Multiple-Domain Optimal Control

by

Ing. Pavel Otta

supervised by *Prof. Ing. Vladimír Havlena, CSc.,*

Dissertation

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

Doctor of Philosophy

in Ph.D. program
Electrical Engineering and Information Technology
in the branch of study
Control Engineering and Robotics

Czech Technical University in Prague,
Dec 2023

„Znát dobré, chtít dobré, konat dobré,
i když se nikdo nedívá."   Jan Amos Komenský

Acknowledgements

Writing this thesis would not have been possible without great people around me. They were inspiring in academics, work, and private life.

I was lucky to have an inspiring advisor, prof. Vladimír Havlena. He was always willing to help and give advice, but he also gave me the freedom to pursue research I liked.

I have spent wonderful years at the Department of Control Engineering. We had countless coffee breaks with Jiří Dostál, Kamil Dolinský, and Radek Beňo. Many lunch breaks with senior researcher Martin Hročík, Zdeněk Hurák, and Petr Hušek.

After that, there was a six-year period when I had the opportunity to work with great researchers, engineers, and mentors at Honeywell. Working with Pavel Trnka and Daniel Pachner was a great pleasure, and I am indebted to them for boosting me tremendously in my work. The latter one has beaten me in scooter racing ever since.

The recent inspiration comes from my current job in Garrett Motion. Thanks to Ondřej Šantin for supporting my studies from the beginning (he was my supervisor at the university before) to the end (he is currently my manager).

My gratitude goes to my friends Jaroslav Tabaček and Jan Prášek for the countless ideas we have shared. I must mention Jirka Zemánek, Martin Gurtner, Štefan Knotek, Ivo Herman, and Dan Martinec. I have enjoyed their company, which was full of brilliant work combined with creativity that I admire. I will always remember the fun with friends from the dormitories.

The greatest support I have received was from my wife, Kamilka. It is her love that makes my life wealthy and childful. I'm grateful for my inspiring children - Toníček, Mikuláš, and Štěpánek. Special thanks to Evička for making our family complete. Last but not least, I thank my parents for their continuous care.

Declaration

This doctoral thesis is submitted in partial fulfillment of the requirements for the degree of doctor (Ph.D.). The work presented in this dissertation is the result of my investigation, except where otherwise stated. I declare that I worked out this thesis independently and quoted all used sources of information following the Methodical instructions of Czech Technical University in Prague (CTU in Prague) about ethical principles for writing an academic thesis. Moreover, I declare that this work was not submitted or accepted for any other degree.

CTU in Prague
Dec 2023

Pavel Otta

Abstract

This work focuses on the solution of the speed profile optimization utilizing time-spatial domain transformation. The work is motivated by practical requirements and comprehensively solves the problem of speed profile optimization, from the problem formulation to the algorithm for the optimization tasks. In the proposed solution, simplicity and reliability are emphasized to facilitate transfer for potential use in the automotive industry.

Practical requirements covered in this work are simplicity, reliability (solution is available at a desired time), the computational efficiency of speed profile optimization with arbitrary (e.g., zero) initial and final speed, and fixed travel time. Further, speed limits, road grades, and an adequately complex vehicle model are assumed. Optimization is robust to the speed limit violation. Moreover, passage through intersections with traffic lights with known passage intervals is also optimized.

Key words: Speed profile optimization, time-spatial domain transformation, quadratic programming solution

Abstrakt

V této práci se zabývám řešením optimalizace rychlostního profilu s využitím transformace mezi časovou a prostorovou doménou. Práce vychází z praktických požadavků a problém optimalizace rychlostního profilu řeší komplexně. To znamená od formulace optimalizační úlohu až po samotný optimalizační algoritmus. Přitom důraz je kladen na jednoduchost a spolehlivost celkového řešení, aby se usnadnilo potenciální užití v automobilovém průmyslu.

Praktické požadavky zahrnuté v této práci jsou jednoduchost, spolehlivost (dostupnost řešení v předem stanovený čas) a efektivita výpočtu rychlostního profilu s libovolnou (např. nulovou) počáteční a koncovou rychlostí a pevně danou dobou jízdy. Řešení dále uvažuje rychlostní omezení, sklon vozovky a přiměřeně složitý model vozidla. Optimalizace je robustní vůči překročení rychlostního limitu. Dále je řešeno plánování průjezdu křižovatkami se světelnou signalizací, jejíchž průjezdové intervaly jsou předem známy.

Klíčová slova: Optimalizace rychlostního profilu, transformace mezi časovou a prostorovou doménou, řešení kvadratického programování

Contents

1 Introduction

Due to the legislator's push for emission reduction, the whole automotive industry is looking for new ways of achieving more efficient, greener solutions for powertrain units to drive cars. One of the promising directions is performing a complete system optimization to achieve even more robust efficiency improvements in combustion, hybrid, or electric vehicles.

Vehicle equipment such as Speed Advisory Systems (SAS) can improve fuel consumption, ride comfort, or reduce idle time on red traffic lights [Wan et al., 2016]. Connected vehicles can also utilize traffic signal information predictively to manage their speed in advance to lower fuel consumption and reduce idling [Mandava et al., 2009] or trip time [Asadi and Vahidi, 2011]. Signal Phase and Timing (SPaT) information of signalized intersections is provided to the vehicle to encourage economical driving while passing through the intersections. In vehicle systems, calculate, based on SPaT, and provide speed advice to the driver, allowing the driver to adapt the vehicle's speed to pass through the upcoming signal on green [Xia et al., 2012].

Another direction of development is on the side of infrastructure, which is outside the scope of this work. The idea is that each intersection has a management system planning the passage of semi or fully autonomous vehicles through the intersection [Au et al., 2015].

Technological enablers are vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication that brings information valuable for planning and optimization to the vehicle. Then, efficient optimization and planning can be performed on board while preserving a high level of vehicle autonomy, which is essential from the safety operation perspective. Current vehicle control units (VCU) are powerful, and multicore processors are being used these days. However, speed profile optimization of relevant routes is still complex. *Speed profile optimization* can be defined as a problem of minimizing fuel consumption subject to speed and acceleration limits and time constraints. Alternatively, a comfort or another safety constraint can also be formulated. The problem is (in principle) even more complex when the traffic light passage comes into place.

In this work, algorithmic aspects of speed profile optimization are explored, and a smart solution is proposed. The proposed solution introduces innovative problem formulation, on the one hand, and a sophisticated algorithm for its solution, on the other hand.

Vehicle Dynamics

A longitudinal vehicle dynamic

$$\begin{aligned} \frac{ds(t)}{dt} &= v(t), \quad s(t_0) = x_0, \\ \frac{dv(t)}{dt} &= a(t), \quad v(t_0) = v_0. \end{aligned} \tag{1.1}$$

is considered. The double integrator is the minimalistic model containing the essence the proposed approach requires. However, the approach can handle more than a simple model, and common extensions (aerodynamic drag, tire resistance, brakes, gravity) can be covered straightforwardly.

Objective Function

Treating the trade-off between time and economy is essential in these tasks - one can imagine an aggressive or passive control strategy. The aggressive strategy minimizes time while it is not economical. The passive one minimizes the economic cost while resulting in long travel time. In this work, the economic cost is minimized subject to the fixed time. The economic cost is represented via positive acceleration (generated by the engine). Acceleration can be seen as a normalized force. Therefore, the stage cost

$$l(a_k) = \bar{v}_k \mathit{\Delta} t_k \frac{\max\{0, a_k\}}{\eta(\max\{0, a_k\}, v_k)}, \tag{1.2}$$

represents unit mass work, which unit is m^2/s^2, and where $\bar{v}_k$ is the average speed over the time increment $\mathit{\Delta} t_k = t_{k+1} - t_k$. Note that in this work, the forward differentiation convention is used. Engine efficiency map $\eta(\max\{0, a_k\}, v_k)$ parametrized by acceleration produced by the engine and the current speed of the vehicle.

Optimal Control Problem

Basic speed profile optimization problem can be formulated as

$$\begin{aligned}
\min \quad & \frac{1}{m} \sum_{k=0}^{N-1} l(a_k), & (1.3a)\\
\text{s.t.} \quad & t_{k+1} = t_k + \mathit{\Delta} t_k,\ t_0 = t_{\text{init}}, & (1.3b)\\
& v_{k+1} = v_k + a_k \mathit{\Delta} t_k,\ v_0 = v_{\text{init}}, & (1.3c)\\
& \mathit{\Phi}(t_k, \underline{v}_k) \le v_k \le \mathit{\Phi}(t_k, \overline{v}_k),\ k = 1, \dots, N & (1.3d)\\
& \underline{a}_k \le a_k \le \overline{a}_k,\ k = 0, \dots, N-1 & (1.3e)\\
& \sum_{k=1}^{N} \mathit{\Delta} t_k = t_f. & (1.3f)
\end{aligned}$$

with travel optimization horizon N and time increment $\mathit{\Delta} t_k = t_{k+1} - t_k$. The optimization problem minimizes the engine's work in the cost function subject to the dynamics, speed and acceleration limits, and fixed final time. It is assumed that vehicle mass m is constant. Hence, it is omitted further in this work.

Challenges of the optimization problem that make the problem difficult to compute in embedded devices are

Challenge 1 Long optimization horizon

Challenge 2 Affine constraints

Challenge 3 Constraints are function of state variable

Time-to-spatial Domain Transformation

A solution based on a straightforward transformation of longitudinal dynamics parametrized in time to the spatial parametrization was evaluated in [1]. The transformation simplifies the state constraint to simple bounds. On the other hand, it makes the model

$$\begin{aligned} \frac{d\tilde{t}(x)}{dx} &= \frac{1}{\tilde{v}(x)}, \quad \tilde{t}(x_0) = t_0, \\ \frac{d\tilde{v}(x)}{dx} &= \tilde{a}(x)\frac{1}{\tilde{v}(x)}, \quad \tilde{v}(x_0) = \tilde{v}_0 \end{aligned} \tag{1.4}$$

nonlinear. The nonlinearity is stronger as the speed approaches zero - the singularity. The transformed variables are time $\tilde{t}(x) = s^{-1}(x) = t$, speed $\tilde{v}(x) = v(\tilde{t}(x))$, and acceleration $\tilde{a}(x) = a(\tilde{t}(x))$ at a particular position $x \in \left[x_0, x_f\right]$ originates from Sampei and Furuta [1986].

Since the optimization shall be running not in time but in space instead, the cost function needs to be updated accordingly

$$l(\tilde{a}_k) = \Delta s_k \frac{\max\{0, \tilde{a}_k\}}{\eta(\max\{0, \tilde{a}_k\}, \tilde{v}_k)}, \tag{1.5}$$

where $\Delta s_k \approx \tilde{v}_k \Delta t_k$, to minimize fuel efficiency parametrized in space.

The optimal control problem in spatial domain can be summarized as

$$\min \quad \sum_{k=0}^{N-1} l(\tilde{a}_k), \tag{1.6a}$$

$$\text{s.t.} \quad \tilde{t}_{k+1} = \tilde{t}_k + \frac{1}{\tilde{v}_k}\Delta s_k, \tilde{t}_0 = \tilde{t}_{\text{init}}, \tag{1.6b}$$

$$\tilde{v}_{k+1} = \tilde{v}_k + \tilde{a}_k \frac{1}{\tilde{v}_k}\Delta s_k, \tilde{v}_0 = \tilde{v}_{\text{init}}, \tag{1.6c}$$

$$\underline{\tilde{v}}_k \le \tilde{v}_k \le \overline{\tilde{v}}_k, \;\; k = 1, \dots, N \tag{1.6d}$$

$$\underline{\tilde{a}}_k \le \tilde{a}_k \le \overline{\tilde{a}}_k, \;\; k = 0, \dots, N-1 \tag{1.6e}$$

$$\sum_{k=1}^{N} \frac{1}{\tilde{v}_k}\Delta s_k = t_f. \tag{1.6f}$$

In problem (1.6a), Challenge 3 is overcome by the transition to the spatial domain. On the other hand, it brings two additional difficulties into the play – singularity for zero speed and a nonlinear dynamical model.

1.1 Goal of This Work

In this thesis, all the challenges, namely

Challenge 1 Long optimization horizon

Challenge 2 Affine constraints

Challenge 3 Constraints are function of state variable

Challenge 4 Singularity for zero speed

Challenge 5 Nonlinear dynamical model

that comes with longitudinal dynamic optimization are addressed, and a suitable solution of optimal control problem (1.3a) is proposed.

2 Multiple-Domain Optimal Control

A novel approach to efficient speed profile optimization over the finite horizon is presented. The approach uses exact model discretization, transition into the spatial domain, and exact linearization of the dynamical model in the spatial domain is presented in this section.

2.1 Longitudinal Vehicle Dynamics

The zero order hold of a_k is used to exact linearization of (1.1)

$$s_{k+1} = s_k + v_k \Delta t_k + \frac{1}{2} a_k \Delta^2 t_k, \tag{2.1}$$

$$v_{k+1} = v_k + a_k \Delta t_k. \tag{2.2}$$

The relation between time and spatial increment given by (2.1)

$$\begin{aligned} \Delta s_k &= v_k \Delta t_k + \frac{1}{2} a_k \Delta^2 t_k, \\ \frac{\Delta s_k}{\Delta t_k} &= \frac{2 v_k + a_k \Delta t_k}{2}, \end{aligned}$$

implies the relation for the time increase

$$\Delta t_k = \frac{1}{\bar{v}_k} \Delta s_k, \quad \Delta s_k = s_{k+1} - s_k, \tag{2.3}$$

where $\bar{v}_k = \frac{v_k + v_{k+1}}{2}$ is the average speed. This is important for the practical requirement that speed can be zero at the beginning and the end of the travel horizon. It is strictly positive otherwise, i.e.

$$\begin{aligned} v_0, v_N &\geq 0, \\ v_{1,\dots,N-1} &> 0, \end{aligned} \qquad \Longrightarrow \qquad \bar{v}_{0,\dots,N-1} > 0. \tag{2.4}$$

The resulting dynamics obtained by substituting (2.3) to (2.2) transformed into the spatial domain is then

$$v_{k+1} = v_k + a_k \frac{1}{\bar{v}_k} \Delta s_k. \tag{2.5}$$

Linear dynamical model transformed into spatial domain became nonlinear in speed. By manipulation, we can simplify the transformed dynamics as

$$\begin{aligned} v_{k+1} - v_k &= a_k \frac{2}{v_k + v_{k+1}} \Delta s_k \\ (v_k + v_{k+1})(v_{k+1} - v_k) &= 2 a_k \Delta s_k \\ v_{k+1}^2 - v_k^2 &= 2 a_k \Delta s_k. \end{aligned}$$

The dynamical model becomes linear once a quadratic speed $q_k = v_k^2$ is substituted as

$$q_{k+1} = q_k + 2 a_k \Delta s_k, \tag{2.6}$$

and due to the non-negativity of speed, also $v_k = \sqrt{q_k}$ holds. Since the dynamic is linear in quadratic speed, a suitable model inspired by A. Vahidi and Peng [2005] is being used in this work. The model, however generic enough to cover also rail vehicles [Davis and Niehaus, 1926] used up to these days, is

$$a_k = \overbrace{a^+}^{\text{thrust}} + \overbrace{a^-}^{\text{brake}} + \overbrace{\alpha + \gamma v_k^2}^{\text{roll. + air res.}} \overbrace{+ g \sin(\tfrac{\pi \varphi_k}{180})}^{\text{gravity}} = a^+ + a^- + a^q + a^\#, \tag{2.7}$$

where $a^q = \gamma q_k$, $a^\# = \alpha + g \sin(\frac{\pi \varphi_k}{180})$ and with calibration parameters α, γ, gravitational acceleration g, and road grade φ_k $[deg]$.

2.2 Speed Profile Optimization (in Space)

In this section, a (linear) mathematical program is described to explain the key ideas. Rather, simple models and cost functions are described to show the naked principles rather than complex ones. However, we will see that the class of problems that can be solved using the proposed approach is rich.

To avoid non-smooth nature of max function in (1.5), auxiliary variables for positive (a^+) and negative (a^-) accelerations are used such that

$$a_k = a_k^+ - a_k^- + a^q + a^\#, \qquad 0 \le a_k^+, \qquad 0 \le a_k^-, \tag{2.8}$$

then the stage cost (1.5) becomes smooth as follows

$$\begin{aligned} l_s(q_k, \epsilon_k, a_k^+) &= \mathcal{A}_k(q_k, a_k^+) + \frac{1}{2}\epsilon_k^2 \\ &\approx \Delta s_k \left(\frac{1}{2} \begin{bmatrix} q_k \\ a_k^+ \end{bmatrix}^T \begin{bmatrix} \mathcal{A}_{k,q_k q_k} & \mathcal{A}_{k,q_k a_k^+} \\ \mathcal{A}_{k,a_k^+ q_k} & \mathcal{A}_{k,a_k^+ a_k^+} \end{bmatrix} \begin{bmatrix} q_k \\ a_k^+ \end{bmatrix} + \begin{bmatrix} q_k \\ a_k^+ \end{bmatrix}^T \begin{bmatrix} \mathcal{A}_{k,q_k} \\ \mathcal{A}_{k,a_k^+} \end{bmatrix} \right) + \frac{1}{2}\epsilon_k^2, \end{aligned} \tag{2.9}$$

where $\mathcal{A}_k(q_k, a_k^+) = \Delta s_k \frac{a_k^+}{\eta(q_k, a_k^+)}$ and ϵ_k is a slack variable that will be used to penalize speed limit violations. The term $\mathcal{A}_k(q_k, a_k^+)$ is approximated around nominal (quadratic) speed and nominal acceleration by second-order Taylor expansion with first and second derivatives denoted as

$$\begin{aligned} \mathcal{A}_{k,x} &= \Delta s_k \frac{\partial}{\partial x} \mathcal{A}_k(x, y)|_{x_{\text{nom}}, y_{\text{nom}}}, \\ \mathcal{A}_{k,xy} &= \Delta s_k \frac{\partial}{\partial x \partial y} \mathcal{A}_k(x, y)|_{x_{\text{nom}}, y_{\text{nom}}}, \end{aligned}$$

evaluated in nominal points $x_{\text{nom}}, y_{\text{nom}}$ at stage k.

The resulting optimization problem over the travel optimization horizon N is

$$
\begin{aligned}
\min \quad & \sum_{k=0}^{N-1} l_s(q_k, \epsilon_k, a_k^+), && (2.10a)\\
\text{s.t.} \quad & q_{k+1} = q_k + 2\Delta s_k a_k, && (2.10b)\\
& 0 < \underline{q}_k \leq q_k + \epsilon_k \leq \overline{q}_k, && (2.10c)\\
& 0 \leq \underline{q}_N \leq q_N + \epsilon_N \leq \overline{q}_N, && (2.10d)\\
& a_k = a_k^+ - a_k^- + a^q + a^\#, && (2.10e)\\
& \underline{a}_k \leq a_k \leq \overline{a}_k, && (2.10f)\\
& 0 \leq a_k^+ \leq \infty, && (2.10g)\\
& 0 \leq a_k^- \leq \infty, \;\; k = 0, \ldots, N-1, && (2.10h)\\
& q_0 = v_0^2, && (2.10i)\\
& q_N = v_f^2, && (2.10j)\\
& \frac{1}{N+1} \sum_{k=1}^{N} q_k = c_f, && (2.10k)
\end{aligned}
$$

parametrized by initial speed v_0, terminal speed v_f, and average quadratic speed c_f.

Recall several important differences in problem (2.10a) compared to (1.6a). First, slack variables were added to overcome infeasibility when speed limits are violated. Second, linear longitudinal vehicle dynamics in quadratic speed are involved. Third, a terminal speed constraint is added. Fourth, the equality constraint that average quadratic speed is equal to parameter c_f is used to make the problem solvable instead of nonlinear constraint (1.6f). Parameter c_f is unknown to be optimized in superordinate optimization later.

In the following, the optimization problem is transformed into the standard form of a mathematical program. Concatenation of the optimization variables

$$
\boldsymbol{q} = \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_N \end{bmatrix}, \; \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_N \end{bmatrix} \in \mathcal{R}^{N+1}, \; \boldsymbol{a} = \begin{bmatrix} a_0 \\ \vdots \\ a_{N-1} \end{bmatrix}, \; \boldsymbol{a}^- = \begin{bmatrix} a_0^- \\ \vdots \\ a_{N-1}^- \end{bmatrix}, \; \boldsymbol{a}^+ = \begin{bmatrix} a_0^+ \\ \vdots \\ a_{N-1}^+ \end{bmatrix} \in \mathcal{R}^N,
$$

leads to a compactified notaion of the linear program

$$
\begin{aligned}
\min \quad & \frac{1}{2}\tilde{\boldsymbol{q}}^T \boldsymbol{P} \tilde{\boldsymbol{q}} + \tilde{\boldsymbol{q}}^T \boldsymbol{S} \tilde{\boldsymbol{a}}^+ + \frac{1}{2}\tilde{\boldsymbol{a}}^{+^T} \boldsymbol{R} \tilde{\boldsymbol{a}}^+ + \tilde{\boldsymbol{q}}^T \tilde{\boldsymbol{p}} + \tilde{\boldsymbol{a}}^{+^T} \tilde{\boldsymbol{r}} + \frac{1}{2}\tilde{\boldsymbol{\epsilon}}^T \tilde{\boldsymbol{\epsilon}}, && \text{(2.11a)}\\
\text{s.t.} \quad & \boldsymbol{A}\tilde{\boldsymbol{q}} = \boldsymbol{B}\tilde{\boldsymbol{a}}, && \text{(2.11b)}\\
& \underline{\tilde{\boldsymbol{q}}} \le \tilde{\boldsymbol{q}} + \tilde{\boldsymbol{\epsilon}} \le \overline{\tilde{\boldsymbol{q}}}, && \text{(2.11c)}\\
& \tilde{\boldsymbol{a}} = \tilde{\boldsymbol{a}}^+ - \tilde{\boldsymbol{a}}^- + \boldsymbol{C}\tilde{\boldsymbol{q}} + \tilde{\boldsymbol{a}}^{\#}, && \text{(2.11d)}\\
& \underline{\tilde{\boldsymbol{a}}} \le \tilde{\boldsymbol{a}} \le \overline{\tilde{\boldsymbol{a}}}, && \text{(2.11e)}\\
& \boldsymbol{0} \le \tilde{\boldsymbol{a}}^+ \le \infty, && \text{(2.11f)}\\
& \boldsymbol{0} \le \tilde{\boldsymbol{a}}^- \le \infty, && \text{(2.11g)}\\
& \boldsymbol{e}_0^T \tilde{\boldsymbol{q}} = v_0^2, && \text{(2.11h)}\\
& \boldsymbol{e}_N^T \tilde{\boldsymbol{q}} = v_f^2, && \text{(2.11i)}\\
& \boldsymbol{1}^T \tilde{\boldsymbol{q}} = (N+1)c_f, && \text{(2.11j)}
\end{aligned}
$$

with cost vectors and matrices

$$
\boldsymbol{P} = \begin{bmatrix} \mathcal{A}_{0,q_k q_k} & & & \\ & \mathcal{A}_{1,q_k q_k} & & \\ & & \ddots & \\ & & & \mathcal{A}_{N,q_k q_k} \end{bmatrix} \in \mathcal{R}^{N+1 \times N+1}, \ \boldsymbol{p} = \begin{bmatrix} \mathcal{A}_{0,q_k} \\ \mathcal{A}_{1,q_k} \\ \vdots \\ \mathcal{A}_{N,q_k} \end{bmatrix} \in \mathcal{R}^{N+1},
$$

$$
\boldsymbol{S} = \begin{bmatrix} \mathcal{A}_{0,a_k^+ q_k} & & & & \\ & \mathcal{A}_{1,a_k^+ q_k} & & & \\ & & \ddots & & \\ & & & \mathcal{A}_{N,a_k^+ q_k} & \boldsymbol{0} \end{bmatrix} \in \mathcal{R}^{N \times N+1},
$$

$$
\boldsymbol{R} = \begin{bmatrix} \mathcal{A}_{0,a_k^+ a_k^+} & & & \\ & \mathcal{A}_{1,a_k^+ a_k^+} & & \\ & & \ddots & \\ & & & \mathcal{A}_{N-1,a_k^+ a_k^+} \end{bmatrix} \in \mathcal{R}^{N \times N}, \ \boldsymbol{r} = \begin{bmatrix} \mathcal{A}_{0,a_k^+} \\ \mathcal{A}_{1,a_k^+} \\ \vdots \\ \mathcal{A}_{N,a_k^+} \end{bmatrix} \in \mathcal{R}^{N},
$$

and with constraint vectors and matrices defined as

$$\boldsymbol{A} = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}, \ \boldsymbol{C} = \begin{bmatrix} \gamma & 0 & & & \\ & \gamma & 0 & & \\ & & \ddots & \ddots & \\ & & & \gamma & 0 \end{bmatrix} \in \mathcal{R}^{N\times N+1},$$

$$\boldsymbol{B} = \begin{bmatrix} 2\Delta s_0 & & & \\ & 2\Delta s_1 & & \\ & & \ddots & \\ & & & 2\Delta s_{N-1} \end{bmatrix} \in \mathcal{R}^{N\times N}, \ \mathbf{1} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \in \mathcal{R}^{N+1},$$

$$\underline{\boldsymbol{a}} = \begin{bmatrix} \underline{a}_0 \\ \vdots \\ \underline{a}_{N-1} \end{bmatrix}, \ \overline{\boldsymbol{a}} = \begin{bmatrix} \overline{a}_0 \\ \vdots \\ \overline{a}_{N-1} \end{bmatrix} \in \mathcal{R}^{N}, \ \underline{\boldsymbol{q}} = \begin{bmatrix} \underline{q}_0 \\ \underline{q}_1 \\ \vdots \\ \underline{q}_N \end{bmatrix}, \ \overline{\boldsymbol{q}} = \begin{bmatrix} \overline{q}_0 \\ \overline{q}_1 \\ \vdots \\ \overline{q}_N \end{bmatrix} \in \mathcal{R}^{N+1},$$

$$\boldsymbol{e}_0 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ \boldsymbol{e}_N = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \in \mathcal{R}^{N+1}.$$

