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

Numerical Algorithms of Quadratic Programming for Model Predictive Control

by

Ondřej Šantin

Presented to the Faculty of Electrical Engineering,
Czech Technical University in Prague,
in Partial Fulfillment of the Requirements
for the Degree of Doctor.

Ph.D. Program: Electrical Engineering and Information Technology
Branch of Study: Control Engineering and Robotics
Supervisor: prof. Ing. Vladimír Havlena, CSc.

Prague, August 2016

Supervisor:
prof. Ing. Vladimír Havlena, CSc.
Department of Control Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2
160 00 Prague 6
Czech Republic

This work is dedicated to my Lord, Jesus Christ.

"There is nothing more practical than a good theory."
Kurt Lewin

Declaration

This doctoral thesis is submitted in partial fulfillment of the requirements for the degree of doctor (Ph.D.). The work submitted in this dissertation is the result of my own investigation, except where otherwise stated. I declare that I worked out this thesis independently and I quoted all used sources of information in accord with Methodical instructions about ethical principles for writing academic thesis. Moreover, I declare that it has not already been accepted for any degree and is also not being concurrently submitted for any other degree.

Prague, August 2016

Ondřej Šantin

Acknowledgements

First of all, I would like to express my gratitude to my dissertation thesis supervisor, prof. Ing. Vladimír Havlena, CSc. He has been a constant source of encouragement and insight during my research and helped me with numerous problems and professional advancements.

I would like to thank Ing. Jaroslav Pekař, PhD. for his great and catching enthusiasm for control problems and optimization. Large thanks go to prof. RNDr. Zdeňek Dostál, DSc. and Ing. Marta Jarošová, PhD. for encouraging and long talks about solvers and nonnegligible help when developing the algorithms.

Special thanks go to the staff of the Department of Control Engineering and Honeywell Prague Laboratory, who maintained a pleasant and flexible environment for my research.

My greatest thanks go to my wife, for her infinite patience and care.

The research in this thesis has been supported by grant MSM6840770038 "Decision and control in process control III" and by the Technology Agency of the Czech Republic (TA01030170).

Abstract

This dissertation thesis deals with the development of algorithms for the effective solution of quadratic programming problems for the embedded application of Model Predictive Control (MPC). MPC is a modern multivariable control method which involves solution of quadratic programming problem at each sample instant. The presented algorithms combine the active set strategy with the proportioning test to decide when to leave the actual active set. For the minimization in the face, we use the Newton directions implemented by the Cholesky factors updates. The performance of the algorithms is illustrated by numerical experiments and the results are compared with the state-of-the-art solvers on benchmarks from MPC. The main contributions of this thesis are three new quadratic programming solvers together with their proof of convergence and properties analysis. Furthermore, the algorithm's implementation is described in detail showing how to exploit the structure of the face problem and resulting Newton direction to reduce the computational complexity of each iteration.

Keywords:

Model Predictive Control, Quadratic Programming, Newton Type Method, Projection, Active Set Strategy.

Abstrakt

Tato práce se zabývá vývojem algoritmů pro efektivní řešení kvadratického programování pro vestavěné aplikace metody prediktivního řízení (MPC). MPC je moderní vícerozměrová řídicí metoda, která v sobě zahrnuje nutnost řešení kvadratického programování v každém vzorkovacím čase. Prezentované algoritmy kombinují metodu aktivních množin a tak zvaného testu proporcionality k rozhodnutí, zda-li má být aktuální množina aktivních omezení změněna. Minimalizace na množině aktivních omezení je provedena pomocí Newtonova směru, vypočítaného Choleskyho faktorizací spolu s aktualizací faktoru. Rychlost prezentovaných algoritmů je ilustrována na numerických experimentech aplikací MPC a výsledky jsou porovnány s řešiči dostupnými v literatuře. Hlavními přínosy práce jsou tři nové algoritmy řešičů kvadratického programování spolu s rozborem jejich vlastností a důkazem jejich konvergence. Vlastní implementace algoritmů je navíc podrobně popsána a je ukázáno, jak je možné využít struktury pomocného problému, řešeného na množině aktivních omezení, spolu se strukturou Newtonova směru k redukci výpočetní složitosti každé iterace.

Klíčová slova:

Prediktivní řízení, kvadratické programování, Newtonova metoda, projekce, metoda aktivních množin.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Acronyms

ADMM Alternating Direction Method of Multipliers

ASM Active Set Method

CG Conjugate Gradient

CGNP Combined Gradient and Newton Projection, presented in Algorithm 2

CV Controlled Variable

DV Disturbance Variable

ECU Engine Control Unit

EGR Exhaust Gas Recirculation

FGM Fast Gradient Method

flops Floating point operations

GPM Gradient Projection Method

GPQP Gradient Projection algorithm for Quadratic Programming (QP), presented in [3]

IPM Interior Point Method

KKT Karush–Kuhn–Tucker

LP Linear Programming

LQR Linear Quadratic Regulator

MAF Mass Air Flow

MAP Manifold Absolute Pressure

MPC Model Predictive Control

mp-QP Multi-parametric Quadratic Programming

MPRGP Modified Proportioning with Reduced Gradient Projections, presented in [28]

MV Manipulated Variable

NPP Newton Projection with Proportioning, presented in Algorithm 4

PLS Projected Line Search, presented in Section 4.6

PND Proportioning with Newton Directions, presented in Algorithm 3

PNM Projected Newton Method, presented in [11]

QP Quadratic Programming

RAM Random Access Memory

RHC Receding Horizon Control

ROM Read Only Memory

SPD Symmetric Positive Definite

SPS Symmetric Positive Semidefinite

SQP Sequential Quadratic Programming

TVA Throttle Valve Actuator

VGT Variable Geometry Turbocharger

WG Wastegate

Number Sets

$\mathbb{N}$	Natural numbers set
$\mathbb{N}_0$	Natural numbers set $\cup \{0\}$
$\mathbb{R}$	Real numbers set

Common Mathematical Functions and Operators

$\mathcal{I}$	Set of indices, $\mathcal{I} \subseteq \mathbb{N}$
$\lvert\mathcal{I}\rvert$	Cardinality of the set $\mathcal{I}$
$\boldsymbol{b}$	Vector $\boldsymbol{b}$
b_i	the i^{th} element of a vector $\boldsymbol{b}$
$\boldsymbol{b}_{\mathcal{I}}$	Vector consisting of b_i, $i \in \mathcal{I}$
$\boldsymbol{b}^T$, $\boldsymbol{A}^T$	Transpose to a vector $\boldsymbol{b}$, Transpose to a matrix $\boldsymbol{A}$
$\lvert\lvert\boldsymbol{b}\rvert\rvert$	Euclidean norm of a vector $\boldsymbol{b}$, $\lvert\lvert\boldsymbol{b}\rvert\rvert^2 = \boldsymbol{b}^T\boldsymbol{b}$
$\lVert\boldsymbol{b}\rVert_{\boldsymbol{A}}$	$\boldsymbol{A}$-energy norm of $\boldsymbol{b}$, $\lVert\boldsymbol{b}\rVert_{\boldsymbol{A}}^2 = \boldsymbol{b}^T\boldsymbol{A}\boldsymbol{b}$, $\boldsymbol{A} \succ 0$
$\dim \boldsymbol{b}$	Dimension of a vector $\boldsymbol{b}$
$\boldsymbol{A}$	Matrix $\boldsymbol{A}$
$a_{i,j}$	Element of a matrix $\boldsymbol{A}$ at the i^{th} row, and the j^{th} column
$\boldsymbol{A}_{\mathcal{I},\mathcal{J}}$	Sub-matrix consisting of $i \in \mathcal{I}$ rows, and $j \in \mathcal{J}$ columns of $\boldsymbol{A}$
$\mathcal{I}_i$	i-th element of indices set
$\boldsymbol{A}^{-1}$	Inverse matrix to a matrix $\boldsymbol{A}$
$\lvert\lvert\boldsymbol{A}\rvert\rvert$	Induced matrix norm of a matrix $\boldsymbol{A}$
$\operatorname{cond} \boldsymbol{A}$	Spectral condition number of a matrix $\boldsymbol{A}$, $\operatorname{cond} \boldsymbol{A} = \lVert\boldsymbol{A}\rVert\lVert\boldsymbol{A}^{-1}\rVert$
$\operatorname{rank} \boldsymbol{A}$	Rank of a matrix $\boldsymbol{A}$ — how many independent rows/columns it has
$\max\{a, b\}$	Maximum of a and b, a when $a \geq b$, b when $a < b$
$\min\{a, b\}$	Minimum of a and b, a when $a \leq b$, b when $a > b$
$\mathcal{O}$	Asymptotic complexity of the algorithm, e.g., $\mathcal{O}(n^2)$ for a quadratic complexity in n
a^k, $\boldsymbol{z}^k$, $\boldsymbol{A}^k$, $\mathcal{I}^k$	Scalar, vector, matrix and set at k-th iteration of the algorithm

Chapter 1

Introduction

This work addresses development of algorithms suitable for solution of convex box constrained Quadratic Programming (QP) problem which arises in the context of embedded applications of Model Predictive Control (MPC), although the methods are not restricted to this application. Firstly we restrict our attention to the control of linear MPC where we assume control of linear time-invariant discrete-time plant. Later we show that one of the developed methods can be applied to a more general framework of nonlinear MPC where it is assumed that plant to be controlled is non-linear time invariant and continuous-time.

In this chapter, firstly, the motivation for our work is described with a statement of the problem to be solved. Then a brief introduction to the previous results is given together with formal goals of this work. Finally, the organization of this thesis is outlined.

1.1 Motivation

Model Predictive Control (MPC) is an advanced multivariable optimization based control strategy which provides a systematic and scalable approach to designing the controller. MPC is widely used in industrial [72] or automotive applications, see e.g.,[C.6, 33, 49, J.2, 54, 53][1] among others. The main strength of MPC is that it can naturally incorporate the constraints on the process inputs, outputs, and/or states [56]. In MPC, the control goals (i.e., tracking of references, constraints) are transformed into an optimization problem. The relative importance of the potentially conflicting goals is translated into the weights of particular terms in the optimization problem cost function. The limitations of the controlled plant are then transformed into the optimization problem constraints.

MPC is a truly model based method, i.e., the model is not used exclusively for tuning purposes but also acts as the cornerstone of a decision to derive the correct control action to mitigate a hazardous situation in the future (e.g., limit violation). This is done through the prediction of plant states and outputs based on the model on the finite horizon to the future.

[1] The letters 'J', 'C' and 'P' in the bibliography represent author's journal, conference and patent type of work respectively.

Thus, the controller can change its control action to prevent a hazardous situation before it occurs. The result of an MPC algorithm is a future sequence of the input variables over the finite horizon. Only the first input move from the sequence is applied to the plant, and the entire calculation is repeated in the next sampling interval based on the updated estimation of the plant state. This is done to provide feedback in the loop to reject the plant model inaccuracy and presence of unmeasured disturbances leading to the so-called Receding Horizon Control (RHC) concept.

The main disadvantage of MPC, compared to traditional control methods, is higher computational complexity associated with the solution of the optimization problem at each sampling interval. As the control goals usually represent a minimization of the loss of the energy, the ℓ_2 norm is often used for individual terms in the cost function. Hence, the optimization problem to be solved is a convex Quadratic Programming (QP) problem.

Hence, MPC comprises at each sampling instant the solution of a QP problem and the parametrization of the system with the new measurement/estimate of the plant state. In the embedded applications of MPC, we are often limited by the available memory and computation power, e.g., available Random Access Memory (RAM) is less than 200kB and CPU frequency is approximately 100MHz in a standard Engine Control Unit (ECU) in automotive applications. Moreover, not all the resources may be available to the MPC solver since many other systems are running within the ECU. On one hand, the number of variables is in the order of tens or hundreds in typical embedded applications of MPC. On the other hand, a typical processor unit of embedded applications is either equipped only with single precision floating point arithmetic or even has to emulate it. Hence, the solver for MPC has to be numerically robust to deal with limited arithmetic accuracy and fast enough to deliver the solution before next sampling period.

In the last two decades, there has been rapid development in optimization algorithms to solve convex QP enabling sampling times in the order of milliseconds or even microseconds. Most of them reduce the solution of the QP problem to a series of unconstrained problems which are solved either by off-line, iterative or direct methods which typically combine the adapted Newton, active set, and gradient methods.

The overview of the most common methods for the solution of the QP for MPC:

Off-line methods are based on [6, 7] where the authors present algorithms for solving Multi-parametric Quadratic Programming (mp-QP) that are used to obtain explicit solutions to the MPC problem. The online phase of the MPC controller is then reduced to look-up table process since the solution is the affine map defined by the parameter value. The main disadvantage of the Explicit MPC is its large growth of the complexity of the mp-QP solution with the increasing number of constraints in the optimization problem.

Fast Gradient Methods (FGMs) or first order methods relies on the projection of the gradient and have attracted the attention of the MPC community recently (e.g., [76, 68, 50]) by its simple iterative scheme and tight certification aspects. The bottleneck is that they share the issue of the sensitivity to the problem scaling [12]. Hence, they might involve a relatively large number of iterations for ill-conditioned Hessian of the QP problems which is often the case in the application of MPC.

Active Set Methods (ASMs) estimate the optimal set of active constraints by the *active set* [65]. They are not sensitive to QP problem scaling but often involve a large number of iterations when there is a rapid change in the optimal active set. Unfortunately, such situation occurs often during transient operations of the MPC applications, where actuators hit their limits to reach new setpoints as fast as possible.

Interior Point Methods (IPMs) solve directly the Karush–Kuhn–Tucker (KKT) conditions of the QP problem by applying Newton method to a sequence of equality constrained problems or to a sequence of modified versions of the KKT conditions [18]. It was shown in [74, 25, 90] that IPMs can exploit the sparse structure of the problem[2] to reduce the computational cost associated with the solution of system of linear equations. Despite this, each iteration of IPMs is often more computationally expensive when compared to ASMs for considered problem size in this work.

The common issue of the gradient-based methods for the solution of QP problem is the strong sensitivity dependence of their rate of convergence on the QP problem conditioning in terms of the spectral condition number of QP problem Hessian. Unfortunately, the practical engineering applications often lead to the ill-conditioned Hessian of the QP problem either due to the MPC tuning (need of aggressive control, e.g., for limit violation) or ill-posed controllability of the system to be controlled.

Furthermore, in embedded systems the maximum execution time of the algorithm is what is important, i.e., not the average computation time. When the maximum computation time allocated for the algorithm is exceeded, the device is often reset by the watchdog and return to the normal operation. This device reset is, of course, undesirable, since it might impact the controller performance and safety: imagine a situation that vehicle's ECU is repeatedly reset due to the fact that the QP solver has not converged in the allocated time when the vehicle is driving on a highway at 70 mph! Such situation might occur for ASMs for rapid change of the optimal active set, or for gradient based methods due to the ill-conditioning of the problem or the large difference of the initial iterate to the solution.

The proposed algorithms in this work solve both issues by utilizing the projection for faster identification of optimal active set and utilizing the second order information by solving the auxiliary optimization problem, so-called face problem, defined by the active set, by use of direct solver via Cholesky factorization. Furthermore, the changes of the active set are controlled to avoid unnecessary expansion or reduction and hence, reduce the cost of the Cholesky factor update leading to faster solution of the face problem. The resulting algorithms identify the optimal active set maximally in 15-20 relatively low complex iterations independently of the initial iterate, the QP problem size, and conditioning.

[2] Using the sparse formulation of the MPC which will be introduced later in Section 2.1.1.2.

1.2 Problem Statement

In this work we shall be concerned with the sequence of problems to find

$$f^*(\boldsymbol{\theta}) \triangleq \min_{\boldsymbol{z}\in\Omega} f(\boldsymbol{z},\boldsymbol{\theta}) = f(\boldsymbol{z}^*(\boldsymbol{\theta}),\boldsymbol{\theta}), \tag{1.1}$$

where $\Omega = \{\boldsymbol{z} : \underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}}, \underline{\boldsymbol{z}} < \overline{\boldsymbol{z}}\}$, $f(\boldsymbol{z},\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{z}^T\boldsymbol{H}(\boldsymbol{\theta})\boldsymbol{z} + \boldsymbol{h}^T(\boldsymbol{\theta})\boldsymbol{z}$, $\underline{\boldsymbol{z}}$ and $\overline{\boldsymbol{z}}$ are given column n-vectors, $\boldsymbol{H}$ is an $n \times n$ Symmetric Positive Definite (SPD) matrix, and $\boldsymbol{h}$ is an n-vector parameterized by a parameter n_θ-vector $\boldsymbol{\theta}$ with a sequence $\theta \in \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots\}$. For a fixed value of the parameter $\boldsymbol{\theta}$ we define $\boldsymbol{h} = \boldsymbol{h}(\boldsymbol{\theta})$, $\boldsymbol{H} = \boldsymbol{H}(\boldsymbol{\theta})$ and

$$q(\boldsymbol{z}) = \tfrac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{h}^T\boldsymbol{z}. \tag{1.2}$$

We assume that the sequence $\{\boldsymbol{\theta}_k\}$ is not known in advance and $\boldsymbol{\theta}_k \approx \boldsymbol{\theta}_{k+1}$, so that we can use $\boldsymbol{z}^*(\boldsymbol{\theta}_k)$ as a good initial approximation for $\boldsymbol{z}^*(\boldsymbol{\theta}_{k+1})$. It is shown later in Chapter 2 that such sequence of parameters is generated in MPC applications.

Remark 1. Since $\boldsymbol{H}(\boldsymbol{\theta})$ is a SPD matrix and Ω is a nonempty convex set, the problem (1.1) is strictly convex QP problem for which there always exists a unique solution, see, e.g., [18].

Remark 2. The limitation of $\boldsymbol{z} \in \Omega$ might look as too restrictive compared to the usually more general form $\boldsymbol{z} \in \{\boldsymbol{z} \in \mathbb{R}^n : \boldsymbol{G}\boldsymbol{z} \leq \boldsymbol{w} + \boldsymbol{S}\boldsymbol{\theta}\}$, e.g., in [8]. On the other hand, we observed that Ω is sufficient for practical applications of MPC (e.g., [49, C.6]) and it enables a fast execution of the projection operation utilized by the proposed methods. Moreover, Ω covers the natural limitations of actuators and both the system state and output via soft constraints formulation. This formulation is often used instead of hard constraints to ensure feasibility of MPC problem solution. The limitation on the rate of change of the actuators might be either substituted by soft constraints or by limiting only the first control move in the actuator's trajectory.

Remark 3. Note that for linear MPC, the Hessian matrix $\boldsymbol{H}$ is constant for all $\boldsymbol{\theta}$ from the sequence.

We show in the following sections, that the problem (1.1) has to be solved at each sampling time in the MPC application with different parameter $\boldsymbol{\theta}$. We suppose that n is relatively small, i.e., $n \lesssim 200$. The reason for this restriction is the applicability of the proposed algorithm for the embedded systems where both the available memory and computational power is limited. Note that only storing the problem data for $n = 200$ takes 79.7 kB in single-precision[3] when only upper triangle of $\boldsymbol{H}$ is stored.

[3] For 4 bytes per single-precision floating-point number according to IEEE 754-1985.

1.3 Goals of the Dissertation Thesis

The main goals of this thesis are as follows:

1. Develop a general type of solver for convex box-constrained QP program suitable for application of MPC in embedded platforms (automotive, aerospace, etc.).
2. The proposed methods should involve low number of low complex iterations with minor impact of QP problem scaling, QP problem size and initial iterate.
3. Compare the performance of the developed and the state-of-the-art methods on selected numerical benchmarks.
4. Draw conclusions and directions of future research.

1.4 Structure of the Dissertation Thesis

The thesis is organized into five main parts as follows:

1. *Introduction*: Chapter 1 describes the motivation of this work together with its goals.
2. *Background and State-of-the-Art*: Chapter 2 introduces the reader to the necessary theoretical background and surveys the current state-of-the-art of MPC as well as the methods used for a solution of the associated optimization problem.
3. *Overview of Our Approach*: Chapter 3 gives a high level overview of the proposed methods and their connection to the existing literature.
4. *Main Results*: Chapters 4-8 describe in details three developed algorithms for solution of problem (1.1). It is shown how the proposed methods differ and how they solve the goals stated in this work. Finally, the performance of the proposed methods is compared to the state-of-the-art methods on the several numerical benchmarks in Chapter 8.
5. *Conclusions*: Chapter 9 summarizes the results of our research, suggests possible topics for further research, and concludes the thesis. There is also a list of contributions of this dissertation thesis.

Some parts of this thesis build on the results that were previously published in collaboration with colleagues. Specifically, some parts of Sections 4.2, 4.5, 4.6, 6.1, 6.2 and partially Chapter 8 were presented in [J.1]. Section 5.1 is based on the research showed in [C.7]. The Newton Projection with Proportioning (NPP) algorithm presented in Chapter 7 was introduced in [C.5], and tip-in manuover of Section 8.4 was studied in [C.4].

Chapter 2

Background and State-of-the-Art

This chapter briefly introduces the Model Predictive Control (MPC) and shows how the underlying optimization problem with the form (1.1) is constructed from the original control goals. Then, the overview of the existing methods which solve problem (1.1) is presented.

2.1 Theoretical Background

MPC is an optimization based multivariable control strategy which can explicitly incorporate the constraints of the real controlled process. In MPC, the current control input is obtained by solving a finite N-horizon open-loop optimal control problem (typically a constrained QP problem) at each sampling instant, using the current state of the system as the initial state [59]. Hence, the so-called Receding Horizon Control (RHC) concept is established, i.e., the plan of control inputs $\boldsymbol{u}(0), \ldots, \boldsymbol{u}(N-1)$ is recomputed at each sampling instant with currently measured/estimated system state $\boldsymbol{x}$ as a parameter. Only the first control move $\boldsymbol{u}(0)$ is applied to the system, cf. [59]. This optimization demands a relatively large amount of computation compared to the traditional control methods and therefore MPC was limited to the processes with relatively large sampling times in the past. As the computation power of the processors increased in recent years and new solution methods were developed, MPC is widely used in industrial (see the excellent survey in [72]) or automotive applications, see, e.g., [C.6, 33, 49].

In this section, the MPC control approach is introduced. It is shown how the control goals and limitations are transformed to the optimization problem – box constrained strictly convex QP problem in a form of (1.1). Such QP has to be solved at each sampling time motivating the development of fast and reliable solvers.

2.1.1 Linear Model Predictive Control

Linear MPC is a strategy which uses a linear plant model to predict the future plant output for computing potential control action. The linear MPC is suitable for control of linear or close to linear plants. The standard approach for control of plants with weak nonlinearity is to use a set of switched local linear MPC controllers, see, e.g., [49, 33] with satisfactory performance. On

the other hand, when the controlled plant contains high nonlinearity, such an approach might lead to poor controller performance, and nonlinear MPC might be a method of choice with the cost of higher computational demand.

There exist various flavors of linear MPC in the literature. These differ mainly in the used model type and form of the cost function which is minimized over the finite horizon by the MPC algorithm. It has been shown that MPC based on the $1/\infty$-norm simplifies the solution of the underlying optimization problem, see, e.g., [6, 92] since it leads to the solution of Linear Programming (LP). On the other hand, the $1/\infty$-norm based MPC tends to generate unwanted aggressive control, hence, most of the industrial application of MPC uses quadratic cost function which generates smoother control trajectories. In this work, we focus on the standard class of the linear MPC problems that uses a linear model of the plant and quadratic costs on both the states/outputs and control inputs subject to box constraints. Further, it is showed how to extend the type of constraints to the output limits while preventing the box constrained form of the resulting optimization problem by constraint softening.

2.1.1.1 Model and Predictions

Most control laws, for example, PID (proportional, integral and derivative), are reactive. Therefore they generate a control action as a response to observed tracking error. On the other hand, MPC is a predictive control strategy which explicitly computes the predicted response of the closed loop system over the finite horizon into future, e.g., to avoid the violation of a limit which will come in the future.

To predict the future response of a process, we must have a model of how the process behaves. In particular, this model must show the dependence of output/state on the current state and the present and future inputs.

A few types of models can be used for prediction. They are either based on input/output description, e.g., FIR, ARX, ARMAX, or the state-space models, see for example [81]. This work will focus on linear time-invariant state-space models.

Consider a discrete-time linear time-invariant system described by a state space model

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \tag{2.1a}$$
$$\boldsymbol{y}(k) = \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k), \tag{2.1b}$$

where $\boldsymbol{x}(k) \in \mathbb{R}^{n_x}$, $\boldsymbol{u}(k) \in \mathbb{R}^{n_u}$, $\boldsymbol{y}(k) \in \mathbb{R}^{n_y}$ are the state vector, the input vector, and the output vector respectively, for all time instants $k \geq 0$. And $\boldsymbol{A} \in \mathbb{R}^{n_x \times n_x}$, $\boldsymbol{B} \in \mathbb{R}^{n_x \times n_u}$, $\boldsymbol{C} \in \mathbb{R}^{n_y \times n_x}$, $\boldsymbol{D} \in \mathbb{R}^{n_y \times n_u}$.

The state and output predictions over a finite prediction horizon $k+1, \ldots k+N$, can be expressed as a function of the initial state or its estimate $\tilde{\boldsymbol{x}}(k)$ and control input sequence $\mathsf{U}(k)$ [56] by the recursive use of (2.1) as

$$\mathsf{X} = \mathsf{A}\tilde{\boldsymbol{x}}(k) + \mathsf{B}\mathsf{U}(k),$$
$$\mathsf{Y} = \mathsf{C}\tilde{\boldsymbol{x}}(k) + \mathsf{D}\mathsf{U}(k),$$

where $\mathsf{U}(k) = \left[\boldsymbol{u}(k)^T,\ \boldsymbol{u}(k+1)^T,\ \ldots\ \boldsymbol{u}(k+N-1)^T\right]^T$ and

$$\mathsf{X} = \left[\boldsymbol{x}(k+1)^T,\ \boldsymbol{x}(k+2)^T,\ \ldots\ \boldsymbol{x}(k+N)^T\right]^T,$$
$$\mathsf{Y} = \left[\boldsymbol{y}(k)^T,\ \boldsymbol{y}(k+1)^T,\ \ldots\ \boldsymbol{y}(k+N-1)^T\right]^T,$$

and

$$\mathsf{A} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^{N-1} \\ \boldsymbol{A}^N \end{bmatrix}, \qquad \mathsf{B} = \begin{bmatrix} \boldsymbol{B} & 0 & & & \\ \boldsymbol{AB} & \boldsymbol{B} & \ddots & & \\ \vdots & & \ddots & & \\ \boldsymbol{A}^{N-2}\boldsymbol{B} & & & \boldsymbol{B} & 0 \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \ldots & \boldsymbol{AB} & \boldsymbol{B} \end{bmatrix},$$
$$\mathsf{C} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \vdots \\ \boldsymbol{CA}^{N-1} \\ \boldsymbol{CA}^N \end{bmatrix}, \qquad \mathsf{D} = \begin{bmatrix} \boldsymbol{D} & & & & \\ \boldsymbol{CB} & \boldsymbol{D} & & & \\ \boldsymbol{CAB} & \boldsymbol{CB} & \ddots & & \\ \vdots & & & & \\ \boldsymbol{CA}^{N-2}\boldsymbol{B} & & & & \\ \boldsymbol{CA}^{N-1}\boldsymbol{B} & \boldsymbol{CA}^{N-2}\boldsymbol{B} & \ldots & \boldsymbol{CB} & \boldsymbol{D} \end{bmatrix}. \tag{2.2}$$

2.1.1.2 Problem Formulations

Individual control problems (e.g. system stabilization, following the output references, etc.) are formulated by the cost function of the optimization problem in the MPC. The associated weights set the relative importance of the potentially conflicting goals. The limitations of the controlled plant are transformed to the constraints of the optimization problem.

There are several approaches how to recast the MPC problem into an optimization problem. In *sparse formulation*, the predictions of system states are added as the decision variables and the set of constraints (2.1a) are incorporated into the problem as equality constraints over the prediction horizon. The resulting optimization problem then has $N(n_x + n_u)$ decision variables, but the problem matrices are sparse. This approach is usually used in combination with IPMs as it can lead to significant speed-ups, mainly for larger prediction horizons. The problem structure can be exploited to have linear computational complexity at each iteration with respect to N, see, e.g., [90].

To eliminate the equality constraints and preserve the linear computational complexity of each iteration, the dual optimization of the sparse problem was formulated and solved by the Gradient Projection algorithm for QP (GPQP) algorithm in [3]. The main drawback of the dual formulation is the double number of the decision variables as compared to the primal one in the case of box constrained QP problem. A different approach to eliminate the equality constraints was developed in [C.7] where they were transformed into the cost function with penalization weighted by the fixed ℓ_2-norm penalty. This trick preserves the problem structure while keeping the simple constraints enabling the application of projection-type of algorithm with linear complexity in prediction horizon for each iteration.

In *dense formulation*, the state variables are eliminated using predictions (2.2) which are substituted to the cost function. Even if the sparse structure is lost, the number of decision variables is reduced to $N \cdot n_u$. Hence this formulation is favorable for the problems with a relatively short prediction horizon ($N < 50$ samples) or for the problems with a large number of states where sparse formulation would lead to high number of optimization variables. Moreover, dense formulation enables to use the move blocking strategies [20] to reduce the number of optimization variables further.

Recently the *sparse condensed formulation* by coordinate transformation using state feedback leading to banded prediction matrices (2.2) and hence sparse Hessian was derived in [51]. It was shown that each iteration of IPMs applied to proposed formulation has complexity linear with respect to N. Unfortunately, the coordinate transformation led to the loss of a simple type of constraints preventing effective use of projection-type methods for the solution of such a problem.

The following text is focused on the dense formulation since it provides the optimization problem which contains only box type of constraints and the least number of optimization variables. Hence, it can be solved by the projection-type methods to speed-up the convergence with the computational complexity which is driven by the number of optimization variables in the proposed methods.

In the following, the two most common MPC control problems are introduced and it is shown how they can be recast to the form of the QP problem (1.1) using the dense formulation. The control input is assumed to belong to the convex compact set $\widetilde{\Omega} = \{\boldsymbol{u} \in \mathbb{R}^{n_u} : \underline{\boldsymbol{u}} \leq \boldsymbol{u} \leq \overline{\boldsymbol{u}}, \underline{\boldsymbol{u}} < 0 < \overline{\boldsymbol{u}}\}$ which contains the origin in its interior. The constant terms are omitted in the following control problem formulations since they do not influence the minimizer.

Regulator Problem The goal of the *regulator problem* is to drive the system state $\boldsymbol{x}$ to the origin while minimizing the control effort. This can be formulated as an MPC problem

$$
\begin{aligned}
f^*_{\mathsf{MPC}_{reg}}(\tilde{\boldsymbol{x}}(k)) \triangleq \min_{\boldsymbol{u}(k),\ldots,\boldsymbol{u}(k+N-1)} \quad & \frac{1}{2}\boldsymbol{x}(k+N)^T\boldsymbol{P}\boldsymbol{x}(k+N) + \\
& \frac{1}{2}\sum_{i=1}^{N-1}\boldsymbol{x}(k+i)^T\boldsymbol{Q}\boldsymbol{x}(k+i) + \frac{1}{2}\sum_{i=0}^{N-1}\boldsymbol{u}(k+i)^T\boldsymbol{R}\boldsymbol{u}(k+i), \\
\text{s.t.} \quad & \boldsymbol{x}(k+1+i) = \boldsymbol{A}\boldsymbol{x}(k+i) + \boldsymbol{B}\boldsymbol{u}(k+i),\ i = 0,\ldots,N-1 \\
& \boldsymbol{u}(k+i) \in \widetilde{\Omega},\ i = 0,\ldots,N-1, \\
& \boldsymbol{x}(k) = \tilde{\boldsymbol{x}}(k), \qquad (2.3a)
\end{aligned}
$$

where matrix $\boldsymbol{Q} \in \mathbb{R}^{n_x \times n_x}$ is Symmetric Positive Semidefinite (SPS), matrices $\boldsymbol{R} \in \mathbb{R}^{n_u \times n_u}$ and $\boldsymbol{P} \in \mathbb{R}^{n_x \times n_x}$ are SPD.

The problem (2.3) can be rewritten in the form (1.1) denoting $\boldsymbol{z} = \mathsf{U} \in \mathbb{R}^n$, $n = N \cdot n_u$,

$$
\begin{aligned}
\boldsymbol{H} &= \mathsf{B}^T\mathsf{Q}\mathsf{B} + \mathsf{R}, & \boldsymbol{h}(\tilde{\boldsymbol{x}}(k)) &= \left(\mathsf{B}^T\mathsf{Q}\mathsf{A}\right)\tilde{\boldsymbol{x}}(k), & \underline{\boldsymbol{z}} &= \mathbf{1}_N \otimes \underline{\boldsymbol{u}}, \\
\mathsf{Q} &= \mathsf{diag}\left(\boldsymbol{I}_N \otimes \boldsymbol{Q}, \boldsymbol{P}\right), & \mathsf{R} &= \boldsymbol{I}_N \otimes \boldsymbol{R}, & \overline{\boldsymbol{z}} &= \mathbf{1}_N \otimes \overline{\boldsymbol{u}},
\end{aligned}
$$

where $\otimes$ refers to the Kronecker product, $\boldsymbol{I}_N$ and $\boldsymbol{1}_N$ being the identity matrix and the column vector of ones of the dimension N, respectively.

Tracking Problem The tracking problem represents the situation when the system outputs should typically follow the given reference trajectory $\boldsymbol{r}(k+i) \in \mathbb{R}^{n_y}$ for $i = 0, \cdots, N-1$. Furthermore, the control movement $\Delta\boldsymbol{u}$ is penalized rather than control effort to guarantee the off-set free tracking in the nominal case without the presence of model inaccuracy and disturbances, see [81] for a detailed explanation.

Hence we design MPC controller as the following optimization problem

$$
\begin{aligned}
f^*_{\mathsf{MPC}_{trc}}(\boldsymbol{\theta}(k)) \triangleq \min_{\boldsymbol{u}(0),\ldots,\boldsymbol{u}(N-1)} \quad & \frac{1}{2}\sum_{i=0}^{i+N-1} \Big((\boldsymbol{y}(k+i) - \boldsymbol{r}(k+i))^T \boldsymbol{Q}_y \left(\boldsymbol{y}(k+i) - \boldsymbol{r}(k+i)\right)\Big) + \\
& \frac{1}{2}\sum_{i=0}^{i+N-1} \left(\Delta\boldsymbol{u}(k+i)^T \boldsymbol{R} \Delta\boldsymbol{u}(k+i)\right), \\
\text{s.t.} \quad & \boldsymbol{x}(k+1+i) = \boldsymbol{A}\boldsymbol{x}(k+i) + \boldsymbol{B}\boldsymbol{u}(k+i),\ i = 0,\ldots,N-1 \\
& \boldsymbol{y}(k+i) = \boldsymbol{C}\boldsymbol{x}(k+i) + \boldsymbol{D}\boldsymbol{u}(k+i),\ i = 0,\ldots,N-1 \\
& \Delta\boldsymbol{u}(k+i) = \boldsymbol{u}(k+i) - \boldsymbol{u}(k-1+i),\ i = 0,\ldots,N-1 \\
& \boldsymbol{u}(k+i) \in \widetilde{\Omega},\ i = 0,\ldots,N-1, \\
& \boldsymbol{x}(k) = \tilde{\boldsymbol{x}}(k), \\
& \boldsymbol{\theta} = \left[\tilde{\boldsymbol{x}}(k)^T, \boldsymbol{r}(k)^T, \boldsymbol{u}(k-1)^T\right]^T,
\end{aligned} \tag{2.4}
$$

where $\boldsymbol{Q}_y \in \mathbb{R}^{n_y \times n_y}$ is SPS and $\boldsymbol{R} \in \mathbb{R}^{n_u \times n_u}$ is SPD. The term $\boldsymbol{u}(k-1) \in \mathbb{R}^{n_u}$ represents the control action applied in the last sampling period. Assuming that the reference trajectory is constant over the prediction horizon, i.e., $\boldsymbol{r}(k) = \boldsymbol{r}(k+i)$ for $i = 0,\ldots,N-1$, the problem (2.4) can be rewritten in the form (1.1) denoting $\boldsymbol{z} = \mathsf{U} \in \mathbb{R}^n$, $n = N \cdot n_u$, $\boldsymbol{\theta} \in \mathbb{R}^{n_x+n_y+n_u}$ as

$$
\begin{aligned}
&\boldsymbol{H} = \mathsf{D}^T\mathsf{Q}\mathsf{D} + \boldsymbol{K}^T\mathsf{R}\boldsymbol{K}, \quad \boldsymbol{h}(\boldsymbol{\theta}) = \left[\ \mathsf{D}^T\mathsf{Q}\left[\ \mathsf{C} \quad -\boldsymbol{1}_N \otimes \boldsymbol{I}_{n_y}\ \right] \quad \boldsymbol{K}^T\mathsf{R}\boldsymbol{M}\ \right]\boldsymbol{\theta}, \quad \underline{\boldsymbol{z}} = \boldsymbol{1}_N \otimes \underline{\boldsymbol{u}}, \\
&\mathsf{Q} = \boldsymbol{I}_N \otimes \boldsymbol{Q}_y, \qquad\qquad \mathsf{R} = \boldsymbol{I}_N \otimes \boldsymbol{R}, \qquad\qquad \overline{\boldsymbol{z}} = \boldsymbol{1}_N \otimes \overline{\boldsymbol{u}},
\end{aligned}
$$

and

$$
\boldsymbol{K} = \begin{bmatrix} \boldsymbol{I}_{n_u} & & & \\ -\boldsymbol{I}_{n_u} & \boldsymbol{I}_{n_u} & & \\ & \ddots & \ddots & \\ & & -\boldsymbol{I}_{n_u} & \boldsymbol{I}_{n_u} \end{bmatrix}, \quad \boldsymbol{M} = \begin{bmatrix} -\boldsymbol{I}_{n_u} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.
$$

2.1.1.3 Soft Constraints

Often, the control goal is at time-step k to keep the j-th system output within limits and/or follow the reference, i.e., to solve (2.3) or (2.4) with additional constraints

$$
\underline{y_j}(k+i) \leq y_j(k+i) \leq \overline{y_j}(k+i),\ i \subseteq \chi, \tag{2.5}
$$

with $\chi = \{i \in \mathbb{N} : 1 \leq i \leq N-1\}$ and $\underline{y_j} < \overline{y_j}$. These constraints can be violated (e.g. as an effect of unfavorable initial condition or disturbance) leading to an unfeasible solution of QP problem (1.1). To prevent this to happen, the common practice in the process industry is to formulate (2.5) as *soft* constraints. These constraints can be violated, but any violation is penalized in the objective function and simple constrained slack variables are added as decision variables, see, e.g. [59, 46]. Hence the following term is added to the cost function of (2.3) or (2.4)

$$\frac{1}{2}\sum_{i\in\chi}\left(\boldsymbol{C}_{j,:}\boldsymbol{x}(k+i)+\boldsymbol{D}_{j,:}\boldsymbol{u}(k+i)-\varepsilon_i\right)^T\rho\left(\boldsymbol{C}_{j,:}\boldsymbol{x}(k+i)+\boldsymbol{D}_{j,:}\boldsymbol{u}(k+i)-\varepsilon_i\right),$$

with $\rho > 0$ and the auxiliary optimization variable $\boldsymbol{\varepsilon} \in \mathbb{R}^{|\chi|}$ constrained as

$$\underline{y_j}(k+i) \leq \varepsilon_i \leq \overline{y_j}(k+i),\ i \subseteq \chi.$$

Note that each sample along the prediction horizon where the soft constraint is considered adds an optimization variable, hence impacting the solution time of the QP solver. To reduce this, often the cardinality of χ set is reduced to select only several points along the prediction horizon using the limited bandwidth of the controlled process, see, e.g. [J.2].

2.2 Previous Results and Related Work

In the last two decades, there has been rapid development in the optimization algorithms for problem (1.1). This effort together with the common growth of the computational power of processors enabled sampling times in orders of milliseconds or even microseconds in MPC applications. Available solvers can be divided into two main groups based on where the most computational complexity is performed: *off-line* and *online* methods. The limitation of the off-line group is the large memory demand depending on the problem size. Hence its use is limited to small size problems. On the other hand, the online, iterative methods enable to solve mid to large scale problems at the cost of bigger associated computational burden.

2.2.1 Explicit Solution

An off-line approach, so called explicit solution, was suggested in [8], where parametric nature of problem (1.1) was used. It was shown that the solution is a piece-wise affine function in the parameter space and the cost function value is a piece-wise quadratic over a polyhedral portion, so-called critical region, of the parameter space. Each critical region represents a portion of parameter space where certain constraints are activated.

The solution of (1.1) is divided into off-line and online phase. In the off-line phase the polyhedral partitioning and corresponding affine mapping of the solution is computed and stored. In the online phase, the solution is a fast look-up table process to search in the critical region in which the current state measurement lies. Then the solution is computed as an affine function based on the found critical region and measured state. It was shown, that the memory footprint

of the explicit solution growths exponentially with the number of constraints [15]. Hence its use is limited to small scale problems only.

To reduce the memory demand of explicit solution and its approximations several semi-explicit approaches were presented. In [92] approximation of explicit solution was used as warm start of ASM for MPC based on 1-norm. The ASM was combined with precomputed and stored factors of associated KKT system in [15] leading to a reduction of memory demand compared to the explicit solution. The partial enumeration presented in [67] reduces memory by using a fact that only a few combinations of active constraints are activated during run time of controller, hence only a subset of critical regions is saved. A simplified problem is computed and the subset of stored critical regions is updated when the algorithm is facing a not stored combination of active constraints in real-time.

Although the explicit solution is superior in solution time, a limiting factor of it and its approximations, like those presented in [52, 9, 44], is the number of critical regions which grows exponentially with the number of constraints in the MPC problem [15]. This growth limits the explicit type of solution to small control problems only, mainly because of limited memory in embedded systems. See the survey of explicit solution [1] for details.

2.2.2 Active Set Methods

Active Set Methods (ASMs) are ones of the most common methods for solving the QP problem (1.1) for a fixed parameter [15]. The main idea of ASMs is to identify the set of active constraints at the solution with so-called *active set* in a finite number of iterations. The ASMs maintain an estimate of the active set, which is a linearly independent set of constraints that are satisfied at the beginning of each algorithm iteration. At each iteration of ASM the active set is updated so that it converges to the optimal one. Further, the auxiliary optimization problem defined by the current active set, so-called face problem, is solved either by direct solvers or by Krylov space methods at each iteration.

The ASMs exist in primal [38, 39], and dual variant [5, 41]. The well-known drawback of the ASMs is that if the initial active set is very different from the optimal one, the algorithm needs many iterations since the active set is changing very slowly [3]. Typically only one change in the active set is performed per iteration. This fact may result in slow convergence for problems which arise in optimal control, where the control variables are often at the constraints for a large portion of the prediction horizon [10] when a transient operation is invoked by the change of the references or due to the effect of external disturbances in the MPC applications.

To improve this, non-feasible ASM tailored for MPC which allows multiple constraints to be added or removed from active set per iteration based on Lagrange multipliers was introduced in [61]. To prevent, but not ensure, returning unfeasible solution when a maximum number of iteration is exceeded, the method updates active set by preferring constraint's indices corresponding to "earlier" control moves in the planned control trajectory.

A different strategy was used in [33, 30] initially presented in [13] where ASM moves along the line in the parameter space towards the new value from the one from previous sample instant. Along the move it is checked whatever some constraint needs to be added or removed, allowing reuse of previous solution and associated factor of the KKT system for factor update

scheme. A drawback of this method is that it needs as many iterations as the difference in the optimal active sets of (1.1) for previous and the new parameter.

A natural benefit of the ASM is a possibility to select the initial iterate by the solution from the previous sample time in MPC by a warm start. The warm-start strategies were studied in more details in [C.8].

2.2.3 Interior Point Methods

Interior Point Methods (IPMs) solve the KKT conditions of QP problem directly by applying Newton method to the sequence of equality constrained problems or to a series of modified versions of the KKT conditions [18]. It was shown in [74, 25, 90] that IPMs can exploit the sparse structure of the problem to reduce the computational cost associated with the solution of the system of linear equations. There exist many variations namely the primal-dual methods, see, e.g., [65] and log-barrier method, see, e.g., [18, 90]. IPMs involve relatively small and a constant number of iterations independently to problem conditioning.

As was presented in [90] the log-barrier method can be effectively used for MPC by exploiting the sparsity pattern of the sparse MPC formulation. Further, the overall computation time is decreased by the early termination of the iterations. This, however, can lead to a violation of the process dynamics since it is considered as equality constraints which can not be fulfilled after a certain number of iterations.

For IPMs, a polynomial runtime guarantee of the number of algorithm iterations can be given. Unfortunately, this estimate is in practice far larger than the observed number of iterations [76]. The IPMs involve a small relatively constant number of iterations, which enhances the solution of a system of linear equations, usually much more expensive than for ASMs. It was shown in [74] that such system of linear equations can be solved at a cost proportional to the length of the prediction horizon when Riccati recursion is used for the sparse formulation of MPC. The same was achieved in [90] with block-wise Cholesky factorization when the warm-start with the fixed barrier parameter and fixed number of iterations were used. This resulted in a suboptimal solution, which can even violate the system dynamics equation.

The FORCES package [25] produces automatically generated code in which the Newton step computation [90] is enhanced by saving two back-substitutions using the rectangular factorization with possible speed-up for special instances. CVXGEN, universal IPM solver applicable also for MPC with automatically generated code was proposed in [57, 58]. Although automatically generated problem tailored code may result in superior solution time, the code size might increase rapidly with the problem size thus preventing embedded application due to the Read Only Memory (ROM) limitations of the target platform.

The proper warm starting of the IPMs is still an open theoretical question despite several recent attempts to solve this. The prediction from the previous sample time was used as the warm start together with a fixed barrier parameter in [90] leading to the sub-optimal solution only. In [84], the idea of constraints tightening was used to reduce the limits on the prediction horizon to obtain feasible point closer to the central path for which IPMs converge quickly at the cost of solving additional linear program.

2.2.4 Gradient Projection Methods

The Gradient Projection Methods (GPMs) are almost as simple as ASMs and provides a more rapid change of the working set than ASMs and they also inherit many good properties as warm-start capability. The GPM was firstly introduced by Rosen [80] as a generalization of the steepest descent method for optimization problems with convex constraints. The Rosen's algorithm moves along the straight line segment until the first new constraint is activated, hence limiting the change of the active set. This has been improved by Goldstein, Levitin and Polyak [42],[55] by allowing the movement along the projected arc on the constraint surface while maintaining the feasibility and allowing a larger change of the active set per iteration. It was shown, that under the non-degeneracy assumption, the algorithm of GPM identifies the optimal active set in a finite number of iterations [19, 63].

