KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

# Penalty and Augmented Lagrangian Methods for Model Predictive Control

**Ben Hermans**

Supervisors:
Prof. dr. ir. P. Patrinos
Prof. dr. ir. G. Pipeleers

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Mechanical Engineering

June 2021

# Penalty and Augmented Lagrangian Methods for Model Predictive Control

**Ben HERMANS**

Examination committee:
Prof. dr. ir. P. Verbaeten, chair
Prof. dr. ir. P. Patrinos, supervisor
Prof. dr. ir. G. Pipeleers, supervisor
Prof. dr. ir. J. Swevers
Prof. dr. ir. S. Vandewalle
Prof. dr. ir. J. Suykens
(KU Leuven)
Prof. dr. M. Diehl
(Albert-Ludwigs-Universität Freiburg)
Prof. dr. A. Themelis
(Kyushu University)

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Mechanical Engineering

June 2021

# Preface

Onderwerp: doctoreren? Ik herinner me nog levendig het moment waarop ik een mail kreeg van Goele met de uitnodiging om eens te komen bespreken of een doctoraat iets voor mij zou zijn. Na overleg met haar en mijn toekomstige co-promotor Panos bleek dat ze iemand zochten voor een onderzoeksproject in numerieke optimalisatie, met als toepassingsgebied modelgebaseerde predictieve controle. Gegeven mijn interesse in numerieke wiskunde heb ik deze kans met beide handen gegrepen, en aan het eind van de zomervakantie begon ik aan een vierjarig avontuur.

Hoewel ik Panos en Goele niet goed kende op voorhand, bleek het al snel dat ik met hen als promotoren met mijn gat in de boter was gevallen. Ze zijn allebei een krak in hun vak, en ze wisten me daarboven altijd goede raad te geven en me bij te sturen, zowel in de details als in het grote plaatje. Ik wil hen dan ook van harte bedanken voor de uitstekende begeleiding die ik heb mogen genieten. Zonder hen was dit proefschrift niet mogelijk geweest.

I would like to especially express my gratitude to Panos, who became my main supervisor during the later years of my PhD. His mathematical rigor was often very useful in grounding my more intuitive thinking. Furthermore, I always admired how fast he was in making connections with existing work and how closely involved he was in my research. It is clearly a sign of his passion regarding the topics, and it is undeniable that some of that has rubbed off on me. For this, for the demanding but enticing research environment that he provided, and for his overall good-natured character, I am deeply grateful.

During the latter half of my PhD, I also had the extremely good fortune of working together with Andreas Themelis, who was a postdoc student under the wings of Panos at that time. His outstanding theoretical knowledge of optimization beautifully complemented my more practical skills, and I sincerely believe our cooperation produced something that was greater than the sum of its parts. I would like to thank him also for reading the manuscript and providing extensive and helpful feedback. Outside of research he can only be described as a very agreeable person, and I think the amount of times we were laughing together is close to infinity, although I am sure he would argue there is no such thing. Moreover, just before he left to Japan, I learned that he also knows a thing or two about table tennis. Luckily, I know a thing or two more.

Daarboven zou ik graag alle leden van de Meco onderzoeksgroep, onder leiding van Goele en Jan, van harte willen bedanken voor de mooie jaren. Bij mijn intrede hebben het komische trio van Tim, Maarten en Ruben, de toenmalige senioren van de groep, en Laurens zich over mij ontfermd. Het begon al goed

toen ik samen met hen en Goele in het eerste weekend van mijn doctoraat deelnam aan de Spartacus Run, een parcours van tien kilometer met veel obstakels, modder, water en zand. Iedere middag aten we ook samen lunch in de cafetaria en werd ik beetje bij beetje ingewijd in het reilen en zeilen van de groep. Zo kwamen de (straffe) verhalen van conferenties en van andere mensen in de groep en het departement al snel naar boven. Daarboven was er vaak wel één of ander sociaal evenement gepland, zoals een uitstap naar de Duvel brouwerij, de jaarlijkse barbecue bij Jan, de jaarlijkse Beneluxmeeting, een avondje bowlen, een exclusief nieuwjaarsdiner, de Meco hike en (in tijden van corona) de Meco get moving challenge. Zo deed ik gelukkig heel af en toe toch ook iets aan sport. Bedankt, Goele, Jan, Laurens, Ruben, Maarten, Tim, Joris, Daniele, Taranjit, Armin, Massimo, Dora, Niels, Erik, Andreas, Bastiaan, Alejandro, Ajay, Mathias, Dries, David, Wim, András, Tommaso, Bence, Le, Matteo, Laurane en Wilm.

I am also grateful to have met and interacted with some of the members of the research group under Panos, namely Pantelis, Puya, Masoud, Mathijs and Peter. I would like to thank Aleksandra, Peter and David, for the collaborations on projects that did not appear in this thesis.

Buiten het werk heb ik nog een hele waslijst aan mensen te bedanken voor hun positieve invloed op de kwaliteit van mijn leven. De prijs voor de eerste plaats gaat naar mijn lief, Eva. Ze staat mij al gedurende vier jaar bij langs de zijlijn. Van mijn onderwerp heeft ze misschien geen kaas gegeten, maar ik weet dat ik op haar kan rekenen en dat ze me steunt in wat ik wil doen. Ik hou dan ook ontzettend veel van haar. Als je dit leest, nog eens bedankt lieve schat. Graag vermeld ik ook even haar ouders die ons menigmaal hebben ontvangen voor een weekje vakantie aan zee. Bedankt, Christine en Johan.

Mijn broer, Raf, zou ik ook ten zeerste willen bedanken. Na ontelbaar veel “ah ket ja ket’s” heeft hij me een paar jaar geleden er dan toch van overtuigd om terug tafeltennis te komen spelen. Hiervan heb ik enorm genoten als tijdsverdrijf, niet alleen door de leuke sport maar ook door zijn aanwezigheid, want we trappen samen veel lol, en daarvoor ben ik hem enorm dankbaar. Daarbuiten zou ik graag Klaas en Taranjit willen bedanken als frequente (voormalige) tafeltennis partners, en David, Margo en Raijko als trainers. Natuurlijk mag ik de leden van mijn club Sukarti in Tienen niet vergeten. Bedankt, Metin, Aline, Jan, Patrick, Jonas, Borris, Kevin, Annelies, Nathan, Wouter, Guy en Jakob.

Verder ben ik veel dank verschuldigd aan mijn ouders. Hoewel ik in deze fase van mijn leven het ouderlijke nestje verlaten heb en eindelijk een beetje volwassen geworden ben, door onder andere mijn eigen was te doen, weet ik zeker dat ik niet geraakt zou zijn waar ik nu ben zonder hen. Ze hadden altijd en hebben nog steeds alleen maar het beste voor met mij, en daarvoor ben ik hen dankbaar.

Ook bedankt aan hun partners. Snep en Menoubie, als jullie dit lezen, een hele dikke “Maa besaaisch!” Mama, voor jou een even dikke “Dag de mammie!”

I would also like to express my deep appreciation for Peter Ralston, whose spirit of investigation and questioning has impacted me in very profound ways. In the same vein, I am deeply grateful to Immanuel Kant for his earth-shattering *Critique of Pure Reason*, which was a great inspiration to me. On an inspirational note, I would like to thank NF for his rap music.

Tot slot zou ik graag alle leden van mijn jury, professoren Jan Swevers, Stefan Vandewalle, Johan Suykens, Moritz Diehl en Andreas Themelis, van harte willen bedanken voor het zorgvuldig lezen en beoordelen van mijn proefschrift.

# Contents

# Abstract

Automation will undoubtedly grow to become the cornerstone of various industries, such as manufacturing, automatic transportation, agriculture, and household appliances. Novel challenging control applications continue to emerge, leading to increased interest in advanced control methodologies, such as the optimization based model predictive control (MPC), since they possess inherent capability to include complex objectives and constraints in their problem formulation. The integration of MPC in practice has not been without roadblocks, however. Challenges arise both in modeling complex scenarios and solving the resulting optimization problems in real-time. This thesis uses penalty and augmented Lagrangian approaches, a class of strategies for constrained optimization, to make headway in both of these areas.

