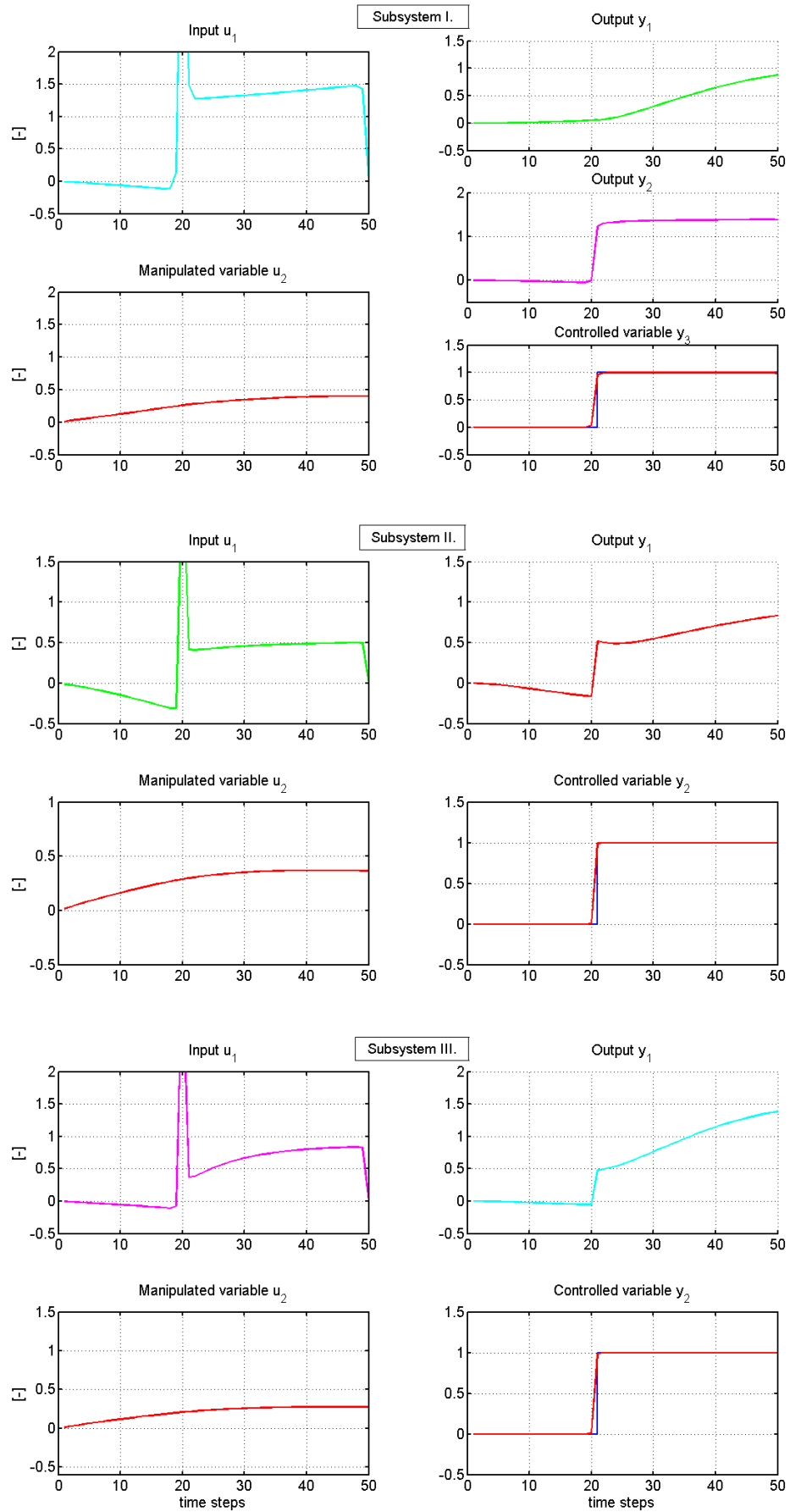
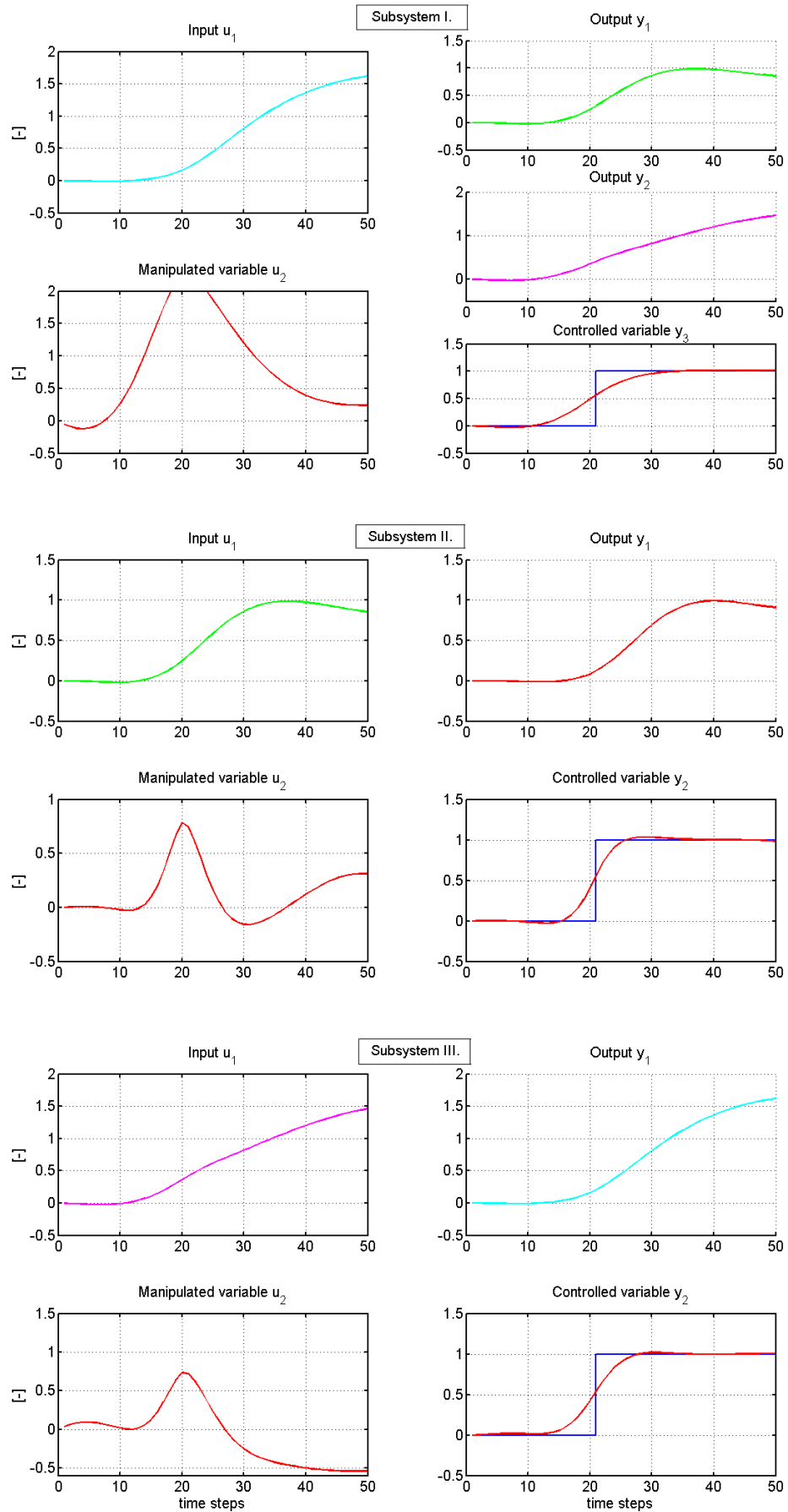
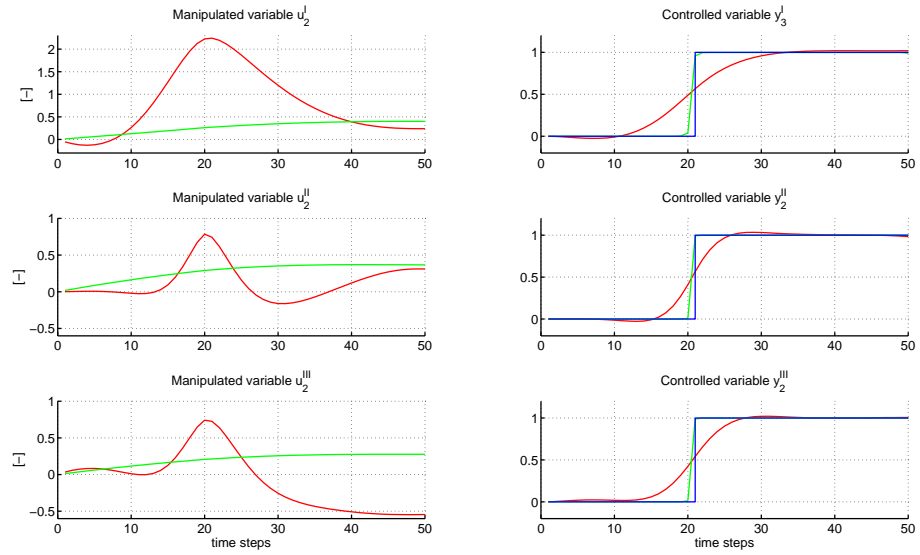
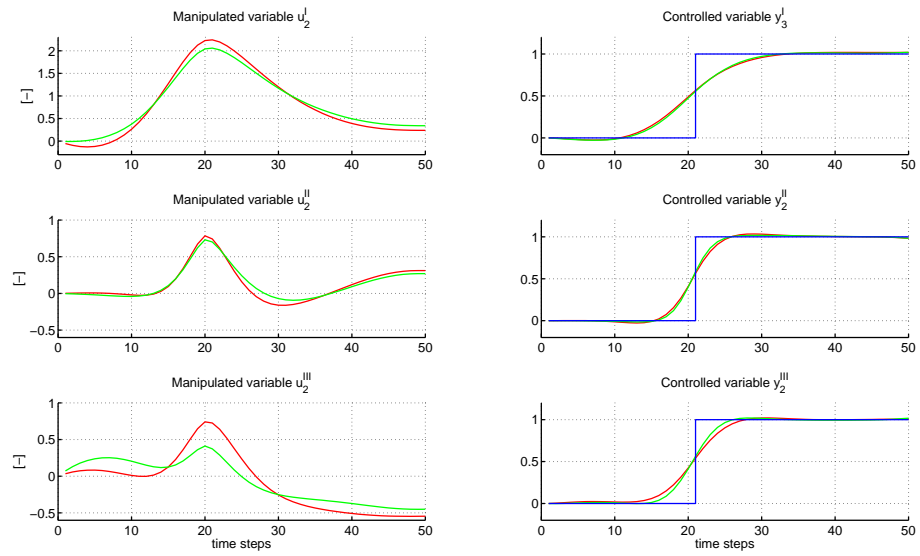


Figure 6.5: Subsystems' inputs and outputs, first iterate

Figure 6.6: Subsystems' inputs and outputs, 1020<sup>th</sup> iterate



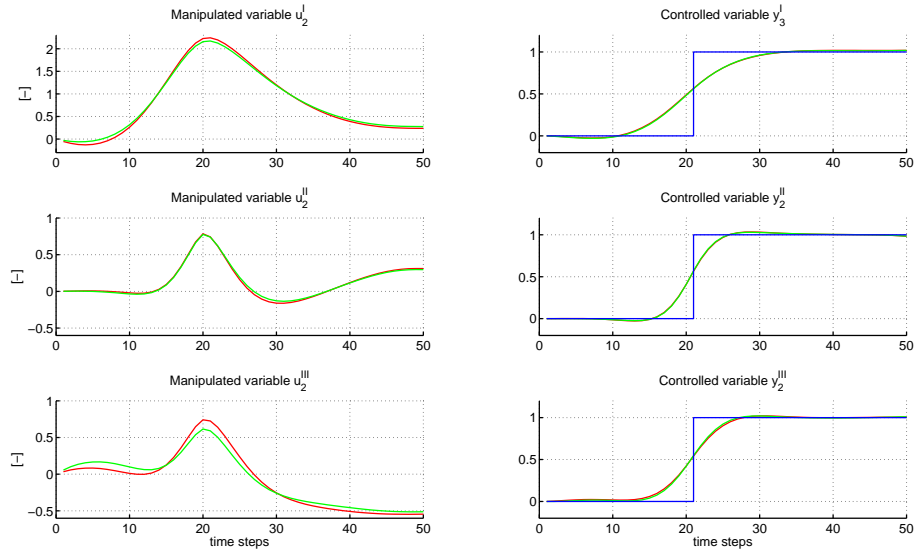
(a) First iterate



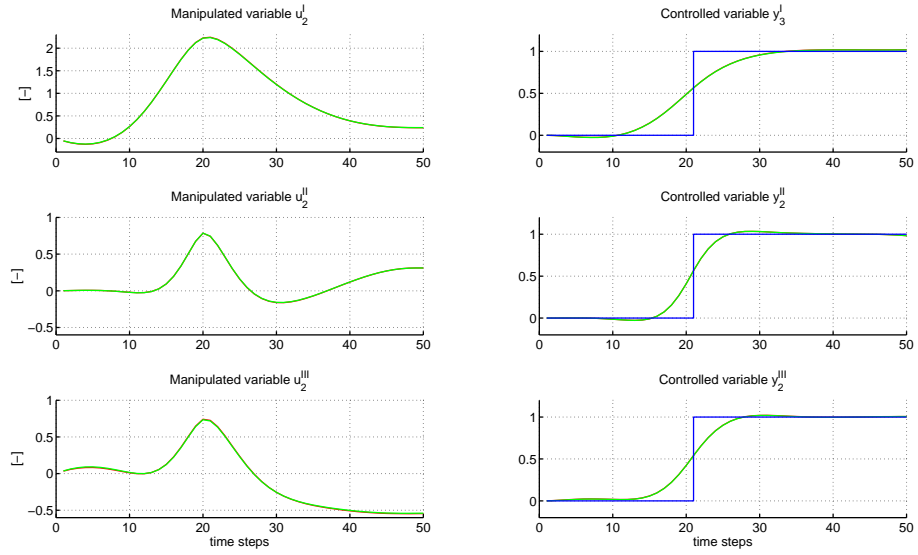
(b) 250<sup>th</sup> iterate

Figure 6.7: Centralized (red) and distributed (green) solution (part 1)