The gradient projection algorithm is then defined by

$$\boldsymbol{z}^{k+1} = \mathcal{P}_{\Omega}(\boldsymbol{z}^k - \alpha^k \nabla q(\boldsymbol{z}^k)),$$

where $\alpha^k > 0$ is either a fixed step-size or one obtained by the line search optimization along the negative gradient direction, $\nabla q(\boldsymbol{z}^k)$ is the gradient of q at $\boldsymbol{z}^k$. The projection of $\boldsymbol{y}$ onto Ω is the mapping $\mathcal{P}_{\Omega} : \mathbb{R}^n \to \Omega$ defined by

$$\mathcal{P}_{\Omega}(\boldsymbol{y}) = \arg\min_{\boldsymbol{z}\in\Omega} ||\boldsymbol{z} - \boldsymbol{y}||. \tag{2.6}$$

One of the main drawbacks of GPMs is that the computation of the projection (2.6) can be expensive for the general type of constraints [63] since it may lead to the QP problem in general. Thus, GPMs are limited to problems involving simple constraint sets such as spheres, Cartesian products of spheres, or the box constraints [10].

Another well-known drawback of GPM is that it shares the slow convergence rates (linear) with the steepest descent method, since, after a finite number of iterations, GPMs become a version of the steepest descent method restricted to the binding constraint manifold [10]. The convergence of the GPM with a fixed step size has been studied recently in [64] showing that GPM shows the global linear convergence of function value residuals with a convergence ratio $1 - O(1/\mathrm{cond}\,\boldsymbol{H})$.[1] Hence, such GPM method requires $O(1)\,\mathrm{cond}\,\boldsymbol{H}\ln(1/\varepsilon)$ iterations in order to meet ε-solution for $\varepsilon > 0$ with respect to the optimal cost function value.

2.2.5 Combination of Gradient Projection and Newton Method

To overcome the slow convergence of GPMs for ill-conditioned QP problems, super-linearly convergent modifications of GPM were presented in the literature. In [65], the GPM iteration serves as a tool for identification of the optimal active set and the solution of face problem defined by current active set is used as an improvement of the projected point at each iteration of the GPM algorithm. The projection of gradient was switched to Newton method once it is detected that probably optimal active set is reached by the algorithm in [10]. The detection is

[1] See Appendix A.1 for an overview of the convergence rates.

based on the heuristic defined by no change in the active set after the trial GPM iteration with a fixed step-size. The fixed step in the trial GPM iteration then allows choosing the compromise of the number of overall algorithm iterations versus the number of solutions of the face problem defined by the current active set. Furthermore, the Newton direction which points towards the face problem minimizer was also projected onto the feasible set, e.g., in [10, 3] to obtain a faster rate of convergence. While algorithm presented in [10] used computational expensive Generalized Armijo rule to project the Newton direction, computational cheaper Projected Line Search (PLS) presented in [65] has been used in [3]. PLS looks for the first local minimizer along the projected path and was in these algorithms used also for projection of gradient.

The disadvantage of the combined GPM and Newton method is the computational cost involved in a solution of the face problem defined by the current active set. This is partially solved by exploiting the structure of a dual problem as in [3] using the Riccati recursion, see, e.g., [4] and references therein. A disadvantage of this approach is that it is not possible to guarantee a primal feasibility of the result in the case of early termination of the iteration process. Another method is involvement of Krylov subspace methods as in [63] for larger problem size.

The common bottleneck of above mentioned combined methods is that the active set changes are not under control in a sense that it is not defined exactly when and which indices will be added to / removed from the active set. Hence, it might happen that some constraint indices are removed and added back to the active set in the following iteration or in the other part of the algorithm. This prevents to use effectively the factor update technique commonly used in the ASMs to speed-up solution of the face problem in case a direct solver is employed.

A different strategy has been presented in [11] where super-linearly convergent algorithm called Projected Newton Method (PNM) uses Generalized Armijo rule at each iteration to project the Newton direction which points towards the minimizer of the face problem defined only by those constraints which have positive associated Lagrange multiplier. Hence, the gradient projection part is completely eliminated and the blocking constraints are released immediately whenever they are detected to be not optimal by the Lagrange multiplier sign. This leads to the fact that the constraints indices might me removed prematurely and added back later to the active set increasing the complexity of the potential factor update technique for the solution of the face problem.

2.2.6 Fast Gradient Projection Methods

Nesterov's Fast Gradient Methods (FGMs) presented in [64] have attracted attention of the MPC community recently. While the complexity of each iteration of FGMs is comparable to GPMs, the FGMs converge with a much smaller convergence ratio $1 - O(1/\sqrt{\operatorname{cond} \boldsymbol{H}})$[2], see, e.g., [64]. FGMs require $O(1)\sqrt{\operatorname{cond} \boldsymbol{H}} \ln(1/\varepsilon)$ iterations, hence much less compared to GPMs especially for ill-conditioned QP problems. The basic idea underlying these methods is to relax the condition of a monotone decrease of sequence $\{q(\boldsymbol{z}^k)\}$. The iterations are based on the

[2] See Appendix A.1 for an overview of the convergence rates.

projection of gradient with fixed step-size computed at the point which is a combination of current and last iterate. FGMs exist in primal, e.g., [64, 76] and dual variants, e.g., [68].

It was shown that the upper bound on the number of iterations for any value of a parameter can be certified for problems with simple bounds off-line [78]. The certification of the upper bound of the required iterations for the dual of the sparse formulation of MPC problem, where the equality constraints corresponding to the state equation are relaxed, is discussed in [77, 76]. The certification results for the dual of dense MPC formulation are presented in [40]. The FGM for linear MPC with general polyhedral constraints on inputs and states applied on the dual sparse MPC problem is developed together with the certificate of iteration bound in [68] to reach primal accuracy. A fixed point implementation of FGM is studied in [69].

A common issue of the FGMs is the sensitivity of the number of iterations to the QP problem scaling in terms of spectral condition number of $\boldsymbol{H}$. Further, as the QP problem conditioning depends on the MPC controller tuning the number of the needed iterations might vary only because the change of the importance of the control goals. Although the number of iterations can be decreased by the proper problem pre-scaling (e.g., [76] or [68]), this cannot be applied in all situations due to its computational complexity. This arises when the Hessian of the QP problem is subject to change at each sample time. Hence the pre-conditioner would need to be computed and applied online. The change of the QP problem Hessian might occur either because of the change of the controller tuning or when the QP problem arises in the application of nonlinear MPC.

2.2.7 Proportioning Algorithms

The inability to effectively control the changes in the active set was declared as important disadvantage of the combined Newton/gradient projection algorithms. It appears that this problem has been solved in a mathematical community by the proportioning strategy. It has been proposed independently by Friedlander and Martinez [36] and Dostál [26] for the solvers combining the conjugate gradients and projecting steps and further developed into the Modified Proportioning with Reduced Gradient Projections (MPRGP) algorithm in [28] where proportioning serves as an effective precision control of the solution of auxiliary linear problems.

The MPRGP algorithm explores the face defined by the current active set by the conjugate gradient until the component of the gradient which corresponds to the active set dominates the violation of the KKT conditions or an unfeasible iteration is generated. In the first case, the active set is expanded by the gradient projection with a fixed step length, otherwise, it is reduced by the so-called proportioning step. The algorithm has been proved to enjoy a global R-linear rate of convergence and was successfully applied to the solution of large problems discretized by billions of variables.

The algorithm presented in [14] executes the proportioning with non-monotone line search whenever a component of the gradient which corresponds to the active set dominates the violation of the KKT conditions. Otherwise, the face problem defined by the current active set is solved. If the face minimizer is not feasible, the backtracking search along the polygonal path defined by the direction to the minimizer is executed. It was proved that the algorithm converges under nondegeneracy assumption. Further, a remedy to guarantee the convergence

in the presence of dual degeneracy based on the modified test of proportionality has been established.

2.2.8 Other Methods

Dual Newton Strategy based on non-smooth Newton method was introduced in [32] and applied to the nonlinear MPC in [35]. The main idea of the approach is to decouple the individual stages along the prediction horizon by dualizing the equality constraints of system dynamics. The resulting algorithm exploits the sparsity structure and enables the use of warm starting. Projection-free quadratic programming algorithm for non-negative least squares derived from the variational calculus was presented in [21]. The converge was proved, and linear convergence rate reported. The main advantage of the method is simple iterate update rule and its parallelization which is a topic for further development. The Alternating Direction Method of Multipliers (ADMM) originally developed in the mid-1970s by Gabay and Mercier in [37] has seen resurgence recently due the increase of computational power, see the survey in [17]. The ADMM method uses Gauss-Seidel-type algorithm to minimize the augmented dual problem. Since it breaks the optimization problem into smaller sub-problems, it has been used as an enabler to distributed MPC in [88]. R-linear rate of convergence of the method has been proved in [73], although the numerical experiments suggest that ADMM often outperforms the FGMs [86].

Chapter 3

Overview of Proposed Approach

Although many approaches of the combination of gradient / Newton projection algorithms are present in the literature, none of them fit for the real-world MPC problem applications to the author's knowledge. Either the algorithm is designed mainly for large scale problems, e.g., [10],[63],[65], hence behaves poorly in the embedded applications where the number of variables is limited. Or it operates in a dual space, e.g., [3], where it is not possible to guarantee a primal feasibility of the result in the case of premature termination of the iteration process due to the limited resources. The second issue of the combined gradient projection algorithms is when and how the second order information should be used to expand the active set and improve the rate of convergence of the algorithm.

An attempt of development of the combined gradient / Newton projection method in context of embedded MPC was done by the author of this thesis in [C.1]. Therein, the algorithm, based on [10], combined the GPM algorithm with projection of the Newton direction computed by the Cholesky factorization and the null space method [65, Ch. 16.2]. With such a set-up, computational complexity of the solution of the face problem at each iteration has been reduced to $\mathcal{O}((n-m)^3)$, with m being the number of currently active constraints. Hence the algorithm is suitable for the application of the MPC, where the control variables are often at the constraints for a large portion of the prediction horizon [10]. The projection of both directions (gradient, Newton direction) is executed via PLS algorithm presented in [65] instead of more computational expensive Generalized Armijo rule. The algorithm uses a heuristic from [10], hence the active set is tested for a change after the application of trial gradient projection iteration with a fixed step-size. Whenever there is no change in the active set, the Newton direction projection using PLS is applied. Otherwise the GPM iteration is executed using PLS. To extend the algorithm for a large scale problems, the null-space method was replaced by the conjugate gradient method in [C.2]. Furthermore, a favorable distribution of the eigenvalues of $\boldsymbol{H}$ arising in the modified MPC problem was exploited in [C.3] to speed-up the solution of auxiliary linear problems by the conjugate gradients combined with the gradient/Newton projection algorithm.

The bottleneck of algorithms [C.1, C.2, C.3] is that the decision when the Newton direction should be computed and applied relies only on a heuristic which depends on the parameter (fixed

step size of trial GPM iteration) which is hard to tune. To overcome this, the heuristic was replaced in [70] by a simple observation that if a new constraint is activated by the application of the Newton step, the active set should be expanded. Otherwise, either algorithm finds an optimum or the active set is reduced to remove blocking constraints to allow a further cost function decrease. Furthermore, as the Newton step points to the solution of the face problem it can be used as a direction for a projection for active set expansion using PLS method. On the other hand, as GPM algorithm assures the algorithm convergence it can be used for active set reduction.

Such algorithm was applied to the dense MPC formulation in [70] in the domain of automated process control utilizing also the Cholesky factorization and null space method. This was later developed as a part of a commercially available software product for the diesel engines air-path control using MPC, see [49]. The authors report the use of the software tool in several embedded applications using either directly ECU or rapid prototyping system: the air-path control of dual loop exhaust gas re-circulation diesel engine [C.6], the temperature control of the diesel oxidation catalyst [54], and the thermal management of combustion engine [53]. Hence, contrary to [C.1, C.2, C.3] the algorithm of [70] executes the solution of the face problem at each iteration and based on the result it decides what should be executed next. The algorithm either applies Newton direction up to the 1st boundary at each iteration, followed by either projection of the Newton direction by PLS [65, Ch. 16.7] or the GPM by PLS. By this, one evaluation of the gradient is eliminated compared to algorithms presented in [3] and [C.1, C.2, C.3]. The reported number of iterations of this method is typically 15 for a wide range of problem sizes which is in agreement with the algorithm of [63] which uses Conjugate Gradient (CG) iterations.

Original algorithm [70] was using PLS for a projection of both the Newton direction and the gradient. The modification, which reduces the computational complexity of GPM iteration we presented in [C.7] where we use a fixed step-size derived from the eigenvalues of $\boldsymbol{H}$ resulting in the same typical number of iterations. Furthermore, the algorithm was applied to the sparse approximation of MPC problem. This was done to decrease the computational cost of a solution of the face problem in primal variables. The result was a linear growth of complexity of each iteration with respect to the prediction horizon compared to traditional cubic one. The projection of the gradient utilizing PLS routine was added into the algorithm of [C.7] in [J.2] for the application to the nonlinear MPC as $\boldsymbol{H}$ is changing at each sampling time. The resulting method was applied to the fuel optimization based cruise controller problem running on the standard production ECU.

The main disadvantage of the combined GPM/Newton method is that the changes in the active set are not under control, preventing effective use of factor update for the solution of the face problem, which is commonly used in ASMs. On the contrary, the proportioning strategy of [36, 26] used for large scale box constrained optimization enables to decide when the active set should be reduced / expanded. The test is based on the detection which part of the KKT conditions dominates their violation. This is exploited in the MPRGP algorithm presented in [28] as effective precision control of the solution of auxiliary linear problems.

In this work we firstly describe in details the algorithm of [C.7] with several improvements for the reduction of the computational complexity of the PLS method. Then we propose

two new algorithms for an efficient solution of QP problems arising from MPC, which try to exploit the advantages of the algorithms MPRGP and [C.7, J.2]. The algorithms exploit the control of MPRGP to "look ahead" in order to avoid unnecessary expansion of the active set in combination with full exploitation of the Newton directions. Resulting algorithms identify the optimal active set in 15-20 relatively low complex iterations in average independently of the initial iterate, the QP problem size, and conditioning.

All of the proposed algorithms are active set based strategies which minimize the face problem by direct solver via Cholesky factorization with factor update. Furthermore, all algorithms use projected Newton direction path utilizing the modified PLS originally presented in [65] to expand the active set along the Newton direction which points towards the face minimizer defined by the active set. It is shown that the original algorithm of PLS can be modified and extended so that it can update the cost function gradient along the projected path with complexity linear in the QP problem size.

The main difference between the proposed algorithms is the way how and when the active set is reduced. The first two algorithms (Combined Gradient and Newton Projection (CGNP) and Proportioning with Newton Directions (PND)) use for the active set reduction a *proportioning* step, which is the projection of the *chopped gradient* with fixed step-size. The chopped gradient represents the components of the gradient which corresponds to the active set and not optimal sign of the associated Lagrange multiplier. Similarly as in ASMs, the CGNP executes the active set reduction only in a case that the face problem minimizer is feasible. As this strategy does not take into account the value of the Lagrange multipliers (but only their sign), it might reduce the active set lately in the iteration, hence limiting unnecessarily further cost function decrease.

To improve this, the PND algorithm uses *proportionality test* introduced independently by Friedlander and Martinez [36] and Dostál [26] for the solvers combining the conjugate gradients and projecting steps. The proportionality test decides which part of gradient dominates the violation of the KKT conditions of the QP problem. It was used as an effective precision control of the solution of auxiliary linear problems by the MPRGP algorithm [28]. The MPRGP algorithm explores the face defined by the current active set by the conjugate gradient until the component of the gradient which corresponds to the active set dominates the violation of the KKT conditions or an unfeasible iteration is generated. In the first case, the active set is expanded by the gradient projection with a fixed step length, otherwise, it is reduced by the proportioning step. Similarly, the PND algorithm expands the active set via projected Newton direction path utilizing the PLS routine until the component of the gradient which corresponds to the active set dominates the violation of the KKT conditions. Then the active set is reduced by the proportioning step. Hence, the PND algorithm tries to exploit the advantages of the algorithms MPRGP and CGNP. The algorithm exploits the control of MPRGP to "look ahead" in order to avoid unnecessary expansion/reduction of the active set in combination with full exploitation of the Newton directions.

The third proposed algorithm which we call Newton Projection with Proportioning (NPP) is inherited from the PND algorithm. It also uses the proportionality test but it executes only projection of the Newton direction for the expansion of the active set while reduction is handled via modified definition of the face problem. Since the proportioning step is eliminated, this algorithm does not require convergent step-size and thus it is suitable, e.g., for nonlinear MPC

where the Hessian matrix $\boldsymbol{H}$ is changing at each sampling period.

This rest of this thesis is organized as follows:

- Chapter 4 presents the shared basic ingredients of the proposed algorithms.
- Chapter 5 describes the CGNP algorithm and gives the proof of convergence together with algorithm properties analysis.
- Chapter 6 depicts the PND algorithm and gives the proof of convergence together with algorithm properties analysis and comparison to CGNP algorithm.
- Chapter 7 shows the NPP algorithm and gives the proof of convergence together with algorithm properties analysis and comparison to CGNP and PND algorithms.
- Chapter 8 compares the performance of implementation of the proposed methods with the state-of-the-art methods on selected numerical experiments.

Some parts of this thesis build on the results that were previously published in collaboration with colleagues. Specifically, some parts of Sections 4.2, 4.5, 4.6, 6.1, 6.2 and partially Chapter 8 were presented in [J.1]. Section 5 is based on the research showed in [C.7]. The NPP algorithm presented in Chapter 7 was introduced in [C.5], and tip-in manuover of Section 8.4 was studied in [C.4].

Chapter 4

Basic Ingredients

Let us first review notations and present shared basic ingredients of the proposed algorithms. Firstly the projection operator which is used as a tool for effective change of the active set is introduced. Then basic notation regarding the most important sets is presented together with definition of chopped and free gradient. Next, the face problem is introduced and solution based on the Cholesky factorization with factor update is given. Further, the *fast computation* of gradient update is introduced. Finally, the tool of expansion/reduction of the active set via projected Newton direction path utilizing the Projected Line Search (PLS) of [65] is repeated with implementation details and tricks leading to its linear computation complexity with respect to QP problem size.

4.1 Projection

The important ingredient of the proposed algorithms is projection of the vector which lies outside the feasible set by the orthogonal projection. This tool is used to expand or release multiple constraint indices from the active set at once allowing rapid identification of the optimal active set.

In general, the projection of $\boldsymbol{z} \in \mathbb{R}^n$ onto set $\Gamma \subset \mathbb{R}^n$ is the mapping $\mathcal{P}_\Gamma : \mathbb{R}^n \to \Gamma$ defined by

$$\mathcal{P}_\Gamma(\boldsymbol{z}) = \arg\min_{\boldsymbol{x}} \|\boldsymbol{x} - \boldsymbol{z}\| \quad \text{s.t.} \quad \boldsymbol{x} \in \Gamma.$$

The computation of the projection can be expensive for general type of constraints [63] since it may lead to the QP problem in general. Thus projection use is effectively limited to problems involving simple constraint sets such as positive orthant, spheres, Cartesian products of spheres or the box constraints [10].

In case of feasible set defined by box constraints as $\Omega = \{\boldsymbol{z} : \underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}}, \underline{\boldsymbol{z}} < \overline{\boldsymbol{z}}\}$ the projection $\mathcal{P}_\Omega$ is defined for any n-vector $\boldsymbol{z}$ by

$$\mathcal{P}_\Omega(\boldsymbol{z}) = \boldsymbol{z}^+,$$

where the entries of $\boldsymbol{z}^+$ are defined as $z_i^+ = \max\{\underline{z}_i, \min\{\overline{z}_i, z_i\}\}$, i.e. the projection is a simple clamping rule.

4.2 Quantitative Refinement of the KKT Conditions

For an arbitrary n-vector $\boldsymbol{z}$, let us define the gradient $\boldsymbol{g}(\boldsymbol{z})$ of the cost function q in (1.2) by

$$\boldsymbol{g}(\boldsymbol{z}) = \boldsymbol{H}\boldsymbol{z} + \boldsymbol{h}.$$

The optimal solution $\boldsymbol{z}^*$ of (1.1) is fully determined by the KKT optimality conditions[1], see Appendix A.2 or [18] for more details. Denoting $\boldsymbol{g}^* = \boldsymbol{g}(\boldsymbol{z}^*)$, the KKT conditions require that for $i = 1, \ldots, n$

$$z_i^* = \begin{cases} \overline{z}_i & \text{implies } g_i^*(\boldsymbol{z}^*) \le 0, \\ \underline{z}_i & \text{implies } g_i^*(\boldsymbol{z}^*) \ge 0, \\ \underline{z}_i \le z_i^* \le \overline{z}_i & \text{implies } g_i^*(\boldsymbol{z}^*) = 0. \end{cases} \tag{4.1}$$

Denoting $\mathcal{I} = \{1, 2, \ldots, n\}$, let us define the upper, lower, and active set of $\boldsymbol{z}$ as $\mathcal{U} = \{i \in \mathcal{I} : z_i = \overline{z}_i\}$, $\mathcal{L} = \{i \in \mathcal{I} : z_i = \underline{z}_i\}$, and $\mathcal{A} = \{\mathcal{U} \cup \mathcal{L}\}$ respectively. The complement of the active set is called the *free set* $\mathcal{F} = \mathcal{I} \backslash \mathcal{A}$. Furthermore, from the definition of box constraints in (1.1) $\mathcal{L} \cap \mathcal{U} = \emptyset$.

To give an alternative reference to the KKT conditions (4.1), let us define similarly as in [27] the *free gradient* $\boldsymbol{\varphi}$ and the *chopped gradient* $\boldsymbol{\beta}$ by

$$\begin{aligned} \varphi_i(\boldsymbol{z}) &= g_i(\boldsymbol{z}) \quad \text{for } i \in \mathcal{F}, \qquad & \varphi_i(\boldsymbol{z}) &= 0 \quad \text{for } i \in \mathcal{A}, \\ \beta_i(\boldsymbol{z}) &= 0 \quad \text{for } i \in \mathcal{F}, \qquad & \beta_i(\boldsymbol{z}) &= g_i^{\#}(\boldsymbol{z}) \quad \text{for } i \in \mathcal{A}, \end{aligned} \tag{4.2}$$

where

$$g_i^{\#}(\boldsymbol{z}) = \begin{cases} \max\{g_i(\boldsymbol{z}), 0\} & \text{if } i \in \mathcal{U}, \\ \min\{g_i(\boldsymbol{z}), 0\} & \text{if } i \in \mathcal{L}. \end{cases}$$

Vectors $-\boldsymbol{\varphi}(\boldsymbol{z})$ and $-\boldsymbol{\beta}(\boldsymbol{z})$ are orthogonal and feasible decrease directions of q at $\boldsymbol{z}$ [27]. See also Figure 4.1. Thus based on the definition (4.2) the KKT conditions are satisfied if and only if the *projected gradient* $\boldsymbol{\nu} = \boldsymbol{\varphi} + \boldsymbol{\beta}$ is equal to zero.

4.3 Face Problem Solution

We shall use the active set strategy which reduces the solution of (1.1) to the solution of a sequence of equality constrained problems. Thus in each iteration, we need to solve the auxiliary

[1] Note that problem (1.1) is not subject to equality constraints, hence the associated Lagrange multiplier is zero. The condition of the non-negativity of the Lagrange multipliers of the simple constraints is reduced to the first two rows of (4.1).

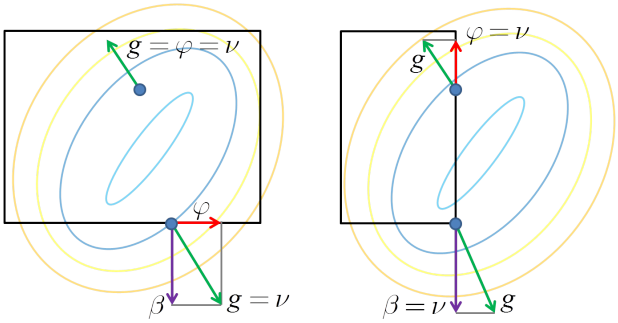


Figure 4.1: Gradient splitting to the free gradient $\boldsymbol{\varphi}$ and the chopped gradient $\boldsymbol{\beta}$ and the definition of the projected gradient $\boldsymbol{\nu}$.

minimization problem - so-called *face problem*

$$\tilde{\boldsymbol{z}} = \arg\min_{\boldsymbol{z}\in\Phi} q(\boldsymbol{z}), \tag{4.3}$$

where

$$\Phi = \{\boldsymbol{z} \in \mathbb{R}^n : z_i = \overline{z}_i \;\text{ for }\; i \in \mathcal{U}^k \;\text{ and }\; z_i = \underline{z}_i \;\text{ for } i \in \mathcal{L}^k\}$$

is the *face* defined by the lower and upper active sets $\mathcal{L}^k$, $\mathcal{U}^k$ of the indices which are predicted to be active in the solution at the k-th iteration of the algorithm. Since $\tilde{\boldsymbol{z}}_{\mathcal{U}^k} = \overline{\boldsymbol{z}}_{\mathcal{U}^k}$ and $\tilde{\boldsymbol{z}}_{\mathcal{L}^k} = \underline{\boldsymbol{z}}_{\mathcal{L}^k}$, the equality constrained problem (4.3) is equivalent to the unconstrained minimization problem defined only for the variables $\boldsymbol{z}_{\mathcal{I}\setminus\mathcal{A}^k}$.

Problem (4.3) can be rewritten formally as

$$\tilde{\boldsymbol{z}} = \arg\min q(\boldsymbol{z}) \quad \text{s.t. } \boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}, \tag{4.4}$$

where $\boldsymbol{A}$ is defined by the rows of the identity matrix with dimension n as $\boldsymbol{A} = \left[\boldsymbol{I}^T_{\mathcal{U}^k}, \boldsymbol{I}^T_{\mathcal{L}^k}\right]^T$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{b} = \left[\overline{\boldsymbol{z}}^T_{\mathcal{U}^k}, \underline{\boldsymbol{z}}^T_{\mathcal{L}^k}\right]^T$ with $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ representing identity matrix. Applying first-order optimality conditions (KKT conditions) and denoting the Lagrange multipliers associated with equality constraints as $\boldsymbol{\lambda}$, the solution of (4.4) reduces to the following set of linear equations [65]

$$\begin{bmatrix} \boldsymbol{H} & -\boldsymbol{A}^{\boldsymbol{T}} \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{z}}^* \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} -\boldsymbol{h} \\ \boldsymbol{b} \end{bmatrix}. \tag{4.5}$$

Assuming $\boldsymbol{z}^k \in \Phi$, let introduce the notation

$$\tilde{\boldsymbol{z}} = \boldsymbol{z}^k + \boldsymbol{p}^k, \tag{4.6}$$

with Newton direction $\boldsymbol{p}^k$ used later in the proposed algorithm. Than (4.5) can be rearranged to the form with so-called KKT matrix [65] as

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{A}^{\boldsymbol{T}} \\ \boldsymbol{A} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} -\boldsymbol{p}^k \\ \boldsymbol{\lambda}^* \end{bmatrix} = \begin{bmatrix} \boldsymbol{g}^k \\ \boldsymbol{c}^k \end{bmatrix}, \tag{4.7}$$

with $\boldsymbol{g}^k = \boldsymbol{H}\boldsymbol{z}^k + \boldsymbol{h}$ and $\boldsymbol{c}^k = \boldsymbol{A}\boldsymbol{z}^k - \boldsymbol{b}$.

The problem (4.3) can be solved by the Krylov subspace methods [82], which are advantageous, e.g., in large-scale optimization. Since a low precision of the solution of auxiliary problems (4.3) is often sufficient, the performance of the approaches of solution of (1.1) based on Krylov subspace methods can be often improved by an efficient adaptive precision control [27]. On the other hand, if the size of the problem is relatively small, as we consider in this work, we can find the exact solution more efficiently by a direct solver as shown in the followings.

There are multiple ways of solving the linear set of equations (4.7) by direct solvers. When $\boldsymbol{H}$ and $\boldsymbol{A}$ are sparse or have a special structure (e.g., arrow shape or banded), the direct solution can be used. The direct method is based on the factorization methods of the KKT matrix and solution of the associated linear equations via two successive back substitutions. Since the KKT matrix is indefinite for $m \geq 1$, see e.g. [65, Section 16.2], the LU factorization or symmetric indefinite factorization have to be employed [65].

Another methods of direct solution of (4.7) are *range space* and *null space* methods. This terminology arises because the working set can be viewed as defining two complementary subspaces: the range space of vectors that can be expressed as linear combinations of the rows of $\boldsymbol{A}$, and the null space of vectors orthogonal to the rows of $\boldsymbol{A}$ [39]. Both methods use elimination of one equation of (4.7) to solve the second one. The main difference of methods is which equation is eliminated. Range space method eliminates the first equation of (4.7) and solves the problem in the Lagrange multipliers space, i.e. with m variables. In null space method, the second equation is eliminated leading to the fact that only free variables have to be solved, i.e. $n - m$ variables. The work associated with both methods is proportional to the number of variables for dense matrix $\boldsymbol{H}$. Hence range space method is more efficient when the number of equality constraints m is small [65]. Conversely, the null space method is the most effective when the number of bounded variables is close to n. Since for optimal control, we can expect many constraints to be active [10] and we usually do not need to form the Lagrange multipliers of the problem (4.4), we focus next to the null space method.

In the null space method the Newton direction $\boldsymbol{p}^k$ is partitioned into two components as

$$\boldsymbol{p}^k = \boldsymbol{Y}\boldsymbol{p}_y^k + \boldsymbol{Z}\boldsymbol{p}_z^k, \tag{4.8}$$

where the columns of $\boldsymbol{Z} \in \mathbb{R}^{n\times(n-m)}$ are a basis for the null space of $\boldsymbol{A}$ and the columns of $\boldsymbol{Y} \in \mathbb{R}^{n\times m}$ are a basis for the range space of $\boldsymbol{A}^T$. The first term represents a particular solution of equality constraints while the second term represents a correction on the subspace of the free variables. Then one can substitute (4.8) into the first equation of (4.7), multiply the equation by $\boldsymbol{Z}^T$ and using the orthogonality property $\boldsymbol{A}\boldsymbol{Z} = 0$ and non-singularity of $[\boldsymbol{Y}|\boldsymbol{Z}] \in \mathbb{R}^{n\times n}$ to receive

$$\left(\boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z}\right)\boldsymbol{p}_z^k = -\left[\boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Y}\boldsymbol{p}_y + \boldsymbol{Z}^T\boldsymbol{g}^k\right]. \tag{4.9}$$

The Lagrange multipliers are get via substitution of (4.8) into the first equation of (4.7) and multiplying by $\boldsymbol{Y}^T$ and solution of the following linear system

$$\left(\boldsymbol{A}\boldsymbol{Y}\right)^T\boldsymbol{\lambda}^* = \boldsymbol{Y}^T\left(\boldsymbol{g}^k + \boldsymbol{H}\boldsymbol{p}^k\right). \tag{4.10}$$

Let assume that $\boldsymbol{z}^k$ in (4.6) fullfills the equality constraints $\boldsymbol{A}\boldsymbol{z}^k = \boldsymbol{b}$. Note that this assumption can be made since in the proposed algorithm the problem (4.4) is solved exactly based on the active set in $\boldsymbol{z}^k$. Therefore, the constraint violation $\boldsymbol{c}^k = \boldsymbol{A}\boldsymbol{z}^k - \boldsymbol{b} = \boldsymbol{0}$ and by non-singularity of $\boldsymbol{AY}$ (see [65]) we can assume that $\boldsymbol{p}_y^k = \boldsymbol{0}$ and (4.9) can be simplified to the following equation

$$\boldsymbol{G}\boldsymbol{p}_z^k = -\boldsymbol{r}^k, \tag{4.11}$$

where $\boldsymbol{G} = \boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z}$ and $\boldsymbol{r}^k = \boldsymbol{Z}^T\boldsymbol{g}^k$ are *reduced Hessian* and *gradient*, respectively.

Thanks to the special structure of matrix $\boldsymbol{A}$ for the box constraints considered within this work, the matrices $\boldsymbol{Z}$ and $\boldsymbol{Y}$ contain columns of an identity matrix hence there is no need to construct them. Rather it is possible to form the reduced Hessian and gradient directly by selection of rows and columns as [C.1]

$$\boldsymbol{G} = \boldsymbol{H}_{\mathcal{I}\setminus\mathcal{A}^k,\mathcal{I}\setminus\mathcal{A}^k}, \qquad \boldsymbol{r}^k = \boldsymbol{g}^k_{\mathcal{I}\setminus\mathcal{A}^k}. \tag{4.12}$$

Then since $\boldsymbol{p}_y^k = \boldsymbol{0}$ and special form of matrix $\boldsymbol{Z}$ the Newton direction is

$$\boldsymbol{p}^k = \begin{cases} \boldsymbol{p}^k_{\mathcal{I}\setminus\mathcal{A}^k} & = -\boldsymbol{G}^{-1}\boldsymbol{r}^k \\ \boldsymbol{p}^k_{\mathcal{A}^k} & = \boldsymbol{0} \end{cases}, \tag{4.13}$$

where notation $\boldsymbol{a}_{\mathcal{B}} = \boldsymbol{e}$ means that $a_{\mathcal{B}_i} = e_i$ for $i = 1, \ldots, |\mathcal{B}|$.

Remark 4. Note that the form of the reduced Hessian, $\boldsymbol{G}$ in (4.12) is independent of whether the lower or the upper bound constraint is activated. This results from the fact that the corresponding rows of constraint gradient $\boldsymbol{A}$ are linearly dependent if either lower or upper constraints of i-th component are activated.

Lemma 4.3.1. Since $\boldsymbol{H} \in \mathbb{R}^{n\times n}$ is a SPD matrix, the reduced Hessian $\boldsymbol{G} = \boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z}$ is also a SPD for a full column rank matrix $\boldsymbol{Z} \in \mathbb{R}^{n\times(n-m)}$.

Proof. Considering positive definiteness of $\boldsymbol{H}$ and substitution $\boldsymbol{x} = \boldsymbol{Z}\boldsymbol{y}$ for $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{y} \in \mathbb{R}^{n-m}$ we can write

$$\boldsymbol{y}^T\boldsymbol{G}\boldsymbol{y} = \boldsymbol{y}^T\boldsymbol{Z}^T\boldsymbol{H}\boldsymbol{Z}\boldsymbol{y} = \boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} > 0 \quad \text{for all } \boldsymbol{y} \in \mathbb{R}^{n-m}, \boldsymbol{y} \neq \boldsymbol{0}$$

proving the lemma. □

As reduced Hessian remains positive definite (see Lemma 4.3.1), the problem (4.13) can be solved, e.g., by the Cholesky factorization (see e.g. [65] for details) of reduced Hessian in $\mathcal{O}\left((n-m)^3\right)$ as

$$\boldsymbol{G} = (\boldsymbol{R}^k)^T\boldsymbol{R}^k,$$

where $\boldsymbol{R}^k \in \mathbb{R}^{(n-m)\times(n-m)}$ is an upper triangular matrix with strictly positive diagonal. Then the forward and backward substitutions (see Appendix B.2 and B.3) can be performed as

$$(\boldsymbol{R}^k)^T\boldsymbol{y} = -\boldsymbol{r}^k \longrightarrow \boldsymbol{y}, \qquad \boldsymbol{R}^k\boldsymbol{p}^k = \boldsymbol{y} \longrightarrow \boldsymbol{p}^k,$$

with $\boldsymbol{y} \in \mathbb{R}^{n-m}$ and assuming the triangularity of the matrix $\boldsymbol{R}^k$ in $2(n-m)^2$ flops when $m = |\mathcal{A}^k|$ denotes the cardinality of the active set $\mathcal{A}^k$.

For the later use the following lemma shows that the Newton direction $\boldsymbol{p}^k$ defined by (4.13) is always a descent direction.

Lemma 4.3.2. Let $\boldsymbol{z} \in \Omega$, $\mathcal{A}$ is the corresponding active set and the Newton direction $\boldsymbol{p}$ is a nonzero solution of

$$\boldsymbol{p} = \begin{cases} \boldsymbol{p}_{\mathcal{I}\setminus\mathcal{A}} & = -\boldsymbol{G}^{-1}\boldsymbol{r} \\ \boldsymbol{p}_{\mathcal{A}} & = \boldsymbol{0} \end{cases},$$

with $\boldsymbol{G} = \boldsymbol{H}_{\mathcal{I}\setminus\mathcal{A},\mathcal{I}\setminus\mathcal{A}}, \boldsymbol{r} = \boldsymbol{g}_{\mathcal{I}\setminus\mathcal{A}}$. Then $\boldsymbol{p}$ is a descent direction for any active set $\mathcal{A}$.

Proof. Since $\boldsymbol{p}_{\mathcal{A}} = 0$ based on the definition in (4.13) only the components $\boldsymbol{p}_{\mathcal{I}\setminus\mathcal{A}}$ have to be analyzed. Then using Lemma 4.3.1, the reduced Hessian $\boldsymbol{G}$ remains positive definite for any $\mathcal{A}$ thus also its inverse. Hence we get

$$\boldsymbol{g}^T\boldsymbol{p} = \boldsymbol{g}_{\mathcal{I}\setminus\mathcal{A}}^T\,\boldsymbol{p}_{\mathcal{I}\setminus\mathcal{A}} = \boldsymbol{g}_{\mathcal{I}\setminus\mathcal{A}}^T\left(-\boldsymbol{G}^{-1}\boldsymbol{r}\right) = -\|\boldsymbol{g}_{\mathcal{I}\setminus\mathcal{A}}\|_{\boldsymbol{G}^{-1}} < 0,$$

which is exactly the definition of the descent direction see e.g. [34, Section 2.3]. □

4.4 Factors Updates

To reduce the cost of calculation of the Cholesky factor corresponding the active set $\mathcal{A}^k$, we employ the factor update scheme. The standard way of factor update [83, 43], deals with only one change of active set in the context of ASMs by rank-one update of the factor. On the other hand, the proposed algorithm can add or remove an arbitrary number of constraints at each iteration. Therefore, instead of successive rank-one update, we employ multiple rank update leading to the decrease of the computational complexity of the update scheme. We distinguish three phases of the update of the active set:

1. detection of the change of the active set from previous iteration,
2. adding new constraints to $\mathcal{A}^k$,
3. removing constraints from $\mathcal{A}^k$.

Each operation has some specifics described in details in the following text.

The Cholesky factor from the last iteration is updated in $\mathcal{O}(n^2)$ Floating point operations (flops) per a change of the active set. Hence the complexity of each update is relatively low, but it is still important to keep the number of changes of the active set small.

Let consider a known upper triangular Cholesky factor $\boldsymbol{R} \in \mathbb{R}^{(n-m^{k-1})\times(n-m^{k-1})}$ of reduced Hessian in (4.12) from $(k-1)$-th iteration corresponding to the active set $\mathcal{A}^{k-1}$ as

$$\boldsymbol{G}^{k-1} = \boldsymbol{H}_{\mathcal{I}\setminus\mathcal{A}^{k-1},\mathcal{I}\setminus\mathcal{A}^{k-1}} = (\boldsymbol{R}^{k-1})^T\boldsymbol{R}^{k-1}.$$

Remark 5. Since the sequence of problem (1.1) is typically solved in the MPC for $\boldsymbol{x}_k \approx \boldsymbol{x}_{k+1}$, it is possible to use a solution from the previous sample as an initial iteration. Moreover, as the solution represents a future trajectory of control inputs in MPC, one can shift the solution from the previous sample to improve the initial iteration to establish a *warm start* setup, see, e.g., [C.8] and references therein. Further, it is possible to reuse the factor of the reduced Hessian in (4.13) to establish a so-called *hot start* of the algorithm.

4.4.1 Detection of Active Set Changes

The update of the factor is different depending on the type of change of the active set. Hence, the changes in the active set are determined and grouped into two sets:

1. Constraint indices to be added: $\mathcal{A}^k_{\text{add}} = \mathcal{A}^k \setminus \mathcal{A}^{k-1}$
2. Constraint indices to be removed: $\mathcal{A}^k_{\text{rem}} = \mathcal{A}^{k-1} \setminus \mathcal{A}^k$

Than based on the active set $\mathcal{A}^{k-1}$ from $(k-1)$-th iteration, the reduced Hessian (4.12) in k-th iteration takes form

$$\boldsymbol{G}^k = \boldsymbol{H}_{\mathcal{I}\setminus\mathcal{A}^k,\mathcal{I}\setminus\mathcal{A}^k} = \boldsymbol{H}_{\left(\mathcal{I}\setminus(\mathcal{A}^{k-1}\cup\mathcal{A}^k_{\text{add}})\right)\cup\mathcal{A}^k_{\text{rem}},\left(\mathcal{I}\setminus(\mathcal{A}^{k-1}\cup\mathcal{A}^k_{\text{add}})\right)\cup\mathcal{A}^k_{\text{rem}}}. \tag{4.14}$$

Therefore the update of the Cholesky factor can be done in two stages: 1) the Cholesky factor of the reduced Hessian corresponding to the active set with added new constraints $\mathcal{A}^{k-1} \cup \mathcal{A}^k_{\text{add}}$ is computed updating the factor from $(k-1)$-th iteration and 2) factor from the previous stage is updated to cover the active set with removed constraints indices $\mathcal{A}^k_{\text{rem}}$ from the active set. The addition of new constraints indices to the active set leads to the decrease of the dimension of the reduced Hessian and hence the factor. Therefore as the cost of factor update is directly related to the computational complexity of the factor update process it is profitable to keep the order of stages as indicated.

4.4.2 Adding New Constraints

According to (4.12) when new constraints $\mathcal{A}^k_{\text{add}}$ are added at the k-th iteration the reduced Hessian takes the form $\boldsymbol{H}_{\mathcal{I}\setminus(\mathcal{A}^{k-1}\cup\mathcal{A}^k_{\text{add}}),\mathcal{I}\setminus(\mathcal{A}^{k-1}\cup\mathcal{A}^k_{\text{add}})}$. Hence the rows and columns corresponding to the $\mathcal{A}^k_{\text{add}}$ have to be removed from the factor $\boldsymbol{R}^{k-1}$. The standard approach in ASMs solvers would be to perform $|\mathcal{A}^k_{\text{add}}|$-times removal of the column of the $\boldsymbol{R}^{k-1}$ and than return the upper triangular form by application of Givens rotation, see Appendix B.4 and references [83, 43]. Each removal of the row/column leads to a decrease of the dimension of the factor. Hence to reduce the computational complexity, we rather apply the removal of all columns at once and apply the Givens rotation to return the upper triangular form of updated factor denoted as $\boldsymbol{R}^k_{\text{add}}$. Hence the update of factor when adding new constraints can be executed in $\mathcal{O}\left(|\mathcal{A}^k_{\text{add}}|(n - m^{k-1} - |\mathcal{A}^k_{\text{add}}|)^2\right)$ instead of $\mathcal{O}\left(\sum_{i=1}^{|\mathcal{A}^k_{\text{add}}|}(n - m^{k-1} - i)^2\right)$ flops.

4.4.3 Removing Constraints

When some constraints are removed from the active set, the dimension of the reduced Hessian is increased by the rows and columns of the Hessian matrix corresponding to the $\mathcal{A}^k_{\text{rem}}$ indices as indicated in (4.14). Hence the Cholesky factor $\boldsymbol{R}^k_{\text{add}}$ can be updated by $|\mathcal{A}^k_{\text{rem}}|$-times application of rank-one update utilizing the back-substitution, see e.g. [83]. In our implementation the algorithm of Cholesky of an augmented matrix [2] has been used. It is a continuation of the Cholesky factorization when new data are appended to the existing factor. The algorithm is described in Appendix B.1 for the sake of completeness.

The benefit of this method is that it has about the same complexity as successive application of rank-one updates and its code can be shared with the Cholesky factorization used in the first iteration when the Cholesky factor is not available. The algorithm of Cholesky augmented matrix [2] starts by initializing the factor in the following form

$$\boldsymbol{R}^k = \begin{bmatrix} \boldsymbol{R}^k_{\text{add}} & \boldsymbol{S} \\ \boldsymbol{0} & \boldsymbol{T} \end{bmatrix}, \quad \boldsymbol{S} = \boldsymbol{H}_{\mathcal{I}\setminus(\mathcal{A}^{k-1}\cup\mathcal{A}^k_{\text{add}}),\mathcal{A}^k_{\text{rem}}}, \quad \boldsymbol{T} = \boldsymbol{H}_{\mathcal{A}^k_{\text{rem}},\mathcal{A}^k_{\text{rem}}}.$$

Then the Cholesky factorization is started from the first row of $\boldsymbol{R}^k$ but only the columns with index greater than $n_f = n - m^{k-1} - |\mathcal{A}^k_{\text{add}}|$ are processed leading to the computational complexity $\mathcal{O}\left(|\mathcal{A}^k_{\text{rem}}|^3\right) + \mathcal{O}\left(|\mathcal{A}^k_{\text{rem}}|^2 n_f\right) + \mathcal{O}\left(|\mathcal{A}^k_{\text{rem}}| n_f^2\right)$, see [2] for details.

4.5 Gradient Updates

To reduce the computational complexity associated with computation of the gradient, we employ gradient update rule based on the formula for $\delta > 0$ and direction $\boldsymbol{d}$

$$\boldsymbol{g}(\boldsymbol{z} + \delta\boldsymbol{d}) = \boldsymbol{H}(\boldsymbol{z} + \delta\boldsymbol{d}) + \boldsymbol{h} = \boldsymbol{g}(\boldsymbol{z}) + \delta\boldsymbol{H}\boldsymbol{d}.$$

Where the most computational demanding part $\boldsymbol{H}\boldsymbol{d}$ can be effectively computed after the face problem is solved and updated over the course of the PLS algorithm as shown later. But firstly the technical lemma showing the structure of the gradient in the face problem minimizer is stated.

Lemma 4.5.1. Let $\boldsymbol{z}^k \in \Omega$ with corresponding active set $\mathcal{A}^k = \mathcal{U}^k \cup \mathcal{L}^k$ and $\tilde{\boldsymbol{z}}^* \in \mathbb{R}^n$ be the solution of the face problem

$$\min_{\boldsymbol{z}\in\Phi} q(\boldsymbol{z}) \quad \text{s.t. } \Phi = \{\boldsymbol{z} : z_i = \overline{z}_i \ \text{ for } \ i \in \mathcal{U}^k \ \text{ and } \ z_i = \underline{z}_i \ \text{ for } i \in \mathcal{L}^k\}.$$

Then $g_i\left(\tilde{\boldsymbol{z}}^*\right) = 0$ for all $i \in \mathcal{I}\setminus\mathcal{A}^k$. Moreover, $\tilde{\boldsymbol{z}}^* \in \Omega$ implies that $\varphi_i\left(\tilde{\boldsymbol{z}}^*\right) = 0$ for all $i \in \mathcal{I}$.