For a stacked vector of optimization variables

$$\mathbb{z} = \left[\tilde{\boldsymbol{q}}^T, \tilde{\boldsymbol{\epsilon}}^T, \ \tilde{\boldsymbol{a}}^T, \ \tilde{\boldsymbol{a}}^{+T}, \ \tilde{\boldsymbol{a}}^{-T}\right]^T \in \mathcal{R}^{5N+2}$$

the resulting mathematical program with affine equality constraint is

$$\min \quad \frac{1}{2}\tilde{\mathbb{z}}^T \tilde{\mathbb{H}} \tilde{\mathbb{z}} + \mathbb{f}^T \mathbb{z}, \tag{2.12a}$$

$$\text{s.t.} \quad \underline{\mathbb{b}} \leq \mathbb{A}\mathbb{z} \leq \overline{\mathbb{b}}, \tag{2.12b}$$

$$\underline{\mathbb{z}} \leq \mathbb{z} \leq \overline{\mathbb{z}}, \tag{2.12c}$$

which is a common form of generic quadratic program where

$$\mathbb{H}=\begin{bmatrix}\boldsymbol{P}&0&0&\boldsymbol{S}^T&0\\0&\boldsymbol{I}&0&0&0\\0&0&0&0&0\\\boldsymbol{S}&0&0&\boldsymbol{R}&0\\0&0&0&0&0\end{bmatrix}\in\mathcal{R}^{(5N+2)\times(5N+2)},\ \mathbb{f}=\begin{bmatrix}\boldsymbol{p}\\0\\0\\\boldsymbol{r}\\0\end{bmatrix},\ \underline{\mathbb{z}}=\begin{bmatrix}-\infty\\-\infty\\\underline{\tilde{a}}\\0\\0\end{bmatrix},\ \overline{\mathbb{z}}=\begin{bmatrix}\infty\\\infty\\\overline{\tilde{a}}\\\infty\\\infty\end{bmatrix},$$

$$\mathbb{A}=\begin{bmatrix}\boldsymbol{A}&0&-\boldsymbol{B}&0&0\\\boldsymbol{I}&\boldsymbol{I}&0&0&0\\-\boldsymbol{C}&0&\boldsymbol{I}&-\boldsymbol{I}&\boldsymbol{I}\\\boldsymbol{o}_0^T&\boldsymbol{0}^T&\boldsymbol{0}^T&\boldsymbol{0}^T&\boldsymbol{0}^T\\\boldsymbol{o}_N^T&\boldsymbol{0}^T&\boldsymbol{0}^T&\boldsymbol{0}^T&\boldsymbol{0}^T\\\boldsymbol{e}^T&\boldsymbol{0}^T&\boldsymbol{0}^T&\boldsymbol{0}^T&\boldsymbol{0}^T\end{bmatrix}\in\mathcal{R}^{(3N+4)\times(5N+2)},\ \underline{\mathbb{b}}=\begin{bmatrix}0\\\underline{\tilde{q}}\\\boldsymbol{a}^{\#}\\v_0^2\\v_f^2\\Nc_f\end{bmatrix},\ \overline{\mathbb{b}}=\begin{bmatrix}0\\\overline{\tilde{q}}\\\boldsymbol{a}^{\#}\\v_0^2\\v_f^2\\Nc_f\end{bmatrix}.$$

The second part of this thesis is dedicated to the solution of problem (2.12).

An example of the optimization (2.12) result is shown in Figure 2.1.

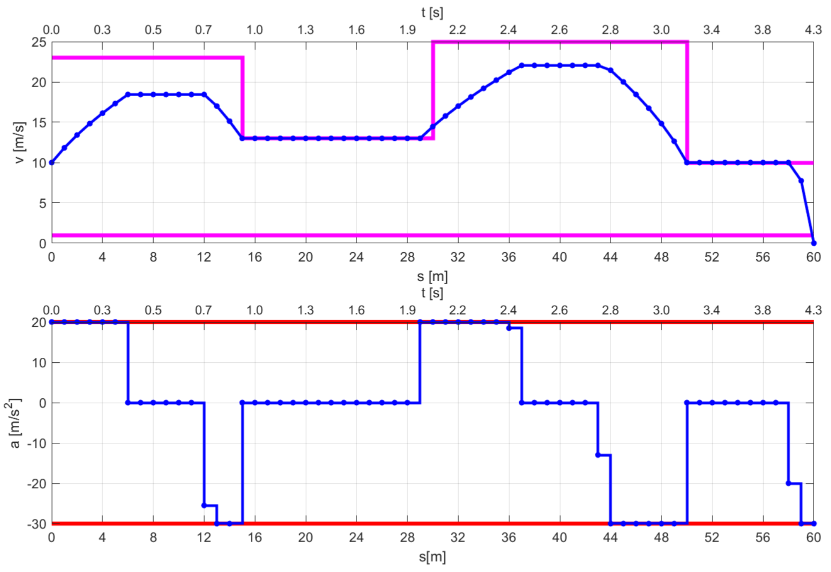


Figure 2.1: Speed and acceleration profiles in the spatial and time domain sampled equidistantly in space.

2.3 Speed Profile Optimization (in Acceleration)

The optimization problem (2.12) size can be dramatically high for long routes, prohibiting optimization on an embedded device. The issue is that the equidistant sampling period has to be small enough to capture the rapid changes in some problematic parts of the route. On the other hand, such a period leads to an oversampling on the parts where road conditions are unchanged for a long distance.

Move blocking [Raphael et al., 2007] is a common approach to reducing degrees of freedom in optimization from the control application. The approach fixes inputs for several consecutive periods. In our problem, there is no reasonable way to choose a period count for which the acceleration should be fixed on a single value. However, when analyzing a solution (Figure 2.1), one shall notice that optimal strategy behaves bang-bang-like. The solution contains intervals where the full acceleration is applied, then some intervals with constant speed, and some with full deceleration. One additional situation is typical when driving a car – no throttle nor brakes, the vehicle is only decelerating by the cause of resistances.

$$\begin{aligned}\tilde{l}_s(\Delta\tilde{s}_l, \tilde{q}_l, \tilde{\epsilon}_l) &= \mathcal{B}_l(q_{l,0}, \Delta\tilde{s}_{l,0}) + \mathcal{B}_l(q_{l,1}, \Delta\tilde{s}_{l,1}) + \frac{1}{2}\sum_{j=0}^{m-1}\tilde{\epsilon}_{l,j}^2 \\ &\approx \frac{1}{2}q_{l,0}\mathcal{B}_{l,q}(q_{l,0}, \Delta\tilde{s}_{l,0})\Delta\tilde{s}_{l,0} + \frac{1}{2}q_{l,1}\mathcal{B}_{l,q}(q_{l,1}, \Delta\tilde{s}_{l,1})\Delta\tilde{s}_{l,1} + \frac{1}{2}\sum_{j=0}^{m-1}\tilde{\epsilon}_{l,j}^2,\end{aligned} \tag{2.13}$$

where $\mathcal{B}_l(q_{l,j}, \Delta\tilde{s}_{l,j}) = \Delta\tilde{s}_{l,j}\frac{a_{l,j}^+}{\eta(q_{l,j}, a_{l,j}^+)}$. For this parametrization to remain in the class of quadratic programming, only first-order terms in Taylor expansion $\mathcal{B}_{l,q}(q_{l,j}, \Delta\tilde{s}_{l,j}) = \Delta\tilde{s}_{l,j}\frac{\partial}{\partial q_{l,j}}\mathcal{B}_l(q_{l,j})|_{q_{\text{nom}}}$ can be used. Hence, adequate nonlinearity approximation of efficiency function η is assumed here.

$$
\begin{aligned}
\min \quad & \sum_{l=1}^{M} \tilde{l}_s(\Delta\tilde{s}_l, \tilde{q}_l, \tilde{\epsilon}_l) + \frac{1}{2}\tilde{\epsilon}_{M,0}^2, && \text{(2.14a)}\\
\text{s.t.} \quad & \tilde{q}_{l-1,1} = \tilde{q}_{l-1,0} + 2\tilde{a}_0 \Delta\tilde{s}_{l-1,0}, \quad \text{(Acceleration mode } \tilde{a}_0 = \overline{a}\text{)} && \\
& \tilde{q}_{l-1,2} = \tilde{q}_{l-1,1} + 2\tilde{a}_1 \Delta\tilde{s}_{l-1,1}, \quad \text{(Constant speed mode } \tilde{a}_1 = 0\text{)} && \\
& \tilde{q}_{l-1,3} = \tilde{q}_{l-1,2} + 2\tilde{a}_2 \Delta\tilde{s}_{l-1,2}, \quad \text{(Zero thrust mode } \tilde{a}_2 = -(a^q + a^{\#})\text{)} && \\
& \tilde{q}_{l,0} = \tilde{q}_{l-1,3} + 2\tilde{a}_3 \Delta\tilde{s}_{l-1,3}, \quad \text{(Breaking mode } \tilde{a}_3 = \underline{a}\text{)} && \text{(2.14b)}\\
& 0 < \underline{\tilde{q}}_{l-1,j} \le \tilde{q}_{l-1,j} + \tilde{\epsilon}_{l-1,j} \le \overline{\tilde{q}}_{l-1,j}, && \text{(2.14c)}\\
& 0 < \underline{\tilde{q}}_{M,0} \le \tilde{q}_{M,0} + \tilde{\epsilon}_{M,0} \le \overline{\tilde{q}}_{M,0}, && \text{(2.14d)}\\
& \sum_{j=0}^{m-1} \tilde{s}_{l,j} = \Delta S_l, l = 1\ldots, M, \quad j = 0,\ldots,3, && \\
& \tilde{q}_{0,0} = v_0^2, && \text{(2.14e)}\\
& \tilde{q}_{M,0} = v_f^2, && \text{(2.14f)}\\
& \frac{1}{mM+1}\left(\sum_{l=1}^{M}\sum_{j=0}^{m-1} \tilde{q}_{l-1,j} + \tilde{q}_{M,0}\right) = \tilde{c}_f. && \text{(2.14g)}
\end{aligned}
$$

Since $\tilde{a}_1 = 0$, the optimization problem (2.14) can be reduced by eliminating constant speed mode from the optimization.

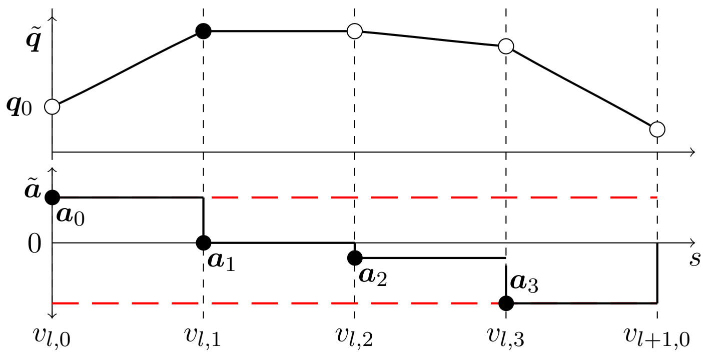


Figure 2.2: Four consecutive modes at each stage element l: 1) Acceleration; 2) Constant speed; 3) Zeros thrust; 4) Breaking.

The road must be separated into stage elements, over which road grade and speed limits are constant values. A stage element l can be defined as illustrated in Figure 2.2, each containing four consecutive driving modes (2.14b) - acceleration, constant speed, zero thrust, and breaking. The stage is defined by the following vectors and matrices

$$\tilde{\boldsymbol{q}}_l = \begin{bmatrix} \tilde{q}_{l-1,1} \\ \tilde{q}_{l-1,2} \\ \tilde{q}_{l-1,3} \\ \tilde{q}_{l,0} \end{bmatrix}, \; \tilde{\boldsymbol{\epsilon}}_l = \begin{bmatrix} \tilde{\epsilon}_{l-1,1} \\ \tilde{\epsilon}_{l-1,2} \\ \tilde{\epsilon}_{l-1,3} \\ \tilde{\epsilon}_{l,0} \end{bmatrix}, \; \Delta\tilde{\boldsymbol{s}}_l = \begin{bmatrix} \Delta\tilde{s}_{l-1,0} \\ \Delta\tilde{s}_{l-1,1} \\ \Delta\tilde{s}_{l-1,2} \\ \Delta\tilde{s}_{l-1,3} \end{bmatrix}, \; \tilde{\mathbf{1}}_l = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \in \mathcal{R}^m,$$

$$\tilde{\boldsymbol{A}}_l = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & 1 & \\ & & -1 & 1 \end{bmatrix}, \; \tilde{\boldsymbol{C}}_l = \begin{bmatrix} 0 & 0 & 0 & -1 \\ & 0 & 0 & 0 \\ & & 0 & 0 \\ & & & 0 \end{bmatrix} \in \mathcal{R}^{m\times m},$$

$$\tilde{\boldsymbol{B}}_l = \begin{bmatrix} 2\tilde{a}_0 & & & \\ & 2\tilde{a}_1 & & \\ & & 2\tilde{a}_2 & \\ & & & 2\tilde{a}_3 \end{bmatrix}, \; \tilde{\boldsymbol{S}}_l = \begin{bmatrix} \mathcal{B}_l, q(q_{l,0}, \Delta\tilde{s}_{l,0}) & & & \\ & \mathcal{B}_l, q(q_{l,0}, \Delta\tilde{s}_{l,1}) & & \\ & & 0 & \\ & & & 0 \end{bmatrix} \in \mathcal{R}^{m\times m},$$

and for the overall optimization problem is

$$\min \quad \frac{1}{2}\tilde{\boldsymbol{q}}^T\tilde{\boldsymbol{S}}^T\Delta\tilde{\boldsymbol{s}} + \frac{1}{2}\Delta\tilde{\boldsymbol{s}}^T\tilde{\boldsymbol{S}}\tilde{\boldsymbol{q}} + \frac{1}{2}\tilde{\boldsymbol{\epsilon}}^T\tilde{\boldsymbol{\epsilon}}, \tag{2.15a}$$

$$\text{s.t.} \quad \tilde{\boldsymbol{A}}\tilde{\boldsymbol{q}} = \tilde{\boldsymbol{B}}\Delta\tilde{\boldsymbol{s}}, \tag{2.15b}$$

$$\tilde{\boldsymbol{E}}\Delta\tilde{\boldsymbol{s}} = \tilde{\boldsymbol{h}}, \tag{2.15c}$$

$$\underline{\tilde{\boldsymbol{q}}} \leq \tilde{\boldsymbol{q}} + \tilde{\boldsymbol{\epsilon}} \leq \overline{\tilde{\boldsymbol{q}}} \tag{2.15d}$$

$$\tilde{\boldsymbol{e}}_0^T\tilde{\boldsymbol{q}} = \tilde{v}_0^2, \tag{2.15e}$$

$$\tilde{\boldsymbol{e}}_{mM+1}^T\tilde{\boldsymbol{q}} = \tilde{v}_f^2, \tag{2.15f}$$

$$\tilde{\mathbf{1}}^T\tilde{\boldsymbol{q}} = (mM+1)\tilde{c}_f, \tag{2.15g}$$

where the related stacked vectors and matrices composed of stage vectors and matrices are

$$\tilde{\boldsymbol{q}} = \begin{bmatrix} \tilde{q}_0 \\ \tilde{\boldsymbol{q}}_1 \\ \vdots \\ \tilde{\boldsymbol{q}}_M \end{bmatrix},\ \tilde{\boldsymbol{\epsilon}} = \begin{bmatrix} \tilde{\epsilon}_0 \\ \tilde{\boldsymbol{\epsilon}}_1 \\ \vdots \\ \tilde{\boldsymbol{\epsilon}}_M \end{bmatrix},\ \tilde{\mathbf{1}} = \begin{bmatrix} 1 \\ \tilde{\mathbf{1}}_1 \\ \vdots \\ \tilde{\mathbf{1}}_M \end{bmatrix},\ \underline{\tilde{\boldsymbol{q}}} = \begin{bmatrix} \underline{\tilde{q}}_0 \\ \underline{\tilde{\boldsymbol{q}}}_1 \\ \vdots \\ \underline{\tilde{\boldsymbol{q}}}_M \end{bmatrix},\ \bar{\tilde{\boldsymbol{q}}} = \begin{bmatrix} \bar{\tilde{q}}_0 \\ \bar{\tilde{\boldsymbol{q}}}_1 \\ \vdots \\ \bar{\tilde{\boldsymbol{q}}}_M \end{bmatrix} \in \mathcal{R}^{mM+1},$$

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} -\tilde{\boldsymbol{e}}_0 & \tilde{\boldsymbol{A}}_1 & & & \\ & \tilde{\boldsymbol{C}}_2 & \tilde{\boldsymbol{A}}_2 & & \\ & & \ddots & \ddots & \\ & & & \tilde{\boldsymbol{C}}_M & \tilde{\boldsymbol{A}}_M \end{bmatrix},\ \tilde{\boldsymbol{S}} = \begin{bmatrix} \tilde{\boldsymbol{S}}_1 & & & & \\ & \tilde{\boldsymbol{S}}_2 & & & \\ & & \ddots & & \\ & & & \tilde{\boldsymbol{S}}_M & \mathbf{0} \end{bmatrix} \in \mathcal{R}^{mM\times(mM+1)},$$

$$\tilde{\boldsymbol{B}} = \begin{bmatrix} \tilde{\boldsymbol{B}}_1 & & & \\ & \tilde{\boldsymbol{B}}_2 & & \\ & & \ddots & \\ & & & \tilde{\boldsymbol{B}}_M \end{bmatrix} \in \mathcal{R}^{mM\times mM},\ \Delta\tilde{\boldsymbol{s}} = \begin{bmatrix} \Delta\tilde{\boldsymbol{s}}_1 \\ \vdots \\ \Delta\tilde{\boldsymbol{s}}_M \end{bmatrix} \in \mathcal{R}^{mM},$$

$$\tilde{\boldsymbol{E}} = \begin{bmatrix} \tilde{\mathbf{1}}_1^T & & \\ & \ddots & \\ & & \tilde{\mathbf{1}}_M^T \end{bmatrix} \in \mathcal{R}^{M\times mM},\ \tilde{\boldsymbol{h}} = \begin{bmatrix} \Delta\tilde{S}_1 \\ \vdots \\ \Delta\tilde{S}_M \end{bmatrix} \in \mathcal{R}^{M}.$$

2.4 Condensed Sparse Problem Formulation

Following the idea in Mancuso and Kerrigan [2011], one can eliminate out input variables ($\Delta\tilde{\boldsymbol{s}}$) to condense the optimization. Only state variables ($\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{\epsilon}}$) remains. To eliminate the input variables, we use (2.15b), from which we can write

$$\Delta\tilde{\boldsymbol{s}} = \tilde{\boldsymbol{P}}\tilde{\boldsymbol{q}}, \quad \tilde{\boldsymbol{P}} = \tilde{\boldsymbol{B}}^{-1}\tilde{\boldsymbol{A}}.$$

Use of this approach is rare since it requires an inverse of $\tilde{\boldsymbol{B}}$. When we first eliminate the constant speed mode from the optimization (due to $\tilde{a}_1 = 0$), the inversion exists, and condensing can proceed. Further, since in our case $\tilde{\boldsymbol{B}}$ is diagonal, then the inverse is a diagonal matrix as well, and $\tilde{\boldsymbol{P}}$ is sparse (bi-diagonal). Hence, variable elimination does not ruin the resulting optimization problem sparsity, yet the number of variables decreases significantly.

The resulting optimization problem, with a stacked vector of optimization variables

$$\tilde{\mathbb{z}} = \left[\tilde{\boldsymbol{q}}^T, \tilde{\boldsymbol{\epsilon}}^T\right]^T \in \mathcal{R}^{2(mM+1)}$$

is the standard mathematical program with affine equality constraint as follows

$$\begin{aligned} \min \quad & \frac{1}{2}\tilde{\mathbb{z}}^T \tilde{\mathbb{H}} \tilde{\mathbb{z}} + \tilde{\mathbb{f}}^T \tilde{\mathbb{z}}, && \text{(2.16a)} \\ \text{s.t.} \quad & \underline{\tilde{\mathbb{b}}} \le \tilde{\mathbb{A}} \tilde{\mathbb{z}} \le \overline{\tilde{\mathbb{b}}}, && \text{(2.16b)} \\ & \underline{\tilde{\mathbb{z}}} \le \tilde{\mathbb{z}} \le \overline{\tilde{\mathbb{z}}}, && \text{(2.16c)} \end{aligned}$$

where

$$\tilde{\mathbb{H}} = \begin{bmatrix} \frac{(\tilde{\boldsymbol{S}}^T \tilde{\boldsymbol{P}} + \tilde{\boldsymbol{P}}^T \tilde{\boldsymbol{S}})}{2} & \\ & \boldsymbol{I} \end{bmatrix} \in \mathcal{R}^{2(mM+1)\times 2(mM+1)},$$

$$\tilde{\mathbb{f}} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \; \underline{\tilde{\mathbb{z}}} = \begin{bmatrix} -\infty \\ -\infty \end{bmatrix}, \; \overline{\tilde{\mathbb{z}}} = \begin{bmatrix} \infty \\ \infty \end{bmatrix} \in \mathcal{R}^{2(mM+1)},$$

$$\tilde{\mathbb{A}} = \begin{bmatrix} \tilde{\boldsymbol{P}} & \mathbf{0} \\ \boldsymbol{I} & \boldsymbol{I} \\ \tilde{\boldsymbol{E}}\tilde{\boldsymbol{P}} & \mathbf{0} \\ \tilde{\boldsymbol{e}}_0^T & \mathbf{0}^T \\ \tilde{\boldsymbol{e}}_{nM+1}^T & \mathbf{0}^T \\ \tilde{\mathbf{1}}^T & \mathbf{0}^T \end{bmatrix} \in \mathcal{R}^{(3mM+4)\times 2(mM+1)},$$

$$\underline{\tilde{\mathbb{b}}} = \begin{bmatrix} \mathbf{0} \\ \underline{\tilde{\boldsymbol{q}}} \\ \tilde{\boldsymbol{h}} \\ \tilde{v}_0^2 \\ \tilde{v}_f^2 \\ (mM+1)\tilde{c}_f \end{bmatrix}, \; \overline{\tilde{\mathbb{b}}} = \begin{bmatrix} \infty \\ \overline{\tilde{\boldsymbol{q}}} \\ \tilde{\boldsymbol{h}} \\ \tilde{v}_0^2 \\ \tilde{v}_f^2 \\ (mM+1)\tilde{c}_f \end{bmatrix} \in \mathcal{R}^{3mM+4}.$$

The first affine constraint comes from the originally simple constraint $\Delta\tilde{s} \ge 0$. If positive definiteness of the $\tilde{\mathbb{H}}$ in (2.16) is required, a common regularization technique, adding a small positive diagonal matrix, can be used. However, problem (2.16) is well-defined and has a unique solution. Therefore, no regularization is typically needed.