In a first part, this thesis considers the application of autonomous navigation in environments with obstacles of general shapes. Previous research in optimization based autonomous navigation is restricted to circular, rectangular, polygonal or obstacles of convex shape. In contrast, we consider obstacles of which the boundaries are defined by smooth functions, allowing for much more generality in the shapes. Although the resulting constraints are nonsmooth, this thesis shows how a quadratic penalty method is theoretically sound and extremely efficient in practice in solving such problems.

In a second part, this thesis constructs a novel quadratic programming (QP) solver, QPALM, based on the proximal augmented Lagrangian method. The inner minimization is tailored to QPs by making use of efficient semismooth Newton directions and optimal step sizes. Research on QP solvers is typically restricted to convex QPs. In contrast, we also consider the possibility of finding stationary points of nonconvex QPs, relying only on minor modifications of the algorithm. QPALM is shown to theoretically exhibit a linear (outer) convergence rate, even for nonconvex QPs, and demonstrated to possess a unique combination of robustness and efficiency when compared to state-of-the-art QP solvers.

# Korte inhoud

Automatisering zal ongetwijfeld de pijler zijn van verschillende industrieën, zoals productie, automatisch transport, agricultuur en huishoudtoestellen. Nieuwe uitdagende toepassingen zorgen voor verhoogde interesse in geavanceerde regeltechnieken zoals modelgebaseerde predictieve controle (MPC), aangezien deze inherent over de mogelijkheid beschikt om complexe objectieve en beperkingen in rekening te brengen. De verspreiding van MPC in de praktijk is echter niet zonder struikelblokken. Zowel in het modelleren van complexe scenario's als in het oplossen van de resulterende optimalisatieproblemen in real time zijn er nog openstaande vragen. Om vooruitgang te boeken in deze twee domeinen maakt dit proefschrift gebruik van penalisatie en geaugmenteerde Lagrangiaanse methodes, een klasse van strategieën voor numerieke optimalisatie.

In het eerste deel van dit proefschrift wordt de toepassing van autonome navigatie in een omgeving met generieke obstakels beschouwd. Voorafgaand onderzoek in navigatie gebaseerd op optimalisatie is beperkt tot cirkelvormige, rechthoekige, veelhoekige en convexe obstakels. We beschouwen daarentegen een algemenere formulatie waarin de grenzen van de obstakels gedefinieerd zijn door gladde functies. Hoewel de resulterende beperkingen niet glad zijn, toont dit proefschrift toch hoe een kwadratische penalisatie methode in theorie gefundeerd is en in de praktijk tot een zeer efficiënte oplossing leidt van zulke problemen.

In het tweede deel van dit proefschrift is het onderwerp QPALM, een nieuwe solver voor kwadratische programmering (QP) op basis van de proximale geaugmenteerde Lagrangiaanse methode. De interne minimalisatie is afgestemd op QPs door gebruik te maken van efficiënte halfgladde Newton richtingen en optimale stapgroottes. Onderzoek in QP solvers is typisch beperkt tot convexe problemen. We beschouwen daarentegen ook de mogelijkheid om stationaire punten te vinden van niet convexe problemen na enkele geringe aanpassingen aan het algoritme. Er wordt theoretisch bewezen dat QPALM aan lineaire snelheid convergeert voor zowel convexe als niet convexe problemen. Daarboven tonen numerieke simulaties dat QPALM over een unieke combinatie van robuustheid en efficiëntie beschikt in vergelijking met bestaande QP solvers.

# List of symbols

| | | |
|---|---|---|
| $\mathbf{1}_n$ | $\mathbb{R}^n$ vector with all elements equal to 1 | 7 |
| $A_{\cdot j}$ | The $j$-th column of $A$ | 6 |
| $A_{ij}$ | The element in the $i$-th row and $j$th column of $A$ | 6 |
| $\{a^k\}_{k\in K}$ | Sequence indexed by elements of the set $K$ | 8 |
| $A_{i\cdot}$ | The $i$-th row of $A$ | 6 |
| $A_{\mathcal{I}\cdot}$ | The submatrix composed of the rows $A_{i\cdot}$ for $i \in \mathcal{I}$ | 6 |
| $A_{\mathcal{I}\mathcal{J}}$ | The submatrix composed of the rows $i \in \mathcal{I}$ and columns $j \in \mathcal{J}$ of $A$ | 6 |
| $A_{\cdot\mathcal{J}}$ | The submatrix composed of the columns $A_{\cdot j}$ for $j \in \mathcal{J}$ | 6 |
| $A^{\mathcal{I}\cdot}$ | $A^{\mathcal{I}\cdot}_{i\cdot} = A_{i\cdot}$ if $i \in \mathcal{I}$ and $\mathbf{0}$ otherwise | 6 |
| $A^{\mathcal{I}\mathcal{J}}$ | $A^{\mathcal{I}\mathcal{J}}_{ij} = A_{ij}$ if $(i,j) \in \mathcal{I}\times\mathcal{J}$ and 0 otherwise | 7 |
| $A^\top$ | Transpose of matrix $A$ | 7 |
| $\partial O$ | Boundary of set $O$ | 8 |
| $C^{1,1}(\mathbb{R}^n)$ | Differentiable functions $\mathbb{R}^n \to \mathbb{R}$ with Lipschitz gradient | 9 |
| $C^k(\mathbb{R}^n)$ | $k$ times continuously differentiable functions $\mathbb{R}^n \to \mathbb{R}$ | 9 |
| $\lvert\mathcal{J}\rvert$ | Cardinality of set $\mathcal{J}$ | 8 |
| $\overline{O}$ | Closure of set $O$ | 8 |
| $h^*$ | Fenchel conjugate of h | 9 |
| $\delta_S$ | Indicator function of set $S$ | 9 |
| $\operatorname{diag}(v)$ | Diagonal matrix with the elements of $v$ on the diagonal | 7 |
| $\operatorname{dist}(x,S)$ | Distance of $x$ from $S$ | 10 |
| $\operatorname{dom} h$ | Domain of extended-real- or set-valued mapping $h$ | 8 |
| $\mathrm{E}(r)$ | Expected value of $r$ | 8 |
| $\operatorname{epi} h$ | Epigraph of extended-real-valued function $h$ | 8 |
| $\operatorname{fix} F$ | Fixed set of (set-valued) mapping $F$ | 10 |
| $h : A \to B$ | Single-valued function | 8 |
| $h^\gamma$ | Moreau envelope of $h$ | 10 |
| $h^\Sigma$ | Moreau envelope of $h$ with matrix step size | 11 |
| $\mathrm{I}$ | Identity matrix of suitable size | 7 |
| $\mathrm{I_n}$ | Identity $n \times n$ matrix | 7 |
| $\mathrm{id}$ | Identity mapping | 9 |
| $\operatorname{int} O$ | Interior of set $O$ | 8 |