Proof. Let the Jacobian matrix of active constraints be $\boldsymbol{A} = \left[\boldsymbol{I}^T_{\mathcal{U}^k}, \boldsymbol{I}^T_{\mathcal{L}^k}\right]^T$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and $\boldsymbol{\lambda}^*$ be associated Lagrange multipliers. Then analyzing the first equation of the KKT conditions (4.5) of the face problem we get

$$\boldsymbol{H}\tilde{\boldsymbol{z}}^* - \boldsymbol{A}^T\boldsymbol{\lambda}^* = -\boldsymbol{h} \longrightarrow \boldsymbol{H}\tilde{\boldsymbol{z}}^* + \boldsymbol{h} = \boldsymbol{g}\left(\tilde{\boldsymbol{z}}^*\right) = \boldsymbol{A}^T\boldsymbol{\lambda}^*.$$

Since the structure of the Jacobian matrix of constraints $\boldsymbol{A}$, there are only zero elements in the i-th columns of $\boldsymbol{A}$ for all $i \in \mathcal{I}\backslash\mathcal{A}^k$. Hence all i-th rows of $\boldsymbol{A}^T$ for all $i \in \mathcal{I}\backslash\mathcal{A}^k$ contain only zeros. Therefore,

$$(\boldsymbol{A}^T\boldsymbol{\lambda}^*)_i = g_i\left(\tilde{\boldsymbol{z}}^*\right) = 0 \text{ for all } i \in \mathcal{I}\backslash\mathcal{A}^k.$$

When $\tilde{\boldsymbol{z}}^* \in \Omega$, the active set of $\tilde{\boldsymbol{z}}^*$ is the same as of $\boldsymbol{z}^k$. Hence, using the definition of the free gradient (4.2), $\varphi_i\left(\tilde{\boldsymbol{z}}^*\right) = 0$ for all $i \in \mathcal{A}^k$ and

$$\varphi_i\left(\tilde{\boldsymbol{z}}^*\right) = g_i\left(\tilde{\boldsymbol{z}}^*\right) = 0 \text{ for all } i \in \mathcal{I}\backslash\mathcal{A}^k,$$

and using $\mathcal{I} = (\mathcal{I}\backslash\mathcal{A}^k) \cup \mathcal{A}^k$ we can conclude that $\varphi_i\left(\tilde{\boldsymbol{z}}^*\right) = 0$ for all $i \in \mathcal{I}$. □

The gradient at the minimizer of the face problem (4.3) has the form

$$\boldsymbol{g}(\tilde{\boldsymbol{z}}^k) = \boldsymbol{g}(\boldsymbol{z}^k + \boldsymbol{p}^k) = \boldsymbol{H}(\boldsymbol{z}^k + \boldsymbol{p}^k) + \boldsymbol{h} = \boldsymbol{g}(\boldsymbol{z}^k) + \boldsymbol{H}\boldsymbol{p}^k.$$

From Lemma 4.5.1 we have

$$g_i(\tilde{\boldsymbol{z}}^k) = 0 \quad \text{for } i \in \mathcal{I}\backslash\mathcal{A}^k$$

and so

$$(\boldsymbol{H}\boldsymbol{p}^k)_i = g_i(\boldsymbol{z}^k) \;\text{ for }\; i \in \mathcal{I}\backslash\mathcal{A}^k.$$

Hence when computing $\boldsymbol{H}\boldsymbol{p}^k$, only the components of $(\boldsymbol{H}\boldsymbol{p}^k)_i$, $i \in \mathcal{A}^k$ with the sparsity pattern of $\boldsymbol{p}^k$ (4.13) need to be evaluated, which requires $2m(n-m)$ flops instead of $2n^2$ as it is depicted in Algorithm 1 with $m = |\mathcal{A}^k|$ where notation $\mathcal{B}_i$ represents the i-th element of indices set $\mathcal{B}$, $\boldsymbol{a}_{\mathcal{B}} = \boldsymbol{e}$ means that $a_{\mathcal{B}_i} = e_i$ for $i = 1, \ldots, |\mathcal{B}|$ and $h_{i,j}$ is element of matrix $\boldsymbol{H}$ on the i-th row and the j-th column.

The graphical comparison of the computational complexity of computation of $\boldsymbol{H}\boldsymbol{p}^k$ for changing m is depicted in Figure 4.2. The Algorithm 1 is compared to the case when only the sparsity pattern of $\boldsymbol{p}^k$ is exploited and without any prior knowledge of $\boldsymbol{p}$. From the figure, it is possible to conclude that Algorithm 1 is about four times faster than the direct computation of $\boldsymbol{H}\boldsymbol{p}^k$. And more than two times faster than an algorithm which would exploit the sparsity pattern of $\boldsymbol{p}^k$ for $m \leq 100$ in case that $n = 200$.

Algorithm 1 Fast computation of $\boldsymbol{H}\boldsymbol{p}^k$ for $\boldsymbol{H} \in \mathbb{R}^{n\times n}$, $\boldsymbol{p}^k \in \mathbb{R}^n$, $p_i^k = 0$ for all $i \in \mathcal{A}^k$, $\boldsymbol{p}_{\mathcal{I}\backslash\mathcal{A}^k} = -\left(\boldsymbol{H}_{\mathcal{I}\backslash\mathcal{A}^k,\mathcal{I}\backslash\mathcal{A}^k}\right)^{-1}\boldsymbol{g}\left(\boldsymbol{z}^k\right)_{\mathcal{I}\backslash\mathcal{A}^k}$.

1: Let $m = |\mathcal{A}^k|$, $\mathcal{F}^k = \mathcal{I}\backslash\mathcal{A}^k$, $(\boldsymbol{H}\boldsymbol{p}^k)_i = 0$ for all $i \in \mathcal{I}$.
2: **for** $i = 1$ to $n - m$ **do**
3: $\quad(\boldsymbol{H}\boldsymbol{p}^k)_{\mathcal{F}_i^k} = g_{\mathcal{F}_i^k}\left(\boldsymbol{z}^k\right)$
4: $\quad$**for** $j = 1$ to m **do**
5: $\quad\quad(\boldsymbol{H}\boldsymbol{p}^k)_{\mathcal{A}_j^k} = (\boldsymbol{H}\boldsymbol{p})_{\mathcal{A}_j^k} + h_{\mathcal{A}_j^k,\mathcal{F}_i^k}\, p^k_{\mathcal{F}_i}$
6: $\quad$**end for**
7: **end for**

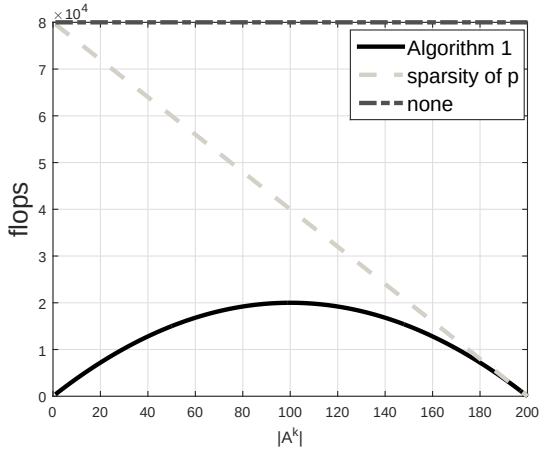


Figure 4.2: Comparison of the computational complexity of $\boldsymbol{H}\boldsymbol{p}^k$ when only sparsity pattern of $\boldsymbol{p}^k$ is used or when Algorithm 1 is used for $n = 200$.

4.6 Projected Line Search

To expand the active set, the proposed algorithms use the projected-Newton-direction path evaluated via Projected Line Search (PLS) of [65]. PLS has been used also in [C.3, C.7, 71] where it enabled fast identification of optimal active set when projecting the Newton direction $\boldsymbol{p}^k$. For the sake of completeness, the PLS is described in details here closely following the original algorithm of [65]. Contrary to the original algorithm of [65] which was used in the context of gradient projection, we are using it to project Newton direction, which is descent. The implementation details exploiting the update of $\boldsymbol{H}\boldsymbol{p}^k$ leading to the linear computational complexity of the PLS with respect to (1.1) problem size are also described. Furthermore, the algorithm to update gradient along the projected path with linear complexity in the QP problem size is described. To simplify the notation, the iteration index of the proposed algorithm is omitted in the description of the PLS algorithm.

The PLS is used to find the first local minimizer along the path towards the optimizer of (4.3). The PLS operation starts by computing first the *breakpoints* - the maximal step-sizes in a Newton direction where constraints are activated as

$$\bar{t}_i = \begin{cases} (\overline{z}_i - z_i) \ / \ p_i & \text{if } p_i > 0 \\ (z_i - \underline{z}_i) \ / \ p_i & \text{if } p_i < 0 \\ \infty & \text{otherwise.} \end{cases} \tag{4.15}$$

The breakpoints $\bar{t}_i$ for $i = 1, \ldots, n$ within vector $\bar{t}$ are sorted and duplicates are removed to construct set of step-sizes $\mathcal{T} = \{t_1, t_2, \ldots t_l\}$ with $0 < t_1 < t_2 < \ldots t_l$. The Shell sort algorithm [85] has been used as it is more convenient for embedded application since it does not use call stack compared, e.g., to Quick sort [91].

The projected-Newton-direction path as a function of any step-size $t \geq 0$ can be described as

$$z_i\left(t\right) = \begin{cases} z_i + t \ p_i & \text{if } t \leq \bar{t}_i \\ z_i + \bar{t}_i \ p_i & \text{otherwise,} \end{cases} \tag{4.16}$$

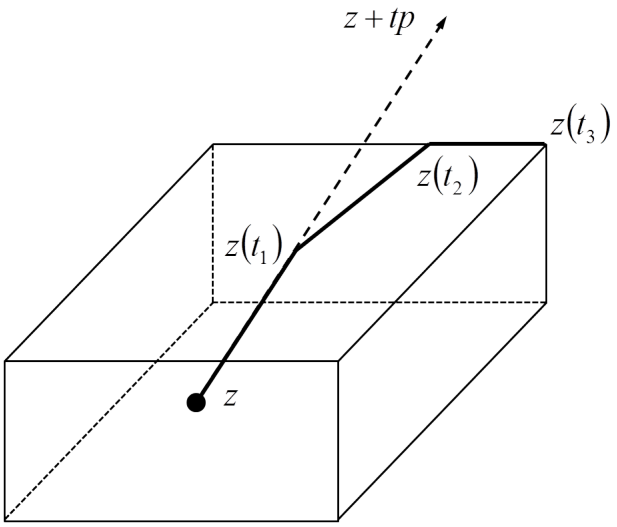


Figure 4.3: The example of piecewise-linear projected Newton direction path as $\boldsymbol{z}(t)$ in $\mathbb{R}^3$ [65].

see Figure 4.3 for graphical interpretation.

The intervals of sorted breakpoints from $\mathcal{T}$, i.e., $[0, t_1], [t_1, t_2], \ldots$ are then explored successively until the local minimizer is found within the interval or the cost function increase is detected. Let suppose that up to t_{j-1} the minimizer has not been found. Hence the line segment for the interval $[t_{j-1}, t_j]$ can be written as

$$\boldsymbol{z}(t) = \boldsymbol{z}(t_{j-1}) + \Delta t\, \boldsymbol{d}^{j-1}, \tag{4.17}$$

where

$$\Delta t = t - t_{j-1} \in [0, t_j - t_{j-1}]$$

and

$$d_i^{j-1} = \begin{cases} p_i & \text{if } t_{j-1} \leq \bar{t}_i \\ 0 & \text{otherwise.} \end{cases} \tag{4.18}$$

Then the one-dimensional quadratic model q_{j-1} is constructed within each $(j-1)$-th line segment $[\boldsymbol{z}(t_{j-1}), \boldsymbol{z}(t_j)]$ and its minimizer with respect to Δt is found as follows

$$\Delta t_{j-1}^* = -q'_{j-1}/q''_{j-1}$$

with

$$q'_{j-1} = \boldsymbol{h}^T \boldsymbol{d}^{j-1} + \boldsymbol{z}(t_{j-1})^T \boldsymbol{H} \boldsymbol{d}^{j-1} = \boldsymbol{g}(\boldsymbol{z}(t_{j-1}))^T \boldsymbol{d}^{j-1}, \qquad q''_{j-1} = \left(\boldsymbol{d}^{j-1}\right)^T \boldsymbol{H} \boldsymbol{d}^{j-1}. \tag{4.19}$$

Several situations may occur:

- if $q'_{j-1} > 0$, i.e. there is growth of functional along the segment and local minimizer of q_{j-1} is at $t = t_{j-1}$; else

- $\Delta t^*_{j-1} \in [0, t_j - t_{j-1}]$ there is a minimizer at $t = t_{j-1} + \Delta t^*_{j-1}$;
- otherwise the search continues on the next interval $[t_j, t_{j+1}]$ using (4.16).

The search direction $\boldsymbol{d}$ in (4.18) is identical for two followings breakpoints except the components corresponding to the newly activated constraints which are set to zero. This can be exploited as suggested in [65] without specific details by updating the term $\boldsymbol{H}\boldsymbol{d}^{j-1}$ in (4.19). For the sake of completeness we give details about the update of $\boldsymbol{H}\boldsymbol{d}^{j-1}$ here. Denoting $\boldsymbol{d}^0 = \boldsymbol{p}$ we can write

$$\left(\boldsymbol{H}\boldsymbol{d}^{j-1}\right)_i = \left(\boldsymbol{H}\boldsymbol{d}^{j-2}\right)_i - \sum_{s \in \mathcal{S}} h_{i,s}\, p_s \quad \mathcal{S} = \{s \in \mathcal{I} : t_{j-1} > \bar{t}_s\},$$

allowing to reduce the computational complexity of exploring each segment to $\mathcal{O}(n|\mathcal{S}|)$ flops where $|\mathcal{S}|$ represents the number of changes in the active set for explored segment which is typically only one. Note that since at the first segment the search direction $\boldsymbol{d}$ is identical to Newton direction $\boldsymbol{p}$, therefore $\boldsymbol{H}\boldsymbol{d}^0 = \boldsymbol{H}\boldsymbol{p}$ which is computed as described in Section 4.5.

Moreover, as PLS gradually moves along the projected-Newton-direction path with updated $\boldsymbol{H}\boldsymbol{d}^{j-1}$, it is possible to avoid the direct computation of the gradient at the resulting point and update it in $\mathcal{O}(n)$ flops. The update is done gradually depending whether the minimizer $\Delta t^*_{j-1} \in [0, t_j - t_{j-1}]$ has been found or not on the line segment. If it has not been found, i.e. the minimizer lies outside the line segment $[\boldsymbol{z}(t_{j-1}), \boldsymbol{z}(t_j)]$, the update is done as follows

$$\boldsymbol{g}(\boldsymbol{z}(t_j)) = \boldsymbol{g}(\boldsymbol{z}(t_{j-1})) + (t_j - t_{j-1})\boldsymbol{H}\boldsymbol{d}^{j-1}, \tag{4.20}$$

while in case that the minimizer $\Delta t^*_{j-1} \in [0, t_j - t_{j-1}]$ has been found we employ

$$\boldsymbol{g}(\boldsymbol{z}(t_j)) = \boldsymbol{g}(\boldsymbol{z}(t_{j-1})) + \Delta t^*_{j-1}\boldsymbol{H}\boldsymbol{d}^{j-1}.$$

The complexity of PLS is described in the following lemma.

Lemma 4.6.1. Let $\boldsymbol{z} \in \mathbb{R}^n$, $m^k = |\mathcal{A}^k|$, $\boldsymbol{p}^k$ be the solution of (4.13) and $\boldsymbol{H}\boldsymbol{p}^k$ is precomputed. Then the computational complexity of projected-Newton-direction path utilizing PLS algorithm together with updating of the gradient is

$$\mathcal{O}_{\mathrm{PLS}}(n) = 2(n - m^k) + \sum_{i=1}^{s}(2nr_i + 10n)$$

where s is the number of explored line segments, r_i number of changes in the active set on the i-th explored line segment.

Proof. First observe that $\boldsymbol{p}^k_{\mathcal{A}^k} = \boldsymbol{0}$ hence the computation of the breakpoints (4.15) can be executed in $2(n - m^k)$ flops. Secondly, at i-th explored segment: the $\boldsymbol{H}\boldsymbol{d}$ has to be updated in $2nr_i$ flops where r_i is the number of changes in the active set; the minimizer along the ray has to be computed using (4.19) in $6n$ flops; move to the end of line segment (4.16) in $2n$ flops; gradient can be updated as (4.20) in $2n$ flops. As a sum of all terms the result is evident. □

Chapter 5

Combined Gradient and Newton Projection

As mentioned in the Chapter 2, the algorithms that combine gradient projection with Newton-like methods were introduced, e.g., in [C.1, 3]. These methods use gradient projection step and add an *improvement step* by computing and application of the Newton direction on the face defined by the active constraints. Since the result of the application of the Newton direction can be infeasible with respect to the constraint, the Newton direction is projected onto a feasible set Ω, e.g., via PLS method or Armijo Generalized rule [10] to expand the active set. The existing algorithms differ when the improvement step is applied. In [3] the Newton direction is used at each iteration right after the gradient projection step is executed. On the other hand, in [C.1] the Newton direction is applied only when the working set is not changing after the application of the projection of gradient with fixed step-size. Hence, the performance of the algorithm strongly depends on the parameter of the heuristic which is not convenient for a real world application. To remove this disadvantage, the algorithm presented in this chapter executes the computation of the Newton direction at each iteration and furthermore the Newton direction is used for control whether the active set should be reduced by the projection of gradient or expanded by the projection of the Newton direction using PLS method.

In this chapter, the specific ingredients of the proposed method are described in detail together with the algorithm referring to the basic ingredients described in Chapter 4. Then the convergence of the algorithm is proved and the main properties of the algorithm are analyzed on the illustrative QP problem.

5.1 Algorithm

The bottleneck of previously mentioned methods is that the gradient always needs to be evaluated twice at each iteration, firstly for the gradient projection step and then for computation of the Newton direction. In this section a new modification of algorithm [3] and [C.7, 71] is presented. The resulting algorithm needs only to evaluate the gradient for the computation of

Algorithm 2 Combined Gradient and Newton Projection (CGNP) algorithm. Given a SPD matrix $\boldsymbol{H}$ of the order n, n-vectors $\boldsymbol{h}$, $\underline{\boldsymbol{z}}$, $\overline{\boldsymbol{z}}$, $\Omega = \{\boldsymbol{z} : \underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}}, \underline{\boldsymbol{z}} < \overline{\boldsymbol{z}}\}$, $\boldsymbol{z}^0 \in \Omega$, $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$, $\varepsilon > 0$ and $k_{\max} > 0$.

{Initialization}
Set $k = 0$, $\boldsymbol{g}\left(\boldsymbol{z}^k\right) = \boldsymbol{H}\boldsymbol{z}^k + \boldsymbol{h}$
for $k = 0$ to $k_{\max}$ **do**
  Solve (4.13) to obtain Newton direction $\boldsymbol{p}^k$.
  Compute $\boldsymbol{H}\boldsymbol{p}^k$ as described in Section 4.5
  $\alpha_f = \min\{1, \max\{\alpha : \boldsymbol{z}^k + \alpha\boldsymbol{p}^k \in \Omega\}\}$
  if $\alpha_f < 1$ **then**
    {Expansion step}
    $\left[\boldsymbol{z}^{k+1}, \boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right)\right] = \mathrm{PLS}\left(\boldsymbol{z}^k, \boldsymbol{p}^k, \boldsymbol{H}\boldsymbol{p}^k, \boldsymbol{g}\left(\boldsymbol{z}^k\right)\right)$
  else
    {Application of full Newton direction}
    $\boldsymbol{z}^{k+1/2} = \boldsymbol{z}^k + \boldsymbol{p}^k$
    $\boldsymbol{g}\left(\boldsymbol{z}^{k+1/2}\right) = \boldsymbol{g}\left(\boldsymbol{z}^k\right) + \left(\boldsymbol{H}\boldsymbol{p}^k\right)$
    if $\|\boldsymbol{\beta}\left(\boldsymbol{z}^{k+1/2}\right)\| \leq \varepsilon$ **then**
      $\{\boldsymbol{z}^{k+1/2}$ satisfies KKT conditions.$\}$
      Terminate with $\boldsymbol{z}^* = \boldsymbol{z}^{k+1/2}$.
    else
      {Proportioning step}
      $\boldsymbol{z}^{k+1} = \mathcal{P}_\Omega\left(\boldsymbol{z}^{k+1/2} - \alpha\boldsymbol{\beta}\left(\boldsymbol{z}^{k+1/2}\right)\right)$
      $\boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{H}\boldsymbol{z}^{k+1} + \boldsymbol{h}$
    end if
  end if
end for

the Newton direction using the gradient update rule. The whole gradient has to be recomputed only after the gradient projection step has been executed. Such situations occur only in the case that the application of the Newton direction does not add any new constraint to the working set. Hence as opposed to [3], the proposed algorithm performs either the projection of the Newton direction or projection of gradient. To prevent unwanted changes in the active set after the projection of the gradient and due to the accumulation of numeric errors, we project only the chopped gradient. To speed-up the computation of the Newton direction the factor update is employed.

Furthermore, when comparing the algorithm presented in [3] with our setup which we call Combined Gradient and Newton Projection (CGNP) (see Algorithm 2), the order of the GPM step and an improvement step is reversed. This is motivated by the fact that most of the time, the controlled plant is in steady state or close to it; hence there is no need to find a new active set, rather only to find an optimal solution on currently active constraints. This modification also allows a simple rule update of the gradient instead of its re-computation after the each step. The overview of the computational complexity of all the algorithm steps is presented in

Table 5.1: Computational complexity of the CGNP algorithm steps.

Algorithm Step	Complexity (flops)	Notation	Described
$\boldsymbol{g}\left(\boldsymbol{z}^k\right) = \boldsymbol{H}\boldsymbol{z}^k + \boldsymbol{h}$	$2n^2$		
$\boldsymbol{R}^{k-1} \rightarrow \boldsymbol{R}^k$	$\mathcal{O}\left(\lvert\mathcal{A}^k_{\text{add}}\rvert(n - m^{k-1} - \lvert\mathcal{A}^k_{\text{add}}\rvert)^2\right) +$ $\mathcal{O}\left(\lvert\mathcal{A}^k_{\text{rem}}\rvert^3\right) + \mathcal{O}\left(\lvert\mathcal{A}^k_{\text{rem}}\rvert^2 n_f\right) +$ $\mathcal{O}\left(\lvert\mathcal{A}^k_{\text{rem}}\rvert n_f^2\right)$	$m^{k-1} = \mathcal{A}^{k-1}$ $n_f = n - m^{k-1} - \lvert\mathcal{A}^k_{\text{add}}\rvert$	Section 4.4
$(\boldsymbol{R}^k)^T\boldsymbol{y} = -\boldsymbol{r}\left(\boldsymbol{z}^k\right)$ $\boldsymbol{R}^k\boldsymbol{p}^k = \boldsymbol{y}$	$2(n - m^k)^2$	$m^k = \lvert\mathcal{A}^k\rvert$	Section 4.3
$\boldsymbol{H}\boldsymbol{p}^k$	$2m^k(n - m^k)$		Section 4.5
$\alpha_f = \min\{1, \max\{\alpha : \boldsymbol{z}^k + \alpha\boldsymbol{p}^k \in \Omega\}\}$	$2n$		Section 5.1.1
$\left[\boldsymbol{z}^{k+1}, \boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right)\right] =$ PLS $\left(\boldsymbol{z}^k, \boldsymbol{p}^k, \boldsymbol{H}\boldsymbol{p}^k, \boldsymbol{g}\left(\boldsymbol{z}^k\right)\right)$	$2(n - m^k) + \sum_{i=1}^{s}(2nr_i + 10n)$	r_i constraints changed on the i-th line segment	Section 4.6
$\boldsymbol{z}^{k+1/2} = \boldsymbol{z}^k + \boldsymbol{p}^k$	n		
$\boldsymbol{g}\left(\boldsymbol{z}^{k+1/2}\right) = \boldsymbol{g}\left(\boldsymbol{z}^k\right) + \left(\boldsymbol{H}\boldsymbol{p}^k\right)$	n		
$\boldsymbol{z}^{k+1} = \mathcal{P}_\Omega\left(\boldsymbol{z}^{k+1/2} - \alpha\boldsymbol{\beta}\left(\boldsymbol{z}^{k+1/2}\right)\right)$	$4n$		Section 5.1.4
$\boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{H}\boldsymbol{z}^{k+1} + \boldsymbol{h}$	$2n^2$		

Table 5.1 concluding that the overall computational complexity of each algorithm iteration is $\mathcal{O}(n^2)$.

5.1.1 Face Problem Solution

The iteration of the CGNP algorithm starts by computation of the Newton direction $\boldsymbol{p}^k$ which points towards the minimizer of the face problem (4.3) defined by the current active set $\mathcal{A}^k$. The face problem is solved by the Cholesky factorization of the reduced Hessian and substitutions as described in Section 4.3 involving gradient at $\boldsymbol{z}^k$. To speed-up the Cholesky factorization process, the factor from the previous iteration is updated to capture the change in the active set as described in Section 4.4. Moreover, when the Hessian of the QP problem (1.1) is not changing, the factor from the previous sample time can be reused together with the initial iterate in MPC applications.

When Newton direction $\boldsymbol{p}^k$ is applied, two things can happen: (i) some new constraints are activated, i.e. $\boldsymbol{z}^k + \boldsymbol{p}^k \notin \Omega$, hence it can be applied only with the limited step size $\mu \in (0, 1)$ such that $\boldsymbol{z}^k + \mu\boldsymbol{p}^k \in \Omega$ (ii) full Newton direction can be applied without violation of any constraints, i.e. $\boldsymbol{z}^k + \boldsymbol{p}^k \in \Omega$.

In order to decide which situation occurs, the maximal step size $\alpha_f \in (0, 1]$ leading to the feasible iterate along the direction $\boldsymbol{p}^k$, i.e., $\alpha_f = \min\{1, \max\{\alpha : \boldsymbol{z}^k + \alpha\boldsymbol{p}^k \in \Omega\}\}$ is computed as

$$\alpha_f = \min\{1, \arg\min_{i \in \mathcal{I}} \alpha_i\}, \qquad \alpha_i = \begin{cases} (\overline{z}_i - z_i) \ / \ p_i & \text{if } p_i > 0 \\ (z_i - \underline{z}_i) \ / \ p_i & \text{if } p_i < 0 \\ \infty & \text{otherwise.} \end{cases} \tag{5.1}$$

5.1.2 Expansion Step

If $\alpha_f < 1$, i.e., the application of Newton direction leads to violation of some constraint and there is at least one missing constraint in the active set. Hence, the Newton direction $\boldsymbol{p}^k$ is used in the projected-Newton-direction path evaluated via PLS algorithm. PLS procedure of [65] is used to obtain the first local minimizer along the projected path completing the iteration. Our implementation of PLS when compared to the original algorithm of [65] additionally successively updates the gradient using the pre-computed $\boldsymbol{H}\boldsymbol{p}^k$ as described in Sections 4.6 and 4.5.

Remark 6. Note that the line search procedure is necessary for projection of Newton direction to preserve a convergence since it is not true in general that projection of the Newton direction with fixed step-size leads to an objective function decrease, see e.g. [3] for details.

5.1.3 Application of Newton Direction

If full Newton direction could be applied without violation of any constraints (i.e., $\alpha_f = 1$) then the intermediate iteration $\boldsymbol{z}^{k+1/2}$ is computed together with the gradient as

$$\boldsymbol{z}^{k+1/2} = \boldsymbol{z}^k + \boldsymbol{p}^k, \qquad \boldsymbol{g}\left(\boldsymbol{z}^{k+1/2}\right) = \boldsymbol{g}\left(\boldsymbol{z}^k\right) + (\boldsymbol{H}\boldsymbol{p}^k),$$

using the pre-computed $\boldsymbol{H}\boldsymbol{p}^k$ as described in Section 4.5.

Since $\boldsymbol{z}^{k+1/2} \in \Omega$ is the solution of the face problem (4.3), Lemma 4.5.1 predicts that free gradient $\boldsymbol{\varphi}\left(\boldsymbol{z}^{k+1/2}\right)$ is zero and hence $\boldsymbol{z}^{k+1/2}$ might be optimum when also the chopped gradient $\boldsymbol{\beta}\left(\boldsymbol{z}^{k+1/2}\right)$ is also zero[1]. Hence $\boldsymbol{z}^{k+1/2}$ is reported as solution of the problem (1.1) if

$$\beta_i\left(\boldsymbol{z}^{k+1/2}\right) = 0 \text{ for all } i \in \mathcal{A} \longrightarrow \|\boldsymbol{\beta}\left(\boldsymbol{z}^{k+1/2}\right)\| = 0.$$

For better numerical stability, the algorithm uses $\varepsilon > 0$ to detect the solution of problem (1.1) as

$$\|\boldsymbol{\beta}\left(\boldsymbol{z}^{k+1/2}\right)\| \leq \varepsilon.$$

5.1.4 Proportioning Step

If $\boldsymbol{z}^{k+1/2}$ is not KKT optimal, there are some blocking constraints which could be removed from the active set to decrease the objective function q. For this purpose, GPM step with PLS is used in [C.1, J.2]. This might lead to the activation of new constraints which might be removed later, hence making the factor update more computationally expensive. In [C.7] the projection of the gradient with fixed step size $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$ has been executed with lower computational complexity. But the issue of newly activated constraints persist if some inaccuracy is present in the computation of the gradient at $\boldsymbol{z}^{k+1/2}$ so that $g_i\left(\boldsymbol{z}^{k+1/2}\right) \neq 0$ for some $i \in \mathcal{I}\backslash\mathcal{A}^k$ contrary to results of Lemma 4.5.1 especially when it is being updated.

In the proposed algorithm we use the projection of the chopped gradient instead as

$$\boldsymbol{z}^{k+1} = \mathcal{P}_\Omega\left(\boldsymbol{z}^{k+1/2} - \alpha\boldsymbol{\beta}\left(\boldsymbol{z}^{k+1/2}\right)\right), \tag{5.2}$$

with fixed step-length $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$. This only releases the not optimal[2] constraint indices

[1] The KKT conditions can be written as $\|\boldsymbol{\nu}\left(\boldsymbol{z}^{k+1/2}\right)\| = 0$, $\boldsymbol{\nu}\left(\boldsymbol{z}^{k+1/2}\right) = \boldsymbol{\varphi}\left(\boldsymbol{z}^{k+1/2}\right) + \boldsymbol{\beta}\left(\boldsymbol{z}^{k+1/2}\right)$

[2] In terms of sign of the associated Lagrange multipliers.

from the active set without activation of new constraints[3]. This has been inspired by the *proportioning step* commonly used in the large scale optimization as an effective tool for releasing of the blocking constraints, see, e.g., [28].

5.2 Convergence

The convergence of Algorithm 2 is based on the simple estimate of the cost function q decrease at the proportioning step when the algorithm moves to the face problem minimizer. The algorithm performs either the projection of the chopped gradient from the solution of face problem in the proportioning step or the projection of the Newton direction utilizing the projected-Newton-direction-path via PLS. Hence it is shown firstly that in both steps the cost function q is decreased leading to the decrease at each iteration. Then based on those results, the final theorem showing the convergence of the proposed algorithm is stated and proved.

Each iteration of CGNP algorithm uses the Newton direction $\boldsymbol{p}$. Lemma 4.3.2 showed that nonzero $\boldsymbol{p}$ is descent direction for any active set. The following lemma shows that there is a positive decrease in the cost function q when Newton direction $\boldsymbol{p}$ is applied with limited step-size.

Lemma 5.2.1. Let $\boldsymbol{z}^k \in \Omega$ and

$$\Delta t_1^* = -\frac{\boldsymbol{g}(\boldsymbol{z}^k)^T \boldsymbol{p}^k}{\|\boldsymbol{H}\| \|\boldsymbol{p}^k\|^2}, \qquad \Delta t \in (0, \Delta t_1^*],$$

and consider nonzero direction $\boldsymbol{p}^k$ to be descent. Then there exists $\zeta > 0$ such that

$$q\left(\boldsymbol{z}^k\right) - q\left(\boldsymbol{z}^k + \Delta t \boldsymbol{p}^k\right) \geq \zeta.$$

Proof. Let $\Delta t > 0$ and using $(\boldsymbol{p}^k)^T \boldsymbol{H} \boldsymbol{p}^k \leq \|\boldsymbol{H}\| \|\boldsymbol{p}^k\|^2$ we get

$$\begin{aligned} q\left(\boldsymbol{z}^k\right) - q\left(\boldsymbol{z}^k + \Delta t \boldsymbol{p}^k\right) &= -\Delta t \boldsymbol{g}(\boldsymbol{z}^k)^T \boldsymbol{p}^k - (1/2)\Delta t^2 (\boldsymbol{p}^k)^T \boldsymbol{H} \boldsymbol{p}^k \\ &\geq -\Delta t \boldsymbol{g}(\boldsymbol{z}^k)^T \boldsymbol{p}^k - (1/2)\Delta t^2 \|\boldsymbol{H}\| \|\boldsymbol{p}^k\|^2 = \Delta q\left(\boldsymbol{z}^k, \boldsymbol{p}^k, \Delta t\right). \end{aligned}$$

Analyzing the derivative of right hand side of inequality w.r.t. Δt

$$\frac{d\, \Delta q\left(\boldsymbol{z}^k, \boldsymbol{p}^k, \Delta t\right)}{d\, \Delta t} = -\boldsymbol{g}(\boldsymbol{z}^k)^T \boldsymbol{p}^k - \Delta t \|\boldsymbol{H}\| \|\boldsymbol{p}^k\|^2,$$

and from the assumption of descent direction $\boldsymbol{p}^k$, i.e., $\boldsymbol{g}(\boldsymbol{z}^k)^T \boldsymbol{p}^k < 0$ one can conclude that $\Delta q\left(\boldsymbol{z}^k, \boldsymbol{p}^k, \Delta t\right)$ is positive and increasing function on interval $\Delta t = (0, \Delta t_1^*]$ with maximum at

$$\Delta t_1^* = -\frac{\boldsymbol{g}(\boldsymbol{z}^k)^T \boldsymbol{p}^k}{\|\boldsymbol{H}\| \|\boldsymbol{p}^k\|^2}, \tag{5.3}$$

[3] If opposite bound is not activated by the projection of the chopped gradient, which rarely happens in MPC applications.

proving the Lemma with

$$\zeta = -\Delta t \boldsymbol{g}(\boldsymbol{z}^k)^T \boldsymbol{p}^k - (1/2)\Delta t^2 \|\boldsymbol{H}\| \|\boldsymbol{p}^k\|^2 > 0,$$

for $\Delta t = (0, \Delta t_1^*]$. □

Now let show that there is decrease in q at each iteration of Algorithm 2 when proportioning step is executed.

Lemma 5.2.2. Let

$$\boldsymbol{d} = \overline{\boldsymbol{z}} - \underline{\boldsymbol{z}}, \quad \boldsymbol{z} \in \Omega, \quad \boldsymbol{\beta} = \boldsymbol{\beta}(\boldsymbol{z}), \quad \alpha \in (0, 2\|\boldsymbol{H}\|^{-1}),$$

and let $\widetilde{\boldsymbol{\beta}}$ be defined by its components

$$\widetilde{\beta}_i = \begin{cases} \min\{\beta_i, d_i/\alpha\} & \text{for } z_i = \overline{z}_i \\ -\min\{-\beta_i, d_i/\alpha\} & \text{for } z_i = \underline{z}_i \\ 0 & \text{elsewhere.} \end{cases} \tag{5.4}$$

Then there exists $\mu > 0$ independent of $\boldsymbol{z}$, such that

$$q(\boldsymbol{z}) - q\left(P_\Omega(\boldsymbol{z} - \alpha\boldsymbol{\beta})\right) \geq \mu \|\widetilde{\boldsymbol{\beta}}\|^2. \tag{5.5}$$

Proof. Let $\alpha = \delta\|\boldsymbol{H}\|^{-1}$, $\delta \in (0, 2)$. Let us see that $\widetilde{\boldsymbol{\beta}}^T\boldsymbol{g} = \widetilde{\boldsymbol{\beta}}^T\boldsymbol{\beta}$. Since by definition (4.2) we get $\beta_i = g_i$ or $\beta_i = 0$. The first case is trivial. If $\beta_i = 0$ it is enough to see that $\widetilde{\beta}_i = 0$ from (5.4). Using the Taylor formula and

$$\alpha\widetilde{\boldsymbol{\beta}}^T\boldsymbol{H}\widetilde{\boldsymbol{\beta}} = \delta\|\boldsymbol{H}\|^{-1}\widetilde{\boldsymbol{\beta}}^T\boldsymbol{H}\widetilde{\boldsymbol{\beta}} \leq \delta\|\widetilde{\boldsymbol{\beta}}\|^2,$$

we get

$$\begin{aligned} q(\boldsymbol{z}) - q\left(P_\Omega(\boldsymbol{z} - \alpha\boldsymbol{\beta})\right) &= q(\boldsymbol{z}) - q(\boldsymbol{z} - \alpha\widetilde{\boldsymbol{\beta}}) \\ &= \alpha\widetilde{\boldsymbol{\beta}}^T\boldsymbol{g} - \frac{\alpha^2}{2}\widetilde{\boldsymbol{\beta}}^T\boldsymbol{H}\widetilde{\boldsymbol{\beta}} = \alpha\widetilde{\boldsymbol{\beta}}^T\boldsymbol{\beta} - \frac{\alpha^2}{2}\widetilde{\boldsymbol{\beta}}^T\boldsymbol{H}\widetilde{\boldsymbol{\beta}} \\ &\geq \alpha\|\widetilde{\boldsymbol{\beta}}\|^2 - \frac{\alpha\delta}{2}\|\widetilde{\boldsymbol{\beta}}\|^2 = \left(\delta - \frac{\delta^2}{2}\right)\|\boldsymbol{H}\|^{-1}\|\widetilde{\boldsymbol{\beta}}\|^2 \\ &= \mu\|\widetilde{\boldsymbol{\beta}}\|^2. \end{aligned}$$

This proves (5.5) with $\mu = \left(\delta - \frac{\delta^2}{2}\right)\|\boldsymbol{H}\|^{-1}$. □

Lemma 5.2.3. Let $\boldsymbol{z}^k, \boldsymbol{z}^{k+1} \in \Omega$ are produced by the CGNP as Algorithm 2. Then there exists $\gamma > 0$ such that

$$q\left(\boldsymbol{z}^k\right) - q\left(\boldsymbol{z}^{k+1}\right) \geq \gamma.$$

Proof. There are two main branches in Algorithm 2. In the first one, the Newton direction $\boldsymbol{p}^k$ which points to the minimizer of face problem (4.3) is projected utilizing the projected-Newton-direction path via PLS algorithm. In the second branch, the Newton direction $\boldsymbol{p}^k$ is applied up to the minimizer of the face problem and the projection of the chopped gradient is executed. The decision of which branch will be executed is based on the feasibility of $\boldsymbol{z}^k + \boldsymbol{p}^k$. Hence to show the decrease in the cost function q, the decrease has to be in both branches.

The decrease of the cost function q in the first branch is based on the successive application of Lemma 5.2.1 for each explored line segment of PLS algorithm. Observe that the Newton direction $\boldsymbol{p}^k$ generated via solution of (4.13) is always descent direction as indicated in Lemma 4.3.2. This is however not true in general for the projected direction $\boldsymbol{d}$ as defined in (4.18) at each explored line segment in PLS algorithm. The exception is the first line segment where it is true since $\boldsymbol{d}^0 = \boldsymbol{p}^k$. Moreover, as the growth of function $q(\boldsymbol{z})$ is checked in PLS as stopping condition of exploration, the PLS always performs at least one line segment exploration with cost function q decrease.

Hence assume that $\boldsymbol{z}^{k+1}$ is generated by the Newton-projected-direction path via PLS algorithm and let $\Delta t^* > 0$ be defined by (5.3) and the difference of two successive ordered breakpoints in PLS be $\delta t_j = t_j - t_{j-1} > 0$. Then there exist $\gamma_{\mathrm{PLS}} > 0$ such that

$$\begin{aligned} q\left(\boldsymbol{z}^k\right) - q\left(\boldsymbol{z}^{k+1}\right) &\geq \sum_{j=1}^{j=w-1} \{-\delta t_j \boldsymbol{g}(\boldsymbol{z}^k(t_{j-1}))^T \boldsymbol{d}^{j-1} - (1/2)\delta t_j^2 \|\boldsymbol{H}\| \|\boldsymbol{d}^{j-1}\|^2\} + \\ &+ (-\Delta t^*_{w-1} \boldsymbol{g}(\boldsymbol{z}^k(t_j))^T \boldsymbol{d}^{w-1} - (1/2)(\Delta t^*_{w-1})^2 \|\boldsymbol{H}\| \|\boldsymbol{d}^{w-1}\|^2) = \\ &= \gamma_{\mathrm{PLS}}, \end{aligned} \tag{5.6}$$

with w denoting the number of successively explored line segments by PLS algorithm where the cost function decrease has been detected. As the terms in the sum are all positive based on the Lemma 5.2.1 the right-hand side of inequality is also positive, hence $\gamma_{\mathrm{PLS}} > 0$.

On the other hand, when $\boldsymbol{z}^k + \boldsymbol{p}^k \in \Omega$ the iterate moves to the face minimizer $\boldsymbol{z}^{k+1/2} = \boldsymbol{z}^k + \boldsymbol{p}^k$ where based on Lemma 5.2.1 there exist $\Delta t^* > 0$ and $\gamma_{\mathrm{NS}} > 0$ such that

$$\begin{aligned} q\left(\boldsymbol{z}^k\right) - q\left(\boldsymbol{z}^{k+1/2}\right) &\geq -\Delta t^* \boldsymbol{g}(\boldsymbol{z}^k)^T \boldsymbol{p}^k - (1/2)(\Delta t^*)^2 \|\boldsymbol{H}\| \|\boldsymbol{p}^k\|^2 \\ &= \frac{1}{2} \frac{(\boldsymbol{g}(\boldsymbol{z}^k)^T \boldsymbol{p}^k)^2}{\|\boldsymbol{H}\| \|\boldsymbol{p}^k\|^2} \\ &= \gamma_{\mathrm{NS}} > 0. \end{aligned}$$

If the iteration $\boldsymbol{z}^{k+1/2}$ is not KKT optimal, the proportioning test is executed. Based on Lemma 5.2.2 there exist $\gamma_{\mathrm{P}} > 0$ such that

$$q\left(\boldsymbol{z}^{k+1/2}\right) - q\left(\boldsymbol{z}^{k+1}\right) \geq \gamma_{\mathrm{P}} > 0.$$

Hence, connecting all results together there exist $\gamma > 0$ such that

$$q\left(\boldsymbol{z}^k\right) - q\left(\boldsymbol{z}^{k+1}\right) \geq \gamma > 0,$$

with $\gamma = \min\{\gamma_{\mathrm{PLS}}, (\gamma_{\mathrm{NS}} + \gamma_{\mathrm{P}})\}$. □

Now we are ready to prove the convergence of Algorithm 2.

Theorem 5.2.4 (CGNP Convergence)**.** Let $\{\boldsymbol{z}^k\}$ denote the iterates generated by the CGNP algorithm for the solution of (1.1) with $\boldsymbol{z}^0 \in \Omega$ and $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$. Then, $\{\boldsymbol{z}^k\}$ converges to the solution $\boldsymbol{z}^*$ of (1.1).

Proof. First notice that $\{q\left(\boldsymbol{z}^k\right)\}$ is based on Lemma 5.2.3 decreasing and bounded from below by the unconstrained minimum of q. Moreover, if $\{\boldsymbol{z}^k\}$ is infinite, then it has an infinite subsequence of iterations where proportioning is called – there can be at most n consecutive expansion steps since the active set is expanding at each step by at least one index. Since Ω is compact, it follows that there is a set of indices $\mathcal{K}$ of the proportioning steps such that $\{\boldsymbol{z}^k\}_{k\in\mathcal{K}}$ converges to $\hat{\boldsymbol{z}}$ and

$$q(\hat{\boldsymbol{z}}) = \inf \left\{q\left(\boldsymbol{z}^k\right)\right\}_{k\in\mathcal{K}}.$$

Using Lemma 5.2.2, we get that there is $\mu > 0$ such that

$$q(\boldsymbol{z}^0) - q(\hat{\boldsymbol{z}}) \geq \sum_{k\in\mathcal{K}} \mu \|\widetilde{\boldsymbol{\beta}}(\boldsymbol{z}^k)\|^2,$$

so $\|\widetilde{\boldsymbol{\beta}}(\boldsymbol{z}^k)\|$ converges to zero. However, after inspecting the definition of $\widetilde{\boldsymbol{\beta}}$, it follows that also $\|\boldsymbol{\beta}(\boldsymbol{z}^k)\|$ converges to zero. Since the proportioning step is executed at the face minimizer where based on the Lemma 4.5.1 $\boldsymbol{\varphi}(\boldsymbol{z}^k) = 0$ we conclude that also $\boldsymbol{\nu}(\boldsymbol{z}^k)$ converges to zero. Thus $\boldsymbol{\nu}(\hat{\boldsymbol{z}}) = \boldsymbol{0}$ and, since the solution is unique, also $\hat{\boldsymbol{z}} = \boldsymbol{z}^*$ and $\{\boldsymbol{z}^k\}$ converges to $\boldsymbol{z}^*$. □

5.3 Step Length Selection

The CGNP solver can be used for QP problem (1.1) with fixed problem Hessian. Here it will be shown how it can be used for QP problems arising in MPC where problem Hessian might change at each sample time.

The convergence of the CGNP algorithm is based on the estimate of the cost function decrease in the proportioning step for selection of step-length $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$. Hence to ensure the convergence, the step-length must be bounded above by inverse of the Hessian norm. When the QP problem Hessian is fixed the Hessian norm can be computed off-line, e.g., by computing the maximal eigenvalue of $\boldsymbol{H}$ for the expected limited range of optimization variables.

On the other hand, when the QP problem Hessian is changing at each sample time, it is needed to ensure that the selected step-length is still convergent and if possible to eliminate need of computing the Hessian norm exactly. As the inverse of the Hessian norm creates the upper bound of the convergent step-length, any upper bound of the Hessian norm leads to convergent step-length. Hence e.g., in case of nonlinear MPC where the QP problem Hessian is subject to change due to different dynamics, the Frobenius norm can be used since it upper bounds the matrix spectral norm [43]. When only controller tuning might change in the linear

MPC application, it is possible to use the range of the tuning parameters to set the upper bound of the QP problem Hessian off-line using the Cauchy inequality as[4]

$$\|\boldsymbol{H}_t\| = \|\mathsf{B}^T\mathsf{Q}_t\mathsf{B} + \mathsf{R}_t\| \leq \|\mathsf{B}^T\mathsf{Q}_t\mathsf{B}\| + \|\mathsf{R}_t\| \leq \|\mathsf{B}^T\|\|\mathsf{Q}_t\|\|\mathsf{B}\| + \|\mathsf{R}_t\|$$

where B, Q_t and R_t are fixed prediction matrix, and tuning matrices at time t defined in Section 1.2. Note that it is assumed that the entries of Q_t and R_t might change in a given range, allowing the setting of their upper bound of norm and hence the upper bound of the QP problem.

5.4 Algorithm Properties

The blocking constraints are removed from the active set only when the face minimizer is feasible, i.e., $\boldsymbol{z}^k + \boldsymbol{p}^k \in \Omega$ in the CGNP algorithm. This is very similar to the ASMs where the constraint index with the largest negative Lagrange multiplier is removed from the active set when the iteration is the minimizer of the face problem, see, e.g., [65, Section 16.5].

But contrary to the ASMs, all the not optimal constraint indices are removed from the active set by the proportioning step, leading to more rapid change in the active set in CGNP algorithm. On the other hand, the situation that $\boldsymbol{z}^k + \boldsymbol{p}^k \in \Omega$ is valid only in rare iterations, hence the algorithm waits for this condition to release the blocking constraints to allow a more rapid change in the cost function q. As a result, the algorithm mostly executes the expansion step and at the later stage it removes the blocking constraints by the proportioning step.

