Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

MASTER THESIS

Numerical Algorithms for Quadratic Programming for Approximated Predictive Control

Author: Bc. Pavel Otta
Supervisor: Ing. Ondřej Šantin

Prague, 2013

Czech Technical University in Prague
Faculty of Electrical Engineering

Department of Control Engineering

DIPLOMA THESIS ASSIGNMENT

Student: **Bc. Pavel Otta**

Study programme: Cybernetics and Robotics
Specialisation: Systems and Control

Title of Diploma Thesis: **Numerical algorithms for quadratic programming for approximated predictive control**

Guidelines:

1. Derive quadratic programming task arising in linear model predictive control. Focus on box constrained control strategies.
2. Describe existing numerical methods for quadratic programming for approximate model predictive control for systems with short sampling period.
3. Implement and test some of the algorithms on suitable benchmarks. Use BLAS and LAPACK libraries as possible.

Bibliography/Sources:

[1] Rossiter J. A., Model-Based Predictive Control, CRC Press, 2003
[2] Boyd S., Vandenberghe L., Convex Optimization, Cambridge University Press, 2004
[3] Bertsekas D. P., Nonlinear programming, Second Edition, Athena Scientific, 1999
[4] Specific journal publications

Diploma Thesis Supervisor: Ing. Ondřej Šantin

Valid until the summer semester 2013/2014

[illegible]
prof. Ing. Michael Šebek, DrSc.
Head of Department

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
Fakulta elektrotechnická
L.S.
-4-

[illegible]
prof. Ing. Pavel Ripka, CSc.
Dean

Prague, May 2, 2013

Prohlášení

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

V Praze, dne 10.5.2013

[illegible]

podpis

Poděkování

Můj obdiv patří člověku, který mě na škole naučil ze všech nejvíce. Proto chci s potěšením poděkovat Ondřejovi Šantinovi za to, že se usilovně věnoval vedení mé diplomové práce, že mě motivoval a ochotně mi předával své znalosti.

Pokud je život velká zavařovací sklenice plná golfových míčků, oblázků a písku, pak sdrdečně děkuji golfovým míčkům.

Acknowledgement

I thank O. Šantin for his inspiring supervision, mentoring and helpful comments. I also thank you very much for your reading.

Abstrakt

Při prediktivním řízení je v každém časovém kroku řešen řídicí optimalizační problem (OCP), který je parametrizován měřením/odhadem současného stavu, vysledek je poté aplikován do řízeného objektu.

OCP může být výhodně reprezentová skrze kvadratické programování (QP). Obecně lze říci, že QP může být formulováno dvěma způsoby. Prvnímu způsobu se říká plné uspořádání, kde QP obsahuje menší počet optimalizovaných proměnných a žádné nulové prvky. Druhému způsobu se říká řídké uspořádání, zde je QP formulováno s více optimalizačními proměnými, ale zato s mnoha nulovými prvky s určitým rozmístěním.

V souvislosti s řízením procesů v reálném čase je potřeba řešit QP v každé časové periodě velmi rychle (řekněme v ms nebo dokonce μs). Proto za účelem snížení výpočetní náročnosti řešení QP, odpovídajícího lineárnímu prediktivnímu řízení (MPC) s horními a dolními omezeními, jsou v této práci představeny dva nové MPC aproximující přístupy:

První přístup (řídký) dovoluje využití specifické struktury MPC. Tato aproximace je založena na myšlence neuvažovat model dynamiky jako pevné omezení, ale raději modifikovat optimalizační kritérium v MPC tak, že každé nenaplnění dynamiky bude penalizováno. Navíc speciální řídká strukura daného problému aproximujícího zdola a shora omezeného MPC je využita při výpočtu gradientu a Newtonova kroku v metodě kombinující Newtonu/gradientní projekci. Je ukázáno na příkladech, že uvažovaná metoda je rychlejší nebo srovnatelně rychlá s ostatními řešiči známými z literatury, zatímco kvalita řízení je zachována.

Druhý přístup (plný) se zaměřuje na snížení počtu volnosti daného QP, protože hlavní slabinou plné formulace je ta, že čas potřebný k jejímu řešení roste kubicky s počtem vstupů a délkou predikčního horizontu. Proto je představen přístup aproximující část predikčního horizontu, a tím redukující velikost odpovídající QP formulace. Navíc je ukázáno, že přestože je pouze začátek horizontu uvažován přesně, kvalita řízení není ovlivněna příliš.

Abstract

In model predictive control optimal control problem (OCP) based on measurement/estimation of current state is solved each sampling time then the the result, inputs control is applied to the plant.

The OCP can be efficiently represented via quadratic programming (QP). In general, there are two types of QP formulation of MPC. The first is called *dense* where QP has smaller number of optimized variables and no zero entries. The second is called *sparse*, here QP is formulated with larger number of optimized variables but with many zero entries and sparsity structure occurs.

In order to control of real-time process applications solution for the QP solved each time should be found very quickly (let say in ms or even μs). Thus in order to reduce the computational complexity of the QP related to box constrained linear model predictive control (MPC) two novel approximations of MPC are introduced in this work:

The first one (sparse) which enables utilization of the MPC specific structure. This approximation is based on idea not to consider the model dynamics as hard constraint but rather modify the objective function of MPC to capture the violation of not fulfilling the model dynamics. Furthermore, the specific sparsity structure of the approximated box constrained MPC problem is exploited in the computation of gradient and Newton's step in the combined Newton/gradient projection algorithm. It is shown by an examples that the proposed method is faster or competitive to other state of the art solvers while retaining a performance level.

The second one (dense) aims on reducing degree of freedom in the QP since the bottleneck of a dense formulation is that its solving time growths cubically with number of inputs and length of prediction horizon. Thus approach which approximate part of the horizon and hence reduces size of the QP formulation is introduced. Moreover it is shown that although only beginning of horizon is assumed exactly overall controller performance will not be influenced much.

Contents

List of Figures

Chapter 1

Introduction

Model Predictive Control (MPC) is a multivariable control strategy which can naturally take into account physical limitation of controlled plant. It is a mathematical method which uses system model to predict its evolution and thus can be able to compute optimal control action.

In MPC at each sampling time optimal control problem often for finite horizon (usually formulated via Quadratic Programming (QP)) parametrized by current state measurement/estimation is solved. The solution, current control actions for inputs is then applied to the plant.

MPC overcomes traditional control methods with systematical approach to controller design for multiple-input multiple-output (MIMO) system and its ability to inter-corporate constraints handling. But the optimization problem arising from MPC has significant computational complexity, the main drawback of MPC and the reason why MPC use was limited to processes with slow dynamics.

First reported application of MPC on industrial process was in 70's [1] on slow sampled plant as a heuristic control technique without a mathematical background. Since the time computation power even of embedded systems rapidly increased and rigorous theory was built, the MPC is more and more popular even for such kind of low cost devices as e.g. power converters [2] in nowadays. Moreover in the last decades research on field of very fast sampled and large-scale systems was announced by [3, 4, 2, 5, 6].

In [7] it was shown that control law in traditional MPC can be expressed as a multi-parametric QP (mp-QP) and precomputed off-line. Authors also have presented algorithm for solving mp-QP which is used to obtain the explicit solution. The rest of (on-line) computational effort is just a searching in look-up table for adequate solution. Small-scale systems are then enabled to use fast sampling with period in order of ms or even μs. On the other hand, memory complexity grows exponentially with number of constraints for this method [7], the alternative algorithm was presented in [8] which for a class of MPC problems reduces both memory demand and computational time.

But there is a different keynote. Since model of the controlled plant is uncertain and also measurement/estimation of system states is inaccurate it does not make any sense to solve associated problem precisely. This idea has motivated research in direction of approximated MPC. Thus, only sub-optimal solution is found for which overall controller performance will be still good enough while computational time or memory demand of optimization task decrease dramatically.

This idea was used for deriving approximated explicit controllers e.g. in [9]. But this idea found its way also to on-line solvers. It was introduced by [10] where the active set strategy (AS) which additionally uses the homotopy of the explicit solution was terminated after certain number of iterations without necessary obtaining of exact solution. In [2] the Nesterov's type of fast gradient projection algorithm (GP) was established with certified minimal number of needed iterations to a given accuracy of objective function. Similar approach was also presented in [4] where a primal interior point method (IP) with fixed barrier parameter was terminated before the optimum was found.

Furthermore the approach of approximating the original objective function of MPC to reduce computation complexity or memory can be found in recent literature. For example see [11, 9, 6] where polytopic approximation of objective function is used to reduce complexity of explicit solution. The modification of objective function is used in [12] to decrease number of needed iterations in the combined Newton-like/gradient projection algorithm.

The authors of [13] reformulated original MPC problem to different variables using a singular value decomposition (SVD) of the problem Hessian. In [14] it was proposed to reformulate the MPC problem to one with reduced number of variables using SVD of simulation data.

It is known that AS methods are very effective in practice but involve many computationally demanding iterations when the active set of constraints needs to be changed a lot for example during the transient when the references are changing. On the other hand, for GP methods it was shown e.g. in [2] that they can rapidly identify a new active set of constraints but they involve many iterations when the problem is ill-conditioned which is the case for many practical implementation of MPC. The solution was introduced in [15, 16] where the GP method was combined with Newton's method.

This work consists two novel approaches. The first proposed method is based on the idea to modify the original MPC problem in a sense that the system dynamics is not as usual considered as hard equality constraint but rather as the soft constraint modeled via quadratic penalty with fixed weight in criterion. This is based on the observation of [4], where it was shown that the performance of sub-optimal controller was still comparable with the "exact" one even when the constraints on the system dynamics were not fulfilled. But here it will be shown that the controller performance would not degrade much if such an assumption is made at the beginning.

Furthermore, it is also shown next, that the resulting optimization problem has special sparsity structure. This structure can be exploited in the combined Newton/gradient projection algorithm [16] to reduce growth of computational complexity of gradient and Newton's step computation from cubic to linear with increasing prediction horizon. But contrary to the original work in [16], in this work the MPC problem is not formulated by dual variables but rather by primal, hence the result of each iteration is feasible with respect to constraints.

The second approach has been motivated by the idea that it makes sense to compute only the current control input because the rest of horizon will be discarded. Thus the idea is to split control horizon and reformulate optimization problem so only the beginning of prediction horizon will be taken exactly while the rest will be approximated by control policy with no constraints. This arise in optimization problem with chosen (smaller) size which corresponds to size of exactly taken part. Moreover it will be shown that although only beginning of horizon is constrained overall controller performance will not be influenced much. This method produces a dense optimization problem (with no zero entries) thus the structure of the problem is lost and hence any general QP solver can be used.

This work is organized as follows: In Chapter 2 an introduction to MPC is presented. In

Chapter 3 modern methods used for solving QP arising in MPC are introduced. Chapter 4 concerning our work. Particularly two novel approaches are proposed: 1) Sparse formulation for large-scale systems which assume system dynamics penalty and exploiting the problem structure; 2) Approximated dense MPC formulation framework which in limit case produce QP where only current control is computed. Thus this formulation should be especially suitable for embedded devices since a small amount of memory (and solving time of course) is required. Each approach is followed by numerical experiments proving their effectiveness. At the end of the work in Chapter 5 this work is summarized.

Appendix A includes recently introduced approach [17] very inspiring for our work since it also takes advantage of the Moore-Penrose pseudoinverse is employed therein. In Appendix B contain of appended CD is shown.

1.1 Notation

In this work, if not defined otherwise italic letters denote vectors or matrices (e.g. v, M), bolt italic denote vectors or matrices are defined by another vectors or matrices (e.g. $\boldsymbol{v}, \boldsymbol{M}$). Identity matrix is expressed by I. Positive definiteness is denoted by $Q > 0$ (resp. $Q \geq 0$ positive semi-definiteness) and for two vector relation operator (e.g. $\underline{x} < x$) has elementwise meaning. Constants are denoted by Greek letters (e.g. α, κ) and sets by calligraphic one (e.g. $\mathcal{A}, \mathcal{X}$). Especially two different discrete times are used in here: k denotes absolute time, while n denotes optimization time. Subscripts denote time instance (e.g. x_k, x_n) or jth component (e.g. u_j) and superscripts $(\cdot)^{(i)}$ denote iteration in algorithm loop. Moore-Penrose pseudoinverse is denoted by $(\cdot)^+$, optimal value by $(\cdot)^*$. And finally $\otimes$ denotes Kronecker product.

Chapter 2

Model Predictive Control

The idea of model predictive control (MPC) is to employ model of controlled plant. Current state is measured or more often estimated, then optimization problem parameterized by received data is computed. There are many variants how objective function for MPC can be defined. One way is to use of l_1, l_2 or l_∞ criterion which leads on linear programming.

Linear programming in MPC was investigate in past (e.g. [18, 19]) since it is much less computationally demanding. But it also suffers many practical drawbacks as it may yields either dead-beat or idle control performance and that may be unsuitable for process control applications [20]. Thus most often in control process applications quadratic objective function is assumed. Then the optimal problem can be efficiently formulated as a quadratic program (QP).

Once QP is solved the optimal control over prediction horizon is obtained but only the first control move is applied to the plant and other results are discarded. In the next period new current state is obtained and optimization for receded prediction horizon is executed. The principle that optimal control problem is computed each time-step is known as receding horizon control (RHC) and entails that resulting control becomes closed-loop.

Thus QP optimization task in MPC is needed to solve each time-step. It may be very time-consuming thus several solutions of MPC were developed, namely *online* where each time QP is solved online and *offline (or explicit)* where control law is derived and solution stored. Hence online computation is only simple evaluation of piecewise linear function defined explicitly [7, 9, 21]. *Hybrid* approach which provides trade-off between computational-time and memory demand has been introduced e.g. in [5] where they use piecewise affine (PWA) approximation of control law to warm-start the online optimization.

MPC is optimal control method and many different optimization criterion, control strategy has been studied and used in practice. For example the most common optimization criterion for regulator control problem used very often in literature is square of states and inputs with fixed weight. For tracking problem rather formulation with tracking error and control increments in optimization criterion is used. Weighting of control increments Δu cause that control strategy will be integrative-like [22]. Another approach where control policy minimizing settling time is proposed in [23].

Number of optimized variables of resulting QP grows linearly with horizon length. Moreover computational effort of solving such QP grows at least quadratically [16, 4]. Conditioning of the problem also determines its computational burden (issue of ill-conditioning is considered e.g. in [2]). Thus methods to decrease degree of freedom of resulting optimization problem are wanted. One of the methods so called input-blocking method [24] makes problem smaller by fixing inputs over several time-steps.

The stability of MPC can be guaranteed trough choosing of three essential ingredients. The ingredients are the *terminal cost function*, the *local control law*, the *terminal constraints* for which stability conditions must held [25]. Next, only an assumption that MPC is close-loop stabilizing is admitted.

In this chapter basics of *constrained finite-horizon linear quadratic regulator* (CFH-LQR) will be described in sense of one step of MPC on regulator problem. Then a few fundamental formulations of optimal control problem will be presented. Description of RHC follows as a concept which transform CFH-LQR open-loop policy into closed-loop one. At the end formulation of tracking problem is provided.

2.1 State-Space Model

This work is focused on discrete time linear time-invariant (LTI) systems only described by

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k, \end{aligned} \tag{2.1}$$

with states $x_k \in \mathbb{R}^{n_x}$, inputs $u_k \in \mathbb{R}^{n_u}$, outputs $y_k \in \mathbb{R}^{n_y}$ and known data as dynamics $A \in \mathbb{R}^{n_x \times n_x}$ as well as input $B \in \mathbb{R}^{n_x \times n_u}$, output $C \in \mathbb{R}^{n_y \times n_x}$ matrices. The $k \in \mathbb{N}^0$ denotes absolute discrete time.

2.2 Prediction

Predictive control uses prediction of system evolution over some finite horizon for decision of optimal control strategy. The prediction is parametrized by currently measured/estimated state x_k. State prediction for LTI (2.1) for prediction horizon with length N can be written by iterating (2.1) as (2.2) [22].

$$\boldsymbol{x} = \boldsymbol{v}x_0 + \boldsymbol{V}\boldsymbol{u}, \; x_0 = x_k \tag{2.2}$$

$$\boldsymbol{v} = \begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^N \end{bmatrix}, \; \boldsymbol{V} = \begin{bmatrix} B & & & & \\ AB & B & & & \\ A^2B & AB & B & & \\ \vdots & \vdots & \vdots & \ddots & \\ A^{N-1}B & A^{N-2}B & A^{N-3}B & \dots & B \end{bmatrix}, \tag{2.3}$$

where $\boldsymbol{x} = \left[x_1^T, x_2^T, \dots, x_N^T\right]^T \in \mathbb{R}^{N \cdot n_x}$, $\boldsymbol{u} = \left[u_0^T, u_1^T, \dots, u_{N-1}^T\right]^T \in \mathbb{R}^{N \cdot n_u}$ and with prediction matrices $\boldsymbol{v} \in \mathbb{R}^{N \cdot n_x \times n_x}$, $\boldsymbol{V} \in \mathbb{R}^{N \cdot n_x \times N \cdot n_u}$.