| | | |
|---|---|---|
| $L_h$ | Lipschitz modulus of $\nabla h$, for $h \in C^{1,1}(\mathbb{R}^n)$ | 9 |
| $\lambda_{\max}(Q)$ | Maximum eigenvalue of $Q \in \mathrm{Sym}(\mathbb{R}^n)$ | 7 |
| $\lambda_{\min}(Q)$ | Minimum eigenvalue of $Q \in \mathrm{Sym}(\mathbb{R}^n)$ | 7 |
| $\mathrm{lev}_{\leq \alpha} h$ | $\alpha$-sublevel set of extended-real-valued function $h$ | 8 |
| $N$ | Control horizon | 6 |
| $\mathbb{N}$ | Natural numbers $\{0, 1, 2, \cdots\}$ | 6 |
| $\lvert A \rvert$ | Number of nonzeros of matrix $A$ | 7 |
| $\lVert \cdot \rVert$ | Euclidean or matrix norm | 7 |
| $\mathcal{N}(\mu, \sigma^2)$ | Normal distribution with mean $\mu$ and standard deviation $\sigma$ | 7 |
| $\mathcal{N}_S(z)$ | The normal cone of the set $S$ at $z$ | 9 |
| $\lVert \cdot \rVert_p$ | $\ell_p$ (semi)norm, for $p \in [1, \infty]$ | 7 |
| $\lVert \cdot \rVert_Q$ | (Semi)norm induced by $Q \in \mathrm{Sym}_+(\mathbb{R}^n)$ | 7 |
| $[r]_+$ | (Elementwise) Positive part of $r$: $\max\{0, r\}$ | 6 |
| $[r]_-$ | (Elementwise) Negative part of $r$: $\max\{0, -r\}$ | 6 |
| $\Pi_S$ | (Set-valued) projection onto $S$ | 9 |
| $\mathrm{prox}_{\gamma h}$ | (Set-valued) proximal mapping of $h$ | 10 |
| $\mathrm{prox}_h^{\Sigma}$ | (Set-valued) proximal mapping of $h$ with matrix step size | 11 |
| $q$ | State vector over all time instances, $q = (q_0, q_1, \cdots, q_N)$ | 6 |
| $q_k$ | State vector at time k | 6 |
| $\mathbb{R}_+$ | Positive reals $[0, \infty)$ | 6 |
| $\mathbb{R}_{++}$ | Strictly positive reals $(0, \infty)$ | 6 |
| $\overline{\mathbb{R}}$ | Extended-real numbers $(-\infty, \infty]$ | 6 |
| $\mathbb{R}$ | Real numbers $(-\infty, \infty)$ | 6 |
| $\partial h$ | (Limiting) subdifferential of $h$ | 10 |
| $\hat{\partial} h$ | Regular subdifferential of $h$ | 10 |
| $\prec \preceq \succeq \succ$ | Partial order relations in $\mathrm{Sym}(\mathbb{R}^n)$ | 7 |
| $\mathrm{Sym}(\mathbb{R}^n)$ | Symmetric $n \times n$ real matrices | 7 |
| $\mathrm{Sym}_+(\mathbb{R}^n)$ | Symmetric $n \times n$ positive semidefinite real matrices | 7 |
| $\mathrm{Sym}_{++}(\mathbb{R}^n)$ | Symmetric $n \times n$ positive definite real matrices | 7 |
| $u$ | Input vector over all time instances $u = (u_0, u_1, \cdots, u_N)$ | 6 |
| $u_k$ | Input vector at time k | 6 |
| $\mathcal{U}(a, b)$ | Uniform distribtion on the interval $[a, b]$ | 7 |
| $\mathrm{Var}(r)$ | Variance of $r$ | 8 |
| $x$ | Primal optimization variables | 6 |
| $x_i$ | The i-th element of x | 6 |
| $y$ | Dual optimization variables | 6 |
| $\mathrm{zer}\, F$ | Zeros of (set-valued) mapping $F$ | 10 |

# List of abbreviations

| | |
|---|---|
| AD | Automatic differentiation |
| ADMM | Alternating direction method of multipliers |
| AGV | Autonomous guided vehicle |
| ALM | Augmented Lagrangian method |
| AMA | Alternating minimization algorithm |
| BFGS | Broyden, Fletcher, Goldfarb, and Shanno |
| DRS | Douglas-Rachford splitting |
| ECQP | Equality constrained quadratic program |
| FBE | Forward-backward envelope |
| FBS | Forward-backward splitting |
| iff | If and only if |
| KKT | Karush-Kuhn-Tucker |
| L-BFGS | Limited memory variant of BFGS |
| LADEL | LDL with Add and DELete update routines |
| LGPL v3 | GNU Lesser General Public License version 3 |
| LICQ | Linear independence constraint qualification |
| LOBPCG | Locally optimal block preconditioned conjugate gradient |
| lsc | Lower semicontinuous |
| MCFQ | Mangasarian-Fromovitz constraint qualification |
| MFCQ | Mangasarian-Fromovitz constraint qualification |
| MHE | Moving horizon estimation |
| MPC | Model predictive control |
| MPCC | Mathematical program with vertical complementarity constraints |
| MPCC-LICQ | LICQ for MPCCs |
| MPSEC | Mathematical program with set exclusion constraints |
| OCP | Optimal control problem |
| OSQP | Operator-splitting QP |
| P-ALM | Proximal augmented Lagrangian method |
| PANOC | Proximal averaged Newton-type method for optimal control |

| | |
|---|---|
| PPA | Proximal point algorithm |
| QP | Quadratic program |
| sgm | Shifted geometric means |
| SMFCQ | Strict Mangasarian-Fromovitz constraint qualification |
| SQP | Sequential quadratic programming |
| SR1 | Symmetric rank-1 |

# Vita

| | |
|---|---|
| **August 8, 1995** | Born, Leuven, Belgium |
| **2012–2014** | B.Sc. in Polytechnics<br>Not finished<br>Royal Military Academy, Belgium |
| **2014–2016** | B.Sc. in Engineering Science<br>Final mark: 83.33% Magna cum laude<br>KU Leuven, Belgium |
| **2015–2017** | M.Sc. in Mechanical Engineering: option Aerospace<br>Final mark: 84.83% Summa cum laude<br>KU Leuven, Belgium |
| **Since 2017** | Ph.D. in Mechanical Engineering<br>KU Leuven, Belgium |

# Publications

## Journal papers

[97] A. Lekić, **B. Hermans**, N. Jovičić and P. Patrinos,
*Microsecond nonlinear model predictive control for DC-DC converters,*
International Journal of Circuit Theory and Applications, Vol. 48, 2020, pp. 406–419
https://onlinelibrary.wiley.com/doi/full/10.1002/cta.2737

[83] **B. Hermans**, G. Pipeleers and P. Patrinos,
*A penalty method for nonlinear programs with set exclusion constraints,*
Automatica, Vol. 127, 2021, Article. 109500
https://www.sciencedirect.com/science/article/abs/pii/S0005109821000200

[85] **B. Hermans**, A. Themelis and P. Patrinos,
*QPALM: A proximal augmented Lagrangian method for nonconvex quadratic programs,*
Mathematical Programming Computation, 2021, Submitted
https://arxiv.org/abs/2010.02653

## Conference proceedings

[82] **B. Hermans**, P. Patrinos and G. Pipeleers,
*A penalty method based approach for autonomous navigation using nonlinear model predictive control,*
6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018, pp. 268-274
https://www.sciencedirect.com/science/article/pii/S2405896318326739

[84] **B. Hermans**, A. Themelis and P. Patrinos,
*QPALM: A Newton-type proximal augmented Lagrangian method for quadratic programs,*
58th IEEE Conference on Decision and Control (CDC), 2019, pp. 4325–4330.
https://ieeexplore.ieee.org/document/9030211

[149] A. Themelis, **B. Hermans** and P. Patrinos,
*A new envelope function for nonsmooth DC optimization,*

59th IEEE Conference on Decision and Control (CDC), 2020, pp. 4697–4702.
https://ieeexplore.ieee.org/document/9304514

[35] P. Coppens, **B. Hermans**, J. Vandersteen, G. Pipeleers and P. Patrinos,
*A new heuristic approach for low-thrust spacecraft trajectory optimization,*
IFAC World Congress, Germany, 2020 (to appear)
ftp://ftp.esat.kuleuven.be/pub/stadius/pcoppens/IFAC_2020/ifac2020pcoppens.pdf

## Abstracts/Presentations/Posters

1. **B. Hermans**, G. Pipeleers and P. Patrinos,
   *A penalty method algorithm for obstacle avoidance using nonlinear model predictive control,*
   37th Benelux Meeting on Systems and Control, Soesterberg, The Netherlands, March 27-29, 2018.
2. **B. Hermans**, A. Themelis, G. Pipeleers and P. Patrinos,
   *QPALM: augmented Lagrangian method for quadratic programs,*
   38th Benelux Meeting on Systems and Control, Center Parcs "De Vossemeren", Belgium, March 19-21, 2019.
3. **B. Hermans**, G. Pipeleers and P. Patrinos,
   *The proximal augmented Lagrangian method for nonconvex quadratic programs,*
   39th Benelux Meeting on Systems and Control, Hotel Mennorode, Elspeet, The Netherlands, March 10-12 2020.

# Chapter 1

## Introduction

Model predictive control (MPC) was introduced in the 1980s as a control method in chemical process plants and oil refineries. The key idea in MPC is to forecast the system behavior over a certain time horizon and then to optimize this forecast obtaining a sequence of control inputs by solving an optimal control problem (OCP). In order to realize a feedback loop and deal with unforeseen disturbances and model-plant mismatch, only the first control input is applied to the system, after which measurements are made and a new but similar optimal control problem is solved. The basic concepts of the MPC scheme, in this case used to track a reference setpoint, are illustrated in Figure 1.1. MPC has proven to be successful in a variety of environments, due to its inherent capability to include complex objectives and constraints in its problem formulation. This gives MPC an enormous edge over classical control methods, such as PIDs, which only consider the current system state and need heuristics and fine-tuning to even work in the presence of constraints.