Example of the optimization result is shown in Figure 2.3.

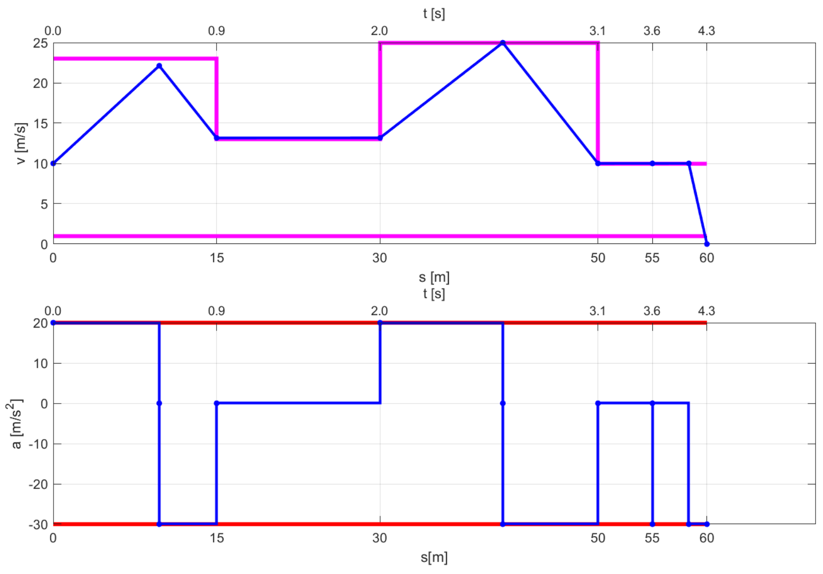


Figure 2.3: Speed and acceleration profiles in the spatial and time domain sampled in acceleration.

2.5 Finding c_f Using Bisection Method

The following idea will be proved: If we choose some final time $t_f > 0$ and find an optimal speed profile, there is a unique $c_f > 0$ corresponding to that speed profile. If we increase t_f, the corresponding c_f is lower and vice versa. Moreover, if we increase t_f, the speed at every moment will be the same or decrease compared to the previous optimal speed profile. If this is true, one can formulate the optimization problem using c_f, which results in a less complex problem. Yet, due to the correspondence between c_f and t_f, one can find such a c_f, which results in a speed profile with desired t_f.

Let us start with the idea described in the lemma below.

Lemma 2.1. *(Increase of c_f in standalone equation)*
Increase of c_f in equation $\frac{1}{N+1}\sum_k q_k = c_f$, where $q_k = v_k^2$ leads to a change $\Delta v_k, k = 0, \ldots, N$ for which holds $\sum_{k=0}^{N} \Delta v_k > 0$. The change magnitude is assumed to be constrained as $|\Delta v_k| < 2\tilde{v}_k, \Delta v_k, k = 0, \ldots, N$.

Proof.

$$
\begin{aligned}
\frac{1}{N}\sum_{k=0}^{N} v_k^2 &< \frac{1}{N}\sum_{k=0}^{N}(v_k + \Delta v_k)^2 \\
\frac{1}{N}\sum_{k=0}^{N} v_k^2 &< \frac{1}{N}\sum_{k=0}^{N} v_k^2 + 2\tilde{v}_k \Delta v_k + \Delta v_k^2 \\
0 &< \frac{1}{N}\sum_{k=0}^{N} \Delta v_k(2\tilde{v}_k + \Delta v_k) \\
0 &< \sum_{k=0}^{N} \Delta v_k
\end{aligned}
$$

□

The condition coming from Lemma 2.1 needs to be revised to establish a tight mapping between parameters t_f and c_f in time and spatial domain, respectively. An optimization context is essential to establish a tight connection between the two. The following lemma has more restriction on the change Δv_k.

Lemma 2.2. *(Increase of c_f in optimization context)*
For increased c_f in optimization (2.16) there is an optimal solution obtained by change $\Delta v_k \geq 0, k = 0, \dots, N$.

Proof. (see Figure 2.4)
Assume the optimal solution of (2.16) for given $c_f^\star$ is known. Also, assume another optimal solution of (2.16) for given c_f^+ such that $c_f^+ > c_f^\star$ is known and in this solution there is at least one instance of k such that $\Delta v_k < 0$. Then, a location s_e exists on the road, where speeds v_e^+ and $v_e^\star$ are equal. Now, one can switch the control policy and apply v^* instead of v^+ since v^* is an optimal strategy from s_e to s_f. It is valid due to the invariance in c. Therefore, we can move an optimal solution up and down in Figure 2.4. Also, due to convexity, there must be optimal trajectories within the red area finishing within the range $[c_f^+, c_f^\star]$. Due to the invariance in c, we can move the area up, enhancing the blue trajectory, where Δv_k are nonnegative. Hence, we can argue that an optimal trajectory exists to c_f^+ without the need for $\Delta v_k < 0$ for any k. □

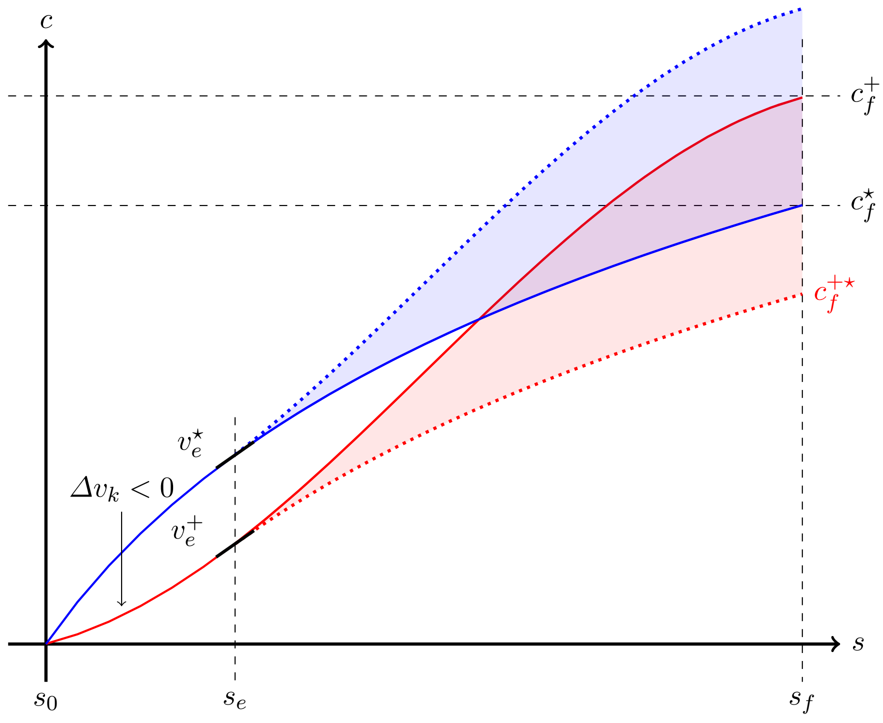


Figure 2.4: Sketch of the proof for Lemma 2.2. *Blue bold curve is an optimal trajectory of $c^\star$ for given $c_f^\star$. The red bold curve is an optimal trajectory of c^+ for given c_f^+ with some k for which $\Delta v_k < 0$. Due to invariance in c, we can construct another optimal trajectory $c^+\star$ ending in $c^+\star_f$, and we know it must be optimal too. Due to convexity, the red transparent area must contain optimal trajectories ending in the range of final values. We can translate the red area into the blue area - optimal solutions starting in s_e from the $c^\star$. Since the c_f^+ is within the blue area, an optimal trajectory must exist without using negative speed at the beginning.*

Theorem 2.3. *Given a final time $t_f > 0$ a positive sequence of $v_0, v_1, \dots, v_N$ exists such that $\sum_{k=0}^{N} \frac{1}{v_k} \Delta s_k = t_f$. Moreover, there exists a c_f such that $\frac{1}{N} \sum_{k=0}^{N} v_k^2 = c_f$ holds in the optimization framework. Increase of c_f implies decrease of t_f and vice versa.*

Proof. Increase of c_f means there is a sequence of $\Delta v_k \geq 0, k = 0, \dots, N$ (see Lemma 2.2) such that

$$\begin{aligned}
\bar{v}_k + \Delta\bar{v}_k &\geq \bar{v}_k \\
\frac{\bar{v}_k + \Delta\bar{v}_k}{\bar{v}_k(\bar{v}_k + \Delta\bar{v}_k)} &\geq \frac{\bar{v}_k}{\bar{v}_k(\bar{v}_k + \Delta\bar{v}_k)} \\
\frac{1}{\bar{v}_k} &\geq \frac{1}{\bar{v}_k + \Delta\bar{v}_k}, k = 0, \dots, N \\
\sum_{k=0}^{N} \frac{1}{\bar{v}_k} &> \sum_{k=0}^{N} \frac{1}{\bar{v}_k + \Delta\bar{v}_k},
\end{aligned}$$

which implies decrease of t_f and vice versa. □

The exact value of c_f for given t_f is unknown analytically and needs to be computed numerically. However, due to Theorem 2.3, we know that increasing c_f decreases t_f and vice versa. Therefore, a bisection algorithm is used to solve

$$\Psi(c_f) = t_f, \tag{2.17}$$

where the lower and upper bounds of c_f can be estimated for a known value of average speed $v_{\text{avg}} = s_f/t_f$. The lower estimate

$$\underline{c}_f = \frac{1}{N+1} \sum_{k=1}^{N} q_k = v_{\text{avg}}^2$$

is acheived when we set $q_k = v_{\text{avg}}^2$ to be a constant for all k. The complicating case is when the upper (or lower) speed limit crosses the average value. In that case, we need to adjust to

$$\underline{c}_f = \min v_{\text{avg}}^2, \min v_{\text{max}}.$$

On the other hand, an upper estimate

$$\bar{c}_f = \frac{1}{N+1}\sum_{k=1}^{N} q_k = \left(2v_{\text{avg}}\right)^2$$

is achieved when q_k is oscillating around the average value v_{avg} (and above zero value) as much as possible. Since we assume soft constraints on the speed limit, in theory, the maximum oscillations' peak-to-peak magnitude is $2v_{\text{avg}}$.

In other words, the lower estimate $\underline{c}_f$ is given by the square of the average value (imagine constant function). In contrast, the upper estimate of $\bar{c}_f$ is given by the average square values (imagine the function oscillating between the two values).

The two approaches from the previous sections, namely, optimization sampled in space and acceleration, are shown in Figure 2.5 and Figure 2.6, respectively, together with the intermediate iterates of the bisection methods.

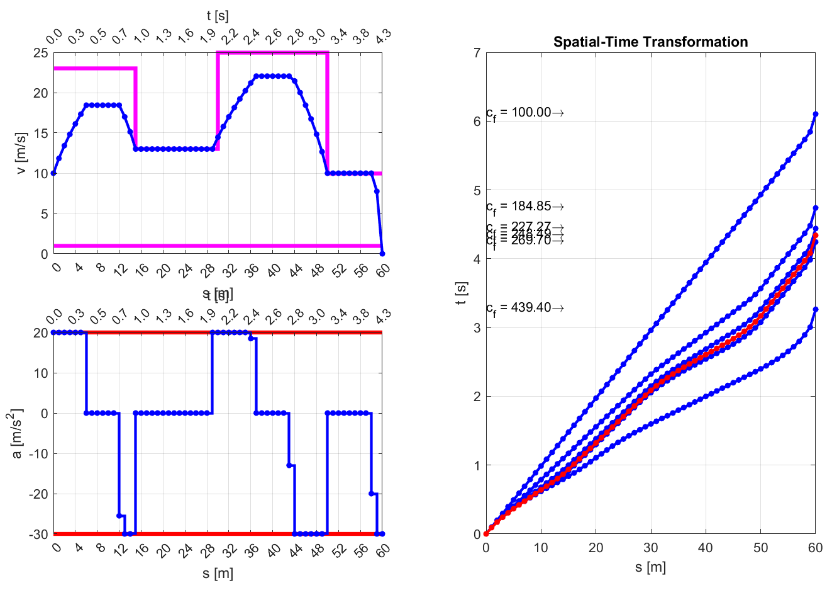


Figure 2.5: Speed and acceleration profiles (left) in the spatial and time domain sampled in space. Spatial-time transformations (right) for intermediate (blue) and resulting (red) values of c_f.

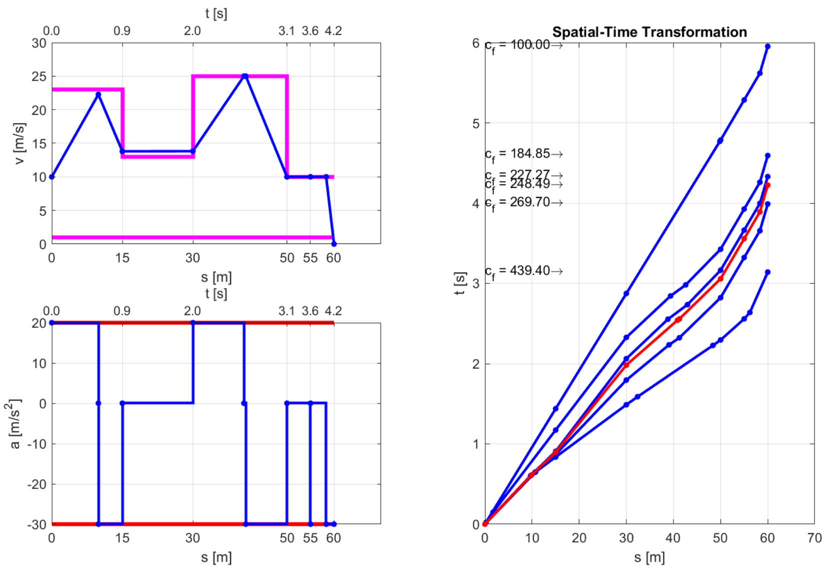


Figure 2.6: Speed and acceleration profiles (left) in the spatial and time domain sampled in acceleration. Spatial-time transformations (right) for intermediate (blue) and resulting (red) values of c_f.

Convergence of the bisection method is similar for both cases, and convergence in detail for the example is shown in Figure 2.7.

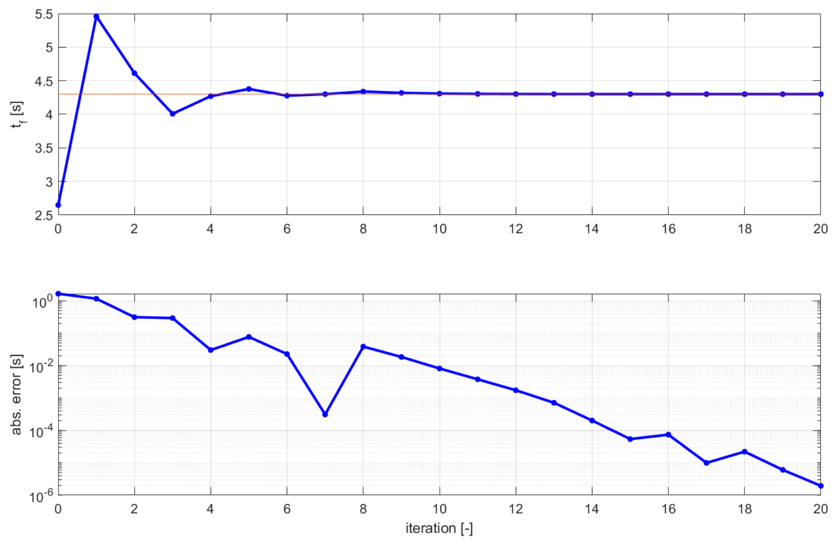


Figure 2.7: Convergence of the bisection method finding c_f towards given t_f on top. The absolute error of the estimated time from the t_f on the bottom.

In this chapter, Challenge 3, Challenge 4, and Challenge 5 have been succesfuly addressed. Therefore, the control problem can be solved via well-defined QP instead of NLP, which iteration can be infeasible due to the undefined value of the objective function when crossing zero speed.

2.6 Distributed Speed Profile Optimization

This section will describe the distributed optimization of multiple consecutive profiles using a decomposition method. A distributed approach benefits long horizons, which can be split into multiple smaller optimizations. It becomes beneficial for traffic light optimization in the next section.

Due to its potential application in automotive, primal decomposition is preferred, i.e., the primal variable is the complicating variable common for two neighbouring sub-problems. The main reason is feasibility preservation, such that after every single iteration, the solution is feasible and can be (potentially) applied, even though not optimal.

The first attempt to coordinate the speed profiles was made with the subgradient method, which convergence can be seen in Figure 2.8. This approach is robust and straightforward (see, Nedic and Ozdaglar [2009]). However, it requires too many (moderately expansive) iterations. Hence, we are looking for a more efficient coordinator. The coordinator proposed further is motivated by the convergence figures of the subgradient methods. We can see the subgradient converge in a piecewise linear manner. Activating a new constraint causes each breakpoint in the piecewise linear lines. Therefore, our efficient coordinator utilizes those breakpoints to accelerate the convergence. These can be analyzed from the optimality condition, and the following approach is related to the Optimality Condition Decomposition (OCD) known from distributed model predictive control (e.g., Segovia et al. [2021]).

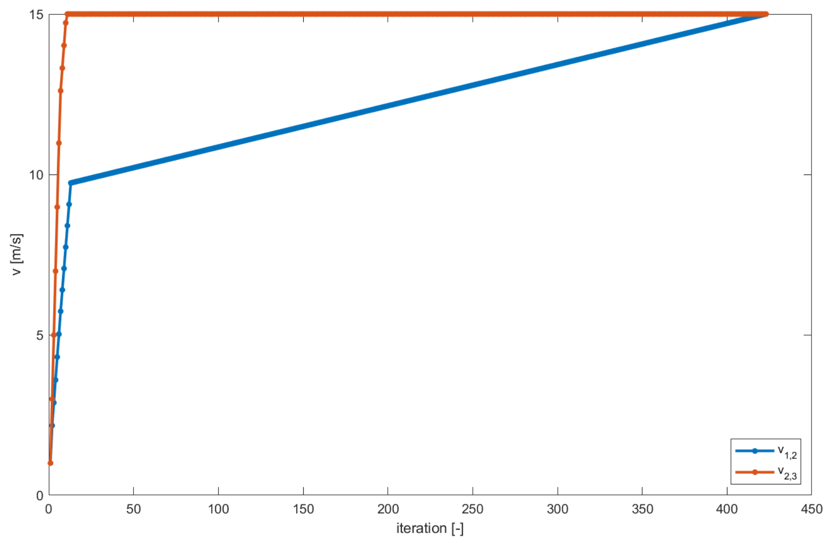


Figure 2.8: Convergence of primal decomposition with the subgradient coordination method from initial conditions $v_{1,2} = 1$ and $v_{2,3} = 1$.

Assume n consequtive optimization problems, where kth optimization problem in form (2.16) can be rewritten as

$$\min \quad \frac{1}{2}\begin{bmatrix}\tilde{\mathbb{z}}' \\ \tilde{\mathbb{p}}\end{bmatrix}^T \begin{bmatrix}\tilde{\mathbb{H}}_{k,z'z'} & \tilde{\mathbb{H}}_{k,z'p} \\ \tilde{\mathbb{H}}_{k,pz'} & \tilde{\mathbb{H}}_{k,pp}\end{bmatrix}\begin{bmatrix}\tilde{\mathbb{z}}' \\ \tilde{\mathbb{p}}\end{bmatrix} + \begin{bmatrix}\tilde{\mathbb{z}}' \\ \tilde{\mathbb{p}}\end{bmatrix}^T \begin{bmatrix}\tilde{\mathbb{f}}_{k,z'} \\ \tilde{\mathbb{f}}_{k,p}\end{bmatrix}, \tag{2.18a}$$

$$\text{s.t.} \quad \underline{\tilde{\mathbb{b}}}_k \leq \tilde{\mathbb{A}}_{k,z'}\tilde{\mathbb{z}}' + \tilde{\mathbb{A}}_{k,p}\tilde{\mathbb{p}} \leq \overline{\tilde{\mathbb{b}}}_k, \tag{2.18b}$$

where $\tilde{\mathbb{p}}$ stands for parameters. For simplicity, the affine constraints also cover the simple bounds that are not written explicitly. The coordinator needs to collect information from the kth sub-problem about the parameter sensitivity and range the parameter can change. Knowing the optimal solutions

$$\tilde{\mathbb{z}}_k^\star = \begin{bmatrix} \tilde{\boldsymbol{q}}_k^\star \\ \tilde{\boldsymbol{\epsilon}}_k^\star \end{bmatrix},$$

of the kth sub-problems, where

$$\tilde{\boldsymbol{q}}_k^\star = \begin{bmatrix} \tilde{q}_{k,0} \\ \tilde{\boldsymbol{q}}_k' \\ \tilde{q}_{k,f} \end{bmatrix}, \quad \tilde{\mathbb{p}}_k = \begin{bmatrix} \tilde{q}_{k,0} \\ \tilde{q}_{k,f} \end{bmatrix}, \quad \tilde{\mathbb{z}}_k' = \begin{bmatrix} \tilde{\boldsymbol{q}}_k' \\ \tilde{\boldsymbol{\epsilon}}_k^\star \end{bmatrix},$$

and where problem (2.18) can be written as a program

$$\begin{aligned} \min \quad & \frac{1}{2}\tilde{\mathbb{p}}^T \hat{\mathbb{H}}_{k,pp}\tilde{\mathbb{p}} + \tilde{\mathbb{p}}^T \hat{\mathbb{f}}_{k,p}, & (2.19a)\\ \text{s.t.} \quad & \underline{\hat{\mathbb{b}}}_k \le \hat{\mathbb{A}}_{k,p}\tilde{\mathbb{p}} \le \overline{\hat{\mathbb{b}}}_k, & (2.19b)\\ & \underline{\hat{\mathbb{p}}}_k \le \tilde{\mathbb{p}}_k \le \overline{\hat{\mathbb{p}}}_k, & (2.19c) \end{aligned}$$

which give the coordinator the information how the parameters of one sub-problem can be tuned. We can analyze the affine constraints in the solution to get two equations and inequality

$$\begin{aligned} \tilde{\mathbb{A}}_{k,z'}^{+}\tilde{\mathbb{z}}' + \tilde{\mathbb{A}}_{k,p}^{+}\tilde{\mathbb{p}} &= \overline{\tilde{\mathbb{b}}}_k^{+}, \quad \tilde{\boldsymbol{\mu}}^{+} > 0, & (2.20a)\\ \underline{\tilde{\mathbb{b}}}_k^{=} \le \tilde{\mathbb{A}}_{k,z'}^{=}\tilde{\mathbb{z}}' + \tilde{\mathbb{A}}_{k,p}^{=}\tilde{\mathbb{p}} &\le \overline{\tilde{\mathbb{b}}}_k^{=}, \quad \tilde{\boldsymbol{\mu}}^{=} = 0, & (2.20b)\\ \tilde{\mathbb{A}}_{k,z'}^{-}\tilde{\mathbb{z}}' + \tilde{\mathbb{A}}_{k,p}^{-}\tilde{\mathbb{p}} &= \underline{\tilde{\mathbb{b}}}_k^{-}, \quad \tilde{\boldsymbol{\mu}}^{-} < 0, & (2.20c) \end{aligned}$$

where $\tilde{\mu}$ are Lagrange multipliers related to the affine constraints (2.18b). From equations (2.20a) and (2.20c) we can express

$$\tilde{\mathbb{z}}' = \mathbb{r} - \mathbb{Q}\tilde{\mathbb{p}}, \text{ where } \mathbb{Q} = \begin{bmatrix} \tilde{\mathbb{A}}_{k,z'}^{+} \\ \tilde{\mathbb{A}}_{k,z'}^{-} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\mathbb{A}}_{k,p}^{+} \\ \tilde{\mathbb{A}}_{k,p}^{-} \end{bmatrix}, \text{ and } \mathbb{r} = \begin{bmatrix} \tilde{\mathbb{A}}_{k,z'}^{+} \\ \tilde{\mathbb{A}}_{k,z'}^{-} \end{bmatrix}^{-1} \begin{bmatrix} \overline{\tilde{\mathbb{b}}}_k^{+} \\ \underline{\tilde{\mathbb{b}}}_k^{-} \end{bmatrix}. \tag{2.21}$$

Since the solution of the linear program has activated as many constraints as the number of optimization variables, inversion inside $\mathbb{Q}$ exists. Further, substituting $\tilde{\mathbb{z}}'$ in inequality (2.20b) and cost (2.18a), we can define sub-problem data from (2.19) to be communicated to the coordinator as

$$\hat{\mathbb{H}}_{k,pp} = \tilde{\mathbb{H}}_{k,pp} - \mathbb{Q}^T \tilde{\mathbb{H}}_{k,z'p} - \hat{\mathbb{H}}_{k,pz'} \mathbb{Q} - \mathbb{Q}^T \hat{\mathbb{H}}_{k,z'z'} \mathbb{Q}, \tag{2.22a}$$

$$\hat{\mathbb{f}}_{k,p} = \tilde{\mathbb{f}}_{k,p} - \mathbb{Q}^T \tilde{\mathbb{f}}_{k,z'} - \hat{\mathbb{H}}_{k,pz'} \mathbb{r} - \mathbb{Q}^T \hat{\mathbb{H}}_{k,z'z'} \mathbb{r}, \tag{2.22b}$$

$$\hat{\mathbb{A}}_{k,p} = \tilde{\mathbb{A}}^{=}_{k,p} - \tilde{\mathbb{A}}^{=}_{k,z} \mathbb{Q}, \tag{2.22c}$$

$$\underline{\hat{\mathbb{b}}}_k = \underline{\tilde{\mathbb{b}}}^{=}_k - \tilde{\mathbb{A}}^{=}_{k,z} \mathbb{r}, \tag{2.22d}$$

$$\overline{\hat{\mathbb{b}}}_k = \overline{\tilde{\mathbb{b}}}^{=}_k - \tilde{\mathbb{A}}^{=}_{k,z} \mathbb{r}. \tag{2.22e}$$

Coordinator collect all sub-problems (2.19) and connects them by set of equalities $q_{k,f} = q_{k+1,0}, \; k = 0, \ldots, n-1$. This coordination scheme follows the approach proposed in Beno et al. [2017] proposed for distributed parameter calibration.