(a) 500<sup>th</sup> iterate



(b) 1020<sup>th</sup> iterate

Figure 6.8: Centralized (red) and distributed (green) solution (part 2)

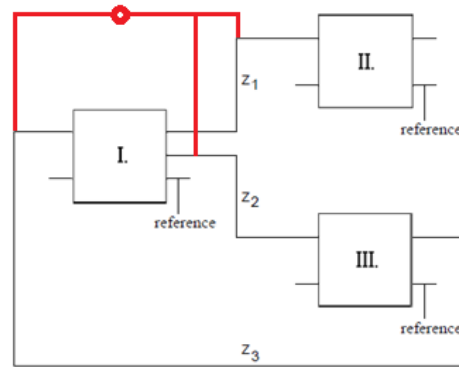


Figure 6.9: Central master algorithm

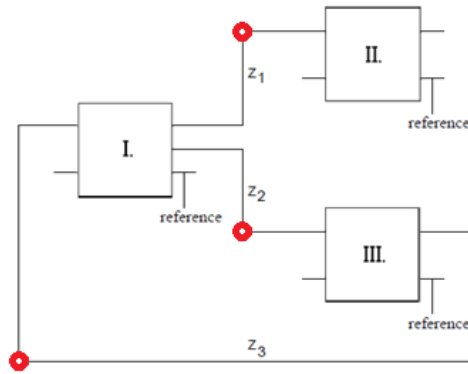


Figure 6.10: Master algorithm distributed to nets

One of the main advantages of these approaches is modularity - when a new subsystem is added into the network, then in the case of central master algorithm only the matrix  $E$  has to be updated (and new Lagrange's multipliers have to be added). In case of master distributed to nets, master is extended to the new net, and similarly in case of master distributed to nodes it is necessary to decide, which node will play role of master for the new net. The latter case is the most interesting one, so we describe it with other two figures - on Figure 6.12 shows the situation when it is still possible to coordinate the nets from one node, but on the Figure 6.13 it is necessary to add new part of master algorithm to the node  $II$ . or  $V$ . (from the figure it is obvious that node  $V$ . was chosen).

By "ordinary" step size rules we meant the rules which are not using function values to determine the sequence  $\alpha^{(k)}$ , that means all the step size rules that were mentioned except Polyak's rule and original Nesterov's rule. The reason is that when the value of  $\alpha^{(k)}$  is determined by function values (see Equations 4.16 or 4.23) the master algo-

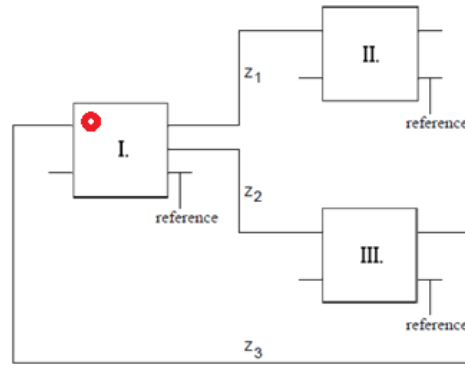


Figure 6.11: Master algorithm distributed to node(s)

rithm needs the information from all subsystems - thus it is not possible to make the computation fully distributed with the approach we presented and it is necessary to use centralized master algorithm (as in the Figure 6.9). Possible solution could bring incremental subgradient methods presented in [NEDI02], but this is out of scope of this master thesis.

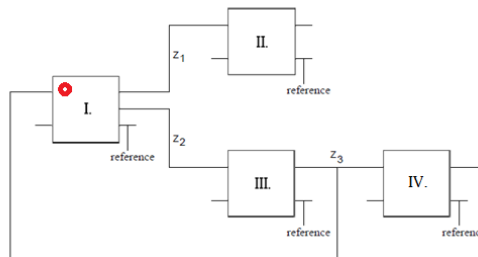


Figure 6.12: Extension of the network (one subsystem added)

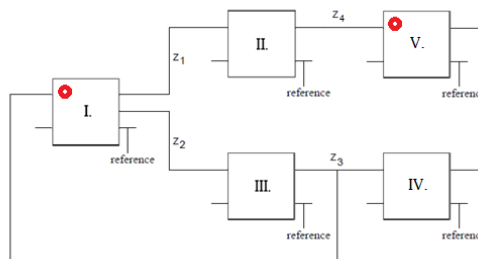


Figure 6.13: Extension of the network (two subsystems added)

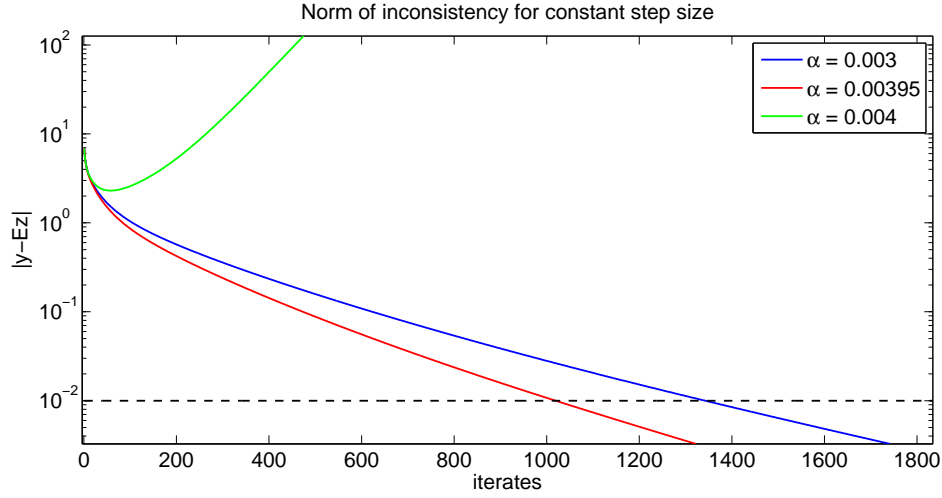


Figure 6.14: Constant step size convergence - basic comparison

### 6.1.5 Analysis of different step size rules

In this subsection we will analyse separately different step size rules used for solving the problem described in 6.1.1. That means that the dependence of convergence of each step size rule on its parameters will be shown. We will use the same measure of inconsistency between interconnecting signals,  $\|y - Ez\|_2$ , during the iterates. As a stopping criterion the value of this term was chosen (in particular  $\|y - Ez\|_2 = 0.01$ ) together with the reasonable number of iterates (maximum number of iterates was set to 12000). For the comparison of different step size rules we will use the number of iterations needed to achieve the value  $\|y - Ez\|_2 = 0.01$ . The graphs will be plotted as continuous for simplicity and the vertical axis on graphs will be in logarithmic scale.

Constant step size convergences for different values of parameter  $\alpha$  are shown in the Figure 6.14. We can see that the bigger the step, the faster is the convergence. But for one particular value of  $\alpha$  the algorithm begins to be unstable and it diverges. This value is specific for each optimization problem, in our case the critical value of  $\alpha$  is in interval  $\alpha_{\text{crit}} \in (0.00395, 0.004)$ . It is also important to remember that this step size rule gives only a suboptimal solution (see 4.3.1) - the situation is depicted in the Figure 6.15, where we can easily see the trade-off, when the lower the value of  $\alpha$  makes the convergence slower, but the algorithm converges to smaller neighborhood of optimal solution. In practical situations we probably won't need such accuracy, we only want to emphasize that the observation corresponds to the theoretical conclusion. For the mutual comparison of all different step size rules, which will be provided in Subsection 6.1.6, let's choose the value  $\alpha = 0.00395$ , which makes the algorithm to reach the norm  $\|y - Ez\|_2 = 0.01$  in 1020 iterations.

Figure 6.16 shows the convergence of constant step length rule for different values

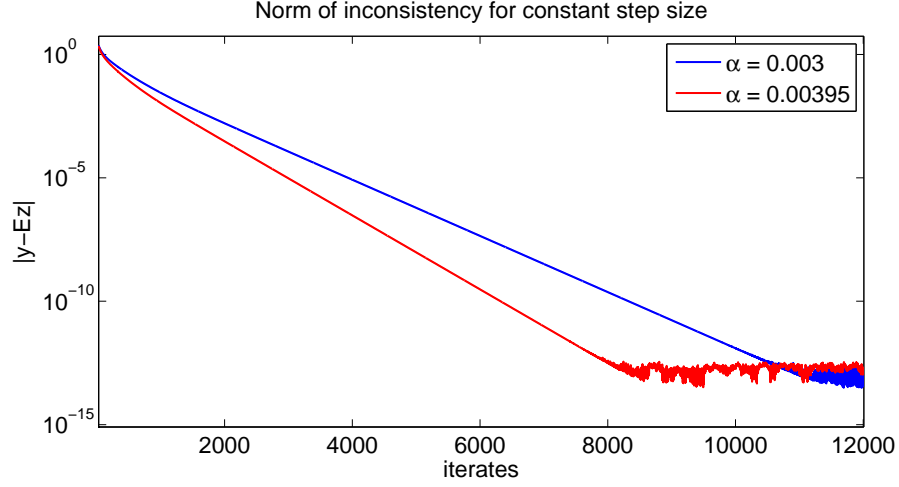


Figure 6.15: Constant step size convergence - suboptimality of the solution

of  $\gamma$  (distance between the successive points). We can see that the method provides suboptimal solution - in 4.3.2 it was stated that it converges to within  $G\gamma/2$  to optimal, where  $G$  is the upper bound of subgradients. Our observation corresponds to that, because for example for value  $\gamma = 0.0001$  the measure of inconsistency stops decreasing in 8485<sup>th</sup> iterate (highlighted in the figure), when the subgradient oscillations begin to have major impact on it. The oscillations can be identified in Figure 6.17a, that depicts the behavior of selected components of subgradient during all iterates. There the feature of the projected subgradient method can be observed - the sum of the pair of corresponding components of subgradient gives zero (the pairs are in colors of appropriate connections in Figure 6.1). We can also notice the slope change of the curves around the iterate number 5300 (precisely at 5286), which has an impact on the behavior of the norm of inconsistency for  $\gamma = 0.0001$ . Figure 6.17b presents the detailed view, showing that the subgradient is oscillating even before 8485<sup>th</sup> iterate.

From the Figure 6.16 it is obvious that the higher value of the parameter  $\gamma$  ensures faster convergence in the beginning, but then the solution begins to oscillate in a bigger neighborhood around the optimal solution. Thus in practical implementations the choice of the parameter should depend on desired size of neighborhood of optimal value, what the algorithm should reach. We can also see that during 12000 iterations no value of the parameter reached the desired threshold  $\|y - Ez\|_2 = 0.01$ . For the comparison of different step size rules we choose the value of parameter  $\gamma = 0.0001$ , which represents “the best” trade-off (by engineering guess) between fast convergence in the beginning and small neighborhood of the optimum.