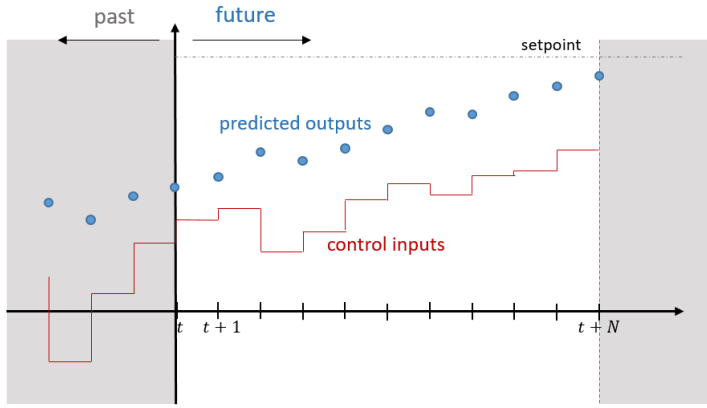



**Figure 1.1:** *Basic MPC scheme.*

The main disadvantage of MPC is the relatively high computational effort, since during every sample time an optimal control problem needs to be solved. Depending on the time horizon, the nonlinearity of the system dynamics and the complexity of the constraints, this might prove to be a challenge. It is in fact an especially demanding task when applying MPC to fast dynamical systems, since the sampling time then needs to be small for a satisfactory controller performance. Furthermore, this optimization is typically carried out on resource-constrained hardware. As such, solving the OCP in a fast and reliable manner is the topic of a substantial body of research, see for instance [131, §8].

One avenue of research investigates the construction of the OCP, in particular regarding the formulation of objective and constraints. For instance, there are two main ways to incorporate the (discrete time) system dynamics, that is the relation between successive states and inputs, into the OCP. First, in the multiple shooting formulation, an equality constraint specifying the relation between the current state and input to the next state is added for every time sample in the horizon to enforce the system dynamics explicitly. Second, these relations can be used to eliminate each state as a function of all the previous inputs and the initial state of the system, resulting in the single shooting formulation. As such, no equality constraints are necessary to enforce the system dynamics, but, certainly for a nonlinear system, the expressions for the objective and remaining constraints become typically much more intricate as a result of this recursive elimination.

In MPC, the objective designating what one wants to optimize is most often composed of a sum of stage costs, each penalizing the state and input at the considered sampling interval or stage. For example, in tracking MPC, the stage cost is typically a quadratic form in the difference between the state/input and the reference state/input. In economic MPC, the stage cost is some more general function that measures the economic progress of the process. Another possibility is for the objective not to be a function of the states and inputs but of a different quantity. Such is the case, for instance, in time-optimal MPC, where the horizon time is considered as the objective, and an equality constraint is listed to specify the required terminal state. In this thesis, usually tracking MPC is considered.

A common type of constraints that appear in an OCP are simple bounds on the states and inputs. Bounds on the states represent process requirements, such as for example demanding the temperature of a reactor to be below a certain threshold. Bounds on the inputs are usually a result of actuator limitations. Other constraints are typically problem-specific and may complicate the OCP significantly. In autonomous navigation for example, the states are required to be outside certain sets which represent obstacles in the environment. The formulation of such set exclusion (or obstacle avoidance) constraints is not straightforward, except in certain special cases. For instance, a circular obstacle can be encoded as one inequality constraint, and convex obstacles can be dealt with using the separating hyperplane theorem. However, the inclusion of general obstacle shapes in optimal control problems still represents an open challenge. The first research objective of this thesis is to provide a methodology for this exact purpose.

The second avenue of research on the topic of numerical optimal control looks more directly into solving OCPs by exploring the application of both existing and new optimization algorithms. Since sequential optimal control problems are very similar, a crucial feature of considered algorithms is that they allow for a meaningful initialization of the decision variables and possibly of some parameters, also known as warm starting or hot starting. This specification excludes interior-point methods, an otherwise popular class of optimization algorithms. Instead, typically sequential quadratic programming (SQP) and recently also first-order methods are used to solve OCPs. Among the latter ones, proximal algorithms have witnessed a recent increase in popularity, given their capability of dealing straightforwardly with certain types of constraints, such as simple bounds on the decision variables. Nevertheless, they lack a systematic way of dealing directly with general constraints, and require therefore additional methodologies for this purpose.

In certain cases, the OCP has a distinct structure. In linear MPC, for example, each optimal control problem is a quadratic program (QP), that is an optimization problem with a quadratic objective function and linear constraints. Given the prevalence of QPs also in other fields such as machine learning, portfolio optimization and as subproblems for sequential quadratic programming, solving QPs has been a key challenge in optimization literature. Typical state-of-the-art solvers rely either on interior-point methods, active-set methods or first-order methods. To the best of our knowledge, the application of another powerful technique for constrained optimization, the (proximal) augmented Lagrangian method, to QPs has hitherto received little attention in comparison. The second research objective of this thesis is to work out a full-fledged QP solver based on the proximal augmented Lagrangian method.

## 1.1 Contributions and structure of the thesis

As mentioned in the introduction, the contribution of this thesis is twofold, and the text is correspondingly divided in two parts. Part I, spanning Chapters 2 to 5, considers the application of MPC to autonomous navigation in the presence of general obstacle shapes. A methodology involving the quadratic penalty method is worked out to deal with the resulting mathematical programs with set exclusion constraints (MPSEC). The division in chapters is as follows:

**Chapter 2 - Autonomous Navigation** This chapter reviews the literature on autonomous navigation. After a broad overview, specific attention is placed on the inclusion of obstacle avoidance constraints in OCPs. In addition, it presents the mathematical formulation of general obstacle shapes considered in this thesis. Finally, it presents the OCP that is considered in this part of the thesis. This OCP is inherently nonconvex due to the obstacle avoidance constraints, and so poses a challenging problem to optimization algorithms.

**Chapter 3 - Penalty Method for Mathematical Programs with Set Exclusion Constraints** This chapter outlines the quadratic penalty method that is used to deal with general obstacle avoidance constraints. This method is observed to exhibit favorable properties for avoiding local minima behind an obstacle, since the first iteration with a low penalty results in a path that connects the starting and destination point without caring (too much) about the obstacles. Then, as the penalty parameters are increased, this path morphs towards the feasible region, resembling a homotopy. For every value of the penalty parameters, the resulting subproblem consists of a differentiable objective and simple bounds on the constraints. Therefore, these subproblems can be efficiently solved by a suitable proximal algorithm, such as the proximal averaged Newton-type method for optimal control (PANOC) [143], developed at KU Leuven. Furthermore, on top of the basic optimization algorithm, several heuristics are implemented to robustify the proposed approach, which is important since the considered problem is inherently nonconvex and no method is guaranteed to avoid all local minima.

**Chapter 4 - Penalty Method for MPSEC: Convergence results** This chapter presents the convergence results that can be obtained for the proposed optimization strategy. These results are nontrivial, because obtaining stationarity conditions for the original problem is problematic, since the obstacle avoidance constraints are inherently nondifferentiable nor can the normal cone to the feasible set be retrieved. As such, this chapter instead formulates an equivalent problem with vertical complementarity constraints, for which different stationarity concepts have been worked out in the literature. The limit points of the iterates generated by the proposed approach, when transformed to this equivalent problem, are then shown to satisfy different sets of stationarity conditions, depending on the assumptions made. Interestingly, the results presented in this chapter mirror some results obtained in the literature when applying a quadratic penalty method directly to a problem with complementarity constraints.

**Chapter 5 - Penalty Method for MPSEC: Numerical results** This chapter presents numerical simulations which demonstrate the applicability of the proposed approach. The resulting MPC controller is shown to be capable of steering two nonlinear vehicle models, a bicycle and a trailer, through a myriad of obstacle configurations. Moreover, the approach is compared to state-of-the-art interior-point and sequential quadratic programming methods applied to both the original problem and the one with complementarity constraints. The results show that our approach is faster, obtains lower objective values on average, and fails less often to solve the optimization problem than the other solvers.