Convergence of the method (Figure 2.9) can be compared with the convergence of the subgradient method (Figure 2.8). For both, the resulting profiles are presented in Figure 2.10.

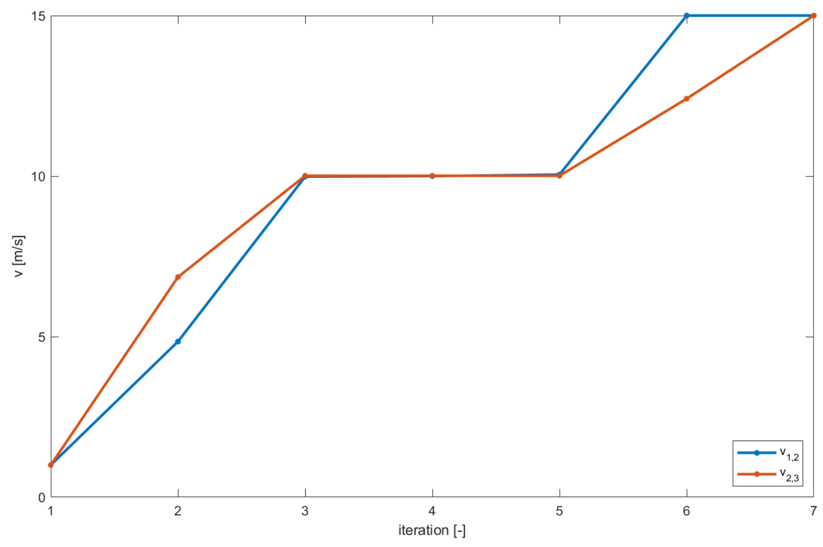


Figure 2.9: Convergence of primal decomposition with the OCD coordination method from initial conditions $v_{1,2} = 1$ and $v_{2,3} = 1$.

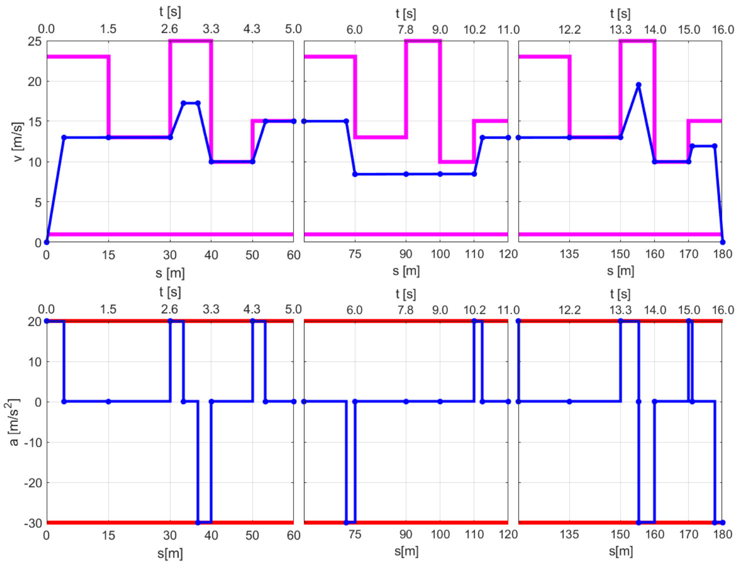


Figure 2.10: Three speed and acceleration profiles in the spatial (bottom axis) and time (top axis) domain of the optimization sampled in acceleration. The profiles are coordinated such that the terminal speed of the previous is identical to the initial speed of the following speed profile.

The overview of the algorithms involved in distributed speed profiles optimization for the example is sketched in Figure 2.11.

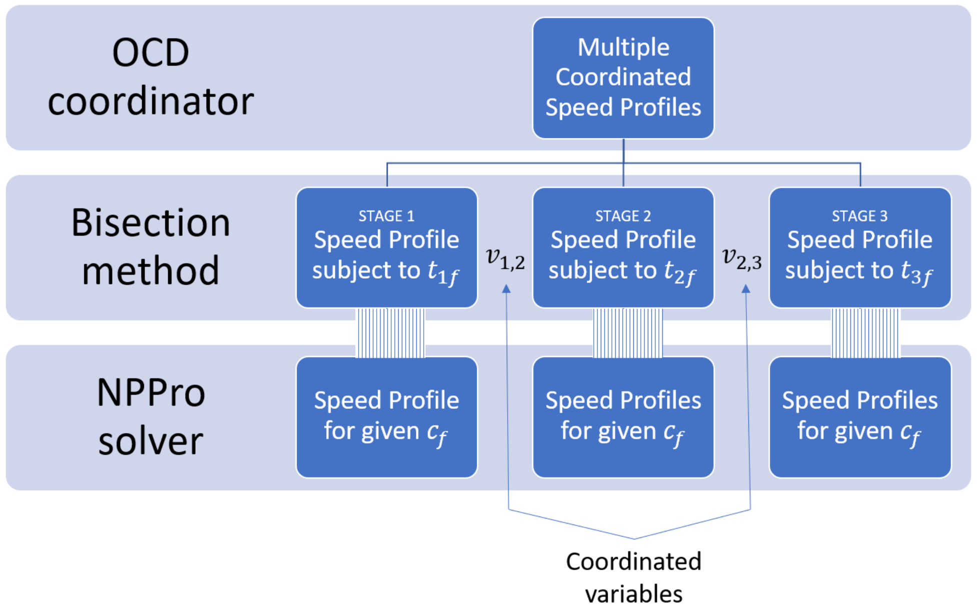

OCD coordinator
Multiple Coordinated Speed Profiles
Bisection method
STAGE 1
Speed Profile subject to t_{1f}
$v_{1,2}$
STAGE 2
Speed Profile subject to t_{2f}
$v_{2,3}$
STAGE 3
Speed Profile subject to t_{3f}
NPPro solver
Speed Profile for given c_f
Speed Profiles for given c_f
Speed Profiles for given c_f
Coordinated variables

Figure 2.11: Computational methods hierarchy in multiple coordinated speed profiles optimization. From the top, the OCD coordinator is described in this section, the bisection method is described in Sec. 2.5, and the NPPro solver shall be described in Chapter 3.

2.7 Traffic Lights Passage Optimization

An additional practical consideration is traffic lights. The traffic lights are assumed to publish time intervals when our vehicle can pass. There can be multiple traffic lights on the way. There is no assumption on the time intervals of the traffic lights, except that the intervals are (reasonably) finite.

The proposed algorithm is first described using the following example. More formally, an algorithm to solve the traffic light passage is given in Alg. 1. It is composed of three steps that are described further.

Note that adding a constraint on traffic light passage time in the global optimization directly would make the problem difficult as it would require multivariable bisection methods - there will be multiple constraints on time. Alg. 1 exploits distributed speed profile optimization from the previous section such that multiple coordinated speed profiles are optimized instead.

Algorithm 1 Traffic Light Passege Active Set (TLPAS) Method

```
Require: Given route with traffic light positions and time intervals in which passage through the traffic lights is allowed
  Set initial route considering no traffic lights
  for $i$ from 1 to $n_{iters}$ do
    Adding conflicting traffic light passages to the working set
    if Checking that no traffic light passage in the candidate is improper then
      return Candidate is the optimal solution
    else
      Removing preventing traffic light passages from the working set
    end if
  end for
```

Adding the conflicting traffic light passages to the working set

First, an optimization with only fixed terminal time is performed, ignoring all traffic lights. If there is no conflict between the solution and the given traffic light intervals, then we are done. The solution is the unconstrained (in the sense of the traffic lights) minimizer. This is not the case in our example Figure 2.12, where we have conflicts with the first and the third traffic lights. We pick one of them (e.g., the first) and resolve this conflict. The resolution is twofold – either our vehicle should speed up to catch the green light below the red crossed interval or slow down and go through once there is green above the crossed red interval – as shown in Figure 2.12. To be at the traffic light at a specific moment is an additional constraint in the two further optimization problems, whose solutions are shown in the Figure 2.13 and Figure 2.16. Therefore, each conflict resolution produces two children in a binary tree, where we look for the solution. Simultaneously, we have split the optimization horizon into two stages - 1) from the start to the first traffic light and 2) from the traffic light to the end. Since a depth first search algorithm is used, sub-problem p_1 (Figure 2.13) is resolved into p_2 (Figure 2.14) and p_3 (Figure 2.15). Finally, p_4 Figure 2.16 is optimized, and since the cost function of p_4 is the cheapest leaf, it is the optimal solution. The related tree is sketched in Figure 2.17.

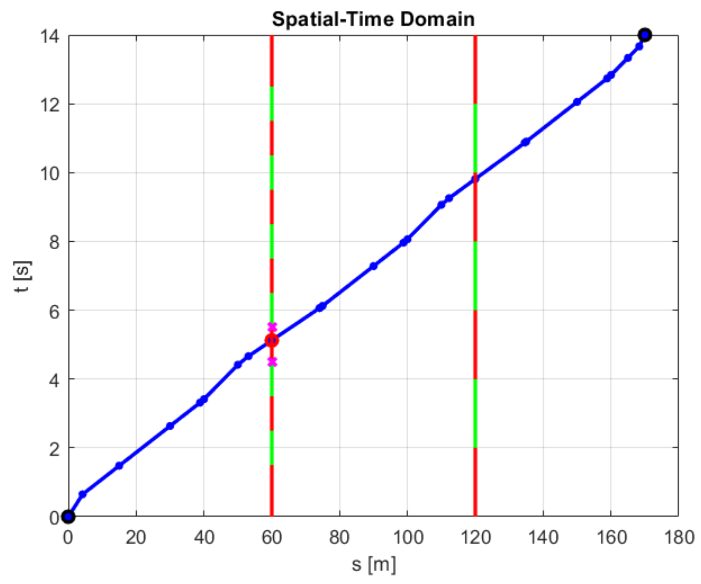


Figure 2.12: Solution of sub-problem p_0. *Traffic lights are not considered in the optimization.*

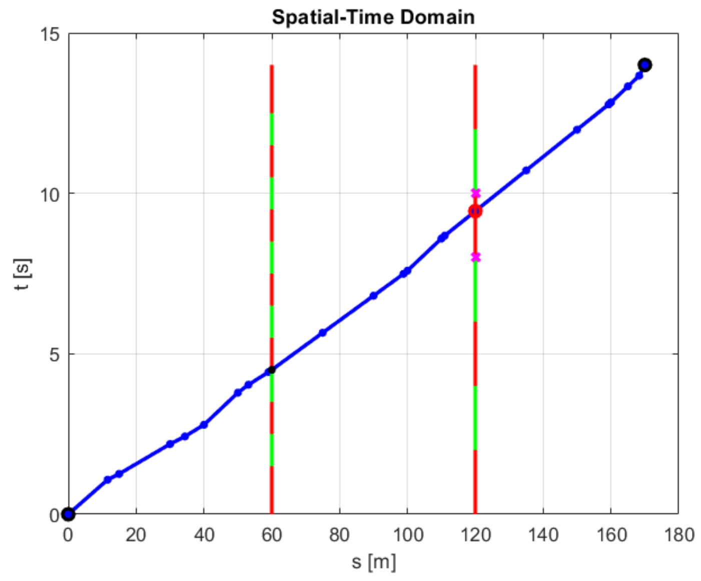


Figure 2.13: Solution of sub-problem p_1. Passage time for the first traffic light was considered in the optimization to pass before the red light.

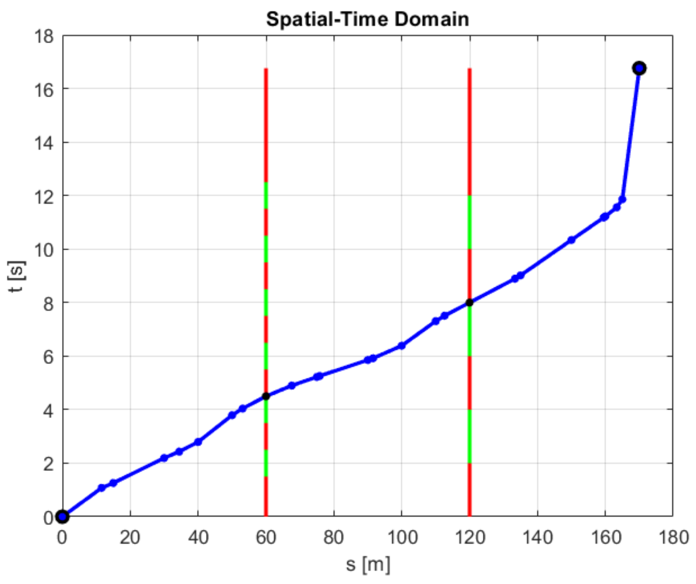


Figure 2.14: Solution of sub-problem p_2. Passage time for the first and second traffic lights was considered in the optimization to pass before and before the red light, respectively.

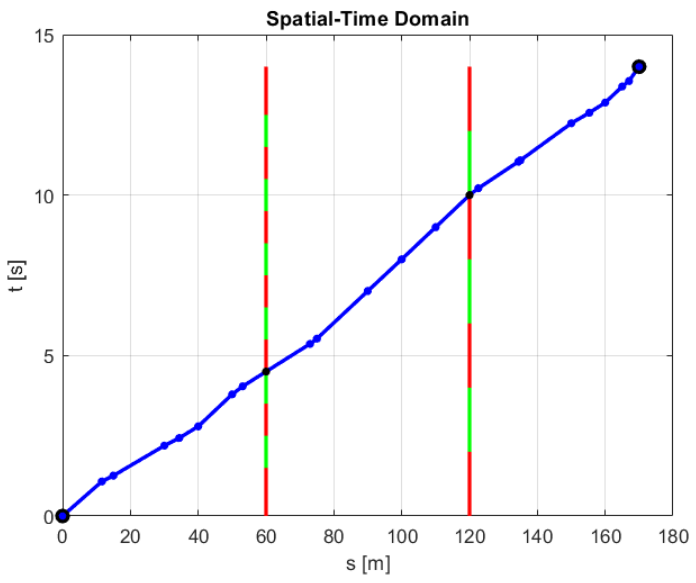


Figure 2.15: Solution of sub-problem p_2. Passage time for the first and second traffic lights was considered in the optimization to pass before and after the red light, respectively.

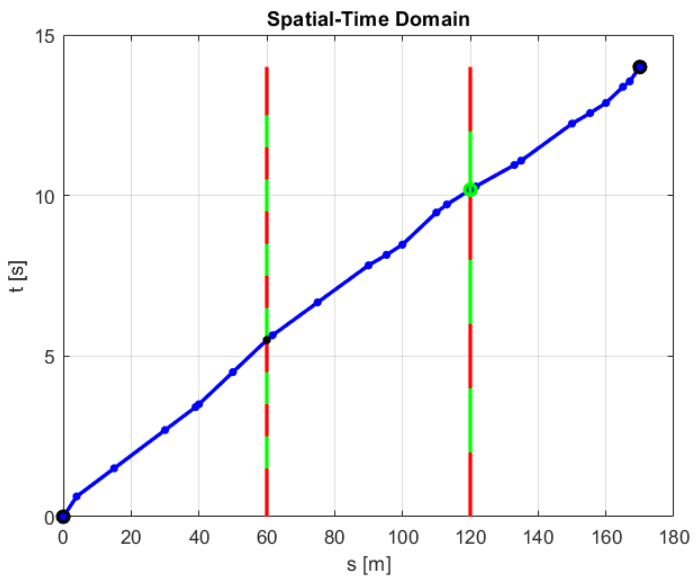


Figure 2.16: Solution of sub-problem p_4. Passage time for the first traffic light was considered in the optimization to pass before the red light.

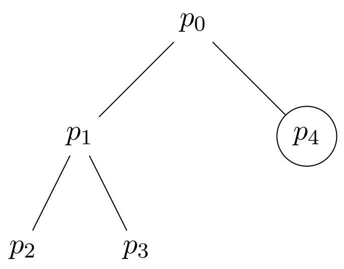


Figure 2.17: A tree represents the traffic lights passage problem. The solution is in the circle.

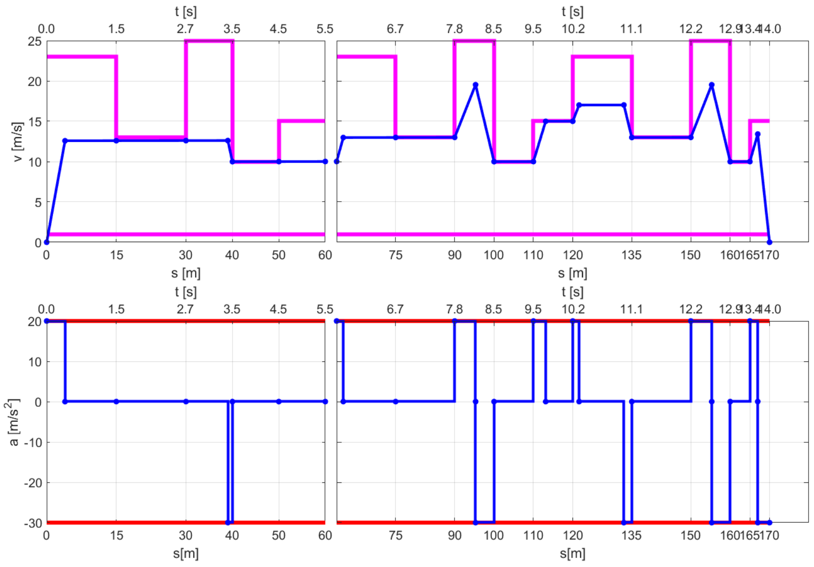


Figure 2.18: The speed and acceleration profiles in the spatial (bottom axis) and time (top axis) domain of the optimization sampled in acceleration for resulting sub-problem p_4.

Candidates with no conflicts (feasible trajectories) are computed, and (tree leaves) are compared. Eventually, a candidate with minimum cost is found. Further, once a cost function of a leaf is obtained, it can be used as an upper-cost estimate, and the remaining tree can be pruned to avoid unneeded computations.

Checking for the Optimality

Adding traffic lights can be seen as activating constraints within the active-set algorithm. The optimality must be checked once the best candidate for the active set is found. Why? A traffic light added to the working set may be no longer needed. It was initially added because it was hitting the red interval. However, due to another traffic light, the trajectory might shift enough that some traffic light is no longer needed. It prevents the minimizer from passing the green interval's interior. Therefore, the optimality requires checking that no blocking traffic lights are present in the optimum. Otherwise, the result is not optimal.

Hence, the stopping criteria verify the Lagrange multipliers $\lambda_{t,k}$ and $\lambda_{t,k+1}$ for equality constraint in the form (2.17) for the interval before and after kth traffic light. Since from the optimization, the available information is related to the average quadratic speed (see, (2.14g)) in $\lambda_{c,k}$ and $\lambda_{c,k+1}$, the conversion can be established as

$$\lambda_{t,k} = \frac{\partial \Psi_k(c_{f,k})}{\partial c_{f,k}} \lambda_{c,k} \tag{2.23}$$

Then the multiplier $\kappa_{t,k}$ for the kth traffic light saying if it is beneficial to arrive rather sooner or later can be established as follows. Assume two consecutive intervals over which the cost function is a simple sum

$$J_k^+ = J_k(t_k) + J_{k+1}(t_{k+1})$$

then, the perturbation of the traffic light in the time domain leads to

$$J_k^+ + \Delta J_k^+ = J_k(t_k + \epsilon) + J_{k+1}(t_{k+1} - \epsilon)$$

and by differentiation and further manipulation, we obtain

$$\begin{aligned}
\Delta J_k^+ &= \frac{\partial J_k(t_k)}{\partial t_k}\epsilon - \frac{\partial J_{k+1}(t_{k+1})}{\partial t_{k+1}}\epsilon \\
\kappa_{t,k} &= \lambda_{t,k} - \lambda_{t,k+1} \\
&= \frac{\partial \Psi(c_{f,k})}{\partial c_{f,k}}\lambda_{c,k} - \frac{\partial \Psi(c_{f,k+1})}{\partial c_{f,k+1}}\lambda_{c,k+1},
\end{aligned}$$

where $\lambda_{t,k} = \frac{\partial J_k(t_k)}{\partial t_k}$ and $\lambda_{t,k+1} = \frac{\partial J_{k+1}(t_{k+1})}{\partial t_{k+1}}$, and the final line was achieved using (2.23). Sign of the multiplier $\kappa_{t,k}$ express the intention to arrive at the traffic light sooner or later. The solution is optimal if the multiplier suggests the time should be increased (or decreased), and a red interval prevents this change. If, on the other hand, the multipliers suggest the duration time should be increased (or decreased), and a green interval allows the change to be done, the solution cannot be optimal. The latter case means the traffic light is preventing constraints and needs to be removed, as described next.

Removing the preventing traffic light passages from the working set

Since the candidate is not optimal, there are constraints (at least one) preventing reaching the optimum. A multiplier $\kappa_{t,k}$ related to the kth traffic light may suggest that releasing the constraint may improve cost. All constraints with such multipliers are, therefore, released at once.

3 Solution of QP with Affine Constraints

In this chapter, a solution for the core optimization problems presented in the previous chapter, namely, quadratic programming (QP), shall be presented to tackle some of the challenges that come with speed profile optimization problems.

The first attempt to deal with long horizons (Challenge 1) was published in [4], where the penalty method was applied to system dynamics to obtain sparse optimization problems suitable for long horizon control applications. Although the formulation is attractive for a relatively small number of optimization variables and a nice sparsity pattern, there are drawbacks. Since the formulation approximates the original problem, it is desired to approximate it by a high penalty parameter. High penalty parameters can cause numerical issues, however. In [2], we address Challenge 1 by an algorithm tailored for control problems with long horizons using sparse linear algebra.

In [3], an algorithm for the solution of Quadratic Programming (QP) with equality constraints is introduced as a predecessor. Finally, in this chapter, Challenge 2 has been resolved successfully by proposing an algorithm for QP with affine (inequality) constraints. Mainly considering [2] and [3] together, a solver aims to control problems with long horizons involving affine constraints.

We are concerned with the QP optimization problem of the following form

$$\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{f}, \tag{3.1a}$$
$$\text{s. t.} \quad \underline{\boldsymbol{x}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{x}}, \tag{3.1b}$$
$$\underline{\boldsymbol{y}} \leq \boldsymbol{A}\boldsymbol{x} \leq \overline{\boldsymbol{y}}, \tag{3.1c}$$

with a cost function given by positive definite Hessian matrix $\boldsymbol{H} \in \mathcal{R}^{n\times n}$, linear term $\boldsymbol{f} \in \mathcal{R}^n$, a box constraints given by $\underline{\boldsymbol{x}} \in \mathcal{R}^n$, $\overline{\boldsymbol{x}} \in \mathcal{R}^n$, an affine constraints given by $\boldsymbol{A} \in \mathcal{R}^{m\times n}$, $\underline{\boldsymbol{y}} \in \mathcal{R}^m$, and $\overline{\boldsymbol{y}} \in \mathcal{R}^m$.