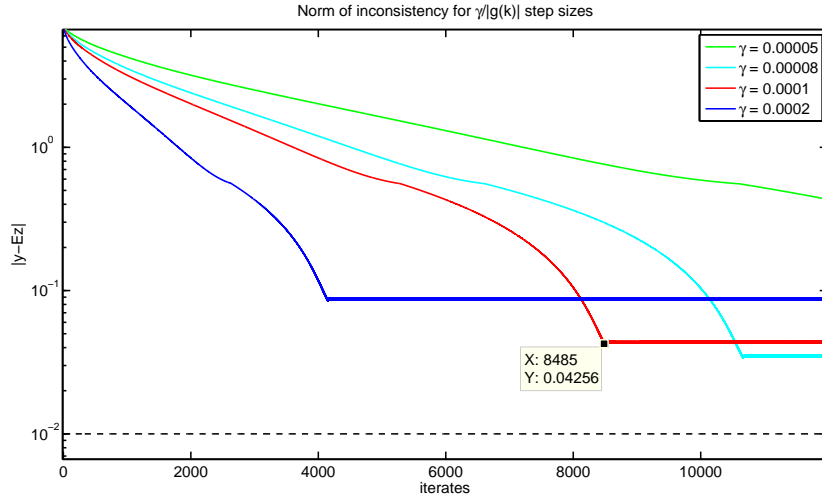
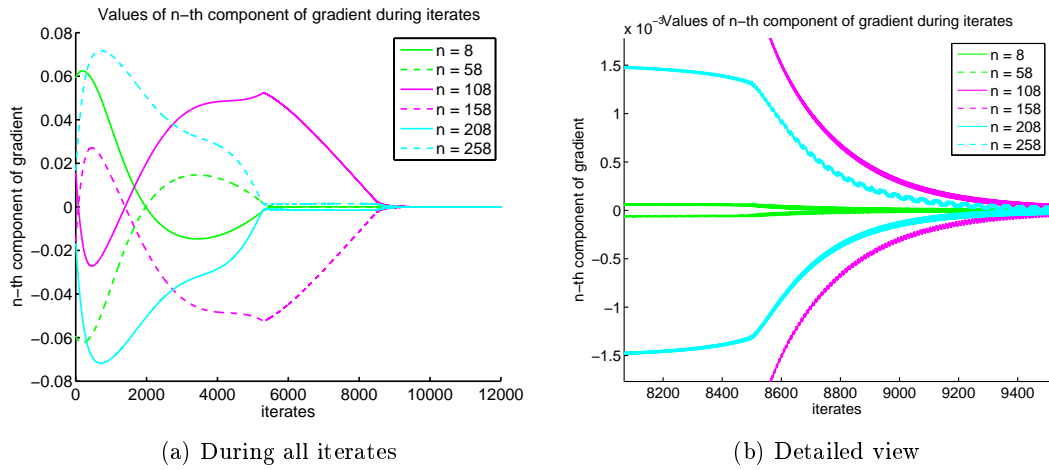


Figure 6.16: Constant step length convergence

Figure 6.17: Selected components of gradient for  $\gamma = 0.001$

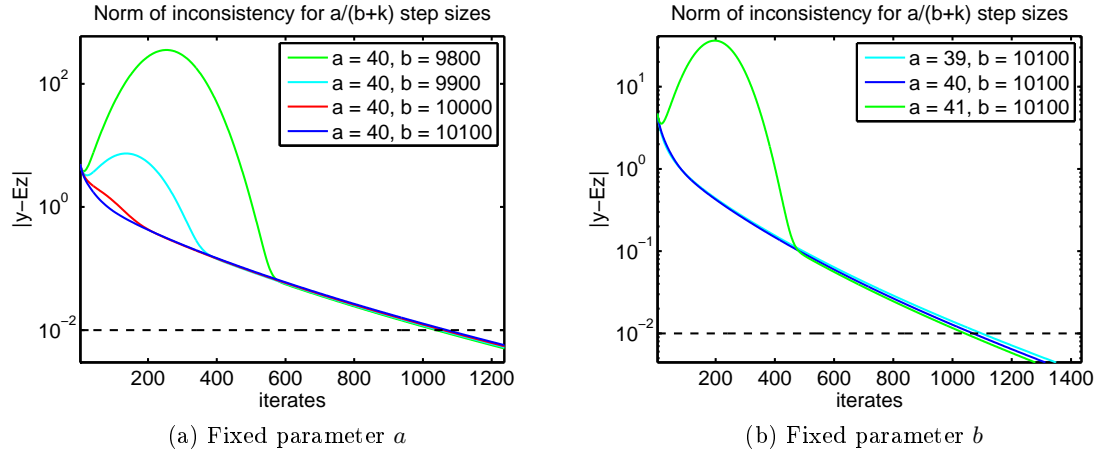


Figure 6.18: Square summable, but not summable step size convergence

Figure 6.18 presents different convergences for square summable, but not summable step size rule. The sequence  $\alpha^{(k)} = a/(b+k)$ , where  $k$  is the iteration number and  $a, b$  are parameters, was chosen for the analysis. In that case we can observe various situations, which are corresponding to the properties of the sequence  $\alpha^{(k)} = a/(b+k)$ . First, the value of parameter  $a$  is fixed to the value  $a = 40$  and the behavior for different values of  $b$  is investigated in the Figure 6.18a, in particular for the values of  $b \in \{9800, 9900, 10000, 10100\}$ . For the lower value of  $b$  there is a higher peak of inconsistency at the beginning of the iterative process - this could be explained by relatively large values of steps  $\alpha^{(k)}$  in the beginning, which make the method almost to diverge, but then the values of  $\alpha^{(k)}$  are decreased when the number of iteration is higher. After this effect disappears, the convergence is slightly better for lower value of  $b$ . When the value of  $b$  is fixed (to the value  $b = 10000$ ), the situation for different values of  $a$  ( $a \in \{39, 40, 41\}$ ) is depicted in the Figure 6.18b. There similar effect can be observed - the difference is that the peak is higher for the higher values of  $a$ , because the parameter is in nominator. Last observation is for ratio  $a/b$  fixed to the value  $a/b = 0.004$ . We can change the absolute value of the nominator and denominator keeping the ratio constant. This adjusts the influence of number of iterations  $k$  in the equation for  $\alpha^{(k)}$ . In the Figure 6.19 different situations for pairs satisfying this condition are demonstrated - in particular for  $b \in \{10^2, 10^3, 10^4, 10^5\}$  and  $a = 0.004 \cdot b$ . For the first case the convergence is slower in comparison with other situations, because the influence of number of iterations is big and makes the values of  $\alpha^{(k)}$  converge to zero rapidly. For higher values of  $b$  the peak effect can be observed again, because higher  $b$  (and thus  $a$ ) means more delay for the influence of number of iteration. For the later comparison of different step size rule let's choose the combination of parameters  $a = 40, b = 10000$ , which reaches the threshold  $\|y - Ez\|_2 = 0.01$  in 1059 iterations.

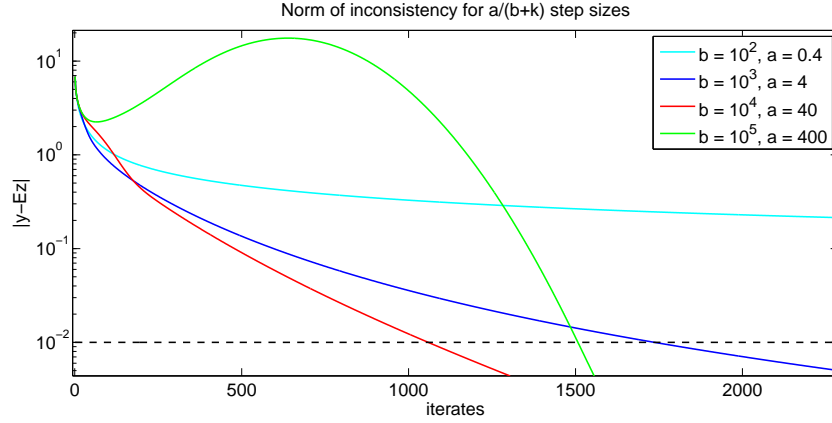
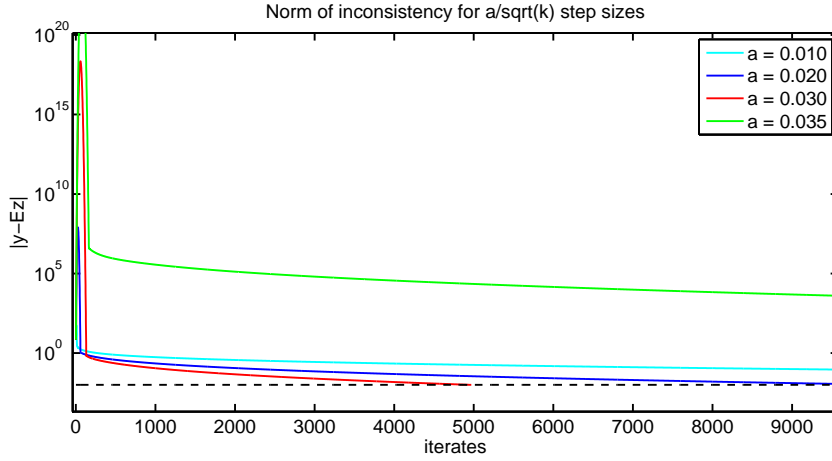
Figure 6.19: Square summable, but not summable step size convergence - fixed ratio  $\frac{a}{b}$ 

Figure 6.20: Non-summable diminishing step size convergence

Convergences for the non-summable diminishing step sizes are depicted in Figure 6.20. As we know from 4.3.4, the method converges to the optimal value. The sequence  $\alpha^{(k)} = a/\sqrt{k}$  was chosen, where  $k$  is iteration number and  $a$  is the parameter. In the figure convergences for the values of  $a \in \{0.01, 0.02, 0.03, 0.035\}$  are shown. We can identify similar effect as in the previous case - for higher value of the parameter  $a$  there is higher peak in the beginning of the algorithm (again the reason for this are the properties of the sequence). It also has an impact on the number of iterates needed for reaching the threshold, but not necessarily the higher value of  $a$  means faster convergence - we can see it on the example of the parameter value  $a = 0.035$ , which doesn't reach the threshold in a reasonable number of iterations. For the comparison of different step size rules let's choose the value of parameter  $a = 0.03$ , which makes the algorithm to converge to desired value of inconsistency in 4969 iterations.

Next figure of this subsection, 6.21, shows the convergence of the non-summable diminishing step length rule. This rule is defined in 4.3.5, the sequence of  $\gamma^{(k)} = a/\sqrt{k}$



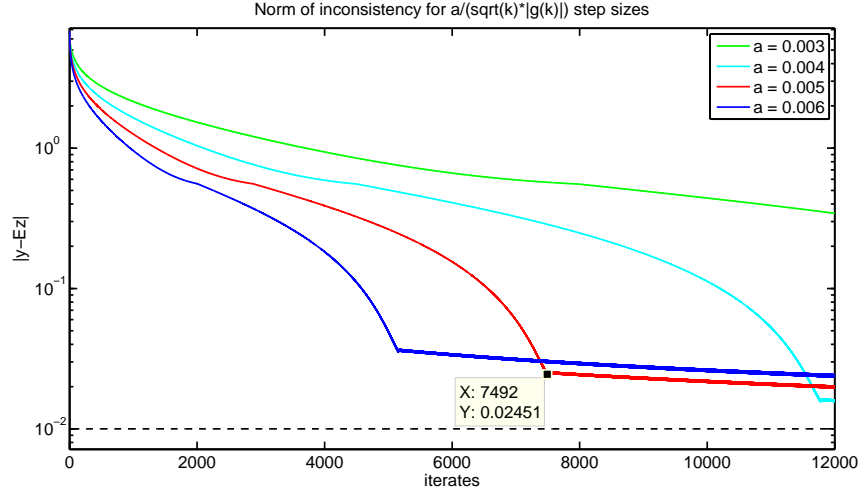


Figure 6.21: Non-summable diminishing step length convergence

was used. The situation is similar to the constant step length rule (Figure 6.16) - the lower value of the parameter  $a$  means slower convergence in the beginning, but the algorithm can reach the smaller neighborhood of the optimum before it begins to oscillate. Then when the diminishing step length converge to the optimal value (we know from 4.3.5 that this method converges to the optimum), the lower parameter ensures faster convergence. Thus in practical implementations the choice of the parameter should depend on desired size of neighborhood of optimal value and possible limit of number of iterations. From Figure 6.21 it is also obvious that during 12000 iterations no value of the parameter reached the desired threshold  $\|y - Ez\|_2 = 0.01$ . For the comparison of different step size rules we choose the value of parameter  $a = 0.005$ , which represents the compromise between fast convergence in the beginning and small neighborhood of the optimum (the point where the algorithm changes its behavior for this particular case, at iteration number 7492, is highlighted in the figure).

Figure 6.22 presents the convergence of the algorithm by Nesterov. The comparison of the original Nesterov's method together with the algorithm in its modified form - for various basic step size rules used for defining the sequence of  $\alpha^{(k)}$  - is depicted there. Particular values of parameters of those rules were chosen in order to obtain as fast convergence as possible (without the proof, only by engineering guess). On the figure we can identify some features of the particular step size rules which we already described. The fastest convergence was achieved by the original Nesterov's algorithm (reaching the threshold in 272<sup>th</sup> iterate), followed by constant step size rule with  $\alpha = 0.0026$  (reaching threshold in 273<sup>th</sup> iterate). The original Nesterov's algorithm was chosen for comparison in Subsection 6.1.6.

As we already explained in Subsection 6.1.4, applying the Polyak's step size rule by

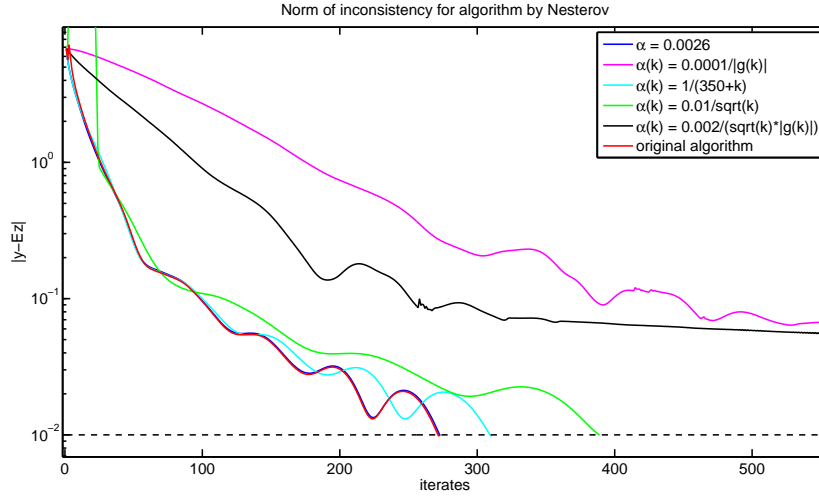


Figure 6.22: Convergence for algorithm by Nesterov

definition needs the central master algorithm. The norm of inconsistency during iterates is depicted in Figure 6.23, both for the algorithm with  $f^*$  known and with  $f^*$  unknown. The Polyak's algorithm for  $f^*$  known reaches the threshold in 252<sup>th</sup> iteration, while for  $f^*$  unknown in 395<sup>th</sup> iteration.