Part II, spanning Chapters 6 to 9, considers the development of QPALM, a quadratic programming solver based on the proximal augmented Lagrangian method (P-ALM). The inner subproblems are solved using a tailored and highly efficient iterative strategy based on semismooth Newton directions, which rely on factorization update routines, and optimal step sizes. The division in chapters is as follows:

**Chapter 6 - Quadratic Programming** This chapter presents an overview of the literature on convex quadratic programming and some of its application domains. As soon as inequality constraints are introduced, the complexity of solving the QP is increased markedly. State-of-the-art solvers typically rely on active-set, interior-point or recently also first-order (proximal) methods. This chapter discusses the main ideas behind these methods and their advantages and disadvantages. In addition, a brief summary is given of existing methods for nonconvex quadratic programming.

**Chapter 7 - Proximal Augmented Lagrangian Method** This chapter presents the application of a different optimization strategy, the proximal augmented Lagrangian method, to quadratic programs. It shows the equivalence between this method and the proximal point algorithm. If the QP is convex, then convergence of P-ALM can straightforwardly be inferred from convergence results for inexact proximal point mappings applied to maximally monotone operators. If the QP is nonconvex, however, convergence is not shown as easily. This chapter extends the convergence results for inexact proximal point iterations, albeit to the special case of a quadratic program. Moreover, it shows the minimal modifications that are required to make P-ALM applicable to nonconvex QPs.

**Chapter 8 - The QPALM Algorithm** This chapter presents the key components of the open-source C implementation of QPALM. The inner minimization strategy consists of semismooth Newton directions and optimal step sizes. To realize an efficient algorithm, tailored linear algebra is considered, in particular for factorization update routines and for finding the minimum eigenvalue of a symmetric matrix. Parameter selection and update rules are often essential to the performance of an algorithm. The same holds for QPALM, and hence this aspect is investigated thoroughly. Finally, some additional aspects that make QPALM into a full-fledged solver, such as infeasibility detection and preconditioning of the problem data, are also discussed in this chapter.

**Chapter 9 - QPALM: Numerical results** This chapter presents several numerical benchmarks in which QPALM is compared with state-of-the-art QP solvers. Since quadratic programming is a popular research field, some collections of QPs have been compiled, amongst which the Maros-Mezaros set [108] is the most well known. This set contains 138 convex QPs obtained from various application domains, many of which are very large-scale and ill conditioned. Another collection is the Cutest [75], which contains also nonconvex QPs. Furthermore, some structured quadratic programs from specific application domains, such as linear MPC and portfolio optimization, are used as benchmark problems. The results show that QPALM strikes a unique balance between robustness against ill conditioning and efficiency, being able to solve almost all difficult QPs (within a time limit) while still being very fast at solving easy QPs. Moreover, it allows for warm starting unlike competitive interior-point methods, making it a preferable choice for certain applications such as MPC.

## 1.2 Preliminary material

### 1.2.1 Numbers

The set of natural numbers is denoted by $\mathbb{N}$, and we adopt the convention that $0 \in \mathbb{N}$. The set of real numbers is denoted by $\mathbb{R}$ and the set of extended-real numbers by $\overline{\mathbb{R}} \coloneqq \mathbb{R} \cup \{\infty\}$.

Given $a, b \in \overline{\mathbb{R}}$ we indicate with $(a, b) \coloneqq \{x \in \mathbb{R} \mid a < x < b\}$ and $[a, b] \coloneqq \{x \in \mathbb{R} \mid a \leq x \leq b\}$, respectively, the open and closed (possibly extended-real) intervals having $a$ and $b$ as endpoints. The half-open intervals $(a, b]$ and $[a, b)$ are defined analogously. Occasionally, $(a, b, \cdots)$ may also indicate a collection of numbers (or vectors), but in this case the context will always be explicit enough to avoid confusion. The set of positive real numbers is indicated as $\mathbb{R}_+ \coloneqq [0, \infty)$, and that of strictly positive real numbers as $\mathbb{R}_{++} \coloneqq (0, \infty)$.

The positive and negative parts of $r \in \mathbb{R}$ are defined as $[r]_+ \coloneqq \max\{0, r\}$ and $[r]_- \coloneqq \max\{0, -r\}$, respectively. Notice that $[r]_+$ and $[r]_-$ are positive numbers such that $r = [r]_+ - [r]_-$.

### 1.2.2 Vectors and matrices

We use $x \in \mathbb{R}^n$ to denote the primal variables of an optimization problem, and $y \in \mathbb{R}^m$ to denote the Lagrange multipliers, also known as the dual variables (in convex optimization). To avoid confusion in the MPC context, where $x$ usually denotes the system states, we instead use $q$ for this purpose. The common notation for system inputs $u$ is maintained in this text.

For a vector $x \in \mathbb{R}^n$, let $x_i$ denote its $i$-th element. In an MPC setting, the previous notation will be overloaded but only with the subscript $k$. As such, $q_k$ and $u_k$ indicate the states and inputs at the $k$-th time interval. The number of time intervals over which the optimization occurs is known as the control horizon, and is denoted as $N$.

The element of a matrix $A \in \mathbb{R}^{m \times n}$ in the $i$-th row and $j$-th column is denoted as $A_{ij} \in \mathbb{R}$. For an index $i \in [1, m]$, let $A_{i\cdot}$ denote the $i$-th row of $A$. Similarly, for a set of indices $\mathcal{I} \subseteq [1, m]$, let $A_{\mathcal{I}\cdot} \in \mathbb{R}^{|\mathcal{I}| \times n}$ be the submatrix comprised of all the rows $i \in \mathcal{I}$ of $A$. In the context of constraints within optimization problems, the notation $A_{\mathcal{I}}$ (without a dot) may also be used, since it is clear that we intend to specify a constraint for each row. Analogously to the row notation, for $j \in [1, n]$, and $\mathcal{J} \subseteq [1, n]$, let $A_{\cdot j}$ denote the $j$-th column of $A$, and $A_{\cdot\mathcal{J}} \in \mathbb{R}^{m \times |\mathcal{J}|}$ the submatrix comprised of the columns $j \in \mathcal{J}$ of $A$. Combined, let $A_{\mathcal{I}\mathcal{J}} \in \mathbb{R}^{|\mathcal{I}| \times |\mathcal{J}|}$ denote the submatrix comprised of all the rows $i \in \mathcal{I}$ and all the columns $j \in \mathcal{J}$ of $A$. Finally, let us denote the matrix

$A^{\mathcal{I}\cdot} \in \mathbb{R}^{m\times n}$ as the matrix with the corresponding rows from $A$ and 0 elsewhere, i.e.

$$A^{\mathcal{I}}_{i\cdot} = \begin{cases} A_{i\cdot} & \text{if } i \in \mathcal{I}, \\ 0 & \text{otherwise,} \end{cases}$$

and similarly $A^{\mathcal{I}\mathcal{J}} \in \mathbb{R}^{m\times n}$ the matrix with elements

$$A^{\mathcal{I}\mathcal{J}}_{ij} = \begin{cases} A_{ij} & \text{if } i \in \mathcal{I} \text{ and } j \in \mathcal{J}, \\ 0 & \text{otherwise.} \end{cases}$$

The $n \times n$ identity matrix is denoted as $\mathrm{I_n}$. The vector of size $n$ with each element equal to one is denoted by $\mathbf{1}_n$. Whenever $n$ is clear from context we simply write I and $\mathbf{1}$. For a vector $v \in \mathbb{R}^n$, let $\operatorname{diag}(v)$ denote the diagonal matrix with the elements of $v$ on the diagonal, that is for $V = \operatorname{diag}(v)$, $V_{ii} = v_i$, $i = 1, \ldots, n$, and $V_{ij} = 0, \forall i \neq j$.

With $\mathrm{Sym}(\mathbb{R}^n)$, $\mathrm{Sym}_+(\mathbb{R}^n)$, and $\mathrm{Sym}_{++}(\mathbb{R}^n)$, we denote respectively the set of symmetric, symmetric positive semidefinite, and symmetric positive definite matrices in $\mathbb{R}^{n\times n}$.