There is an increasing use of optimization based on QP in the automotive community. The application includes Model Predictive Control (MPC), data fitting, autonomous driving, Support Vector Machine (SVM), contact problem, etc.

The most iconic is a repetitive solution of QP providing real-time control action, known as MPC. MPC is a control strategy that uses a mathematical model of a system to predict its future behaviour and optimize control actions. MPC has been widely applied in various applications, including thermal management [Karnik et al., 2016], energy management of power split [Sotoudeh and Homchaudhuri, 2020], predictive cruise control [D'Amato et al., 2016], and airpath control [Sankar et al., 2019]. It has been shown to improve system performance, increase flexibility, and handle uncertainty and disturbances. However, implementing MPC can be complex, with extensive memory and computational requirements. Despite these challenges, MPC has demonstrated promising results in applications and is expected to become the dominating control strategy [Bemporad et al., 2018], [Dahl et al., 2018].

A challenge regarding the solution to QP problems is conditioning the problem Hessian matrix. That could arise from the presence of nonlinearities in the system. To address the ill-conditioning of QP problems in the automotive domain, it is essential to carefully design the MPC system and choose an appropriate QP solver that can handle the specific characteristics of the problem. In addition, it may be necessary to use regularization or other techniques to improve the stability and convergence of the QP solver. Another challenge is the presence of constraints, which can limit the feasible region of the problem and make it more difficult to find the optimal solution. Especially when the active set changes rapidly, convergence may become an issue.

In this work, we present a QP solver developed with a focus on embedded applications. Our primary goal is to address difficulties arising in automotive in particular. We are motivated by applications of MPC with output limits that lead to QP with affine constraints. The workaround is to use the penalty method to eliminate the constraints. The drawback of this approximating approach is that the more precise the approximation is, the more ill-conditioned the Hessian matrix is. We are interested in solving the exact problem directly. Our priorities are reliability, memory resources, and speed in terms of the maximum number of executed iterations, which determines the solution time of the QP solver, which is crucial for the overall performance of the MPC on the embedded system. The resulting solver implements a method called Newton, Projection, and Proportioning (NPPro), whose name emphasizes it is the continuator of the Newton Projection with Proportioning (NPP) method [Šantin, 2016] extended for QP with generic constraints. Although the solver is compared with other solvers in double precision, it is primarily capable of running in single precision.

Related Work

There exist many state-of-the-art QP solvers in the literature. They differ in solution methods and are implemented with different purposes in mind. There are active-set (qpOASES Ferreau et al. [2014], quadprog Goldfarb and Idnani [1983]), interior-point (CVXOPT Andersen et al. [2013], ECOS Domahidi et al. [2013], qpSWIFT Pandala et al. [2019]), augmented-Lagrangian (OSQP Stellato et al. [2020], ProxQP), gradient (FGP Richter et al. [2012]) based methods. Some primarily used for desktop/cloud computation are MOSEK, Gurobi, and CVXOPT. Some embedded solvers are qpOASES, OSQP, ODYS [Cimini et al., 2017].

The development of our method originates in the idea of Newton Projection (NP) algorithm [Bertsekas, 1982] for box-constrained QP, developed by Dimitri Bertsekas. The NP algorithm uses a Newton-like iteration with projection with Armijo rule onto the box constraints to get the solution to the QP problem, using information about the gradient to identify the non-optimal elements of the active set.

The Combined Gradient and Newton Projection (CGNP) algorithm [Šantin and Havlena, 2011] is a variant of the NP algorithm that combines the gradient projection and Newton methods to decrease the computation burden. While NP iterations are a sequence of Newton steps computation, CGNP [Šantin and Havlena, 2011] uses computationally cheap trial gradient projection step with a fixed step-size to trigger either gradient step (trial step leads to active set change) or Newton step projection with exact line search [Nocedal and Wright, 2006] (no active set change detected). This leads to a reduction of the required computation of Newton's steps.

The CGNP algorithm was later evolved to NPP algorithm [Šantin, 2016], which replaced the gradient step with the proportionality test introduced originally in Dostál [1997]. The proportionality test is used to "look ahead" to reduce the number of unwanted changes in the active set, hence decreasing the computational complexity of the factor update scheme used to solve the auxiliary face problem.

There has been published an NPPsparse algorithm [2] based on NPP adding the utilization of fixed sparsity structure of QP problems arising in the sparse formulation of the MPC with long prediction horizon. It utilizes sparse linear algebra and the iterative solution of linear systems. Compared to the previous method, it can handle equality constraints with a specific sparsity pattern. It was shown in [2] that the algorithm outperforms the others for longer horizons as the computational complexity grows only linearly with the prediction horizon length.

Content of This Chapter

This chapter is organized as follows. In Sec. 3.1, high-level architecture of the solver is presented. In Sec. 3.2, the main algorithm is described in detail. Numerical results for the proposed solver are presented in Sec. 4.

3.1 Architecture

Compared to box-constrained QP, the presence of affine constraints brings in the need to deal with additional challenges in designing a numerically stable and efficient QP solver. The difficulties are that affine constraints can be linearly dependent and that finding a feasible initial guess is generally challenging.

We design NPPro as a two-phase active-set method with an initial preprocessing stage. Preprocessing the input data leads to a more numerically stable and efficient algorithm. Phase I of the NPPro algorithm computes (if it exists) a feasible initial guess, which serves as a starting point for iterations of Phase II. In contrast to box-constrained QP, obtaining such a feasible initial guess for (3.1) leads, in general, to the necessity of solving linear programming (LP) problems. With an initial feasible approximation, the algorithm enters Phase II, searching for the optimal solution of (3.1). The overall architecture of the NPPro algorithm is summarized in Figure 3.1.

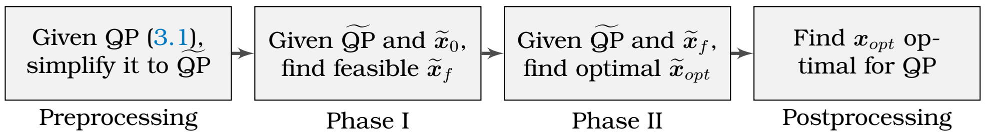

Given QP (3.1), simplify it to $\widetilde{\mathrm{QP}}$
Preprocessing
Given $\widetilde{\mathrm{QP}}$ and $\tilde{\boldsymbol{x}}_0$, find feasible $\tilde{\boldsymbol{x}}_f$
Phase I
Given $\widetilde{\mathrm{QP}}$ and $\tilde{\boldsymbol{x}}_f$, find optimal $\tilde{\boldsymbol{x}}_{opt}$
Phase II
Find $\boldsymbol{x}_{opt}$ optimal for QP
Postprocessing

Figure 3.1: Architecture of the NPPro solver

Preprocessing

The purpose of preprocessing is to simplify the problem to make the solution less computationally demanding. The preprocessing has to be as cheap as possible to bring the benefit. Numerous standard ways exist to preprocess the general input data, such as removing redundant constraints, problem scaling, and feasibility checks. See Gould and Toint [2004], Mészáros and Suhl [2003] for an overview. The NPPro employs several of those techniques as follows.

Bounds compatibility

Given a box constraint $\underline{x}_i \leq x_i \leq \overline{x}_i$, it is checked that lower and upper bounds define a non-empty set, i.e., $\underline{x}_i \leq \overline{x}_i, \quad i = 1, \dots, n$. Similarly, given an affine constraint $\underline{y}_i \leq y_i \leq \overline{y}_i$, it is checked that $\underline{y}_i \leq \overline{y}_i, \quad i = 1, \dots, m$.

Empty row in $\boldsymbol{A}$

If there is an empty i-th row in $\boldsymbol{A}$, the constraint can be read as $\underline{y}_i \leq 0 \leq \overline{y}_i$. If the constraint is fulfilled, the row will be removed as it does not affect the problem solution. Otherwise, the problem is infeasible.

Singleton row in A
If there is an i-th row in $\boldsymbol{A}$ with only one non-zero element at j-th position, the constraint $\underline{y}_i \leq a_{ij}x_j \leq \overline{y}_i$ can be converted into a box constraint

$$\underline{y}_i/a_{ij} \leq x_j \leq \overline{y}_i/a_{ij},\ a_{ij} > 0 \qquad \text{or} \qquad \overline{y}_i/a_{ij} \leq x_j \leq \underline{y}_i/a_{ij},\ a_{ij} < 0.$$

Row normalization in A
If the number of non-zero elements in i-th row of $\boldsymbol{A}$ is greater than one, the constraint $\underline{y}_i \leq \boldsymbol{A}_i\boldsymbol{x} \leq \overline{y}_i$ is normalized as $\underline{y}_i/\|\boldsymbol{A}_i\| \leq \left(\boldsymbol{A}_i/\|\boldsymbol{A}_i\|\right)\boldsymbol{x} \leq \overline{y}_i/\|\boldsymbol{A}_i\|$, where $\|.\|$ is the vector 2-norm if not specified further.

Bound tightening
If the i-th affine constraint in (3.1c) has finite bounds $\underline{y}_i$ and $\overline{y}_i$, it is possible to imply bound for x_k for which corresponding a_{ik} is non-zero. We can write

$$a_{ik}x_k + \sum_{\substack{j\neq k,\ a_{ij}>0,\\ j=1}}^{n} a_{ij}\underline{x}_j + \sum_{\substack{a_{ij}<0,\\ j=1}}^{n} a_{ij}\overline{x}_j \leq \sum_{j=1}^{n} a_{ij}x_j \leq \overline{y}_i, \qquad a_{ik} > 0 \tag{3.2a}$$

$$a_{ik}x_k + \sum_{\substack{a_{ij}>0,\\ j=1}}^{n} a_{ij}\underline{x}_j + \sum_{\substack{j\neq k,\ a_{ij}<0,\\ j=1}}^{n} a_{ij}\overline{x}_j \leq \sum_{j=1}^{n} a_{ij}x_j \leq \overline{y}_i, \qquad a_{ik} < 0 \tag{3.2b}$$

for affine upper bound and similarly

$$\underline{y}_i \leq \sum_{j=1}^{n} a_{ij}x_j \leq a_{ik}x_k + \sum_{\substack{j\neq k,\ a_{ij}>0,\\ j=1}}^{n} a_{ij}\overline{x}_j + \sum_{\substack{a_{ij}<0,\\ j=1}}^{n} a_{ij}\underline{x}_j, \qquad a_{ik} > 0 \tag{3.3a}$$

$$\underline{y}_i \leq \sum_{j=1}^{n} a_{ij}x_j \leq a_{ik}x_k + \sum_{\substack{a_{ij}>0,\\ j=1}}^{n} a_{ij}\overline{x}_j + \sum_{\substack{j\neq k,\ a_{ij}<0,\\ j=1}}^{n} a_{ij}\underline{x}_j, \qquad a_{ik} < 0 \tag{3.3b}$$

for affine lower bound. For given $\underline{x}_k$ and $\overline{x}_k$ we can rearrange the relations above to obtain

$$x_k \leq \min\left(\overline{x}_k, \frac{1}{a_{ik}}\left(\overline{y}_i - \sum_{\substack{j\neq k,\ a_{ij}>0,\\ j=1}}^{n} a_{ij}\underline{x}_j - \sum_{\substack{a_{ij}<0,\\ j=1}}^{n} a_{ij}\overline{x}_j\right)\right), \qquad a_{ik} > 0 \tag{3.4a}$$

$$x_k \leq \min\left(\overline{x}_k, \frac{1}{a_{ik}}\left(\underline{y}_i - \sum_{\substack{a_{ij}>0,\\ j=1}}^{n} a_{ij}\overline{x}_j - \sum_{\substack{j\neq k,\ a_{ij}<0,\\ j=1}}^{n} a_{ij}\underline{x}_j\right)\right), \qquad a_{ik} < 0 \tag{3.4b}$$

for upper bound tightening and similarly

$$x_k \geq \max\left(\underline{x}_k, \frac{1}{a_{ik}}\left(\underline{y}_i - \sum_{\substack{j\neq k,\ a_{ij}>0,\\ j=1}}^{n} a_{ij}\overline{x}_j - \sum_{\substack{a_{ij}<0,\\ j=1}}^{n} a_{ij}\underline{x}_j\right)\right), \qquad a_{ik} > 0 \tag{3.5a}$$

$$x_k \geq \max\left(\underline{x}_k, \frac{1}{a_{ik}}\left(\overline{y}_i - \sum_{\substack{a_{ij}>0,\\ j=1}}^{n} a_{ij}\underline{x}_j - \sum_{\substack{j\neq k,\ a_{ij}<0,\\ j=1}}^{n} a_{ij}\overline{x}_j\right)\right), \qquad a_{ik} < 0 \tag{3.5b}$$

for lower bound tightening. This technique is helpful, especially when variable x_k was originally unbounded, to reduce search space or identify infeasibility.

Phase I

The purpose of the Phase I of the NPPro algorithm is to find the initial feasible guess $\boldsymbol{x}_f \in \mathcal{R}^n$ as close as possible to the initial guess $\boldsymbol{x}_0 \in \mathcal{R}^n$. Supplying $\boldsymbol{x}_0$ enables utilization of prior information about a feasible starting point available in some applications or warm-start (starting from the previous solution expecting the current solution should be sufficiently close when solving a series of problems). In Phase I, the goal is to find projection of $\boldsymbol{x}_0$ to the constraints, formally, $\boldsymbol{x}_f$ is a solution of the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{x}_f} \quad & \|\boldsymbol{x}_f - \boldsymbol{x}_0\|, \\ \text{s. t.} \quad & \underline{\boldsymbol{x}} \leq \boldsymbol{x}_f \leq \overline{\boldsymbol{x}}, \\ & \underline{\boldsymbol{y}} \leq \boldsymbol{A}\boldsymbol{x}_f \leq \overline{\boldsymbol{y}}. \end{aligned} \tag{3.6}$$

The cost function minimizes a distance from $\boldsymbol{x}_0$. Note that the distance is treated in a general sense to seek the most computationally efficient solution. There are four cases the solver currently distinguishes based on the provided data:

No constraints

No computation is required, any $\boldsymbol{x}_0$ is feasible, therefore, $\boldsymbol{x}_f = \boldsymbol{x}_0$.

Box constraints

If the problem (3.1) does not contain any finite affine constraints, i.e., if only constraints on individual components of $\boldsymbol{x}$ are present, the situation is simple. Given (possibly infeasible) $\boldsymbol{x}_0$, the projection onto box constraints is given by

$$\boldsymbol{x}_f = \min(\max(\underline{\boldsymbol{x}},\ \boldsymbol{x}_0),\ \overline{\boldsymbol{x}}).$$

In terms of complexity, only simple element-wise computation of $\boldsymbol{x}_f$ is required.

Linearly independent constraints

Feasible guess $\boldsymbol{x}_f$ can be found via a solution of a linear system with an indefinite matrix, in the case when the total set of finite constraints (box and affine together) form a linearly independent set. Denoting as $\boldsymbol{P}_x$ a submatrix of the identity matrix, with columns corresponding to the box-constrained components, the feasible guess $\boldsymbol{x}_f$ is a solution of the following linear system

$$\begin{bmatrix} \boldsymbol{I} & \begin{bmatrix} \boldsymbol{P_x} & \boldsymbol{A}^T \end{bmatrix} \\ \begin{bmatrix} \boldsymbol{P}_{\boldsymbol{x}}^T \\ \boldsymbol{A} \end{bmatrix} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_f \\ \boldsymbol{\lambda}_f \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_0 \\ \begin{bmatrix} \boldsymbol{P}_{\boldsymbol{x}}^T \boldsymbol{x}_c \\ \boldsymbol{y}_c \end{bmatrix} \end{bmatrix}, \tag{3.7}$$

with clamped x_c and y_c defined as

$$\begin{aligned} \boldsymbol{x}_c &= \min(\max(\underline{\boldsymbol{x}},\ \boldsymbol{x}_0),\ \overline{\boldsymbol{x}}), \\ \boldsymbol{y}_c &= \min(\max(\underline{\boldsymbol{y}},\ \boldsymbol{A}\boldsymbol{x}_0),\ \overline{\boldsymbol{y}}). \end{aligned}$$

Due to the (assumed) linear independence of the constraints, the matrix $[\boldsymbol{P_x}\ \ \boldsymbol{A}^T]$ has a full column rank, and thus the whole system is regular and solvable for any initial x_0. In terms of complexity, a linear system (3.7) has to be solved to get $\boldsymbol{x}_f$.

Generic Constraints

When the set of all finite constraints is linearly dependent, it is necessary to find $\boldsymbol{x}_f$ via the solution of an augmented LP problem. Compared to the problem formulation proposed in Nocedal and Wright [2006], where a vector of slack variables (one slack for each constraint) was added, it is introduced only a scalar slack variable common for all constraints for memory saving. Having any initial x_0, we first compute

$$\boldsymbol{x}_c = \min(\max(\underline{\boldsymbol{x}},\ \boldsymbol{x}_0),\ \overline{\boldsymbol{x}})$$

and the projection onto the feasible set is consequently formulated as the following Linear Programming (LP).

$$\begin{aligned} \min_{\boldsymbol{x}_f,\kappa} \quad & \kappa \\ \text{s.t.} \quad & \underline{\boldsymbol{x}} \le \boldsymbol{x}_f \le \overline{\boldsymbol{x}} \\ & \underline{\boldsymbol{y}} \le \boldsymbol{A}\boldsymbol{x}_f + \boldsymbol{r}\kappa \le \overline{\boldsymbol{y}} \\ & 0 \le \kappa, \end{aligned} \tag{3.8}$$

where

$$\boldsymbol{r} = \min(\max(\underline{\boldsymbol{y}} - \boldsymbol{y}_c, \boldsymbol{0}), \overline{\boldsymbol{y}} - \boldsymbol{y}_c), \quad \text{where} \quad \boldsymbol{y}_c = \boldsymbol{A}\boldsymbol{x}_c.$$

The initial guess feasible for this LP problem is naturally the extended vector $[\boldsymbol{x}_c^T, 1]^T$. Feasibility of the initial guess can be verified by putting $\boldsymbol{x}_f = \boldsymbol{x}_c$ and $\kappa = 1$ in (3.8).

This is the most expansive option in terms of complexity as an LP has to be solved before the QP.

Phase II

We solve the QP problem in Phase II using a primal active set method. Similarly, as with other active set methods, to do so, we solve a sequence of equality-constrained QP problems with changing set of active constraints. As the computation of a new Newton step is computationally the most costly part of the algorithm (Sec. 3.2), the intention is to decrease the number of iterations by allowing a rapid change in the active working set imposed by a projection operation together with look ahead feature of proportionality test to prevent from the unnecessary expansion of the active set to reducing the computation burden associated with factor updates in Newton step computation.

3.2 Algorithm

This section describes the algorithm used to solve (3.1) in Phase II given a feasible initial guess from Phase I. With subtle modifications, the same algorithm is also used to solve the augmented LP (3.8) in Phase I of the NPPro solver.

In the following, basic ingredients are described in detail, and the algorithm is built at this section's end.

Optimality Conditions

For brevity, we reformualte problem (3.1) into box constrained with equality contraints as follows

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{f}, \\ \text{s. t.} \quad & \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} = \boldsymbol{0}, \\ & \underline{\boldsymbol{x}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{x}}, \\ & \underline{\boldsymbol{y}} \leq \boldsymbol{y} \leq \overline{\boldsymbol{y}}, \end{aligned} \tag{3.9}$$

where $\boldsymbol{y} \in \mathbb{R}^m$ is an auxiliary optimization variable.

We will follow [Nocedal and Wright, 2006, Chapter 12] and derive the optimality (KKT) conditions for problem (3.9). We define the Lagrangian function L associated with (3.9) as

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}, \boldsymbol{\mu}_{\underline{\boldsymbol{x}}}, \boldsymbol{\mu}_{\overline{\boldsymbol{x}}}, \boldsymbol{\mu}_{\underline{\boldsymbol{y}}}, \boldsymbol{\mu}_{\overline{\boldsymbol{y}}}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x} + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) \\ + \boldsymbol{\mu}_{\overline{\boldsymbol{x}}}^T(\boldsymbol{x} - \overline{\boldsymbol{x}}) - \boldsymbol{\mu}_{\underline{\boldsymbol{x}}}^T(\boldsymbol{x} - \underline{\boldsymbol{x}}) + \boldsymbol{\mu}_{\overline{\boldsymbol{y}}}^T(\boldsymbol{y} - \overline{\boldsymbol{y}}) - \boldsymbol{\mu}_{\underline{\boldsymbol{y}}}^T(\boldsymbol{y} - \underline{\boldsymbol{y}}), \end{aligned} \tag{3.10}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is Lagrange multiplier associated with the equality constraint, $\boldsymbol{\mu}_{\underline{\boldsymbol{x}}}, \boldsymbol{\mu}_{\overline{\boldsymbol{x}}} \in \mathbb{R}^n$, and $\boldsymbol{\mu}_{\underline{\boldsymbol{y}}}, \boldsymbol{\mu}_{\overline{\boldsymbol{y}}} \in \mathbb{R}^m$ are the Lagrange multipliers associated with lower and upper bounds imposed on variables $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively.

The optimality conditions give that for any solution $\boldsymbol{x}$ of (3.9), there exists an auxiliary variable $\boldsymbol{y}$ and Lagrange multipliers $\boldsymbol{\lambda}$, $\boldsymbol{\mu}_{\underline{\boldsymbol{x}}}$, $\boldsymbol{\mu}_{\overline{\boldsymbol{x}}}$, $\boldsymbol{\mu}_{\underline{\boldsymbol{y}}}$, $\boldsymbol{\mu}_{\overline{\boldsymbol{y}}}$ satisfying KKT conditions

$$\begin{aligned} \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f} + \boldsymbol{A}^T\boldsymbol{\lambda} + \boldsymbol{\mu}_{\overline{\boldsymbol{x}}} - \boldsymbol{\mu}_{\underline{\boldsymbol{x}}} &= 0, \\ -\boldsymbol{\lambda} + \boldsymbol{\mu}_{\overline{\boldsymbol{y}}} - \boldsymbol{\mu}_{\underline{\boldsymbol{y}}} &= 0, \\ \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} &= 0, \\ \boldsymbol{x} - \underline{\boldsymbol{x}}, \overline{\boldsymbol{x}} - \boldsymbol{x}, \boldsymbol{y} - \underline{\boldsymbol{y}}, \overline{\boldsymbol{y}} - \boldsymbol{y} &\geq 0, \\ \boldsymbol{\mu}_{\underline{\boldsymbol{x}}}, \boldsymbol{\mu}_{\overline{\boldsymbol{x}}}, \boldsymbol{\mu}_{\underline{\boldsymbol{y}}}, \boldsymbol{\mu}_{\overline{\boldsymbol{y}}} &\geq 0, \\ {\boldsymbol{\mu}_{\underline{\boldsymbol{x}}}}^T(\boldsymbol{x} - \underline{\boldsymbol{x}}) = {\boldsymbol{\mu}_{\overline{\boldsymbol{x}}}}^T(\overline{\boldsymbol{x}} - \boldsymbol{x}) = {\boldsymbol{\mu}_{\underline{\boldsymbol{y}}}}^T(\boldsymbol{y} - \underline{\boldsymbol{y}}) = {\boldsymbol{\mu}_{\overline{\boldsymbol{y}}}}^T(\overline{\boldsymbol{y}} - \boldsymbol{y}) &= 0. \end{aligned} \tag{3.11}$$

Lagrange multipliers $\boldsymbol{\mu}_{\underline{\boldsymbol{x}}}$ and $\boldsymbol{\mu}_{\overline{\boldsymbol{x}}}$ can be stored only as one vector $\boldsymbol{\mu}_{\boldsymbol{x}}$. The reason is that only the lower or upper bound can be active. It follows that $\boldsymbol{\mu}_{\underline{\boldsymbol{x}}}^T\boldsymbol{\mu}_{\overline{\boldsymbol{x}}} = 0$. Analogous remark holds for multipliers $\boldsymbol{\mu}_{\underline{\boldsymbol{y}}}$ and $\boldsymbol{\mu}_{\overline{\boldsymbol{y}}}$.