For the optimal value unknown it is not possible to use the estimate by Bertsekas, because the assumption  $f_{best}^{(k)} = \max_{0 \leq i \leq k} f(x^{(i)}) > 0$  does not hold (see 4.4.2 for detailed explanation), so the estimate by Nedic was used, modified for searching for the maximum. As we described in 4.4.2, the algorithm uses four parameters - initial value of the target parameter  $\delta_0$ , its minimal value  $\delta_{min}$ , and parameters for its update  $\sigma$  and  $\beta$  (fifth parameter could be  $\kappa^{(k)}$  from 4.17). Those parameters should be adjusted in order to get “the best” result, which can make the implementation a bit tricky. In this particular case the values

$$\delta_0 = 1, \delta_{min} = 0.05, \sigma = 1.4, \beta = 0.9 \quad (6.2)$$

(and  $\kappa = 1$  for all iterations) were chosen.

For this algorithm it can be interesting to observe the function values - in the Figures 6.24 and 6.25 we can see the distance to the optimal value during iterates for the  $f^*$  known and  $f^*$  unknown.

### 6.1.6 Comparison of different step size rules

In this subsection we compare together convergences of all the step size rules presented. This comparison is provided in the Table 6.1 and the Figure 6.26, where we used those parameters of various step size rules, which were identified by red color in previous

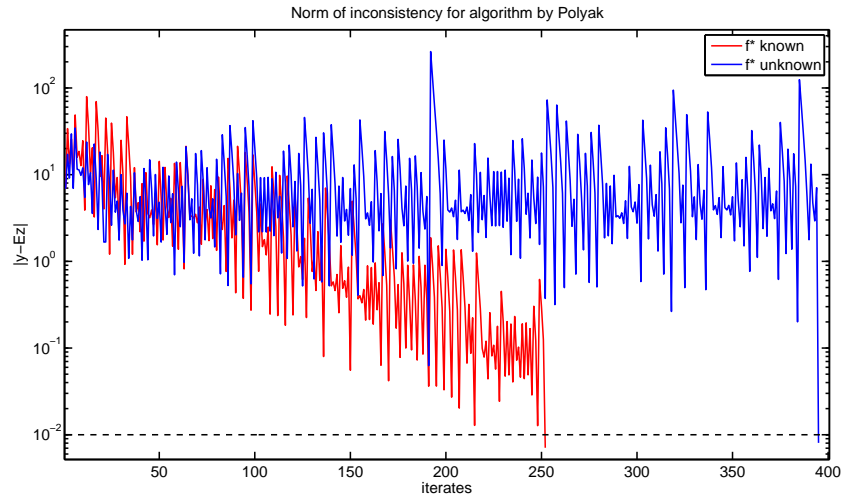


Figure 6.23: Convergence for algorithm by Polyak

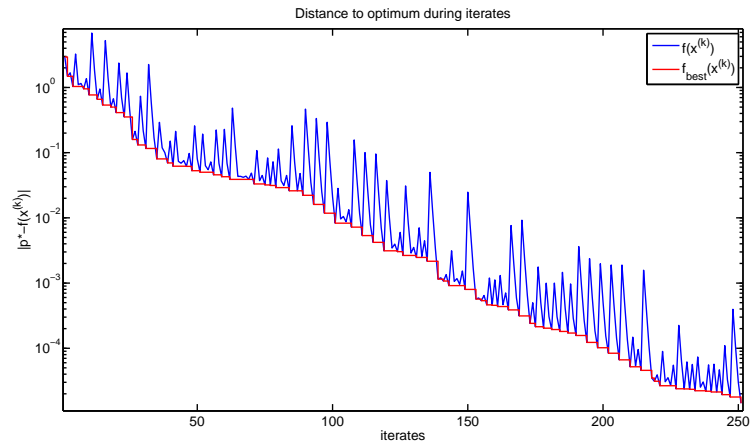


Figure 6.24: Distance to the optimum - Polyak,  $f^*$  known

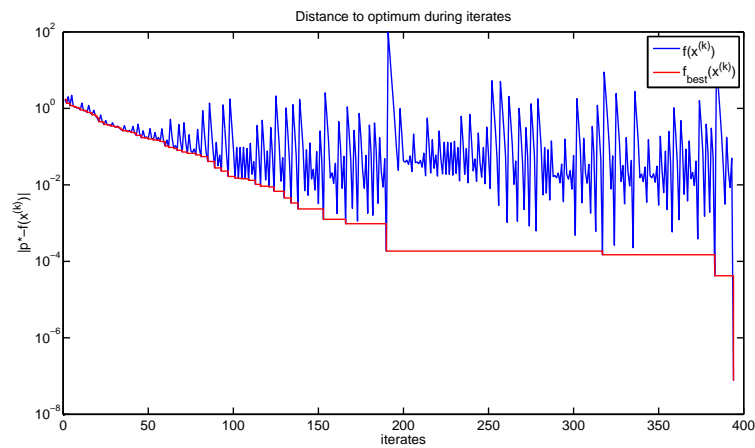


Figure 6.25: Distance to the optimum - Polyak,  $f^*$  unknown

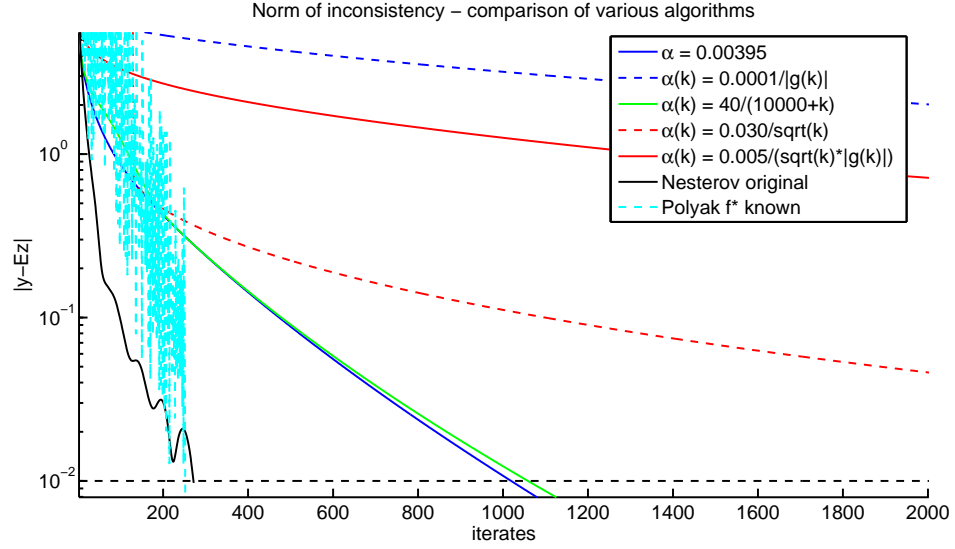


Figure 6.26: Comparison of convergences of all step size rules used

step size rule $\alpha^{(k)} = \dots$	parameter(s)	# of iterates to reach threshold
$\alpha$	$\alpha = 0.00395$	1020
$\gamma / \ g^{(k)}\ _2$	$\gamma = 0.0001$	> 12000
$a/(b+k)$	$a = 40, b = 10000$	1059
$a/\sqrt{k}$	$a = 0.030$	4969
$a/(\sqrt{k} \ g^{(k)}\ _2)$	$a = 0.005$	> 12000
Nesterov - original	-	272
Nesterov - modified	$\alpha = 0.0026$	273
Polyak - $f^*$ known	-	252
Polyak - $f^*$ unknown	see Eq. 6.2	395

Table 6.1: Comparison of convergences of all step size rules used

subsections. The algorithms by Polyak (with  $f^*$  known) and Nesterov give the fastest convergence. The algorithm by Polyak has an obvious advantage - it already knows the optimal value. In addition, both of those algorithms (together with version of Polyak with  $f^*$  unknown) use the information from all subsystems, as we said in Subsection 6.1.4. This can be one of the factors that makes their convergence that fast. Modified Nesterov's algorithm (with constant step size) has good convergence properties, although it does not need the function values of subsystems and thus the coordinator does not need to be implemented centrally. Polyak's algorithm with  $f^*$  unknown and modified Nesterov's algorithm are not depicted in the Figure 6.26 for sake of simplicity.

As we already indicated in last subsection, the number of parameters which have to be tuned in case of particular step size rule, is relevant for practical application. The most parameters (four) are needed in case of Polyak's algorithm with  $f^*$  unknown. For the square summable, but not summable step size rule, using the sequence  $a/(b+k)$ , there are two parameters, for other basic step size rules there is only one parameter to be tuned. In case of Nesterov's algorithm in its original form there is no particular parameter, although as we stated in Subsection 4.4.3, the Lipschitz constant could simplify the computation. In case of its modification the situation is the same as in the case of basic step size rules. In case of Polyak's algorithm with  $f^*$  known only that  $f^*$  can be perceived as the parameter, but it is not subject to tuning in a common sense.

We can conclude this section by stating that for this engineering problem the modified Nesterov's algorithm (with constant step size) was the most suitable, ensuring reasonably good convergence properties with relatively easy implementation.

## 6.2 Static control - industrial energy network

### 6.2.1 Problem formulation

The second problem of practical part of the thesis is based on the industrial energy network consisting of three boilers, three headers, three turbines, one condensation unit and duct connecting them, see Figure 6.27. The steam produced by boilers (let's mark them  $B_1$ ,  $B_2$  and  $B_3$ ) is stored in headers ( $H_0$ ,  $H_1$  and  $H_2$ ), and is used by turbines ( $T_1$ ,  $T_2$  and  $T_3$ ) to produce electric energy. There is also demand on steam from headers (values  $D_0$ ,  $D_1$  and  $D_2$ , together  $D = [D_0, D_1, D_2]^T$ ), which has to be satisfied. The condenser  $C$ , where surplus of steam can be condensed, is connected to turbine  $T_2$ . We want to maximize the profit in the network ensuring that the demand is satisfied and flows in the duct are within specified limits.

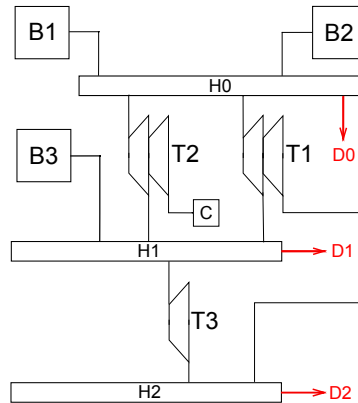


Figure 6.27: Industrial energy network scheme

For all boilers cost functions are defined representing the costs of producing the

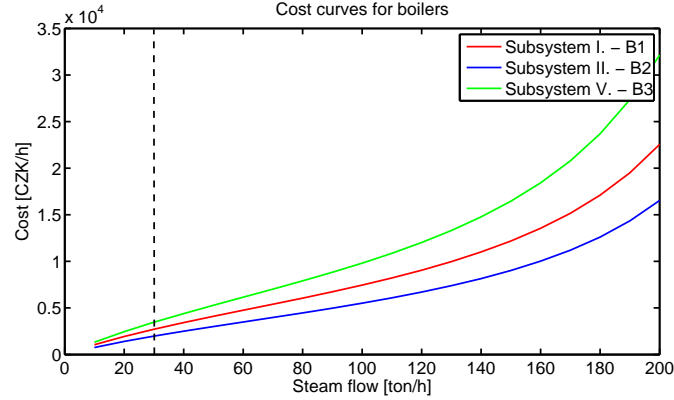


Figure 6.28: Cost curves for boilers

steam flow. The cost curves were derived from efficiency curves, which are part of boiler static tests in [BOIL10], using the equation

$$C = C_F \cdot \frac{d_H}{V_{hh}} \cdot \frac{y}{\eta(y)}, \quad (6.3)$$

where the scalar values' meaning is

$C$	total cost of produced steam [CZK/h],
$y$	steam flow produced [ton/h],
$\eta = \eta(y)$	efficiency, $\eta \in \langle 0, 1 \rangle[-]$ ,
$C_F$	price of fuel [CZK/ton],
$d_H$	steam enthalpy difference (gain) [kJ/kg],
$V_{hh}$	fuel higher heating value (HHV) [kJ/kg].

Reasonable practical values  $V_{hh} = 17400$  kJ/kg,  $d_H = 2517$  kJ/kg,  $C_F = 400$  CZK/ton were considered, giving the equation

$$C = 57.86 \frac{y}{\eta(y)}, \quad (6.4)$$

for what the efficiency curve for oxygen level  $L_{O_2} = 3.0\%$  was selected. The result was adjusted for three different boilers, depicted in Figure 6.28. There it can be seen that the curves are not convex - it is necessary to omit the part of the curves below 30 ton/h to preserve the convex nature of the problem. This is completely acceptable from the practical point of view, because the boilers are not working at those low values of steam flow.

In case of turbines the revenue from production of electric power is defined - the quadratic model of turbine was taken from [TURB10], where for two sections it has a

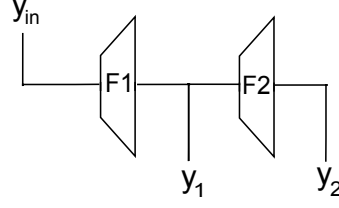


Figure 6.29: Two-section turbine scheme

form of

$$P = a_0 + \sum_{i=1}^2 b_i F_i + \sum_{i=1}^2 c_i F_i^2, \quad (6.5)$$

where  $P$  is the power generated in [MW] and  $F_i$  are flows through sections in [ton/h]. The equation was transformed to have the flows through outputs  $y_1 = F_1 - F_2$  and  $y_2 = F_2$  as the independent variables (see Figure 6.29). The revenue as the function of steam flow through outputs is then

$$R = C_E \cdot P(y_1, y_2), \quad (6.6)$$

where  $C_E$  is the price of electricity [CZK/MW]. The resulting revenue curves for price  $C_E = 1440$  CZK/MW were adjusted for different two-stage turbines and depicted in Figure 6.30. One-stage turbine revenue curve is shown in Figure 6.31.