Note that this property is good for reduction of unwanted changes in the active set, hence limiting the complexity of the factor update scheme used for solution of the face problem. On the other hand, the CGNP algorithm does not take into account the absolute value of the Lagrange multipliers of the blocking constraints in the active set. Hence the algorithm might have slower decrease of q at the beginning iterations while having more rapid decrease of q when blocking constraints are removed from the active set at the later iterations of the algorithm.

The benefit of CGNP is that the gradient can be updated over the course of the iterations, except when the proportioning step is executed where it must be recomputed from scratch.

5.4.1 Sensitivity to Step Size

In this section we show how the iterations generated by the CGNP algorithm are dependent on the choice of the step-size of the proportioning step $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$ with $\varepsilon = 1e{-}6\|\boldsymbol{h}\|$. Typical convergence of CGNP algorithm is depicted for the numerical example of a randomly generated QP problem with $n = 100$, $\operatorname{cond}\boldsymbol{H} = 1.17e5$. The Hessian and linear part of the QP problem have been generated in the MATLAB environment using `randn` command while constraints were constructed randomly around the unconstrained minimizer by `rand` command. In the following two figures, the square, circle, cross and diamond markers indicate that the iteration has been the result of initialization, expansion step, proportioning step and full application of Newton direction, respectively.

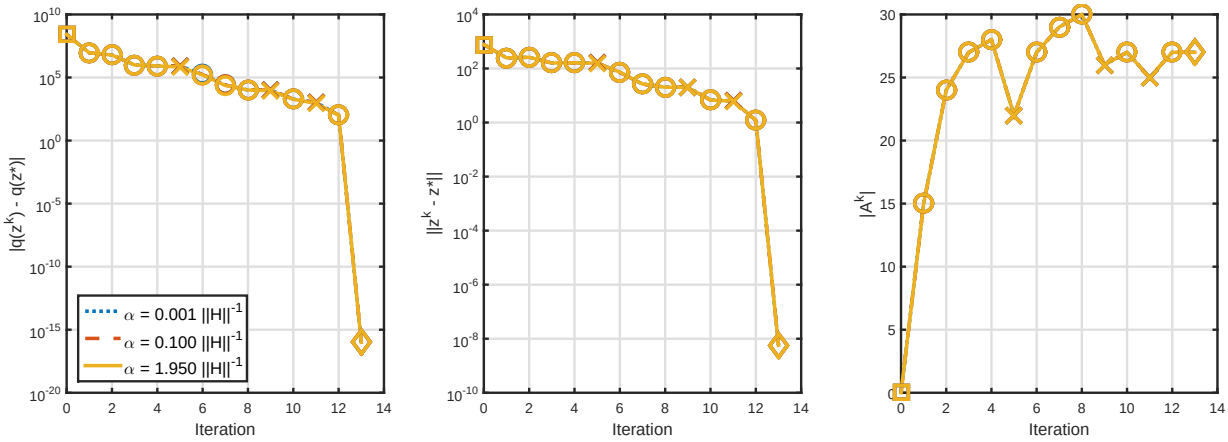


Figure 5.1: Comparison of convergence of CGNP algorithm for several step-sizes α of proportioning step for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}}+\underline{\boldsymbol{z}})/2$. Algorithm in this case generates very similar iterations, with the same active set.

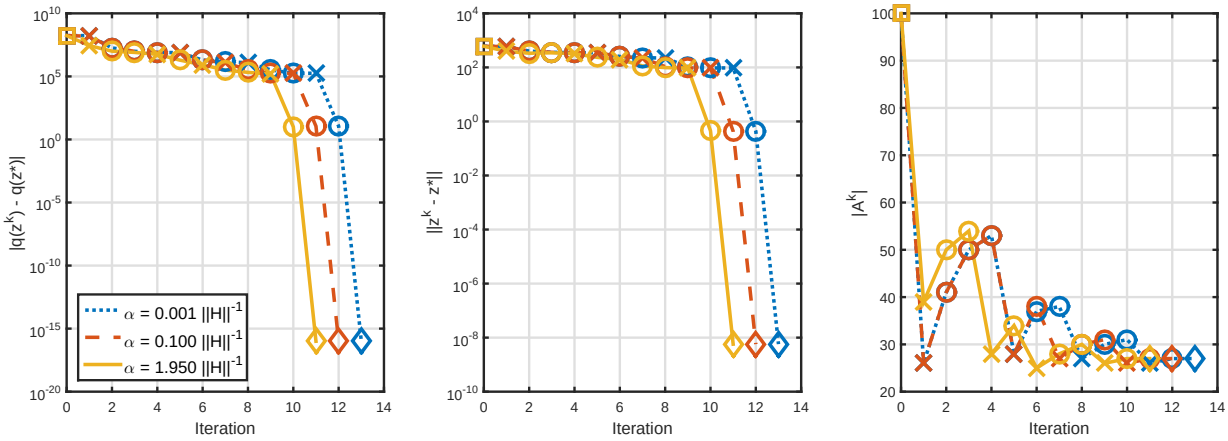


Figure 5.2: Comparison of convergence of CGNP algorithm for several step-sizes α of proportioning step for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$.

Two scenarios have been tested by varying the initial iterate. The purpose of the variance is to show how the algorithm might behave when the active set has to be mostly expanded or reduced. In the first setup, the algorithm has been started from the center of the feasible set with $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$, hence $\mathcal{A}^0 = \emptyset$. The typical convergence of the CGNP algorithm for a different choice of α is depicted in Figure 5.1 where $|q(\boldsymbol{z}^k) - q(\boldsymbol{z}^*)|$, $\|\boldsymbol{z}^k - \boldsymbol{z}^*\|$ and the cardinality of active set $|\mathcal{A}^k|$ are shown as a function of algorithm iteration index k. It is evident that the algorithm starts generating the iterations via expansion steps (circles) and then in about the 5-th, 9-th and 11-th iterations the proportioning step is executed (cross) as indicated in the drop of the number of constraint indices in the active set. Note that the iterations are almost the same, even for very different α which differs in power of four, showing that algorithm is very

[4]Proof can be found in [87].

insensitive to this parameter for this setup. Note that the full step application of the Newton direction (diamond) is always executed at the final iteration.

In the second setup, the initial iteration has been set to upper bound, i.e., $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$, i.e., $\mathcal{A}^0 = \mathcal{I}$. Hence all upper constraints have been active at the initial iteration, and the CGNP algorithm has to start with the proportioning step to remove the blocking constraint indices from the active set. In this example, the algorithm is sensitive to the choice of α as illustrated in Figure 5.2. The reason is that the application of proportioning with larger step-size leads to the activation of the opposite bound (lower bound) which is not happening for a very short choice of α. This is evident e.g., from the first iteration where the active set cardinality for choice of $\alpha = 1.95\|\boldsymbol{H}\|^{-1}$ drops to 39 while for a shorter choice it is 26 although the initial iterate has been the same. As a result, the algorithm involves more iterations for a shorter choice of α, but our experiments suggest that the difference is often in order of $1-5$ additional iterations.

5.4.2 Comparison to GPQP Algorithm

To compare the CGNP algorithm to the existing methods, the GPQP method of [3] has been implemented in MATLAB® environment. The GPQP method performs the gradient projection via PLS followed by the solution of face problem defined by the current active set. When the face minimizer is not feasible it is projected using the projected Newton direction path as in the CGNP algorithm via PLS. Compared to the original algorithm reported in [3] we apply the GPQP algorithm directly to the primal dense formulation problem and the Cholesky factorization with updates introduced in Section 4.4 is used for solution of face problem. Furthermore, we have implemented a gradient update for projection of gradient and Newton direction along the projected path in PLS. The same stopping condition has been implemented in both algorithms. Additionally, to indicate the effectiveness of the proportioning step in CGNP, we implemented a modified algorithm denoted as CGNP PLS which uses PLS routine to project the gradient instead of the proportioning step (5.2) together with an update of the gradient.

The comparison of convergence of both algorithms for the randomly generated QP problem introduced in Section 5.4.1 for initial iterate $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$ is depicted in Figure 5.3. Firstly, observe that the GPQP algorithm converges in less number of iterations (6 compared to 13 for CGNP). This is because the proportioning (crosses) in the CGNP is executed late in the iteration process while the projection of the gradient performed at each iteration of GPQP enables the drop of the not-optimal constraint indices from the active set in a faster way. On the other hand, the cost of PLS in the gradient projection step together with need of recomputation of the gradient might lead to the significant increase of cost of each GPQP iteration compared to CGNP. From our simulation, it seems that the update of the gradient along the PLS introduced in Section 4.6 is an important ingredient to reduce the cost of projection in GPQP. When it is used, the GPQP involves 26.8% less number of flops, while when it is not used, the GPQP involves 4.5% less flops compared to the CGNP algorithm in our setup while involving far fewer iterations to converge.

The impact of the computation cost of PLS used in the gradient projection in GPQP is even more significant when the initial iterate is chosen as $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$ as shown in Figure 5.4. Both

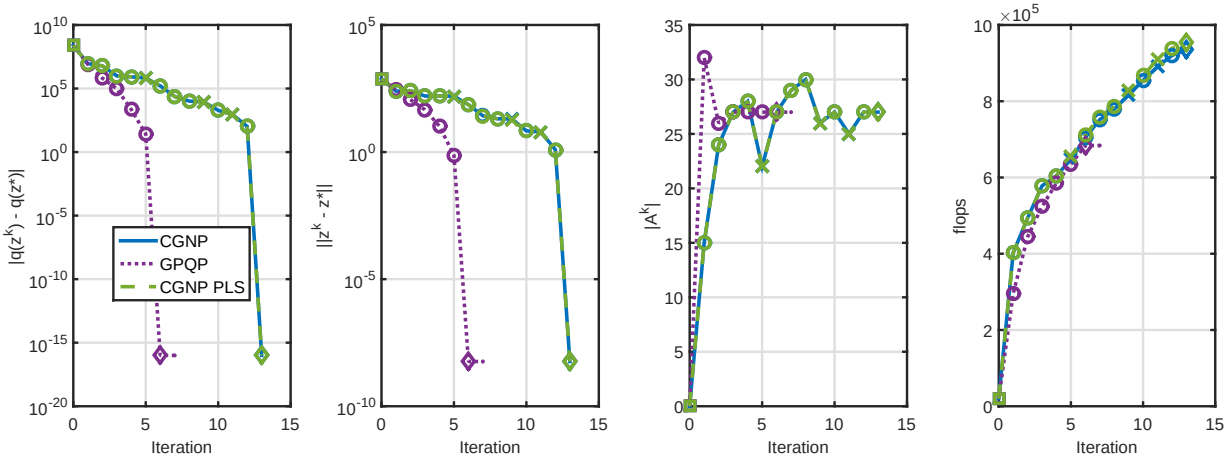


Figure 5.3: Comparison of convergence of CGNP and GPQP algorithms for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$.

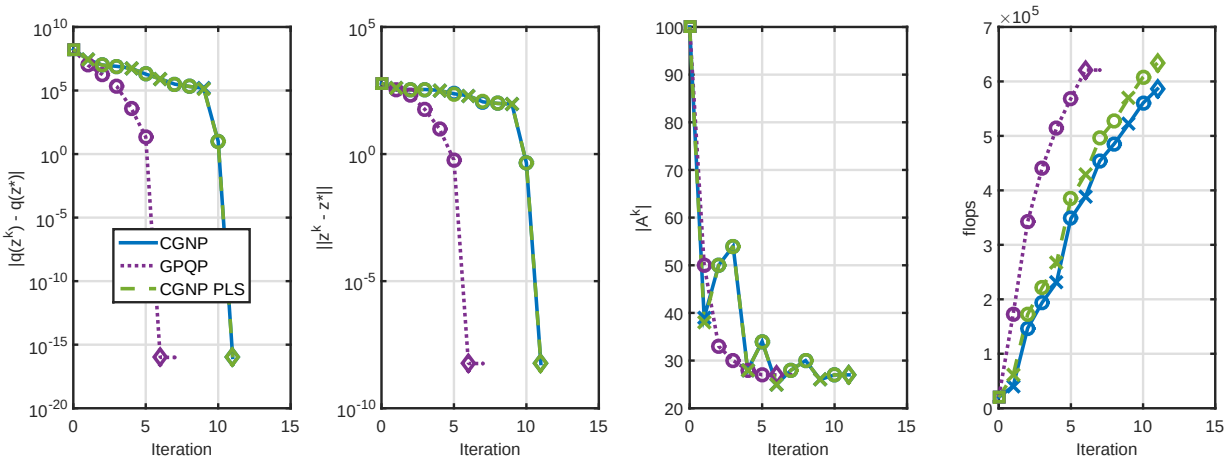


Figure 5.4: Comparison of convergence of CGNP and GPQP algorithms for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$.

algorithms have to perform drop of constraint indices from the active set. The GPQP converges in 6-th iteration compared to 11 iterations needed for CGNP.

The cost of the gradient projection via PLS at each iteration of GPQP leads to the fact that the algorithm involves 5.9% more flops compared to the CGNP algorithm. Furthermore, when the updates of the gradient are not used in the PLS in GPQP, the algorithm needs 41.5% more flops than compared to CGNP.

The iterations generated by the CGNP and CGNP PLS algorithms are very similar for both test cases although each proportioning of the CGNP PLS algorithm is more expensive due to additional cost of PLS routine compared to the proportioning step (5.2). It leads to the fact that the CGNP PLS algorithm involves 2.0% and 8.1% more flops compared to the CGNP algorithm for test cases respectively. Our further experiments suggest that it is true in general

that projection of gradient via PLS instead of the proportioning step leads to increase of flops in a range of 2-10% compared to the CGNP algorithm.

5.5 Summary

The algorithm combining the projection of chopped gradient and Newton direction has been presented as CGNP method. It computes at each iteration the solution of the face problem defined by current active set by direct solver with Cholesky factor update. If the solution of the face problem is not feasible, the algorithm uses projected-Newton-direction path utilizing PLS routine to expand the active set. On the other hand, when the solution of the face problem is feasible with respect to QP problem constraints, it employs the proportioning step via projection of chopped gradient with fixed step-size to free all blocking constraint indices from the active set. Another option would be to employ PLS routine along the gradient as in GPQP of [3], which would potentially release only a subset of blocking constraints. On the other hand, the drawback would be the additional computational cost of PLS as was previously indicated in our experiments and the previous section.

It was observed that CGNP algorithm is not sensitive to the choice of step-size of proportioning step on randomly generated numerical experiment when opposite bounds are not activated during the course of the algorithm. On the other hand, the large but still convergent step size of proportioning leads to better algorithm performance when opposite bounds are activated during the course of algorithm iterations. This is because it helps with the active set expansion with low computation cost.

The benefit of the proposed method compared to the first order methods is its insensitivity to QP problem conditioning since it employs the solution of the face problem. Compared to the ASMs, the CGNP algorithm enables quicker identification of the optimal active set by expansion/reduction of the active set by the projection. The main disadvantage of the CGNP algorithm is its inability to take into account the absolute value of Lagrange multipliers indicating the not-optimality of blocking constraint indices in the active set. The algorithm instead waits for the condition that the face problem minimizer is feasible with respect to the QP constraints. As a result, the decrease of the cost function q due to the removal of the blocking constraints from the active set can occur late in the algorithm iterations.

Chapter 6

Proportioning with Newton Directions

The main drawback of the CGNP algorithm presented in Chapter 5 is that the absolute values of Lagrange multipliers are not taken into account for the decision whether the active set should be expanded or reduced. As a result, the algorithm waits for the condition that $\boldsymbol{z}^k + \boldsymbol{p}^k \in \Omega$ to release the not-optimal constraint indices from the active set, to allow further decrease of the cost function q.

In this chapter, the algorithm which combines the active set strategy with the proportioning test is introduced. The proportioning test is used to decide when to leave the actual active set. For the minimization in the face, the proposed algorithm uses the Newton directions implemented by the Cholesky factors updates.

A similar strategy to the proposed algorithm has been independently proposed in [14]. Compared to the algorithm presented therein we use rather a free gradient in the proportioning test instead of a projected gradient as suggested in [26]. Moreover, the proportioning step is executed with fixed step-size instead of non-monotone line search. When solution of the face problem is not feasible, we employ the projected Newton direction path using the PLS algorithm which enables the effective update of the gradient when compared to the non-monotone line search in the algorithm of [14].

Firstly, the specific ingredients of the proposed method are described in detail together with the algorithm referring to the basic ingredients described in Chapter 4. Then the convergence of the algorithm is proved and the main properties of the algorithm are analyzed on the illustrative QP problem together with the comparison to the CGNP algorithm.

6.1 Algorithm

The proposed algorithm tries in each iteration to reduce effectively the part of the gradient which dominates the error in the KKT conditions. If it is the free gradient, i.e., if the iterate is proportional, then the algorithm uses the Newton direction computed by direct solver employing the factor update from the previous iteration to solve the face problem as described in Section 4.4. When the solution of the face problem (4.3) lies outside the feasible set Ω, the working set is expanded using the projected-Newton-direction path utilizing PLS algorithm

Algorithm 3 Proportioning with Newton Directions (PND)) algorithm. Given a SPD matrix $\boldsymbol{H}$ of the order n, n-vectors $\boldsymbol{h}$, $\underline{\boldsymbol{z}}$, $\overline{\boldsymbol{z}}$, $\Omega = \{\boldsymbol{z} : \underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}}, \underline{\boldsymbol{z}} < \overline{\boldsymbol{z}}\}$, $\boldsymbol{z}^0 \in \Omega$, $\Gamma > 0$, $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$ and $\varepsilon > 0$.

{Initialization}
Set $k = 0$, $\boldsymbol{g}\left(\boldsymbol{z}^k\right) = \boldsymbol{H}\boldsymbol{z}^k + \boldsymbol{h}$
while $\|\boldsymbol{\nu}\left(\boldsymbol{z}^k\right)\| \geq \varepsilon$ **do**
  if $\|\boldsymbol{\beta}(\boldsymbol{z}^k)\| \leq \Gamma\|\boldsymbol{\varphi}(\boldsymbol{z}^k)\|$ **then**
    {Proportional $\boldsymbol{z}^k$}
    Solve (4.13) to obtain Newton direction $\boldsymbol{p}^k$.
    Compute $\boldsymbol{H}\boldsymbol{p}^k$ as described in Section 4.5
    $\alpha_f = \max\{\alpha : \boldsymbol{z}^k + \alpha\boldsymbol{p}^k \in \Omega\}$
    if $\alpha_f < 1$ **then**
      {Expansion step}
      $\left[\boldsymbol{z}^{k+1}, \boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right)\right] = \mathrm{PLS}\left(\boldsymbol{z}^k, \boldsymbol{p}^k, \boldsymbol{H}\boldsymbol{p}^k, \boldsymbol{g}\left(\boldsymbol{z}^k\right)\right)$
    else
      {Full Newton direction can be applied to remain feasible}
      $\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \boldsymbol{p}^k$
      $\boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{g}\left(\boldsymbol{z}^k\right) + \boldsymbol{H}\boldsymbol{p}^k$
    end if
  else
    {Proportioning step}
    $\boldsymbol{z}^{k+1} = \mathcal{P}_\Omega\left(\boldsymbol{z}^k - \alpha\boldsymbol{\beta}\left(\boldsymbol{z}^k\right)\right)$
    $\boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{H}\boldsymbol{z}^{k+1} + \boldsymbol{h}$
  end if
  $k = k + 1$
end while
$\boldsymbol{z}^* = \boldsymbol{z}_k$

(Section 4.6). In case that iteration is not proportional, the algorithm uses projection of the chopped gradient with fixed step-size to remove blocking constraints.

The Proportioning with Newton Directions (PND) algorithm presented as Algorithm 3 uses the proportioning test to limit the number of unwanted changes of the active set hence reduces the computational complexity of the factor update scheme used for the face problem solution. The overview of computational complexity of all algorithm steps is presented in Table 6.1 concluding that overall computational complexity of each the algorithm iteration is $\mathcal{O}(n^2)$.

6.1.1 Proportioning Test

The presented algorithm starts with testing of which part of the gradient should be reduced, i.e., free gradient $\boldsymbol{\varphi}(\boldsymbol{z}^k)$ or the chopped gradient $\boldsymbol{\beta}(\boldsymbol{z}^k)$. The decision is based on the proportioning

Table 6.1: Computational complexity of the PND algorithm steps.

Algorithm Step	Complexity (flops)	Notation	Described
$\boldsymbol{g}\left(\boldsymbol{z}^k\right) = \boldsymbol{H}\boldsymbol{z}^k + \boldsymbol{h}$	$2n^2$		
$\|\boldsymbol{\beta}(\boldsymbol{z}^k)\| \leq \Gamma\|\boldsymbol{\varphi}(\boldsymbol{z}^k)\|$	$2n$		Section 6.1.1
$\boldsymbol{R}^{k-1} \rightarrow \boldsymbol{R}^k$	$\mathcal{O}\left(\lvert\mathcal{A}^k_{\text{add}}\rvert(n - m^{k-1} - \lvert\mathcal{A}^k_{\text{add}}\rvert)^2\right) + \mathcal{O}\left(\lvert\mathcal{A}^k_{\text{rem}}\rvert^3\right) + \mathcal{O}\left(\lvert\mathcal{A}^k_{\text{rem}}\rvert^2 n_f\right) + \mathcal{O}\left(\lvert\mathcal{A}^k_{\text{rem}}\rvert n_f^2\right)$	$m^{k-1} = \mathcal{A}^{k-1}$ $n_f = n - m^{k-1} - \lvert\mathcal{A}^k_{\text{add}}\rvert$	Section 4.4
$(\boldsymbol{R}^k)^T\boldsymbol{y} = -\boldsymbol{r}\left(\boldsymbol{z}^k\right)$ $\boldsymbol{R}^k\boldsymbol{p}^k = \boldsymbol{y}$	$2(n - m^k)^2$	$m^k = \lvert\mathcal{A}^k\rvert$	Section 4.3
$\boldsymbol{H}\boldsymbol{p}^k$	$2m^k(n - m^k)$		Section 4.5
$\alpha_f = \max\{\alpha : \boldsymbol{z}^k + \alpha\boldsymbol{p}^k \in \Omega\}$	$2n$		Section 5.1.1
$\left[\boldsymbol{z}^{k+1}, \boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right)\right] =$ PLS $\left(\boldsymbol{z}^k, \boldsymbol{p}^k, \boldsymbol{H}\boldsymbol{p}^k, \boldsymbol{g}\left(\boldsymbol{z}^k\right)\right)$	$2(n - m^k) + \sum_{i=1}^{s}(2nr_i + 10n)$	r_i constraints changed on the i-th line segment	Section 4.6
$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \boldsymbol{p}^k$	n		
$\boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{g}\left(\boldsymbol{z}^k\right) + (\boldsymbol{H}\boldsymbol{p}^k)$	n		
$\boldsymbol{z}^{k+1} = \mathcal{P}_\Omega\left(\boldsymbol{z}^k - \alpha\boldsymbol{\beta}\left(\boldsymbol{z}^k\right)\right)$	$4n$		Section 6.1.3
$\boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{H}\boldsymbol{z}^{k+1} + \boldsymbol{h}$	$2n^2$		

test. Given $\Gamma > 0$, we call the iteration $\boldsymbol{z}^k$ *proportional* if the inequality

$$\|\boldsymbol{\beta}(\boldsymbol{z}^k)\| \leq \Gamma\|\boldsymbol{\varphi}(\boldsymbol{z}^k)\|, \tag{6.1}$$

holds [28]. This test tries to limit both the unnecessary expanding of the active set and the premature releasing of the constraints from the set $\mathcal{A}^k$ by balancing the violation of the KKT conditions on the active and the free set.

6.1.2 Proportional Iteration

If the proportioning test (6.1) holds, the iteration is proportional and the algorithm should reduce the violation of the KKT conditions in the free gradient $\boldsymbol{\varphi}(\boldsymbol{z}^k)$. To eliminate the free gradient, the Newton direction $\boldsymbol{p}^k$ which points towards the minimizer of face problem defined by the current active set $\mathcal{A}^k$ is computed by solution of (4.13). The update of the Cholesky factor of the reduced Hessian from the previous iteration is used as described in details in Section 4.4.

When the Newton direction $\boldsymbol{p}^k$ is applied it may result in one of two scenarios: (i) some new constraints are activated, i.e., $\boldsymbol{z}^k + \boldsymbol{p}^k \notin \Omega$, hence it can be applied only with limited step size $\mu \in (0, 1)$ such that $\boldsymbol{z}^k + \mu\boldsymbol{p}^k \in \Omega$ (ii) full Newton direction can be applied without violation of any constraints, i.e. $\boldsymbol{z}^k + \boldsymbol{p}^k \in \Omega$.

In order to decide which situation occurs, the maximal step size $\alpha_f \in (0, 1]$ leading to feasible iterate along the direction $\boldsymbol{p}^k$, i.e., $\alpha_f = \min\{1, \max\{\alpha : \boldsymbol{z}^k + \alpha\boldsymbol{p}^k \in \Omega\}\}$ is computed by (5.1).

6.1.2.1 Expansion Step

If $\alpha_f < 1$, i.e., $\boldsymbol{z}^k + \boldsymbol{p}^k \notin \Omega$, there is at least one missing constraint in the active set. Hence, the Newton direction $\boldsymbol{p}^k$ is used in the projected-Newton-direction path evaluated via PLS algorithm. PLS procedure of [65] is used to obtain the first local minimizer along the projected path completing the iteration. Our implementation of PLS when compared to the original algorithm of [65], additionally successively updates the gradient using the pre-computed $\boldsymbol{H}\boldsymbol{p}^k$ along the projected Newton direction path as described in Sections 4.6 and 4.5.

6.1.2.2 Application of Newton Direction

If full Newton direction could be applied without violation of any constraints (i.e., $\alpha_f = 1$) then the iterate is updated together with the gradient by

$$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \boldsymbol{p}^k, \qquad \boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{g}\left(\boldsymbol{z}^k\right) + (\boldsymbol{H}\boldsymbol{p}^k),$$

using the pre-computed $\boldsymbol{H}\boldsymbol{p}^k$ as described in Section 4.5.

6.1.3 Proportioning Step

When the iteration is not proportional, i.e., (6.1) does not hold, we employ the proportioning step

$$\boldsymbol{z}^{k+1} = \mathcal{P}_\Omega\left(\boldsymbol{z}^k - \alpha\boldsymbol{\beta}\left(\boldsymbol{z}^k\right)\right), \tag{6.2}$$

with fixed step-length $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$. Hence all constraint indices indicated as not-optimal by a nonzero component of $\boldsymbol{\beta}\left(\boldsymbol{z}^k\right)$ will be removed from the active set. We project the chopped gradient $\boldsymbol{\beta}\left(\boldsymbol{z}^k\right)$ instead of the full gradient as done in [C.7] to disable the addition of new constraints to the active set by the proportioning step. This limits unwanted changes in the active set. Note that the use of a fixed step-length reduces the computational complexity when compared to the case when feasible minimizer is searched along the chopped gradient as in [14, 28]. Moreover, as the chopped gradient $\boldsymbol{\beta}$ is eliminated[1] by the application of the proportioning step, see definition in (4.2), the consecutive iteration is proportional and the corresponding face problem will be solved by direct solver. After the application of the proportioning step, the gradient cannot be updated as described in Section 4.5 and has to be computed from scratch.

6.2 Convergence

The proof of convergence of the PND algorithm is based on a simple estimate of the decrease of the cost function q in the proportioning step proved in Lemma 5.2.2.

Theorem 6.2.1 (PND Convergence)**.** Let $\{\boldsymbol{z}^k\}$ denote the iterates generated by the PND algorithm for the solution of (1.1) with $\boldsymbol{z}^0 \in \Omega$, $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$, and $\Gamma > 0$. Then, $\{\boldsymbol{z}^k\}$ converges to the solution $\boldsymbol{z}^*$ of (1.1).

[1] If some new constraint has not been activated.

Proof. First notice that $\{q\left(\boldsymbol{z}^k\right)\}$ is decreasing and bounded from below by the unconstrained minimum of q. Moreover, if $\{\boldsymbol{z}^k\}$ is infinite, then it has an infinite subsequence of non-proportional iterates – there can be at most n consecutive expansion steps since the active set is expanding at each step by at least one index. Since Ω is compact, it follows that there is a set of indices $\mathcal{K}$ of the proportioning steps such that $\{\boldsymbol{z}^k\}_{k\in\mathcal{K}}$ converges to $\hat{\boldsymbol{z}}$ and

$$q(\hat{\boldsymbol{z}}) = \inf\left\{q\left(\boldsymbol{z}^k\right)\right\}_{k\in\mathcal{K}}.$$

Using Lemma 5.2.2, we get that there is $\mu > 0$ such that

$$q(\boldsymbol{z}^0) - q(\hat{\boldsymbol{z}}) \geq \sum_{k\in\mathcal{K}} \mu \|\widetilde{\boldsymbol{\beta}}(\boldsymbol{z}^k)\|^2,$$

so $\|\widetilde{\boldsymbol{\beta}}(\boldsymbol{z}^k)\|$ converges to zero. However, after inspecting the definition of $\widetilde{\boldsymbol{\beta}}$ in (5.4), it follows that also $\|\boldsymbol{\beta}(\boldsymbol{z}^k)\|$ converges to zero. Since for a given $\Gamma > 0$ the non-proportional iterates satisfy

$$\Gamma\|\boldsymbol{\varphi}(\boldsymbol{z}^k)\| < \|\boldsymbol{\beta}(\boldsymbol{z}^k)\|,$$

we conclude that also $\boldsymbol{\varphi}(\boldsymbol{z}^k)$ converges to zero and so does $\boldsymbol{\nu}(\boldsymbol{z}^k)$. Thus $\boldsymbol{\nu}(\hat{\boldsymbol{z}}) = \mathbf{0}$ and, since the solution is unique, also $\hat{\boldsymbol{z}} = \boldsymbol{z}^*$ and $\{\boldsymbol{z}^k\}$ converges to $\boldsymbol{z}^*$. □

Remark 7. The proportioning step is a simple and efficient way of releasing the indices from the active set. However, there are some other options. For example, we can try to use (possibly reduced) conjugate gradient step length in the direction $\boldsymbol{\beta}(\boldsymbol{z}^k)$ or the gradient projection step. The latter is supported by the following theorem which guarantees sufficient decrease of the cost function.

Theorem 6.2.2. Let $\boldsymbol{z}^*$ denote the unique solution of (1.1), let $\lambda_{\min}(\boldsymbol{H})$ denote the smallest eigenvalue of $\boldsymbol{H}$, $\boldsymbol{z} \in \Omega$, $\boldsymbol{g} = \boldsymbol{H}\boldsymbol{z} + \boldsymbol{h}$, and $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$. Then

$$q\big(P_\Omega\left(\boldsymbol{z} - \alpha\boldsymbol{g}\right)\big) - q(\boldsymbol{z}^*) \leq \eta(\alpha)\big(q(\boldsymbol{z}) - q(\boldsymbol{z}^*)\big),$$

where

$$\eta(\alpha) := \max\{1 - \alpha\lambda_{\min}, 1 - (2\|\boldsymbol{H}\|^{-1} - \alpha)\lambda_{\min}\}.$$

Proof. See Bouchala et al. [16]. □

6.3 Algorithm Properties

Firstly observe, that similar to the CGNP, the proposed algorithm executes either projection of the chopped gradient or Newton direction via projected Newton direction path. Compared to the CGNP, the PND algorithm uses the violation of the KKT condition in the proportioning test to decide when the blocking constraint indices should be released from the active set to allow further cost function q reduction.

Hence, opposed to the CGNP, the blocking constraints can be removed from the active set earlier or even later in the iterations depending on the proportionality parameter $\Gamma > 0$. By this parameter, it is possible to "tune" the behavior of the algorithm with respect to when the blocking constraints should be released from the active set. Therefore for the choice of $\Gamma \to 0$ the algorithm prefers releasing the blocking constraints early. On the other hand, the algorithm favors the expansion of the active set firstly with the choice of $\Gamma \to \infty$. Note that the proportionality test also limits premature release of the constraint indices from the active set, hence reducing the computational complexity of the factor update scheme used for face problem solution.

Like the CGNP algorithm, the PND algorithm involves the selection of convergent α step-size for the proportioning step. The issue with its estimation is valid for both algorithms as discussed in Section 5.3. Moreover, the gradient can be updated at each step except when the proportioning step is executed.

To show the numerical behavior of the proposed algorithm, the sensitivity to the proportioning step-size α and proportioning parameters Γ is studied on the same randomly generated QP problem from Section 5.4.1 with $n = 100$ and $\operatorname{cond} \boldsymbol{H} = 1.17e5$. Then the results of CGNP and PND algorithms are compared for a fixed parameter.

In the following figures, the square, circle, cross and diamond markers indicate that the iteration has been the result of initialization, expansion step, proportioning step and full application of the Newton direction, respectively.

6.3.1 Sensitivity to Step Size

In this section we show how the iterations generated by Algorithm 3 depend to the choice of the step-size of the proportioning step $\alpha \in (0, 2\|\boldsymbol{H}\|^{-1})$ for the QP problem of Section 5.4.1. Other parameters have been set as $\varepsilon = 1e{-}6\|\boldsymbol{h}\|$ and $\Gamma = 1$.

Two scenarios have been tested by varying the initial iterate. The purpose of this is to show how the algorithm might behave when the active set has to be mostly expanded or has to be reduced. In the first setup, the algorithm has been started from the center of the feasible set as $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$, hence $\mathcal{A}^0 = \emptyset$. The typical convergence of the PND algorithm is depicted in Figure 6.1 where $|q(\boldsymbol{z}^k) - q(\boldsymbol{z}^*)|$, $\|\boldsymbol{z}^k - \boldsymbol{z}^*\|$, the cardinality of active set $|\mathcal{A}^k|$ and cumulative flops are shown as function of algorithm iteration index k. It is evident that the algorithm starts generating the iterations via expansion steps (circles) and then in the 4th and 8th iterations the proportioning step is executed (cross) as indicated in the drop of the number of constraint indices in the active set. Note that the iterations are almost the same even for a very different α which differs in power of four, showing that algorithm is very insensitive to this parameter for this setup. It is worth mentioning the consecutive application of the full Newton direction and the proportioning step in the 7th and 8th iterations. This follows immediately from the result of Lemma 4.5.1, since the application of Newton direction will solve the face problem, hence eliminating the free gradient. As a consequence, the next iteration is considered to be not proportional[2] by the proportioning test (6.1).

[2] If $\boldsymbol{z}^{k+1}$ is not KKT optimal.

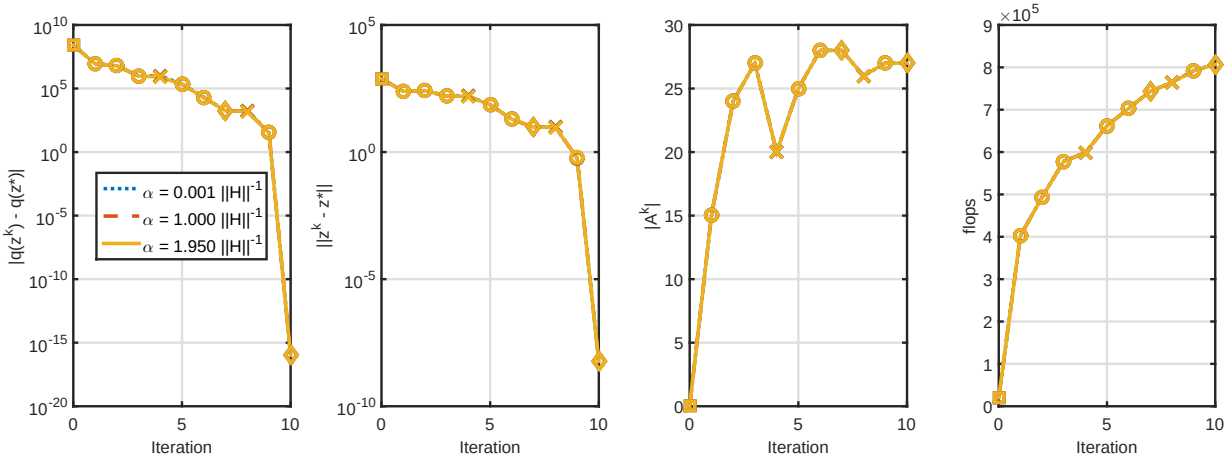


Figure 6.1: Comparison of convergence of PND algorithm for several step-sizes α of proportioning step for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}}+\underline{\boldsymbol{z}})/2$.

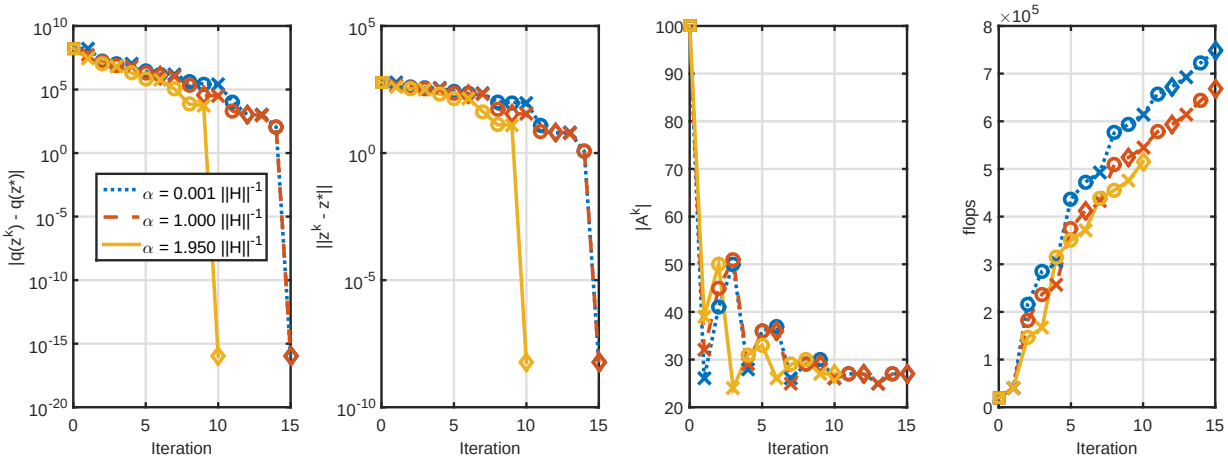


Figure 6.2: Comparison of convergence of PND algorithm for several step-sizes α of proportioning step for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$.

In the second setup, the initial iterate has been set to upper bound i.e., $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$, i.e., $\mathcal{A}^0 = \mathcal{I}$. Hence all the constraints have been activated at the initial iteration, and PND algorithm has to start with the proportioning step to remove blocking constraint indices from the active set. In this example, the algorithm is sensitive to the choice of α as illustrated in Figure 6.2. The reason is that the application of proportioning with larger step-size leads to the activation of the opposite bound (lower bound) which is not happening for a very short choice of α. This is evident e.g., from the first iteration where the active set cardinality for choice of $\alpha = 1.95\|\boldsymbol{H}\|^{-1}$ drops to 39 while for shorter choice it is 32 or 26 respectively although the initial iterate has been the same. As a result, the algorithm involves more iterations for shorter choice of α, but our experiments suggest that the difference is often in order of $1-5$ additional iterations. It is worth noting that the difference of the number of involved flops for the selection

of α for shortest and longest choice in the test which is about 31 % of the flops needed to solve the QP problem with $\alpha = 1.95\|\boldsymbol{H}\|^{-1}$. This indicates that the algorithm performance can be degraded for too short α for the case that opposite bounds are activated during the iterations.

6.3.2 Sensitivity to Proportioning Parameter

Since the proportioning parameter $\Gamma > 0$ influences the decision when the blocking constraints are removed from the active set, its selection has a large impact to the algorithm performance. To illustrate this, the QP problem of Section 5.4.1 has been solved by the PND algorithm for a set of choices as $\Gamma = \{0.001, 1, 10\}$, $\varepsilon = 1e{-}6\|\boldsymbol{h}\|$ and fixed $\alpha = 1.95\|\boldsymbol{H}\|^{-1}$ since this value showed best results in Section 6.3.1.

Two scenarios have been tested varying the initial iterate. The purpose of this is to show how the algorithm might behave when the active set has to be mostly expanded or has to be reduced. In the first setup, the algorithm has been started from the center of the feasible set as $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$, hence $\mathcal{A}^0 = \emptyset$. The typical convergence of the PND algorithm is depicted in Figure 6.3. The results confirm, that for a low choice of Γ, the PND algorithm prefers release of not optimal constraints in the iterations. This can be observed e.g., for $\Gamma = 0.1$ where the proportioning step (cross) is executed already in the 3rd iteration. On the other hand, the first proportioning step is executed in the 6th iteration preceded by the application of full Newton direction (diamond) which follows from Lemma 4.5.1 for $\Gamma = 10$.

Comparing the cumulative flops in Figure 6.3, it seems that Γ should be selected small for this setup. But our further experiments suggest that, e.g., choice of $\Gamma = 0.01$ leads to about the same flops as for $\Gamma = 1$ and $\Gamma = 0.001$ leads to increase of flops of about 5 % compared to the choice with $\Gamma = 1$. Hence the sensitivity of Γ to the number of performed flops of the algorithm to converge to the solution with a given accuracy is not monotone function, complicating its optimal selection.

In the second scenario, the initial iterate has been set to upper bound, i.e., $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$, i.e., $\mathcal{A}^0 = \mathcal{I}$. Hence all constraints have been active at the initial iteration, and the PND algorithm has to start with proportioning step to remove blocking constraint indices from the active set. Similarly to the results with initial iterate as the center of feasible set, the minimum flops is performed by the algorithm with Γ close to 1 as shown in Figure 6.4. The reason is that for low choice of Γ the active set is changed often rapidly leading to the computational expensive updates of the Cholesky factor in the face problem solution. This is indicated by the large increase of cumulative flops, e.g., in the 4th iteration for $\Gamma = 0.1$. Interesting are the two successive applications of the proportioning step (crosses) for $\Gamma = 0.1$ in the 2nd and 3rd iteration. This is because the proportioning step in the 2nd iteration adds some constraint indices to the active set leading to non-proportional iteration due to the low Γ.

From the computational complexity point of view the minimum number of flops is executed for $\Gamma = 1$ in our setup. Both higher and lower values of Γ than 1 leads to higher computational complexity. This contradicts the result from the previous experiment where lower computational complexity is for $\Gamma < 1$. But as in real-world application of the algorithm both setups ($\mathcal{A}^0 = \emptyset$ and $\mathcal{A}^0 = \mathcal{I}$) can happen we suggest using Γ close to 1. This choice is also supported by the numerical experiments done later in Section 8.

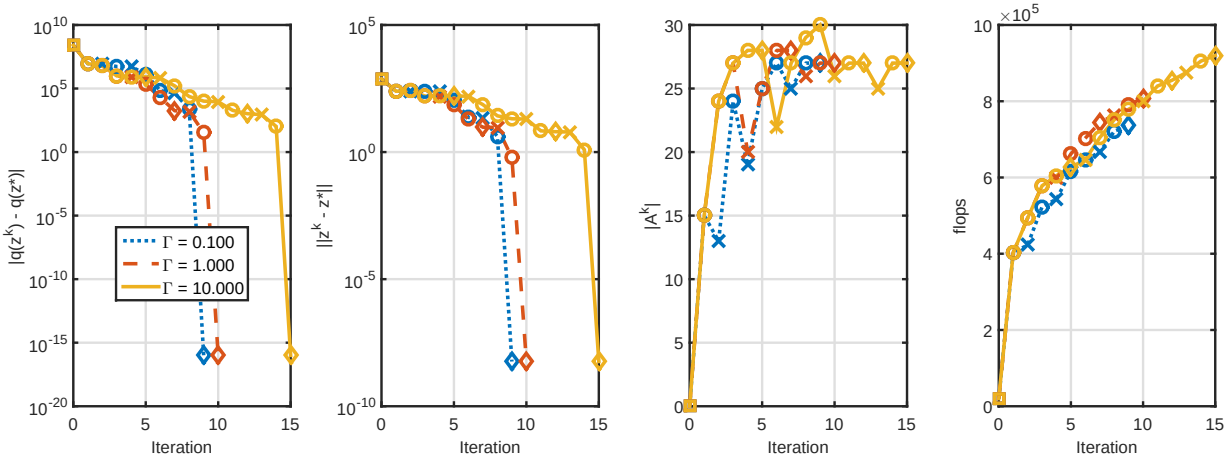


Figure 6.3: Comparison of convergence of PND algorithm for several values of proportioning parameter Γ for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}}+\underline{\boldsymbol{z}})/2$.

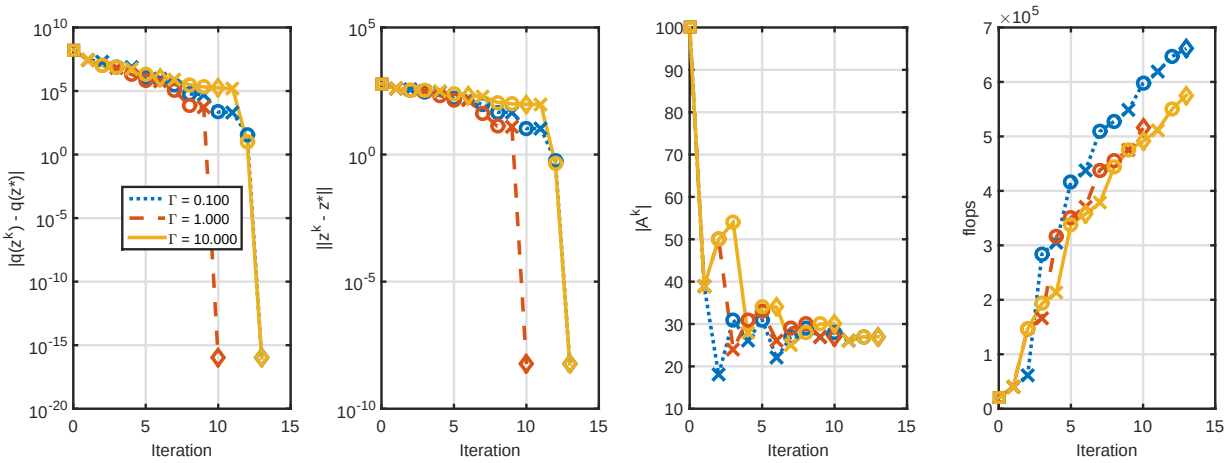


Figure 6.4: Comparison of convergence of PND algorithm for several values of proportioning parameter Γ for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$.

6.3.3 Comparison to CGNP Algorithm

The main difference between CGNP and PND algorithms is when the blocking constraint indices are removed from the active set. While in the CGNP algorithm the blocking constraints are released only in the rare situation that the solution of the face problem is feasible, the decision based on the violation of KKT conditions via proportioning test (6.1) is used in PND algorithm.

6.3.3.1 Random QP Problem

To show the effect of this difference on the algorithms convergence, the QP problem of Section 5.4.1 has been solved with $\varepsilon = 1e{-}6\|\boldsymbol{h}\|$ and fixed $\alpha = 1.95\|\boldsymbol{H}\|^{-1}$ by the CGNP and PND algorithms. Based on the results from Section 6.3.2 we select $\Gamma = 1$.