The minimum and maximum eigenvalues of $Q \in \mathrm{Sym}(\mathbb{R}^n)$ are denoted as $\lambda_{\min}(Q)$ and $\lambda_{\max}(Q)$, respectively. For $Q, R \in \mathrm{Sym}(\mathbb{R}^n)$ we write $Q \succeq R$ to indicate that $Q - R \in \mathrm{Sym}_+(\mathbb{R}^n)$, and similarly $Q \succ R$ indicates that $Q - R \in \mathrm{Sym}_{++}(\mathbb{R}^n)$. Any matrix $Q \in \mathrm{Sym}_+(\mathbb{R}^n)$ induces the semi-norm $\|\cdot\|_Q$ on $\mathbb{R}^n$, where $\|x\|_Q^2 \coloneqq x^\top Q x$; in case $Q = \mathrm{I}$, that is, for the Euclidean norm, we omit the subscript and simply write $\|\cdot\|$. For $p \in [1, \infty]$, the $\ell^p$ norm on $\mathbb{R}^n$ is denoted by $\|\cdot\|_p$, where

$$\|x\|_\infty \coloneqq \max\{|x_i| \mid i = 1 \ldots n\}, \quad \text{and} \quad \|x\|_p \coloneqq \left(\textstyle\sum_{i=1}^n |x_i|^p\right)^{1/p}$$

for $p \in [1, \infty)$.

The transpose of a matrix $A \in \mathbb{R}^{m\times n}$ is denoted by $A^\top$ and the number of its nonzero elements by $|A|$.

### 1.2.3 Statistics

The notation $\mathcal{N}(\mu, \sigma^2)$ represents the normal distribution with mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \geq 0$. As such, the standard normal distribution is given as $\mathcal{N}(0, 1)$. Furthermore, let $a \sim \mathcal{N}(\mu, \sigma^2)$ denote that the number $a \in \mathbb{R}$ is randomly generated from this distribution, or that it is a random variable which follows such a distribution. Similarly, for vectors $v, \mu \in \mathbb{R}^n$ and a matrix $\Sigma \in \mathrm{Sym}_+(\mathbb{R}^n)$, the notation $v \sim \mathcal{N}(\mu, \Sigma)$ denotes that the vector $v$ is drawn from or follows the multivariate normal distribution with mean $\mu$ and covariance matrix $\Sigma$.

The notation $\mathcal{U}(a, b)$ denotes the uniform distribution on the interval $[a, b]$.

For a random variable $r$, let $\mathrm{E}(r)$ denote the expected value of $r$, and let $\mathrm{Var}(r)$ denote the variance of $r$.

### 1.2.4 Sequences

The notation $\{a^k\}_{k\in K}$ represents a sequence indexed by elements of the set $K$, and given a set $E$ we write $\{a^k\}_{k\in K} \subset E$ to indicate that $a^k \in E$ for all indices $k \in K$. We say that $\{a^k\}_{k\in K} \subset \mathbb{R}^n$ is SUMMABLE if $\sum_{k\in K} \|a^k\|$ is finite, and SQUARE-SUMMABLE if $\{\|a^k\|^2\}_{k\in K}$ is summable.

We say that the sequence converges to a point $a \in \mathbb{R}^n$

- $Q$-LINEARLY if there exists $\rho \in [0, 1)$ such that $\|a^{k+1} - a\| \leq \rho\|a^k - a\|$ for every $k$;
- $R$-LINEARLY if there exists a sequence $\{\varepsilon_k\}_{k\in\mathbb{N}}$ which is $Q$-linearly convergent to 0 and it holds that $\|a^k - a\| \leq \varepsilon_k$;
- SUPERLINEARLY if either $a^k = a$ for some $k \in \mathbb{N}$, or $\|a^{k+1}-a\|/\|a^k-a\| \to 0$ as $k \to \infty$.

### 1.2.5 Sets

The CLOSURE of a set $O \subseteq \mathbb{R}^n$ is denoted as $\overline{O}$, its INTERIOR as $\operatorname{int} O$ and its BOUNDARY as $\partial O := \overline{O} \setminus \operatorname{int} O$.

A set $C$ is CONVEX if it contains the line segment between any two points in the set, that is, if $\forall x, y \in C, \theta \in [0, 1] : \theta x + (1 - \theta)y \in C$.

The cardinality of a finite set $\mathcal{J}$, i.e. the amount of (different) elements, is denoted as $|\mathcal{J}|$. It is always clear from the context whether the number of nonzeros of a matrix or the cardinality of a set of numbers is intended.

### 1.2.6 Functions and mappings

Given a function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$, its EPIGRAPH is the set

$$\operatorname{epi} h := \{(x, \alpha) \in \mathbb{R}^n \times \mathbb{R} \mid h(x) \leq \alpha\},$$

while its DOMAIN is

$$\operatorname{dom} h := \{x \in \mathbb{R}^n \mid h(x) < \infty\},$$

and for $\alpha \in \mathbb{R}$ its $\alpha$-LEVEL SET is

$$\operatorname{lev}_{\leq\alpha} h := \{x \in \mathbb{R}^n \mid h(x) \leq \alpha\}.$$

Function $h$ is said to be LOWER SEMICONTINUOUS (LSC) if $\operatorname{epi} h$ is a closed set in $\mathbb{R}^{n+1}$ ($h$ is also said to be CLOSED); equivalently, $h$ is lsc iff for all $\bar{x} \in \mathbb{R}^n$ it holds that

$$h(\bar{x}) \leq \liminf_{x \to \bar{x}} h(x).$$

All level sets of an lsc function are closed. We say that $h$ is PROPER if $\operatorname{dom} h \neq \emptyset$, and that it is LEVEL BOUNDED if for all $\alpha \in \mathbb{R}$ the level set $\operatorname{lev}_{\leq \alpha} h$ is a bounded subset of $\mathbb{R}^n$.

Function $h$ is CONVEX if for any $x, y \in \mathbb{R}^n$ and for $\theta \in [0,1]$ it holds that $h(\theta x + (1-\theta y)) \leq \theta h(x) + (1-\theta)h(y)$. It is STRONGLY CONVEX if $\exists \mu \in \mathbb{R}_{++}$ such that $h - \frac{\mu}{2}\|\cdot\|^2$ is convex, and similarly $h$ is HYPOCONVEX if $\exists \sigma \in \mathbb{R}_{++}$ such that $h + \frac{\sigma}{2}\|\cdot\|^2$ is convex.

The FENCHEL CONJUGATE of a proper closed convex function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$ is the convex function $h^* : \mathbb{R}^n \to \overline{\mathbb{R}}$ defined as

$$h^*(y) = \sup_x x^\top y - h(x). \tag{1.1}$$

The class of functions $h : \mathbb{R}^n \to \mathbb{R}$ that are $k$ times continuously differentiable is denoted as $C^k(\mathbb{R}^n)$. We write $h \in C^{1,1}(\mathbb{R}^n)$ to indicate that $h \in C^1(\mathbb{R}^n)$ and that $\nabla h$ is (globally) Lipschitz continuous with modulus $L_h$, i.e. $\forall x, y \in \mathbb{R}^n : \|\nabla h(x) - \nabla h(y)\| \leq L_h\|x - y\|$. To simplify the terminology, we will say that such an $h$ is $L_h$-SMOOTH. For an $L_h$-smooth $h$ the following quadratic upper bound exists, see e.g. [17, Prop. A.24],

$$h(y) \leq h(x) + \nabla h(x)^\top (y - x) + \frac{L_h}{2}\|y - x\|^2 \quad \forall x, y \in \mathbb{R}^n. \tag{1.2}$$

A function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$ is COERCIVE if

$$\lim_{\|x\| \to \infty} h(x) = \infty.$$

The IDENTITY mapping $\mathrm{id} : \mathbb{R}^n \to \mathbb{R}^n$ maps its argument to itself, that is $\mathrm{id}(x) = x$.

The INDICATOR FUNCTION of a set $S \subseteq \mathbb{R}^n$ is the function $\delta_S : \mathbb{R}^n \to \overline{\mathbb{R}}$ defined as

$$\delta_S(x) = \begin{cases} 0 & \text{if } x \in S, \\ \infty & \text{otherwise.} \end{cases} \tag{1.3}$$

If $S$ is nonempty and closed, then $\delta_S$ is proper and lsc.