Newton Direction Computation

We search in each iteration k for an update

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{x}^{(k)} + \boldsymbol{d}_{\boldsymbol{x}}, \\ \boldsymbol{y} &= \boldsymbol{y}^{(k)} + \boldsymbol{d}_{\boldsymbol{y}}, \end{aligned} \tag{3.12}$$

towards the optimal solution of the face problem [1]. Substituting (3.12) into (3.13), we can find the update as a solution of the following system

$$\begin{bmatrix} \boldsymbol{H} & \boldsymbol{0} & \boldsymbol{A}^T & \boldsymbol{P_x} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{P_y} \\ \boldsymbol{A} & -\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{P_x^T} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{P_y^T} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{d_x} \\ \boldsymbol{d_y} \\ \boldsymbol{\lambda} \\ \boldsymbol{\mu_x} \\ \boldsymbol{\mu_y} \end{bmatrix} = \begin{bmatrix} -\left(\boldsymbol{f} + \boldsymbol{H}\boldsymbol{x}^{(k)}\right) \\ \boldsymbol{0} \\ \boldsymbol{y}^{(k)} - \boldsymbol{A}\boldsymbol{x}^{(k)} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}. \tag{3.13}$$

Given working set $\mathcal{W}^{(k)} = (\mathcal{W}_x^{(k)}, \mathcal{W}_y^{(k)})$, the columns of matrices $\boldsymbol{P_x} = \{\boldsymbol{e}_i, \quad i \in \mathcal{W}_x^{(k)}\}$ and $\boldsymbol{P_y} = \{\boldsymbol{e}_j, \quad j \in \mathcal{W}_y^{(k)}\}$ are columns of identity matrix corresponding to the elements in the working set.

We decompose $\boldsymbol{d_x} = \boldsymbol{P_x}\tilde{\boldsymbol{d}}_{\boldsymbol{x}} + \boldsymbol{Z_x}\bar{\boldsymbol{d}}_{\boldsymbol{x}}$ onto $\tilde{\boldsymbol{d}}_{\boldsymbol{x}}$, the direction coresponding to the active box constraints, and $\bar{\boldsymbol{d}}_{\boldsymbol{x}}$, the direction corresponding to the non-active box constraints. Moreover, $\boldsymbol{Z_x}$ is null space of associated box constraints such that $\boldsymbol{P_x^T}\boldsymbol{Z_x} = \boldsymbol{0}$, specifically $\boldsymbol{Z_x} = \{\boldsymbol{e}_i, \quad i \notin \mathcal{W}_x^{(k)}\}$[2]. Similarly, one can decompose $\boldsymbol{d_y} = \boldsymbol{P_y}\tilde{\boldsymbol{d}}_{\boldsymbol{y}} + \boldsymbol{Z_y}\bar{\boldsymbol{d}}_{\boldsymbol{y}}$ based on activation of affine constraints. Further, $\boldsymbol{Z_y}$ is designed such that $\boldsymbol{P_y^T}\boldsymbol{Z_y} = \boldsymbol{0}$, specifically $\boldsymbol{Z_y} = \{\boldsymbol{e}_j, \quad j \notin \mathcal{W}_y^{(k)}\}$.

Since the equality constraint is artificial, it has a purpose only when affine constraints are active. It is suitable to decompose $\boldsymbol{\lambda} = \boldsymbol{P_y}\tilde{\boldsymbol{\lambda}} + \boldsymbol{Z_y}\bar{\boldsymbol{\lambda}}$ according to activation of the affine constraints. Therefore, given working set $\mathcal{W}^{(k)}$, the particular solutions $\tilde{\boldsymbol{d}}_{\boldsymbol{x}}, \tilde{\boldsymbol{d}}_{\boldsymbol{y}}, \tilde{\boldsymbol{\lambda}} = 0$.

Note that we assume the equality constraint hold, therefore, $\boldsymbol{y}^{(k)} - \boldsymbol{A}\boldsymbol{x}^{(k)} = \boldsymbol{0}$. Then problem (3.13) can be reduced to

$$\begin{bmatrix} \boldsymbol{Z_x^T}\boldsymbol{H}\boldsymbol{Z_x} & \boldsymbol{0} & \boldsymbol{Z_x^T}\boldsymbol{A}^T\boldsymbol{P_y} \\ \boldsymbol{0} & \boldsymbol{0} & -\boldsymbol{Z_y^T}\boldsymbol{P_y} \\ \boldsymbol{P_y^T}\boldsymbol{A}\boldsymbol{Z_x} & -\boldsymbol{P_y^T}\boldsymbol{Z_y} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{d}}_{\boldsymbol{x}} \\ \bar{\boldsymbol{d}}_{\boldsymbol{y}} \\ \bar{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{Z_x^T}\boldsymbol{g} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \tag{3.14}$$

with gradient $\boldsymbol{g} = \boldsymbol{H}\boldsymbol{x}^{(k)} + \boldsymbol{f}$ and where complement to the last equation

[1]Face problem is the problem on a subspace specified by the current working set $\mathcal{W}^{(k)}$.

[2]The shorhand notation $i \notin \mathcal{W}_x^{(k)}$ and $j \notin \mathcal{W}_y^{(k)}$ stands for $i \in \mathcal{I}_x \backslash \mathcal{W}_x^{(k)}$ and $j \in \mathcal{I}_y \backslash \mathcal{W}_y^{(k)}$, where $\mathcal{I}_x = \{1, \dots, n\}$ and $\mathcal{I}_y = \{1, \dots, m\}$, respectively.

is

$$\boldsymbol{Z}_{\boldsymbol{y}}^T \boldsymbol{A} \boldsymbol{Z}_{\boldsymbol{x}} \bar{\boldsymbol{d}}_{\boldsymbol{x}} = \boldsymbol{Z}_{\boldsymbol{y}}^T \boldsymbol{Z}_{\boldsymbol{y}} \bar{\boldsymbol{d}}_{\boldsymbol{y}}.$$

Since $\boldsymbol{P}_{\boldsymbol{y}}^T \boldsymbol{Z}_{\boldsymbol{y}} = \boldsymbol{0}$, we can further reduce (3.14) to

$$\begin{bmatrix} \bar{\boldsymbol{H}} & \bar{\boldsymbol{A}}_0^T \\ \bar{\boldsymbol{A}}_0 & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{d}}_{\boldsymbol{x}} \\ \tilde{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{d}} \\ \boldsymbol{0} \end{bmatrix}, \tag{3.15}$$

with

$$\bar{\boldsymbol{d}}_{\boldsymbol{y}} = \bar{\boldsymbol{A}}_{\boldsymbol{I}} \bar{\boldsymbol{d}}_{\boldsymbol{x}},$$

where

$$\begin{aligned} \bar{\boldsymbol{H}} &= \boldsymbol{Z}_{\boldsymbol{x}}^T \boldsymbol{H} \boldsymbol{Z}_{\boldsymbol{x}}, \\ \bar{\boldsymbol{A}}_0 &= \boldsymbol{P}_{\boldsymbol{y}}^T \boldsymbol{A} \boldsymbol{Z}_{\boldsymbol{x}}, \\ \bar{\boldsymbol{A}}_{\boldsymbol{I}} &= \boldsymbol{Z}_{\boldsymbol{y}}^T \boldsymbol{A} \boldsymbol{Z}_{\boldsymbol{x}}, \\ \bar{\boldsymbol{d}} &= -\boldsymbol{Z}_{\boldsymbol{x}}^T \boldsymbol{g}, \\ \bar{\boldsymbol{d}}_{\boldsymbol{x}} &= \boldsymbol{Z}_{\boldsymbol{x}}^T \boldsymbol{d}_{\boldsymbol{x}}, \\ \tilde{\boldsymbol{\lambda}} &= \boldsymbol{P}_{\boldsymbol{y}} \boldsymbol{\lambda}, \end{aligned} \tag{3.16}$$

with a general direction $\bar{\boldsymbol{d}}$ on the right hand side. It follows from the definition of $\boldsymbol{Z}_{\boldsymbol{x}}$ that the Newton direction $\boldsymbol{d}_{\boldsymbol{x}}$ can be reconstructed from $\bar{\boldsymbol{d}}_{\boldsymbol{x}}$ by filling zeros at the entries of constraints, which are not in the working set $\mathcal{W}_x^{(k)}$. Note that all the data, e.g., $\bar{\boldsymbol{H}}$, $\bar{\boldsymbol{A}}_0$, $\bar{\boldsymbol{A}}_{\boldsymbol{I}}$, $\bar{\boldsymbol{d}}$, $\bar{\boldsymbol{d}}_{\boldsymbol{x}}$ since $\boldsymbol{Z}_{\boldsymbol{x}}$, $\boldsymbol{Z}_{\boldsymbol{y}}$ are changing each iteration k, yet the superscript k was ommited for sake of brevity here.

Note there is a corner case when rank($\bar{\boldsymbol{H}}$) = rank($\bar{\boldsymbol{A}}_0$), the solution of (3.15) is inevitably $\boldsymbol{d}_{\boldsymbol{x}} = 0$, since there is no degree of freedom left. Also, no direction towards minimizer may exist as we are already in the minimizer of the face problem. These two cases can be detected by orthogonality condition

$$0 = \boldsymbol{d}_{\boldsymbol{x}} \cdot \boldsymbol{g} \tag{3.17}$$

Solution Based on Schur Complement

Computation of saddle point problem (3.15) is the most computationaly demanding part of the algorithm. This problem, for non-singular $\bar{\boldsymbol{H}}$, can be handled using Schur complement method [Wang and Boyd, 2010] as

$$\begin{bmatrix} \bar{H} & \bar{A}_0^T \\ \bar{A}_0 & 0 \end{bmatrix} \begin{bmatrix} \bar{d}_x \\ \tilde{\lambda} \end{bmatrix} = \begin{bmatrix} \bar{d} \\ 0 \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} \bar{H} & \bar{A}_0^T \\ 0 & \bar{S} \end{bmatrix} \begin{bmatrix} \bar{d}_x \\ \tilde{\lambda} \end{bmatrix} = \begin{bmatrix} \bar{d} \\ \bar{A}_0 \bar{H}^{-1} \bar{d} \end{bmatrix},$$

where Schur complement is given by

$$\bar{S} = \bar{A}_0 \bar{H}^{-1} \bar{A}_0^T.$$

Then problem (3.15) can be solved by Alg. 2.

Algorithm 2 Solution of (3.15)

```
1: $\bar{H} = \bar{R}^T \bar{D} \bar{R}$
2: $\bar{X}_0 = (\bar{R}^T \bar{D})^{-1} \bar{A}_0^T$
3: $\bar{S} = \bar{X}_0^T \bar{D} \bar{X}_0 = \bar{R}_0^T \bar{D}_0 \bar{R}_0$
4: $\tilde{\lambda} = (\bar{R}_0^T \bar{D}_0 \bar{R}_0)^{-1} \bar{A}_0 \bar{H}^{-1} \bar{d} = \bar{R}_0^{-1} \bar{Q}_0^T \bar{H}^{-1} \bar{d}$
5: $\bar{d}_x = (\bar{R}^T \bar{D} \bar{R})^{-1} (\bar{d} - \bar{A}_0^T \tilde{\lambda})$
```

In step 1, $\bar{H}$ is decomposed with diagonal matrix $\bar{D}$ and unit upper-triangular matrix (upper-triangular matrix with unit diagonal) $\bar{R}$. In step 2, $\bar{X}_0$ is obtained after solution of triangle system with $\bar{R}^T$ followed by scaling by $\bar{D}$. In step 3, the decomposition of the Schur complement can be obtained by triangularization $\left|\bar{D}; \bar{X}_0\right| \to \left|\bar{D}_0; \bar{R}_0\right|$. In step 4, the right-hand side is computed (using $\bar{H}^{-1} = \bar{R}^{-T} \bar{D}^{-1} \bar{R}^{-1}$), then forward substitution, scaling, and backward substitutions series is applied to obtain $\tilde{\lambda}$. The definition of $\bar{Q}_0^T$ comes from the decomposition of $\bar{A}_0 = \bar{R}_0^T \bar{D}_0 \bar{Q}_0^T$. Similarly, in step 5, the right-hand side is computed, then the series is applied to obtain $\bar{d}_x$.

Solution Based on Preconditioned MINRES

For completeness, it should be mentioned an alternative approach proposed in [2]. The Newton step can be directly computed from (3.15), or by using the Schur complement method (Sec. 3.2), or the Riccati recursion [Axehill and Hansson, 2008].

Recall linear system

$$\begin{bmatrix} \bar{\boldsymbol{H}} & \bar{\boldsymbol{A}}_0^T \\ \bar{\boldsymbol{A}}_0 & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{d}}_x \\ \tilde{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{d}} \\ \boldsymbol{0} \end{bmatrix} \quad \Longleftrightarrow \quad \mathbb{G}\mathbb{z} = \mathbb{h}$$

from (3.15) that is indefinite, which prohibits using a method such as conjugate gradients. Here, an iterative method is used to exploit the maximal sparsity of the Karush–Kuhn–Tucker (KKT) matrix $\mathbb{G}$. The linear system is solved using the preconditioned MINimum RESidual (MINRES) method [Paige and Saunders, 1975] instead. For the sake of simplicity, the iteration index of the NPPro algorithm is omitted in the description of the MINRES method. The index l denotes an inner iteration of the MINRES method here.

It is an iterative Krylov subspace method suitable for solving symmetric indefinite problems. The MINRES minimizes residual norm $\left\|\mathbb{r}^l\right\| = \left\|\mathbb{h} - \mathbb{G}\mathbb{z}^l\right\|$ of the Krylov subspace at each iteration l. In this paper, a finite termination property is assumed, i.e., the exact solution to (3.15) is found in the finite steps of the MINRES method. This assumption is based on the fact that the number of iterations of the Krylov method is bounded by the maximal subspace dimension of the well-defined matrix $\mathbb{G}$ [Liesen and Tichý, 2014].

The MINRES convergence can be speeded-up significantly when a (sparse) suitable positive definite preconditioner is used. A survey on preconditioners for saddle-point problems can be found in Benzi and Wathen [2008]. An augmented Lagrangian-based preconditioner used in this paper has already been successfully used in control [Quirynen et al., 2018] but also in other applications [Greif and Schötzau, 2007; Benzi and Wang, 2011]. The solution is found in very few iterations

$$\mathbb{L}^{-1}\mathbb{G}\mathbb{L}^{-T}\left(\mathbb{L}^T\mathbb{z}^l\right) = -\mathbb{L}^{-1}\mathbb{h}, \tag{3.18}$$

where $\mathbb{L}$ is a sparse Cholesky decomposition of the preconditioner

$$\mathbb{P} = \begin{bmatrix} \bar{\boldsymbol{H}} + \gamma\bar{\boldsymbol{A}}_0^T\bar{\boldsymbol{A}}_0 & \\ & \frac{1}{\gamma}\boldsymbol{I} \end{bmatrix} = \mathbb{L}\mathbb{L}^T. \tag{3.19}$$

The preconditioner $\mathbb{P} \succ$ is decomposed by the sparse Cholesky factorization every single iteration once before MINRES is called. The effect of the preconditioner highly depends on the preconditioning parameter $\gamma > 0$. The higher the value, the faster the MINRES converges, as the eigenvalues of $\mathbb{L}^{-1}\mathbb{H}\mathbb{L}^{-T}$ in (3.18) are grouped tighter. Fewer iterations are needed to find the solution, as convergence of the MINRES method depends on several distinct eigenvalues [Benzi and Wathen, 2008]. However, too high values of γ can cause numerical instability, as the matrix $\mathbb{L}$ becomes ill-conditioned. The decomposition cost can also be reduced by applying the factor-update scheme proposed in Kirches et al. [2011].

Hessian Factor Updates

Assume the Schur complement method (Sec. 3.2) is used further. The factor $\bar{\boldsymbol{R}}$ of the UDU decomposition of the reduced Hessian $\bar{\boldsymbol{H}}$ obtained by selecting rows and columns corresponding to the actual working set is needed for every iteration. Hessian decomposition computed from scratch is computationally expensive. Therefore, Hessian factor updates provide a significant speed-up. Constraints are classified into three classes, depending on how the working set has been changed – those that remain (non-)active, those that change from non-active to active, and those that change from active to non-active.

Reduction In this case, the non-active box constraint is becoming active, therefore, the column corresponding to the constraint has to be removed from the factor. The removal of the constraint cause that Hessian factor is no longer triangular matrix and thus has to be triangularized by dyadic reduction [Peterka, 1986]. For example, having

$$\left|\bar{\boldsymbol{d}}_0; \bar{\boldsymbol{R}}_0\right| = \left| \begin{bmatrix}\bar{\boldsymbol{d}}_{01}\\ \bar{\boldsymbol{d}}_{02}\\ \bar{\boldsymbol{d}}_{03}\end{bmatrix}; \begin{bmatrix}\bar{\boldsymbol{R}}_{01} & \bar{\boldsymbol{R}}_{02} & \bar{\boldsymbol{R}}_{03}\end{bmatrix}\right|$$

one wants to remove $\bar{\boldsymbol{d}}_{02}$ and column $\bar{\boldsymbol{R}}_{02}$ from the original factor. In the following sketch of UD update – reduction – non-zero elements (denoted by $+$) that are zeroed by dyadic reductions are depicted by $\ominus$, affected by $\oplus$.

$$\left|\begin{bmatrix}\bar{\boldsymbol{d}}_{01}\\ \bar{\boldsymbol{d}}_{03}\end{bmatrix};\begin{bmatrix}\bar{\boldsymbol{R}}_{01} & \bar{\boldsymbol{R}}_{03}\end{bmatrix}\right| \ni \left[\begin{array}{ccc|cc} 1 & + & + & + & + \\ 0 & 1 & + & + & + \\ 0 & 0 & 1 & + & + \\ 0 & 0 & 0 & + & + \\ 0 & 0 & 0 & 1 & + \\ 0 & 0 & 0 & 0 & 1 \end{array}\right] \to \begin{bmatrix} 1 & + & + & + & + \\ 0 & 1 & + & + & + \\ 0 & 0 & 1 & + & + \\ \cancel{0} & \cancel{0} & \cancel{0} & \cancel{\ominus} & \cancel{\ominus} \\ 0 & 0 & 0 & 1 & \oplus \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \in \left|\bar{\boldsymbol{d}}_1;\bar{\boldsymbol{R}}_1\right|,$$

The resulting factor $\left|\bar{\boldsymbol{d}}_1;\bar{\boldsymbol{R}}_1\right|$ is then obtained by removing the zeroed row.

Augmentation In this case, the active constraint is becoming non-active. Therefore, the Hessian column corresponding to the constraints has to be added to the factor. Adding a column as the last one of the factor remains the triangular shape of the factor. Nevertheless, the column has to be processed by the UDU factorization method.

$$\begin{bmatrix}\bar{\boldsymbol{H}} & \boldsymbol{s}\\ \boldsymbol{s}^T & t\end{bmatrix} = \begin{bmatrix}\bar{\boldsymbol{R}}^T & \boldsymbol{0}\\ \boldsymbol{u}^T & v\end{bmatrix}\begin{bmatrix}\bar{\boldsymbol{D}} & \boldsymbol{0}\\ \boldsymbol{0} & k\end{bmatrix}\begin{bmatrix}\bar{\boldsymbol{R}} & \boldsymbol{u}\\ \boldsymbol{0} & v\end{bmatrix} \tag{3.20}$$

UDU decomposition of augmented matrix is given in (3.20). The decomposition is obtained in the following three steps:

$$\bar{\boldsymbol{H}} = \bar{\boldsymbol{R}}^T\bar{\boldsymbol{D}}\bar{\boldsymbol{R}} \quad \text{(already available)} \tag{3.21a}$$

$$\boldsymbol{s} = \bar{\boldsymbol{R}}^T\bar{\boldsymbol{D}}\boldsymbol{u} \quad \text{(solve for } \boldsymbol{u}\text{)} \tag{3.21b}$$

$$t - \boldsymbol{u}^T\bar{\boldsymbol{D}}\boldsymbol{u} = vkv \quad \text{(update left hand side, then decompose for } v \text{ and } k\text{)} \tag{3.21c}$$

Since the decomposition in (3.21a) is already known, only (3.21b) and (3.21c) are proceed in this UD update – augmentation. Multiple columns can be augmented at once by assuming block variant of (3.20).

Projected Line Search

The projected line search expands the working set of active constraints. The proposed method enhances Cauchy point computation originally proposed for box constraints in [Nocedal and Wright, 2006, Chapter 16.7] for QP with affine constraints. Further, we differentiate (inner) iterations t of the projected line search algorithm performed within (outer) iteration k of the main algorithm.

Breakpoints Computation

When the Newton direction $\boldsymbol{d}_{\boldsymbol{x}}$ is applied, two scenarios may happen: (i) full Newton direction can be applied without violation of any constraints, i.e., $\boldsymbol{x}^{(k)} + \boldsymbol{d}_{\boldsymbol{x}}$ is a feasible point with respect to all (affine and box) constraints, (ii) some new constraints are activated, i.e., $\boldsymbol{x}^{(k)} + \boldsymbol{d}_{\boldsymbol{x}}$ is not feasible and the Newton direction can be applied only with a limited step $\alpha^{(t)} \in [0, 1)$ such that $\boldsymbol{x}^{(t)} + \alpha^{(t)} \boldsymbol{d}_{\boldsymbol{x}}^{(t)}$ remains feasible, where $\boldsymbol{x}^{(0)} = \boldsymbol{x}^{(k)}$ and $\boldsymbol{d}_{\boldsymbol{x}}^{(0)} = \boldsymbol{d}_{\boldsymbol{x}}^{(k)}$. This step $\alpha^{(t)}$ is a minimum of breakpoints [3] $\tilde{\alpha}^{(t)}$ computed at every iteration t for each currently inactive constraint such that this constraint become active. Having $\boldsymbol{y}^{(t)} = \boldsymbol{A}\boldsymbol{x}^{(t)}$ and $\boldsymbol{d}_{\boldsymbol{y}} = \boldsymbol{A}\boldsymbol{d}_{\boldsymbol{x}}$, we need to ensure, that

$$\begin{aligned} \underline{\boldsymbol{x}}_i &\leq (\boldsymbol{x}^{(t)} + \alpha^{(t)} \boldsymbol{d}_{\boldsymbol{x}})_i \leq \overline{\boldsymbol{x}}_i, \quad \forall i \notin \mathcal{W}_x^{(t)}, \\ \underline{\boldsymbol{y}}_j &\leq (\boldsymbol{y}^{(t)} + \alpha^{(t)} \boldsymbol{d}_{\boldsymbol{y}})_j \leq \overline{\boldsymbol{y}}_j, \quad \forall j \notin \mathcal{W}_y^{(t)}. \end{aligned}$$

[3]Breakpoint is where one or more constraints that were previously inactive become active.

Thus, in order to decide which situation occurs, the maximal feasible step size is computed by

$$
\begin{aligned}
\alpha^{(t)} &= \min\{1, \min_{i \notin \mathcal{W}_x^{(t)}} (\alpha_{\boldsymbol{x}})_i, \min_{j \notin \mathcal{W}_y^{(t)}} (\alpha_{\boldsymbol{y}})_j\}, \text{ with} \\
(\alpha_{\boldsymbol{x}})_i &= \frac{\overline{\boldsymbol{x}}_i - \boldsymbol{x}^{(t)}}{(\boldsymbol{d}_{\boldsymbol{x}})_i}, \quad (\boldsymbol{d}_{\boldsymbol{x}})_i > 0 \\
&= \frac{\boldsymbol{x}^{(t)} - \underline{\boldsymbol{x}}_i}{(\boldsymbol{d}_{\boldsymbol{x}})_i}, \quad (\boldsymbol{d}_{\boldsymbol{x}})_i < 0 \\
&= \infty, \quad \text{otherwise} \\
(\alpha_{\boldsymbol{y}})_j &= \frac{\overline{\boldsymbol{y}}_j - \boldsymbol{y}^{(t)}}{(\boldsymbol{d}_{\boldsymbol{y}})_j}, \quad (\boldsymbol{d}_{\boldsymbol{y}})_j > 0 \\
&= \frac{\boldsymbol{y}^{(t)} - \underline{\boldsymbol{y}}_j}{(\boldsymbol{d}_{\boldsymbol{y}})_j}, \quad (\boldsymbol{d}_{\boldsymbol{y}})_j < 0 \\
&= \infty, \quad \text{otherwise.}
\end{aligned}
\tag{3.22}
$$

Moreover, the corresponding index is noted and prepared to be added to the updated working set.