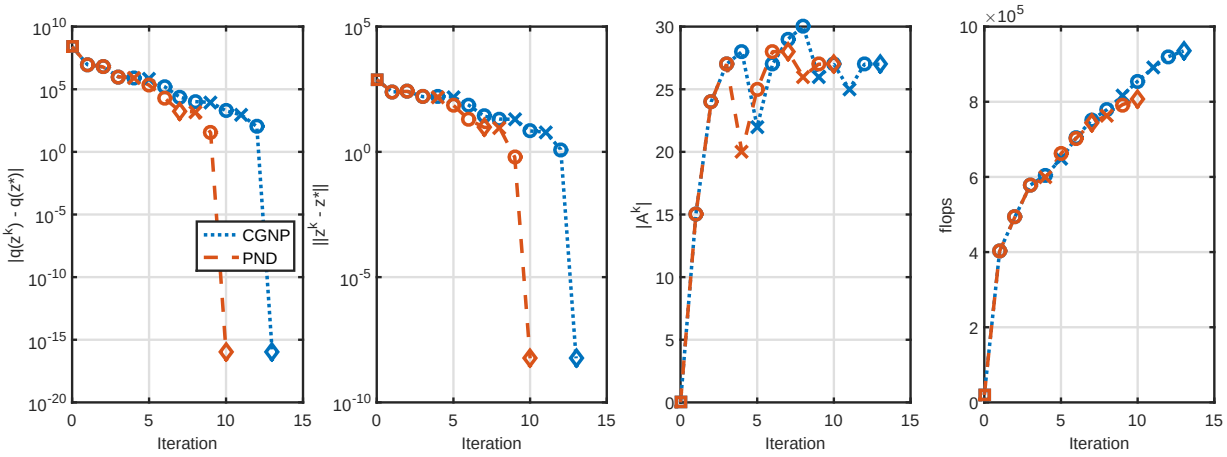


Figure 6.5: Comparison of convergence of CGNP and PND algorithms for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$.

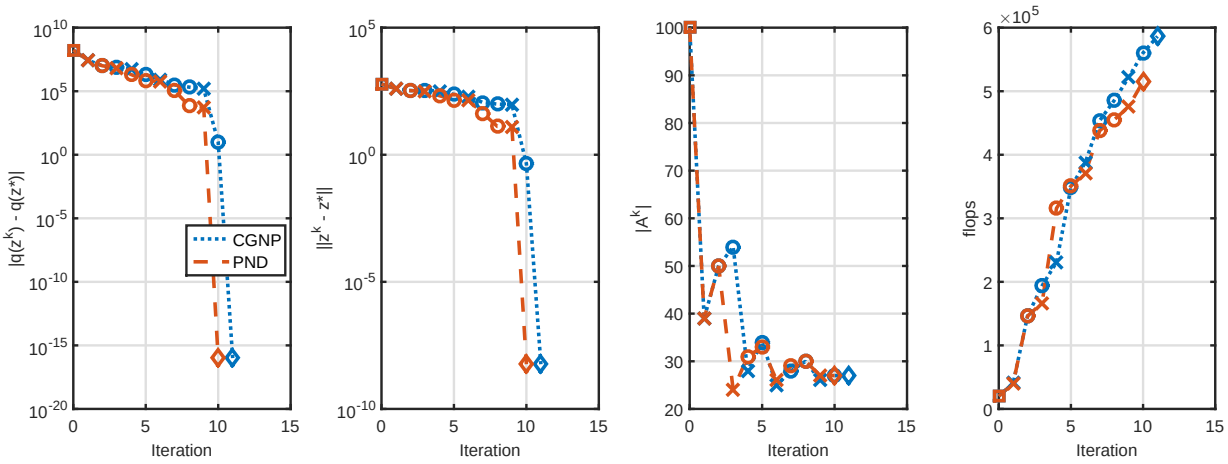


Figure 6.6: Comparison of convergence of CGNP and PND algorithms for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$.

Again two scenarios have been studied with varying the initial iteration. In the first setup, the algorithms have been started from the center of the feasible set as $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$, hence $\mathcal{A}^0 = \emptyset$. The comparison of convergence of CGNP and PND algorithms is depicted in Figure 6.5. Both algorithms start with the expansion steps (circles) generating the same iterate up to the 4th iteration when PND executes the proportioning since the iteration is detected as not proportional by the proportionality test (6.1). This confirms the hypothesis that the PND algorithm allows the earlier release of blocking constraints by the execution of the proportioning step (crosses). As the blocking constraints are released earlier for PND algorithm the number of involved iterations is reduced and also the cost function decrease is steeper than for CGNP. In terms of the computational complexity, PND algorithm involves 14% less flops compared to CGNP in this setup.

In the second scenario, the initial iterate has been set to upper bound, i.e., $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$, i.e., $\mathcal{A}^0 = \mathcal{I}$. Hence all constraints have been activated at the initial iteration, and both algorithms have to start with the proportioning step to remove blocking constraint indices from the active set. Similar to the case $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$, the proportioning step is executed earlier for PND algorithm allowing further cost function decrease earlier than for CGNP. As a result, PND involves one iteration less and 12% reduction of the flops compared to CGNP algorithm.

6.3.3.2 Illustrative QP problem

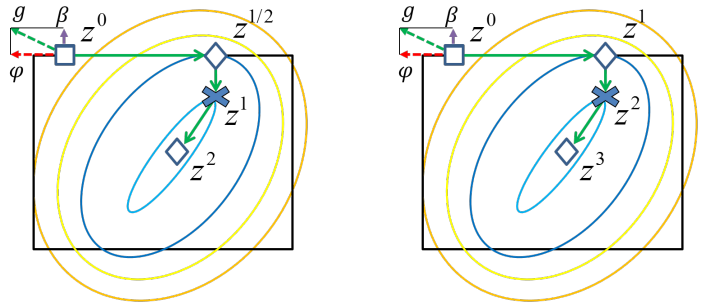
To show the difference between CGNP and PND algorithms in more detail, the illustrative QP problem with $n = 2$ has been setup. Two scenarios depending on the proportionality of the initial iterate for a fixed $\Gamma = 1$ have been investigated. The reason for such analysis is to show how the algorithms differ in the behavior when iteration is proportional or not during the iterations.

In the first case, the initial iterate $\boldsymbol{z}^0$ has been generated to be proportional based on the test (6.1). The comparison of the iterate for such a case is depicted in Figure 6.7. As in previous figures, the square, circle, cross and diamond markers indicate that the iteration has been the result of initialization, expansion step, proportioning step and full application of the Newton direction, respectively.

As the initial iterate $\boldsymbol{z}^0$ is proportional, both algorithms start by the computation of the Newton direction. In both algorithms, the whole Newton step can be applied up to the face minimizer. Hence for CGNP algorithm the Newton direction is applied (diamond) and intermediate iterate $\boldsymbol{z}^{1/2}$ is obtained by the step 12 of Algorithm 2. As the KKT conditions are not fulfilled, the proportioning step (cross) is executed to generate iterate $\boldsymbol{z}^1$. The solution is then generated in the 2nd iteration by application of the whole Newton direction (diamond). On the other hand, the full application of the Newton direction (diamond) concludes the first iteration of PND algorithm. The proportioning step (cross) is executed in the 2nd iteration to create iterate $\boldsymbol{z}^2$, successive by the application of the full Newton direction (diamond) in the 3rd iteration.

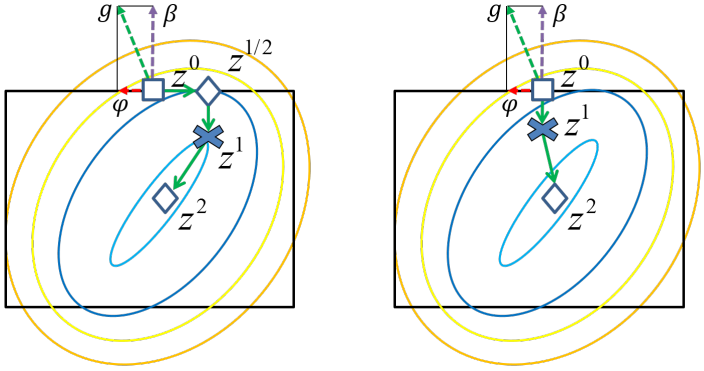
Therefore for the case that the iterate is proportional such that the minimizer of the corresponding face problem is feasible and is not KKT optimal, the PND algorithm involves one additional iteration compared to the CGNP algorithm since the proportioning step is executed in the following iteration of PND as a consequence of Lemma 4.5.1. On the other hand, the number of the involved flops is about the same for both algorithms for such a situation, since both algorithms execute the same functionality which is only split into one additional iteration for PND.

In the second example presented in Figure 6.8, the initial iterate $\boldsymbol{z}^0$ is generated as not proportional based on the test (6.1). Since CGNP algorithm does not distinguish the proportionality of the iterate it generates the same sequence of iterates as in the case for initial proportional iterate. On the other hand, PND algorithm recognizes the non-proportionality of $\boldsymbol{z}^0$ and executes the proportioning step (cross) in the first iteration followed by the full application of the Newton direction (diamond) in the second iteration. Hence both algorithms involve the same number of iterations. But they execute different parts of the algorithm. For example,



(a) CGNP **(b)** PND

Figure 6.7: Comparison of iterations generated by CGNP and PND when solving ilustrative QP problem with proportional initial iteration.



(a) CGNP **(b)** PND

Figure 6.8: Comparison of iterations generated by CGNP and PND when solving ilustrative QP problem with non-proportional initial iteration.

the CGNP executes the computation of Newton direction (i.e., update of the factor and two substitutions as described in Section 4.3) twice. Whereas PND executes the computation of Newton direction only once in the last iteration. Hence PND might involve less flops compared to the CGNP when iteration is detected as not proportional by the early release of the blocking constraints.

6.4 Summary

The main building blocks of the PND algorithm are very similar to the CGNP algorithm presented in Section 5. The PND algorithm consists of face problem minimization by direct solver using Cholesky factor update, expansion of the active set via projected Newton direction path utilizing the PLS and the proportioning step via projection of chopped gradient with fixed step-size. Further, PND algorithm adds proportionality test (6.1), based on which it decides

whether the active set will be expanded or reduced. Since the proportionality test takes directly into account the absolute value of not optimal Lagrange multipliers, PND removes the main disadvantages of the CGNP algorithm. Hence the proportionality test in PND enables earlier reduction of the active set compared to CGNP algorithm. Moreover, by the proportionality parameters Γ in the proportional test, it is possible to "tune" the preference of the algorithm for reduction of the active set (for low values of Γ) or rather the expansion (for high values of Γ).

Similar to CGNP, the PND algorithm indicates insensitivity to the choice of the proportioning step-size when opposite bounds are not activated. It was shown that PND always needs one additional iteration compared to CGNP when the proportional iteration leads to the feasible face minimizer. On the other hand, when the iteration is not proportional, the PND algorithm might save solution of face problem compared to CGNP algorithm.

Contrary to the algorithm presented in [14], PND uses computationally cheaper fixed step-size instead of non-monotone line search. Furthermore, PLS routine with linear computational complexity in QP problem size is used in PND rather than expensive backtracking search to expand the active set.

Chapter 7

Newton Projection with Proportioning

The bottleneck of the PND algorithm presented in Chapter 6 is need of matrix norm of the QP problem Hessian for selection of step-size α in the proportioning step. This can introduce significant computational overhead as described in Section 5.3. Moreover, as the proportioning step uses chopped gradient for projection it is not possible to update the gradient as described in Section 4.5 reusing the computed $\boldsymbol{Hp}$ leading to unwanted expensive computation. The algorithm introduced in this section shows that both bottlenecks can be resolved by elimination of the projection of the chopped gradient in the proportioning step.

In this chapter firstly, the specific ingredients of the proposed method are described in detail together with the algorithm referring to the basic ingredients described in Chapter 4. Then the convergence of the algorithm is proved and main properties of the algorithm are analyzed in the illustrative QP problem and compared to CGNP and PND algorithms.

7.1 Algorithm

Analyzing the alternative refinement of the KKT conditions in (4.1) and definition of the chopped gradient $\boldsymbol{\beta}$ in (4.2) one can deduce that $\boldsymbol{\beta}$ indicates the not-optimality[1] of the i-th activated constraints by nonzero element of β_i. In PND algorithm (Algorithm 3) all the blocking constraints are released by the proportioning step (6.2). Hence by applying the proportioning step, the blocking constraints are removed from the active set without activation new constraints[2] leading to zero of the chopped gradient $\boldsymbol{\beta}$ based on the definition in (4.2). As a consequence, the next iteration is considered as the proportional since

$$\|\boldsymbol{\beta}(\boldsymbol{z}^k)\| = 0 \leq \Gamma\|\boldsymbol{\varphi}(\boldsymbol{z}^k)\|,$$

and the face problem (4.3) is solved considering the released constraints by the proportional step as free. Simply said, the proportioning step in PND serves mainly as a tool for deactivation

[1] In terms of the sign of the Lagrange multipliers.

[2] If the opposite bound is not activated by the proportioning step, which happens rarely in the application of MPC.

of the constraints which are then assumed to be not active in the face problem in the following iteration.

The idea of the presented algorithm which we call Newton Projection with Proportioning (NPP) is to completely eliminate the projection of the chopped gradient in the proportioning step. Instead, when the iteration is detected not to be proportional (see Section 6.1.1), the algorithm releases all the blocking constraints indicated by non-zero (not optimal) components of the chopped gradient by the modification of the set which defines the face problem.

The release of the not optimal constraint indices from the set defining the face problem based on the gradient entries has been done also in [11] in the context of the Projected Newton Method (PNM) algorithm. There, the constraint indices have been released immediately whenever they become not-optimal over the course of the algorithm iterations. On the other hand, the NPP algorithm executes a release of the constraint indices only when the iteration becomes not proportional based on the test (6.1). This allows to limit the unwanted changes in the active set and reduce the computational complexity of the factor update scheme used for the solution of the face problem. Furthermore, from a scratch computation of the gradient after the proportioning step is eliminated compared to the PND algorithm.

The proposed algorithm NPP is presented in Algorithm 4. Similar to the PND algorithm it uses the proportionality test (6.1) to decide which part of the gradient will be reduced. The procedure starts by checking the KKT condition for iterate via a norm of the projected gradient as defined in Section 4.2. In case that it is not optimal, the free and the chopped gradient are computed and the proportioning test (6.1) is evaluated. An overview of the computational complexity of all the algorithm steps is presented in Table 7.1 concluding that the overall computational complexity of each algorithm iteration is $\mathcal{O}(n^2)$.

7.1.1 Proportioning

Depending on the proportionality test (6.1) the algorithm defines the *face upper* and *lower sets* at the k-th algorithm iteration as

$$\mathcal{U}_F^k = \begin{cases} \mathcal{U}^k & \text{if } \|\boldsymbol{\beta}(\boldsymbol{z}^k)\| \leq \Gamma\|\boldsymbol{\varphi}(\boldsymbol{z}^k)\|, \\ \{i \in \mathcal{U}^k : \beta_i\left(\boldsymbol{z}^k\right) = 0\} & \text{otherwise} \end{cases} \tag{7.1}$$

$$\mathcal{L}_F^k = \begin{cases} \mathcal{L}^k & \text{if } \|\boldsymbol{\beta}(\boldsymbol{z}^k)\| \leq \Gamma\|\boldsymbol{\varphi}(\boldsymbol{z}^k)\|, \\ \{i \in \mathcal{L}^k : \beta_i\left(\boldsymbol{z}^k\right) = 0\} & \text{otherwise} \end{cases} . \tag{7.2}$$

Furthermore, the *face active set* is defined as $\mathcal{A}_F^k = \mathcal{U}_F^k \cup \mathcal{L}_F^k$. Note that the check for zero components of the chopped gradient in (7.1) and (7.2) can be done exactly since the zeros are produced by the definition (4.2).

When the iteration is proportional, the face active set $\mathcal{A}_F^k$ is the same as the active set $\mathcal{A}^k$. On the other hand, when the iteration is detected to be non-proportional, the face active set contains only those constraint indices from the active set which are indicated by the zero components of the chopped gradient.

Algorithm 4 Newton Projection with Proportioning (NPP) algorithm. Given a SPD matrix $\boldsymbol{H}$ of the order n, n-vectors $\boldsymbol{h}$, $\underline{\boldsymbol{z}}$, $\overline{\boldsymbol{z}}$, $\Omega = \{\boldsymbol{z} : \underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}}, \underline{\boldsymbol{z}} < \overline{\boldsymbol{z}}\}$, $\boldsymbol{z}^0 \in \Omega$, $\Gamma > 0$ and $\varepsilon > 0$.

{Initialization}
Set $k = 0$, $\boldsymbol{g}\left(\boldsymbol{z}^k\right) = \boldsymbol{H}\boldsymbol{z}^k + \boldsymbol{h}$
while $\|\boldsymbol{\nu}\left(\boldsymbol{z}^k\right)\| \geq \varepsilon$ **do**
  if $\|\boldsymbol{\beta}(\boldsymbol{z}^k)\| \leq \Gamma \|\boldsymbol{\varphi}(\boldsymbol{z}^k)\|$ **then**
    {Proportional $\boldsymbol{z}^k$}
    $\mathcal{U}_F = \mathcal{U}$,
    $\mathcal{L}_F = \mathcal{L}$, $\mathcal{A}_F = \mathcal{A}$
  else
    {Non-proportional $\boldsymbol{z}^k$}
    $\mathcal{U}_F = \{i \in \mathcal{U} : \beta_i\left(\boldsymbol{z}^k\right) = 0\}$,
    $\mathcal{L}_F = \{i \in \mathcal{L} : \beta_i\left(\boldsymbol{z}^k\right) = 0\}$, $\mathcal{A}_F = \{i \in \mathcal{I} : i \in \mathcal{U}_F \cup \mathcal{L}_F\}$
  end if
  Solve (7.4) to obtain Newton direction $\boldsymbol{p}^k$ assuming face active set $\mathcal{A}_F$.
  Compute $\boldsymbol{H}\boldsymbol{p}^k$ as described in Section 4.5 assuming face active set $\mathcal{A}_F$.
  $\alpha_f = \max\{\alpha : \boldsymbol{z}^k + \alpha \boldsymbol{p}^k \in \Omega\}$
  if $\alpha_f < 1$ **then**
    {Expansion step}
    $\left[\boldsymbol{z}^{k+1}, \boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right)\right] = \mathrm{PLS}\left(\boldsymbol{z}^k, \boldsymbol{p}^k, \boldsymbol{H}\boldsymbol{p}^k, \boldsymbol{g}\left(\boldsymbol{z}^k\right)\right)$
  else
    {Full Newton direction can be applied to remain feasible}
    $\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \boldsymbol{p}^k$
    $\boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{g}\left(\boldsymbol{z}^k\right) + \boldsymbol{H}\boldsymbol{p}^k$
  end if
  $k = k + 1$
end while
$\boldsymbol{z}^* = \boldsymbol{z}_k$

7.1.2 Modified Face Problem

The NPP algorithm solves at each iteration a modified face problem

$$\tilde{\boldsymbol{z}} = \arg \min_{\boldsymbol{z} \in \Psi} q(\boldsymbol{z}), \tag{7.3}$$

where

$$\Psi = \{\boldsymbol{z} : z_i = \overline{z}_i \ \text{ for } \ i \in \mathcal{U}_F^k \ \text{ and } \ z_i = \underline{z}_i \ \text{ for } i \in \mathcal{L}_F^k\},$$

which can be solved with the same procedure as described in Sections 4.3 and 4.4 with little abuse of notation, denoting the active set $\mathcal{A}^k$ as the face active set $\mathcal{A}_F^k$ and the computation of the Newton step (4.13) is obtained as

$$\boldsymbol{p}^k = \begin{cases} \boldsymbol{p}^k_{\mathcal{I} \setminus \mathcal{A}_F^k} & = -\boldsymbol{G}^{-1}\boldsymbol{r}^k \\ \boldsymbol{p}^k_{\mathcal{A}_F^k} & = \boldsymbol{0}, \end{cases} \tag{7.4}$$

Table 7.1: Computational complexity of the NPP algorithm steps.

Algorithm Step	Complexity (flops)	Notation	Described
$\boldsymbol{g}\left(\boldsymbol{z}^k\right) = \boldsymbol{H}\boldsymbol{z}^k + \boldsymbol{h}$	$2n^2$		
$\|\boldsymbol{\beta}(\boldsymbol{z}^k)\| \leq \Gamma\|\boldsymbol{\varphi}(\boldsymbol{z}^k)\|$	$2n$		Section 6.1.1
$\boldsymbol{R}^{k-1} \rightarrow \boldsymbol{R}^k$	$\mathcal{O}\left(\lvert\mathcal{A}^k_{\text{add}}\rvert(n - m^{k-1} - \lvert\mathcal{A}^k_{\text{add}}\rvert)^2\right) +$ $\mathcal{O}\left(\lvert\mathcal{A}^k_{\text{rem}}\rvert^3\right) + \mathcal{O}\left(\lvert\mathcal{A}^k_{\text{rem}}\rvert^2 n_f\right) +$ $\mathcal{O}\left(\lvert\mathcal{A}^k_{\text{rem}}\rvert n_f^2\right)$	$m^{k-1} = \mathcal{A}_F^{k-1}$ $n_f = n - m^{k-1} - \lvert\mathcal{A}^k_{\text{add}}\rvert$	Section 4.4
$(\boldsymbol{R}^k)^T\boldsymbol{y} = -\boldsymbol{r}\left(\boldsymbol{z}^k\right)$ $\boldsymbol{R}^k\boldsymbol{p}^k = \boldsymbol{y}$	$2(n - m^k)^2$	$m^k = \lvert\mathcal{A}_F^k\rvert$	Section 4.3
$\boldsymbol{H}\boldsymbol{p}^k$	$2m^k(n - m^k)$		Section 4.5
$\alpha_f = \max\{\alpha : \boldsymbol{z}^k + \alpha\boldsymbol{p}^k \in \Omega\}$	$2n$		Section 5.1.1
$\left[\boldsymbol{z}^{k+1}, \boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right)\right] =$ PLS $\left(\boldsymbol{z}^k, \boldsymbol{p}^k, \boldsymbol{H}\boldsymbol{p}^k, \boldsymbol{g}\left(\boldsymbol{z}^k\right)\right)$	$2(n - m^k) + \sum_{i=1}^{s}(2nr_i + 10n)$	r_i constraints changed on the i-th line segment	Section 4.6
$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \boldsymbol{p}^k$	n		
$\boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{g}\left(\boldsymbol{z}^k\right) + (\boldsymbol{H}\boldsymbol{p}^k)$	n		

with

$$\boldsymbol{G} = \boldsymbol{H}_{\mathcal{I}\setminus\mathcal{A}_F^k, \mathcal{I}\setminus\mathcal{A}_F^k}, \qquad \boldsymbol{r}^k = \boldsymbol{g}^k_{\mathcal{I}\setminus\mathcal{A}_F^k}.$$

If the iteration is proportional, the face problem (7.3) is solved assuming that the face active set is the same as the active set. On the other hand, when the iteration is not proportional, the constraint indices which block further cost function decrease are released from the face active set. The face problem is solved as in the PND solver via Cholesky factorization with the factor update and substitutions as described in Section 4.3 and 4.4.

When the Newton direction $\boldsymbol{p}^k$ is applied two scenarios may happen: (i) some new constraints are activated, i.e., $\boldsymbol{z}^k + \boldsymbol{p}^k \notin \Omega$, hence it can be applied only with a limited step size $\mu \in (0, 1)$ such that $\boldsymbol{z}^k + \mu\boldsymbol{p}^k \in \Omega$; (ii) full Newton direction can be applied without violation of any constraints, i.e., $\boldsymbol{z}^k + \boldsymbol{p}^k \in \Omega$. In order to decide which situation occurs, the maximal step size $\alpha_f \in (0, 1]$ leading to the feasible iterate along the direction $\boldsymbol{p}^k$, i.e., $\alpha_f = \min\{1, \max\{\alpha : \boldsymbol{z}^k + \alpha\boldsymbol{p}^k \in \Omega\}\}$ is computed by (5.1).

7.1.3 Expansion Step

If the face problem (7.3) minimizer is not feasible, i.e., $\boldsymbol{z}^k + \alpha_f\boldsymbol{p}^k \in \Omega$, $\alpha_f < 1$, the active set should be expanded to allow further cost function q decrease. For that, the projected-Newton-direction path utilizing the PLS algorithm described in Section 4.6 is used and the gradient is updated along the projected path.

7.1.4 Application of Newton Direction

If the solution of the modified face problem (7.3) is feasible, i.e., $\boldsymbol{z}^k + \boldsymbol{p}^k \in \Omega$, the Newton direction $\boldsymbol{p}^k$ which points towards the face minimizer is applied and the gradient is updated by

$$\boldsymbol{z}^{k+1} = \boldsymbol{z}^k + \boldsymbol{p}^k, \qquad \boldsymbol{g}\left(\boldsymbol{z}^{k+1}\right) = \boldsymbol{g}\left(\boldsymbol{z}^k\right) + (\boldsymbol{H}\boldsymbol{p}^k),$$

using the pre-computed $\boldsymbol{H}\boldsymbol{p}^k$ as described in Section 4.5.

7.2 Convergence

In this section the convergence of NPP algorithm (Algorithm 4) is proved. The proof is based on the idea of cost function decrease at each iteration together with the finite number of faces which algorithm can explore. To prove the convergence of NPP algorithm, the lemma showing the cost function decrease is firstly stated and proved.

Lemma 7.2.1. Let $\boldsymbol{z}^k, \boldsymbol{z}^{k+1} \in \Omega$ are produced by the NPP algorithm (Algorithm 4). Then there exists $\gamma > 0$ such that

$$q\left(\boldsymbol{z}^k\right) - q\left(\boldsymbol{z}^{k+1}\right) \geq \gamma.$$

Proof. First observe that the Newton direction $\boldsymbol{p}^k$ which points towards the minimizer of the modified face problem (7.3) is used in the projected-Newton-direction path in the PLS algorithm to find the first local minimizer along the projected path as described in Section 4.6. Second observe that the Newton direction $\boldsymbol{p}^k$ generated via solution of (4.13) is always descent direction as indicated in Lemma 4.3.2. This is however not true in general for the projected direction $\boldsymbol{d}$ as defined in (4.18) at each explored line segment in PLS algorithm. The only exception is the first line segment where it is true since $\boldsymbol{d}^0 = \boldsymbol{p}^k$. Moreover, as the growth of function $q\left(\boldsymbol{z}\right)$ is checked in PLS as a stopping condition of exploration, the PLS always performs, at least, one line segment exploration with cost function decrease. Note that the application of the whole Newton direction, if leads to feasible iterate, is equivalent to running the PLS algorithm which would find $\Delta t^* = 1$. Hence, only the application of the PLS has to be analyzed to prove the algorithm convergence.

Let $\Delta t^* > 0$ is defined by (5.3) and difference of the two successive ordered breakpoints by $\delta t_j = t_j - t_{j-1} > 0$. Then using (4.17), (4.18) and Lemma 5.2.1 we get

$$\begin{aligned} q\left(\boldsymbol{z}^k\right) - q\left(\boldsymbol{z}^{k+1}\right) &\geq \sum_{j=1}^{j=w-1} \{-\delta t_j \boldsymbol{g}(\boldsymbol{z}^k(t_{j-1}))^T \boldsymbol{d}^{j-1} - (1/2)\delta t_j^2 \|\boldsymbol{H}\| \|\boldsymbol{d}^{j-1}\|^2\} + \\ &+ (-\Delta t^*_{w-1} \boldsymbol{g}(\boldsymbol{z}^k(t_j))^T \boldsymbol{d}^{w-1} - (1/2)(\Delta t^*_{w-1})^2 \|\boldsymbol{H}\| \|\boldsymbol{d}^{w-1}\|^2) = \\ &= \gamma > 0, \end{aligned} \tag{7.5}$$

with w denoting the number of line segments successively explored by the PLS algorithm for which the cost function decrease has been detected. As the terms in the sum are all positive based on the Lemma 5.2.1, the right-hand side of inequality (7.5) is also positive which proves the lemma. □

Now we are ready to prove the convergence of Algorithm 4.

Theorem 7.2.2 (NPP Convergence)**.** Let $\{\boldsymbol{z}^k\}$ denote the iterates generated by the NPP algorithm for the solution of (1.1) with $\Gamma > 0$. Then, $\{\boldsymbol{z}^k\}$ converges to the solution $\boldsymbol{z}^*$ of (1.1).

Proof. Based on Lemma 7.2.1 the $\{q\left(\boldsymbol{z}^k\right)\}$ is a decreasing function such that

$$q\left(\boldsymbol{z}^{k+1}\right) < q\left(\boldsymbol{z}^k\right),$$

and which is bounded from below by the unconstrained minimum of q. From the fact that each iteration $\boldsymbol{z}^k \in \Omega$ is based on the solution of the modified face problem defined by the face active set $\mathcal{A}_F^k$ and processed by the PLS algorithm it follows that no face active set can reappear. From the uniqueness of solution of (1.1) and as the number of different face active sets is finite, we can conclude that NPP algorithm finds the solution of (1.1) in the finite number of iterations. □

7.3 Algorithm Properties

When compared to CGNP and PND algorithms, the NPP algorithm depicted in Algorithm 4 executes only projection of the Newton direction which points towards the minimizer of a modified face problem (7.3). Hence, there is missing a step of projection of the chopped gradient. This served as a tool for releasing of the not-optimal constraint indices from the active set in CGNP and PND algorithms. The NPP algorithm instead uses a modification of the set of constraints which defines the face problem when the iteration is detected non-proportional by the proportionality test (6.1). As a consequence, the NPP algorithm does not require any knowledge of the QP problem Hessian norm, which is convenient, e.g., in the MPC applications where QP problem Hessian might change at each sample time due to the change of controller tuning or in a case of nonlinear MPC.

Similarly as for PND algorithm, by use of the proportionality test (6.1), the violation of KKT conditions due to the blocking constraints which are present in the active set is taken into account. Furthermore, the proportionality parameter $\Gamma > 0$ used in the proportionality test defines the preference of the algorithm when the blocking constraint indices should be released from the active set. Therefore, for a choice of $\Gamma \to \infty$ the algorithm prefers the active set expansion. The blocking constraint indices are removed later in the iteration process when the KKT conditions violation caused by the chopped gradient dominates the violation by the free gradient. On the other hand, the algorithm releases all blocking constraints at each iteration similar to PNM of [11] for $\Gamma \to 0$. Therefore, the proportionality test with $\Gamma > 0$ serves as a tool which helps to reduce a premature release of the constraints and decreasing the computational complexity of the factor update.

When the iteration is considered not proportional, the NPP algorithm can save one iteration compared to the PND since it can produce the proportioning and expansion in one iteration. Whereas PND has to execute the proportioning step firstly and in the following iteration the active set is expanded. Further, when the proportioning step would lead to the activation of some opposite bounds, the active set is not changed[3] for PND algorithm. But in the NPPalgorithm, the blocking constraints have to be firstly released from the face active set in one iteration and in the next iteration added back which increases the computational complexity of factor update.

[3] Since the active set is defined as $\mathcal{A} = \mathcal{L} \cup \mathcal{U}$

To show the numerical behavior of the proposed algorithm, the sensitivity to the proportioning parameter Γ is studied on the same randomly generated QP problem from Section 5.4.1 with $n = 100$ and $\operatorname{cond} \boldsymbol{H} = 1.17e5$. Then the convergence of NPP algorithm is compared to CGNP and PND algorithms for a fixed parameter.

In the following figures, the square, circle, cross, diamond and star markers indicate that the iteration has been the result of initialization, expansion step without proportioning, expansion with proportioning step and application of full Newton direction without proportioning, and with proportioning, respectively.

7.3.1 Sensitivity to Proportioning Parameter

Since the proportioning parameter $\Gamma > 0$ influences the decision when the blocking constraints are removed from the active set, its selection has large impact to the algorithm performance. To illustrate this, QP problem of Section 5.4.1 has been solved by NPP algorithm for set of choices $\Gamma = \{0.1, 1, 10\}$ and $\varepsilon = 1e{-}6\|\boldsymbol{h}\|$.

Two scenarios have been tested varying the initial iterate. In the first setup, the algorithm has been started from the center of the feasible set as $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$, hence $\mathcal{A}^0 = \emptyset$. The typical convergence of NPP algorithm is depicted in Figure 7.1. The results confirm that for the low choices of Γ the proportioning (cross) is executed more often while for large values it is executed only rarely and algorithm prefers the expansion without proportioning (circle). Since NPP algorithm can expand the active set at the same iteration when proportioning is executed (cross) the optimal active set is changing rapidly, leading to the low number of involved iterations. From the computational complexity point of view, least flops is executed for the choice $\Gamma = 0.1$ in our setup. On the other hand, decreasing Γ further leads to increase of the executed flops since a premature release of the blocking constraints from the active set which have to be returned back to the active set later.

In the second scenario, the initial iterate has been set to the upper bound and $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$, i.e., $\mathcal{A}^0 = \mathcal{I}$. Hence all the upper bound constraints are active at the initial iteration, and NPP algorithm has to start with the proportioning step to remove the blocking constraint indices from the active set, see Figure 7.2. Again, the choices of relatively low Γ leads to more frequent execution of the proportioning. Moreover, in this example the proportioning (crosses) is executed at each iteration for $\Gamma = 0.1$ leading to the need of only 7 iterations of the algorithm. Similar to the case for $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$, the choice of $\Gamma = 0.1$ leads to the lowest number of flops but with only about 1.5% margin compared to the choice $\Gamma = 10$ and less than 1% to the case $\Gamma = 1$.

Hence, the choice of the Γ parameter to minimize the number of involved flops is not straightforward. However, numerical experiments also in Section 8 suggest that the value lower or close to one works generally fine giving a balanced algorithm tuning.

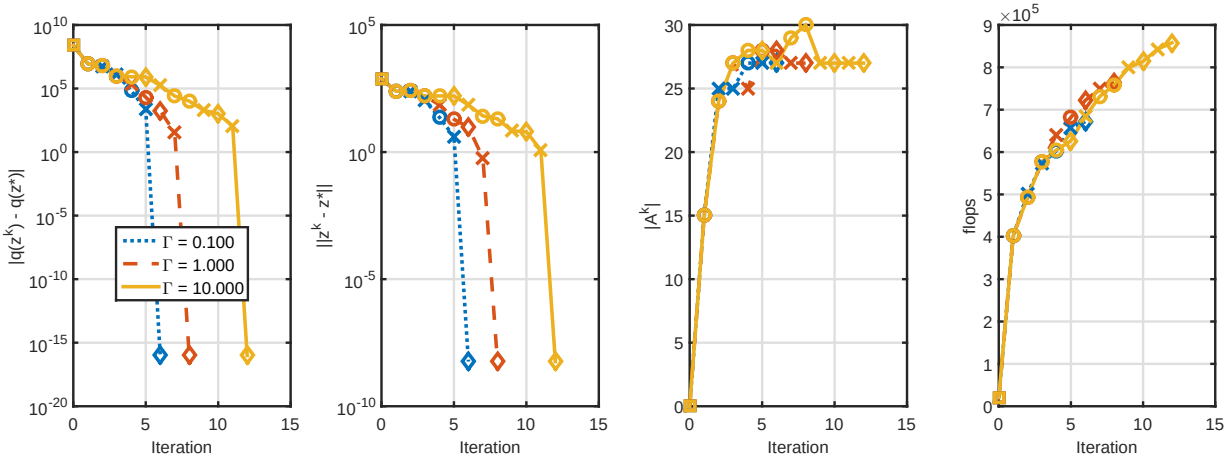


Figure 7.1: Comparison of convergence of NPP algorithm for several values of proportioning parameter Γ for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}}+\underline{\boldsymbol{z}})/2$.

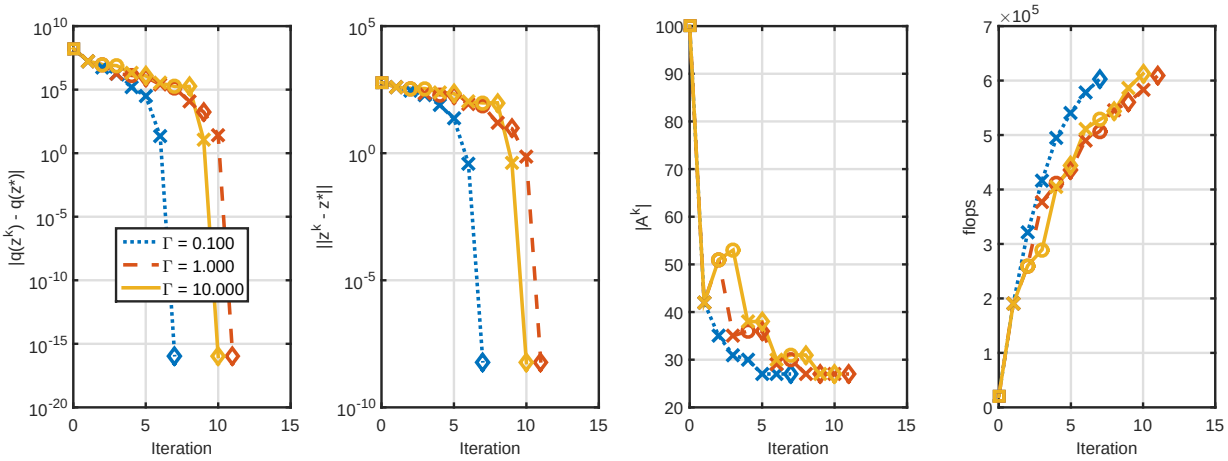


Figure 7.2: Comparison of convergence of NPP algorithm for several values of proportioning parameter Γ for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$.

7.3.2 Comparison to CGNP, PND and PNM Algorithms

All presented algorithms (CGNP, PND and NPP) are based on the projection of the Newton direction. The main difference between the algorithms is in the way how and when the blocking constraint indices are removed from the active set. To show the differences between the algorithms, their convergence is compared on the random QP problem. Moreover, the algorithm iterates are compared on the illustrative QP problem with $n = 2$ to show behavior under the specific circumstances.

To illustrate the difference of NPP algorithm to PNM of [11], PNM has been implemented in MATLAB® environment and included in the following tests. The PNM implementation uses Generalized-Armijo rule as described in [11]. We choose the Armijo rule parameters $\sigma = 1e-4$ and $\beta = 0.8$ since they lead to the best results in the test. To speed-up the face problem

solution, the Cholesky factor updates have been employed in a similar way how it is done for the proposed algorithms.

7.3.2.1 Random QP Problem

To show the effect of the algorithm differences, the QP problem of Section 5.4.1 has been solved with $\varepsilon = 1e{-}6\|\boldsymbol{h}\|$ and fixed $\alpha = 1.95\|\boldsymbol{H}\|^{-1}$ by the PNM, CGNP, PND and NPP algorithms. Further, the proportionality parameter of PND algorithm has been set to $\Gamma = 1$. Based on results from Section 7.3.1 we select $\Gamma = 0.1$ for the NPP algorithm.

Again two scenarios have been studied with varying the initial iterate. In the first setup, the algorithms have been started from the center of the feasible set as $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$. The comparison of the convergence of the algorithms is depicted in Figure 7.3. As the constraint indices can be added and removed from the active set at the same iteration for the NPP algorithm, only 6 iterations are involved compared to 10 and 15 iterations for PND and CGNP algorithm respectively. Comparing computational complexity, the PND and CGNP algorithms involve about 20 % and 39% more flops compared to the NPP algorithm. About 30% of the difference of the flops compared to the PND algorithm is due the fact that NPP algorithm does not execute the proportioning step by the projection of the chopped gradient. Therefore, the gradient can be updated when the proportioning is executed for NPP, whereas it has to be computed from the scratch in the PND algorithm. The rest is caused by the fact that NPP algorithm involves fewer iterations due to the faster convergence by the fact that the face problem is minimized at each iteration. When comparing the convergence of proposed methods to the PNM algorithm, it is visible that all converge in less number of iterations with a much fewer involved number of flops. Our observation is that the main reason for this is that the Generalized Armijo rule leads to adding of the too many new constraints which are later removed from the active set. This is visible, e.g., in the 5th iteration where the cardinality of the active set for PNM is higher than for all other algorithms. Also, as the removal of the blocking constraints is performed immediately (compare 4th iteration of NPP and PNM), the cost of the factor update is increased for PNM since the constraints are returned back to the active set.

In the second scenario, the initial iterate has been set to the upper bound, i.e., $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$, i.e., $\mathcal{A}^0 = \mathcal{I}$. The comparison of convergence of the proposed algorithms for such a setup is presented in Figure 7.4. It was already studied in Section 5.4.1 and 6.3.1 that for this example the opposite bounds are activated during the algorithm's iterations. As it was already mentioned, that this leads to the increase of the computational complexity of the factor updates in NPP algorithm since the blocking constraints indices are removed from the face active set and later added back. The consequence of such a phenomena is that NPP algorithm involves for about 17 % more flops compared to PND algorithm despite that the NPP converges in only 7 compared to 10 iterations for PND algorithm. This phenomenon can be observed on the steep slope of the cumulative flops for NPP compared to the both other algorithms. When comparing the convergence of PNM and NPP algorithms, it is visible that both generate very similar iterates at the beginning of iterations. The reason is that both algorithms have to perform the removal of the blocking constraints since the iterates are not proportional. Hence,

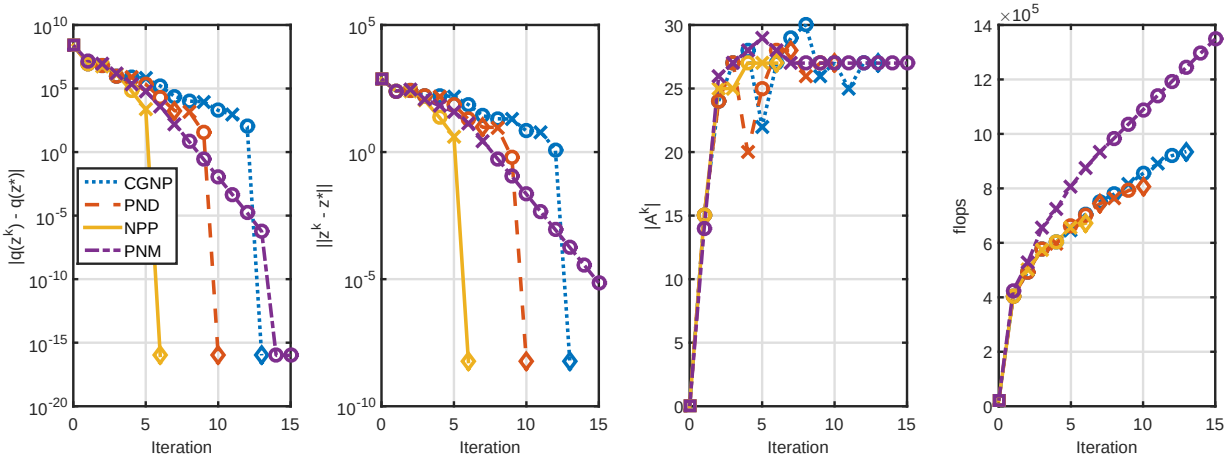


Figure 7.3: Comparison of convergence of CGNP, PND, NPP and PNM algorithms for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = (\overline{\boldsymbol{z}} + \underline{\boldsymbol{z}})/2$.

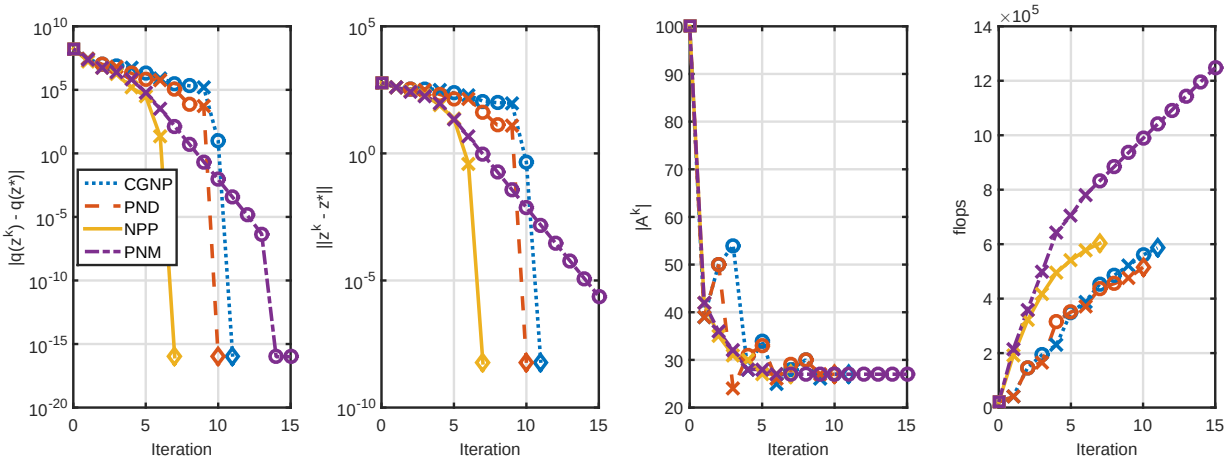


Figure 7.4: Comparison of convergence of CGNP, PND, NPP and PNM algorithms for example randomly generated QP problem with $n = 100$, and $\boldsymbol{z}^0 = \overline{\boldsymbol{z}}$.

no pure expansion step is performed in the NPP algorithm and the Armijo rule and the projected Newton direction path via PLS produce the similar results. The main difference is in the 7th iteration of NPP where PLS enables much faster convergence since the full Newton direction is applied to the solution of the face minimizer. On the other hand, PNM performs the Armijo rule with a limited step-size, leading to the fact that more iterations are involved.

To summarize the results so far, the NPP algorithm is superior for QP problems where opposite bounds are not activated during the iterations. When such event occurs it is better to use the PND algorithm which involves less number of flops compared to NPP algorithm since it can add the opposite bounds without change of the active set[4].

[4]It is expected that step-size of proportioning step of PND is such that activation of the opposite bounds is detected.

7.3.2.2 Illustrative QP Problems

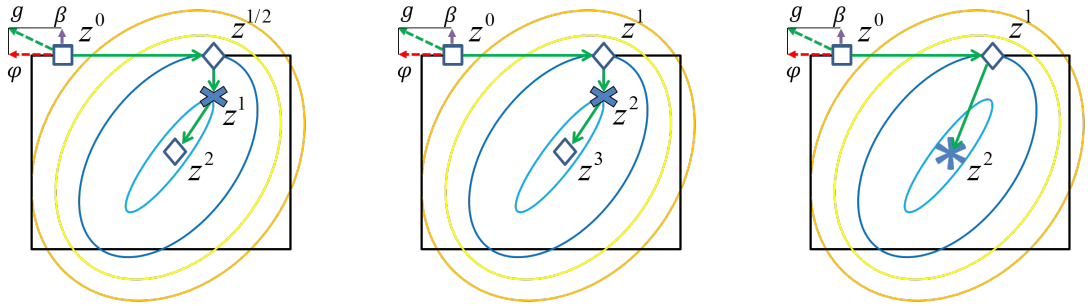
To show the difference of the proposed algorithms in more details, two illustrative QP problems with $n = 2$ have been setup. In the first setup, the QP problem from Section 6.3.3 has been used in two scenarios depending on the proportionality of the initial iterate for a fixed $\Gamma = 1$. The reason for such an analysis is to show how the algorithms differ in the behavior when iteration is proportional or not during the iteration. In the second setup, the phenomena of activation of opposite bounds discussed in the previous section are analyzed in more detail.