2.3 Constrained Finite-Horizon Linear Quadratic Regulator

Usually when MPC is introduced in literature only *constrained finite-horizon linear quadratic* (CFH-LQR) is described with assumption that RHC concept is applied. Thus by solving CFH-LQR which is open-loop problem based on new measurement in every time-step the overall strategy becomes closed-loop called MPC.

2.3.1 Problem Statement

It is defined general quadratic *initial cost function* l_0, *stage cost function* l_n and *terminal cost function* l_N to precise description of whole optimization horizon as

$$
\begin{aligned}
l_0(u_0) &= u_0^T R_0 u_0 + (2x_0^T S_0 + r_0^T)u_0 && (2.4a)\\
l_n(x_n, u_n) &= \begin{bmatrix} x_n \\ u_n \end{bmatrix}^T \begin{bmatrix} Q_n & S_n^T \\ S_n & R_n \end{bmatrix} \begin{bmatrix} x_n \\ u_n \end{bmatrix} + q_n^T x_n + r_n^T u_n && (2.4b)\\
l_N(x_N) &= x_N^T Q_N x_N + q_N^T x_N, && (2.4c)
\end{aligned}
$$

where $Q_n = Q_n^T \geq 0$ is state, $R_n = R_n^T > 0$ is input and $S_n \geq 0$ is cross weighting matrix. Optimization problem which need to be solved minimizes square power of states and inputs over optimization horizon subject to linear state and input constraints with respect to system dynamics. Such problem is stated as

$$
\begin{aligned}
\min \quad & l_0(u_0) + \sum_{n=1}^{N-1} l_n(x_n, u_n) + l_N(x_N)\\
\text{s.t.} \quad & x_{n+1} = Ax_n + Bu_n,\ x_0 = x_k \text{ is current state}\\
& E_n x_n + F_n u_n \leq f_n\\
& E_N x_N \leq f_N \text{ (or even } E_N x_N = e_N)\\
& n = 0, 1, \ldots, N-1. && (2.5)
\end{aligned}
$$

The problem (2.5) is very general but complex and hence is usually simplified. In many applications states and inputs cross weighting is not required or it is possible to find transformation such that $S_n = 0,\ n = 0 \ldots N-1$ [26]. In many application and also in this work linear terms are considered equal to 0.

Over the horizon it makes sense to have constant weights $Q_{n\neq 0} = Q,\ R_n = R, n = 0, \ldots, N-1$. Remind that current (or initial) state x_k is not minimized since it cannot be influenced by new control action.

Due to stability issue terminal cos is usually added [25]. Then for choosing Q_N several strategies are relevant: Q_N is the stabilizing solution of Riccati equation $A^T Q_N A - Q_N - A^T Q_N B(B^T Q_N B + R)^{-1} B^T Q_N A + Q = 0$ or Q_N is obtained from Lyapunov equation $A^T Q_N A - Q_N = Q$.

In (2.5) general state-input dependent polyhedron-shape constraints are assumed. But it shows in control practice that box-like limitation are usually sufficient [15]. Moreover hard state constraints can make problem infeasible thus they are usually assumed as soft to prevent controller are halted [27].

Furthermore if one accept assumption $S_n = 0,\ n = 0, 1, \ldots, N-1$ constraints split into state and input independent constraints and computational complexity is decreased significantly since it can be exploited by optimization algorithm [4].

Therefore simplified MPC minimization problem considered in this work is defined as

$$\begin{aligned} \min \quad & x_N^T Q_N x_N + \sum_{n=1}^{N-1} + x_n^T Q x_n + \sum_{n=0}^{N-1} + u_n^T R u_n \\ \text{s.t.} \quad & x_{n+1} = A x_n + B u_n \\ & \underline{x} \le x_n \le \overline{x} \\ & \underline{u} \le u_n \le \overline{u} \\ & n = 0, 1, \ldots, N-1. \end{aligned} \tag{2.6}$$

The constant vectors $\underline{x}$, $\underline{u}$ denotes lower bounds and $\overline{x}$, $\overline{u}$ upper bounds for states and inputs.

2.4 Basic Formulations

In this section two basic (exact) formulations of optimization problem (2.6) are shown. First, *primal*[1] *dense* formulation is shown and then formulation usually denoted as *sparse*. For comparison of these two formulation see e.g. [28].

Advanced concepts have been discovered recently. For example in [29] the sparse but compact approach was presented. The idea introduced therein is to specifically express the inputs as an affine function of the states such that resulting closed-loop dynamics matrix becomes nilpotent. Then Hessian of the proposed optimization problem will be compact and sparse.

2.4.1 Primal Dense Formulation

So called *primal dense*, *condensed* or *simultaneous* formualtion of QP is described here. By injecting (2.2) into (2.6) exact condensed QP problem is obtained as follows

$$\begin{aligned} \min \quad & \frac{1}{2} \boldsymbol{u}^T \boldsymbol{H} \boldsymbol{u} + \boldsymbol{f}^T \boldsymbol{u} \\ \text{s.t.} \quad & \underline{\boldsymbol{x}} - \boldsymbol{v} x_k \le \boldsymbol{V} \boldsymbol{u} \le \overline{\boldsymbol{x}} - \boldsymbol{v} x_k \\ & \underline{\boldsymbol{u}} \le \boldsymbol{u} \le \overline{\boldsymbol{u}} \end{aligned} \tag{2.7}$$

with lower bounds $\underline{\boldsymbol{u}} = \left[\underline{u}_0^T, \underline{u}_1^T, \ldots, \underline{u}_{N-1}^T\right]^T \in \mathbb{R}^{N \cdot n_u}$, upper bounds $\overline{\boldsymbol{u}} = \left[\overline{u}_0^T, \overline{u}_1^T, \ldots, \overline{u}_{N-1}^T\right]^T \in \mathbb{R}^{N \cdot n_u}$ and where Hessian matrix $\boldsymbol{H} \in \mathbb{R}^{N \cdot n_u \times N \cdot n_u}$ and linear part $\boldsymbol{f} \in \mathbb{R}^{N \cdot n_u}$ are

$$\boldsymbol{H} = \boldsymbol{V}^T \boldsymbol{Q} \boldsymbol{V} + \boldsymbol{R} \tag{2.8a}$$

$$\boldsymbol{f}^T = \boldsymbol{V}^T \boldsymbol{Q} \boldsymbol{v} x_k \tag{2.8b}$$

with weighting matrices $\boldsymbol{Q} = blkdiag(Q, Q, \ldots, Q_N)$ for states and $\boldsymbol{R} = blkdiag(R, R, \ldots, R)$ for inputs.

In this case the Hessian contains in general no zero entries (see (2.9)) hence the formulation is called dense or condensed. Due to high power order of A in the Hessian the *condition*

[1]Note that variable elimination applied on sparse formulation leads to dense one. We distinguish *primal* and *dual* dense (or condensed) formulation in dependency which variables are eliminated. If state variables are eliminated as it is usual we call the formulation *primal* otherwise if inputs are eliminated we call the formulation *dual*.

number[2] is relatively high in contrast with other formulations described later. The specific structure can be seen in Hessian

$$H = \begin{bmatrix} \sum_{n=0}^{N-1} B^T {A^n}^T Q {A^n}^T B^T + R & \sum_{n=0}^{N-2} B^T {A^{n+1}}^T Q A^n B & \dots & B^T {A^{N-1}}^T QB \\ \sum_{n=0}^{N-2} B^T {A^n}^T Q A^{n+1} B & \sum_{n=0}^{N-2} B^T {A^n}^T Q {A^n}^T B^T + R & \dots & B^T {A^{N-2}}^T QB \\ \vdots & \vdots & \ddots & \vdots \\ B^T Q A^{N-1} B & B^T Q A^{N-2} B & \dots & B^T QB + R \end{bmatrix}. \tag{2.9}$$

For this formulation input constraints are handled implicitly but simple form of state constraints is not preserved.

2.4.2 Sparse Formulations

Approaches discussed here are just straightforward transcription of considered minimization problem (2.6). It can be done in two ways, using two different ordering of optimization variables. For sake of completeness they are shown both in this section: 1) Ordering where input and state variables are grouped separately; 2) Ordering where states and inputs are alternating, integrated together.

In sparse formulations of MPC the optimization variables are states and inputs. Optimization variable vector of sparse optimization problem is denoted by $\boldsymbol{z}$ and is defined in following sections. Regardless the variables ordering degree of freedom is $n_z = N(n_x + n_u)$ in both cases (in contrast to primal condensed formulation where it was only $N \cdot n_u$). On the other hand sparse pattern in the Hessian matrix occurs and can be exploited.

2.4.2.1 Formulation with Separated States and Inputs

Next it is shown how problem (2.6) can be reformulated. The prediction (2.2) can be rewritten in different manner as

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} = \boldsymbol{d} \tag{2.10}$$

with appropriate matrices, summarized below

$$\boldsymbol{A} = \begin{bmatrix} -I & & & \\ A & -I & & \\ & A & -I & \\ & & & \ddots \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} B & & & \\ & B & & \\ & & B & \\ & & & \ddots \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} -Ax_k \\ 0 \\ 0 \\ \vdots \end{bmatrix}. \tag{2.11}$$

For the definition of box constraints in a way as were defined in previous section and for $\underline{\boldsymbol{x}} = \left[\underline{x}^T, \underline{x}^T, \dots, \underline{x}^T\right]^T \in \mathbb{R}^{N \cdot n_x}$ and $\overline{\boldsymbol{x}} = \left[\overline{x}^T, \overline{x}^T, \dots, \overline{x}^T\right]^T \in \mathbb{R}^{N \cdot n_x}$ optimization problem (2.6) can be rewritten into

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \frac{1}{2}\boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u} \\ \text{s.t} \quad & \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} = \boldsymbol{d} \\ & \underline{\boldsymbol{x}} \le \boldsymbol{x} \le \overline{\boldsymbol{x}} \\ & \underline{\boldsymbol{u}} \le \boldsymbol{u} \le \overline{\boldsymbol{u}}, \end{aligned} \tag{2.12}$$

[2] For symmetric positive definite (SPD) matrix A condition number κ can be defined as $\kappa = \frac{L}{\mu}$, where L denotes the biggest while μ the smallest eigenvalue of A.

where equality constraint denotes system dynamics and it is equivalent to (2.2).

In formalism of QP task the previous problem can be viewed as problem where minimization variable is $\boldsymbol{z} = \left[\boldsymbol{u}^T, \boldsymbol{x}^T\right]$, hence one can rewrite (2.12) as

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} \\ \text{s.t} \quad & \boldsymbol{C}\boldsymbol{z} = \boldsymbol{d} \\ & \underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}}, \end{aligned} \tag{2.13}$$

where by comparing (2.12) with (2.13) appropriate matrices are obtained in form

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{R} & \\ & \boldsymbol{Q} \end{bmatrix}, \ \boldsymbol{C} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A} \end{bmatrix}, \ \underline{\boldsymbol{z}} = \begin{bmatrix} \underline{\boldsymbol{u}} \\ \underline{\boldsymbol{x}} \end{bmatrix}, \ \overline{\boldsymbol{z}} = \begin{bmatrix} \overline{\boldsymbol{u}} \\ \overline{\boldsymbol{x}} \end{bmatrix}. \tag{2.14}$$

This kind of ordering is very illustrative and was used e.g. in [26] where it was also shown that the problem structure can be exploited even when the problem is formulated as a dense.

2.4.2.2 Formulation with Integrated States and Inputs

Here the idea used e.g. in [4] is followed. In this paper they used substitution where states and inputs are optimization variables, integrated together in sequence

$$\boldsymbol{z} = \left[u_k^T, x_{k+1}^T, u_{k+1}^T \ldots, u_{k+N-1}^T, x_{k+N}^T\right]^T \in \mathbb{R}^{n_z}. \tag{2.15}$$

With such ordering the optimization problem (2.6) can be stated in QP form as

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{C}\boldsymbol{z} = \boldsymbol{d} \\ & \underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}}, \end{aligned}$$

where appropriate matrices are defined as

$$\begin{gathered} \boldsymbol{H} = \begin{bmatrix} R & & & & \\ & Q & & & \\ & & R & & \\ & & & \ddots & \\ & & & & Q_N \end{bmatrix}, \ \boldsymbol{C} = \begin{bmatrix} B & -I & & & & & \\ & A & B & -I & & & \\ & & & \ddots & & & \\ & & & & A & B & -I \end{bmatrix}, \\ \underline{\boldsymbol{z}} = \begin{bmatrix} \underline{u}^T & \underline{x}^T & \underline{u}^T & \ldots & \underline{x}^T \end{bmatrix}^T, \ \overline{\boldsymbol{z}} = \begin{bmatrix} \overline{u}^T & \overline{x}^T & \overline{u}^T & \ldots & \overline{x}^T \end{bmatrix}^T. \end{gathered} \tag{2.16}$$

Remind (2.11), where $\boldsymbol{d} = \left[-A^T, 0^T, 0^T, \ldots\right]^T$.

This formulation is reasonable where advanced concepts as penalty method or barrier method are used since problem sparsity is preserved and can be exploited further [4].

2.5 Receding Horizon Control

Until now just open-loop optimal control problem was considered. To obtain feedback loop *receding horizon control* (RHC) concept is used. The idea of RHC is to solve open-loop optimal control problem every time period since new state measurement/estimation is available. Every time period optimization problem is solved and solution for whole prediction horizon is found but only first control action is applied to the plant. Next time new measurement is obtained and another open-loop optimization problem solved. Therefore it becomes a closed-loop control.

In Figure 2.5.1 principle of RHC is illustrated when also input-blocking [24] is considered. Then N_C denoted correction horizon after which control is assumed to be constant. Remind that this technique is often used to reduction of computational demand.

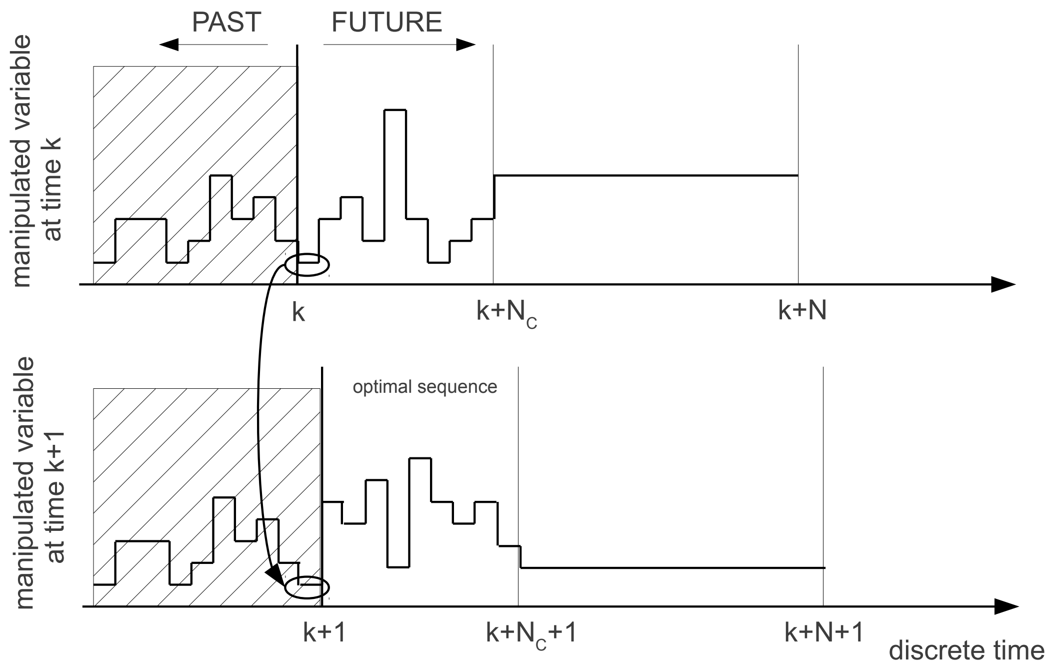


Figure 2.5.1: Receding horizon control - two time-steps.

2.6 Tracking Problem

This whole chapter has been describing regulator problem. Its definition, reformulation into various QP tasks and RHC concept on which MPC is based were shown. Here also CFH-LQR will be considered. But *tracking problem* with constant reference will be described here for further use in Chapter 4 where for purpose of demonstration rather tracking then regulator problem solution is shown.