Exact Line Search

Given direction $\boldsymbol{d}_{\boldsymbol{x}}^{(t)}$ on interval $[\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t+1)}]$, the exact line search method finds $\beta^{(t)}$ such that

$$
\min_{\gamma} q(\boldsymbol{x}^{(t)}) \tag{3.23}
$$

where

$$
q(\boldsymbol{x}^{(t)}) = \frac{1}{2}\left(\boldsymbol{x}^{(t)} + \gamma^{(t)}\boldsymbol{d}_{\boldsymbol{x}}^{(t)}\right)^T \boldsymbol{H}\left(\boldsymbol{x}^{(t)} + \gamma^{(t)}\boldsymbol{d}_{\boldsymbol{x}}^{(t)}\right) + \left(\boldsymbol{x}^{(t)} + \gamma^{(t)}\boldsymbol{d}_{\boldsymbol{x}}^{(t)}\right)^T \boldsymbol{f}. \tag{3.24}
$$

By differentiation of q with respect to β the condition for minimizer is

$$
q' = \gamma^{(t)} \boldsymbol{d}_{\boldsymbol{x}}^{(t)^T} \boldsymbol{H} \boldsymbol{d}_{\boldsymbol{x}}^{(t)} + \boldsymbol{d}_{\boldsymbol{x}}^{(t)^T} \boldsymbol{H} \boldsymbol{x}^{(t)} + \boldsymbol{d}_{\boldsymbol{x}}^{(t)^T} \boldsymbol{f} = 0, \tag{3.25}
$$

resulting in the exact step size

$$\gamma^{(t)} = \frac{\boldsymbol{d}_{\boldsymbol{x}}^{(t)^T} \boldsymbol{H} \boldsymbol{x}^{(t)} + \boldsymbol{d}_{\boldsymbol{x}}^{(t)^T} \boldsymbol{f}}{\boldsymbol{d}_{\boldsymbol{x}}^{(t)^T} \boldsymbol{H} \boldsymbol{d}_{\boldsymbol{x}}^{(t)}}. \tag{3.26}$$

and step size restricted to the interval

$$\beta^{(t)} = \min(\max(0, \gamma^{(t)}), \alpha^{(t)}) \tag{3.27}$$

returns

0	if $\boldsymbol{x}^{(t)}$ is the local minimizer
$\gamma^{(t)}$	if the minimizer is within the interval $[\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t+1)}]$
$\alpha^{(t)}$	if the minimizer is situated on a following interval

such that $\boldsymbol{x}^{(t)} + \beta^{(t)} \boldsymbol{d}_{\boldsymbol{x}}^{(t)}$ minimizes the cost function.

Projected Direction Computation

Having a computed direction step $\boldsymbol{d}_{\boldsymbol{x}}^{(0)} = \boldsymbol{d}_{\boldsymbol{x}}^{(k)}$, we would like to project it, to reduce computational cost and to minimize the number of outer iterations. However, due to the presence of affine constraints, the situation is more complex. While it is straightforward to project the direction $\boldsymbol{d}_{\boldsymbol{x}}^{(t)}$ such that the projected direction $\boldsymbol{d}_{\boldsymbol{x}}^{(t+1)}$ satisfy $(\boldsymbol{d}_{\boldsymbol{x}}^{(t+1)})_i = 0$ for all active constraints $i \in \mathcal{W}_x^{(t+1)}$, we also need to guarantee that the projected direction step $\boldsymbol{d}_{\boldsymbol{x}}^{(t+1)}$ satisfy $(\boldsymbol{d}_{\boldsymbol{y}}^{(t+1)})_j = 0$ for all active constraints $j \in \mathcal{W}_y^{(t+1)}$. This leads to the least-squares problem

$$\begin{aligned} \min_{\boldsymbol{d}_{\boldsymbol{x}}^{(t+1)}} \quad & \|\boldsymbol{d}_{\boldsymbol{x}}^{(t+1)} - \boldsymbol{d}_{\boldsymbol{x}}^{(t)}\|^2 \\ \text{s. t.} \quad & \boldsymbol{P}_{\boldsymbol{x}}^T \boldsymbol{d}_{\boldsymbol{x}}^{(t+1)} = \boldsymbol{0}, \\ & \boldsymbol{P}_{\boldsymbol{y}}^T \boldsymbol{A} \boldsymbol{d}_{\boldsymbol{x}}^{(t+1)} = \boldsymbol{0}, \end{aligned} \tag{3.28}$$

with the associated linear system

$$\begin{bmatrix} \boldsymbol{I} & \bar{\boldsymbol{A}}_0^T \\ \bar{\boldsymbol{A}}_0 & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t+1)} \\ \tilde{\boldsymbol{\lambda}}^{(t+1)} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t)} \\ \boldsymbol{0} \end{bmatrix}, \tag{3.29}$$

where

$$\bar{\boldsymbol{A}}_0 = \boldsymbol{P}_{\boldsymbol{y}}^T \boldsymbol{A} \boldsymbol{Z}_{\boldsymbol{x}}, \quad \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t)} = \boldsymbol{Z}_{\boldsymbol{x}}^T \boldsymbol{d}_{\boldsymbol{x}}^{(t)}, \quad \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t+1)} = \boldsymbol{Z}_{\boldsymbol{x}}^T \boldsymbol{d}_{\boldsymbol{x}}^{(t+1)}, \quad \tilde{\boldsymbol{\lambda}}^{(t+1)} = \boldsymbol{P}_{\boldsymbol{y}}^T \boldsymbol{\lambda}^{(t+1)}.$$

Computation of problem (3.29) is closely related to the problem (3.15) as a special case $\bar{\boldsymbol{H}} = \boldsymbol{I}$, where $\boldsymbol{\lambda}^{(t+1)}$ plays the role of least-square residual. This problem is handled directly by Schur complement method as

$$\begin{bmatrix} \boldsymbol{I} & \bar{\boldsymbol{A}}_0^T \\ \bar{\boldsymbol{A}}_0 & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t+1)} \\ \tilde{\boldsymbol{\lambda}}^{(t+1)} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t)} \\ \boldsymbol{0} \end{bmatrix} \quad \Longrightarrow \quad \begin{bmatrix} \boldsymbol{I} & \bar{\boldsymbol{A}}_0^T \\ \boldsymbol{0} & \bar{\boldsymbol{S}} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t+1)} \\ \tilde{\boldsymbol{\lambda}}^{(t+1)} \end{bmatrix} = \begin{bmatrix} \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t)} \\ \bar{\boldsymbol{A}}_0 \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t)} \end{bmatrix}.$$

The computation requires three steps summarized in Alg. 3.

Algorithm 3 Solution of (3.29)

```
1: $\bar{\boldsymbol{S}} = \bar{\boldsymbol{A}}_0 \bar{\boldsymbol{A}}_0^T = \bar{\boldsymbol{R}}_0^T \bar{\boldsymbol{D}}_0 \bar{\boldsymbol{R}}_0$
2: $\tilde{\boldsymbol{\lambda}}^{(t+1)} = (\bar{\boldsymbol{R}}_0^T \bar{\boldsymbol{D}}_0 \bar{\boldsymbol{R}}_0)^{-1} \bar{\boldsymbol{A}}_0 \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t)} = \bar{\boldsymbol{R}}_0^{-1} \bar{\boldsymbol{Q}}_0^T \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t)}$
3: $\bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t+1)} = \bar{\boldsymbol{d}}_{\boldsymbol{x}}^{(t)} - \bar{\boldsymbol{A}}_0^T \tilde{\boldsymbol{\lambda}}^{(t+1)}$
```

Constraint Factor Updates

The factor $\bar{\boldsymbol{R}}_0$ of the UDU decomposition of the reduced constraint matrix $\bar{\boldsymbol{A}}_0^T$ obtained by selecting rows corresponding to the actual working set $\mathcal{W}_x$ and columns corresponding to the actual working set $\mathcal{W}_y$ is needed for every iteration. Column (augmentation/reduction) updates were already described in Sec. 3.2. Row (update/downdate) updates are described in the following.

Box Constraint Deactivation (Rank-One Update) We have a factor $\bar{\boldsymbol{R}}_0$, which rows and columns are selected according to $\mathcal{W}_x$ and $\mathcal{W}_y$, respectively. By deactivating an ith box constraint, the index is removed from $\mathcal{W}_x$ and factor is updated by corresponding data $\boldsymbol{z}^T = (\bar{\boldsymbol{A}}^T)_{i,\mathcal{W}_y}$. The update is performed using, e.g., Dyadic reduction as follows

$$\left\|\begin{bmatrix}\boldsymbol{d}_0\\1\end{bmatrix};\begin{bmatrix}\boldsymbol{R}_0\\\boldsymbol{z}^T\end{bmatrix}\right\| \ni \left[\begin{array}{ccccc}1&+&+&+&+\\0&1&+&+&+\\0&0&1&+&+\\0&0&0&1&+\\0&0&0&0&1\\\hline +&+&+&+&+\end{array}\right] \to \left[\begin{array}{ccccc}1&\oplus&\oplus&\oplus&\oplus\\0&1&\oplus&\oplus&\oplus\\0&0&1&\oplus&\oplus\\0&0&0&1&\oplus\\0&0&0&0&1\\\hline \ominus&\ominus&\ominus&\ominus&\ominus\end{array}\right] \in \left\|\begin{bmatrix}\boldsymbol{d}_1\\v\end{bmatrix};\begin{bmatrix}\boldsymbol{R}_1\\\boldsymbol{0}\end{bmatrix}\right\|.$$

Box Constraint Activation (Rank-One Dowdate) We have a factor $\bar{\boldsymbol{R}}_0$, which rows and columns are selected according to $\mathcal{W}_x$ and $\mathcal{W}_y$, respectively. By activating an ith box constraint, the index is added to $\mathcal{W}_x$ and factor is updated by corresponding data $\boldsymbol{z}^T = (\bar{\boldsymbol{A}}^T)_{i,\mathcal{W}_y}$. The update is performed using, e.g., Dyadic reduction as follows

$$\left\|\begin{bmatrix}v\\\boldsymbol{d}_0\end{bmatrix};\begin{bmatrix}1&\boldsymbol{0}\\\boldsymbol{q}&\boldsymbol{R}_0\end{bmatrix}\right\| \ni \left[\begin{array}{c|ccccc}1&0&0&0&0&0\\\hline +&1&+&+&+&+\\+&0&1&+&+&+\\+&0&0&1&+&+\\+&0&0&0&1&+\\+&0&0&0&0&1\end{array}\right] \to \left[\begin{array}{c|ccccc}1&\oplus&\oplus&\oplus&\oplus&\oplus\\\hline \ominus&1&\oplus&\oplus&\oplus&\oplus\\\ominus&0&1&\oplus&\oplus&\oplus\\\ominus&0&0&1&\oplus&\oplus\\\ominus&0&0&0&1&\oplus\\\ominus&0&0&0&0&1\end{array}\right] \in \left\|\begin{bmatrix}1\\\boldsymbol{d}_1\end{bmatrix};\begin{bmatrix}1&\boldsymbol{z}^T\\\boldsymbol{0}&\boldsymbol{R}_1\end{bmatrix}\right\|,$$

where $v = 1-\boldsymbol{q}^T\mathrm{diag}(\boldsymbol{d}_0)\boldsymbol{q}$ and $\boldsymbol{q}$ is the solution of equation $\boldsymbol{R}_0^T\mathrm{diag}(\boldsymbol{d}_0)\boldsymbol{q} = \boldsymbol{z}$.

$\boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}}$ Updates

In each iteration t of the projection algorithm, the gradient can be updated along the piece-wise affine path as

$$\boldsymbol{g}(\boldsymbol{x}^{(t+1)}) = \boldsymbol{g}(\boldsymbol{x}^{(t)} + \boldsymbol{d}_{\boldsymbol{x}}^{(t)}) = \boldsymbol{H}(\boldsymbol{x}^{(t)} + \boldsymbol{d}_{\boldsymbol{x}}^{(t)}) + \boldsymbol{f} = \boldsymbol{g}(\boldsymbol{x}^{(t)}) + \boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}}^{(t)}. \quad (3.30)$$

Therefore, the gradient can be computed only once at the beginning of the optimization, and then only cheap updates, as shown below, can be provided. Computation of $\boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}}^{(t)}$ is also required by computation described in Sec. 3.2.

One can possibly compute $(\boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}})^{(t)} = \boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}}^{(t)}$ at every iteration t. A significant computational reduction can be achieved when an ith box constraint is hit. In that case, the update to be performed is

$$(\boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}})^{(t+1)} = \begin{cases} (\boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}}^{(t)})_j, & \forall j \notin \mathcal{W}_x^{(t)}, \\ (\boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}})_j^{(t)} - (\boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}}^{(t)})_j, & \forall j \in \mathcal{W}_x^{(t)}. \end{cases} \tag{3.31}$$

Only elements related to the active box constraints can be updated when a new box constraint is activated. When an affine constraint is activated, the elements related to the non-active constraints could be updated if Lagrange multipliers $\boldsymbol{\lambda}$ are maintained along the projected path. The Lagrange multipliers are not maintained along the projected path in the proposed algorithm.

Summary

The projection algorithm is based on the exact line search described for the gradient method and box-constrained problem in Nocedal and Wright [2006]. We enhanced the method for QP with affine constraints and projecting Newton direction in general rather than gradient. We have enhanced the original algorithm presented in Nocedal and Wright [2006] by the ability to project also onto the affine constraints by maintaining the affine equality constraints as indicated in Alg. 4.

Stopping Condition

Having gradient $\boldsymbol{g} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}$, the residuals of the first two equations in (3.13) are given by $\boldsymbol{\mu}_{\boldsymbol{x}} = \boldsymbol{g} - \boldsymbol{A}^T\boldsymbol{\lambda}$, $\boldsymbol{\mu}_{\boldsymbol{y}} = \boldsymbol{\lambda}$. Let us define (likewise in Dostál [2009]) free residuals φ and chopped residuals β by components i as

$$(\boldsymbol{\varphi}_{\boldsymbol{z}})_i = (\boldsymbol{\mu}_{\boldsymbol{z}})_i, \text{ for } i \notin \mathcal{W}_{\boldsymbol{z}}, \qquad (\boldsymbol{\varphi}_{\boldsymbol{z}})_i = 0, \text{ for } i \in \mathcal{W}_{\boldsymbol{z}}, \tag{3.32a}$$

$$(\boldsymbol{\beta}_{\boldsymbol{z}})_i = 0, \text{ for } i \notin \mathcal{W}_{\boldsymbol{z}}, \qquad (\boldsymbol{\beta}_{\boldsymbol{z}})_i = (\boldsymbol{\mu}_{\boldsymbol{z}}^{\#})_i, \text{ for } i \in \mathcal{W}_{\boldsymbol{z}}, \tag{3.32b}$$

where

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

and subscript $\boldsymbol{z} \in \{\boldsymbol{x}, \boldsymbol{y}\}$ means the same holds for both variables $\boldsymbol{x}$ and $\boldsymbol{y}$. We need to distingues activated upper bound $\mathcal{U}_{\boldsymbol{x}}$ and lower bound $\mathcal{L}_{\boldsymbol{x}}$, such that the following holds $\mathcal{W}_{\boldsymbol{x}} = \mathcal{U}_{\boldsymbol{x}} \cup \mathcal{L}_{\boldsymbol{x}}$ and $\mathcal{U}_{\boldsymbol{x}} \cap \mathcal{L}_{\boldsymbol{x}} = \emptyset$.

Algorithm 4 Projected line search for QP with affine constraints. Given $\boldsymbol{H} \succ \boldsymbol{0}$, $\boldsymbol{A}, \boldsymbol{x}^{(k)}, \boldsymbol{d}_{\boldsymbol{x}}^{(k)}, \boldsymbol{f}$ and bounds $\underline{\boldsymbol{x}}$, $\overline{\boldsymbol{x}}$, $\underline{\boldsymbol{y}}$ and $\overline{\boldsymbol{y}}$ and the working set $\mathcal{W}^{(k)} = \mathcal{W}_x^{(k)} \cup \mathcal{W}_y^{(k)}$, project $\boldsymbol{d}_{\boldsymbol{x}}$ starting from $\boldsymbol{x}^{(k)}$ until cost function stop decreasing.

```
1: $t = 0$
2: $\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(k)}$, $\boldsymbol{d}_{\boldsymbol{x}}^{(t)} = \boldsymbol{d}_{\boldsymbol{x}}^{(k)}$, $(\boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}})^{(t)} = \boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}}^{(k)}$, $\mathcal{W}_x^{(t)} = \mathcal{W}_x^{(k)}$ $\mathcal{W}_y^{(t)} = \mathcal{W}_y^{(k)}$
3: while true do
4:   Breakpoint Computation (3.22) producing $\alpha^{(t)}$
5:   Exact Line Search (3.27) producing $\beta^{(t)}$ for given $\alpha^{(t)}$
6:   Update projection $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} + \beta^{(t)}\boldsymbol{d}_{\boldsymbol{x}}^{(t)}$
7:   Update working set $\begin{cases} \mathcal{W}_x^{(t+1)} = \mathcal{W}_x^{(t)} \\ \cup \{i : \text{box constraint activated by } \beta^{(t)}\} \\ \mathcal{W}_y^{(t+1)} = \mathcal{W}_y^{(t)} \\ \cup \{j : \text{affine constraint activated by } \beta^{(t)}\} \end{cases}$
8:   if $\beta^{(t)} < \alpha^{(t)}$ then
9:     Stop with projection $\boldsymbol{x}^{(t+1)}$
10:  end if
11:  Projected Direction Computation (3.29) producing $\boldsymbol{d}_{\boldsymbol{x}}^{(t+1)}$
12:  Update $(\boldsymbol{H}\boldsymbol{d}_{\boldsymbol{x}})^{(t+1)}$ according to (3.31)
13:  $t = t + 1$
14: end while
```

The same holds for affine bounds denoted by subscript $\boldsymbol{y}$.

For the optimal solution holds that all resiaduals are equal to zero, i.e. the stopping condition is

$$0 = \|\boldsymbol{\varphi_x}\| + \|\boldsymbol{\beta_x}\| = \|\boldsymbol{\varphi_y}\| + \|\boldsymbol{\beta_y}\| \,. \tag{3.33}$$

Proportionality Test and Proportioning

In NPP Šantin [2016], it was observer that proportinality test Dostál [1997] can server as an effective tool for decision when to release constraints from the working set in order to reduce the computation burden of factor updates. Here we extend the idea to affine constraints case.

In the k-th iteration, we work with the working set

$$\mathcal{W}^{(k)} = (\mathcal{W}_x^{(k)}, \mathcal{W}_y^{(k)}), \quad \mathcal{W}_x^{(k)} \subset \{1, \ldots, n\}, \quad \mathcal{W}_y^{(k)} \subset \{1, \ldots, m\} \tag{3.34}$$

and we assume we have a k-th iteration $\boldsymbol{x}^{(k)}$, $\boldsymbol{y}^{(k)}$ satisfying

$$\begin{aligned} (\boldsymbol{x}^{(k)})_i &= \underline{\boldsymbol{x}}_i, \quad \text{or} \quad (\boldsymbol{x}^{(k)})_i = \overline{\boldsymbol{x}}_i, \quad \text{for } i \in \mathcal{W}_x^{(k)}, \\ (\boldsymbol{y}^{(k)})_j &= \underline{\boldsymbol{y}}_j, \quad \text{or} \quad (\boldsymbol{y}^{(k)})_j = \overline{\boldsymbol{y}}_j, \quad \text{for } j \in \mathcal{W}_y^{(k)}, \end{aligned}$$

i.e., the working set $\mathcal{W}^{(k)}$ contains only active constraints.

The proportionality test compares free and chopped residual defined in Sec. 3.2. We say the residuals are **not proportional** when the affect of not optimal active constraints in the working set has larger effect that the free components if and only if

$$\Gamma_{\boldsymbol{x}} \|\boldsymbol{\varphi_x}\| < \|\boldsymbol{\beta_x}\| \,, \text{ or} \tag{3.35a}$$

$$\Gamma_{\boldsymbol{y}} \|\boldsymbol{\varphi_y}\| < \|\boldsymbol{\beta_y}\| \,. \tag{3.35b}$$

Hence the not optimal active constraints in the working set have a more significant effect than the free components.

The proportioning is the process of making residual proportional in the sense of condition by enforcing $\|\boldsymbol{\beta_x}\| = 0$, where $\Gamma_{\boldsymbol{z}}, \boldsymbol{z} \in \{\boldsymbol{x}, \boldsymbol{y}\}$ are the proportionality parameter. The smaller $\Gamma_{\boldsymbol{z}}$ is, the preferable it is to release constraints from the working set. The bigger $\Gamma_{\boldsymbol{z}} > 0$ is, the preferable it is not to release constraints. Please note that the algorithm allows independent control of the preference of active set reduction for box and affine constraints by having a separate decision of residuals. However, we can typically choose $\mathit{\Gamma}_{\boldsymbol{x}} = \mathit{\Gamma}_{\boldsymbol{y}} = 1$. Constraints are released from the working sets as follows

$$\mathcal{W}_{\boldsymbol{x}}^{(k+1)} = \{i \in \mathcal{W}_{\boldsymbol{x}}^{(k)} : (\boldsymbol{\beta_x})_i = 0\}, \tag{3.36a}$$

$$\mathcal{W}_{\boldsymbol{y}}^{(k+1)} = \{i \in \mathcal{W}_{\boldsymbol{y}}^{(k)} : (\boldsymbol{\beta_y})_i = 0\}. \tag{3.36b}$$

Summary

The NPPrimal algorithm is a primal active-set method, which (in its outer loop) finds a step from the k-th iteration to the $(k+1)-$th iteration by solving an equality-constrained QP quadratic subproblem in which some of the inequality constraints and all the equality constraints are imposed as equalities. These active constraints are called the working set $\mathcal{W}^{(k)}$. The solution of this subproblem gives a direction $\boldsymbol{d_x}$, such that $\boldsymbol{x}^{(k)} + \boldsymbol{d_x}$ is a minimizer of the cost function $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{f}$ with respect to the working set $W^{(k)}$.

In the inner loop, the computed direction $\boldsymbol{d_x}$ is gradually projected concerning maintaining the feasibility of the updated approximation of the solution. Thus, in the next step, the feasibility of $\boldsymbol{x}^{(k)} + \boldsymbol{d_x}$ is checked and the largest possible $\alpha \in [0, 1]$ is computed, such that $\boldsymbol{x}^{(k)} + \alpha\boldsymbol{d_x}$ remains feasible. The approximation is updated (ensuring we keep decreasing the cost function), the working set is accordingly expanded, the new projected Newton direction is computed, and the inner loop goes to its next iteration. This approach benefits from the fact that computing the projected Newton direction is computationally substantially cheaper than computing the Newton direction based on the Newton step. Thus we expand the active working set more cheaply. However, the computation is not straightforward in contrast to the NPP algorithm. Due to affine constraints, it requires a solution of the least square subproblem in each inner iteration.

After the projection of the inner loop, associated Lagrange multipliers $\boldsymbol{\lambda}$ are computed. Suppose the computed Newton direction is not a decent direction (when rounding errors are taken into account, this can happen even for nonzero Newton direction) or the active working set was not changed from the previous iteration. In that case, the signs of Lagrange multipliers are examined to decide whether we have reached the optimum or whether some constraints should be released from the active working set. The release of the not optimal constraint indices from the set defining the face problem based on the gradient entries has also been done in Bertsekas [1982]. The main differentiator of the proposed algorithm is to use the proportionality test (Sec. 3.2). When not proportional, all blocking constraints are released. Otherwise, potential blocking constraints are left, and the working set is expanded in the next iteration. This mechanism limits unwanted changes in the active set and reduces the computational complexity.

We enclose this overview of the NPPrimal algorithm with the pseudocode in Algorithm 5.

Convergence

The proof of convergence is based on the idea that the cost function decreases at each iteration and that the number of iterations is finite. In each iteration, the only move of the algorithm is arranged by the projection of the direction towards the face minimizer. In the worst case, the projected direction aims uphill. No move is allowed, and the algorithm picks another face problem. The rigorous proof is left out of here as it closely follows those already published in Šantin [2016] and [2].