In the first case, the initial iterate $\boldsymbol{z}^0$ for illustrative QP problem of Section 6.3.3 has been generated to be proportional based on the test (6.1). The comparison of the iterates for such a case is depicted in Figure 7.5. All the proposed algorithms start by the computation of the Newton direction defined by the active set (diamond). Since the minimizer of the face is feasible, the whole Newton direction can be applied and all the algorithms move to the face minimizer. Next, while CGNP and PND algorithm executes the proportioning step, the NPP modifies the face active set based on the chopped gradient. Hence, it releases the blocking constraint and the application of the whole Newton direction with the proportioning (star) returns the QP problem solution. Hence, when compared to PND, the NPP algorithm saves one iteration. Note that in this setup, the NPP algorithm saved compared to PND also evaluation of one gradient since it can update the gradient after the application of the proportioning easily as described in Section 4.5.

In the second example presented in Figure 7.6, the initial iterate $\boldsymbol{z}^0$ for illustrative QP problem of Section 6.3.3 is generated as a non-proportional based on the test (6.1). Since CGNP algorithm does not distinguish the proportionality of the iterate it generates the same sequence of iterates as in the case for the initial proportional iterate. Since the iteration is detected to be non-proportional, the NPP algorithm releases the blocking constraints by the modification of the face active set concluding the iteration by the application of the whole Newton direction with the proportioning (star). PND algorithm firstly executes the proportioning step (cross) and the full Newton direction is applied (diamond) in the following iteration. Hence NPP algorithm again saved one iteration compared to the PND algorithm. Note that in this setup, the NPP algorithm saved compared to PND also evaluation of one gradient since it can update the gradient after the application of the proportioning easily as described in Section 4.5.

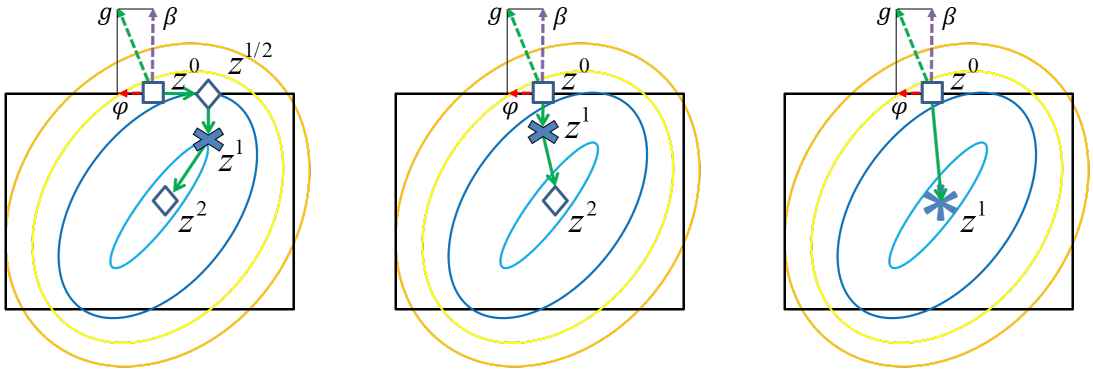
To study the phenomena of activation of the opposite bounds during the iterations for the proposed algorithms, the QP problem illustrated in Figure 7.7 has been setup. The initial iterate $\boldsymbol{z}^0$ is generated such that it is not proportional based on the test (6.1) for a given $\Gamma > 0$. Since CGNP does not detect the proportionality of the iterate it starts by the computation of the face minimizer followed by the proportioning step. Based on the assumption of sufficiently long convergent step-size, this step leads to the activation of the opposite bound. Since the active set remains the same[5], the face problem in the following iteration can be solved without any factor updates if the factor of the reduced Hessian has been computed for the initial iterate. Similar is executed for PND solver starting with the proportioning step. On the other hand, the face active set is modified for NPP algorithm releasing the blocking constraint index. Therefore the face problem is simplified to unconstrained optimization, leading to the need of the factor

[5]Remind the definition of the active set as $\mathcal{A} = \mathcal{L} \cup \mathcal{U}$.



(a) CGNP **(b)** PND **(c)** NPP

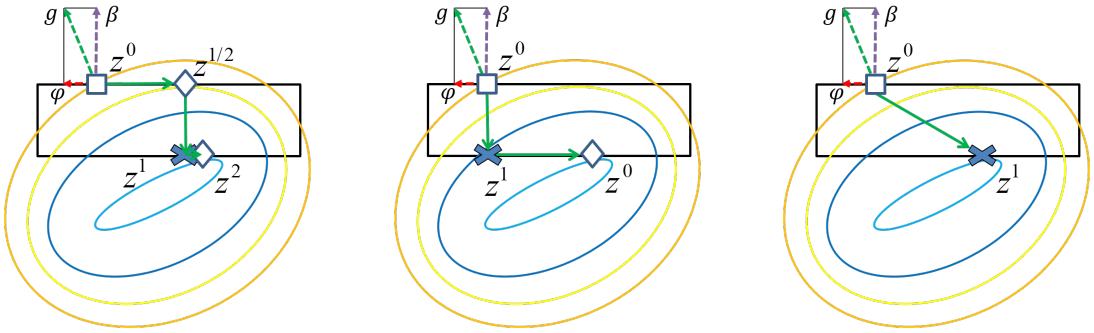
Figure 7.5: Comparison of iterations generated by CGNP, PND and NPP for ilustrative QP problem with proportional initial iteration.



(a) CGNP **(b)** PND **(c)** NPP

Figure 7.6: Comparison of iterations generated by CGNP, PND and NPP for ilustrative QP problem with non-proportional initial iteration.

update. The direction towards the unconstrained minimizer is used in the projected-Newton-direction path utilizing PLS leading to the solution in the first iteration. When comparing the active set of initial iterate and the solution, one can conclude that NPP made one unwanted change in the active set, hence increasing the computational complexity of the algorithm. Note that in the case of too short step-size of the proportioning step of CGNP and PND algorithms to activate the opposite bounds, the factor update would be also necessary for the mentioned algorithms leading to the superior computational complexity for NPP algorithm.



(a) CGNP **(b)** PND **(c)** NPP

Figure 7.7: Comparison of iterations generated by CGNP, PND and NPP for ilustrative QP problem with non-proportional initial iteration where solution lies on the opposite bound.

7.4 Summary

The main idea of the NPP algorithm is to remove the proportioning step in the PND algorithm via projection of the chopped gradient. Instead as a shortcut, the algorithm modifies the face problem to take into account only those active constraints for which the Lagrange multipliers are indicated to be optimal by the chopped gradient. This results in saving one additional iteration compared to the PND algorithm in the case of non-proportional iteration which leads to the feasible face minimizer. It also enables updating the gradient after each step compared to the PND where the gradient has to be recomputed from the scratch after the proportioning step.

A similar approach with the face modification was developed in the Projected Newton Method (PNM) of [11] without referencing to the chopped gradient definition. The main difference of the NPP and PNM algorithms is the proportionality test in the NPP algorithm. It prevents premature release of the blocking constraints from the active set compared to the PNM where they are removed immediately when they are detected to be not optimal by a wrong sign of the gradient component and potentially added back later. This prevention of premature release of constraints leads to the reduction of the computational cost of the update of the factors used for the face problem minimization.

The main benefit of the NPP algorithm is that norm of the QP problem Hessian does not need to be known for convergent step-size of the proportioning like in the PND algorithm. This is attractive for the QP problems where Hessian is subject to change: e.g., in nonlinear MPC or linear MPC with varying controller tuning. Compared to the PND algorithm, the NPP might involve a higher computational cost of the factor update when the opposite bound is activated. This is because the active set defining the face problem remains the same for PND but the face active set will change for NPP.

Chapter 8

Numerical Experiments

In this chapter, the performance of the proposed methods is compared with the state-of-the-art solvers in several experiments. In particular, we are referring to qpOASES, an online active set strategy, C++ implementation of [31] with hot start functionality used in all experiments together with the option for MPC problems; FiOrdOs [89], FGM with automatically generated C code; FORCES [25] automatically generated C code of IPM solver for MPC. All algorithms use default settings except the FiOrdOs, where the maximum number of iteration was set to 2000 and stopping condition to reach $\varepsilon_{\mathrm{FGM}}$-solution with respect to the optimal cost function value via so-called gradient map based stopping condition as proposed in [79]. The value $\varepsilon_{\mathrm{FGM}} = 1e{-}3$ was used, otherwise, no effort has been taken to optimize it to the experiments.

The presented algorithms were implemented in ANSI-C language using the single precision floating point arithmetic (4 bytes per floating point number) and were, as other solvers, called from MATLAB® environment via C-MEX interface with parameters set $\alpha = 1.95\|\boldsymbol{H}\|^{-1}$ and $\varepsilon = 1e{-}6\|\boldsymbol{h}\|$. The proportioning parameter of PND and NPP was set to $\Gamma = 1$.

All numerical experiments were executed on a laptop with Intel i7-4800MQ 2.7GHz processor utilizing only one core for all the computation. Since in embedded applications the solution has to be ready before the next sampling time the maximum computation time for each simulation was recorded. To eliminate outliers in the execution times caused by the other running programs on the computer, the solvers were called 50 times for each QP problem and the minimum time measured was accepted. Hence reported times are the results of max-min operation for each simulation if not stated otherwise. For the sake of completeness, we show also the mean computation times during the simulation.

The computation times of presented algorithms have been measured by the `QueryPerformanceCounter` C function which gives microsecond accuracy. The computation time of FORCES and qpOASES was provided by the compiled code of the available packages. Since FiOrdOs package does not produce the computation time, we have measured the MEX function call time by the MATLAB®'s `tic` and `toc` functions. All solvers have been using the warm start strategy using a shifted solution from the previous sample instant. Additionally, the CGNP, PND, NPP and qpOASES reused the factors from the previous sample instant.

The algorithms in the test are firstly compared on the benchmark problem of oscillating masses presented in [90]. Then their speed of convergence with respect to problem size and cardinality of an optimal active set is studied on the MPC problem with the randomly generated system. Further, the performance of algorithms is analyzed for the practical problem of linear control of the light-duty diesel engine air path. The final section provides the details of the comparison of the solvers for the nonlinear control of the heavy-duty diesel engine air path.

8.1 Oscillating Masses

The setup of the first experiment is similar to [90]. It consists of a sequence of six masses connected by the coil springs. The first and the last masses are connected to the walls. The weight of each mass is $m = 1$ kg, the spring constant is $k = 1$ N/m and there is no damping. There are three control inputs, i.e., $\boldsymbol{u} \in \mathbb{R}^3$, which exert the tensions between different masses. The setup is shown in Figure 8.1.

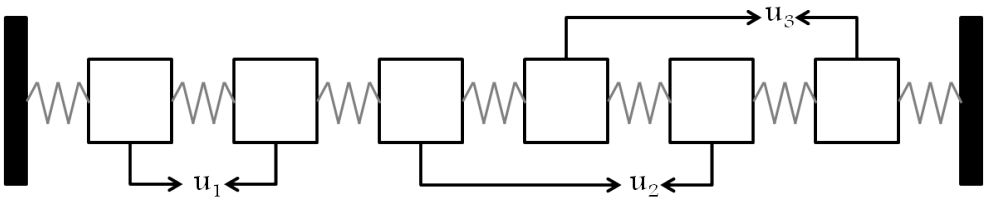
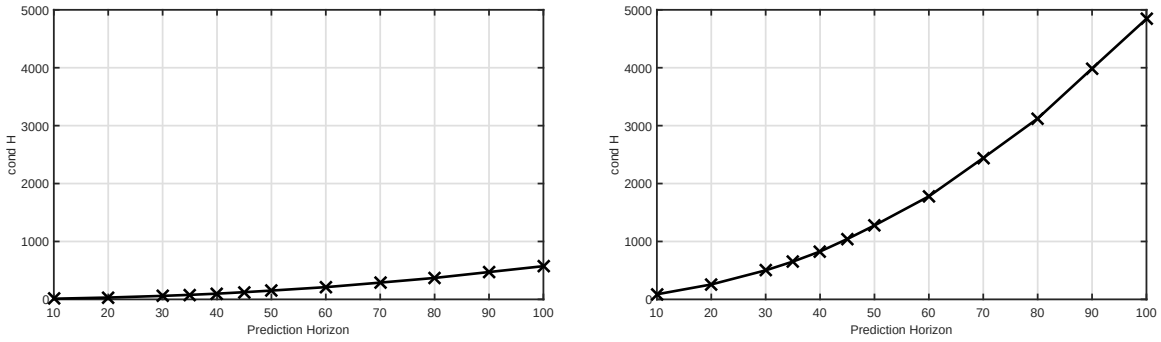


Figure 8.1: Oscillating mass model taken from [90]. Boxes represent the masses and dark regions on each side represent walls.

The setup can be described as continuous-time linear time invariant system

$$\begin{aligned}
m\,\ddot{s}_1 &= -ks_1 + k(s_2 - s_1) + u_1 \\
m\,\ddot{s}_2 &= -k(s_2 - s_1) + k(s_3 - s_2) - u_1 \\
m\,\ddot{s}_3 &= -k(s_3 - s_2) + k(s_4 - s_3) + u_2 \\
m\,\ddot{s}_4 &= -k(s_4 - s_3) + k(s_5 - s_4) + u_3 \\
m\,\ddot{s}_5 &= -k(s_5 - s_4) + k(s_6 - s_5) - u_2 \\
m\,\ddot{s}_6 &= -k(s_6 - s_5) - ks_6 - u_3.
\end{aligned} \tag{8.1}$$

The system state $\boldsymbol{x} \in \mathbb{R}^{12}$, $\boldsymbol{x} = \left[\boldsymbol{s}^T, \dot{\boldsymbol{s}}^T\right]^T$ then represents the displacement from the steady-state and the velocity of an individual mass. We assume the control limits $-0.5 \leq \boldsymbol{u} \leq 0.5$, and the presence of a random bounded external disturbance $\boldsymbol{w} \in \mathbb{R}^6$ with a uniform distribution on $[-0.5, 0.5]$ which acts additionally on the displacement state of each mass, see [90] for more details about the setup. The continuous-time system (8.1) has been discretized with the sampling time $T_s = 0.5$ s and its state space representation $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ has been found. The control objective is to stabilize each mass in its origin, i.e., to solve (2.3) with $\boldsymbol{R} = \boldsymbol{I}$ and $\boldsymbol{Q} = \boldsymbol{P} = \mu\boldsymbol{I}$, $\mu > 0$ at each sample time $t > 0$ with the new estimate of the current system



(a) $\mu = 1$ **(b)** $\mu = 1e3$

Figure 8.2: Dependency of the condition number of the QP problem Hessian as a function of prediction horizon for the oscillating masses for a different tuning $\boldsymbol{Q} = \mu \boldsymbol{I}$.

state $\tilde{\boldsymbol{x}}(t)$ along the simulation. Simulations with all solvers were carried out for 1000 seconds, i.e., 2000 sampling instants. The maximum and the average computation times during the simulation for a fixed prediction horizon were saved. The solution of the QP problem at each sample time has been repeated 50 times and the minimum time has been accepted to eliminate the outlines in the computation time due to the interrupts from the other running tasks on the computer.

In order to show the dependency of the performance of the proposed algorithms on the number of variables of the resulting QP problem we perform simulation of the model and controller with the prediction horizon $N \in \{10, 20, \ldots, 100\}$. The number of variables of associated QP problem is $N\, n_u$, i.e., $n \in \{30, 60, \ldots, 300\}$. Furthermore, to test the solver performance for worse-conditioned problems, we choose the two different controller tuning by changing μ: 1) *well-conditioned* with $\mu = 1$ where $\operatorname{cond} \boldsymbol{H} \approx 1e2$ and 2) *worse-conditioned* with $\mu = 1e3$ with $\operatorname{cond} \boldsymbol{H} \approx 1e3$. The growth in $\operatorname{cond} \boldsymbol{H}$ is caused only by the change of controller tuning. Note that we are referring to the conditioning of the QP problem in the sense of the spectral condition number of the $\boldsymbol{H}$. The dependency of the condition number of the QP problem Hessian on the prediction horizon is depicted in Figure 8.2.

To speed up the convergence of the proposed algorithms the solution from the previous sample time $\mathsf{U}^*(k-1) = \left[\boldsymbol{u}^\star(0)^T, \ldots, \boldsymbol{u}^\star(N-1)^T\right]^T$ was used as a warm start (initial iterate). We have used one sample shifted solution with keeping the last control move constant, i.e.

$$\mathsf{U}^0(k) = \begin{cases} \boldsymbol{u}^0(i) = \boldsymbol{u}^\star(i+1) & \text{for all } i = 0, \cdots, N-2, \\ \boldsymbol{u}^0(i) = \boldsymbol{u}^\star(N-1) & \text{otherwise} \end{cases},$$

although the proposed methods could be combined also with other variants of the warm start. One possible option is the result of Linear Quadratic Regulator (LQR) or, e.g., combination of shifted solution and LQR as described in [C.8]. Moreover, the factor of the reduced Hessian from the previous sample time has been used to speed up the factorization process when solving the face problem in the proposed algorithms.

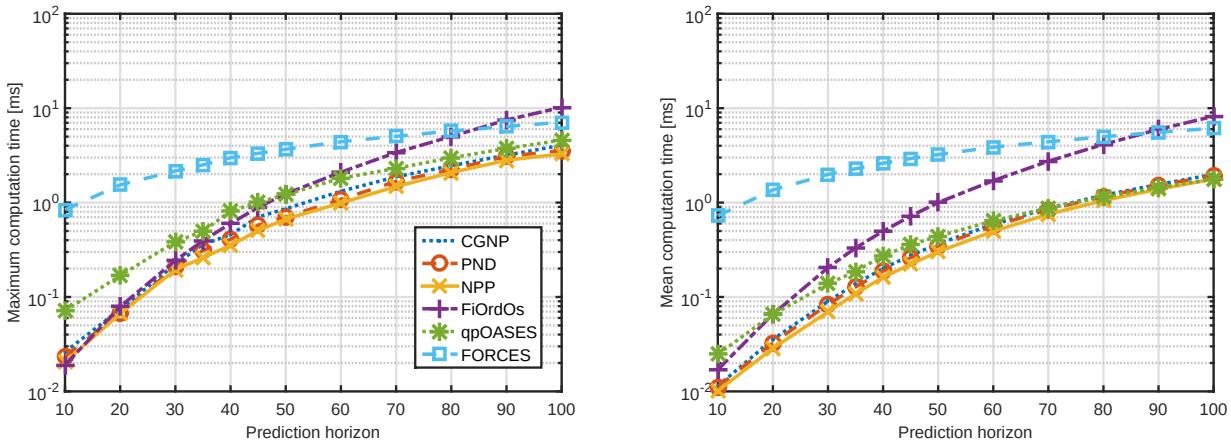


Figure 8.3: Maximum and mean computation time for well-conditioned oscillating masses.

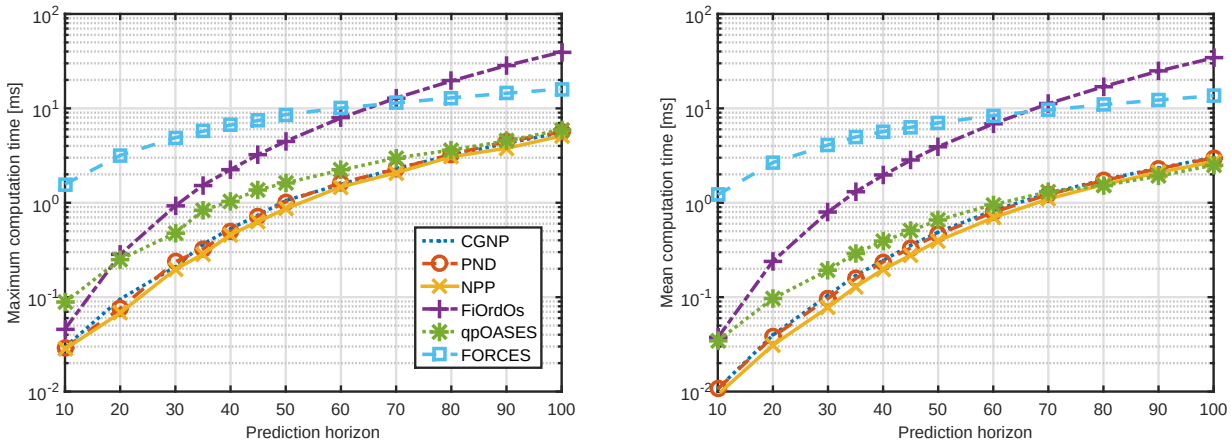
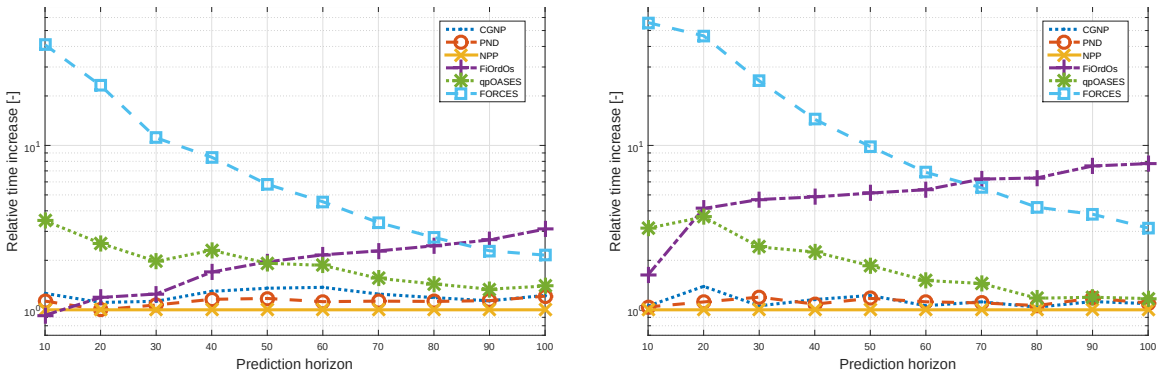


Figure 8.4: Maximum and mean computation time for worse-conditioned oscillating masses.

Both Figure 8.3 and Figure 8.4 show that for short to medium size of the prediction horizons all the proposed solvers are several times faster than FORCES, although it is the only solver in the test which has a linear computational complexity of each iteration with respect to the prediction horizon. To our understanding this is because the multiplicative constant of linear complexity is high; hence on a given prediction horizon range the complexity of each iteration is actually higher than for other solvers. It is also evident that FiOrdOs is very sensitive to the QP problem conditioning, leading to the fact that it is faster than the proposed algorithms (CGNP, PND and NPP) up to $N = 10$ for well conditioned QP problems ($\mu = 1$) but more than four times slower for worse-conditioned QP problems ($\mu = 1e3$) for $N \geq 20$. Moreover, as the condition number of $\boldsymbol{H}$ increases with prediction horizon (see Figure 8.2), the FiOrdOs is about eight times slower compared to NPP algorithm for $\mu = 1e3$ and $N = 100$.

On the other hand, the qpOASES and the proposed algorithms indicate small sensitivity to QP problem conditioning. However, compared to qpOASES, which needs as many iterations as the difference in the active set from one sample instant to the another, the projection in



(a) $\mu = 1$ **(b)** $\mu = 1e3$

Figure 8.5: Relative maximum computation time of solvers in test compared to NPP for oscillating masses.

the proposed algorithms helps to reduce the computation time when the working set changes dramatically from the previous sample time.

When comparing the performance of the algorithms, the least computation time is obtained for the NPP solver. This is valid for both the maximum and mean value of the computation times consistently for all the simulations. To be able to compare the relative performance of the solvers when compared to NPP for a particular prediction horizon N, a relative maximum computation time of solver $\mathrm{s} \in \mathcal{S}$,

$$\mathcal{S} = \{\mathrm{CGNP},\ \mathrm{PND},\ \mathrm{NPP},\ \mathrm{FiOrdOs},\ \mathrm{qpOASES},\ \mathrm{FORCES}\},$$

compared to the baseline solver NPP is defined as

$$r_{\mathrm{s},N} = \frac{\max t_{\mathrm{s},N}}{\max t_{\mathrm{NPP},N}}, \tag{8.2}$$

with $\max t_{s,N}$ being the maximum computation time of solver s for the prediction horizon N along the simulation.

The relative maximum computation times of other solvers is depicted in Figure 8.5. In average, the PND is about 20% slower and CGNP solver involves about 40 % more computation time than NPP for both well and worse conditioned case. This confirms the results from Section 7.3.2 where similar performance difference was observed in terms of flops. The relative increase of the computation time reduces for higher prediction horizons for the qpOASES solver. To the author's knowledge, this is because as the prediction horizon growths the number of QP problem variables rises leading to the more computational expensive factor update. Hence, the rapid change of the active set by the projection might lead to the higher computational complexity than a successive change of the active set by one optimal element only as it is done in qpOASES.

Figures 8.6 and 8.7 indicate the maximum and mean number of iterations for each solver to converge to the solution with a given accuracy over the simulation. The number of iterations of

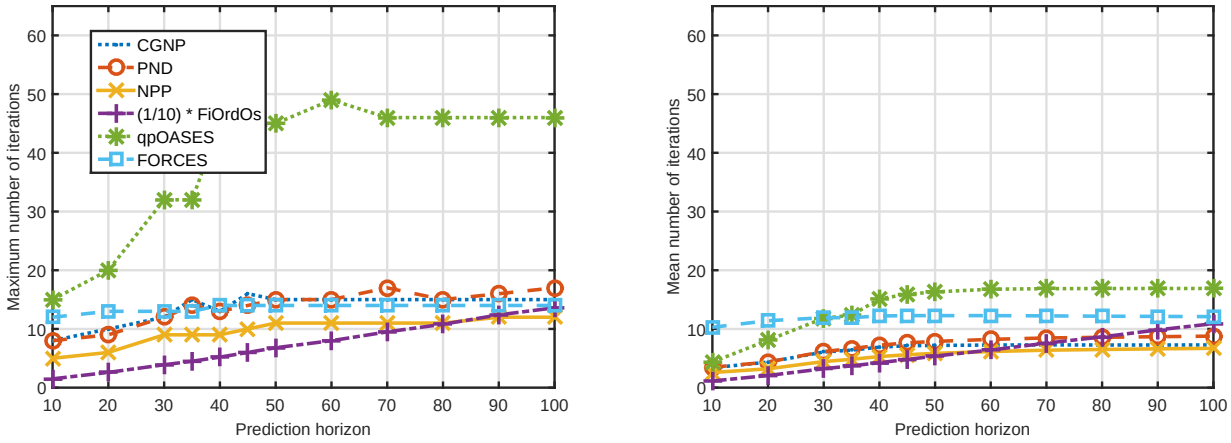


Figure 8.6: Maximum and mean number of iterations for well-conditioned oscillating masses.

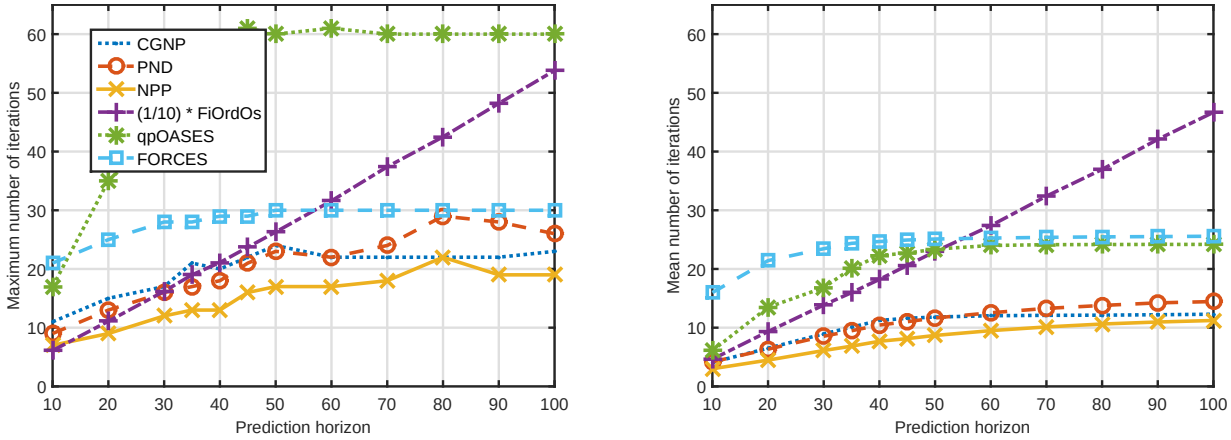
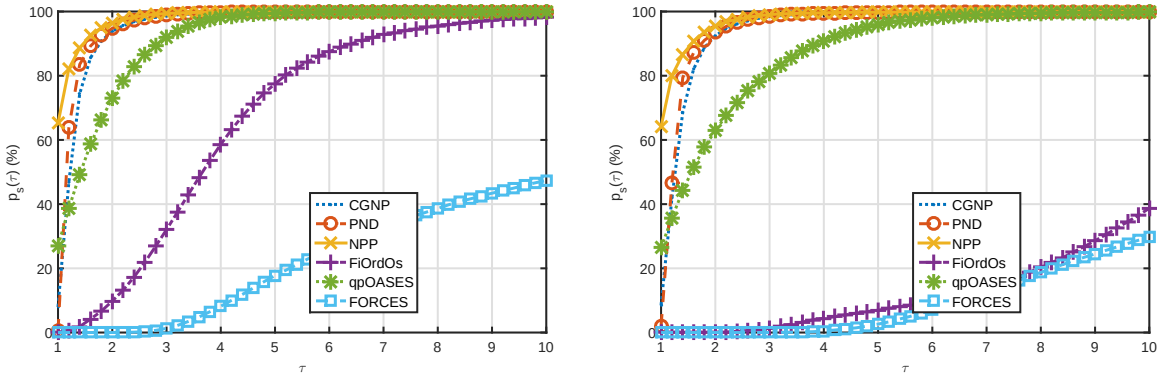


Figure 8.7: Maximum and mean number of iterations for worse-conditioned oscillating masses.

FiOrdOs has been divided by 10 for better overview when compared to the other solvers. Results indicate good scalability of the proposed algorithms in terms of the number of iterations for a different QP problem size. All proposed algorithms require less than 15 iterations in average with a maximum number of 29 iterations for PND, 24 for CGNP and 22 for the NPP algorithm for the worse-conditioned QP problem.

To compare the methods in more detail, we employ the standard Dolan-More performance profiles presented in [24]. These profiles visualize how many percents of all problem instances are solved within τ - factor times the time of the fastest solver for this instance. The computation times of each QP problem from the simulation with oscillating masses have been merged for all selection of prediction horizon N separately for each solver. There were exactly $2000 \cdot 10 = 20000$ QP problem instances solved by each solver and the particular choice of μ. Figure 8.8a shows that 65.2% of QP problem instances are solved by the NPP as the fastest solver and that 96.4% instances is solved by NPP solver within two times (i.e., $\tau = 2$) the fastest solver for well-conditioned oscillating masses. The similar performance of CGNP and PND solvers is



(a) $\mu = 1$ **(b)** $\mu = 1e3$

Figure 8.8: Performance profiles of [24] for computation times of solvers in test for oscillating masses with all settings of prediction horizon.

indicated by the fact that they solve 93.9% and 94.8% of test cases for $\tau = 2$. The qpOASES solves 73.0% of instances for $\tau = 2$, while FiOrdOs than solves 32.0% instances for $\tau = 3$. The poor performance of the FORCES solver is indicated by the fact that it solves only 47.4% instances within ten times the computation time of the fastest solver.

The sensitivity of the FiOrdOs to the QP problem conditioning is demonstrated in Figure 8.8b for rather worse-conditioned oscillating masses as it solves only 1.4% problem instances for $\tau = 3$. On the other hand, NPP solves 64.2% of problem instances with the lowest computation time and 95.5 % with $\tau = 2$. The qpOASES then solves 62.9% of problem instances for $\tau = 2$ showing only small sensitivity to the QP problem conditioning. The performance profiles of CGNP and PND solvers shows similar computation times and insensitivity to QP problem conditioning compared to the profiles for well conditioned oscillating masses in Figure 8.8a.

To compare the algorithms with respect to the changes in the optimal active set, the cardinality of optimal active set $|\mathcal{A}^*|$ and its absolute difference from the previous sample time $|\mathcal{A}(t)^*| - |\mathcal{A}(t-1)^*|$ were recorded during the simulation. This is illustrated in Figure 8.9 for time segment $t \in (520, 560)$ seconds and choice $N = 50$ and $\mu = 1e3$. It is evident that the computation time and the number of iterations are strongly correlated with the difference in the optimal active set for qpOASES. Observe, e.g., time segment $t \in (535, 550)$ seconds where an increase in the number of changes in active set leads to higher number of iterations, hence computation time. On the other hand, all proposed methods keep about the same iteration number and computation time for the same time segment. This is caused by the fact that the new active set can be found very rapidly via projection in the proposed methods.

8.2 Random System

The controlled system is often close to the steady state conditions where typically no constraints are activated. This is modified by either change of the setpoints, limits, or the effect of external

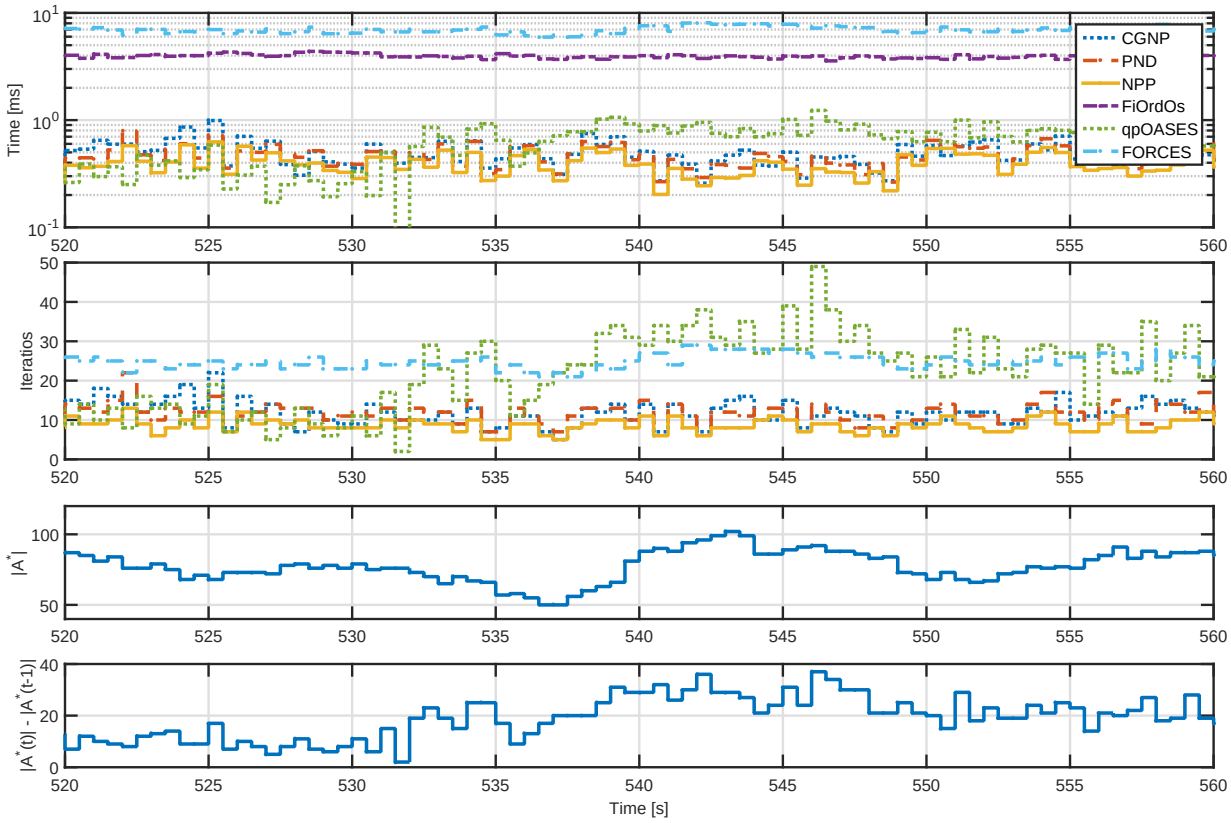


Figure 8.9: Comparison of computation times and number of iterations for oscillating masses for simulation time $t \in (520, 560)$ seconds and $N = 50$ with $\mu = 1e3$.

disturbances leading to transient operation of the MPC controller which tries to stabilize the system back. During the transient operation the constraints are activated as the actuators of the plant are saturated to speed up the stabilization process. This leads to the fact that the optimal active set has to be updated by the QP solver.

In the following experiment, the proposed method is compared to the state-of-the-art solvers to investigate how quickly the optimal active set is identified during the MPC controller transient operation when the external disturbance causes changes in the system state estimate $\tilde{\boldsymbol{x}}$ from the steady state where $\tilde{\boldsymbol{x}} = 0$. In the previous experiment, it was shown how the proposed method can use a solution from the previous sample time when there is the only slight change in $\tilde{\boldsymbol{x}}$. Contrarily, the experiment suggested in this section clarifies the behavior of the proposed method when the solution from previous sample time serves as poor initial iteration. The reason for that is a large change of the system state due to the external disturbance. As such, this experiment serves as a tool for estimating the worst case performance of the methods which can be expected during the transient operation of the MPC controller.

To this end, a random linear Schur stable[1] discrete time system with $n_x = 15$ and $n_u = 5$ was generated by Matlab's function `drss`. The control limits were set as $-0.1 \leq \boldsymbol{u} \leq 0.1$ and

[1]Schur stable system is such that all eigenvalues of $\boldsymbol{A}$ are within the open unit disk.

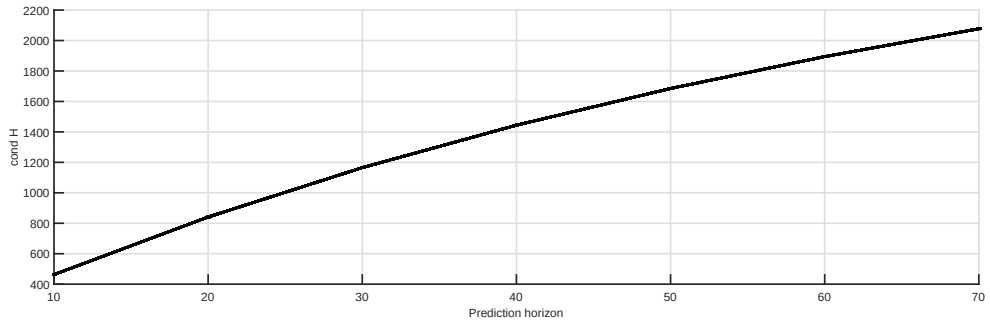


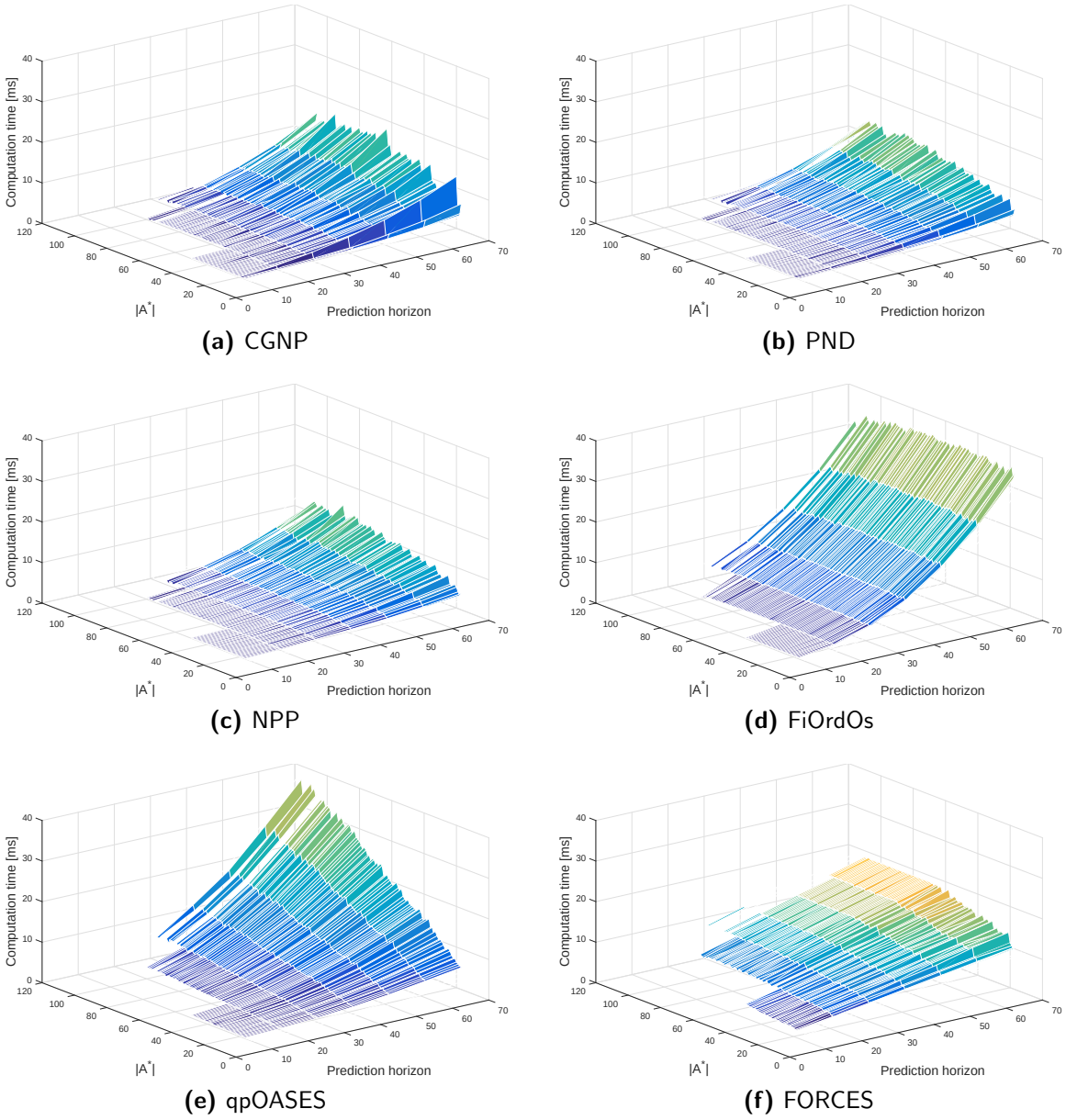
Figure 8.10: Condition number of QP problem Hessian for randomly generated system for increasing prediction horizon.

the controller was tuned with $\boldsymbol{R} = \boldsymbol{I}$, $\boldsymbol{Q} = 1e3 \cdot \boldsymbol{I}$ and $\boldsymbol{P}$ as solution of Lyapunov equation, see [8]. The controller has been set with enlarging prediction horizon $N \in \{10, 20, \ldots, 70\}$. The resulting QP problem conditioning has been $\mathrm{cond}\, \boldsymbol{H} \approx 1e3$. For each value of N, there were randomly generated 1000 current state estimates $\tilde{\boldsymbol{x}}$ where each component has a uniform distribution on $[-10, 10]$. The center of the feasible set has been used as an initial iteration for all generated $\tilde{\boldsymbol{x}}$, and all solvers were initialized before running at steady state condition $\tilde{\boldsymbol{x}} = \boldsymbol{0}$. The solution from the steady-state conditions have been used as an initial iteration for all solvers.
Hence the MPC problem (2.3) has been solved for a fixed N and for randomly generated $\tilde{\boldsymbol{x}}$ from uniform distribution $[-10, 10]$ and $\mathsf{U}^0 = \boldsymbol{0}$.

Figure 8.11 illustrates the maximal computation times[2] for the various state estimate $\tilde{\boldsymbol{x}}$ leading to the same cardinality of the optimal active set $|\mathcal{A}^*|$. It is evident that qpOASES needs more than twice the computation time compared to CGNP and almost four times more compared to PND and NPP algorithms for large $|\mathcal{A}^*|$. The reason is, contrary to proposed methods, the inability of qpOASES to change active set by more than one component at one iteration. As a result, qpOASES involves a relatively large number of iterations as indicated in Figure 8.12e. The FiOrdOs is 2-3 times slower compared to NPP and shows a large increase of the computation time with increasing of the prediction horizon. The reason is its sensitivity to the QP problem conditioning, see Figure 8.10. The FORCES algorithm shows persistently higher computation times compared to the NPP algorithm, especially for prediction horizons below 40 where it is more than 5 times slower independently on the cardinality of the optimal active set. Consistently to the results from the previous sections, the CGNP and PND algorithms involve about 20-40 % and 10-20 % more computation time respectively compared to the NPP algorithm.

Figure 8.12 illustrates the mean number of iterations of the tested solvers for the various state estimate $\tilde{\boldsymbol{x}}$ leading to the same cardinality of optimal active set $|\mathcal{A}^*|$. It was already mentioned that number of involved iterations growth proportionally to $|\mathcal{A}^*|$ for qpOASES, see Figure 8.11d. Since the conditioning of the QP problem (see Figure 8.10) the FiOrdOs solver

[2]The QP problem for each generated state estimate $\tilde{\boldsymbol{x}}$ was solved 50 times for each solver and minimum time was accepted.