For the problem formulation, we can choose the formulation for maximizing the profit (revenues minus cost (expenditures) [GAAP11]) for all network, or minimizing its negative value (cost minus revenues). Let's choose the second option, so we want to

$$\text{minimize } J = \sum_{i=1}^3 C_i - \sum_{j=1}^3 R_j,$$

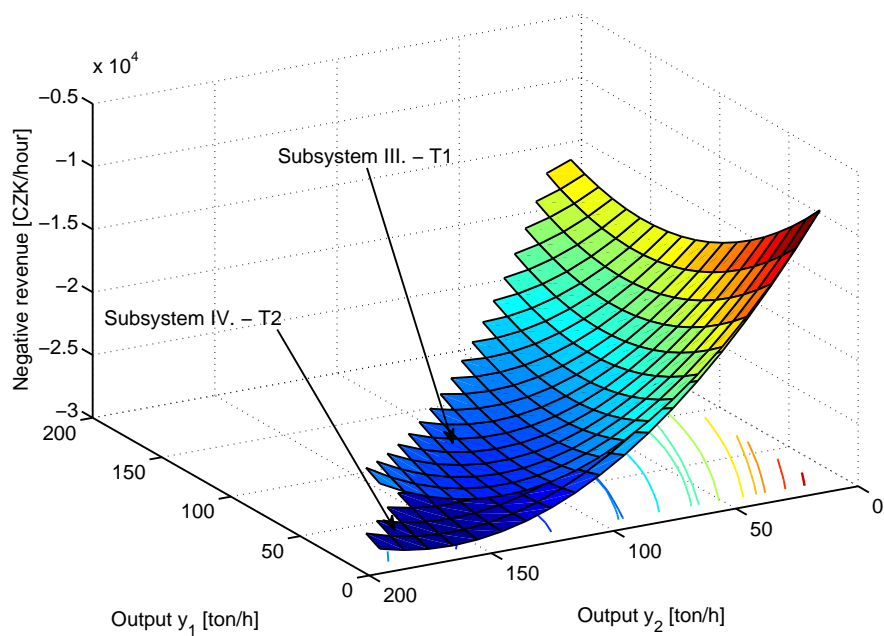
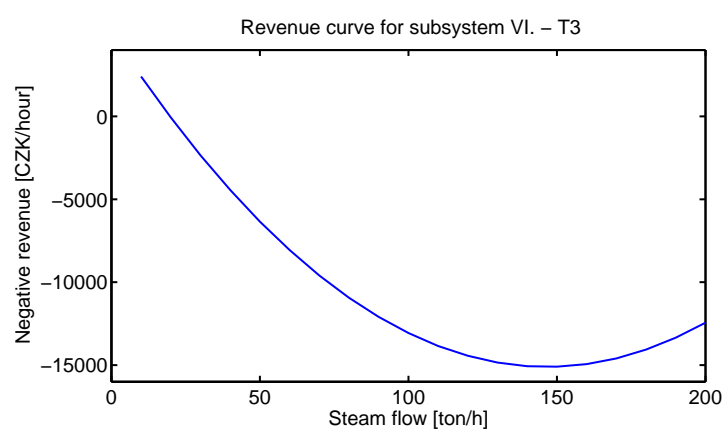
where the values  $C_i$  and  $R_j$  correspond to Equations 6.3 or 6.6 for particular device. The objective function is measured in [CZK/h]. When the notation  $J^M$  is introduced (again  $M$  is indexing all subsystems,  $M \in \{I., \dots, VI.\}$ , see Figure 6.32), then for boilers ( $M_B \in \{I., II., V.\}$ ) the objective function is

$$J^M = J^{M_B}((y)_{M_B}) = C^{M_B}((y)_{M_B}),$$

and for turbines ( $M_T \in \{III., IV., VI.\}$ )

$$J^M = J^{M_T}((y)_{M_T}) = -R^{M_T}((y)_{M_T}).$$

As we said, there is a demand, which is specified as  $D = [50, 150, 200]^T$ , and the steam flow in duct  $y$  is limited (to not damage the duct) by the interval  $y \in \langle 10, 200 \rangle$  ton/h.

Figure 6.30: Revenue functions of turbines  $T_1$  and  $T_2$ Figure 6.31: Revenue curve for turbine  $T_3$



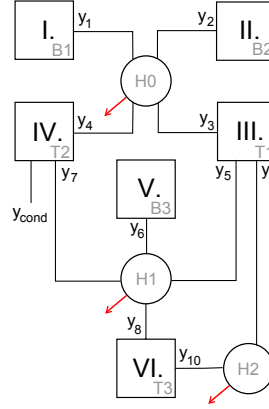


Figure 6.32: Industrial energy network - block diagram

Now we have all information needed to formulate the optimization problem:

$$\text{minimize } \sum_M J^M((y)_M) \quad (6.7)$$

subject to  $10 \preceq y \preceq 200$

$$y_3 = y_5 + y_9$$

$$y_4 = y_7 + y_{\text{cond}}$$

$$y_8 = y_{10}$$

$$50 = y_1 + y_2 - y_3 - y_4$$

$$150 = y_5 + y_6 + y_7 - y_8$$

$$200 = y_9 + y_{10},$$

where first three equality constraints represent the balance equations (sum of inputs must equal to the sum of outputs) in turbines and the last three equality constraints balance equations at headers - to ensure the physical feasibility. To sum up, we are searching for the optimal amount of steam flows in duct (vector  $y$ ) satisfying the physical constraints and the demand.

### 6.2.2 Distributed algorithm and its solution

To solve this problem we derive the graph corresponding to the problem formulation - as depicted in Figure 6.32. On that graph there are two types of nodes - “standard” nodes, where the local optimization takes place, and “net constraint” nodes representing the condition of the net adjacent to them, which has to be satisfied. Balance equations at turbines can be incorporated into optimization of subproblems, as well as inequality constraints on  $y$ . On the other hand balance equations at headers involve different elements of vector  $y$ , implying the “net constraint”.

The graph can be represented by matrix (we are not considering the flow  $y_{\text{cond}}$  which is not connecting two nodes and can be derived from values  $y_4$  and  $y_7$ )

$$E = I_{10}$$

together with net constraint matrix

$$E_C = \begin{bmatrix} 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}.$$

This means that we want to solve the problem (supposing that the balance equations of turbines and duct constraints on  $y$  are incorporated into subproblems)

$$\begin{aligned} & \text{minimize} && \sum_M J^M((y)_M) \\ & \text{subject to} && (y)_M = (E)_M z \\ & && D = E_C z. \end{aligned} \tag{6.8}$$

The cost function is  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , where in this particular case  $n = 10$ , and let  $r$  denote the rank of  $E_C$ , in this case  $r = 3$ .

We know from Section 5.4 that  $z$  is the vector of common values of public variables on the hyperedges. We can incorporate the net constraints to the algorithm described in 5.4.2 and solve it in a way presented there, if we notice that the set

$$\{z | F = E_C z, z \in \mathbb{R}^n\} = \{z_0 + Nw | w \in \mathbb{R}^{n-r}\}. \tag{6.9}$$

There  $z_0$  is any particular solution of  $F = E_C z$  and  $N$  is a matrix whose range is null space of  $E_C$ . Then the problem

$$\text{minimize } \tilde{f}(w) = f(z_0 + Nw)$$

is an unconstrained problem with  $w \in \mathbb{R}^{n-r}$ . From its solution  $w^*$  we can calculate solution of the constrained Problem 6.8 as

$$z^* = z_0 + Nw^*.$$

So during the algorithm 5.2 the master computes values of public variables at each net from the equation

$$y = Ez = E(z_0 + Nw) = Ez_0 + ENw.$$

Let's introduce matrix  $E_N = EN$ . Then

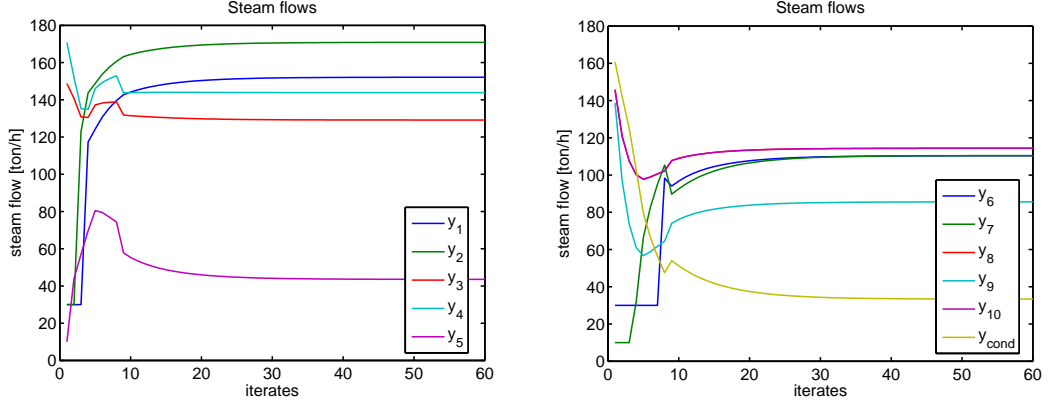


Figure 6.33: Distributed solution obtained by constant step size rule

$$\begin{aligned}
 y - Ez_0 &= E_N w \\
 E_N^T (y - Ez_0) &= E_N^T E_N w \\
 (E_N^T E_N)^{-1} E_N^T y - (E_N^T E_N)^{-1} E_N^T E z_0 &= w.
 \end{aligned} \tag{6.10}$$

The vector  $(z_0 + Nw)$  contains the values of common public variables at each net respecting the net constraints and is used for updating the prices (Lagrange's multipliers). This approach corresponds to the explanation in Chapter 10 of the book [BOYD09].

Figure 6.33 shows how the allocation in the network is changing during iterates of the algorithm, using constant step size rule with  $\alpha = 0.45$ . At the first iterate the subsystems' separate optimal solutions are obtained - the boilers ( $y_1$ ,  $y_2$  and  $y_6$ ) have the initial amount of steam flow at the lowest possible value (because the production of steam only induces costs), but after a few iterates they have to produce the steam in order to satisfy the demand and obtain physically realizable solution. The boiler B<sub>3</sub> (subsystem V., flow  $y_6$ ) initiates its production as the last one (because its price curve is the most expensive from the three boilers), influencing the allocation of steam to turbines - for example in case of turbine T<sub>1</sub> (subsystem III., flows  $y_3$ ,  $y_5$  and  $y_9$ ) the splitting into two output flows is then changed making the flow  $y_9$  more favorable, because it creates more revenue. Particular values after the 60<sup>th</sup> iterate, which we consider as a solution, are shown at Table 6.2. The norm of inconsistency reaches the value of  $\|y - Ez\|_2 = 0.0103$ . The optimal function value (negative profit) is  $f_{\text{distrib}}^* = -25.1 \cdot 10^3$  CZK/h.

### 6.2.3 Comparison to centralized approach

The solution obtained by centralized approach can be seen in the Table 6.3. Comparing it with the Table 6.2 we can see that the biggest difference is 0.77 ton/h in the  $y_{\text{cond}}$  value,

Flow	Value [ton/h]	Flow	Value [ton/h]
$y_1$	152.13	$y_6$	110.37
$y_2$	170.86	$y_7$	110.46
$y_3$	129.15	$y_8$	114.40
$y_4$	143.85	$y_9$	85.60
$y_5$	43.55	$y_{10}$	114.40
		$y_{\text{cond}}$	33.39

Table 6.2: Distributed solution -  $\alpha = 0.45$ , after 60<sup>th</sup> iterate

Flow	Value [ton/h]	Flow	Value [ton/h]
$y_1$	151.98	$y_6$	109.89
$y_2$	170.74	$y_7$	111.14
$y_3$	128.96	$y_8$	114.40
$y_4$	143.76	$y_9$	85.60
$y_5$	43.36	$y_{10}$	114.40
		$y_{\text{cond}}$	32.62

Table 6.3: Centralized solution

which is not exceeding the relative error 3%. In case of other steam flows the situation is better, so we can conclude that the solution obtained by decentralized solution after 60 iterates is acceptable (higher precision could be reached by higher numbers of iterates). Thus in next subsection the threshold for measure of inconsistency will be set to the value  $\|y - Ez\|_2 = 0.01$  for selected step size rules comparison. The optimal function value (negative profit) is  $f_{\text{central}}^* = -24.7 \cdot 10^3$  CZK/h.

#### 6.2.4 Comparison of selected step size rules

In this subsection the comparison of selected step size rules is provided. Step size rules easily implementable and with reasonably fast convergence were selected - the constant step size (with  $\alpha = 0.8$ ), the square summable, but not summable step size (with the sequence  $\alpha^{(k)} = a/(b+k) = 30/(20+k)$ ), the non-summable diminishing step size (with the sequence  $\alpha^{(k)} = a/\sqrt{k} = 2/\sqrt{k}$ ) and modified Nesterov's algorithm with constant step size ( $\alpha = 0.55$ ). The values of parameters were again chosen to obtain "the best" results for each step size rule (by engineering guess, without proof).

Figure 6.34 shows the convergence of those step size rules, with stopping criterion  $\|y - Ez\|_2 = 0.01$ . The number of iterates needed to reach this stopping criterion is pointed out in the Table 6.4. Nesterov's modified rule with constant step size reaches the threshold in the lowest number of iterates. Second best convergence is provided by constant step size rule, while the two last rules have the same number of iterates for given threshold.