Algorithm 5 NPPrimal algorithm. Given $\boldsymbol{H}$, $\boldsymbol{A}, \boldsymbol{x}_0, \boldsymbol{f}$, bounds $\underline{\boldsymbol{x}}$, $\overline{\boldsymbol{x}}$, $\underline{\boldsymbol{y}}$, $\overline{\boldsymbol{y}}$, the initial working set W_0, and maximal number of iterations k_{max}.

```
1:  Set $k = 1$
2:  Gradient Computation $\boldsymbol{g}^{(k)} = \boldsymbol{H}\boldsymbol{x}^{(k)} + \boldsymbol{f}$
3:  **while** $k < k_{max}$ **do**
4:    *{Newton direction computation}*
5:    Newton Direction Computation Sec. 3.15 producing $\boldsymbol{d}_{\boldsymbol{x}}^{(k)}, \boldsymbol{\lambda}^{(k)}$
6:    **if** $0 = \boldsymbol{d}_{\boldsymbol{x}}^{(k)} \cdot \boldsymbol{g}^{(k)}$ (3.17) **then**
7:      *{Newton direction projection}*
8:      Projected Line Search (Alg. 4)
9:      Solve $\boldsymbol{Z}_{\boldsymbol{x}}^T \boldsymbol{A}^T \boldsymbol{\lambda}^{(k+1)} = -\boldsymbol{Z}_{\boldsymbol{x}}^T \boldsymbol{g}^{(k+1)}$ for $\boldsymbol{\lambda}^{(k+1)}$
10:   **else**
11:     $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)}$
12:   **end if**
13:   *{Optimality testing}*
14:   **if** $0 = \|\boldsymbol{\varphi}_{\boldsymbol{x}}\| + \|\boldsymbol{\beta}_{\boldsymbol{x}}\| = \|\boldsymbol{\varphi}_{\boldsymbol{y}}\| + \|\boldsymbol{\beta}_{\boldsymbol{y}}\|$ (3.33) **then**
15:     Stop with the solution $\boldsymbol{x}^{(k+1)}$
16:   **else**
17:     *{Proportionality testing}*
18:     **if** $\Gamma_{\boldsymbol{x}} \|\boldsymbol{\varphi}_{\boldsymbol{x}}\| < \|\boldsymbol{\beta}_{\boldsymbol{x}}\|$ (3.35a) **then**
19:       $\mathcal{W}_{\boldsymbol{x}}^{(k+1)} = \{i \in \mathcal{W}_{\boldsymbol{x}}^{(k)} : (\boldsymbol{\beta}_{\boldsymbol{x}})_i = 0\}$ (3.36a) *{Proportioning in $\boldsymbol{x}$}*
20:     **else**
21:       $\mathcal{W}_{\boldsymbol{x}}^{(k+1)} = \mathcal{W}_{\boldsymbol{x}}^{(k)}$
22:     **end if**
23:     **if** $\Gamma_{\boldsymbol{y}} \|\boldsymbol{\varphi}_{\boldsymbol{y}}\| < \|\boldsymbol{\beta}_{\boldsymbol{y}}\|$ (3.35b) **then**
24:       $\mathcal{W}_{\boldsymbol{y}}^{(k+1)} = \{i \in \mathcal{W}_{\boldsymbol{y}}^{(k)} : (\boldsymbol{\beta}_{\boldsymbol{y}})_i = 0\}$ (3.36b) *{Proportioning in $\boldsymbol{y}$}*
25:     **else**
26:       $\mathcal{W}_{\boldsymbol{y}}^{(k+1)} = \mathcal{W}_{\boldsymbol{y}}^{(k)}$
27:     **end if**
28:   **end if**
29:   $k = k + 1$
30: **end while**
```

4 Numerical Results

The proposed solutions in Chapter 2 and Chapter 3 are modular - the speed profile optimizer (Chapter 2) can use any suitable QP solver, not only NPPro presented in Chapter 3. Hence, benchmark results related to the NPPro solver and speed profile optimization can be read independently.

The NPPro is generated by Matlab Embedded Coder® as a standalone C-code for double and single precision with no dependencies. Interfaces allow you to call the solver from C++, Python, MATLAB, or Simulink. The interface also allows providing a Hessian matrix in full or economic (only upper triangle) data storage format, which might be preferable due to Random Access Memory (RAM) savings. Numerical results were obtained from a computer with an Intel Core(TM) i5-12600HX processor.

An open-source benchmark environment called *qpbenchmark* [Caron et al., 2023] was used. It interfaces state-of-the-art solvers to Python and allows the use of various test sets. Since the NPPro solver is primarily designed for small, dense, strictly convex problems, we restrict the benchmark to the subset of problems with sizes $n \leq 1000$ and $m \leq 1000$ and strictly positive definite Hessian.

Maros and Meszaros

Maros and Meszaros have developed a benchmark problem set for evaluating optimization algorithms' performance [Maros and Mészáros, 1999]. These problems consist of a set of problems with varying sizes (the number of variables and constraints) and difficulty levels and are designed to test the performance of optimization algorithms.

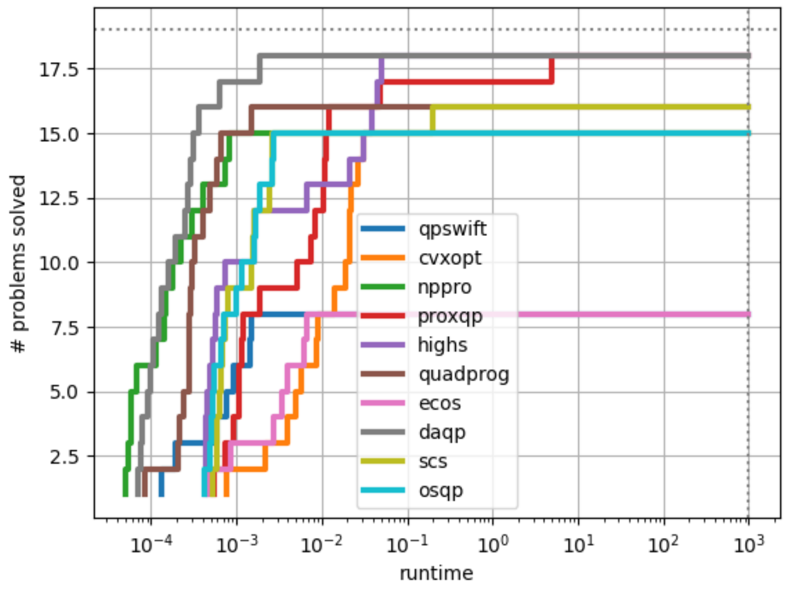


Figure 4.1: Subset of Maros-Meszaros problems with positive definite Hessian and both number of variables and constraints less than or equal a thousand.

The results in Figure 4.1 show how many problems can be solved by a solver within a given runtime (in seconds). It can be seen in the figure the proposed NPPro solver can solve 15 of 19 benchmark problems. NPPro finds the solution very fast compared to other solvers - delivering all 15 cases among the three fastest solvers. On the other hand, there is still room for reliability improvements as four of the challenging problems were not solved properly. Note that *quadprog* refers to Python implementation of quadratic programming [McGibbon, 2023].

MPC Benchmarks

The test set comprises QPs arising from three different MPC problem formulations. The first is motivated by a centroidal model predictive problems, a well-known approach in the community of legged robots when the mass of the legs is neglected compared to the one of the base [Léziart, 2022]. The second is extended walking stabilization based on linear inverted pendulum tracking by quadratic programming-based wrench distribution and a whole-body admittance controller that applies both end-effector and CoM strategies [Caron et al., 2019]. The third comes from MPC for balancing Upkie wheeled biped [Bambade et al., 2023].

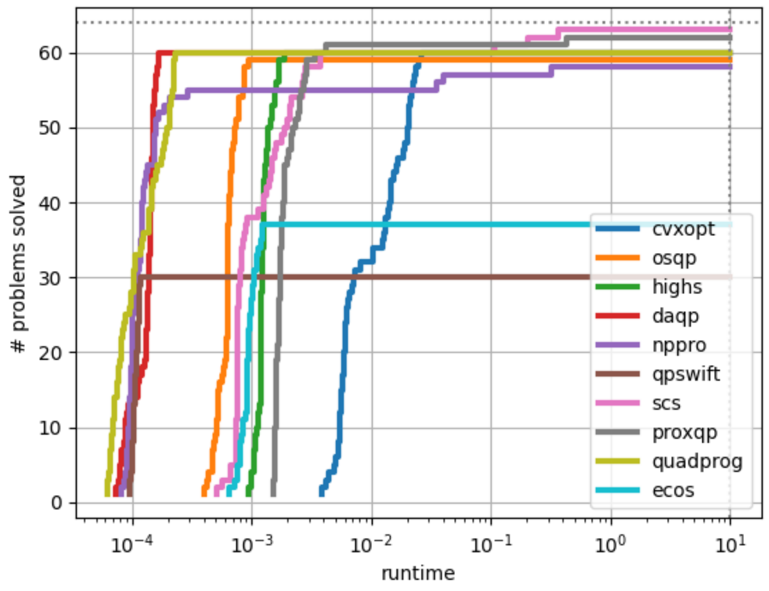


Figure 4.2: Set of QPs arising from MPC problems with positive definite Hessian and both number of variables and constraints less than or equal a thousand.

The comparison of the NPPro solver on the various MPC problems with other state-of-the art solvers is shown in Figure 4.2. Also in this benchmark the NPPro solver is among three fastest QP solvers with total 58 of 64 solved problems.

Oscillating Masses

The NPPro has also been benchmarked on an embedded device with Infineon TC387 (300 MHz). The solved QP problems arise from the oscillating masses benchmark based on Wang and Boyd [2010]. It comprises six masses connected by spring dampers (see Figure 4.3). The first and the last masses are connected to the walls. The weight of each mass is 1 kg, and the spring constant is 1 N/m without damping. The system state $\boldsymbol{x} \in \mathcal{R}^{12}$ represents the displacement and velocity of an individual mass. The displacement is allowed in a $[-4, 4]$ range. Three control inputs, i.e., $\boldsymbol{u} \in \mathcal{R}^3$, represent tensions between different masses. We assume control limits $-0.5 \leq \boldsymbol{u} \leq 0.5$, and the presence of random bounded external disturbance $\boldsymbol{v} \in \mathcal{R}^6$ with a uniform distribution on $[-0.5, 0.5]$, which acts additionally on the displacement state of each mass. See Wang and Boyd [2010] for more details about the setup. The control objective is stabilising each mass's origin using MPC with softened output limits.

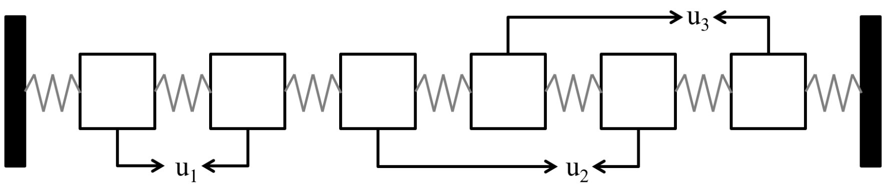
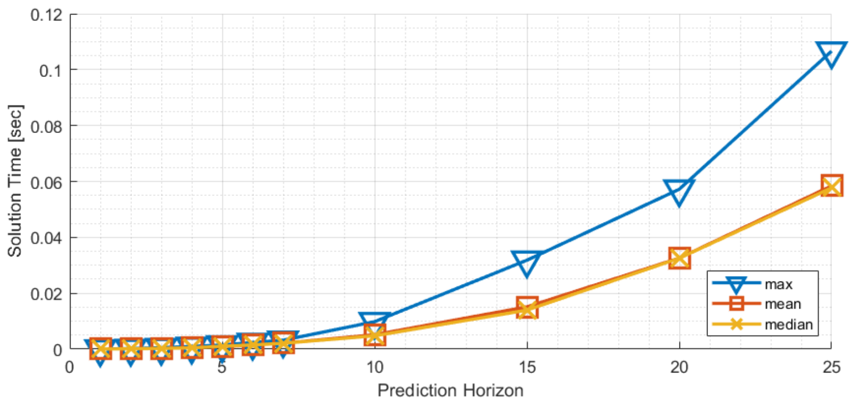


Figure 4.3: Oscillating mass model [Wang and Boyd, 2010]. Boxes represent the masses and dark regions on each side represent walls.

To show the relation of the solver's performance on the number of variables, we perform 100 simulation steps of the model controlled with the prediction horizon $N \in \{1, 2, 3, 4, 5, 6, 7, 10, 15, 20, 25\}$. The experiment was repeated 5 times, and minimum time was taken to eliminate volatility caused by the (non-real-time) operating system. In Figure 4.4, the maximum, mean, and median solution time and number of iterations over the simulation horizon are plotted, respectively.



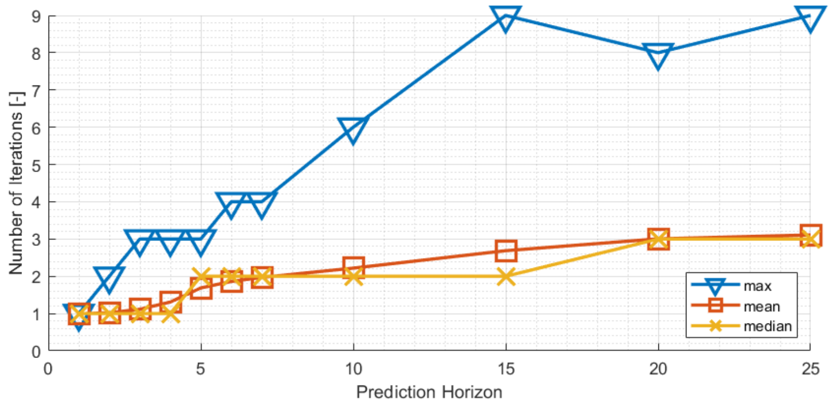


Figure 4.4: Solution times and number of iterations from Oscillating Masses benchmark running on an embedded device with Infineon TC387 (300 MHz).

5 Conclusion

The contribution of this work is twofold. First, it presents an innovative concept for optimizing speed profiles, including traffic light passage planning. A reliable arrangement is proposed, which results, in the end, in solving a moderate series of QP problems. Second, this work introduces an efficient method for solving generic QP problems with affine constraints. The usefulness and reliability of the latter are being proved in practice. All the approaches were developed with a focus on simplicity and reliability. The motivation was to overcome the challenges enumerated in the introduction of this text. I believe, the quality of the speed profile optimization proposed in this work shall also be proven by an application soon.

Bibliography

A. Vahidi, A.S. and Peng, H. (2005). Recursive least squares with forgetting for online estimation of vehicle mass and road grade: theory and experiments. *Vehicle System Dynamics*, 43(1), 31–55.

Andersen, M.S., Dahl, J., Vandenberghe, L., et al. (2013). Cvxopt: A python package for convex optimization. *Available at cvxopt. org*, 54.

Asadi, B. and Vahidi, A. (2011). Predictive cruise control: Utilizing upcoming traffic signal information for improving fuel economy and reducing trip time. *IEEE Transactions on Control Systems Technology*, 19(3), 707 – 714.

Au, T.C., Zhang, S., and Stone, P. (2015). Autonomous intersection management for semi-autonomous vehicles. In *Handbook of Transportation*.

Axehill, D. and Hansson, A. (2008). A dual gradient projection quadratic programming algorithm tailored for model predictive control. In *47th IEEE Conference on Decision and Control*, 3057–3064. IEEE, Cancun.

Bambade, A., Schramm, F., El-Kazdadi, S., Caron, S., Taylor, A., and Carpentier, J. (2023). Proxqp: an efficient and versatile quadratic programming solver for real-time robotics applications and beyond.

Bemporad, A., Bernardini, D., Long, R., and Verdejo, J. (2018). Model predictive control of turbocharged gasoline engines for mass production. In *WCX World Congress Experience*. SAE International.

Beno, R., Pachner, D., and Vladimir, H. (2017). Robust numerical approach to steady-state calibration of mean-value models. *Control Engineering Practice*, 61, 186–197.

Benzi, M. and Wang, Z. (2011). Analysis of augmented lagrangian-based preconditioners for the steady incompressible navier–stokes equations. *SIAM Journal on Scientific Computing*, 33(5), 2761–2784.

Benzi, M. and Wathen, A.J. (2008). *Some Preconditioning Techniques for Saddle Point Problems*, 195–211. Springer Berlin Heidelberg, Berlin, Heidelberg.

Bertsekas, D.P. (1982). Projected Newton Methods for Optimization Problems with Simple Constraints. *SIAM J. Control Optim.*, 20(2), 221–246. doi: 10.1137/0320018.

Caron, S., Kheddar, A., and Tempier, O. (2019). Stair climbing stabilization of the hrp-4 humanoid robot using whole-body admittance control. In *2019 International Conference on Robotics and Automation (ICRA)*, 277–283.

Caron, S., Zaki, A., Otta, P., Arnström, D., and Carpentier, J. (2023). qpbenchmark: Benchmark for quadratic programming solvers available in Python. URL https://github.com/qpsolvers/qpbenchmark.

Cimini, G., Bemporad, A., and Bernardini, D. (2017). ODYS QP Solver. ODYS S.r.l. (http://odys.it/qp).

Dahl, J., Wassén, H., Santin, O., Herceg, M., Lansky, L., Pekar, J., and Pachner, D. (2018). Model predictive control of a diesel engine with turbo compound and exhaust after-treatment constraints. *IFAC-PapersOnLine*, 51(31), 349–354.

D'Amato, A., Ozatay, E., Michelini, J., Szwabowski, S., Filev, D., Santin, O., Pekar, J., and Beran, J. (2016). Cruise controller with fuel optimization based on adaptive nonlinear predictive control. *SAE International Journal of Passenger Cars - Electronic and Electrical Systems*, 9(2), 262–274.

Davis, H.H. and Niehaus, F.W. (1926). PERSISTENT OMPHALOMESENTERIC DUCT AND URACHUS IN THE SAME CASE. *Journal of the American Medical Association*, 86(10), 685–686.

Domahidi, A., Chu, E., and Boyd, S. (2013). Ecos: An socp solver for embedded systems. In *2013 European Control Conference (ECC)*, 3071–3076.

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

Dostál, Z. (1997). Box constrained quadratic programming with proportioning and projections. *SIAM J. Optim.*, 7(3), 871–887.

Ferreau, H.J., Kirches, C., Potschka, A., Bock, H.G., and Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Math. Program. Comput.*, 6(4), 1–37.

Goldfarb, D. and Idnani, A. (1983). A numerically stable dual method for solving strictly convex quadratic programs. *Math. Program.*, 27(1), 1–33.

Gould, N. and Toint, P.L. (2004). Preprocessing for quadratic programming. *Math. Program.*, 100(1), 95–132.

Greif, C. and Schötzau, D. (2007). Preconditioners for the discretized time-harmonic maxwell equations in mixed form. *Numerical Linear Algebra with Applications*, 14(4), 281–297.

Karnik, A.Y., Fuxman, A., Bonkoski, P., Jankovic, M., and Pekar, J. (2016). Vehicle powertrain thermal management system using model predictive control. *SAE International Journal of Materials and Manufacturing*, 9(3), 525–533.

Kirches, C., Bock, H.G., Schlöder, J.P., and Sager, S. (2011). A factorization with update procedures for a kkt matrix arising in direct optimal control. *Mathematical Programming Computation*, 3(4), 319 – 348.

Liesen, J. and Tichý, P. (2014). Convergence analysis of krylov subspace methods. *GAMM-Mitteilungen*, 27(2), 153–173.

Léziart, P.A. (2022). *Locomotion control of a lightweight quadruped robot*. Ph.D. thesis.

Mancuso, G.M. and Kerrigan, E.C. (2011). Solving constrained lqr problems by eliminating the inputs from the qp. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 507–512.

Mandava, S., Boriboonsomsin, K., and Barth, M. (2009). Arterial velocity planning based on traffic signal information under light traffic conditions. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, 1–6.

Maros, I. and Mészáros, C. (1999). A repository of convex quadratic programming problems. *Optimization Methods and Software*, 11.

McGibbon, R. (2023). quadprog (python). URL https://github.com/quadprog/quadprog.

Mészáros, C. and Suhl, U. (2003). Advanced preprocessing techniques for linear and quadratic programming. *OR Spectrum*, 25, 575–595.

Nedic, A. and Ozdaglar, A. (2009). Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1), 48–61.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer, New York, 2nd edition.

Paige, C.C. and Saunders, M.A. (1975). Solution of sparse indefinite systems of linear equations. *SIAM J. Numerical Analysis*, 12, 617–629.

Pandala, A.G., Ding, Y., and Park, H.W. (2019). qpswift: A real-time sparse quadratic program solver for robotic applications. *IEEE Robotics and Automation Letters*, 4(4), 3355–3362.

Peterka, V. (1986). Control of uncertain processes: applied theory and algorithms. *Kybernetika*, 22(7), 1–3.

Quirynen, R., Knyazev, A., and Di Cairano, S. (2018). Block Structured Preconditioning within an Active-Set Method for Real-Time Optimal Control . In *Proceedings of the European Control Conference*. IEEE, Brussels.

Raphael, C., Pascal, G., Eric C., K., and Morari, M. (2007). Move blocking strategies in receding horizon control. *Journal of Process Control*, 17(6), 563 – 570.

Richter, S., Jones, C.N., and Morari, M. (2012). Computational Complexity Certification for Real-Time MPC With Input Constraints. *IEEE Trans. Autom. Control*, 57(6), 1391–1403.

Sampei, M. and Furuta, K. (1986). On time scaling for nonlinear systems: Application to linearization. *IEEE Transactions on Automatic Control*, 31(5), 459–462.

Sankar, G., Shekhar, R., Manzie, C., Sano, T., and Nakada, H. (2019). Model predictive controller with average emissions constraints for diesel airpath. *Control Engineering Practice*, 90, 182–189.

Šantin, O. (2016). Numerical algorithms of quadratic programming for model predictive control. *Dissertation thesis, CTU Prague.*

Segovia, P., Puig, V., Duviella, E., and Etienne, L. (2021). Distributed model predictive control using optimality condition decomposition and community detection. *Journal of Process Control*, 99, 54–68.

Sotoudeh, S.M. and Homchaudhuri, B. (2020). A robust mpc-based hierarchical control strategy for energy management of hybrid electric vehicles in presence of uncertainty. *2020 American Control Conference (ACC)*, 3065–3070.

Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2020). OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4), 637–672.

Šantin, O. and Havlena, V. (2011). Combined Gradient and Newton Projection Quadratic Programming Solver for MPC. In *18th IFAC World Congress*, 5567–5572. IFAC, Milano.

Wan, N., Vahidi, A., and Luckow, A. (2016). Optimal speed advisory for connected vehicles in arterial roads and the impact on mixed traffic. *Transportation Research Part C: Emerging Technologies*, 69, 548–563.

Wang, Y. and Boyd, S. (2010). Fast model predictive control using online optimization. *IEEE Trans. Control Syst. Technol.*, 18(2), 267–278.

Xia, H., Boriboonsomsin, K., Schweizer, F., Winckler, A., Zhou, K., Zhang, W.B., and Barth, M. (2012). Field operational testing of eco-approach technology at a fixed-time signalized intersection. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 188–193.

Author’s publications

Related to the thesis

Published journal papers

[1] L. Eriksson, A. Thomasson, K. Ekberg, A. Reig, M. Eifert, F. Donatantonio, A. D’Amato, I. Arsie, C. Pianese, P. Otta, M. Held, U. Vögele, Ch. Endisch, ”**Look-ahead controls of heavy duty trucks on open roads — six benchmark solutions**,”*Control Engineering Practice*, vol. 83, pp. 45-66, 2019.

[2] P. Otta, J. Burant, O. Šantin, V. Havlena, "**Newton projection with proportioning using iterative linear algebra for model predictive control with long prediction horizon**,” *Optimization Methods and Software*, vol. 91, pp. 1075-1098, 2019.

Conditionally accepted journal papers

[3] P. Otta, O. Šantin, V. Havlena, "**NPPro: A Newton Projection with Proportioning Solver for Quadratic Programming with Affine Constraints**," *Optimization Methods and Software*, Submitted in June 2023.

Published conference papers

[4] P. Otta, O. Šantin, V. Havlena, "**Tailored QP algorithm for Predictive Control with dynamics penalty**," in *22nd Mediterranean Conference on Control and Automation (MED), Palermo, Italy, 2014*, pp. 5396–5401.

[5] P. Otta, O. Šantin, V. Havlena, "**On the Quadratic Programming**

Solution for Model Predictive Control with Move Blocking," in *23rd International Conference on Process Control, Strbske Pleso, Slovakia, 2021*, pp. 9177–9182.

Not related to the thesis

Published conference papers

[6] P. Otta, O. Šantin, V. Havlena, "**Measured-state driven warm-start strategy for linear MPC**," in *14th European Control Conference, Linz, Austria, 2015*, pp. 1864–1869.

Published patent

[7] D. Pachner, V. Havlena, P. Otta, "**Method and apparatus for tuning a regulatory controller**," *US Patent App. 17/141,806, 2022.*