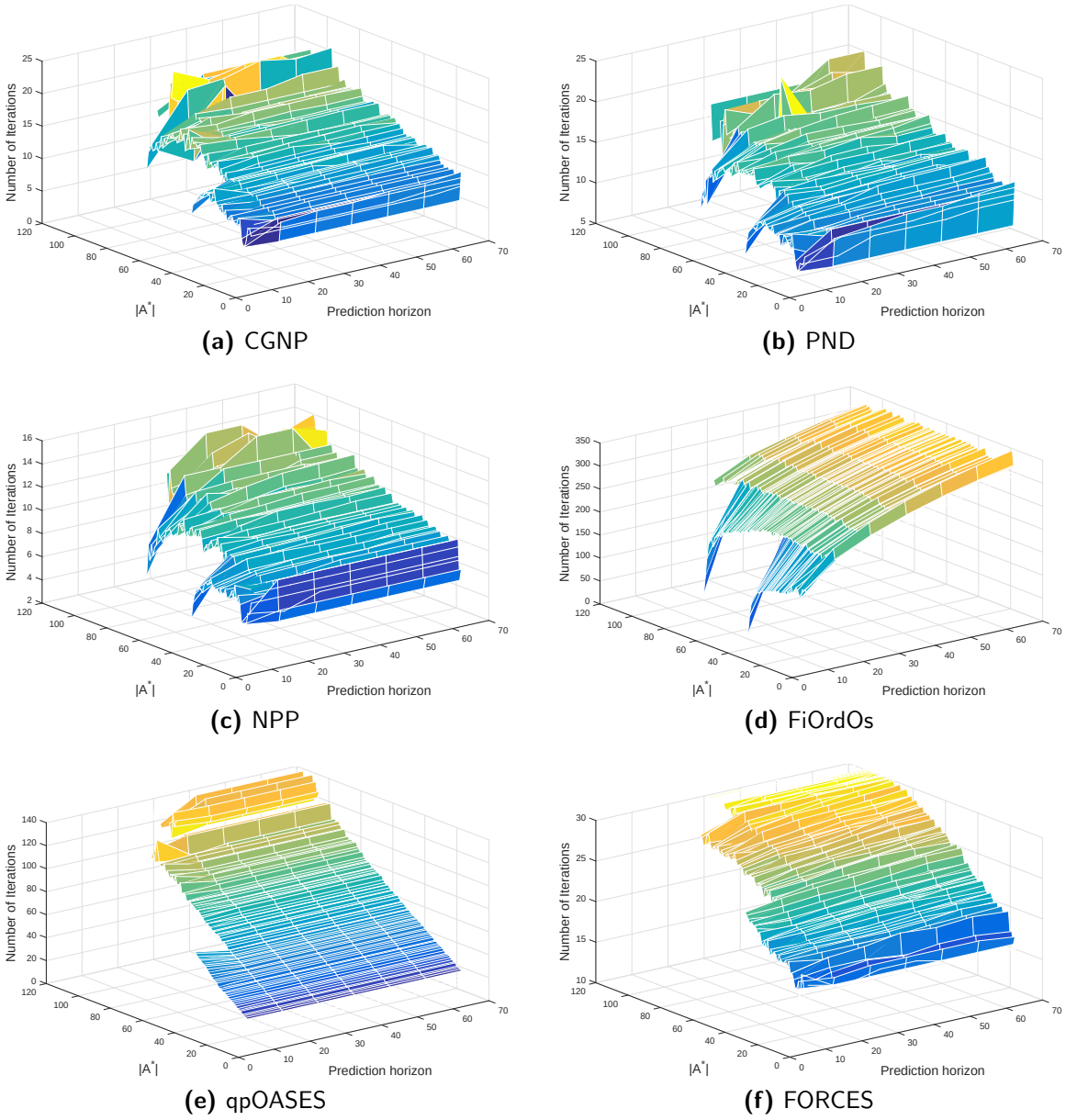
(a) CGNP **(b)** PND

(c) NPP **(d)** FiOrdOs

(e) qpOASES **(f)** FORCES

Figure 8.11: Comparison of maximal solver times for random system with variable state estimate $\tilde{x}$ leading to various cardinality of optimal active set $|\mathcal{A}^*|$.

involves many iterations to converge as indicated in Figure 8.12d. Figure 8.12f demonstrates that the number of involved iterations is relatively insensitive to the QP problem size and cardinality of the optimal active set for FORCES solver. The number of iterations is within 15 to 30 iterations. The similar can be observed for the proposed algorithms, where NPP algorithm involves up to 15 iterations in average and up to 20 iterations are needed for CGNP and PND algorithms in average to converge to the optimum.



(a) CGNP **(b)** PND

(c) NPP **(d)** FiOrdOs

(e) qpOASES **(f)** FORCES

Figure 8.12: Comparison of mean number of involved iterations for random system with variable state estimate $\tilde{\boldsymbol{x}}$ leading to various cardinality of optimal active set $|\mathcal{A}^*|$.

8.3 Diesel Engine Linear Control

To show the practical use of the proposed methods, the solvers were used for the solution of the MPC problem arising in the linear control of a turbo diesel engine. A nonlinear control-oriented model of four-cylinder turbocharged 2.2 liter diesel engine was built using the OnRAMP Design Suite tool [49] fitted to the real-time data collected on the test bench. A linear model at engine

speed 1500 rpm and injection quantity 40 mg/stroke was derived by the linearization inside the tool with resulting 23 states. An Exhaust Gas Recirculation (EGR) valve and Variable Geometry Turbocharger (VGT) valve were used as actuators. Similarly to [33], the Mass Air Flow (MAF) and the Manifold Absolute Pressure (MAP) were tracked to the references.

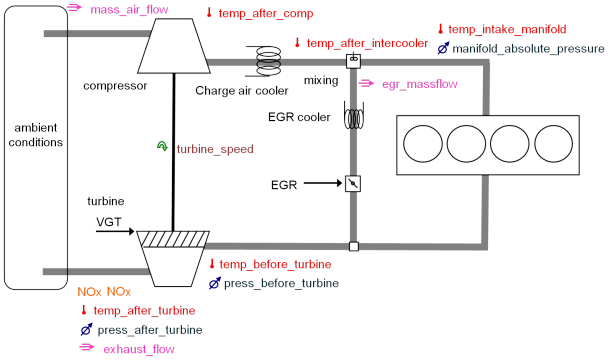


Figure 8.13: Turbocharged 2.2 liter diesel engine layout with sensor placement [49, Training material].

The layout of the engine together with the sensor placement is depicted in Figure 8.13. The fresh air coming from the ambient conditions is compressed by the compressor which is connected to the common shaft together with a turbine. The compressed air is cooled down by the charge air cooler to increase its density. The air is further mixed with the part of the exhaust gas recirculated through the EGR branch to decrease the oxygen concentration and increase of average specific heat capacity of gas mixture in the combustion chamber. This is done to reduce the formation of nitrogen oxides (NOx) which are dangerous for the environment and are subject to legal restrictions. Such a gas mixture is fed to the intake manifold and then to the engine cylinders where together with the injected fuel are burned creating power which drives the engine speed rotation. The mixture of the exhaust gas is from the exhaust manifold partly fed to the EGR branch cooled down by the EGR cooler to increase its density. The mass flow through the EGR branch is mainly controlled by the position of the EGR valve. The rest of the exhaust mixture drives the turbine where part of its enthalpy is transformed into the rotation energy of the turbine shaft, hence driving the compressor. The efficiency of the enthalpy transformation into the rotation energy of the turbine-compressor shaft is controlled by the variable geometry of the turbine housing which is affected by the VGT valve. See, e.g., [C.6] and references therein for more details about the control of the air path of the turbocharged diesel engines.
Hence, we designed an MPC controller for a tracking problem as (2.4) with system outputs $\boldsymbol{y} = [MAF, MAP]^T$ and control inputs $\boldsymbol{u} = [EGR, VGT]^T$. The form of the problem (2.4) assures zero tracking error in nominal conditions [56] and can be formulated as (1.1), see Section 2.1.1.2 for details.

The engine model was discretized with sampling period 50 ms and the prediction horizon was selected as five seconds. The controller was tuned with $\boldsymbol{R} = \text{diag}\,(0.1, 0.1)$ and $\boldsymbol{Q}_y = \text{diag}\,(1, 200)$. Additionally, to decrease the number of optimization variables to $n = N_{block} n_u$,

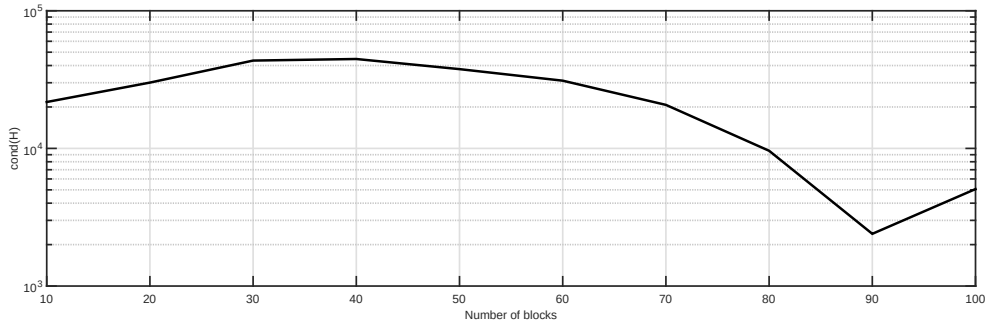
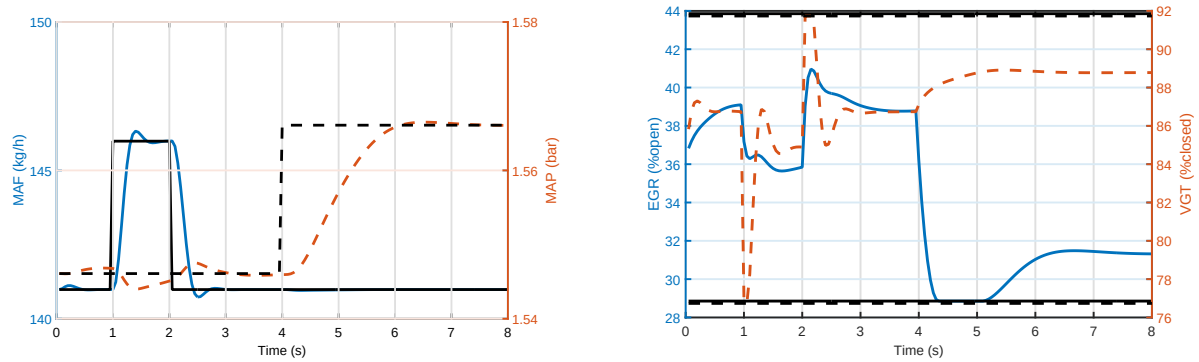


Figure 8.14: Condition number of Hessian of the QP problem in diesel engine tracking simulation for $N_{\text{block}} \in \{10, 20, \ldots, 100\}$.

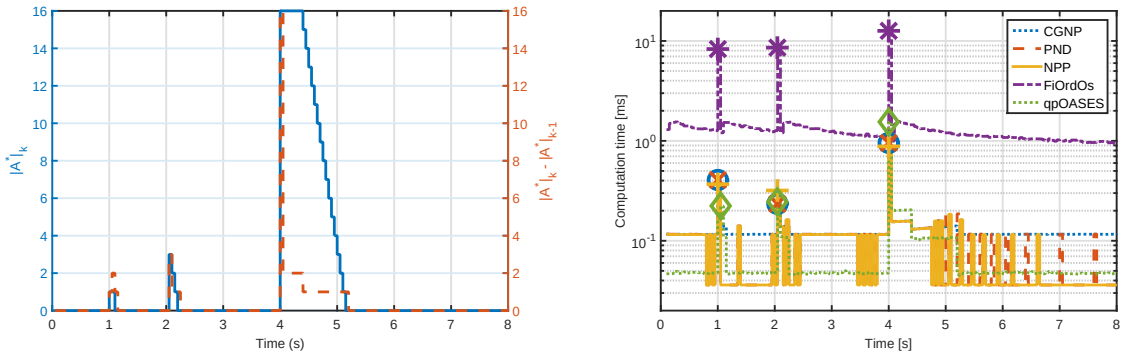
the input blocking strategy described in [20] was employed. It keeps the inputs constant over N_{block} time-steps, so-called blocks, while having usually a minor effect on the controller performance. In our experiments we set the length of first $(N_{block} - 1)$-th blocks as one sample and the last block has length $(N - N_{block} + 1)$. The condition number of QP problem to be solved as a function of the number of blocks is depicted in Figure 8.14.

To benchmark the solvers, practical trajectories of MAF and MAP references were set in the simulation with number of blocks as $N_{\text{block}} \in \{10, 20, \ldots, 100\}$. The limits for both actuators were selected as -10% and $+5\%$ from the linearization point. Step changes of 5 kg/h in MAF and 0.02 bar in MAP reference in Figure 8.15 led to the saturation of actuators, leading to the activation of new constraints, see Figure 8.16a. This results in an increase in the demanded number of iterations of all solvers which is evident from the difference of maximal and mean values of iterations in Figure 8.18 and solution times in Figure 8.17. While the number of iterations increases dramatically for solvers in the test, the maximum number of iterations of proposed methods remain under nine in all instances since the fast identification of optimal active set by projection is used. The NPP algorithm appears to involve the least computation time for all number of blocks. Hence NPP is more than 58 % faster in the worst solution time than qpOASES for $N_{\text{block}} \geq 30$ and more than 141 % faster for $N_{\text{block}} \leq 30$. The FiOrdOs algorithm involves more than 24 times computation time of NPP for $N_{\text{block}} \leq 70$ in the worst case. The speed of FiOrdOs is improved for a higher number of blocks since the drop in the condition number of the QP problem, see Figure 8.14. The sensitivity to the QP problem scaling is a well-known issue of all first-order methods [12]. Our further simulations indicate that when optimal diagonal preconditioner introduced in [76] is used, the number of involved iterations of FiOrdOs can be decreased to ≈ 100 in this experiment. Such an algorithm then involves two to three times higher computation time compared to NPP which solves not preconditioned QP problem. When comparing the proposed methods, both CGNP and PND algorithms involve about 10% more computation time compared to the NPP for all choice of the number of blocks. Since FORCES package cannot directly solve the problem with the cost function of type (2.4) it was not involved in this test.



(a) (b)

Figure 8.15: Diesel engine tracking problem simulation for $N_{\text{block}} = 100$. (a) The MAF and its setpoint are in solid line whereas MAP in dashed. (b) The EGR and its limits are in solid while VGT in dashed lines.



(a) (b)

Figure 8.16: Diesel engine tracking problem simulation for $N_{\text{block}} = 100$. (a) The number of active constraints in solid line and the number of changes in the optimal active set from last sample instant in dashed line. (b) The computation times during experiment. Markers show three largest computation times for each solver.

8.4 Diesel Engine Nonlinear Control

To demonstrate the performance of the NPP solver in the nonlinear MPC application, the solver has been integrated into the commercially available Honeywell Nonlinear MPC Framework [48]. The Nonlinear MPC Framework is a MATLAB® toolbox which implements a single shooting method solved by the Sequential Quadratic Programming (SQP) method for the general type of model and control setup. In SQP method, the quadratic model (1.1) of Lagrangian is constructed subject to the linearization of the constraints. As such, the QP problem Hessian is subject to change at each SQP iteration based on the result of model simulation and the construction of the cost function sensitivities, see, e.g., [65, Chapter 18], [23].

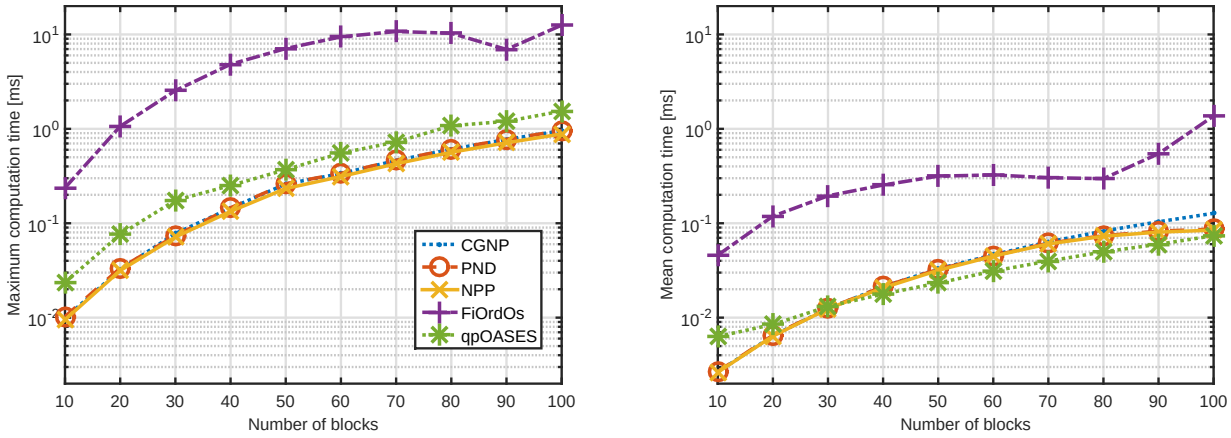


Figure 8.17: Maximum and mean computation time for engine tracking simulation for $N_{\text{block}} \in \{10, 20, \ldots, 100\}$.

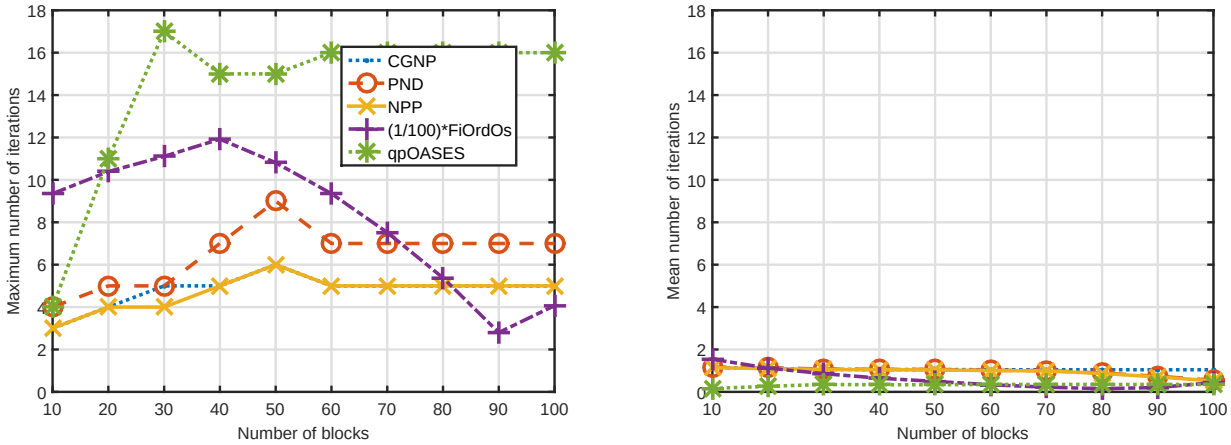


Figure 8.18: Maximum and mean number of involved iterations for engine tracking simulation for $N_{\text{block}} \in \{10, 20, \ldots, 100\}$.

For the purpose of solver performance comparison, we select the control of the air-path of a 12.8 liter heavy-duty turbocharged diesel engine in the so-called tip-in maneuver studied in [60]. This experiment starts with the engine in the close to the idle conditions and continues with the sudden increase of the torque demand to full load. Hence, the injected fuel is dramatically increased and the engine speed is ramping up as a vehicle is speeding up. Such experiment is challenging mainly because it exercises the engine over a set of rapidly changing operating points, which are far from steady-state equilibria [22]. Therefore, the linear MPC control which uses linearized models in a fixed operating points might have issues to deal with all physical constraints of the engine. The most important are the prevention of the turbocharger surge, turbo over speed and too high pressure in the intake manifold which might destroy the manifold, see [22] and reference therein for more details.

The layout of the engine used together with the sensor placement is depicted in Figure 8.19.

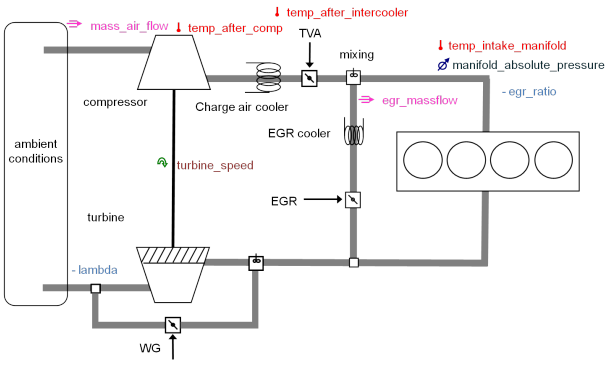


Figure 8.19: Turbocharged 12.8 liter TVA-EGR-WG diesel engine layout with sensor placement.

Similar to the EGR-VGT engine described in Section 8.3, the EGR valve serves as a tool to reduction of NOx. Moreover, the engine configuration uses the Throttle Valve Actuator (TVA) to create a pressure drop across it when less fresh air is needed or the air needs to be warmed up, mainly during the engine warm-up or during engine braking mode. To control the amount of energy from the exhaust gas, the turbine is bypassed by the wastegate branch with the Wastegate (WG) valve.

The control oriented mean value model[3] of the 12.8 liter turbocharged diesel engine was built from test cell data and turbocharger map in the Honeywell OnRAMP Design Suite [49]. The complexity of the model was further reduced in [60] by the elimination of all states except the turbo speed so that all other states and Controlled Variables (CVs) were expressed as static polynomials in the Manipulated Variables (MVs) $\boldsymbol{u}$, Disturbance Variables (DVs) $\boldsymbol{d}$ and turbo speed state $x = N_{\text{turbo}}$ as

$$\begin{aligned} \dot{x}(t) &= \Theta(x(t), \boldsymbol{u}(t), \boldsymbol{d}(t)) \\ \boldsymbol{y}(t) &= \Pi(x(t), \boldsymbol{u}(t), \boldsymbol{d}(t)), \end{aligned} \tag{8.3}$$

where $\Theta(x(t), \boldsymbol{u}(t), \boldsymbol{d}(t))$ and $\Pi(x(t), \boldsymbol{u}(t), \boldsymbol{d}(t))$ represent polynomial functions and

$$\boldsymbol{u} = [\text{EGR}, \text{WG}, \text{TVA}]^T, \qquad \boldsymbol{d} = [N_{\text{Eng}}, \text{IQ}, \text{SOI}]^T, \qquad \boldsymbol{y} = [N_{\text{turbo}}, \text{MAP}, \text{EGR}_{\text{ratio}}, \lambda]^T$$

with N_{Eng} [rpm], IQ [mg/stroke], and SOI [ATDC[4]] being the engine speed, injection quantity and advanced crank shaft angle of start of injection of the fuel. The EGR [% open], TVA [% open], WG [% open] are positions of EGR, TVA, and WG valves respectively; turbo speed N_{turbo} [krpm] and MAP [kPa] is absolute manifold pressure. The EGR ratio [%] and relative air fuel ratio λ [-] are defined as [47]

$$\text{EGR}_{\text{ratio}} = 100\frac{\dot{m}_{\text{EGR}}}{\dot{m}_{\text{EGR}} + \dot{m}_{\text{air}}}, \qquad \lambda = \frac{\dot{m}_{\text{air}}/\dot{m}_{\text{fuel}}}{\text{AFR}_{\text{stoich}}}$$

[3] Due the confidentiality reasons, all signals are scaled.
[4] ATDC means degrees After Top Dead Centre.

where $\dot{m}_{\mathrm{EGR}}$ and $\dot{m}_{\mathrm{air}}$ are mass flows through the EGR valve and the fresh air respectively; $\dot{m}_{\mathrm{fuel}}$ is the injected fuel flow and $\mathrm{AFR}_{\mathrm{stoich}} = 14.5$ is stoichiometric air-to-fuel ratio for a diesel fuel. It was shown in [66] that model in a form of (8.3) enables fast evaluation of Jacobians while still leading to good data fit both in the steady-state and transient operation. Furthermore, the model outputs were scaled internally for better numerical robustness.

The control strategy was selected to follow the setpoints of MAP and EGR ratio while respecting the actuators limits (0-100)%. The MAP and EGR ratio setpoints were given as a function of the engine speed and the injected fuel. There are only two setpoints but three available actuators are given (TVA, EGR, WG), hence one actuator was fixed to prevent the ill-conditioned control problem. In our case, this is done by attracting the TVA to a preferred position. Since partly closed TVA causes pumping losses, a fully open position was selected as the preferred position. Also, a partly opened WG causes inefficiency in the turbo, hence a fully closed position of WG was chosen as the preferred position with a low weight in the MPC controller cost function. To prevent the overspeed of the turbo, the maximum limit to the turbo speed was added into the optimization. Additionally to avoid the generation of excessive smoke, the minimum limit to the relative air-fuel ratio was added.

The control goals were transformed into the finite dimensional nonlinear MPC problem with time discretisation of actuators by zero-order hold with sampling period $T_s = 0.1$ s and prediction horizon $T = 2$ s ($N = T/T_s = 20$) as

$$
\begin{aligned}
\min_{\mathsf{U}} \int_0^T & (J_{\mathrm{CV}}(\tau) + J_{\mathrm{MV}}(\tau) + J_{\mathrm{dMV}}(\tau))d\tau \\
\text{s.t.} \quad & \dot{x}(t) && = \Theta(x(t), \boldsymbol{u}(t), \boldsymbol{d}(t)) \text{ for } t \in [0, T], \\
& \boldsymbol{y}(t) && = \Pi(x(t), \boldsymbol{u}(t), \boldsymbol{d}(t)), \text{ for } t \in [0, T], \\
& \lambda(t) && \geq \lambda_{\min}, \text{ for } t \in [0, T], \\
& N_{\mathrm{turbo}}(t) && \leq N_{\max}, \text{ for } t \in [0, T], \\
& \underline{\boldsymbol{u}} && \leq \boldsymbol{u}(kT_s) \leq \overline{\boldsymbol{u}}, \text{ for } k = 0, \ldots, N-1, \\
& N_{\mathrm{turbo}}(0) && = \tilde{N}_{\mathrm{turbo}},
\end{aligned}
$$

with $\mathsf{U} = \left[\boldsymbol{u}(0)^T, \boldsymbol{u}(T_s)^T, \ldots \boldsymbol{u}((N-1)T_s)^T\right]^T \in \mathbb{R}^{3 \cdot N}$, estimate of turbo speed $\tilde{N}_{\mathrm{turbo}}$ and

$$
\begin{aligned}
J_{\mathrm{CV}}(\tau) =& q_{\mathrm{EGR_{ratio}}} \|\mathrm{EGR_{ratio}}(\tau) - \mathrm{EGR}^{\mathrm{SP}}_{\mathrm{ratio}}\|^2 + q_{\mathrm{MAP}} \|\mathrm{MAP}(\tau) - \mathrm{MAP}^{\mathrm{SP}}\|^2, \\
J_{\mathrm{MV}}(\tau) =& r_{\mathrm{TVA}} \|\mathrm{TVA}(\tau) - \mathrm{TVA}^{\mathrm{SP}}\|^2 + r_{\mathrm{WG}} \|\mathrm{WG}(\tau) - \mathrm{WG}^{\mathrm{SP}}\|^2, \\
J_{\mathrm{dMV}}(\tau) =& r_{\mathrm{dTVA}} \|\mathrm{TVA}(\tau) - \mathrm{TVA}(\tau - T_s)\|^2 + r_{\mathrm{dEGR}} \|\mathrm{EGR}(\tau) - \mathrm{EGR}(\tau - T_s)\|^2 + \\
& + r_{\mathrm{dWG}} \|\mathrm{WG}(\tau) - \mathrm{WG}(\tau - T_s)\|^2.
\end{aligned}
$$

To assure feasibility of the solution, the limits of the relative air-to-fuel ratio and turbo speed were formulated as soft constraints. Each violation is penalized in the cost function by the quadratic term with an associated weight ($q_{\lambda_{\min}}$ and $q_{N_{\max}}$) as described in Section 2.1.1.3, hence two auxiliary (T/T_s)-vectors have been added and the following box constrained optimization

Table 8.1: Controller tuning overview of diesel engine air-path control.

CV weights	MV weights	MV movement weights	Constraints weights
$q_{\mathrm{EGR_{ratio}}} = 10$ $q_{\mathrm{MAP}} = 10$	$r_{\mathrm{TVA_{ratio}}} = 100$ $r_{\mathrm{WG}} = 1$	$r_{d\mathrm{TVA_{ratio}}} = 10$ $r_{d\mathrm{WG}} = 1$ $r_{d\mathrm{EGR}} = 1$	$q_{\lambda_{\min}} = 10$ $q_{N_{\max}} = 10$

Table 8.2: Overview of the setpoints and constraints of diesel engine air-path control.

CV setpoints	MV setpoints	CV limits	MV limits
$\mathrm{EGR}^{\mathrm{SP}}_{\mathrm{ratio}} = \Gamma(N_{\mathrm{Eng}}, \mathrm{IQ})$ $\mathrm{MAP}^{\mathrm{SP}} = \Delta(N_{\mathrm{Eng}}, \mathrm{IQ})$	$\mathrm{TVA}^{\mathrm{SP}} = 100$ $\mathrm{WG}^{\mathrm{SP}} = 0$	$N_{\max} = 102$ $\lambda_{\min} = 1.2$	$0 \leq$ TVA ≤ 100 $0 \leq$ EGR ≤ 100 $0 \leq$ WG ≤ 100

problem was obtained

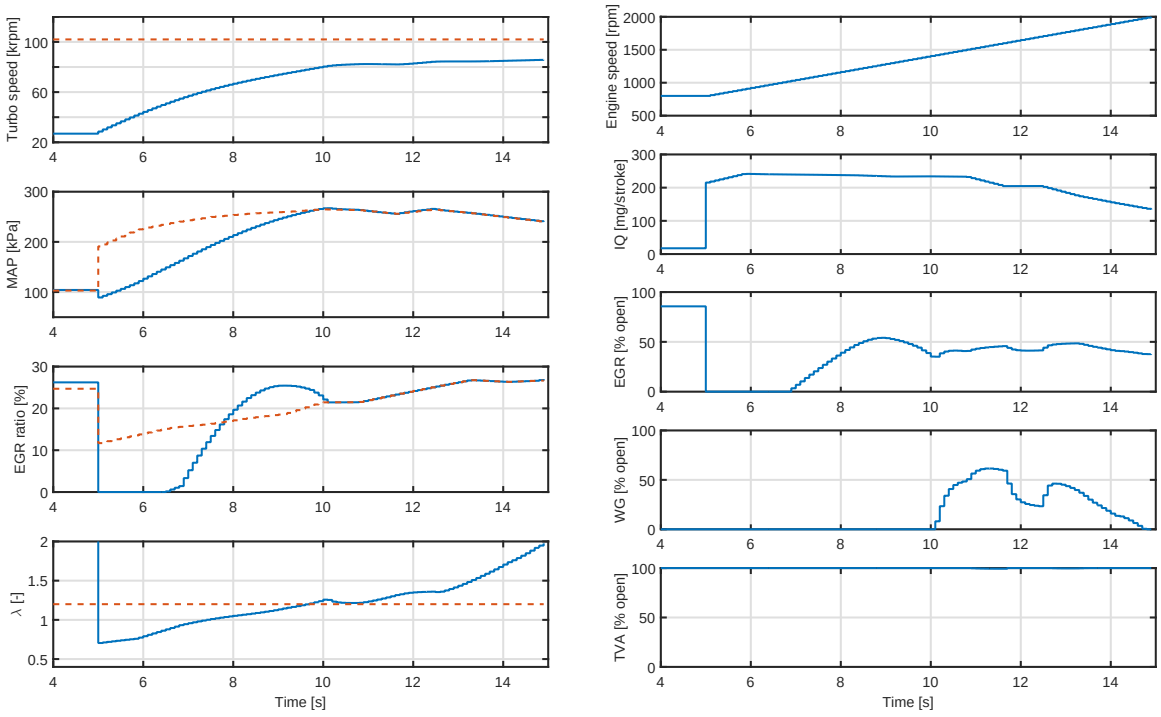
$$\begin{aligned}
\min_{\mathsf{Z}} \int_0^T & (J_{\mathrm{CV}}(\tau) + J_{\mathrm{MV}}(\tau) + J_{\mathrm{dMV}}(\tau) + J_{\lambda}(\tau) + J_{\mathrm{turbo}}(\tau))d\tau \\
\text{s.t.} \quad & \dot{x}(t) = \Theta(x(t), \boldsymbol{u}(t), \boldsymbol{d}(t)) \text{ for } t \in [0, T], \\
& \boldsymbol{y}(t) = \Pi(x(t), \boldsymbol{u}(t), \boldsymbol{d}(t)), \text{ for } t \in [0, T], \\
& z_{\lambda}(kT_s) \geq \lambda_{\min}, \text{ for } k = 1, \ldots, N, \\
& z_{\mathrm{turbo}}(kT_s) \leq N_{\max}, \text{ for } k = 1, \ldots, N, \\
& \underline{\boldsymbol{u}} \leq \boldsymbol{u}(kT_s) \leq \overline{\boldsymbol{u}}, \text{ for } k = 0, \ldots, N-1, \\
& N_{\mathrm{turbo}}(0) = \tilde{N}_{\mathrm{turbo}},
\end{aligned} \tag{8.4}$$

with $\mathsf{Z} = \left[\mathsf{U}^T, \boldsymbol{z}_{\lambda}^T, \boldsymbol{z}_{\mathrm{turbo}}^T\right]^T \in \mathbb{R}^{3 \cdot N + 2 \cdot N}$, $\boldsymbol{z}_{\lambda} = [z_{\lambda}(T_s), z_{\lambda}(2T_s), \ldots z_{\lambda}(T)]^T \in \mathbb{R}^N$, $\boldsymbol{z}_{\mathrm{turbo}} = [z_{\mathrm{turbo}}(T_s), z_{\mathrm{turbo}}(2T_s), \ldots z_{\mathrm{turbo}}(T)]^T \in \mathbb{R}^N$; $J_{\lambda}(\tau) = q_{\lambda_{\min}} \sum_{k=1}^{N} \|z_{\lambda}(kT_s) - \lambda(kT_s)\|^2$ and $J_{\lambda}(\tau) = q_{N_{\max}} \sum_{k=1}^{N} \|z_{\mathrm{turbo}(kT_s)} - N_{\mathrm{turbo}}(kT_s)\|^2$. Therefore, assuming the prediction horizon $N = T/T_s = 20$, the number of the optimization variables in optimization problem (8.4) is $3N + 2N = 100$.

The overview of the controller tuning, setpoints and limits is shown in Table 8.1 and Table 8.2, respectively. Tracking weights of both CVs have been tuned with the same importance while the weight on the movement of the TVA valve was selected to be high compared to MVs to prevent sudden action by the TVA which can lead to undesirable pumping losses and slow response. The weight of the relative air-to-fuel ratio limit was selected very high to assure that EGR valve would be closed and TVA fully open during the tip-in maneuver to speed up the increase of the MAP which is directly proportional to the delivered engine torque, therefore vehicle acceleration.

The goal of the controller during the tip-in maneuver is to deliver as fast increase of MAP as possible while assuring that the turbo will not be over-speeded and MAP signal will not grow too much to destroy the manifold and smoke peak generated will be minimized. The smoke is formed since there is not enough fresh air delivered by the compressor for the injected fuel as indicated by the drop of λ in Figure 8.20a at time 5 seconds. Hence, the controller has to

assure the fresh air increase by closing the EGR, fully opening the TVA valves and closing the WG bypass of the turbo to increase its speed to drive more air by the compressor as shown in Figure 8.20b. Hence the $\mathrm{EGR}_{\mathrm{ratio}}$ setpoint is not respected. Note, that in the situation of very low air-to-fuel ratio, such as at about time 5 seconds in Figure 8.20a, the smoke limiter would be activated leading to the fuel deration in practical applications. On the other hand, when MAP approaches the setpoint at about time 8.5 seconds, the controller has to stop its further increase to prevent the turbo over speed and damage of the manifold by opening the WG bypass and simultaneously adapt the EGR valve to control the $\mathrm{EGR}_{\mathrm{ratio}}$ to its setpoint. At the end of the experiment, the WG valve returns to the preferred position to maximize the flow coming directly to the turbo to drive the compressor.



(a) Outputs **(b)** Inputs

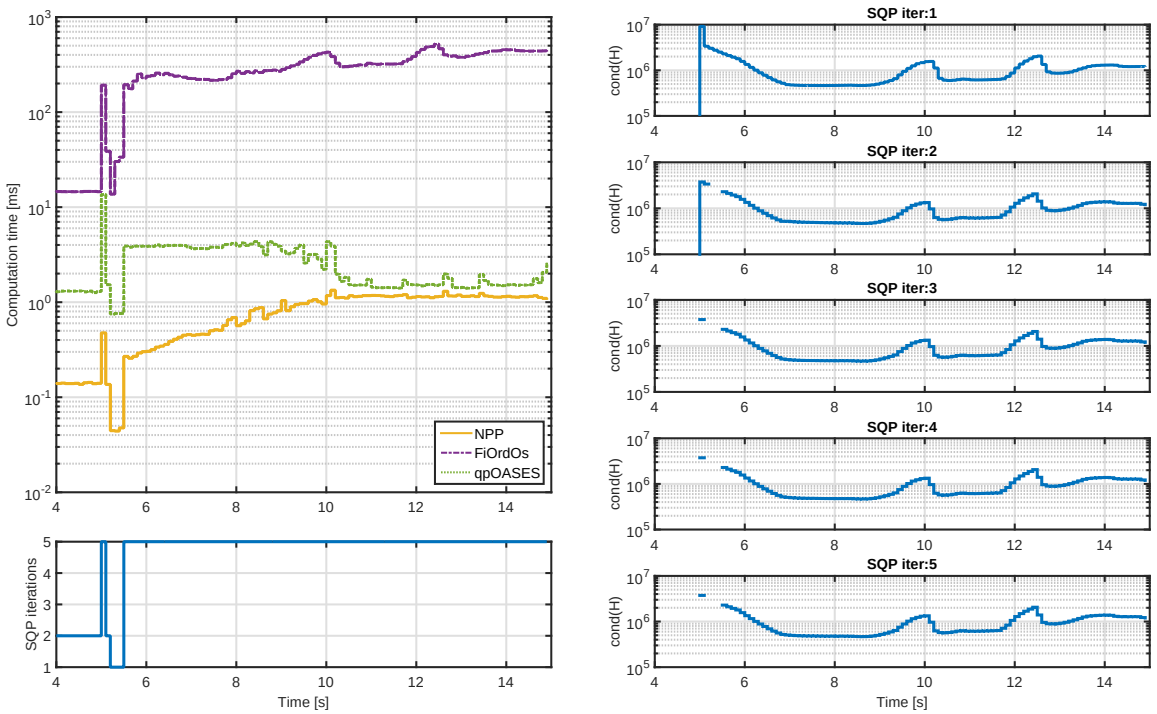
Figure 8.20: Time traces of the model outputs and inputs during the tip-in manoeuvre during the simulation with MPC controller. The limits and setpoints are plotted in red dashed lines, while the signals are in solid blue line.

The optimization problem (8.4) has to be solved at each sampling time. In the Nonlinear MPC Framework of [48] it is done by the SQP method where a succession of convex box constrained QP problems is solved based on the updated system linearization along the predicted inputs trajectory over the prediction horizon[5]. As a result, each QP problem in the succession

[5]Description of the linearization along the trajectory is out of the scope of this work. Hence, the reader is kindly referred to, e.g., [29] for more details.

differs also in the problem Hessian. The maximum number of SQP iterations, hence a maximum number of QP problems solved at each sampling period, was set to 5.

Since the NPP solver has shown the superior performance of the proposed methods, it has been compared to the state-of-the-art solvers. To be able to apply the FiOrdOs solver to the problem (8.4) the Hessian of the problem was selected as a parameter, and its eigenvalues were computed automatically at each solver call to set the convergent step-size. No pre-conditioner has been applied to the QP problem before the call of FiOrdOs. Its maximum number of iterations has been set to 20,000 per each QP problem, otherwise, the settings were kept the same as in previous benchmarks. The qpOASES solver utilized the extended online active set strategy introduced in [33] to hot-start the solution of QP problem even when Hessian was changed. Since the FORCES uses sparse MPC formulation it is not directly applicable to the problem of form (8.4), hence was not involved in this test. The zero vector was used as initial iterate for all solvers, except the qpOASES which use solution from the previous solved QP problem to hot start the algorithm.



(a) Computation Time & SQP iterations

(b) Condition number of Hessian of QP problem

Figure 8.21: Comparison of the sum of solution times in all SQP iterations for the solvers during the tip-in manoeuvre together with the condition number of the Hessian of the solved QP problem at particular SQP iteration.

The comparison of the sum of the computation times over all the performed SQP iterations for each solver in the test is depicted in Figure 8.21a together with the number of performed SQP iteration over the simulation, which was the same for all solvers. All solvers show a significant

increase of computation time at 5 seconds, where there is a step introduced in the injection quantity in Figure 8.20b. The reason is twofold: firstly many new constraints are activated due the fact that EGR has to close immediately; secondly, the condition number of the QP problem Hessian increases as there is a violation of the λ minimum limit. The projection enables fast identification of the new optimal active set for the NPP solver so that it involves maximally only 5 iterations per QP subproblem within SQP iterations. On the other hand, the qpOASES solver needs maximally 71 iterations to converge, see Figure 8.22. This causes a large difference in the average and maximum computation times for qpOASES. The ill-conditioning of the QP problem Hessian visible in Figure 8.21b causes a fact that FiOrdOs method involves thousands of iterations leading to the computation times up to 500 ms, hence higher than the sampling period. The sum of the computation times for all SQP iterations is under 1.34 ms for NPP solver while qpOASES involves maximally 13.6 ms to converge. The average computation time of the NPP is 0.6 ms while the qpOASES involves 2.3 ms and FiOrdOs 222.8 ms in average for this experiment. The computation time of the NPP is 30% betters when compared to qpOASES at the end of the transient (for $t \geq 10$ s). To the author's knowledge, this is mainly due to the cost of factorization of Hessian matrix at each SQP iteration since the number of iterations is one for both algorithms. While qpOASES has to perform factorization of full Hessian, the NPP performs the only factorization of reduced Hessian with lower computational complexity.

8.5 Summary

The performed numerical experiments confirm that all proposed methods converge to the solution with a given accuracy in a finite number of iterations. Compared to the first order method - FiOrdOs, it was observed that the average number of iterations is insensitive to the QP problem size and its conditioning. All proposed algorithms typically converge in less than 15 for NPP and less than 20 for CGNP and PND algorithms. When warm start is used, the number of iterations is further reduced. On the one hand, the similar insensitivity to the problem size also indicates IPM solver FORCES. On the other hand, as each iteration of this algorithm is computationally more expensive it is several times slower compared to the proposed methods for the problems in the test with the indication that for large problem size the benefit of the proposed method is reduced.

The important advantage of all proposed methods compared to qpOASES is the ability to identify the optimal active set quickly by the use of projected Newton direction path and proportioning. This reduces the maximum number of iterations, hence maximum computation time in case that active set has to change very rapidly. As observed in experiments, such situations occur very often in the applications of MPC where the rapid change of actuators is required with activation of actuator limits. As a consequence, all proposed methods are comparable and mostly superior to the state-of-the-art methods.

The best solution time performance (both in average and in maximum execution time) was delivered by the NPP algorithm. Moreover, it was shown that this algorithm is suitable also for application in the nonlinear MPC since it does not require the convergent step-size of the proportioning step as in CGNP and PND algorithms.

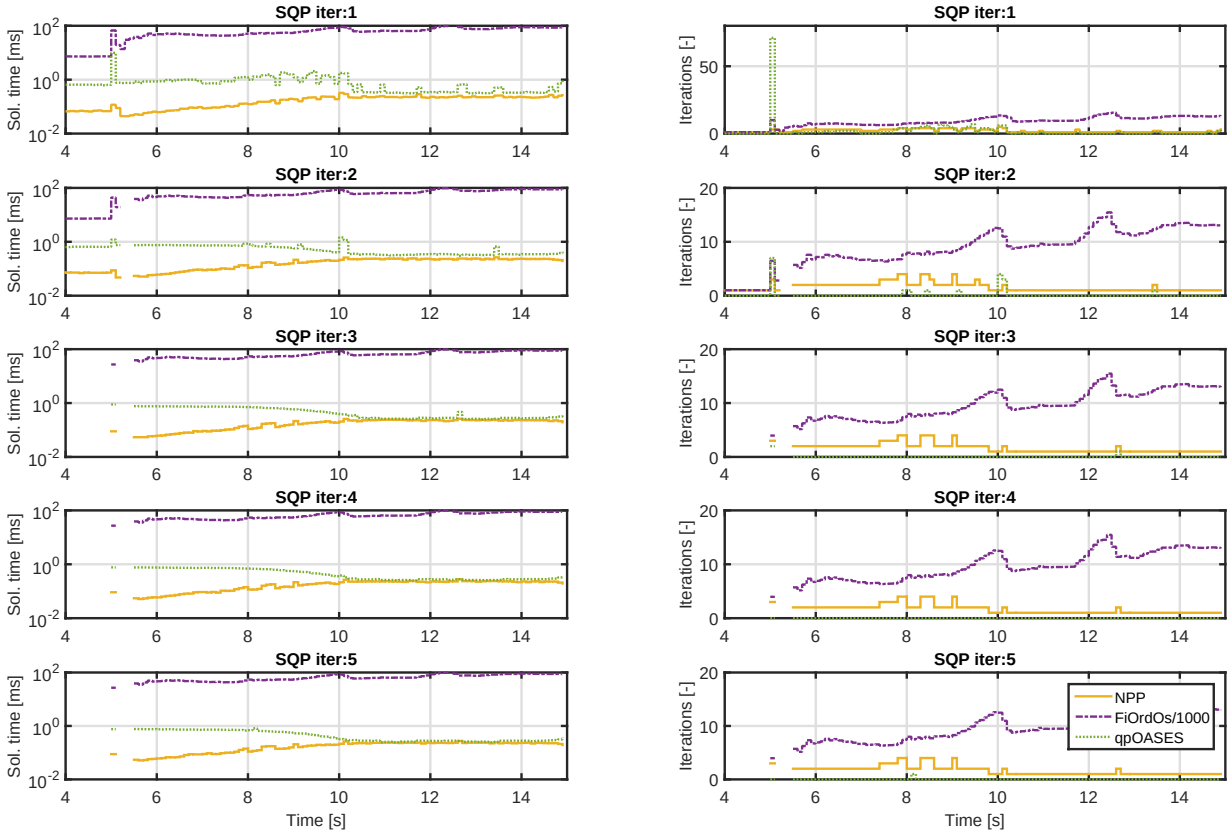


Figure 8.22: Comparison of the solution time and number of involved iterations of the solvers in particular SQP iteration during the tip-in manoeuvre. For better overview the number of iterations of FiOrdOs has been divided by 1000.

The memory requirements of the proposed algorithms can be divided into two parts. First, QP problem data can be stored in ROM in case of linear MPC or in RAM in nonlinear MPC applications. In our implementation, we exploit the symmetry of $\boldsymbol{H}$, hence only the upper triangular part, $0.5(n^2+n)+n$ floating point numbers, has to be stored. The proposed methods were implemented without any dynamic memory allocation. This is important for the embedded applications, e.g., in automotive, aearospace or mission critical areas, where the dynamic allocation is prohibited due to the embedded coding standards, e.g., [62]. The implementation of the proposed methods involves a static allocation of six temporary n-vectors and a matrix with n^2 floating numbers, which is used to speed up the process of the Cholesky factor update process using the lower triangular part of mentioned matrix as temporary space when a new constraint is added to the active set. In this case, the column of the factor is removed and the rest of the factor is moved to fill-in the gap in the column. The potential reduction of the RAM memory requirements to $0.5(n^2+n)+6n$ can be achieved by using a more complex indexing (pointer logic) so that there is no need to move the factor columns in the memory. A different approach to reduce the memory footprint is to modify the algorithm to work directly on the factored form of the QP problem Hessian, e.g., in LDL^T factorization, and

then perform a LDL^T factorization update procedure as described in [45]. Then it is possible to decrease the memory footprint of RAM + ROM to $n^2 + 6n$ floating point numbers.

Chapter 9

Conclusions

9.1 Summary

This work addressed development of algorithms suitable for solution of convex box constrained QP. It was shown in Chapter 2.1 that such a problem arises in the application of linear and nonlinear MPC, further the most common formulations of linear MPC were presented. It was shown that when dense formulation with ℓ_2 norm is used to formulate the individual control goals, the resulting optimization problem is a convex QP with dense Hessian matrix. The number of optimization variables then depends linearly on the prediction horizon. Furthermore, it was shown that the practical application of MPC can be formulated with box constraints when considering the output constraints as soft limits.

An overview of the existing methods for MPC was given in Chapter 2.2. It was shown that the QP problem arising in the MPC can be solved either off-line or online. The offline method, so-called explicit solution, is suitable only for small control problems with a limited number of dimension of the state vector due to the exponential growth of memory complexity of the online part of the algorithm. This bottleneck motivates the development of online methods. Most of them reduce the solution of a QP problem to a series of unconstrained problems which are solved either by iterative or direct methods. The methods include ASMs, where the main idea is to identify the active set at the solution in a finite number of iterations. The current active set defines the face problem which is solved by either direct solvers with factor updates or by Krylov subspace methods. ASMs are very efficient in practice, mainly due to their relative insensitivity to the QP problem conditioning. The well-known drawback of the ASMs is that if the initial active set is very different when compared to the optimal one, the algorithm needs many iterations to converge. On the other hand, IPMs involve relatively constant small number of iterations independently on the QP problem conditioning and size.