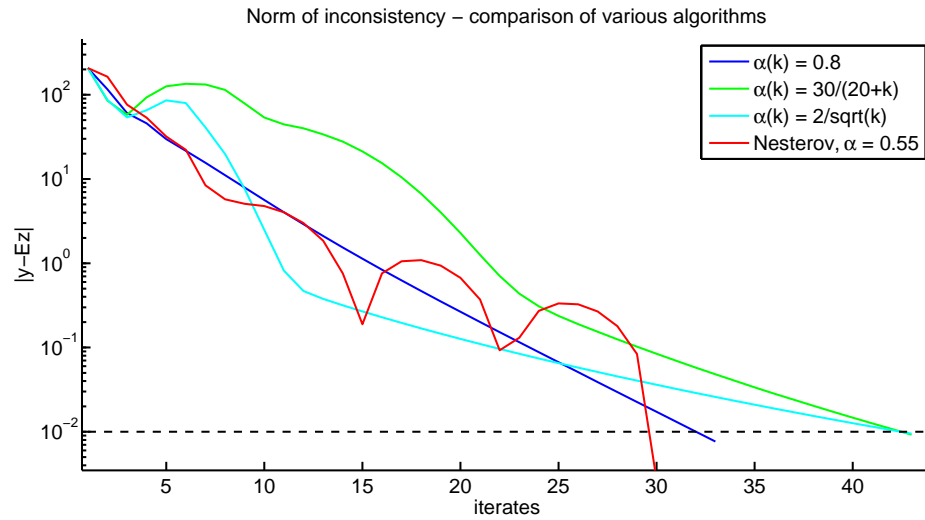


Figure 6.34: Comparison of convergences of all step size rules used

step size rule $\alpha^{(k)} = \dots$	parameter(s)	# of iterates to reach threshold
$\alpha$	$\alpha = 0.8$	33
$\frac{a}{b+k}$	$a = 30, b = 20$	43
$\frac{a}{\sqrt{k}}$	$a = 2$	43
Nesterov - modified	$\alpha = 0.55$	30

Table 6.4: Comparison of convergences of all step size rules used

### 6.2.5 Influence of electricity price

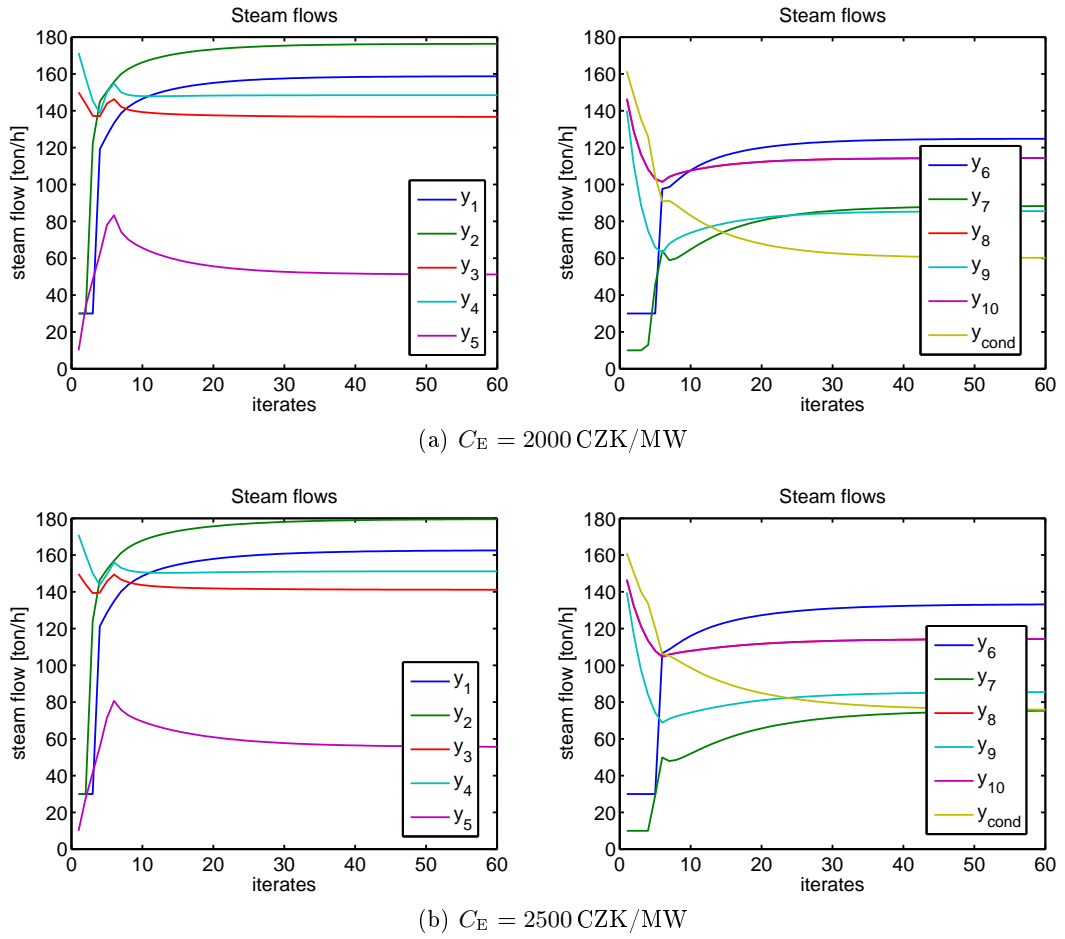
As an additional problem, let's investigate how the allocation in steam network would be influenced by different prices of electricity. For example there could be three tariffs, which influence the revenue we get from turbines and thus the optimal allocation of steam flows. We choose three values of price  $C_E \in \{1440, 2000, 2500\}$  CZK/MW and compare the solutions. In practice it would be possible to prepare the solutions for different situations (different price of electricity, or price of fuel) offline and then to use allocation tables to select appropriate optimal allocation for particular conditions.

The part of results obtained for the different prices of electricity is shown in the Table 6.5, as well as in Figure 6.35 (for the price  $C_E = 1440$  CZK/MW the results were already shown in Figure 6.33). There we see that for higher prices of electricity the higher amount of steam is produced in boilers and higher volume goes to the condenser - satisfying the demand in headers the condenser presents the degree of freedom of the system. The difference in revenue generated by turbines is big enough to overcome the difference of costs implied in boilers, which have to work in less efficient area. That makes the bigger profit, which is depicted in the Table 6.6.

Subsystem	$C_E = 1440$	$C_E = 2000$	$C_E = 2500$
$B_1$	152.13	158.74	162.53
$B_2$	170.86	176.35	179.52
$B_3$	110.37	124.83	133.14
$T_1$ - first output	43.55	51.17	55.68
$T_1$ - second output	85.59	85.55	85.47
$T_2$ - first output	110.46	88.28	75.26
$T_2$ - condenser	33.39	60.19	75.91
$T_3$ - first output	114.40	114.37	114.32

Table 6.5: Flows [ton/h] at selected subsystems for different  $C_E$  [CZK/MW]

	$C_E = 1440$	$C_E = 2000$	$C_E = 2500$
Profit [CZK/h]	$25.1 \cdot 10^3$	$49.0 \cdot 10^3$	$71.1 \cdot 10^3$

Table 6.6: Profit for different  $C_E$  [CZK/MW]Figure 6.35: The solutions for different  $C_E$

## Chapter 7

# Conclusion

### 7.1 Investigative part

The first objective of this thesis was to investigate the decomposition methods for optimization problems and apply this framework into model predictive control. The overview of model predictive control was done first (in Chapter 2), as well as the review of basic optimization terms such as duality, Lagrangian or KKT conditions (Chapter 3). Overview of both topics was supported by examples to get better insight into the subject matter - the MPC with blocking and soft constraint and the physical meaning of Lagrange multipliers were presented.

Based on the background research, which was focused on decomposition methods and distributed MPC, its historical roots and state-of-the-art (presented in Appendix A), the subgradient methods were chosen to solve the master problem of distributed algorithm. Those subgradient methods were investigated in more detail (in Chapter 4), especially the projected subgradient method for dual problems (in Subsection 4.2). In addition, different step size rules were presented, together with advanced step size rules by Polyak and Nesterov. The decomposition methods were summarized and the general form for both primal and dual decomposition was applied to basic example of interconnected systems (Chapter 5).

### 7.2 Implementation part

The core of the application part was to apply the decomposition concepts to engineering problems, which in the large scale can be too demanding (even intractable) for centralized solution (Chapter 6) - the ability to handle large problems is the first main advantage of decomposition methods. The first problem was the control of three interconnected systems with given reference signals, that are controlled by MPC. Dual decomposition algorithm was implemented in MATLAB to solve the problem of optimal

control in one sampling period, using projected subgradient method with affine projection to solve that dual problem. As the stopping criterion the norm of inconsistency was the most suitable. The solution was compared to the centralized one and it was shown that for sufficient number of iterations they are the same. All the step size rules mentioned were implemented.

The second example was the static optimization of industrial energy network, composed of three boilers, three turbines and three headers. The problem was to find optimal load allocation for steam produced by boilers and used by turbines to maximize the profit, respecting given physical constraints and satisfying the steam demand. The framework of decomposition techniques had to be enhanced by “net constraint” in this case. The solution obtained by the distributed approach was again successfully compared to the centralized solution.

### 7.3 Innovative part

Different possibilities of practical deployment of decomposition methods were depicted - master algorithm can be situated centrally, distributed to nets or located in particular subsystems. This brings the second advantage of the concept, the modularity. Various features of those solutions were commented, giving the reasons why for the algorithms by Nesterov (in original form) and Polyak the master have to be located centrally.

Convergence properties (in terms of inconsistency norm) of all the step size rules were investigated thoroughly using the MPC problem. The features of all step size rules were commented. The comparison of the convergences was provided, showing that the algorithms of Nesterov and Polyak with  $f^*$  known give the fastest convergence rates. One of the reasons is that they use the function values from all subsystems (thus located centrally). The modification of Nesterov’s algorithm (with constant step size or square summable, but not summable step size rule) gives comparable convergence rates without need for those function values. In case of basic step size rules the fastest convergence was achieved by the constant and the square summable, but not summable step size rule. From practical point of view the number of parameters, which have to be tuned for particular problem, is also important and was commented. Nesterov’s modified algorithm needs only one (or two) parameters to tune, so from the analysis it looks as the most suitable option for solving the master algorithm.

In the industrial energy network problem the convergence properties of selected step size rules were also compared, the good performance of Nesterov’s modified algorithm was repeated even in solution of this problem. The influence of price of electricity to steam flow allocation was investigated - the higher price of electricity makes boilers to work even in the lower efficiency zone, because turbines then generate revenues overcoming the increase in costs.



# Bibliography

- [BAKU08] BAKULE, L. *Decentralized control: An overview*, Annual Reviews in Control, Volume 32, Issue 1, April 2008, Pages 87-98, ISSN 1367-5788.
- [BEND62] BENDERS, J.F. Partitioning procedures for solving mixed-variables programming problems. *Numerische Matematik*. 1962, 4, s. pp. 238-252.
- [BERT99] BERTSEKAS, Dimitri. *Nonlinear Programming*. Second edition. Belmont, Massachusetts : Athena Scientific, 1999. 791 p. ISBN 1-886529-00-0.
- [BOIL10] Boiler Static Tests: Report, Release 110 Draft. *Honeywell International Inc.* 2010, HPL-21-307-0311-01, p. 1-29.
- [BOYD07] BOYD, Stephen, et al. *Notes on Decomposition Methods* [online]. Stanford : Stanford University, 2007-02-12 [retrieved 2011-01-28]. Accessible from: <[http://www.stanford.edu/class/ee364b/...notes/decomposition\\_notes.pdf](http://www.stanford.edu/class/ee364b/...notes/decomposition_notes.pdf)>.
- [BOYD08] BOYD, Stephen; MUTAPCIC, Almir. *Subgradient Methods*. In Notes for EE364b, Stanford University, Winter 2006-2007. [online] April 13, 2008 [retrieved 2011-04-08]. Accessible from: <[http://www.stanford.edu/class/ee364b/...lectures/subgrad\\_method\\_notes.pdf](http://www.stanford.edu/class/ee364b/...lectures/subgrad_method_notes.pdf)>
- [BOYD09] BOYD, Stephen; VANDENBERGHE, Lieven. *Convex optimization* [online]. Seventh printing with corrections 2009. New York, USA : Cambridge University Press, 2004 [retrieved 2011-01-28]. Accessible from: <[http://www.stanford.edu/~boyd/cvxbook/bv\\_cvxbook.pdf](http://www.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf)>. ISBN 0521833787.
- [CHAL10] CHALOULOS, Georgios; HOKAYEM, Peter; LYGEROS, John. Distributed Hierarchical MPC for Conflict Resolution in Air Traffic Control. *2010 American Control Conference : Baltimore, MD, USA* .
- [CHEN82] CHEN, C.C.; SHAW, L. On receding horizon control. *Automatica*, 16(3):349-352, 1982.