2.6.1 Augmented State-Space Model

State-space reformulation for tracking problem with constant reference is described in this section. Hence augmented state-space model formulation consists of additionally reference and previous input (for the purpose of so called *incremental control* formulation [22] where control increments instead of control itself is optimized). Note that such prediction is parametrized by initial augmented state vector in other words by initial state, reference and input control from previous time-step thus augmented state vector length is $n_{\tilde{x}} = n_x + n_u + n_y$.

$$\begin{bmatrix} x_{k+1} \\ u_k \\ r_{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} A & B & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}}_{\tilde{\boldsymbol{A}}} \underbrace{\begin{bmatrix} x_k \\ u_{k-1} \\ r_k \end{bmatrix}}_{\tilde{x}_k} + \underbrace{\begin{bmatrix} B \\ I \\ 0 \end{bmatrix}}_{\tilde{\boldsymbol{B}}} \Delta u, \; \tilde{x}_k = \begin{bmatrix} x_k \\ u_{k-1} \\ r_k \end{bmatrix} \tag{2.17}$$

$$\tilde{y}_k = \underbrace{\begin{bmatrix} C & 0 & 0 \end{bmatrix}}_{\tilde{\boldsymbol{C}}} \tilde{x}_k \tag{2.18}$$

$$\tilde{e}_k = \underbrace{\begin{bmatrix} C & 0 & -I \end{bmatrix}}_{\tilde{\boldsymbol{C}}_e} \tilde{x}_k, \tag{2.19}$$

where $\tilde{x}_k \in \mathbb{R}^{N \cdot n_{\tilde{x}}}$, $\Delta u_k = u_k - u_{k-1}$ and $\tilde{y}_k, \tilde{e}_k \in \mathbb{R}^{n_y}$ thus known data are $\tilde{\boldsymbol{A}} \in \mathbb{R}^{n_{\tilde{x}} \times n_{\tilde{x}}}$, $\tilde{\boldsymbol{B}} \in \mathbb{R}^{n_{\tilde{x}} \times n_u}$ and $\tilde{\boldsymbol{C}}, \tilde{\boldsymbol{C}}_{\boldsymbol{e}} \in \mathbb{R}^{n_y \times n_{\tilde{x}}}$. Remind that $\tilde{e}_k = \tilde{y}_k - r_k$.

2.6.2 Prediction Based on the Augmented State-Space Model

Let describe prediction with augmented state-space model for output error obtained by iterating (2.19) and assumption of (2.17) as

$$\tilde{\boldsymbol{e}} = \boldsymbol{w}\tilde{x}_0 + \boldsymbol{W}\boldsymbol{\Delta u}, \; \tilde{x}_0 = \tilde{x}_k \tag{2.20}$$

$$\boldsymbol{w} = \begin{bmatrix} \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}} \\ \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}}^2 \\ \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}}^3 \\ \vdots \\ \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}}^N \end{bmatrix}, \; \boldsymbol{W} = \begin{bmatrix} \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{B}} & & & & \\ \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}} & \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{B}} & & & \\ \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}}^2\tilde{\boldsymbol{B}} & \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}} & \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{B}} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}}^{N-1}\tilde{\boldsymbol{B}} & \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}}^{N-2}\tilde{\boldsymbol{B}} & \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{A}}^{N-3}\tilde{\boldsymbol{B}} & \dots & \tilde{\boldsymbol{C}}_{\boldsymbol{e}}\tilde{\boldsymbol{B}} \end{bmatrix}, \tag{2.21}$$

where $\tilde{\boldsymbol{e}} = \left[\tilde{e}_0^T, \tilde{e}_1^T, \dots, \tilde{e}_{N-1}^T\right]^T \in \mathbb{R}^{N \cdot n_y}$ and where also augmented prediction matrices are $\boldsymbol{w} \in \mathbb{R}^{N \cdot n_y \times n_{\tilde{x}}}$, $\boldsymbol{W} \in \mathbb{R}^{N \cdot n_y \times N \cdot n_u}$.

2.6.3 Problem Statement

Since the objective of tracking problem is to track the reference criterion for the problem can be stated as

$$\min \quad \tilde{e}_N^T \tilde{Q}_N \tilde{e}_N + \sum_{n=1}^{N-1} \tilde{e}_n^T \tilde{Q} \tilde{e}_n + \sum_{n=0}^{N-1} \Delta u_n^T R \Delta u_n,$$

where $\tilde{Q}, \tilde{Q}_N \in \mathbb{R}^{n_y \times n_y}$ are tracking weighting matrices. Another requirement is usually to track asymptotically hence Δu is weighting and control strategy become integrative-like. For more details see [22]. Therefore reference tracking problem can be stated as

$$\begin{aligned}
\min \quad & \tilde{e}_N^T \tilde{Q}_N \tilde{e}_N + \sum_{n=1}^{N-1} \tilde{e}_n^T \tilde{Q} \tilde{e}_n + \sum_{n=0}^{N-1} \Delta u_n^T R \Delta u_n \\
\text{s.t.} \quad & x_{n+1} = A x_n + B u_n \\
& \underline{\tilde{e}} \leq e_n \leq \overline{\tilde{e}} \\
& \underline{u} \leq u_n \leq \overline{u} \\
& n = 0, 1, \ldots, N-1.
\end{aligned} \tag{2.22}$$

Chapter 3

Methods for Solving MPC Problem

Model predictive control can be formulated as a convex optimization problem for which many algorithms already exist. The most often problem is formulated as quadratic programming (QP) for which very efficient methods are known. By quadratic program is in this work mentioned optimization problem with quadratic objective function minimized subject to linear equality and inequality constraints.

Namely the two main approaches used to solve QP arising in MPC are active-set method (ASM) and interior-point (IP) method but also many variations of these algorithms were developed.

In short, active-set methods are those which solves a QP such that they are finding active set of the optimizer. Once the active set is found it is easy to compute the minimizer. Interior-point method are those which use some kind of barrier method to approximate the original problem by unconstrained one in every iteration. These related nonlinear problems are then solved in sequence.

In each implementation of such algorithm is opportune to use linear algebra library with already optimized performance to increase performance of the QP solver. The two very popular and widely used linear algebra libraries are Basic Linear Algebra Subroutines (BLAS) [30] and Linear Algebra PACKage (LAPACK) [31].

The former provides routines in three levels: Level 1) routines with algorithmic complexity $\mathcal{O}(n)$ such as computation of euclidean norm, copying vector, vector swap or scalar-vector multiplication; Level 2) routines with complexity $\mathcal{O}(n^2)$ such as matrix-vector multiplication, hermitian rank or symmetric rank computation; Level 3) routines with complexity $\mathcal{O}(n^3)$ such as matrix-matrix multiplication or solving triangular matrix with multiple right hand sides. All these routines are present in version for single, double, complex and double complex variables and utilized for different matrix types as banded, packaged, symmetric or general.

Similarly the latter, library providing routines for solving systems of simultaneous linear equations, least-squares problem of linear systems of equations, eigenvalue problems or singular value problems. Common matrix factorization as LU, Cholesky, QR, SVD or Schur are provided therein. Our implementations aspire to efficient use of these libraries.

Many different solvers which are focused on optimization in MPC were developed recently. Namely, solver *qpOASES* is an open-source C++ implementation of active-set method which is able to exploit sparsity structure which arises in some MPC formulations and it was used e.g. for control of a diesel engine [10]. Another open-source project, ACADO Toolkit, package of algorithms for automatic control (including MPC), state and parameter estimation etc. was developed in [32]. A Matlab toolbox called FiOrdOs, C-code generator of first-order methods was developed at ETH Zurich [33]. Code generator CVXGEN [34] for convex optimization problems was developed at Stanford University.

In this chapter several different method will be shown. First, in Section 3.1 primal barrier interior-point method is described. The rest of the algorithms in Sections 3.2, 3.3, 3.4 can be seen as a variation on active-set method.

Different notation to the rest of the work will be used here. Thus x does not denote state or u does not denote input anymore in this chapter.

3.1 Barrier Interior-Point Method

Interior-point methods are algorithms which are able to solve convex minimization problem subject to linear equality and even inequality constraints. Let us assume such convex optimization problem in a form

$$\min_x \quad \boldsymbol{f}(x) \quad \text{s.t.} \quad Ax \leq b, \quad Cx = d, \tag{3.1}$$

where $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^{p\times n}$, $d \in \mathbb{R}^p$ with number of inequality constraints m and equality constraints p are known data and where $\boldsymbol{f}(x) = (1/2)x^TGx + cx$ is convex quadratic function with $0 < G = G^T \in \mathbb{R}^{n\times n}$, $c \in \mathbb{R}^n$.

The idea is to solve sequence of equality constrained QP (EQP). Each of these EQP approximates original problem in the beginning of each iteration by barrier function. Such problem can be defined as

$$x^*(t) = \arg \quad \min_x \quad t\boldsymbol{f}(x) + \boldsymbol{\phi}(x) \quad \text{s.t.} \quad Cx = d, \tag{3.2}$$

where ${x^{(i)}}^*(t)$ is optimizer for the approximate problem (3.1) at ith iteration and $\boldsymbol{\phi}(x)$ denotes barrier function. Solutions of approximated problem (3.2) $x^*(t^{(i)})$ is denoted as *central points* and whole sequence $(x^*(t^{(i)}), i = 0, \dots, i_{max})$ as *central path*. Computation of $x^*(t^{(i)})$ every iteration starts from the previous central point. In the end the solution of entire QP is reached.

For this purpose starting point x for initialization optimization in next iteration must be updated. In [35] it was shown that solution will be $\epsilon-$optimal simply if t satisfies $(m/t) < \epsilon$. At the end of the iteration barrier parameter t is increased.

Parameter t denotes approximation accuracy. It is obvious that parameter t provides ratio of how much original objective and how much barrier function is weighted. For $t \to \infty$ more accurate central point is founded (centering). But exact computing of x^* is not necessary since even for inexact centering ${x^{(i)}}^*(t)$ the centering path converges to optimal point.

The exact barrier function $\boldsymbol{\phi}(x)$ can be expressed by indicator function. But such function is not differentiable in general and thus Newton's method cannot be applied [35]. Thus rather worse approximating logarithmic barrier is defined as

$$\boldsymbol{\phi}(x^{(i)}) = \sum_{j=1}^{m} -\log(b_j - a_j^T x),$$

where b_j are components of b and a_j^T are rows of A.

System of optimality conditions, so called KKT system of (3.2) has to be solved in the algorithm 1 each iteration. For this purpose several approaches how to solve KKT system can be found e.g. in [36].

The algorithm of primal interior-point method or also barrier method is summarized below.

Algorithm 1: Primal Barrier method [35]

```
1: Set feasible $x$, $t^{(0)} > 0$, $\mu > 1$, tolerance $\epsilon > 0$.
2: for $i = 0$ to $i_{\max}$ do
3:     Compute $x^*(t^{(i)})$ by solving KKT system of (3.2) which is initialized by previous
       central point $x$.
4:     Update $x = x^*(t^{(i)})$.
5:     if $(m/t) < \epsilon$ then
6:         Stop with $x^* = x$.
7:     end if
8:     Increase $t$; $t^{(i+1)} = \mu t^{(i)}$.
9: end for
```

Note that choice of parameter μ involves a trade-off between number of inner and outer iteration. Large μ means fewer outer iteration but more inner iteration etc.. In practise, parameter μ value is typically chosen in order a decades [35]. But also it has been observed that for purpose of MPC fixed barrier parameter with combination of fixed maximum number of iterations is sufficient then suboptimal solution may be obtained for medium sized QP in order to ms without significant loss of performance [37].

Also note that maximum number of iterations is guaranteed for this algorithm and it is a polynomial in the dimension and accuracy of the solution.

3.2 Active-Set Method

Algorithm of primal active-set method (ASM) shown in this section is more precisely described e.g. in [36].

In ASM also rather sequence of EQP is solved as in IP. But no approximation of inequality constraints is provided in ASM rather active set of the optimal point is identifying step by step. Once active-set is known it is easy to find problem solution by solving EQP subject to active (equality) constraints.

In order to simplification some of technique for equality constrained QP such e.g. Schur-complement or Null-space method can be used and general QP reduced into the form of (3.3) [36]. Thus main focus is to solve inequality QP (IQP) as follows

$$\min_x \quad \frac{1}{2}x^T Gx + cx \quad \text{s.t.} \quad a_j^T x \leq b_j, j \in \mathcal{I}. \tag{3.3}$$

In ASM problem (3.3) is usualy transform such that newton's step is computed so the problem can be reformualted as

$$p^{(i)} = \arg \min_p \quad \frac{1}{2}p^T Gp + g^{(i)}p \quad \text{s.t.} \quad a_j^T p = 0, j \in \mathcal{W}^{(i)}, \tag{3.4}$$

where $\mathcal{W}^{(i)}$ is *working set* at ith iteration defined such that consists all active constraints or $a_j^T x^{(i)} = b_j,\ j \in \mathcal{W}^{(i)}$ and $a_j^T x^{(i)} < b_j,\ j \notin \mathcal{W}^{(i)}$ and where the gradient is defined as $g^{(i)} = Gx^{(i)} + c$.

The subproblem (3.4) is need to be solved at each iteration it is expressed in terms of the Newton's step $p = x - x^{(i)}$. Then potential optimizer is then found by $x^{(i+1)} = x^{(i)} + \alpha^{(i)} p^{(i)}$. Step-size at ith iteration for each component is found by

$$\alpha^{(i)} = \min\left(1, \min_{j \notin \mathcal{W}^{(i)}, a_j^T p^{(i)} > 0} \frac{b_j - a_j^T x^{(i)}}{a_j^T p^{(i)}}\right). \tag{3.5}$$

In nontrivial case where some constraints are active in optimum algorithm works as follows:

Let have optimization variable x consist of components x_j. Then for problem minimizer $a_j^T x^* = b_j,\ j \in \mathcal{A}$ and $a_j^T x^* < b_j,\ j \notin \mathcal{A}$ where $\mathcal{A}$ is set of active constraints. For known active set the minimizer can be computed simply such that the problem subject to inactive constraints only is solved. Unfortunately such prior knowledge is always unknown and thus must be found step by step. For this purpose working set is defined and updated every iteration until the active set is reached.

At the beginning, algorithm start at position $x^{(0)}$ for which $a_j^T x^{(0)} = b_j,\ j \in \mathcal{W}^{(0)}$ and $a_j^T x^{(0)} < b_j,\ j \notin \mathcal{W}^{(0)}$. Then Newton's step is computed and two option may occur according to solution of (3.5):

$\alpha^{(i)} \neq 1$ Constraint j for which minimum in (3.5) is found is called *blocking constraint*[36]. The constraint must be added to the working set and new iteration is needed.

$\alpha^{(i)} = 1$ No constraint is violated. Then current position is tested for optimal (Karush-Kuhn-Tucker (KKT)) conditions. If the current point is KKT optimal denoted by x^* it is accepted and the Newton step is applied. Otherwise constraint which disallow us to find better solution must be removed from the working set.

The algorithm is summarized below followed by algorithm properties.

Algorithm 2: Active-Set Method for Convex QP [36]

```
1:  Set $\mathcal{W}^{(0)}$ such that it is subset of active constrain at $x^{(0)}$.
2:  for $i = 0$ to $i_{\max}$ do
3:    Solve (3.4) to find $p^{(i)}$
4:    if $p^{(i)} = 0$ then
5:      if $x^{(i)}$ satisfies KKT conditions then
6:        Stop with $x^* = x^{(i)}$
7:      else
8:        Remove constraint which prevent to find better solution
9:      end if
10:   else $\{p^{(i)} \neq 0\}$
11:     Compute $\alpha^{(i)}$ from (3.5)
12:     $x^{(i+1)} = x^{(i)} + \alpha^{(i)} p^{(i)}$
13:     if if $j$th constraint minimize (3.5) then
14:       $\mathcal{W}^{(i+1)}$ is obtained by adding $j$th constraint to $\mathcal{W}^{(i)}$
15:     else
16:       $\mathcal{W}^{(i+1)} = \mathcal{W}^{(i)}$
17:     end if
18:   end if
19: end for
```

Algorithm is simple, effective and suitable even for complex-shaped constraints. Its computational burden is typically $\mathcal{O}(n^2)$ (when updates are used in the factorization process during Newton step computation) where n is number of problem variables. Important drawback of this method is that only one constraint can be added/removed in working set each iteration. Thus when the initial working set is far away from the optimal one, algorithm converges in large number of iteration for large-scale problems.

Moreover variations as primal, primal-dual and dual ASM are known. For example dual ASM method tailored for MPC is presented in [10]. Algorithm proposed therein is able to exploit solution of previous optimization to improve its performance but in contrast to primal barrier method it is not provide feasible point at each iteration.