GPMs can identify the optimal active set quickly utilizing the projection, but their convergence is strongly influenced by the condition number of $\boldsymbol{H}$. This was improved by FGMs for which the rate of convergence is driven by the square root of the condition number of $\boldsymbol{H}$. Although the convergence of FGMs is better than for GPMs, the methods still involve a large number of low complexity iterations for ill-conditioned QP problems.

It was shown that the combined gradient / Newton projection algorithms which combine the GPM with the Newton methods offer relative insensitivity of the rate of convergence to the conditioning of $\boldsymbol{H}$. This insensitivity is given by utilizing the second order information when solving the face problem defined by the current active set at a cost of more complex iteration. The main bottleneck of these methods is that the active set changes are not under control in a sense that it is not defined exactly what part of the constraint indices and/or when the algorithm will add or release constraint indices from the active set. Therefore, it might happen that individual constraint index is removed from the active set and added back later in the following iteration or by the other part of the algorithm. It appears that this issue was solved in a mathematical community by the proportioning strategy. It was proposed for the solvers combining the conjugate gradients and projecting steps and further developed into the MPRGP algorithm where it enables to decide when the active set should be reduced/expanded. The proportioning strategy is based on the balancing of the violation of the KKT conditions on the active and the free set.

In chapter 3, the background and overview of the proposed algorithms for a solution of QP problems arising from MPC were presented. Similarly, as ASMs they try to find the optimal combination of indices of constraints in the optimum to solve the QP problem. All proposed methods minimize the face problem defined by the current active set (or face active set for the NPP algorithm) by the direct solver employing the Cholesky factorization with multiple rank update to speed up the computation. Hence, all proposed algorithms share the relative insensitivity of the number of iterations to the conditioning of $\boldsymbol{H}$. Furthermore, the proposed algorithms use projected Newton direction path via PLS routine as a tool for effective active set expansion. On the other hand, it was shown that the presented algorithms differ in the way how/when the active set is reduced.

The main shared building blocks of the presented algorithms were presented in Chapter 4. The procedure for m-rank update of the Cholesky factor for the face problem solution was introduced together with the update rule of the cost function gradient in $\mathcal{O}(n)$ flops. Additionally the PLS routine of [65] was extended to update also the cost function gradient in $\mathcal{O}(n)$ flops.

In Chapter 5 the CGNP algorithm was presented, and its convergence analyzed. The algorithm uses the PLS routine to expand the active set along the projected Newton direction path and proportioning step for its reduction. It enables to update the gradient in $\mathcal{O}(n)$ flops at each iteration except the case when the proportioning step is executed. The convergence of the CGNP algorithm was proved based on the simple estimate of the cost function decrease at the proportioning step when the algorithm moves to the face problem minimizer. The disadvantage of the CGNP algorithm is that it might block the cost function decrease since it does not take into account the value of Lagrange multipliers, and applies the active set reduction only in the case that the face problem minimizer is feasible.

It was shown that the cost function decrease was improved in the PND algorithm, presented in Chapter 6, by an introduction of the proportionality test. Based on this test, the active set reduction is executed by the proportioning step whenever the component of the gradient which corresponds to the active set dominates the violation of the KKT conditions. Otherwise, the PND algorithm uses projected Newton direction path for active set expansion. Hence the values of the Lagrange multipliers (components of the gradient) drive the decision when the

reduction/expansion of the active set should be performed. The convergence of the PND algorithm was proved based on the estimate of the cost function decrease at the proportioning step.

The NPP algorithm, presented in Chapter 7, removes the main two drawbacks of the PND algorithm, i.e., the need for convergent step-size and the inability to perform a computationally inexpensive update of gradient after the proportioning step. The cause of both bottlenecks is the proportioning step. Therefore, instead of proportioning step, the set defining the face problem is modified in the NPP algorithm whenever the component of the gradient that corresponds to the active set dominates the violation of the KKT conditions. Moreover, it was shown that this reduces the number of algorithm iterations compared to PND algorithm whenever the iteration is considered non-proportional by the proportioning test. On the other hand, the NPP algorithm might involve more computation burden for factor update compared to PND when opposite bounds are activated during the algorithm iterates. This is because the active set defining the face problem remains the same for PND but the face active set will change for NPP.

It was shown that the computational complexity of each iteration of all the proposed methods is $\mathcal{O}(n^2)$. The most computationally expensive part of the algorithms is the update of the Cholesky factor which involves $\mathcal{O}((n - m^k)^2)$ flops, where m^k is the number of constraint indices defining the face problem at kth iteration.

For the MPC application, it is important to note that all the proposed algorithms can be warm-started, i.e. the solution from the previous sample can be used to calculate the initial iterate to reduce the number of algorithm iterations. Moreover, when the Hessian of the QP problem remains fixed, the Cholesky factor from the previous sample time can be used to speed-up the face problem solution.

As was shown on numerical experiments in Chapter 8, contrary to the first order methods, the presented algorithms show relative insensitivity to the problem scaling[1] as the direct solver solves the face problem. Hence after a finite number of iterations, the algorithms eventually reduce to the unconstrained Newton method restricted on the subspace of active constraints which is solved by one iteration when the Newton step is applied. This together with the ability of fast identification of the optimal active set by the projected Newton direction path and the proportioning leads to the fact that all the proposed algorithms converge maximally in 15-20 relatively low complexity iterations independently on the initial iterate, QP problem size and conditioning, as was observed on all numerical examples. Further, when warm start is used, the number of iterations decreases to 5-10, to our observation. As a consequence, all proposed methods are comparable and mostly superior to the state-of-the-art methods. The best solution time performance (both in average and in maximum execution time) was observed for the NPP algorithm. Moreover, it was shown that this algorithm is suitable also for application in the nonlinear MPC since it does not require the convergent step-size of the proportioning step as in CGNP and PND algorithms.

[1] In terms of spectral condition number of QP problem Hessian

9.2 Contributions of the Dissertation Thesis

The main contributions of the thesis are summarized in the following list.

Tutorial results

- Overview of the linear MPC framework and its formulation as QP problem (Section 2.1).
- Overview of the existing methods used for a solution of an optimization problem of MPC (Section 2.2).

Preliminary algorithmic results

- The m-rank update procedure for the update of the Cholesky factorization for the face problem solution (Section 4.4).
- An effective approach of computing the update of the cost function gradient in $2m(n-m)$ flops (Section 4.5).
- Extension of the PLS algorithm of [65] with an effective update of the cost function gradient with complexity $\mathcal{O}(n)$ (Section 4.5).

Main algorithmic and theoretical results

- Introduction of CGNP algorithm together with convergence and computational complexity analysis (Chapter 5).
- Introduction of PND algorithm together with convergence and computational complexity analysis (Chapter 6).
- Introduction of NPP algorithm together with convergence and computational complexity analysis (Chapter 7).
- All the presented algorithms use the PLS routine in order to change rapidly the active set via projected-Newton-direction path. This leads to the low number of required iterations of all proposed algorithms independently on the quality of the initial iteration and the problem size. This feature improves the state-of-the-art active-set based strategies where the number of required iterations depends heavily on the number of needed changes in the active set.
- Thanks to the utilization of the second order information in the solution of the auxiliary face problem, all proposed algorithms show relative insensitivity to the QP problem scaling in terms of the conditioning of the problem Hessian.

- The PND and NPP algorithms represent an effective approach of utilization of the proportionality test to "look ahead" in order to reduce the number of unwanted changes in the active set, hence decreasing the computational complexity of the factor update scheme used for the solution of the auxiliary face problem.
- The unwanted blocking of the cost function decrease in the CGNP algorithms is removed by the active set reduction executed based on the proportionality test in the PND and NPP algorithms.
- The proportioning step in the CGNP and PND algorithms serves as a cheap way of the active set reduction. The requirement of convergent fixed stepsize and the inability to update cost function gradient after the proportioning step is removed in the NPP algorithm by the utilization of the modified face problem which is solved at each algorithm iteration.

Application results

- A comparison of introduced algorithms to the state-of-the-art methods on several benchmark problems (Chapter 8).

9.3 Future Work

The author of the dissertation thesis suggests to explore the following:

MPC problem approximation It is well-known that due to the effect of model inaccuracy and measurement noise it is not required to satisfy the equality constraints resulting from process dynamics exactly. When looking at the problem from the opposite side considering the previous statement, one can modify the original formulation of MPC and solve its approximation. The reason for this is the computationally less complex iterations leading from the exploitation of the modified QP problem structure. Such approximation of MPC was presented in [C.7] to establish the linear computational complexity of each iteration with the prediction horizon in connection with a preliminary version of the CGNP algorithm. This was achieved by the utilization of the sparse MPC formulation and equality constraints of the system evolution considered as soft constraints with ℓ_2 penalization of each violation. Hence the projection operation onto constraint set remains simple allowing to apply the proposed algorithms or their modifications to such approximation exploiting all the advantages of the sparse formulation.

Large-scale MPC problem approximation The solution of the face problem within the proposed methods can be substituted by the CG iterations and the resulting algorithm applied to the approximation of the MPC problem as presented in [C.3]. The approximation is based on the relaxation where the tracking error is considered in the cost function over the prediction horizon. Such approximation leads to the structured spectrum of the eigenvalues of $\boldsymbol{H}$. Structured eigenvalues spectrum then implies fast convergence of the CG iterations, hence reduction of

the computational complexity. This would be although more favorable for MPC problems with the higher number of variables, i.e., $n > 200$ and/or for the application with a very limited RAM.

Semi-explicit approach The proposed methods could be combined with a semi-explicit approach presented in [P.1] to completely or partly remove the most computationally complex part of the algorithms - factor update. The semi-explicit approach of [P.1] stores Cholesky factors of reduced Hessians for a selected combinations of active constraints. As an option some large factors can be stored only partly, i.e. only a few rows of them and the rest can be computed on-line. As a result, the method could trade-off between the available memory to store the partial factors and the associated computational effort to finalize the factorization. This approach is mainly targeted for MPC problems with the lower number of optimization variables due to the combinatorial nature of the possible combinations of active constraints.

Alternative solutions approaches Another possible way of research is to compute the first control move only. Since MPC is based on receding horizon concept only the first input move from the computed trajectory of inputs is actually applied to the process. Thus the others control moves are discarded, but they are needed to be computed in a traditional way. By computing the only the first control move or at least detection that its value will remain the same in the algorithm iterations, the computational burden would be drastically decreased.

9.4 Fulfillment of the Stated Goals

The fulfillment of the stated goals according to their formulations in Section 1.3 on page 5 is summarized below.

1. The goal of development of the general type of solver for the embedded application of MPC is satisfied in Chapters 5-7 by the introduction of the CGNP, PND and NPP algorithms. The suitability of the proposed methods for the embedded applications is indicated by the low number of the required iterations (typically <15-20), $\mathcal{O}(n^2)$ computational complexity of each iteration and n^2 floating point numbers memory requirements as shown in Chapter 8.

2. The fast convergence of the proposed methods independently on the QP problem scaling in terms of conditioning of the problem Hessian is obtained by the use of the second-order information in the solution of the auxiliary face problem defined by the current active set as shown in Section 4.3. The active set expansion by the PLS via projected-Newton-direction path and proportioning step offers a good scalability of the proposed algorithms regarding the QP problem size and relative insensitivity to the quality of the initial iterate as indicated by Chapter 8. The goal of the low complex iterations of the proposed methods is fulfilled by the utilization of the gradient update rule and by the modification of the PLS routine by the exploitation of the Newton direction and the face problem

structure as shown in Section 4.5 and 4.6. Furthermore, the computational complexity of $\mathcal{O}(n^2)$ per iteration is achieved by the m-rank update of the Cholesky factor introduced in Section 4.4 for the solution of the face problem. Moreover, the proportionality test, introduced in the PND and NPP algorithms, leads to the reduction of the computational complexity of the Cholesky factor update between the algorithm iterations, by avoiding the unwanted changes in the active set.

3. The goal of the comparison of the performance of the developed and state-of-the-art methods is completed in Chapter 8. It is shown that the proposed methods are comparable and mostly superior in the solution time when compared to the other solvers available in the literature on all numerical examples. The best solution time performance (both in average and in maximum execution time) was observed for the NPP algorithm.

4. Conclusions and possible future research are indicated in this chapter.

Bibliography

[1] A. Alessio and A. Bemporad. A survey on explicit model predictive control. In *Nonlinear model predictive control*, pages 345–369. Springer Berlin Heidelberg, 2009.

[2] B. Alkire. Cholesky factorization of augmented positive definite matrices. Technical report, Electrical Engineering Department, University of California, Los Angeles, 2002.

[3] 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*, pages 3057–3064. IEEE, Dec. 2008.

[4] D. Axehill and M. Morari. An alternative use of the Riccati recursion for efficient optimization. *Systems & Control Letters*, 61(1):37–40, Jan. 2012.

[5] R. Bartlett and L. T. Biegler. QPSchur: A dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming. *Optimization and Engineering*, 7(1):5–32, Mar. 2006.

[6] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming-the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, Dec. 2002.

[7] A. Bemporad, M. Morari, V. Dua, and E. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[8] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, Jan. 2002.

[9] 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*, 56(12):2883–2897, Dec. 2011.

[10] D. P. Bertsekas. On the Goldstein-Levitin-Polyak gradient projection method. *IEEE Transactions on Automatic Control*, 21(2):174–184, 1976.

[11] D. P. Bertsekas. Projected Newton Methods for Optimization Problems with Simple Constraints. *SIAM Journal on Control and Optimization*, 20(2):221–246, 1982.

[12] D. P. Bertsekas. *Nonlinear programming, Second Edition*. Athena Scientific, 1999.

[13] M. J. Best. An Algorithm for the Solution of the Parametric Quadratic Programming Problem. *Applied Mathematics and Parallel Computing*, pages 57–76, 1996.

[14] R. H. Bielschowsky, A. Friedlander, F. Gomes, J. Martınez, and M. Raydan. An adaptive algorithm for bound constrained quadratic minimization. *Investigación Operativa*, 7(1-2):67–102, 1997.

[15] F. Borrelli, M. Baotić, J. Pekar, and G. Stewart. On the computation of linear model predictive control laws. *Automatica*, 46(6):1035–1041, jun 2010.

[16] J. Bouchala, Z. Dostal, and P. Vodstrcil. Separable spherical constraints and the decrease of a quadratic function in the gradient projection step. *J. Optimization Theory and Applications*, 157(1):132–140, 2013.

[17] S. Boyd, N. Parikh, B. P. E Chu, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[18] S. Boyd and L. Vandenberghe. *Convex Optimization*, volume 98. Cambridge University Press, 2004.

[19] J. V. Burke and J. J. Moré. On the identification of active constraints. *SIAM Journal on Numerical Analysis*, 25(5):1197–1211, 1988.

[20] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6):563–570, July 2007.

[21] S. D. Cairano, M. Brand, and S. Bortoff. Projection-free parallel quadratic programming for linear model predictive control. *International Journal of Control*, 86(8):1367–1385, aug 2013.

[22] D. Cieslar, P. Dickinson, A. Darlington, K. Glover, and N. Collings. Model based approach to closed loop control of 1-D engine simulation models. *Control Engineering Practice*, 29:212–224, 2014.

[23] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*. Phd thesis, Ruprecht-Karls-Universität Heidelberg, 2001.

[24] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, Jan. 2002.

[25] A. Domahidi, A. U. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones. Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control. In *51st IEEE Conference on Decision and control*, pages 668–674, Maui, Hawaii, USA, 2012. IEEE.

[26] Z. Dostál. Box constrained quadratic programming with proportioning and projections. *SIAM Journal on Optimization*, 7(3):871–887, 1997.

[27] Z. Dostál. *Optimal Quadratic Programming Algorithms: With Applications to Variational Inequalities*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[28] Z. Dostál and J. Schöberl. Minimizing quadratic functions subject to bound constraints with the rate of convergence and finite termination. *Comput. Optim. Appl.*, 30(1):23–43, Jan. 2005.

[29] P. Falcone, M. Tufo, F. Borrelli, J. Asgari, and H. Tseng. A linear time varying model predictive control approach to the integrated vehicle dynamics control problem in autonomous systems. *2007 46th IEEE Conference on Decision and Control*, pages 2980–2985, 2007.

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

[31] H. J. Ferreau, C. Kirches, A. Potschka, H. G. Bock, and M. Diehl. qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):1–37, 2014.

[32] H. J. Ferreau, A. Kozma, and M. Diehl. A parallel active-set strategy to solve sparse parametric quadratic programs arising in MPC. *Nonlinear Model Predictive Control*, 4(1):74–79, 2012.

[33] H. J. 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*, 31(2):293–301, 2007.

[34] R. Fletcher. *Practical Methods of Optimization, 2nd Edition*. John Wiley & Sons, 2000.

[35] J. V. Frasch, M. Vukov, H. J. H. Ferreau, and M. Diehl. A new quadratic programming strategy for efficient sparsity exploitation in sqp-based nonlinear mpc and mhe. In *Proceedings of the 19th IFAC World Congress*, pages 2945–2950, Cape Town, South Africa, Aug. 2014.

[36] A. Friedlander and J. Martínez. On the maximization of a concave quadratic function with box constraints. *SIAM Journal on Optimization*, 4:177–192, 1994.

[37] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[38] P. E. Gill, N. Gould, and W. Murray. A weighted gram-schmidt method for convex quadratic programming. *Mathematical programming*, 30(2):176–195, 1984.

[39] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Procedures for optimization problems with a mixture of bounds and general linear constraints. *ACM Transactions on Mathematical Software (TOMS)*, 10(3):282, 1984.

[40] P. Giselsson. Execution time certification for gradient-based optimization in model predictive control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 3165–3170. IEEE, Dec. 2012.

[41] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical programming*, 27:1–33, 1983.

[42] A. Goldstein. Convex programming in Hilbert space. *Bulletin of the American Mathematical Society*, 70(5):709–710, 1964.

[43] G. Golub and C. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore and London, 1996.

[44] P. Grieder. Complexity reduction of receding horizon control. *IEEE Conference on Decision and Control*, 3(9-12):3179–3190, 2003.

[45] V. Havlena. Simple predictive LQ controller design. *Advanced Methods in Adaptive Control for Industrial Applications*, 158(2):137–148, 1993.

[46] V. Havlena and J. Findejs. Application of model predictive control to advanced combustion control. *Control Engineering Practice*, 13(6):671–680, jun 2005.

[47] J. Heywood. *Internal Combustion Engine Fundamentals*. Automotive technology series. McGraw-Hill, 1988.

[48] Honeywell. Nonlinear Model Predictive Control Framework;[user's Guide], 2015.

[49] Honeywell. OnRAMP Design Suite;[user's Guide], 2015.

[50] J. L. Jerez, P. J. Goulart, S. Richter, G. Constantinides, E. C. Kerrigan, and M. Morari. Embedded Online Optimization for Model Predictive Control at Megahertz Rates, Mar. 2013.

[51] J. L. Jerez, E. C. Kerrigan, and G. A. Constantinides. A sparse and condensed QP formulation for predictive control of LTI systems. *Automatica*, 48(5):999–1002, 2012.

[52] C. N. Jones and M. Morari. Polytopic Approximation of Explicit Model Predictive Controllers. *IEEE Transactions on Automatic Control*, 55(11):2542–2553, Nov. 2010.

[53] A. Karnik, D. Pachner, D. Fuxman, A.and Germann, M. Jankovic, and C. House. Model Predictive Control for Engine Powertrain Thermal Management Applications. In *SAE Technical Paper 2015-01-0336*. SAE, 2015.

[54] Y. Kim, G. Van Nieuwstadt, M.and Stewart, and J. Pekar. Model Predictive Control of DOC Temperature during DPF Regeneration. In *SAE Technical Paper 2014-01-1165*. SAE, 2014.

[55] E. Levitin and B. T. Polyak. Constrained minimization methods. *Zh. Vychisl. Mat. Mat. Fiz.*, 6(5):787–823, 1966.

[56] J. Maciejowski. *Predictive control with constraints*. Pearson Education, Harlow, UK, 2001.

[57] J. Mattingley and S. Boyd. CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, nov 2011.

[58] J. Mattingley, Y. Wang, and S. Boyd. Receding Horizon Control: Automatic generation of high-speed solvers. *IEEE Control Systems*, 31(3):52–65, 2011.

[59] D. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, June 2000.

[60] O. Mikulas. A framework for nonlinear model predictive control. Master's thesis, Czech Technical University in Prague, Prague, Czech Republic, January 2016.

[61] R. Milman and E. Davison. A Fast MPC Algorithm Using Nonfeasible Active Set Methods. *Journal of Optimization Theory and Applications*, 139(3):591–616, may 2008.

[62] MISRA C:2012 Guidelines for the use of the C language in critical systems. Standard, MISRA Consortium, Nuneaton, UK, Mar. 2013.

[63] J. Moré. Gradient projection techniques for large-scale optimization problems. In *Proceedings of the 28 th IEEE Conference on Decision and Control*, volume 13, pages 378–381, Tampa, Florida, 1989.

[64] Y. Nesterov. *Introductory lectures on convex optimization: A Basic Course*. Kluwer Academic Publishers, 2004.

[65] J. Nocedal and S. Wright. *Numerical optimization*. Springer Verlag, 1999.

[66] D. Pachner, L. Lansky, D. Germann, and M. Eigenmann. Fitting turbocharger maps with multidimensional rational functions. In *SAE Technical Paper 2015-01-1716*. SAE, 2015.

[67] G. Pannocchia, J. B. Rawlings, and S. J. Wright. Fast , large-scale model predictive control by partial enumeration. *Automatica*, 43:852 – 860, 2007.

[68] P. Patrinos and A. Bemporad. An Accelerated Dual Gradient-Projection Algorithm for Embedded Linear Model Predictive Control. *IEEE Transactions on Automatic Control*, 59:18–33, Jan. 2014.

[69] P. Patrinos, A. Guiggiani, and A. Bemporad. Fixed-point dual gradient projection for embedded model predictive control. In *Control Conference (ECC), 2013 European*, pages 3602–3607, Zurich, Switzerland, 2013.

[70] J. Pekar and G. Stewart. Using model predictive control to optimize variable trajectories and system control, Aug. 6 2013. US Patent 8,504,175.

[71] J. Pekar and G. Stewart. Using model predictive control to optimize variable trajectories and system control, 2013.

[72] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

[73] A. Raghunathan and S. Di Cairano. Optimal Step-Size Selection in Alternating Direction Method of Multipliers for Convex Quadratic Programs and Model Predictive Control. *21st International Symposium on Mathematical Theory of Networks and Systems*, pages 807–814, 2014.

[74] C. Rao, S. Wright, and J. Rawlings. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99(3):723–757, Mar. 1998.

[75] W. Rath. Fast Givens rotations for orthogonal similarity Transformations. *Numer. Math.*, 40:47–56, 1982.

[76] S. Richter. *Computational Complexity Certification of Gradient Methods for Real-Time Model Predictive Control*. PhD thesis, ETH Zurich, Zurich, Switzerland, Nov. 2012.

[77] S. Richter, C. N. Jones, and M. Morari. Certification aspects of the fast gradient method for solving the dual of parametric convex programs. *Mathematical Methods of Operations Research*, 77(3):305–321, Dec. 2012.

[78] S. Richter, C. N. Jones, and M. Morari. Computational Complexity Certification for Real-Time MPC With Input Constraints. *IEEE Transactions on Automatic Control*, 57(6):1391–1403, Jun. 2012.

[79] S. Richter and M. Morari. Stopping Criteria for First-Order Methods. Technical report, ETH Zurich, 2012.

[80] J. B. Rosen. Nonlinear programming. The gradient projection method. *Bull. Amer. Math. Soc.*, Abstract No. 80(63):25–26, 1957.

[81] J. Rossiter. *Model-Based Predictive Control: A Practical Approach*. CRC Press, 2003.

[82] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, second edition, 2003.

[83] K. Scheinberg. An efficient implementation of an active set method for SVMs. *The Journal of Machine Learning Research*, 7:2237–2257, 2006.

[84] A. Shahzad and P. J. Goulart. A new hot-start interior-point method for model predictive control. *IFAC Proceedings Volumes*, 44(1):2470 – 2475, 2011.

[85] D. Shell. A high-speed sorting procedure. *Communications of the ACM*, 2(7):30–32, 1959.

[86] L. E. Sokoler, G. Frison, M. S. Andersen, and J. B. Jorgensen. Input-constrained model predictive control via the alternating direction method of multipliers. *2014 European Control Conference (ECC)*, 115(2):115–120, 2014.

[87] G. Stewart. Matrix Algorithms: Basic Decompositions, Volume I. *Society for Industrial and Applied Math*, 4(1), 1998.

[88] T. H. Summers and J. Lygeros. Distributed model predictive consensus via the alternating direction method of multipliers. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 79–84, Oct. 2012.

[89] F. Ullmann. FiOrdOs: A Matlab toolbox for C-code generation for first order methods. Master thesis, Swiss Federal Institute of Technology Zurich, ETH, Zurich, 2011.

[90] Y. Wang and S. Boyd. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2010.

[91] Wikipedia. Shellsort, 2015.

[92] 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*, 56(7), 2011.

Publications of the Author

International Journals with Impact Factor

[J.1] O. Šantin, M. Jarošová, V. Havlena, Z. Dostál. Proportioning with second-order information for model predictive control. *Optimization Methods and Software*, doi: 10.1080/10556788.2016.1213840, 2016. Co-authorship: 40%

Reviewed International Journals

[J.2] O. Santin, J. Pekar, J. Beran, A. D´Amato, E. Ozatay, J. Michelini, S. Szwabowski, D. Filev. Cruise Controller with Fuel Optimization based on Adaptive Nonlinear Model Predictive Control. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, V125(7), also as SAE Technical Paper 2016-01-0155, 2016. Co-authorship: 12.5%

Patents

[P.1] J. Pekař, O. Šantin. Solver, system and method for solving quadratic programming problems with bound constraints utilizing a semi-explicit quadratic programming. US 2012/0059782, 2012. Co-authorship: 50%

[P.2] J. Pekař, O. Šantin. Configurable inferential sensor for vehicle control systems. US 2016/0125672, 2016. Co-authorship: 50%

International Conference Papers

[C.1] O. Šantin, V. Havlena. Combined Gradient and Newton Projection Quadratic Programming Solver for MPC. *18th IFAC World Congress*, Milano, Italy, 5567–5572, 2011.

Co-authorship: 50%

[C.2] O. Šantin, V. Havlena. Gradient Projection Based Algorithm for Large Scale Real Time Model Predictive Control. *23rd Chinese Control and Decision Conference*, Mianyang, China, 3906-3911, 2011. Co-authorship: 50%

[C.3] O. Šantin, V. Havlena. Combined Partial Conjugate Gradient and Gradient Projection solver for MPC. *IEEE Multi-Conference on Systems and Control*, Denver, USA, 1270-1275, 2011. Co-authorship: 50%

Cited by:

- M.A.C Fernandes. Linear Programming Applied to Blind Signal Equalization. *International Journal of Electronics and Communications* 69(1): 408-417, 2014, doi:10.1016/j.aeue.2014.10.017.
- M. Konnik. Online Constrained Receding Horizon Control for Astronomical Adaptive Optics. *PhD Thesis*, University of Newcastle, Australia, 2013.

[C.4] O. Šantin, O. Mikuláš, D. Pachner, M. Herceg, J. Pekař. Towards ECU-Ready Nonlinear Model Predictive Control: Tip-in Maneuver Case Study. *54th IEEE Conference on Decision and Control*, Las Vegas, USA, accepted, 2016. Co-authorship: 20%

[C.5] O. Šantin, J. Pekař. Application of Newton Projection with Proportioning Solver to Automotive. *4th European Conference on Computational Optimization*, Lueven, Belgium, accepted, 2016. Co-authorship: 80%

[C.6] N. Khaled, J. Pekar, A. Fuxman, M. Cunningham, O. Santin. Multivariable Control of Dual Loop EGR Diesel Engine with a Variable Geometry Turbo. *SAE Technical Paper 2014-01-1357*, 2014, doi:10.4271/2014-01-1357. Co-authorship: 20%

Cited by:

- V. Alfieri, D. Pachner. Enabling Powertrain Variants through Efficient Controls Development. *SAE Technical Paper 2014-01-1160*, 2014, doi:10.4271/2014-01-1160.
- H. Borhan, G. Kothandaraman, B. Pattel. Air handling control of a diesel engine with a complex dual-loop EGR and VGT air system using MPC. *Proceedings of the American Control Conference 2015*, 4509-4516, 2015, doi:10.1109/ACC.2015.7172039.
- K. Bencherif, D. von Wissel, L. Lansky, D. Kihas. Model Predictive Control as a Solution for Standardized Controller Synthesis and Reduced Development Time Application Example to Diesel Particulate Filter Temperature Control. *SAE Technical Paper 2015-01-1632*, 2015, doi:10.4271/2015-01-1632.
- E. R. Gelso, J. Dahl. Air-Path Control of a Heavy-Duty EGR-VGT Diesel Engine. *Proceedings of the 8th IFAC International Symposium on Advances in Automotive Control*, Norrköping, Sweden, 2016, doi:10.1016/j.ifacol.2016.08.086.

- E. Bouvier, D. Kihas, J. S. Roux, S. Vankayala, D. Jeckel, C. Riviere, M. Uchanski. The influence of advanced boosting on transient NOx control in Light Vehicle Diesel engines. *SIA Powertrain 2016 Conference*, Rouen, France, 2016
- D. Kihas, G. Stewart, J. Pekar, D. Pachner, M. Uchanski. Systematic Process for Physics Based Modeling and Model Predictive Control of Engine and Aftertreatment Systems. *7th Emission Control Conference*, Dresden, Germany, 2014
- D. von Wissel, A. Husson, V. Talon, L. Lansky, D. Pachner, M. Uchanski. Reducing Engine Calibration Time and Cost with Model Predictive Control. *IAV 10th Automotive Powertrain Control Systems Conference*, Berlin, Germany, 2014

[C.7] P. Otta, O. Šantin, V. Havlena. Tailored QP Algorithm for Predictive Control with Dynamics Penalty. *Proceedings of the 22nd Mediterranean Conference on Control & Automation*, Palermo, Italy, 384-389, 2014. Co-authorship: 33%

[C.8] P. Otta, O. Šantin, V. Havlena. Measured-State Driven Warm-Start Strategy for Linear MPC. *Proceedings of the European Control Conference*, Brussels, 3137-3141, 2015. Co-authorship: 33%

[C.9] P. Otta, O. Šantin. Fast QP Algorithm for Dynamics Penalty Model Predictive Control. *POSTER 2013*, Prague, Czech Republic, 2013. Co-authorship: 50%

Curriculum Vitae of the Author

Ondřej Šantin was born in Turnov, Czech Republic, in 1985. He received his bachelor degree in Electrical Engineering from Czech Technical University (CTU) Prague, Prague, the Czech Republic in 2007 and his master degree in Electrical Engineering and Cybernetics at the same university in 2009. At the same year, 2009, he started his Ph.D. studies in the Control Engineering and Robotics at CTU Prague. The main topic of the research are the numerical algorithms of quadratic programming for the Model Predictive Control (MPC).

Ondřej is author or co-author of several international conference papers in the area of control and optimization (*IFAC World Congress 2011, IEEE MCS 2011, IEEE CDC 2016, ECC 2015*) and automotive applications of advanced control methods (*SAE World Congress 2014, 2016*). He is a holder of two US patents in the domain of applications of advanced control methods to the automotive. The most important theoretical results were published in the international impacted journal *Optimization Methods and Software* (IF: 0.841) and the application results in the reviewed *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*.

In 2009, Ondřej joined Honeywell Prague Laboratory in Prague where he worked as a Research Engineer developing the real-time optimization algorithms, specifically the quadratic programming solver for the MPC applications for automotive. Besides that he did a modeling and control of diesel engines and application of the MPC to the several automotive control problems. From 2012, he is with Honeywell Automotive Software performing the research and development of the numerical algorithms for the MPC technology and its applications to the automotive area cooperating with leading manufacturers of cars and trucks worldwide.

Prague, August 2016

Ondřej Šantin

Appendix A

Support for Optimization

A.1 Rate of Convergence

To measure how fast the iterative algorithm converges to the solution $\boldsymbol{x}^*$ we define the algorithm's rate of convergence. The most common types of converge are defined closely following the definition of the convergence rates in [65, Chapter A.2.].

Lemma A.1.1 (*Q-linear convergence*)**.** Let $\{\boldsymbol{x}_k\}$ be a sequence in $\mathbb{R}^n$ that converges to $\boldsymbol{x}^*$. We say that the convergence is *Q-linear* if there is a constant $r \in (0, 1)$ such that

$$\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} \leq r, \quad \text{for all } k \text{ sufficiently large.}$$

Lemma A.1.2 (*Q-superlinear convergence*)**.** Let $\{\boldsymbol{x}_k\}$ be a sequence in $\mathbb{R}^n$ that converges to $\boldsymbol{x}^*$. We say that the convergence is *Q-superlinear* if

$$\lim_{k\to\infty} \frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|} = 0.$$

Lemma A.1.3 (*Q-quadratic convergence*)**.** Let $\{\boldsymbol{x}_k\}$ be a sequence in $\mathbb{R}^n$ that converges to $\boldsymbol{x}^*$. We say that the convergence is at least *Q-quadratic* if

$$\frac{\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^*\|}{\|\boldsymbol{x}_k - \boldsymbol{x}^*\|^2} \leq M \quad \text{for } M > 0.$$

Lemma A.1.4 (*R-linear convergence*)**.** Let $\{\boldsymbol{x}_k\}$ be a sequence in $\mathbb{R}^n$ that converges to $\boldsymbol{x}^*$. We say that the convergence is *R-linear* if there is a sequence of nonnegative scalars $\{v_k\}$ such that

$$\|\boldsymbol{x}_k - \boldsymbol{x}^*\| \leq v_k \quad \text{for all } k,$$

and $\{v_k\}$ converges Q-linearly to zero.

In the work, we omit the letter Q and simply talk about linear, superlinear convergence, and so on. Whenever we want to emphasize that we are talking about R-type of convergence we use the letter "R".

A.2 Optimality Conditions

To certify that the output of the solver is solution of an optimization problem one needs to have a certificate. For convex problems, the fulfillment of so-called KKT optimality conditions serves as a necessary and sufficient proof of the optimality of the solution, see [18, Chapter 5.5.3].
For the sake of completeness and better reader overview, the KKT optimality conditions are shown for (1.1) with a fixed parameter by following [18, Chapter 5.5].
With a fixed parameter $\boldsymbol{\theta}$, the problem (1.1) can be written as

$$q^* \triangleq q(\boldsymbol{z}^*) = \min_{\boldsymbol{z}\in\Omega} \tfrac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{h}^T\boldsymbol{z}, \tag{A.1}$$

with strictly feasible set $\Omega = \{\boldsymbol{z} : \underline{\boldsymbol{z}} \leq \boldsymbol{z} \leq \overline{\boldsymbol{z}}, \underline{\boldsymbol{z}} < \overline{\boldsymbol{z}}\}$ and with $\boldsymbol{H}$ as a SPD matrix, $\boldsymbol{h}$ and $\boldsymbol{z}$, $\underline{\boldsymbol{z}}$, $\overline{\boldsymbol{z}}$ are n-vectors.

We define the Lagrangian function L associated with the problem (A.1) as

$$L(\boldsymbol{z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \tfrac{1}{2}\boldsymbol{z}^T\boldsymbol{H}\boldsymbol{z} + \boldsymbol{h}^T\boldsymbol{z} + \boldsymbol{\lambda}^T(\underline{\boldsymbol{z}} - \boldsymbol{z}) + \boldsymbol{\mu}^T(\boldsymbol{z} - \overline{\boldsymbol{z}}), \tag{A.2}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ and $\boldsymbol{\mu} \in \mathbb{R}^n$ are the dual variables or *Lagrange multiplier vectors* of lower and upper bound respectively. We refer λ_i as the Lagrange multiplier associated with the inequality $\underline{\boldsymbol{z}}_i \leq \boldsymbol{z}_i$ constraint; similarly we refer μ_i as the Lagrange multiplier associated with the inequality $\boldsymbol{z}_i \leq \overline{\boldsymbol{z}_i}$.

The KKT conditions of (A.1) are then derived from the fact that gradient of (A.2) at $\boldsymbol{x}^*$ and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ must vanish; hence,

$$\begin{aligned}
\underline{z}_i - z_i^* &\leq 0, && i = 1, \ldots, n \\
z_i^* - \overline{z_i} &\leq 0, && i = 1, \ldots, n \\
\lambda_i^* &\geq 0, && i = 1, \ldots, n \\
\mu_i^* &\geq 0, && i = 1, \ldots, n \\
\lambda_i^*(\underline{z}_i - z_i^*) &= 0, && i = 1, \ldots, n \\
\mu_i^*(z_i^* - \overline{z_i}) &= 0, && i = 1, \ldots, n \\
\boldsymbol{H}\boldsymbol{z}^* + \boldsymbol{h} - \boldsymbol{\lambda}^* + \boldsymbol{\mu}^* &= 0. && \text{(A.3)}
\end{aligned}$$

Additionally, since the upper and lower of i-th bound constraint cannot be active at the same time[1]

$$\lambda_i \cdot \mu_i = 0, \qquad i = 1, \ldots, n.$$

Combining this with equality (A.3) and the definition of gradient of q as $\boldsymbol{g}^* = \boldsymbol{H}\boldsymbol{z}^* + \boldsymbol{h}$ one can deduce that Lagrange multipliers are defined exactly by the gradient as

$$\lambda_i^* = \begin{cases} g_i^* & \text{for } i \in \mathcal{L} \\ 0 & \text{elsewhere,} \end{cases} \qquad \mu_i^* = \begin{cases} -g_i^* & \text{for } i \in \mathcal{U} \\ 0 & \text{elsewhere.} \end{cases}$$

[1] That follows from the fact that a feasible set Ω is strictly feasible.

Appendix B

Algorithms

B.1 Cholesky Factorization of an Augmented Matrix

The Cholesky factorization of augmented matrix has been presented in [2] as an effective tool for update of the Cholesky factor when new rows and columns are appended to the original matrix. The algorithm is shown here for the sake of completeness.

Let $\boldsymbol{X}^{(0)} \in \mathbb{R}^{n\times n}$ be positive definite matrix and $\boldsymbol{R}^{(0)} \in \mathbb{R}^{n\times n}$ be its upper triangular Cholesky factor with strictly positive diagonal so that $\boldsymbol{X} = (\boldsymbol{R}^{(0)})^T\boldsymbol{R}^{(0)}$, see e.g. [65] for details about Cholesky factorization.

Suppose an augmented positive definite matrix $\boldsymbol{X}^{(1)} \in \mathbb{R}^{(n+m)\times(n+m)}$ defined as

$$\boldsymbol{X}^{(1)} = \begin{bmatrix} \boldsymbol{X}^{(0)} & \boldsymbol{S} \\ \boldsymbol{S}^T & \boldsymbol{T} \end{bmatrix},$$

where $\boldsymbol{S} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{T} \in \mathbb{R}^{m\times m}$. Then its Cholesky factor will take the form

$$\boldsymbol{R}^{(1)} = \begin{bmatrix} \boldsymbol{R}^{(0)} & \boldsymbol{U} \\ \boldsymbol{0} & \boldsymbol{V} \end{bmatrix},$$

with $\boldsymbol{U} \in \mathbb{R}^{n\times m}$ and $\boldsymbol{V} \in \mathbb{R}^{m\times m}$ which is an upper triangular with strictly positive diagonal which both need to be computed. The Cholesky factorization of augmented matrix presented in [2] computes $\boldsymbol{R}^{(1)}$ from a given $\boldsymbol{R}^{(0)}$ by initialing matrix $\boldsymbol{W} \in \mathbb{R}^{(n+m)\times(n+m)}$ as

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{R}^{(0)} & \boldsymbol{S} \\ \boldsymbol{0} & \boldsymbol{T} \end{bmatrix}, \tag{B.1}$$

and running the Algorithm 5 in $\mathcal{O}(m^3) + \mathcal{O}(m^2n) + \mathcal{O}(mn^2)$ flops. Note that when $m = 0$ the Algorithm 5 performs standard Cholesky factorization of top left block of matrix $\boldsymbol{W}$.

Algorithm 5 Cholesky factorization of an augmented matrix. Given $\boldsymbol{W} \in \mathbb{R}^{(n+m)\times(n+m)}$ defined as (B.1).

```
   [W] = cholaug(W,m,n)
 1: if m = 0 then
 2:    q = 1
 3: else
 4:    q = n + 1
 5: end if
 6: for i = 1 to n + m do
 7:    if i ≥ q then
 8:       w_{i,i} = sqrt(w_{i,i} − Σ_{k=1}^{i−1} w_{k,i}^2)
 9:    end if
10:    for j = max{i + 1, q} to n + m do
11:       w_{i,j} = (w_{i,j} − Σ_{k=1}^{i−1} w_{k,i} w_{k,j}) / w_{i,i}
12:    end for
13: end for
```

B.2 Forward Substitution

Having a non-singular upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ and vector $\boldsymbol{b} \in \mathbb{R}^n$ the solution of the following set of linear equations

$$\boldsymbol{R}^T\boldsymbol{x} = \boldsymbol{b},$$

with $\boldsymbol{x} \in \mathbb{R}^n$ can be executed in $\mathcal{O}(n^2)$ flops by Algorithm 6 adopted from [87]. Note that the straight modification allows in-place operation overwriting the right hand side vector $\boldsymbol{b}$.

Algorithm 6 Forward substitution, adoption of [87, Section 2.1]. Given nonsingular upper tringular matrix $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$. Solves $\boldsymbol{R}^T\boldsymbol{x} = \boldsymbol{b}$.

```
   [x] = forwardsub(R,b,n)
 1: for k = 1 to n do
 2:    x_k = b_k
 3:    for j = 1 to k − 1 do
 4:       x_k = x_k − r_{j,k} x_j
 5:    end for
 6:    x_k = x_k / r_{k,k}
 7: end for
```

B.3 Backward Substitution

Having a non-singular upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ and vector $\boldsymbol{b} \in \mathbb{R}^n$ the solution of the following set of linear equations

$$\boldsymbol{R}\boldsymbol{x} = \boldsymbol{b},$$

with $\boldsymbol{x} \in \mathbb{R}^n$ can be executed in $\mathcal{O}(n^2)$ flops by Algorithm 7 adopted from [87]. Note that the straight modification allows in-place operation overwriting the right hand side vector $\boldsymbol{b}$.

Algorithm 7 Backward substitution, adoption of [87, Section 2.4]. Given nonsingular upper tringular matrix $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{R}^n$. Solves $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{b}$.

`[`$\boldsymbol{x}$`] = backwardsub(`$\boldsymbol{R}$`,`$\boldsymbol{b}$`,`n`)`

1: **for** $k = n$ **to** 1 **do**
2: $\quad x_k = b_k$
3: $\quad$ **for** $j = k + 1$ **to** n **do**
4: $\quad\quad x_k = x_k - r_{k,j}\, x_j$
5: $\quad$ **end for**
6: $\quad x_k = x_k \,/\, r_{k,k}$
7: **end for**

B.4 Givens Rotation

A Givens rotation or plane rotation is an orthogonal transformation and serves as an effective tool for triangularization of close to triangular matrices by introducing zeros in the matrix. A Givens rotation is defined as orthogonal matrix of the form [87]

$$\boldsymbol{P}^{i,j} = \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & -s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix}, \tag{B.2}$$

where $c^2 + s^2 = 1$, c and s appears at the intersections i-th and j-th rows and columns.

The construction and purpose of the Givens rotation is demonstrated on the example. Let suppose the close to triangular matrix $\boldsymbol{X} \in \mathbb{R}^{4 \times 4}$ in the form

$$\boldsymbol{X} = \begin{bmatrix} \mathtt{X} & \mathtt{X} & \mathtt{X} & \mathtt{X} \\ 0 & \mathtt{a} & \mathtt{X} & \mathtt{X} \\ 0 & \mathtt{b} & \mathtt{X} & \mathtt{X} \\ 0 & 0 & 0 & \mathtt{X} \end{bmatrix},$$

where X, a and b denote nonzero elements. It is evident that matrix $\boldsymbol{X}$ is almost upper triangular except the element under the main diagonal $x_{3,2} = \mathtt{b}$ which is nonzero. To annihilate the nonzero element we construct plane rotation matrix in the form

$$\boldsymbol{P}^{2,3} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & -s & 0 \\ 0 & s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

with

$$c = \frac{a}{\sqrt{a^2 + b^2}}, \quad s = \frac{b}{\sqrt{a^2 + b^2}}.$$

Then application of the Givens rotation leads to the upper triangular matrix $\boldsymbol{Y} \in \mathbb{R}^{n \times n}$

$$\boldsymbol{Y} = \boldsymbol{P}^{2,3}\boldsymbol{X} = \begin{bmatrix} \mathtt{X} & \mathtt{X} & \mathtt{X} & \mathtt{X} \\ 0 & \hat{\mathtt{X}} & \hat{\mathtt{X}} & \hat{\mathtt{X}} \\ 0 & 0 & \hat{\mathtt{X}} & \hat{\mathtt{X}} \\ 0 & 0 & 0 & \mathtt{X} \end{bmatrix},$$

where $\hat{\mathtt{X}}$ indicates changed non-zero element.

Observing the form of the matrix $\boldsymbol{P}^{i,j}$ in (B.2) one can deduce that Givens rotation modifies only the i-th and j-th rows taking into account only the i-th and j-th rows. The following two algorithms implement the construction of the Givens rotation and its application to the rows of the matrix. From the numerical perspective the Givens rotation enjoys the same stability properties as Householder transformations, see [87] more details of numerical stability analysis. From the computational perspective, each application of Givens rotation can be executed in $6n$ flops. The computational complexity can be reduced by use of "Fast Givens Rotation", see [75].

Algorithm 8 Generation of Givens rotation [87, Section 4.1.3]. Given quantities a and b, generates the Givens rotation in c and s overwriting a with $\sqrt{a^2 + b^2}$ and b with 0.

`[`c`, s, a, b ] = rotgen(`a`, b)`

1: $\tau = |a| + |b|$
2: **if** $\tau = 0$ **then**
3: $\quad c = 1$; $s = 0$; return
4: **end if**
5: $\nu = \tau * \sqrt{(a/\tau)^2 + (b/\tau)^2}$
6: $c = a/\nu$; $s = b/\nu$
7: $a = \nu$; $b = 0$

Algorithm 9 Application of Givens rotation [87, Section 4.1.3]. Applies Givens rotation defined by c and s to two n-vectors x and y, overwriting them. It involves need of temporary n-vector $\boldsymbol{t}$.

`[`$\boldsymbol{x}$`, `$\boldsymbol{y}$`] = rotapp(`$\boldsymbol{x}$`, `$\boldsymbol{y}$`, `c`, `s`)`

1: $\boldsymbol{t} = c\boldsymbol{x} + s\boldsymbol{y}$
2: $\boldsymbol{y} = c\boldsymbol{y} - s\boldsymbol{x}$
3: $\boldsymbol{x} = \boldsymbol{t}$