- [CUTL80] CUTLER, C.R.; RAMAKER, B.L. Dynamic matrix control - a computer control algorithm. *Proceedings of the Joint Automatic Control Conference, 1980*. Vol. 1. San Francisco, CA. Paper No. WP5-B.
- [DANT60] DANTZIG, G.B.; WOLFE, P. Decomposition principle for linear programs. *Operations Research*, 8: 101-111, 1960.
- [DING10] DING, Bao-Cang. *Modern Predictive Control*. USA : CRC Press, 2010. 276 p. ISBN 978-1-4200-8530-3.
- [EVER63] EVERETT, Hugh. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research*. May-June 1963, Vol. 11, No. 3, , s. pp. 399-417.
- [GAAP11] GAAP [online]. 2011-10-24 [retrieved 2011-12-19]. Rozdíl v českých a anglických pojmech. Accessible at:  
<[http://www.gaap.cz/index.php?ln=1&tm=5&om=191...&z\\_id=a\\_diskuse\\_c&msg=4101&sl=4101](http://www.gaap.cz/index.php?ln=1&tm=5&om=191...&z_id=a_diskuse_c&msg=4101&sl=4101)>
- [GONZ08] GONZAGA, Clovis; KARAS, Elizabeth. Optimal steepest descent algorithms for unconstrained convex problems: fine tuning of Nesterov's method. [online] *Optimization Online*, 2008. [retrieved 2011-12-26]. Accessible at:  
<[http://www.optimization-online.org/DB\\_FILE/2008/08/2062.pdf](http://www.optimization-online.org/DB_FILE/2008/08/2062.pdf)>
- [HAVL96] HAVLENA, Vladimír; ŠTECHA, Jan. *Moderní teorie řízení*. Praha : Vydavatelství ČVUT, 1996. 291 s. ISBN 80-01-01076-7
- [HAVL05] HAVLENA, Vladimír; LU, Joseph. A distribute automation framework for plant-wide control, optimisation, scheduling and planning. *Proceedings of the 16th IFAC World Congress : Prague, Czech Republic*. July 2005.
- [HONE11] HONEYWELL PRAGUE LABORATORY. Test of Nesterov Accelerated Gradient Method. M-file. Prague, 2011.
- [INST10] Institut fur Automatic [online]. 2010-04-05 [retrieved 2011-01-12]. Stochastic MPC Group. Accessible at: <<http://control.ee.ethz.ch/~smcpc/>>.
- [JADB09] JADBABAIE, Ali; OZDAGLAR, Asuman; ZARGHAM, Michael. A Distributed Newton Method for Network Optimization. *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference : Shanghai, P.R. China, December 16-18, 2009*.
- [JAVA10] JAVALERA, Valeria; MORCEGO, Bernardo; PUIG, Vicenç. Negotiation and Learning in Distributed MPC of Large Scale Systems. *2010 American*

*Control Conference : Marriott Waterfront, Baltimore, MD, USA June 30-July 02, 2010.*

- [KEVI05] KEVICZKY, Tamas; BORRELLI, Francesco; BALAS, Gary J. Stability analysis of decentralized RHC for decoupled systems : *Proceedings of the Joint 44th IEEE Conference on Decision and Control and European Control Conference*, pages 1689–1694, Seville, Spain, December 2005
- [KULH11] KULHÁNEK, Petr. *TF1: Teoretická mechanika: Studijní text pro doktorské studium* [online]. 2. doplněné vydání. Praha: FEL ČVUT, 2011 [retrived 2012-01-02]. Accessible at: <<http://www.aldebaran.cz/studium/mechanika.pdf>>
- [LANG03] LANGBORT, C.; D'ANDRE, R. Distributed control of heterogeneous systems interconnected over an arbitrary graph. *Decision and Control, 2003. Proceedings. 42nd IEEE Conference, pp. 2835- 2840 Vol.3, 9-12 Dec. 2003*
- [LANG04] LANGBORT, C., et al. A decomposition approach to distributed analysis of networked systems. *43rd IEEE Conference on Decision and Control. December 14-17, 2004, pp. 3980-3985.*
- [LAU72] LAU, R.; PERSIANO, R. M.; VARAIYA, P. Decentralized information and control: A network flow example. *IEEE Trans. Autom. Control., vol. AC17, pp. 466-473, August 1972*
- [LASD68] LASDON, Leon. Duality and Decomposition in Mathematical Programming. *IEEE Transactions on Systems and Cybernetics*. July 1968, Vol. ssc-4, No. 2.
- [MACE02] MACIEJOWSKI, Jan. Predictive Control : with Constraints. 2nd edition. Essex, England : Pearson Education Limited, 2002. 309 p. s. ISBN 0-201-39823-0.
- [MAES09] MAESTRE, J. M.; MUNOZ DE LA PENA, D.; CAMACHO, E. F. . Distributed MPC based on a cooperative game. *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference : Shanghai, P.R. China, December 16-18, 2009.*
- [MATH11] Mathworks [online]. 1994-2011 [cit. 2011-10-07]. *MATLAB - The Language Of Technical Computing*. Accessible at: <<http://www.mathworks.com/products/matlab/>>
- [MESA70] MESAROVIČ, M. D.; MACKO, D; TAKAHARA, Y. *Theory of Hierarchical, Multilevel Systems*. Academic Press, New York, London, 1970.

- [MENI09] MENINGHED, K.; AUBRUN, Ch.; YAME, J. . Distributed State Estimation and Model Predictive Control : Application to Fault Tolerant Control. *2009 IEEE International Conference on Control and Automation Christchurch, New Zealand, December 9-11, 2009*.
- [NECO08] NECOARA, Ion; SUYKENS, Johan. A proximal center-based decomposition method for multi-agent convex optimization. *47th IEEE Conference on Decision and Control : Cancun, Mexico, Dec. 9-11, 2008*
- [NEDI02] NEDIC, Angelia. *Subgradient Methods for Convex Minimization* [online]. Massachusetts Institute of Technology, 2002. 174 p. Dissertation. Accessible at: <[https://netfiles.uiuc.edu/angelia/www/mit\\_thesis.pdf](https://netfiles.uiuc.edu/angelia/www/mit_thesis.pdf)>
- [NEDI08] NEDIC, Angelia. *Lecture 19: Subgradient methods* [online]. MIT, 2008 [retrieved 2011-05-12]. Accessible at: <<https://netfiles.uiuc.edu/angelia/www/>>
- [NEST83] NESTEROV, Y. A method for unconstrained convex minimization problem with the rate of convergence  $O(1/k^2)$ . *Doklady AN SSSR* (translated as Soviet Math Docl), 269:543 – 547, 1983.
- [NEST04] NESTEROV, Y. *Introductory lectures on convex optimization*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2004. ISBN 1-4020-7553-7.
- [NEST05] NESTEROV, Y. Smooth minimization of non-smooth functions. *Mathematical Programming (A)*, 103(1):127–152, 2005.
- [NOCE06] NOCEDAL, Jorge; WRIGHT, Stephen. *Numerical Optimization*. 2nd edition. USA : Springer, 2006. 664 p. 85 illus s. ISBN 978-0-387-30303-1.
- [PALO06] PALOMAR, Daniel; CHIANG, Mung. A Tutorial on Decomposition Methods for Network Utility Maximization. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, VOL. 24, NO. 8, AUGUST 2006.
- [POLY87] POLYAK, Boris. *Introduction to Optimization*. USA : Optimization New York, 1987. 464 p. ISBN 0-911575-14-6.
- [RANT09] RANTZER, Anders. Dynamic Dual Decomposition for Distributed Control. *American Control Conference : St.Luis, USA, June 10-12, 2009*.
- [ROSS03] ROSSITER, J. A. *Model-based predictive control, a practical approach*. CRC Press, 2003. Boca Raton-London-New York. 318 p. ISBN 0-8493-1291-4

- [SCH085] SCHOR, N. Z. *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, 1985. Berlin-Heidelberg-New York-Tokyo. 162 p. ISBN 3-540-12763-1
- [SCAT09] SCATTOLINI, R. *Architectures for distributed and hierarchical Model Predictive Control - A review*. Journal of Process Control, Volume 19, Issue 5, May 2009, Pages 723-731, ISSN 0959-1524.
- [STECH00] ŠTECHA, Jan. *Optimální rozhodování a řízení*. Vyd.1. Praha : ČVUT, 2000. 242 p. ISBN 80-01-02083.
- [STECH10] ŠTECHA, Jan; PEKAŘ, Jaroslav. *Short Course on Model Predictive Control*. Prague : 2010. 54 p.
- [TEBB01] TEBBOTH, James Richard. *A Computational Study of Dantzig-Wolfe Decomposition* [online]. University of Birmingham : 2001. 237 p. Dissertation. Accessible at: <[http://www.blisworthhouse.co.uk/...](http://www.blisworthhouse.co.uk/...OR/Decomposition/tebbboth.pdf)>...OR/Decomposition/tebbboth.pdf>.
- [TURB10] Turbine Static Tests : Report; Release 2Draft. *Honeywell International Inc.* 2010, HPL-21-307-0322-01, p. 1-13.
- [VENK06] VENKAT, Aswin. *Distributed Model Predictive Control: Theory and Applications* [online]. University of Wisconsin-Madison : 2006. 352 p. Dissertation. Accessible at: <<http://jbrwww.che.wisc.edu/theses/venkat.pdf>>.
- [WAKA08] WAKASA, Yuji, et al. Decentralized Model Predictive Control via Dual Decomposition. *Proceedings of the 47th IEEE Conference on Decision and Control Cancun, Mexico, Dec. 9-11, 2008*
- [WAKA10] WAKASA, Yuji; TANAKA, Kanya; NISHIMURA, Yuki. A Distributed Consensus Algorithm via LMI-Based Model Predictive Control and Primal/Dual Decomposition Methods. *2010 IEEE International Conference on Control Applications : Yokohama, Japan, September 8-10, 2010*.
- [WANG73] WANG, S.; DAVISON, E.J. On the stabilization of decentralized control systems. *IEEE Trans. Automatic Control*, vol. 18, no. 5, p. 473-478, 1973.
- [YAOL09] YAO-LIANG, Yu. *Nesterov's optimal gradient method* [online]. University of Alberta, 2009 [retrieved 2011-12-26]. Accessible at: <[http://webdocs.cs.ualberta.ca/~yaoliang/Non-smooth Optimization.pdf](http://webdocs.cs.ualberta.ca/~yaoliang/Non-smooth%20Optimization.pdf)>
- [ZENG08] ZENG, Jing; XUE, Ding-Yu; YUAN, De-Cheng. Research and Development Trend of Distributed MPC. *Fourth International Conference on Natural Computation, 2008*.

# Appendix A

## Background research

The purpose of this appendix is to give an overview of state-of-the-art in the fields of our interest - the decomposition methods and distributed MPC. The appendix is divided into two sections, which contain description of recently published papers. The two sections are not disjoint, so in some cases the papers were placed into one group subjectively.

### A.1 Decomposition methods

#### A.1.1 Distributed Newton method for network optimization [JADB09]

In this paper the distributed methods for solving minimum cost network optimization problem are investigated. The problem can be formulated as convex optimization problem with linear equality constraints.

The authors compare two methods - dual decomposition method using subgradients and constrained Newton method. The major shortcoming of the former method is slow convergence rate. The latter method, which is of main interest in this paper, is based on representation of the dual Newton direction as the solution of a discrete Poisson equation. This representation is possible because of the sparsity of the incidence matrix of a network, and it makes possible local computation of Newton directions.

It is shown that the Newton method has good convergence properties even when the direction is computed with some (sufficiently small) error. To get the superlinear local convergence properties, the backtracking step size rule is used. The simulations on different graphs show that the Newton method used outperforms the dual subgradient method algorithm - not only in terms of a run time, but it also exhibits a tighter variance.

### A.1.2 A proximal center-based decomposition method for multi-agent convex optimization [NECO08]

In this paper new proximal center-based method is introduced for dual decomposition of separable convex optimization problems. This method is inspired by smoothing the Lagrangian, which was introduced by Nesterov [NEST05]. Its main advantage is that it selects step parameters optimally (in contrast to older methods, where the step parameter is difficult to tune), thus improves the bounds on the number of iterations.

The method is a two-level algorithm and contrary to most other proximal-based methods it gives more freedom to the selection of next iterates (they are not forced to be close to the previous ones). It uses an optimal first-order oracle scheme to update the multipliers (first-order oracle returns  $f(x)$  and the gradient).

The authors present efficiency estimate results for general case of this method. They prove the theorems about the bounds on the duality gap for the method. Next they show how to choose the smoothness parameter optimally.

In the last part some applications of this method are briefly presented – the distributed MPC framework and network optimization problem. The results of numerical experiment in case of network optimization are presented, examining the number of iterations needed and the accuracy of the approximation of the optimum.

### A.1.3 A decomposition approach to distributed analysis of networked systems [LANG04]

The article presents distributed algorithm for analysis of well-posedness and stability in case of a system composed of different sub-units, interconnected over an arbitrary graph. It follows the notation, theorems and the framework introduced in paper [LANG03] - those are reviewed in the beginning, together with the theorem about conditions, when the system is well-posed and stable (well-posedness guarantees that all signals circulating in the loops of the interconnected system are uniquely determined for any initial conditions). The authors then use a decomposition method to solve analysis conditions of this theorem in a distributed fashion.

The primal decomposition with subgradient method is used to solve a set of coupled linear matrix inequalities coming from the analysis. The LMI conditions are reformulated as an optimization problem. The master problem obtained by primal decomposition is not differentiable (although its decomposed parts are convex), the subgradient method used for the solution keeps the parallelizable fashion of the algorithm.

Authors solve an example problem - five interconnected subsystems with large dimension of interconnection signals - with the algorithm presented (on one machine), while centralized approach fails because of lack of memory. In future work they plan to investigate the role of synchronization in truly distributed algorithms in more detail.