3.3 Fast Gradient Projection

The Gradient projection method (GP) was introduced in [38] as a generalization of the steepest descent method of optimization problems with convex constraints. In general, the GP solves problem

$$\min_x \quad \boldsymbol{f}(x) \quad \text{s.t.} \quad x \in \mathcal{X}, \tag{3.6}$$

where $f(x)$ is a continuously at least once differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathcal{X} \subseteq \mathbb{R}^n$ is a non empty closed convex set.

The projection of y onto $\mathcal{X}$ is the mapping $\mathcal{P} : \mathbb{R}^n \to \mathcal{X}$ defined by

$$\mathcal{P}(y) = \arg\min_x ||x - y|| \quad \text{s.t.} \quad x \in \mathcal{X}. \tag{3.7}$$

The gradient projection algorithm is then defined by

$$x^{(i+1)}(\alpha^{(i)}) = \mathcal{P}(x^{(i)} - \alpha^{(i)} \nabla \boldsymbol{f}(x^{(i)})), \tag{3.8}$$

where $\alpha^{(i)} > 0$ is the step-size, and $\nabla \boldsymbol{f}(x^{(i)})$ is the gradient of f at $x^{(i)}$.

Several line search algorithms for GP were introduced, namely generalized Armijo procedure [39] or the exact line search which search for the first local minimizer, called the Cauchy point x_C (see [36] for detailed explanation).

The main drawback of GP is that the computation of the projection is expensive for general type of constraints since it may lead to the QP in general [12]. On the other hand the projection on the set defined by box with lower and upper limits can be done very efficiently by component-wise median operation [12].

Since GP is based only on the gradient it inherits the convergence from the steepest descend method hence for the problems which are ill-conditioned it needs many iterations to converge. Thus there is shown modification of original method, fast gradient method [2] based on Nesterov's gradient projection method [40]. Similar gradient method for MPC was improved in [41] where problem structure was exploited in gradient computation.

Let us assumed minimization problem (3.6) in simplified form with quadratic criterion as follows

$$\min_x \frac{1}{2} x^T H x \quad + \quad f^T x, \; x \in \mathcal{X}, \tag{3.9}$$

where $0 < H = H^T \in \mathbb{R}^{n \times n}$, $f \in \mathbb{R}^n$ and where $\mathcal{X}$ is box constrained feasible set defined such that $\underline{x} \le x \le \overline{x}, \; \forall x \in \mathcal{X}$.

In contrast with traditional gradient method this method is not descent anymore. Thus $f(x^{(i+1)}) < f(x^{(i)}), \forall i \in \mathbb{N}^0$ but rather condition for descent sequence is considered $\lambda_j \to 0$ and $\phi_j(x) \geq (1-\lambda_j)f(x) + \lambda_j \phi_0(x), \forall j \geq 0, \forall x \in \mathcal{X}$ where $\{\phi_j(x)\}_0^\infty$ and $\{\lambda_j\}_0^\infty$ are called estimate sequences.

In the algorithm fixed step-size $-(1/L)$ is used with no addition computational effort although it has been proved that it has same rate of convergence as other step-size rules [42]. Lipschitz parameter $L = \lambda_{\max}(H)$ and convexity parameter $\mu = \lambda_{\min}(H)$ have to be known.

Briefly summarize, at the beginning initial values are set. Additionally, $\alpha^{(i)}, \beta^{(i)}, i = 1, \ldots, i_{max}$ can be precomputed (computed *offline*) for time saving as was shown in [43]. First of all, gradient $\Delta \boldsymbol{f}^{(i)} = Hx^{(i)} + f^{(i)}$ is computed and $x_f^{(i)}$ found. In the next step simple projection by median function applied element-wisely is used and feasible adept to be optimum found. Then algorithm check this point $x^{(i+1)}$ on KKT condition and stops if positive otherwise initial position for next iteration $y^{(i+1)}$ is calculated. Parameter $\alpha^{(i+1)} \in (0,1)$ is computed from $(\alpha^{(i+1)})^2 = (1-\alpha^{(i)})(\alpha^{(i)})^2 + \mu\alpha_{i+1}/L$ in original [43].

Algorithm is stopped when optimum is found (KKT conditions are satisfied). But usually many iteration to converge is needed and KKT checking computational complexity may not be neglected. Thus fixed iterations are preferred for real-time application. Minimal count of iteration necessary for given ϵ-optimal solution can be computed for this method [43].

The algorithm of fast gradient method is summarized lower.

Algorithm 3: Fast gradient projection for box constrained convex minimization [43]

```
1: Set feasible $y^0 = x^0 \in \mathcal{X}$, $0 < \sqrt{\frac{\mu}{L}} \leq \alpha_0 < 1$
2: for $i = 1$ to $i_{\max}$ do
3:     $x_f^{(i)} = y^{(i)} - (1/L)\Delta \boldsymbol{f}(y^{(i)})$
4:     $x^{(i+1)} = \text{median}(\underline{x}, {x_f}^{(i)}, \overline{x})$
5:     if $x^{(i+1)}$ satisfies KKT conditions then
6:         Stop with $x^* = x^{(i+1)}$
7:     end if
8:     $\alpha^{(i+1)} = (-(\alpha^{(i)})^2 L + \mu + \sqrt{4(\alpha^{(i)})^2 L^2 + ((\alpha^{(i)})^2 L - \mu)^2})/2L$
9:     $\beta^{(i)} = \alpha^{(i)}(1-\alpha^{(i)})/((\alpha^{(i)})^2 + \alpha^{(i+1)})$
10:    $y^{(i+1)} = x^{(i+1)} + \beta^{(i)}(x^{(i+1)} - x^{(i)})$
11: end for
```

Note that since GP is gradient based, the speed of convergence depends heavily to the problem conditioning. To overcome this technique of preconditioning which provides coordinate transformation $z = T^{-1}x$, where $T \in \mathbb{R}^{N \cdot n_u \times N \cdot n_u}$ is diagonal SPD matrix (preconditioner) are used. Hessian in new coordinates z will be $H_{new} = T^T HT$. Moreover, optimal transformation T^* can be found by solving $\min_T \frac{\lambda_{max}(H_{new})}{\lambda_{min}(H_{new})}$ [44].

3.4 Combined Newton/Gradient Projection

The bottleneck of gradient projection method is that it finds optimum in large number of iterations. Whereas in [45] gradient projection technique which find solution for bounded quadratic problem with a thousands of variables in a few iterations (less than 15) system was introduced. Also the algorithm which combines gradient projection with Newton-like methods were introduced in [15] and [16].

These methods use an *Newton's step* of result of iteration (3.8) by solving the following problem for better convergence rate [16] whereas gradient projection accelerate identification of optimal active set.

In the algorithm at the beginning gradient initialized by $x^{(i)}$ is computed and direction of negative gradient is projected further. The solution of projection is denoted as Cauchy point $x_C^{(i)}$. Next improvement step $x_+^{(i)}$, Newton's step is obtained by solving

$$x_+^{(i)} = \arg\min_{x^{(i)}} \boldsymbol{f}(x^{(i)}) \tag{3.10a}$$

$$\text{s.t.} \quad x_j^{(i)} = x_{Cj}^{(i)}, \quad j \in \mathcal{W}^{(i)}, \tag{3.10b}$$

where $\mathcal{W}^{(i)}$ denotes the working set of active constraints in the Cauchy point $x_C^{(i)}$ at ith iteration of the algorithm. The Cauchy point $x_C^{(i)}$ denotes the first local minimizer found in this case by exact line search along the procedure path.

Since the result of (3.10) can be infeasible with respect to constraint $x_+^{(i)} \in \mathcal{X}$, the direction form Cauchy point towards $x_+^{(i)}$ is projected onto feasible set by the same technique as the negative gradient is in the gradient projection step by

$$x_N^{(i)}(\beta^{(i)}) = \mathcal{P}(x^{(i)} + \beta^{(i)} p^{(i)}), \tag{3.11}$$

where $p^{(i)} = x_+^{(i)} - x_C^{(i)}$ is the Newton's step for problem (3.10) and $\beta^{(i)}$ is found nor by median function as in previous section but rather exact line search procedure (see [36] for details) is used since it cause better convergence and solution is denoted the first local minimizer along the projected path. Furthermore modification of exact line search method such that residuum of the Newton's projection is exploited may yields in even better convergence rate. Method is summarized in Algorithm 4.

Algorithm 4: Combined Newton/Gradient Projection Algorithm [16]

```
 1: Set feasible $x^{(0)}$ ans $i_{\max} \in \mathbb{N}$.
 2: for $i = 0$ to $i_{\max}$ do
 3:   if $x^{(i)}$ satisfies the KKT conditions then
 4:     Stop with $x^* = x^{(i)}$
 5:   end if
 6:   Gradient Step: Starting from $x^{(i)}$ find gradient step.
 7:   Gradient Projection: Project gradient to find the Cauchy point $x_C^{(i)}$ by (3.8).
 8:   Newton's Step: Find an improvement point $x_+^{(i)}$ by solving (3.10).
 9:   Newton's Projection: Starting from $x_C^{(i)}$ project $p^{(i)} = x_+^{(i)} - x_C^{(i)}$ to find first local
      minimizer $x_N^{(k)}$ by (3.11).
10:   $x^{(i)} = x_N^{(i)}$
11: end for
```

In [16] this algorithm but in combination with dual variables was shown. This algorithm with tailored Newton's step and gradient step exploiting problem structure was also used in our work and the modifications of Newton's/gradient steps will be described latter as well as algorithm properties which will be also demonstrated by examples.

Chapter 4

Approximated MPC

In nowadays computational power of embedded systems is good enough that various optimization problems can be solved online. But also there is need of controllers for extremely fast systems for which even powerful micro-controllers is not sufficient solution and faster and faster algorithms have to be developed. One way how to obtain them is to approximate the original problem.

These techniques are based on the idea that since model of the controlled plant is uncertain, system states are measured/estimated inaccurately and various disturbances effect on our real system it does not make sense to make exact control law and rather perform some faster approximated one instead.

Since computational effort needed to solve control optimization problem very depends on its formulation it makes sense to study how problem can be formulated properly. It already has been shown two basic formulation - primal dense and sparse in Section 2.4.

Dual dense formulation where states instead of inputs are optimized was introduced in [17] recently (briefly described in Appendix A). It seems to be complementary formulation to the other two basic formulation. But thanks to the approximation it can be solved faster then other two exact formulation.

Another approximate schema has been introduced in [4] in context of primal barrier interior-point method. It has been shown that even such approximate schema may has good performance for fast sampled systems (in order of ms).

In this section two novel approaches of fast MPC are introduced. The first method is inspirited by [4] where it was shown that even if each QP is solved suboptimally the controller performance is still good enough even without fulfilled dynamics constraints of the model. But in contrary to [4] in this work constraints of system dynamics are assumed as soft from the beginning and rather combined Newton/gradient method is used instead of IP.

The second method on the other side is approach following the condensed MPC methodology (e.g. [15]) inspired by [17] mentioned before. The improved method proposed here is aimed to decrease number of variables of resulting optimization problem and thus can be viewed as an alternative to input blocking (see [24]).

For both novel methods it is shown that with assumed approximations computational time demand is reduced and resulting control policies are thus suitable for real-time control. For the latter approach also less amount of memory is required.

4.1 Dynamic Penalty Method

The proposed method is based on the idea to modify the original MPC problem in a sense that the system dynamics is not as usual considered as hard equality constraint but rather as the soft constraint modeled as quadratic penalty with fixed weight. This is based on the observation of [4], where it was shown that the performance of suboptimal controller was still comparable with the "exact" one even when the constraints on the system dynamics were not fulfilled. But here it will be shown that the controller performance would not degrade much if such an assumption is made at the beginning. Also it will be shown that the structure of such approach can be exploited in Algorithm 4. Preliminary results of this section were presented in [46].

4.1.1 Exact Problem

Let us to remind the sparse formulation (Section 2.4.2). For optimization variable $\boldsymbol{z} = \left[u_0^T, x_1^T, u_1^T, \ldots, x_N^T\right]^T \in \mathbb{R}^{n_z}$ problem of MPC (2.6) can be rewritten into the common form of quadratic program as

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & J(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{f} \\ \text{s.t.} \quad & \boldsymbol{C}\boldsymbol{z} = \boldsymbol{d} \\ & \underline{\boldsymbol{z}} \le \boldsymbol{z} \le \overline{\boldsymbol{z}} \end{aligned} \tag{4.1}$$

with known data

$$\boldsymbol{H} = \begin{bmatrix} R & & & & \\ & Q & & & \\ & & R & & \\ & & & \ddots & \\ & & & & Q_N \end{bmatrix}, \quad \boldsymbol{f} = \boldsymbol{0}$$

$$\boldsymbol{C} = \begin{bmatrix} B & -I & & & \\ A & B & -I & & \\ & & \ddots & & \\ & & A & B & -I \\ & & & & E \end{bmatrix}, \quad \boldsymbol{d} = \begin{bmatrix} -Ax_k \\ 0 \\ \vdots \\ 0 \\ e \end{bmatrix}$$

$$\underline{\boldsymbol{z}} = \begin{bmatrix}\underline{\boldsymbol{u}}^T & \underline{\boldsymbol{x}}^T & \underline{\boldsymbol{u}}^T & \ldots & \underline{\boldsymbol{x}}^T\end{bmatrix}^T, \quad \overline{\boldsymbol{z}} = \begin{bmatrix}\overline{\boldsymbol{u}}^T & \overline{\boldsymbol{x}}^T & \overline{\boldsymbol{u}}^T & \ldots & \overline{\boldsymbol{x}}^T\end{bmatrix}^T. \tag{4.2}$$

Remember that constraint $Ex_N = e$ with $E \in \mathbb{R}^{n_x \times n_x}$ is provided due to stability enforcement [41].

4.1.2 Approximated Problem

Let define residuum of not fulfilled the equality constraint in (4.1) as

$$\boldsymbol{r}(\boldsymbol{z}) = \boldsymbol{d} - \boldsymbol{C}\boldsymbol{z}.$$

Hereafter the equality constraint will be omitted, instead every deviation will be penalized by added quadratic term in new criterion

$$J(\boldsymbol{z}; \boldsymbol{\Psi}) = \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{f} + \boldsymbol{r}(\boldsymbol{z})^T\boldsymbol{\Psi}\boldsymbol{r}(\boldsymbol{z}), \tag{4.3}$$

where $\boldsymbol{\Psi} = blkdiag(P, \dots, P) \in \mathbb{R}^{(N+1)n_x \times (N+1)n_x}$ is penalty parameter with $P \in \mathbb{R}^{n_x \times n_x}$, $P = P^T \succ 0$.

See that with $\boldsymbol{\Psi} \to \infty$ system dynamic behavior (2.1) will be assumed more strictly since only additional term will be minimized. On the other hand with $\boldsymbol{\Psi} \to 0$ system dynamics become more soft hence can be violated. Note that penalty $\boldsymbol{\Psi}$ also affects the problem conditioning. Thus dynamics softness must be choose properly.

Next, (4.1) can be rewritten into

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & J(\boldsymbol{z}; \boldsymbol{\Psi}) = \frac{1}{2}\boldsymbol{z}^T \boldsymbol{G} \boldsymbol{z} + \boldsymbol{z}^T \boldsymbol{h} \\ \text{s.t.} \quad & \underline{\boldsymbol{z}} \le \boldsymbol{z} \le \overline{\boldsymbol{z}} \end{aligned} \tag{4.4}$$

with structured Hessian

$$\begin{aligned} \boldsymbol{G} = \boldsymbol{H} + \boldsymbol{C}^T \boldsymbol{\Psi} \boldsymbol{C} = & \\ = & \begin{bmatrix} \boldsymbol{\phi_a} & \boldsymbol{\phi_e} & & & & & & \\ \cdot & \boldsymbol{\phi_b} & \boldsymbol{\phi_c} & \boldsymbol{\phi_d} & & & & \\ & \cdot & \boldsymbol{\phi_a} & \boldsymbol{\phi_e} & & & & \\ & \cdot & \cdot & \boldsymbol{\phi_b} & \cdots & & & \\ & & & \vdots & \ddots & \vdots & & \\ & & & & \cdots & \boldsymbol{\phi_b} & \boldsymbol{\phi_c} & \boldsymbol{\phi_d} \\ & & & & & \cdot & \boldsymbol{\phi_a} & \boldsymbol{\phi_e} \\ & & & & & \cdot & \cdot & \boldsymbol{\phi_f} \end{bmatrix} \end{aligned} \tag{4.5}$$

$$\begin{aligned} \boldsymbol{\phi_a} &= R + B^T P B & \boldsymbol{\phi_d} &= -A^T P \\ \boldsymbol{\phi_b} &= Q + P + A^T P A & \boldsymbol{\phi_e} &= -B^T P \\ \boldsymbol{\phi_c} &= A^T P B & \boldsymbol{\phi_f} &= Q_N + P + F^T P F \end{aligned}$$

and appropriate linear part $\boldsymbol{h} = \boldsymbol{f} - \boldsymbol{C}^T \boldsymbol{\Psi} \boldsymbol{d}$.