The PROJECTION onto a nonempty closed convex set $S \subseteq \mathbb{R}^n$ is defined by $\Pi_S(x) = \arg\min_{z \in S} \|z - x\|$ or, equivalently, the unique point $z \in S$ satisfying the inclusion

$$x - z \in \mathcal{N}_S(z), \tag{1.4}$$

where

$$\mathcal{N}_S(z) := \left\{ v \in \mathbb{R}^n \mid v^\top (z - z') \leq 0 \ \forall z' \in S \right\} \tag{1.5}$$

is the normal cone of the set $S$ at $z$. With $\operatorname{dist}(x, S) := \inf_{z\in\mathbb{R}^n} \|z - x\|$ we indicate the DISTANCE of $x$ from $S$.

Given a mapping $F : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$, we say that a point $x$ is FIXED (for $F$) if $x \in F(x)$, while $x$ is a ZERO (of $F$) if $0 \in F(x)$. The FIXED SET (i.e., the set of fixed points) and the ZERO SET (i.e., the set of zeros) of $F$ are respectively denoted by

$$\operatorname{fix} F := \{x \in \mathbb{R}^n \mid x \in F(x)\},$$

and

$$\operatorname{zer} F := \{x \in \mathbb{R}^n \mid 0 \in F(x)\}.$$

A point-to-set mapping $\mathcal{M} : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is MONOTONE if $(x - x')^\top(\xi - \xi') \geq 0$ for all $x, x' \in \mathbb{R}^n$, $\xi \in \mathcal{M}(x)$ and $\xi' \in \mathcal{M}(x')$. It is MAXIMALLY MONOTONE if, additionally, there exists no monotone operator $\mathcal{M}' \neq \mathcal{M}$ such that $\mathcal{M}(x) \subseteq \mathcal{M}'(x)$ for all $x \in \mathbb{R}^n$. The RESOLVENT of a maximally monotone operator $\mathcal{M}$ is the single-valued (in fact, Lipschitz-continuous) mapping $(\mathrm{id} + \mathcal{M})^{-1}$, where $(\mathrm{id} + \mathcal{M})^{-1}(x)$ is the unique point $\bar{x} \in \mathbb{R}^n$ such that $x - \bar{x} \in \mathcal{M}(\bar{x})$. For a linear mapping $\Sigma$, $\Sigma\mathcal{M}$ is the operator defined as $\Sigma M(x) := \{\Sigma y \mid y \in \mathcal{M}(x)\}$.

Given a proper and lsc function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$, we denote by $\hat{\partial} h : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ the REGULAR SUBDIFFERENTIAL of $h$, where

$$v \in \hat{\partial} h(x) \quad \Leftrightarrow \quad \liminf_{\substack{x'\to x \\ x'\neq x}} \frac{h(x') - h(x) - v^\top(x' - x)}{\|x' - x\|} \geq 0. \tag{1.6}$$

The (limiting) SUBDIFFERENTIAL of $h$ is $\partial h : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$, where $v \in \partial h(x)$ iff $x \in \operatorname{dom} h$ and there exists a sequence $\{x^k, v^k\}_{k\in\mathbb{N}} \subseteq \operatorname{gph} \hat{\partial} h$ such that

$$\lim_{k\to\infty} (x^k, h(x^k), v^k) = (x, h(x), v).$$

Note that if $h$ is proper lsc and convex, then

$$\hat{\partial} h(x) = \partial h(x) = \left\{v \in \mathbb{R}^n \mid h(x') \geq h(x) + v^\top(x' - x)\ \forall x' \in \mathbb{R}^n\right\},$$

**Definition 1.1** (Proximal mapping)**.** *The PROXIMAL MAPPING of $h : \mathbb{R}^n \to \overline{\mathbb{R}}$ with parameter $\gamma > 0$ is the set-valued map $\operatorname{prox}_{\gamma h} : \mathbb{R}^n \rightrightarrows \operatorname{dom} h$ defined as*

$$\operatorname{prox}_{\gamma h}(x) := \operatorname*{arg\,min}_{w\in\mathbb{R}^n} \left\{h(w) + \tfrac{1}{2\gamma}\|w - x\|^2\right\}. \tag{1.7}$$

We say that a function $h$ is PROX-BOUNDED if $h + \frac{1}{2\gamma}\|\cdot\|^2$ is lower bounded for some $\gamma > 0$. The value function of the minimization problem defining the proximal mapping is the MOREAU ENVELOPE with parameter $\gamma$, denoted $h^\gamma : \mathbb{R}^n \to \mathbb{R}$, namely

$$h^\gamma(x) := \inf_{w\in\mathbb{R}^n} \left\{h(w) + \tfrac{1}{2\gamma}\|w - x\|^2\right\}. \tag{1.8}$$

We can extend these definitions in case $\gamma$ is replaced by a matrix $\Sigma \in \mathrm{Sym}_{++}(\mathbb{R}^n)$. The proximal mapping of $h$ with (matrix) step size $\Sigma$ is the set-valued mapping $\mathrm{prox}_h^{\Sigma} : \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ given by

$$\mathrm{prox}_h^{\Sigma}(x) := \operatorname*{arg\,min}_{w \in \mathbb{R}^n} \left\{ h(w) + \tfrac{1}{2}\|w - x\|^2_{\Sigma^{-1}} \right\}, \tag{1.9}$$

and the corresponding Moreau envelope is $h^{\Sigma} : \mathbb{R}^n \to \mathbb{R}$ defined as

$$h^{\Sigma}(x) := \min_{w \in \mathbb{R}^n} \left\{ h(w) + \tfrac{1}{2}\|w - x\|^2_{\Sigma^{-1}} \right\}.$$

It follows from the above definition that $\bar{x} \in \mathrm{prox}_h^{\Sigma}(x)$ iff

$$h^{\Sigma}(x) = h(\bar{x}) + \tfrac{1}{2}\|x - \bar{x}\|^2_{\Sigma^{-1}} \leq h(x') + \tfrac{1}{2}\|x - x'\|^2_{\Sigma^{-1}} \quad \forall x' \in \mathbb{R}^n. \tag{1.10}$$

Moreover, for every $x \in \mathbb{R}^n$ it holds that

$$\Sigma^{-1}(x - \bar{x}) \in \hat{\partial} h(\bar{x}), \quad \text{where} \quad \bar{x} \in \mathrm{prox}_h^{\Sigma}(x). \tag{1.11}$$

# Part I

## Penalty Method for Autonomous Navigation

# Chapter 2

## Autonomous Navigation

This chapter discusses existing methods for the problem of navigating a vehicle through an obstructed environment, also known as autonomous navigation, motion planning or path planning. Typically, a distinction is made between coupled and decoupled approaches. Coupled approaches compute the optimal trajectory and the control inputs to steer the vehicle along this path simultaneously, whereas decoupled approaches first compute a geometric path and use a second component, such as an MPC controller, to track this path.

Section 2.1 gives a succinct examination of the main motivation for this research, the autonomous guided vehicle (AGV). Section 2.2 provides a brief overview of approaches that focus on the generation of a geometric path only. Section 2.3 focuses instead on optimization approaches that compute simultaneously the path and the required control inputs to track this path. The last subsection presents also the formulation used in this work to deal with general obstacle shapes.

## 2.1 Introduction

Automated guided vehicles have grown increasingly more popular in modern industry [156]. Originally invented in the 1950s to automate material flow, the first era of AGVs did not require a path planning component, but was instead guided over tracks. These tracks either generated a magnetic field or were of a special color relative to the floor so that the AGV could drive along them. This implementation may have been sufficient for very repetitive material transport, but modern applications typically require a more unfettered approach. For example, driverless cars, fruit-picking robots, lawn mowing robots, cleaning robots and AGVs in modern e-commerce warehouses all depend upon the autonomous generation of a collision-free trajectory that is dynamically updated to account for uncertainty in the vehicle model or the environment. As a consequence, there is a growing interest in more flexible motion planning techniques.

Key components of an AGV include sensors to obtain information from the environment, actuators to drive the vehicle and a logical unit that computes the trajectory. In this research, we abstract from the aspect of the sensors and instead consider the environment, the obstacle shapes and the position of the vehicle as known information. Regarding the actuators, we make the assumption that they apply the commanded input to the vehicle immediately, although in practice they typically rely on fast PIDs to track this input. Instead, the aspect that is of prime interest here is the path planning component. The following section looks at one type of approaches that deal with the generation of a feasible geometric path.