#### **A.1.4 Distributed consensus algorithm via LMI-based model predictive control and primal/dual decomposition methods [WAKA10]**

This paper focuses on a consensus problem - a design of distributed strategies of a group of systems, which asymptotically converge to a common value, a consensus point. The systems/agents have general dynamics and are organized into a cycle graph, where they can communicate with the neighboring systems/agents.

The paper presents a centralized model predictive consensus based on linear matrix inequality (LMI) method. Then it focuses on the solution by decomposition methods - the primal decomposition (which gives an upper bound of the optimal value of optimization problem) and dual decomposition method (which gives the lower bound).

The algorithm combining the two methods is proposed to reduce the number of iterations, which means the reduction of communication between the systems/agents, while keeping the convergence to a consensus point. The effectiveness of the proposed algorithm is illustrated on the numerical example.

#### **A.1.5 Dynamic dual decomposition for distributed control [RANT09]**

In the beginning the article explains basic concepts of dual decomposition techniques, also from the game theory point of view. It uses dual decomposition technique for decomposition of feedback systems including dynamics in both decision variables and prices.

The author presents and proves the theorem, which shows how bounds on the global distance from optimality can be derived from corresponding bounds for individual agents, as well as it presents another possibility of introducing prices. Those theorems are used to perform distributed performance validation of decentralized control laws for the linear system. Then the approach for synthesis of feedback controllers is depicted by presenting the distributed gradient algorithm.

#### **A.1.6 A tutorial on decomposition methods for network utility maximization [PALO06]**

This paper gives an overview of decomposition methods and deals with the application of those methods into network utility maximization (NUM).

First, the convexity, Lagrange duality and Jacobi and Gauss–Seidel iterations are reviewed, as well as an implication of different time scales of variable updates. The authors then present basic „blocks“ for distributed algorithm design based on primal and dual decomposition, as well as indirect decomposition (reformulation the original problem by introducing of auxiliary variables). From those blocks the hierarchical decomposition with different schemes can be implemented, resulting in different speed



and robustness of convergence, amount and symmetry of message passing, amount and symmetry of local computation, implications to engineering implementations, etc.

The understanding of the decomposable structures in NUM is crucial to both resource allocation and functionality allocation. In the last part of the paper, authors present recent examples from the field of NUM on systematic search for alternative decompositions (on the example of quality of service rate allocation), decoupling techniques for coupled objective functions and decoupling techniques for coupled constraint sets that are not readily decomposable by re-parametrization (on the example of uplink power control wireless networks).

## A.2 Distributed MPC

### A.2.1 Research and development trend of distributed MPC [ZENG08]

The paper gives an overview of actual development in distributed MPC and presents how recent literature deals with problem decomposition and assignment as well as cooperation/communication between controllers.

Three classes of MPC framework are introduced, depending on different domains of application - single MPC controller used as a replacement of decentralized PID controllers, multiple MPC controllers engineered as a replacement of decentralized PID controllers and MPC used as supervisory layer in a cascaded setting. The authors treat the term „distributed“ (system consisting of subsystems) as more general than „decentralized“ (system consisting of strictly independent subsystems).

Authors distinguish centralized system model decomposition, when centralized model is constructed first and then decomposed into subsystems, and decomposition by direct design of subsystems and connections between them. They also present various control strategies for distributed problems found in the literature, regarding mainly the communication between agents – for example partitioning controllers into groups, coordination of controllers or control network capacity. Different methods for system analysis of distributed MPC are also described.

### A.2.2 Distributed model predictive control: theory and applications [VENK06]

In this dissertation the framework for control of large systems is developed through the suitable integration of subsystem-based MPCs. The author shows that modeling the interactions between subsystems and exchanging trajectory information among MPCs is not sufficient to improve controller performance and does not provide even closed-loop stability.

A cooperative distributed MPC framework, in which the objective functions of the

local MPCs are modified to achieve system-wide control objectives, is proposed. For this framework properties such as feasibility, optimality and closed-loop stability are established.

Two distributed state estimation strategies are presented, as well as subsystem based disturbance modeling framework. Distributed MPC algorithm is extended for distributed constrained LQR (linear quadratic regulation) and also augmented to allow asynchronous operation among MPCs (integration of MPCs with varying computational time requirements without requiring all MPCs to operate at the slowest computational rate). The partial cooperation is described, which results in simpler controller network structure and reduction in communication, but on the other hand there is no stability guarantee (except of special cases).

### **A.2.3 Decentralized model predictive control via dual decomposition [WAKA08]**

The article uses dual decomposition algorithm for decentralized model predictive control, particularly the case when control outputs of subsystems have coupling constraints represented by linear equations. Those constraints can represent a formation constraints of multiple vehicles. The case when neighboring SISO (single input and single output) subsystems communicate is considered - those pairs (groups) form a chain structure (so in fact it is a distributed approach).

The procedure of derivation of the algorithm follows [BOYD09] - the Lagrangian is formed and then dual function and dual problem are formulated (strong duality is assumed). Because the objective function of the dual problem is not differentiable, the subgradient method is used to solve the problem, with subgradient projection to the hyperplane associated with the equality constraint. The subgradients are communicated in each pair (group). This framework is modular - new subsystem can be added easily.

The case of subsystems with the same dynamics is considered, what simplifies the computations. Moreover, the extension of a chain structure is introduced - the case of subsystem belonging to more than two communication groups. The method is illustrated on two numerical examples - with 3 and 15 subsystems.

### **A.2.4 Stability analysis of decentralized RHC for decoupled systems [KEVI05]**

The article deals with system composed of distinct dynamical subsystems that can be independently actuated, having common objective and (input and state) constraints (RHC stands for receding horizon control). The interaction between the subsystems is local and is represented by an interaction graph. The centralized MPC problem is decomposed so that each controller is associated to a different subsystem, local control

inputs are computed based on the states of the subsystem and its neighbors.

However, local MPC designs can lead to instability of the overall system and the inputs computed locally are not guaranteed to be globally feasible (because at the particular controller the prediction of neighboring states is done independently from their subproblem solutions).

Three different approaches to analyzing the stability of the entire system are presented and the theorem stating sufficient conditions for the asymptotic stability of the closed-loop system is proved - the stability is related to the prediction mismatch (between the predicted and actual control solutions of neighbors) and the initial conditions of the overall system. The smaller the mismatch, the larger the set of initial states for which the system is asymptotically stable.

Authors investigate the influence of exchanging the optimal solutions between neighbors – it has beneficial effect on proving the stability, if it leads to reduced prediction mismatch. They present rather counter-intuitive fact that approaching equilibrium might need increasing exchange of information. However, the simulations don't show this need, because the prediction error converges to zero at a fast rate. The prediction length is also discussed – the prediction errors can increase with longer prediction horizon, which causes worse performance or can even lead to instability. Proposed decentralized (distributed) framework was applied to a number of control problems, for example formation flight or paper machine control.

#### **A.2.5 Distributed MPC based on a cooperative game [MAES09]**

The paper presents distributed MPC algorithm for control of two constrained linear systems coupled through inputs (this class of systems is used in modeling of supply chains). The algorithm presented keeps the communication between two controllers low, while obtaining a Pareto optimal solution. In terms of game theory, both agents play cooperative game at each time step – they share information about their strategies and cost functions in order to choose the strategy minimizing global cost function.

The authors state and proof sufficient conditions for (practical) stability of the closed-loop system, which were obtained by terminal constraint/terminal region approach. They also provide the procedure of controller design to satisfy these conditions. The approach is illustrated by an example with system consisting of two double integrators with coupled inputs, with state and input constraints.

#### **A.2.6 Distributed state estimation and model predictive control: application to fault tolerant control [MENI09]**

This paper deals with unconstrained distributed model predictive control of complex and interconnected systems (flows of material/energy/information, the linear intercon-

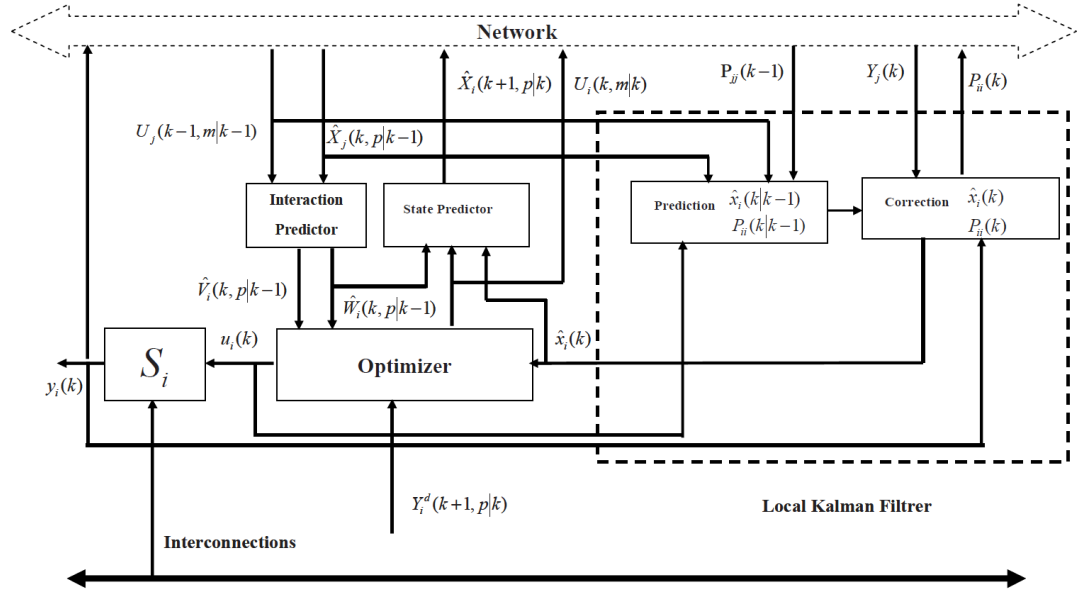


Figure A.1: Block diagram of controller

nections are supposed).

The cooperative strategy between controllers is used to achieve desired performance; local state feedback is employed, using distributed Kalman filters for unmeasurable states. Each controller is divided into three function blocks – the optimizer, the state predictor and the interaction predictor (see Figure A.1).

The basic idea of the algorithm is as follows: in one step, each sub-controller receives predicted future state trajectories and control inputs from other controllers. It combines it with its local state trajectory and control input – computes the prediction of the subsystem interactions. It receives state estimate from local Kalman filter and desired trajectory (reference) over the horizon, computes the optimal control strategy and sends it to the network (to other controllers). Then it applies the first element of the optimal control strategy, computes the (local) future state trajectory and broadcasts it to other controllers.

Presented approach is applied to a Fault Tolerant Control (FTC) problem in a distributed framework. In the numerical simulation, the actuator failure is simulated. Authors would like to explore the distributed detection and isolation based on distributed MPC formulation in the future work.

### A.2.7 Negotiation and learning in distributed MPC of large scale systems [JAVA10]

The authors give a review of distributed MPC algorithms together with reasons why they are used for control of large scale systems. They describe the Multi Agent MPC

architecture based on partitioning of the large scale system. The architecture consists of two main types of elements - the MPC Agent, who controls its particular partition of the system, and Negotiator Agent, who determines the value of one or more shared variables between two MPC Agents.

The presented architecture is non-iterative (information is distributed among regulators only once at each sampling period) and cooperative (regulators optimize global objective function). Negotiator Agents are using reinforced learning techniques based on their experience - the algorithm used comes from Q-learning algorithm, which keeps the reinforcement gained for each state and action (the data from centralized MPC could be used for initializing the algorithm).

Exploiting this approach the model of water network consisting of 8 tanks (states) and 11 valves (control variables) is controlled, partitioned into two subsystems. The results of distributed control are compared with the centralized MPC - the decentralized approach converges to the centralized MPC; it gives worse value of objective function, but on the other hand in some states the distributed control gives lower error.

The control architecture described in this paper is actually being deployed on the water network in Barcelona, with about 200 sectors and about 400 control points.

#### **A.2.8 Distributed hierarchical MPC for conflict resolution in air traffic control [CHAL10]**

The paper presents the control scheme for the conflict resolution in air traffic management. The control hierarchy comprises three levels – the lowest level is the Flight Management System, which simulates the real aircraft dynamics. It uses inputs generated by middle level, the navigation functions. The navigation function (often used in robotics) uses simplified planar dynamics of the aircraft and computes a potential field, where negative gradient guides the aircraft to the target, avoiding obstacles. The main drawback of this method is that it doesn't comprehend the constraints, which are crucial for the aircraft motion – for example minimum and maximum speed, thrust or turning radius. That is why MPC is used at a highest level, to ensure the trajectories respect those aerodynamic constraints.

MPC problem formulated by centralized approach (navigation functions) is non-convex, with unbounded noise (from the wind). In order to deal with this, variation of simulated annealing is used, based on Markov Chain Monte Carlo methods. The optimization problem is then solved in a distributed way in each aircraft. Trajectories are solved sequentially in each aircraft (in a round-robin fashion) – the first aircraft minimizes its own cost function taking into account only the dynamic constraints of other aircrafts, then it broadcasts the solution to all other aircrafts. This solution is used as a constraint for the second aircraft optimization problem etc. To not prioritize the first aircraft, the sequence of aircrafts could be random in each step or the „fairness“

factor could be incorporated into the cost function of the aircrafts.

The results of presented algorithms are presented in comparison with centralized solution, together with computation times. The best results are achieved by the fairness factor introduced into cost function. Distributing of the problem didn't change the feasibility of the original centralized problem. In future work the stochastic alternative will be investigated.

## Appendix B

### Contents of the CD attached

The CD attached contains an electronic version of this thesis - file Petr\_Endel\_2012.pdf.  
The MATLAB source codes are treated as confidential and are property of Honeywell company.