Note that number of square blocks in the Hessian is equal to length of optimization horizon N. Note also that Hessian has specific sparse structure suitable to be exploited in computations of gradient or/and Newton's step. This will be presented in following Section 4.1.3.1.

4.1.3 Algorithm Tailoring for Approximated MPC

In this section description how problem sparsity with specific structure can be exploited in context of combined Newton/gradient projection algorithm (described in Section 3.4). Note that the most computational demanding part of Algorithm 4 is step where computation of the Newton's step is provide. Which in case of general QP leads to solution of set of linear equations. The solution is usually obtained by the factorization approach using Cholesky or LDL factorization followed by forward and back substitution. Hence the complexity of the solution of (3.10) growths cubically in the number of optimization variables, thus also in the prediction horizon.

In [16] the method based on Riccati recursion was used for reducing the complexity of Newton's step computation to growth only linearly in the prediction horizon if dual QP problem is solved. Similar was achieved in [4] where the IP method which exploited the sparsity structure of sparse MPC was introduced.

On the other hand in [15] there was introduced an approach based on null space method (see e.g. [36]) for box constrained primal dense formulation of MPC which reduces the number of variables in (3.10) to those which are inactive, but the cubic dependence to prediction horizon was preserved.

In the following, the algorithm which can take the best from both last mentioned approaches is presented. It uses null space method to eliminate the variables which are currently on their boundaries and benefits from the sparse structure of the problem to obtain linear growth of complexity in prediction horizon.

4.1.3.1 Treating Hessian Structure

The computation of the gradient for the cost function in (4.4) requires approximately order n_z^2 flops. In contrast computation of Newton's step has order n_z^3 flops. Thus the most challenging is to minimize computational burden for Newton's step nevertheless time-saving variants of other methods are mentioned below.

Gradient Computation In gradient computation the matrix-vector multiplication $\boldsymbol{Gz}$ is crucial. Thus generic solver carry out this operation in $2N^2(n_u+n_x)^2$ flops. On the other hand by utilizing the *block-tridiagonal* structure of (4.5), this can be reduced to $N(2n_u^2+6n_x^2+8n_un_x)$ flops, here only the linear growth of complexity in horizon is obtained.

Newton's Step Computation Two things are provided to computational saving: 1) Hessian size reduction by null-space method; 2) Computation of size-reduced Newton's step with exploiting the Hessian structure.

Former one is based on idea that the Newton's step should be zero in directions where the constraints are active. Therefore the reduced Hessian $\bar{\boldsymbol{G}} = {Z^{(i)}}^T \boldsymbol{G} Z^{(i)}$ and reduced gradient $\bar{\boldsymbol{g}} = {Z^{(i)}}^T \nabla J(\boldsymbol{z}^{(i)};\boldsymbol{\Psi})$ can be defined, where $Z \in \mathbb{R}^{n_z-n_a\times n_z-n_a}$ is null-space of inequalities constraints matrix (for details of null-space method see [36]) and n_a is the number of active constraints.

Then reduced Newton's step $\bar{p}$ including elements $\bar{p}_j \in \mathcal{W}^{(i)}$ where $\mathcal{W}^{(i)}$ is the working set in ith iteration of Algorithm 4 and it can be computed by solving of

$$\bar{\boldsymbol{G}}\bar{p}^{(i)} = -\bar{\boldsymbol{g}}. \tag{4.6}$$

When $\bar{p}$ is obtained one can compute original full-sized Newton's step $p^{(i)} = {Z^{(i)}}^T \bar{p}^{(i)}$.

Furthermore it has been shown in [15] that for box constrained problem it is not needed to explicitly form the null space matrix $Z^{(i)}$, but reduced Hessian and gradient can be obtained directly by only removing the rows and columns corresponding to currently active constraints from the Hessian of the original problem (4.4). Then the new Hessian $\bar{\boldsymbol{G}}$ order is reduced to $n_r = n_z - n_a$.

Direct methods for efficient solving (4.6) - Cholesky factorization $\bar{\boldsymbol{L}}\bar{\boldsymbol{L}}^T = \bar{\boldsymbol{G}}$, forward-substitution $\bar{\boldsymbol{L}}\bar{y} = -\bar{\boldsymbol{g}}$ and back-substitution $\bar{\boldsymbol{L}}^T\bar{p}^{(i)} = \bar{y}$ must be computed. Note that

Cholesky factor $\bar{\boldsymbol{L}}$ of reduced Hessian $\bar{\boldsymbol{G}}$ has also sparsity structure as follows

$$\bar{\boldsymbol{L}} = \begin{bmatrix} \bar{L}_{11} & & & & & \\ \bar{L}_{21} & \bar{L}_{22} & & & & \\ & \bar{L}_{32} & \bar{L}_{33} & & & \\ & & & \ddots & & \\ & & & & \bar{L}_{N-1,N-1} & \\ & & & & \bar{L}_{N,N-1} & \bar{L}_{N,N} \end{bmatrix}. \tag{4.7}$$

This structure can be exploited by the following algorithm (see e.g. [28, 4]):

$$\begin{aligned} \bar{L}_{11}\bar{L}_{11}^T &= \bar{G}_{11} \\ \bar{L}_{21}\bar{L}_{11}^T &= \bar{G}_{21} \\ \bar{L}_{22}\bar{L}_{22}^T &= \bar{G}_{22} - \bar{L}_{21}\bar{L}_{21}^T, \text{ and so on...} \end{aligned}$$

Such factorization can be handled with $N(1/3n_u^3 + 10/3n_x^3 + 3n_u^2 n_x + 6n_x^2 n_u)$ flops in worst case when no constraint is active, instead of $1/3N^3(n_x + n_u)^3$ flops when whole matrix is decomposed, see that cubic dependency to prediction horizon is reduced to linear only.

Also substitutions is possible to optimize. For lower factor (4.7) The forward substitution can be written straightforwardly as

$$\begin{aligned} \bar{L}_{11}y_1 &= -\bar{g}_1 \\ \bar{L}_{22}y_2 &= -(\bar{g}_2 + \bar{L}_{21}\bar{y}_1) \\ \bar{L}_{33}y_3 &= -(\bar{g}_3 + \bar{L}_{32}\bar{y}_2), \text{ and so on...} \end{aligned}$$

For the sake of completeness even tailored method of back substitution is mentioned below.

$$\begin{aligned} \bar{L}_{N,N}^T\bar{p}_N^{(i)} &= \bar{y}_N \\ \bar{L}_{N-1,N-1}^T\bar{p}_{N-1}^{(i)} &= \bar{y}_{N-1} + \bar{L}_{N,N-1}^T\bar{p}_N^{(i)} \\ \bar{L}_{N-2,N-2}^T\bar{p}_{N-2}^{(i)} &= \bar{y}_{N-2} + \bar{L}_{N-1,N-2}^T\bar{p}_{N-1}^{(i)}, \text{ and so on...} \end{aligned}$$

Then each substitution required $N(3n_x^2 + n_u^2 + 4n_u n_x)$ flops in contrast to $N^2(n_u + n_x)^2$ flops if the matrices are dense.

Exact Line Search See [36] for details of exact line-search method for box constrained problems. This method is provided in Algorithm 4 twice each iteration. In the line-search method computation of product $\bar{d}^T\bar{\boldsymbol{G}}\bar{d}$ where $\bar{d}$ is a direction of search (in Algorithm 4 it can be either the negative gradient or Newton's step) the most computationally demanding part. Here also the same procedure as for gradient computation can be used hence the computation can be done with $\mathcal{O}(N(n_x^2 + n_u^2))$.

4.1.4 Numerical Experiments

In this section the performance of proposed method is shown on two examples: oscillating masses and random systems (for its origin see [4]). All numerical tests were executed on computer with Intel®Core™2 Duo CPU P8400@2.26GHz, running Linux. Implementation was written as a C-MEX function for Matlab environment. Subroutines from BLAS and LAPACK libraries were employed to carry out linear algebra operations. Time was measured by *clock_gettime*() function with up to nanosecond resolution.

For all tests the default parameters of MPC are used: $R = I$, $Q = I$, $Q_N = I$, $E = 0$.

4.1.4.1 Oscillating Masses Simulation

Oscillating masses model is represented by six masses connected by string between walls. It is described via states $x \in \mathbb{R}^{12}$ as masses horizontal position, velocity and controlled by inputs $u \in \mathbb{R}^3$ as forces work between some of this masses (see Figure 4.1.1). Each mass position saturates in ± 4 from its natural steady state position. Control limitation is stated on ± 0.5. Each of the masses is perturbed by unknown disturbance in range ± 0.5. The regulator problem is solved, i.e. stabilization of each mass in its origin.

The following test run simulations over 1000 steps on a random setting of oscillating masses model. The horizon length was fixed $N = 10$ for all simulations.

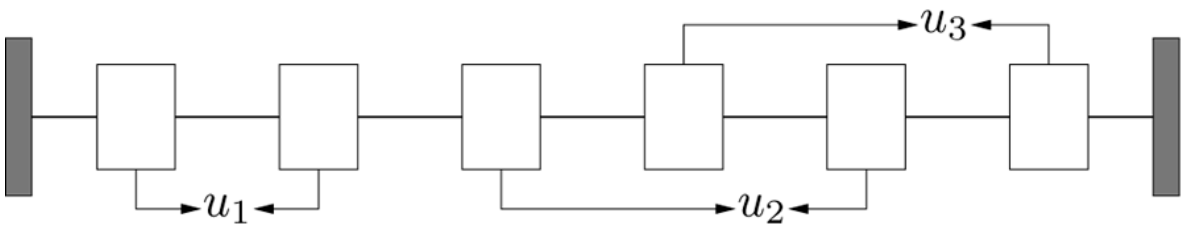
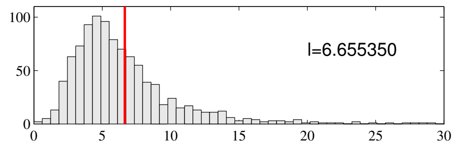


Figure 4.1.1: Oscillating masses model [4]. *Six masses between the wall. Force acting between some of masses is denoted by arrows.*

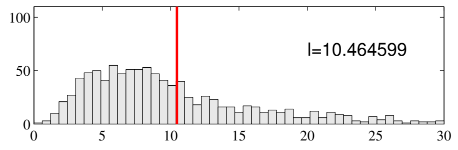
In Figure 4.1.2 and Figure 4.1.3 histogram of the stage costs

$$l_k = x_k Q x_k + u_k R u_k \tag{4.8}$$

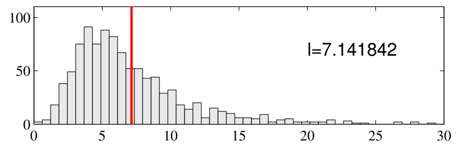
of the simulations ($k = 1, \ldots, 1000$) are shown. Mean value of stage cost l is computed as mean value of stage costs l_k over the whole horizon. Exact solution is found by solving (2.16) by *quadprog* routine from Optimization Toolbox in Matlab.



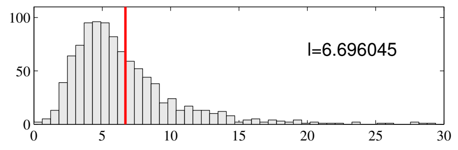
(a) Exact MPC



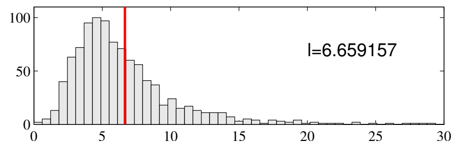
(b) $\Psi = 10^1 I$



(c) $\Psi = 10^2 I$

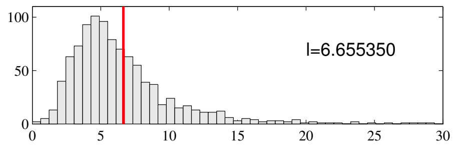


(d) $\Psi = 10^3 I$

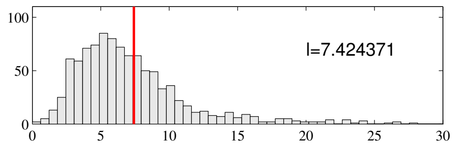


(e) $\Psi = 10^4 I$

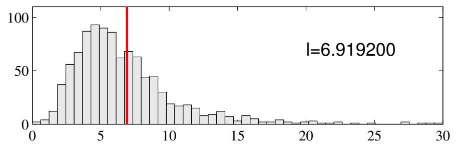
Figure 4.1.2: Stage cost histograms for simulation of oscillating masses - influence of the penalty parameter Ψ and unbound maximum number of iteration ($i_{max} = \infty$). *The solid red line donates mean of the distribution. The number in upper right hand side corner denotes mean value of stage costs.*



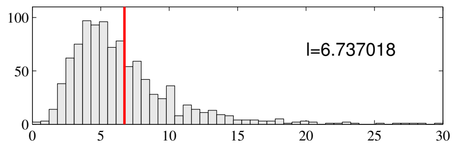
(a) Exact MPC



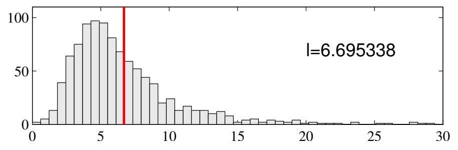
(b) $i_{max} = 2$



(c) $i_{max} = 3$



(d) $i_{max} = 5$



(e) $i_{max} = 10$

Figure 4.1.3: Stage cost histograms for simulation of oscillating masses - influence of the maximum number of iteration i_{max} of Algorithm 4 with fixed penalty parameter $\Psi = 10^3 I$. *The solid red line donates mean of the distribution. The number in upper right hand side corner denotes mean value of stage costs.*

Exact problem was approximated by penalty with fixed $\mathbf{\Psi}$. Our method in the first case solves each QP to the optimal regardless to how many iterations it cost. It is evident that with increasing the penalty parameter $\mathbf{\Psi}$ the mean value of the solution is closer to the exact one (Figure 4.1.2).

In Figure 4.1.3 maximum number of iterations instead of penalty parameter was tuned and penalty parameter was fixed $\Psi = 10^3 I$. It can be seen that for the testing system maximum number of iterations $i_{max} = 5$ could be set for the algorithm (4) in controller to get approximately constant time demand with minimum impact to controller quality.

From the results in Figure 4.1.2 we established for the system that the penalty parameter $\Psi = 10^3 I$ is sufficient. Signal of one state and one control input for this penalty parameter are just for the sake of completeness shown in Figure 4.1.4 where each problem were solved to optimality. It can be seen that resulting control has almost the same quality.

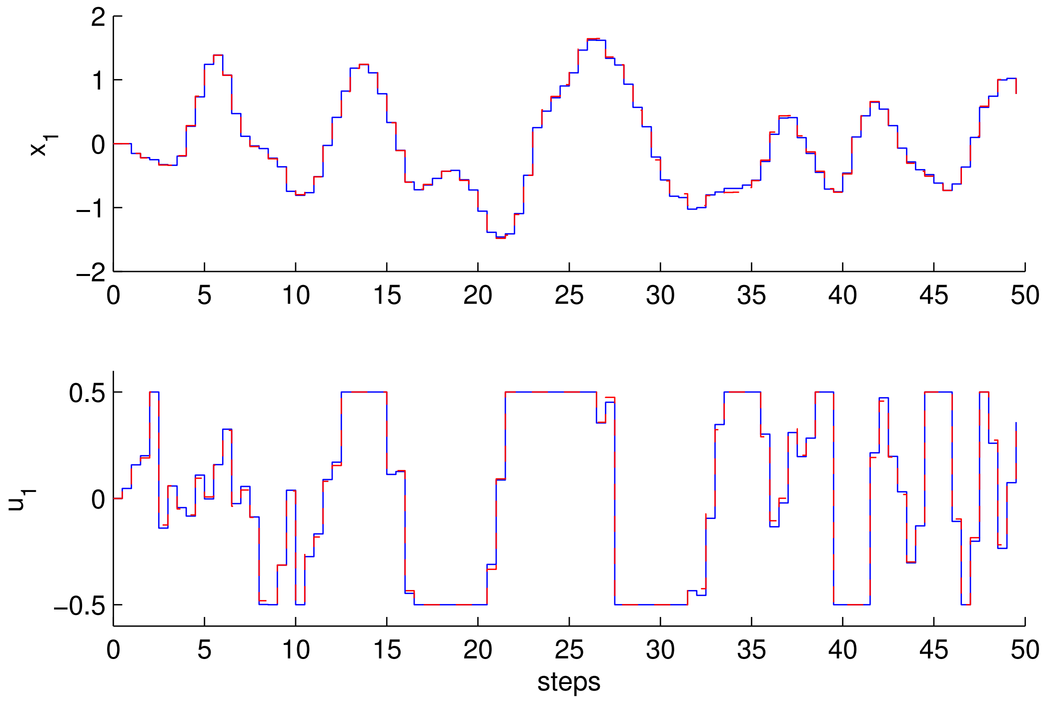


Figure 4.1.4: Oscillating masses simulation of first state and input. In this case parameters were set as $\Psi = 10^3 I$, $i_{max} = \infty$. *Blue solid line as reference for exact MPC, dashed red line for approximate MPC.*

4.1.4.2 Random Systems

In the following two tests set of 1000 random stable systems (except that poles on unit circle are included) with the fixed parameters $n_x = 4$, $n_u = 2$, $N = 10$ were taken. In this case limits were chosen ± 10 for state and ± 0.2 for inputs. The initial states were picked randomly around the origin in range $(-1, 1)$.

Single QP Solution First, random systems were taken and its appropriate QP of approximated problem solved to optimality. Results are shown in Figure 4.1.5 where it may look like that choice of $\Psi > 5e3I$ leads in general to good approximation since median of relative error of the cost function is around 1.72%. The relative error of cost function is defined as

$$error_J = 100(J - J_{ref})/J_{ref}, \tag{4.9}$$

where $J = \boldsymbol{u}^T \boldsymbol{H} \boldsymbol{u} + \boldsymbol{f}^T \boldsymbol{u}$ is cost function computed with optimal solution $\boldsymbol{u}$ of the proposed approximate method while in J_{ref} exact solution of (2.7) is used.

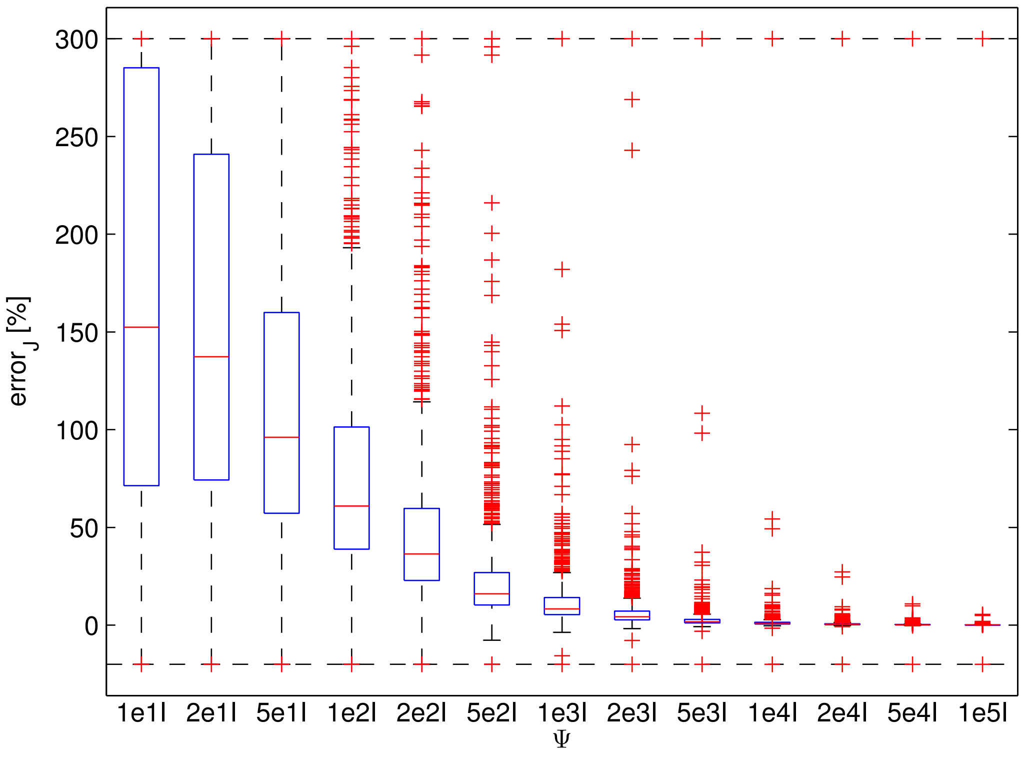


Figure 4.1.5: Relation of penalty parameter Ψ to cost function error (4.9) of QPs related to the random systems. *For each box, red line in the center is the median, the bottom of the box is the 25th percentile, the top is the 75th percentile. Red crosses represent the outliers.*

Control Simulation Solution In this test each random system was controlled over 100 time-steps. In contrary, Figure 4.1.6 illustrates how relative error of mean value of stage costs depends on penalty parameter. The error of mean relative stage cost is defined as

$$error_l = 100(l - l_{ref})/l_{ref}, \tag{4.10}$$

where l denotes mean stage cost of our method while l_{ref} denotes mean stage cost of reference (exact solution of (2.7)). Figure 4.1.6 proofs that even if individual QP is solved suboptimally (even with large cost function error, see Figure 4.1.5) resulting control performance may be even still good enough hence to get well performed controller even $\Psi = 10^3 I$ can be used. In this simulation (Figure 4.1.6) a set only of 100 random systems was controlled over 100 steps due to the test complexity hence the results may not be exceedingly accurate nevertheless the trend is obvious.

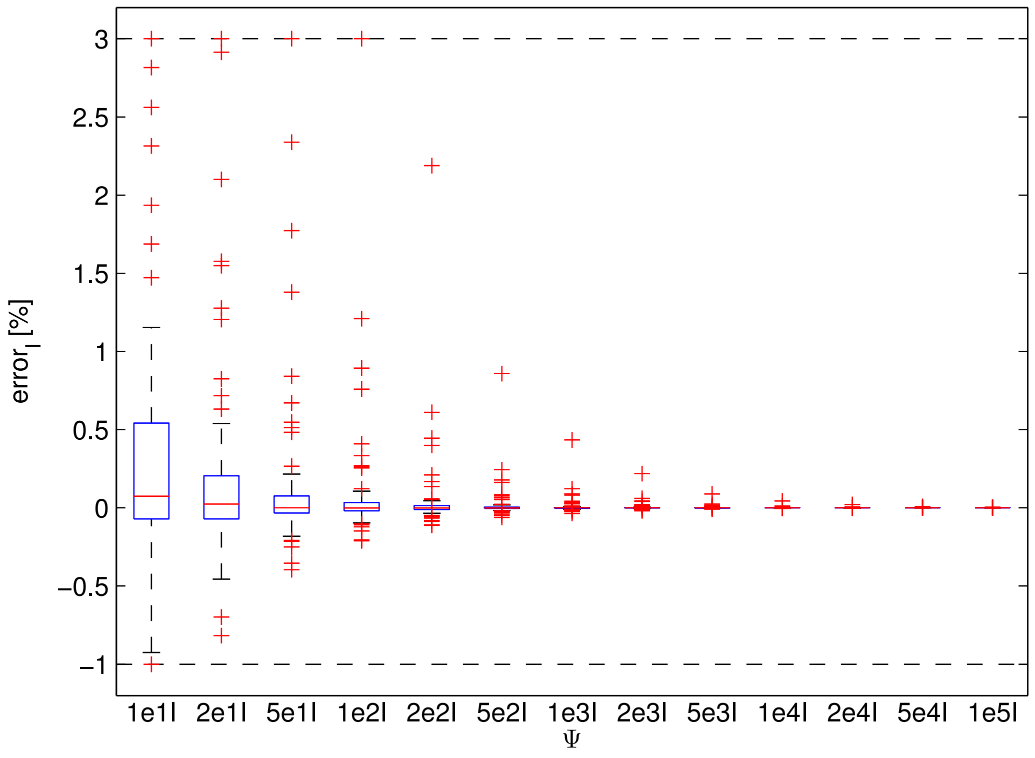


Figure 4.1.6: Relation of penalty parameter Ψ to mean stage cost error (4.10) of control simulations of random systems. *For each box, red line in the center is the median, the bottom of the box is the 25th percentile, the top is the 75th percentile. Red crosses represent the outliers.*

Comparison of Proposed Methods Finally time comparison with several of state-of-the-art approximate approaches introduced in [4, 2, 47] is presented. In this experiment four different methods are compared on small or medium-scale problems only.

Let us briefly summarize properties of these methods:

Presented The presented method solves sparse MPC problem (4.4) where system dynamics is assumed to be soft, penalized by $\Psi = 10^3 I$. Combined Newton/gradient method (Algorithm 4) modified as described in Chapter 4 is used with earlier termination such that algorithm is stopped when maximum number of iterations $i_{max} = 5$ is reached.

IP Primal Interior-Point (IP) method solves sparse formulation of MPC as defined in [4]. For this purpose modified package published on Stanford University web pages related to [4] was used. In this case dynamics is considered as hard and constraints are handled by barrier method. Interior-point solver with infeasible start was executed with $i_{max} = 3$ (fixed number of Newton's step) and barrier parameter was fixed $t = 10^2$ for all problems.

Next QP algorithms based on condensed formulation (2.7) of MPC were treated.

FGP Fast gradient projection (FGP) represents basic implementation (without problem preconditioning) of algorithm presented in [2]. This approach is one of the "exact" ones and thus was taken as reference. FGP solver was set to stop when ϵ-optimal solution is found for $\epsilon = 10^{-9}$.

ASM Finally, solver *qpOASES* [47] was employed. Package qpOASES implements sophisticated active-set method (ASM) with warm-start.

In the test there were generated 3 random linear systems of (2.1) with additional state disturbance in range ± 0.5 with different n_u, n_x. The simulation with 1000 time steps for 3 different lengths of prediction horizon were executed. Experiment was executed 3 times and minimal time for each step was accepted. The resulting times written in Table 4.1 are mean value of accepted times in ms. The relative stage cost error (4.10) was below 2 % in all cases when as a reference FGP was taken. The formulated problems (2.7) had condition numbers $\kappa \sim 10^3$.

n_x	n_u	N	Presented	IP	FGP	ASM
4	2	10	**0.026**	0.036	0.052	0.128
4	2	20	**0.053**	0.066	0.099	0.222
4	2	30	**0.092**	0.097	0.183	0.372
7	3	10	**0.034**	0.053	0.064	0.124
7	3	20	**0.076**	0.103	0.165	0.265
7	3	30	0.164	0.152	0.785	0.506
10	4	10	**0.058**	0.078	0.069	0.172
10	4	20	0.175	0.154	0.645	0.734
10	4	30	0.329	0.238	0.785	1.091

Table 4.1: Comparison of computation times on random systems simulation. *Presented denotes our method; IP denotes solver developed in [4]; FGP [2]; ASM denotes solver qpOASES performed especially for MPC [47]. All times proposed in the table are in ms.*

In Table 4.1 several trends can be observed. See the last column of the table, the ASM has the worst times which is caused by the fact that problems were solved exactly by active-set method where Newton's step computation is executed each iteration and the algorithm needs many iterations to find exact solution since there were many active constraints in optimum. See also that times growth rapidly since number of problem variables increases cubically in $N \cdot n_u$ for the problem formulation. Almost same tendency can be viewed in results for FGP where a lot of gradients were computed.

An approximated methods based on sparse formulation follow. See the times growth linearly in case of IP and superlinearly in case of presented method. What causes the superlinear trend in computational demand of the presented method? Null-space method entails that Newton's steps computed in Algorithm 4 will be cheaper since reduced Hessian is used, on the other hand exact line search method computed twice per iteration is additional effort.

Hence the presented method is suitable especially for small-scale problems where the best results of all compared methods were achieved.

4.2 Approximated Dense MPC Formulation Framework

The main disadvantages of QP arising in condensed MPC (Section 2.4.1) are ill-conditioning and that problem structure is lost which causes that computational complexity of the problem grows cubically with number of optimized variables.

To deal with the computational demand some techniques were developed. The most essential is input blocking [24] which reduces problem dimension by reducing degrees of freedom - problem is reformulated in a way that only control over several time-steps at the beginning of the horizon is computed. For the rest of the horizon constant control is assumed. This method decrease computational burden significantly and hence is very popular in practice, but resulting control policy may be very conservative. From this point of view it would be much better if not constant control would be considered for the rest of the horizon and hence control policy would not be so conservative.

In this section a novel formulation of MPC which reduces computational effort similarly as input blocking will be introduced. But in the proposed strategy for the rest of the horizon not constant control is assumed .

The main idea is to split optimization horizon into two parts for inputs and then inject appropriate prediction such that only start of prediction horizon will be optimized and the rest will be approximated. This give us new generalized view on condensed formulation of MPC. In this view classical condensed formulation presented in Section 2.4.1 is just limit (exact) case where $N_u = N$ (N_u is control optimization horizon defined lower) furthermore it will be shown that solution for $N_u = 1$ may be still competitive in performance but with extreme computational saving.

For approach introduced in Section 4.1.4 only quantitative change in performance has been seen, but for this approach qualitative change in control policy is expected thus rather tracking problem will be provided in Section 4.2.3.

In following section modified prediction will be derived. Then, in next section novel formulation of the old problem will be stated. At the end properties of the presented method will be discussed.

4.2.1 Modified Prediction Based on Augmented State-Space Model

Recall for augmented state-space model based prediction for output error (2.20).

$$\tilde{\boldsymbol{e}} = \boldsymbol{w}\tilde{x}_k + \boldsymbol{W}\Delta\boldsymbol{u}. \tag{4.11}$$

In this section it will be shown how this prediction (4.11) can be rewritten into a general form which allow us reformulate the problem as it is described in following section.

Our approach requires to split prediction matrix $\boldsymbol{W}$ as follows

$$\boldsymbol{w} = \begin{bmatrix} \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}} \\ \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}}^2 \\ \vdots \\ \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}}^N \end{bmatrix}, \quad \boldsymbol{W} = \left[\begin{array}{c|c} \boldsymbol{W}_0 & \boldsymbol{W}_{N_u} \end{array}\right] = \left[\begin{array}{cc|cc} \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{B}} & & & \\ \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}} & \ddots & & \\ \vdots & \ddots & \ddots & \\ \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{A}}^{N-1}\tilde{\boldsymbol{B}} & \dots & \dots & \tilde{\boldsymbol{C}}_e\tilde{\boldsymbol{B}} \end{array}\right]. \tag{4.12}$$

It allows us to write (4.11) in different manner

$$\tilde{\boldsymbol{e}} = \boldsymbol{w}\tilde{x}_k + \left[\begin{array}{c|c} \boldsymbol{W}_0 & \boldsymbol{W}_{N_u} \end{array}\right] \begin{bmatrix} \Delta\boldsymbol{u}_0 \\ \Delta\boldsymbol{u}_{N_u} \end{bmatrix}, \tag{4.13}$$

where the horizon for control is spitted into two parts: Start of the horizon is denoted as $\boldsymbol{\Delta u}_0 = \Delta u_0, \dots, \Delta u_{N_u-1}$; End of the horizon is denoted as $\boldsymbol{\Delta u}_{N_u} = \Delta u_{N_u}, \dots, \Delta u_{N-1}$, for sake of space saving subscripts consist the initial subscript in appropriate part of the horizon only. Thus additional parameter for the presented framework must be defined as $N_u \in \mathbb{N},\ 0 < N_u \leq N$.

Finally, the generalized prediction (4.13) can be expressed in a way that only start of the control horizon with the current state is isolated on the ride hand side as below

$$\begin{bmatrix} \boldsymbol{\Delta u}_{N_u} \\ \tilde{\boldsymbol{e}} \end{bmatrix} = \boldsymbol{P w} \tilde{x}_k + \boldsymbol{P W}_0 \boldsymbol{\Delta u}_0, \tag{4.14}$$

where $\boldsymbol{P} = \begin{bmatrix} -\boldsymbol{W}_{N_u} & \boldsymbol{I} \end{bmatrix}^+$. Remind that $(.)^+$ donates Moore-Penrose pseudo-inversion.

4.2.2 Problem Reformulation

Objective function of reference tracking problem (2.22) can be then rewritten as

$$\begin{aligned} J &= \frac{1}{2}\tilde{e}_N^T Q_N \tilde{e}_N + \frac{1}{2}\sum_{n=1}^{N-1} \tilde{e}_n^T Q \tilde{e}_n + \frac{1}{2}\sum_{n=0}^{N_u-1} \Delta u_n^T R \Delta u_n + \frac{1}{2}\sum_{n=N_u}^{N-1} \Delta u_n^T R \Delta u_n = \\ &= \frac{1}{2}\begin{bmatrix} \boldsymbol{\Delta u}_{N_u} \\ \tilde{\boldsymbol{e}} \end{bmatrix}^T \underbrace{\begin{bmatrix} \boldsymbol{R}_{N_u} & \\ & \begin{bmatrix} \boldsymbol{Q} & \\ & Q_N \end{bmatrix} \end{bmatrix}}_{\boldsymbol{\Gamma}} \begin{bmatrix} \boldsymbol{\Delta u}_{N_u} \\ \tilde{\boldsymbol{e}} \end{bmatrix} + \frac{1}{2}\boldsymbol{\Delta u}_0^T \boldsymbol{R}_0 \boldsymbol{\Delta u}_0 \end{aligned} \tag{4.15}$$

with known data $\boldsymbol{R}_0 \in \mathbb{R}^{N_u n_u \times N_u n_u}$, $\boldsymbol{R}_{N_u} \in \mathbb{R}^{(N-N_u)n_u \times (N-N_u)n_u}$, $\boldsymbol{Q} \in \mathbb{R}^{(N-1)n_y \times (N-1)n_y}$ and $Q_N \in \mathbb{R}^{n_y \times n_y}$.

Remind that QP arising in MPC can be stated symbolically as

$$\begin{aligned} \min \quad & \text{control criterion} \\ \text{s.t.} \quad & \text{prediction} \\ & \text{physical limitation of plant inputs} \end{aligned} \tag{4.16}$$

It is usual in condensed formulations to eliminate prediction (equality constraint) by its introduction into criterion.

Then such modified criterion where new prediction (4.13) was eliminated is

$$J = \frac{1}{2}\boldsymbol{\Delta u}_0^T \boldsymbol{W}_0^T \boldsymbol{Q}_0 \boldsymbol{W}_0 \boldsymbol{\Delta u}_0 + \boldsymbol{\Delta u}_0^T \boldsymbol{W}_0^T \boldsymbol{Q}_0 \boldsymbol{w} \boldsymbol{\Delta} \tilde{\boldsymbol{x}}_k + \frac{1}{2}\boldsymbol{\Delta u}_0^T \boldsymbol{R}_0 \boldsymbol{\Delta u}_0 \tag{4.17}$$

with properly defined weighting matrix $\boldsymbol{Q}_0 = \boldsymbol{P}^T \boldsymbol{\Gamma} \boldsymbol{P},\ \boldsymbol{Q}_0 \in \mathbb{R}^{N \cdot n_y \times N \cdot n_y}$.

4.2.2.1 Coordinate Transformation

The tracking criterion (4.15) is defined with control increments hence variables optimized by appropriate QP would be also increments of inputs. It is disadvantageous that the result has to be recalculated to another coordinated each time to be applied to the plant. Also it is important to provide hard constraints which arise from limitation of the plant. But if box constrained problem is assumed control increment formulation (briefly described in Section 2.6) causes this simple boundaries would be lost and cannot be exploited in the optimization algorithm. Thus well known trick, coordinate transformation

$$\Delta \boldsymbol{u}_0 = \boldsymbol{K}\boldsymbol{u}_0 + \boldsymbol{M}u_{-1} \tag{4.18}$$

$$\boldsymbol{K} = \begin{bmatrix} I & & & \\ -I & I & & \\ & -I & I & \\ & & \ddots & \ddots \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} -I \\ 0 \\ 0 \\ \vdots \end{bmatrix} \tag{4.19}$$

is provided to preservation of simple constraints. This transformation is based on simple observation that $\Delta u_n = u_n + u_{n-1}$.

Reformulated objective function which is optimized eventually is

$$\begin{aligned} J = &\frac{1}{2}\boldsymbol{u}_0^T\boldsymbol{K}^T\boldsymbol{W}_0^T\boldsymbol{Q}_0\boldsymbol{W}_0\boldsymbol{K}\boldsymbol{u}_0 + \boldsymbol{u}_0^T\boldsymbol{K}^T\boldsymbol{W}_0^T\boldsymbol{Q}_0\boldsymbol{W}_0\boldsymbol{M}u_{-1} + \\ &+\frac{1}{2}\boldsymbol{u}_0^T\boldsymbol{K}^T\boldsymbol{W}_0^T\boldsymbol{Q}_0\boldsymbol{w}\tilde{\boldsymbol{x}}_k + \frac{1}{2}\boldsymbol{u}_0^T\boldsymbol{K}^T\boldsymbol{R}_0\boldsymbol{K}\boldsymbol{u}_0 + \boldsymbol{u}_0^T\boldsymbol{K}^T\boldsymbol{R}_0\boldsymbol{M}u_{-1}, \end{aligned} \tag{4.20}$$

where all constant terms were already omitted, since they do not influence the minimizer.

Thus resulting control problem which need to be solved is stated in term of quadratic programming as

$$\begin{aligned} \min_{\boldsymbol{u}_0} \quad & \frac{1}{2}\boldsymbol{u}_0^T\boldsymbol{H}\boldsymbol{u}_0 + \boldsymbol{u}_0^T\boldsymbol{f} \\ \text{s.t} \quad & \underline{\boldsymbol{u}}_0 \leq \boldsymbol{u}_0 \leq \overline{\boldsymbol{u}}_0 \end{aligned} \tag{4.21}$$

with

$$\boldsymbol{H} = \boldsymbol{K}^T(\boldsymbol{W}_0^T\boldsymbol{Q}_0\boldsymbol{W}_0 + \boldsymbol{R}_0)\boldsymbol{K} \tag{4.22}$$

$$\boldsymbol{f} = \boldsymbol{K}^T\boldsymbol{W}_0^T\boldsymbol{Q}_0\boldsymbol{w}\tilde{\boldsymbol{x}}_0 + \boldsymbol{K}^T\boldsymbol{W}_0^T\boldsymbol{Q}_0\boldsymbol{W}_0\boldsymbol{M}u_{-1}. \tag{4.23}$$

Note that the Hessian size is only $N_u n_u \times N_u n_u$ where N_u is chosen such that $0 < N_u \leq N$. Thus in limit case problem size can be even $n_u \times n_u$. On the other hand only first input control is optimized hence only first control input can be constrained. It will be shown in next section how much is the loss of performance caused by this approximation unimpressive.

4.2.3 Properties of the Proposed Method

In the new formulation framework proposed in this work theoretical extension of fundamental MPC formulation is provided. One can choose length of horizon N_u then for first part $0 \rightarrow N_u$ exact control strategy will be computed and for the rest of the horizon approximated control strategy will be used instead. Note that for the second part of the horizon $N_u \rightarrow N$ the constraints are not considered.

This difference in control strategy may be unsuitable for some kind of process control application i.e. where the actuators are saturated all the time.

On the other hand increasing the parameter N_u cause large improvement in a way that the result of the approximate strategy will be much closer to the exact one and also computational demand will be much smaller then for exact dense formulation.

On the example of a system which step responses can be seen in Figure 4.2.1 a properties of the approximated approach will be shown. The system has $n_x = 6, n_u = 3, n_y = 2$ and length of prediction horizon for controller was set as $N = 10$.

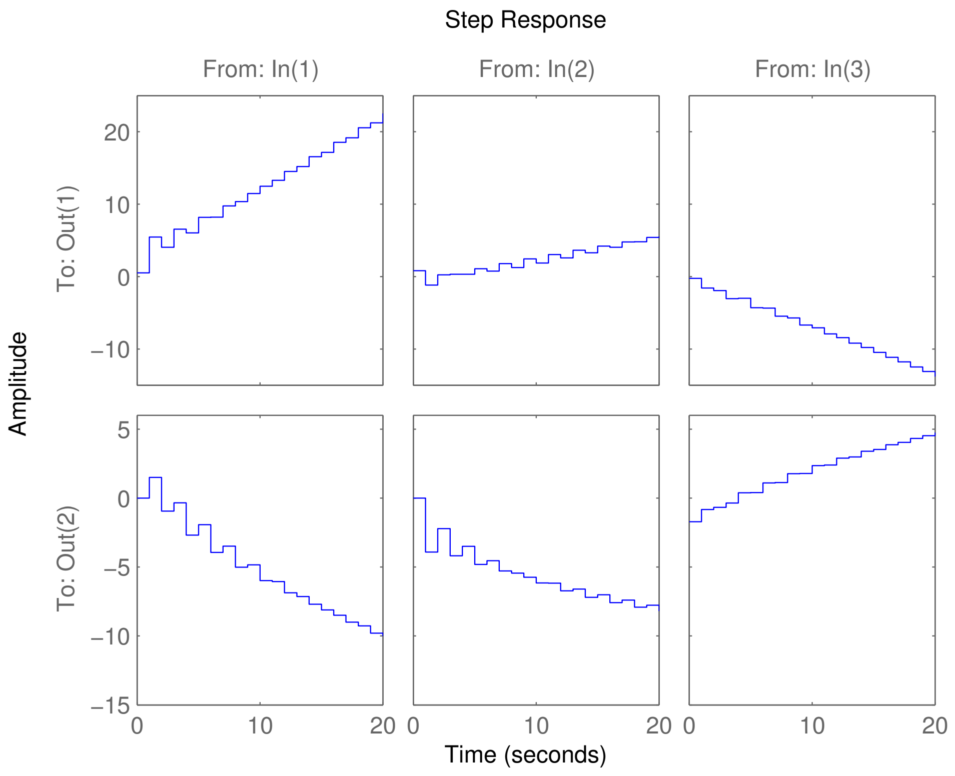


Figure 4.2.1: Step responses of an example system.

First, it can be seen in Figure 4.2.2 that for an unconstrained case of the multiple-input multiple-output (MIMO) system same solution as in exact case is obtained. It means that for systems for which no constraints is considered only small optimization problem (with n_u optimized variables) have to be solved.

Next, it is shown on constrained case for which several degree of approximation were chosen how much different the control policy is. Example is served in Figure 4.2.3 for different values of $N_u = 1, 2, 5, 10$ (exact solution). It can be seen that only a half-length approximation has very similar results with much less computational effort in contrast with exact solution.

On the next Figure 4.2.4 appropriate problem conditioning is shown. It can be seen that even for small level of approximation $N_u = N - 1$ the optimization problem is much less conditioned. This is caused by splitting the prediction matrix $\boldsymbol{W}$ which consist high power orders of $\tilde{\boldsymbol{A}}$.

Because a problem arising from the formulation framework has lost sparsity structure any generic solver can be used with no privilege. For the computational times see e.g. [15] where times for different-sized QP for another combined Newton/gradient method are presented.

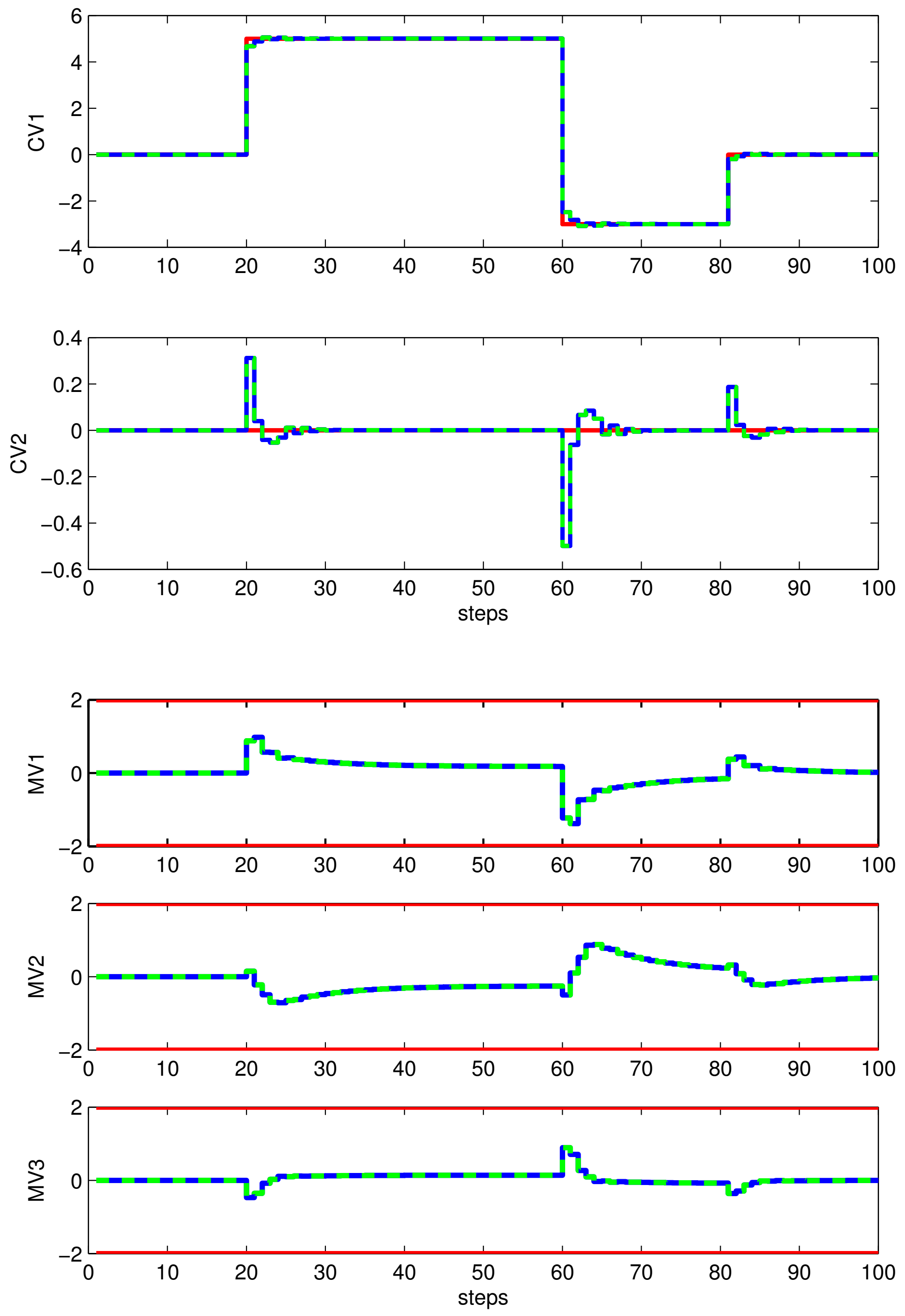


Figure 4.2.2: Unbounded approximate control. *CV denotes controlled variable, MV denotes manipulated variable. Red color sign reference value in CVs while bounding in MVs. Blue solid line denotes exact solution ($N_u = N$) and green dashed one denotes approximate solution for $N_u = 1, 2, 5$.*

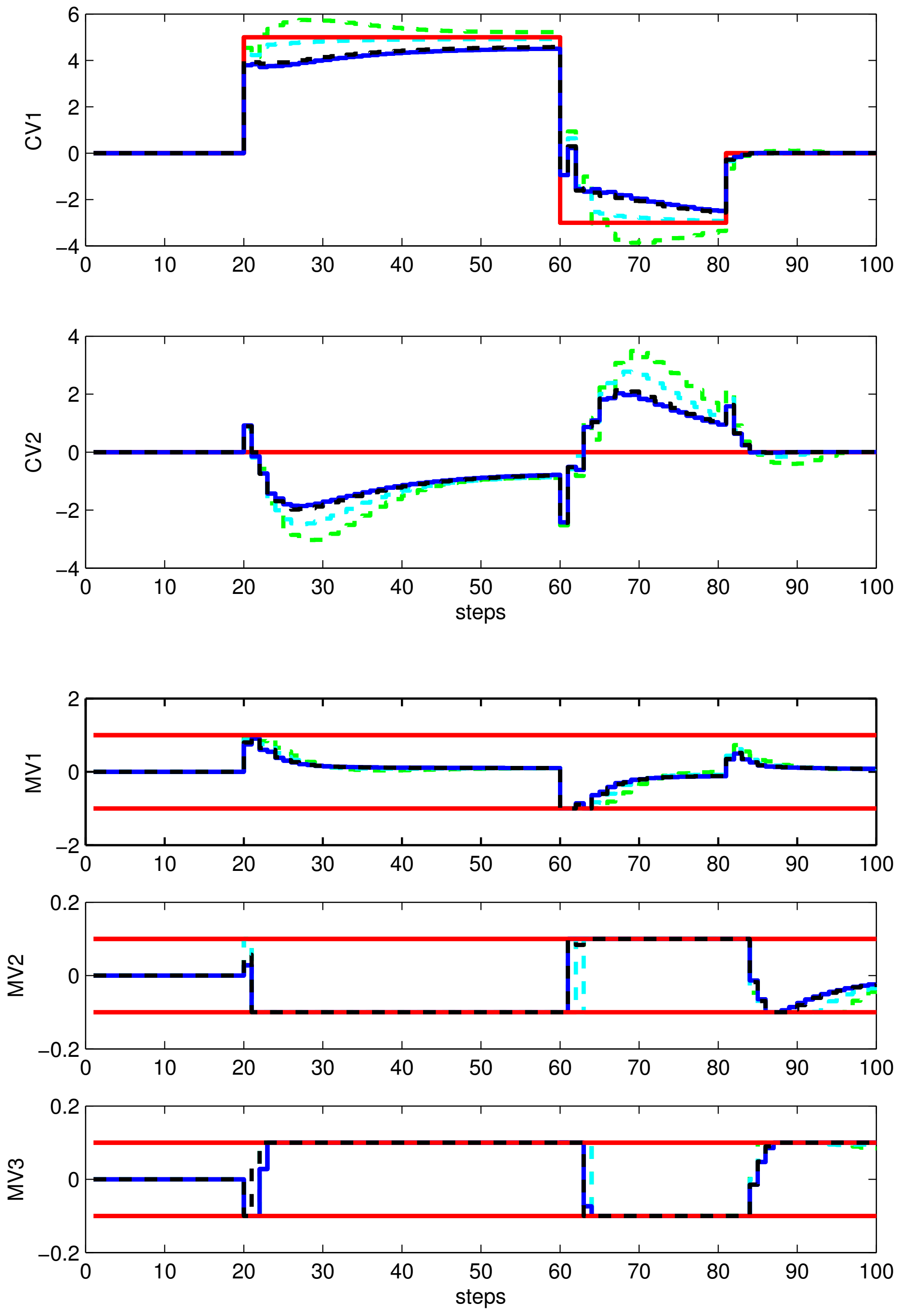


Figure 4.2.3: Bounded approximate control. *CV denotes controlled variable, MV denotes manipulated variable. Red color sign reference value in CVs while bounding in MVs. Blue solid line denotes exact solution $N_u = N$ (30 optimized variables) moreover approximate solutions are black dashed line for $N_u = 5$ (15 o.v.), cyan dashed line for $N_u = 2$ (6 o.v.) and green dashed line for $N_u = 1$ (3 o.v.).*

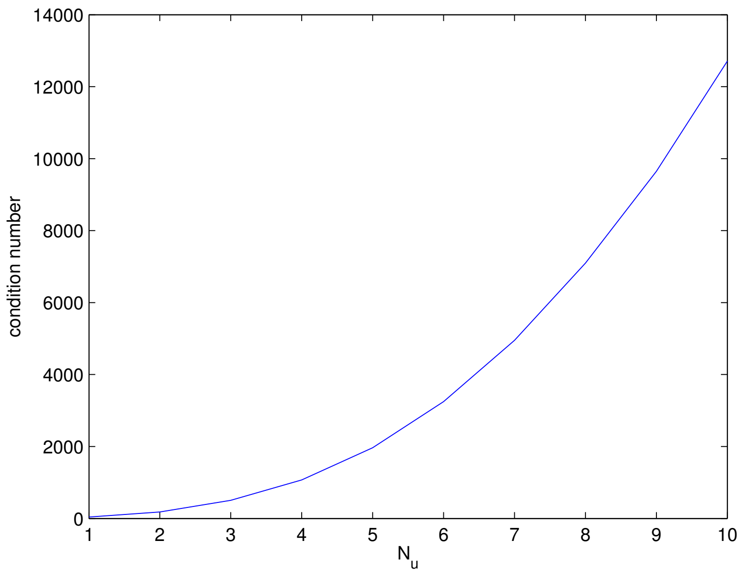


Figure 4.2.4: Conditioning of the problem Hessian for different N_u for the example (Figure 4.2.1).

Chapter 5

Conclusion

Model-based predictive control is a multivariable control technique which is able to provide optimal control policy with consideration of physical limitation of controlled process. Optimal control problem (OCP) for finite horizon is solved each time-period and resulting control action for current inputs only is applied to the plant.

Since OCP must be solved each time for purpose of fast sampled system effective OCP (very often and also in our case quadratic programming) formulations and appropriate algorithms have to be developed. Efficient One way how to overcome computational limits are various approximation-based techniques.

In this work fundamentals of linear MPC were shown. Some basic exact as well as advanced approximated approaches were described with manifold references on the literature. Also several state of the art algorithms for solving QP problem arising in MPC were proposed and their advantages and disadvantages discussed. All of this aim to the main load of this work - to develop fast approximate algorithm for model predictive control.

In this work two novel approaches for fast MPC has been introduced. The first one, sparse MPC where soft dynamics of the system is a priory assumed solved by combined Newton/gradient method. By numerical examples it is shown that this approach is for small and medium-scale instances of random dynamical systems faster then other state of the art methods while the control performance is preserved.

The second approach is condensed approximated MPC framework which main object is to reduce number of optimized variable in a way that no constraints will be assumed for certain part of prediction horizon. Thus require memory demand which is very important in embedded systems will be saved. Furthermore if degree of freedom of OCP will be decreased also computational effort will be decreased. This properties have been also shown here. Also it is shown that conditioning of the resulting optimization problem arising in the framework is significantly improved.

Moreover even if some part of the prediction horizon is approximated such that no constraints for this part are provided it seems that overall controller performance is not harmed dramatically. Thus the method introduced in this work may be an alternative approach to the input blocking technique which is widely use nowadays to decrease degree of freedom of OCP.

In short, in this work new theoretical results were derived and novel formulations of MPC stated. Also with this work two novel implementations of online MPC were developed and tested.

Future work

In the future work the latter introduced approach will be investigate properly. Appropriate attention will be focused on pseudoinversion technique and its consequences on overall control process. Another goal will be to extend the proposed approximation idea for dual dense MPC where states are the optimized variables (inspiration in [17]). Also problem structure could be possibly exploited if appropriate reformulation will be developed (inspiration in [26]). Although the problem conditioning is better in contrary to standard dense formulation some kind of preconditioning could be used to achieve even faster control. The theoretical results obtained in control could be possibly extend for moving horizon estimation (MHE). Finally, application on real-world problem is expected.

Bibliography

[1] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413 – 428, 1978.

[2] S. Richter, S. Mariethoz, and M. Morari, "High-speed online MPC based on a fast gradient method applied to power converter control," in *American Control Conference (ACC)*, pp. 4737–4743, IEEE, 2010.

[3] G. Pannocchia, J. Rawlings, and S. Wright, "Fast, large-scale model predictive control by partial enumeration," *Automatica*, vol. 56126, no. October, 2007.

[4] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, 2010.

[5] M. N. Zeilinger, C. N. Jones, and M. Morari, "Real-Time Suboptimal Model Predictive Control Using a Combination of Explicit MPC and Online Optimization," *IEEE Transactions on Automatic Control*, vol. 56, pp. 1524–1534, July 2011.

[6] A. Bemporad, A. Oliveri, T. Poggi, and M. Storace, "Ultra-Fast Stabilizing Model Predictive Control via Canonical Piecewise Affine Approximations," *IEEE Transactions on Automatic Control*, vol. 56, pp. 2883–2897, Dec. 2011.

[7] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002.

[8] F. Borrelli, J. Pekar, M. Baotic, and G. Stewart, "On the Computation of Linear Model Predictive Control Laws," *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 7375–7380, Dec. 2009.

[9] A. Bemporad and C. Filippi, "Suboptimal explicit receding horizon control via approximate multiparametric quadratic programming," *Journal of optimization theory and applications*, vol. 117, no. 1, pp. 9–38, 2003.

[10] H. Ferreau, P. Ortner, P. Langthaler, L. Re, and M. Diehl, "Predictive control of a real-world Diesel engine using an extended online active set strategy," *Annual Reviews in Control*, vol. 31, no. 2, pp. 293–301, 2007.

[11] C. N. Jones and M. Morari, "Polytopic Approximation of Explicit Model Predictive Controllers," *IEEE Transactions on Automatic Control*, vol. 55, no. 11, pp. 2542–2553, 2010.

[12] O. Santin and V. Havlena, "Combined partial conjugate gradient and gradient projection solver for MPC," *2011 IEEE International Conference on Control Applications (CCA)*, pp. 1270–1275, Sept. 2011.

[13] O. J. Rojas, G. C. Goodwin, M. M. Serón, and A. Feuer, "An SVD based strategy for receding horizon control of input constrained linear systems," *International Journal of Robust and Nonlinear Control*, vol. 14, pp. 1207–1226, Sept. 2004.

[14] J. Unger, M. Kozek, and S. Jakubek, "Reduced order optimization for model predictive control using principal control moves," *Journal of Process Control*, vol. 22, no. 1, pp. 272–279, 2012.

[15] O. Santin and V. Havlena, "Combined Gradient and Newton Projection Quadratic Programming Solver for MPC," *Proceedings of the 18th IFAC World Congress*, pp. 5567–5572, 2011.

[16] D. Axehill and A. Hansson, "A dual gradient projection quadratic programming algorithm tailored for model predictive control," in *47th IEEE Conference on Decision and Control*, pp. 3057–3064, Ieee, 2008.

[17] G. M. Mancuso and E. C. Kerrigan, "Solving constrained LQR problems by eliminating the inputs from the QP," *IEEE Conference on Decision and Control and European Control Conference*, pp. 507–512, Dec. 2011.

[18] P. Campo and M. Morari, "∞-Norm formulation of model predictive control problems," *American Control Conference, 1986*, 1986.

[19] H. Genceli and M. Nikolaou, "Robust stability analysis of constrained l1-norm model predictive control," *AIChE Journal*, vol. 39, no. 12, 1993.

[20] C. V. Rao and J. B. Rawlings, "Linear programming and model predictive control," *Journal of Process Control*, vol. 10, pp. 283–289, Apr. 2000.

[21] K. I. Kouramas, N. P. Faísca, C. Panos, and E. N. Pistikopoulos, "Explicit/multi-parametric model predictive control (MPC) of linear discrete-time systems by dynamic and multi-parametric programming," *Automatica*, vol. 47, no. 8, pp. 1638–1645, 2011.

[22] J. Rossiter, *Model-Based Predictive Control: A Practical Approach.* CRC Press control series, CRC PressINC, 2003.

[23] L. Van den Broeck, M. Diehl, and J. Swevers, "A model predictive control approach for time optimal point-to-point motion control," *Mechatronics*, vol. 21, pp. 1203–1212, Oct. 2011.

[24] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari, "Move blocking strategies in receding horizon control," *Journal of Process Control*, vol. 17, pp. 563–570, July 2007.

[25] D. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

[26] D. Axehill and M. Morari, “An alternative use of the Riccati recursion for efficient optimization,” *Systems & Control Letters*, vol. 61, pp. 37–40, Jan. 2012.

[27] A. Bemporad, “Model predictive control design: New trends and tools,” *Decision and Control, 2006 45th IEEE Conference on*, no. 1, pp. 6678–6683, 2006.

[28] D. Dimitrov, A. Sherikov, and P.-B. Wieber, “A sparse model predictive control formulation for walking motion generation,” *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2292–2299, Sept. 2011.

[29] J. L. Jerez, E. C. Kerrigan, and G. a. Constantinides, “A sparse and condensed QP formulation for predictive control of LTI systems,” *Automatica*, vol. 48, pp. 999–1002, May 2012.

[30] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, “Basic linear algebra subprograms for fortran usage,” *ACM Trans. Math. Softw.*, vol. 5, pp. 308–323, Sept. 1979.

[31] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide.* Philadelphia, PA: Society for Industrial and Applied Mathematics, third ed., 1999.

[32] B. Houska, H. J. Ferreau, and M. Diehl, “ACADO toolkit—An open-source framework for automatic control and dynamic optimization,” *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[33] F. Ullmann, “FiOrdOs: A Matlab toolbox for C-code generation for first order methods,” no. December, 2011.

[34] J. Mattingley, Y. Wang, and S. Boyd, “Code generation for receding horizon control,” *2010 IEEE International Symposium on Computer-Aided Control System Design*, pp. 985–992, Sept. 2010.

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

[36] J. Nocedal and S. J. Wright, *Numerical optimization.* Springer verlag, 1999.

[37] Y. Wang and S. Boyd, “Performance bounds for linear stochastic control,” *Systems & Control Letters*, vol. 58, pp. 178–182, Mar. 2009.

[38] A. A. Goldstein, “Convex programming in Hilbert space,” *Bull. Amer. Math. Soc*, vol. 70, no. 5, pp. 709–710, 1964.

[39] D. P. Bertsekas, “On the Goldstein-Levitin-Polyak gradient projection method,” *IEEE Transactions on Automatic Control*, vol. 21, no. 2, pp. 174–184, 1976.

[40] Y. Nesterov, “A method of solving a convex programming problem with convergence rate O (1/k2),” *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[41] M. Kogel and R. Findeisen, “Fast predictive control of linear systems combining Nesterov's gradient method and the method of multipliers,” *IEEE Conference on Decision and Control and European Control Conference*, pp. 501–506, Dec. 2011.

[42] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Mathematics and its applications, Kluwer Academic Publishers, 2004.

[43] S. Richter, C. N. Jones, and M. Morari, "Real-time input-constrained MPC using fast gradient methods," *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pp. 7387–7393, Dec. 2009.

[44] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*. Society for Industrial and Applied Mathematics, Jan. 1994.

[45] J. More, "Gradient projection techniques for large-scale optimization problems," in *Decision and Control, 1989., Proceedings of the 28th IEEE Conference on*, pp. 378–381 vol.1, 1989.

[46] P. Otta and O. Santin, "Fast QP Algorithm for Dynamics Penalty Model Predictive Control," *Accepted in Poster2013, 17th International Student Conference of Electrical Engineering*, 2013.

[47] H. Ferreau, H. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC," *International Journal of Robust and Nonlinear Control*, no. July 2007, pp. 816–830, 2008.

Appendix A

Dual Condensed Formulation

In this section it is briefly described formulation [17] which is complementary to the rest of basic formulation (2.4), although the formulation was developed recently.

The idea is to use opposite approach then was shown in primal dense formulation (2.4.1) that is to express inputs in prediction equation as shown below

$$\boldsymbol{u} = \acute{\boldsymbol{v}} x_k + \acute{\boldsymbol{V}} \boldsymbol{x} \tag{A.1}$$

$$\acute{\boldsymbol{v}} = -\boldsymbol{V}^{+}\boldsymbol{v} = \begin{bmatrix} -B^{+}A \\ 0 \\ 0 \\ \vdots \end{bmatrix}, \quad \acute{\boldsymbol{V}} = \boldsymbol{V}^{+} = \begin{bmatrix} B^{+} & & & \\ -B^{+}A & B^{+} & & \\ & -B^{+}A & B^{+} & \\ & & & \ddots \end{bmatrix}. \tag{A.2}$$

See sparse character of prediction matrices.

By injecting (A.1) to the (2.6) QP problem with interesting features is obtained and it can be stated as

$$\begin{aligned} \min \quad & \frac{1}{2}\boldsymbol{x}^T H \boldsymbol{x} + f^T \boldsymbol{x} \\ \text{s.t.} \quad & \underline{\boldsymbol{x}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{x}} \\ & \underline{\boldsymbol{u}} - \acute{\boldsymbol{v}} x_k \leq \boldsymbol{V}^{+}\boldsymbol{x} \leq \overline{\boldsymbol{u}} - \acute{\boldsymbol{w}} x_k \end{aligned} \tag{A.3}$$

with Hessian matrix H and linear part f as follows

$$H = \boldsymbol{Q} + \acute{\boldsymbol{V}}^T \boldsymbol{R} \acute{\boldsymbol{V}} \tag{A.4}$$

$$f^T = \acute{\boldsymbol{V}}^T \boldsymbol{R} \acute{\boldsymbol{v}} x_k, \tag{A.5}$$

where Hessian has sparse structure

$$H = \begin{bmatrix} Q + B^{T+}(R + A^T R A)B^{+} & -B^{T+}A^T R B^{+} & & \\ -B^{T+}RAB^{+} & Q + B^{T+}(R + A^T R A)B^{+} & \ddots & \\ & \ddots & \ddots & -B^{T+}A^T R B^{+} \\ & & -B^{T+}RAB^{+} & Q + B^{T+}RB^{+} \end{bmatrix}. \tag{A.6}$$

This formulation is quite interesting since many positive features are reached: Hessian is sparse with block-tridiagonal structure, condition number is low in contrast to primal dense formulation. On the other hand significant drawback occur - input constraints are not handled implicitly. Anyway problem dimension is stated as $N \cdot n_x$.

Appendix B

CD-ROM

Files on appended CD are categorize into two folders as follows

Latex contains working files of master thesis report.

Matlab contains both developed methods.

Root folder contains also the Master Thesis document which includes Diploma Thesis Assignment and Prohlášení as well.