## 2.2 Decoupled approaches

Given the above developments, finding the shortest path, or at least a feasible path from a starting to a destination point, has been the subject of extensive research, and various methods exist, such as graph-search and potential field methods. The reader is referred to [95, 96] for some textbook overviews on these methods, of which some material is summarized here. As shown in §3, a simple graph-search method can be used to complement a coupled approach as a sort of fail-safe. The main disadvantage of decoupled approaches is that they do not provide the control inputs required to track the resulting path. As such, not only is a second component needed to compute these inputs, but also a modicum of optimality is sacrificed, since no control inputs might exist that can track the path exactly.

### 2.2.1 Graph-search methods

Graph-search methods try to obtain a global path and rely first and foremost on a discretization of the space. In case of a two dimensional space, this discretization often uses uniform squares, illustrated in Figure 2.1. When the midpoint of a square is inside the free space, it is counted as a node. Conversely, when it is inside an obstacle, the square is not added. Then, a graph of nodes is constructed by connecting all horizontally, vertically and diagonally neighboring nodes [95, §6]. This graph is also known as a road map. Similar ideas to decomposing the space include exact cell decomposition, which tries to fit trapezoidal shapes on the free space [95, §5].

Some other methods to obtain the vertices of a road map exist. A relatively old idea is to construct the visibility graph, by connecting the vertices of (polygonal) obstacles [64]. As such, the obtained path will pass by the corners of obstacles, as shown on Figure 2.2, which might be infeasible in case of uncertainty or disturbances. A more recent idea emphasizing safe passage is the use of Voronoi diagrams [147], which try to obtain nodes with maximal clearance with respect to all the obstacles, as illustrated in Figure 2.3. Using a Voronoi diagram results in more distance between the vehicle and the obstacles, which might be useful in the presence of large uncertainty, but the path will likely be less optimal.

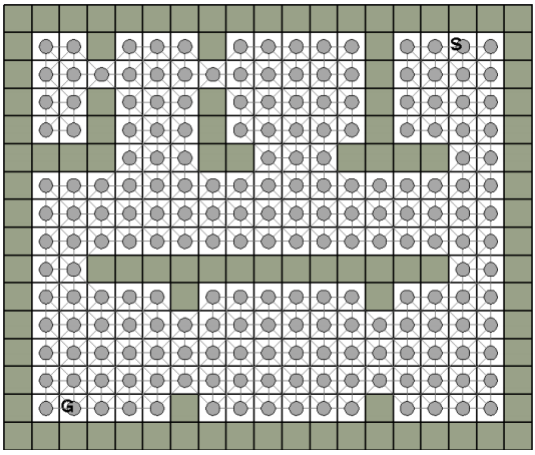



**Figure 2.1:** *Discretization of the space in squares [23]. The nodes (circles) indicating free space are connected to all neighboring nodes, with S the starting and G the goal node, while the colored squares represent obstacles.*

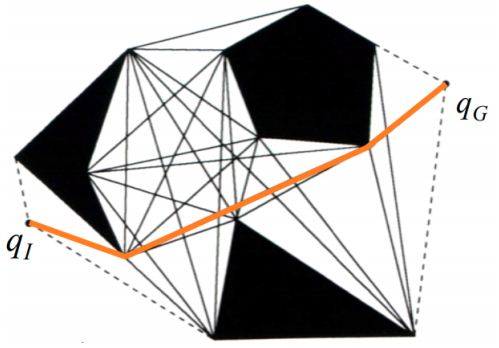



**Figure 2.2:** *Visibility graph with initial position $q_I$ and destination $q_G$ [95, §4.1].*

The above methods of discretizing the space belong to the class of combinatorial planners. A big disadvantage of these is that the number of nodes can grow exponentially as the space increases, or if the free space is multi-dimensional. Sampling-based planners instead use probabilistic methods to sample the free space randomly to try to connect the starting point and the destination. As a result, they might construct a road map much faster than combinatorial planners, but they are known to encounter problems in certain situations, such as narrow passages, and they also sacrifice any kind of optimality. Examples of sampling-based planners include probabilistic road maps and rapidly exploring random trees [96, §5].

Once the road map is constructed, a search algorithm is employed to find the shortest

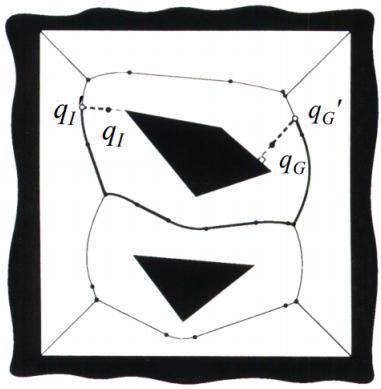



**Figure 2.3:** *Construction of the vertices of a Voronoi diagram [95, §4.2].*

path connecting the start and destination, typically the well-known Dijkstra's algorithm [45] or one of its variants. Dijkstra's algorithm keeps track of a priority queue of visited nodes along with their cost, which is the distance covered from the start to these nodes. Starting from the starting point as the only node in the queue, the algorithm iteratively visits all neighboring nodes of the first node in the queue and updates the minimum distances if necessary. This procedure is repeated until the destination is the first node in the queue, which means that one has found the shortest path.

A disadvantage of Dijkstra's algorithm is that it executes an uninformed search. In other words, no relation between the visited nodes and the destination is taken into account. As such, the algorithm may visit many nodes which lead away from the destination instead of towards it during the search. In many cases, additional information can be incorporated using heuristics to combat this issue. For example, the popular A* algorithm [81] computes the cost of the nodes as the sum of the distance covered from the start and the expected (Euclidean) distance to the destination. As a result, the algorithm tends to first visit nodes that decrease the distance towards the destination. For a heuristic to be admissible, the actual distance to the destination should never be overestimated. As such, in a grid-based space as in Figure 2.1, the Euclidean distance is indeed an admissible heuristic cost. The resulting path found by A* for Figure 2.1 is drawn in Figure 2.4. Another variant, D* [145], which starts at the destination and works its way back to the start in a similar manner, has been invented to deal with partially unknown environments. When a path is discovered to be blocked, only few cells need to be recomputed in D*, whereas A* would have to replan from the blocked cell.

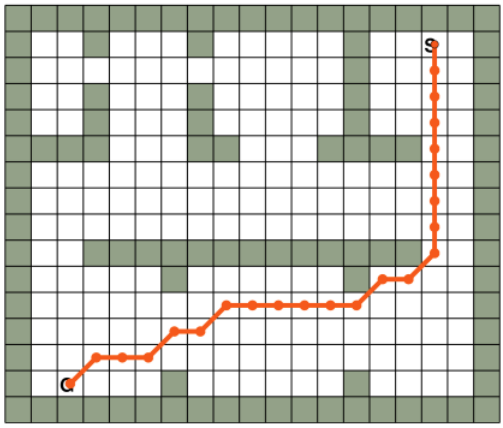



**Figure 2.4:** *Resulting path obtained by applying A* to Figure 2.1 [23].*

### 2.2.2 Potential field methods

Potential field methods [62, 112] consider the problem from a different angle. They associate an attractive potential field to the goal and repulsive potential fields to the obstacles. As such, the potential function at a point in space $q$ is given by $U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q)$, which is illustrated for a 2D space with two rectangular obstacles in Figure 2.5. The exact formulation of the potential functions is a subject of much research, and is outside the scope of this short overview.

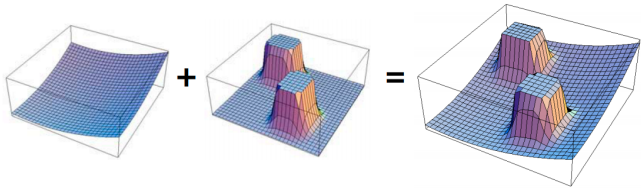



**Figure 2.5:** *Potential field as a result of attractive and repulsive potentials [23].*

The force working on the vehicle as a result of these potentials is given by $F(q) = -\nabla U(q)$. At every time instant, this force is used as the reference for a local feedback control law. Therefore, the required computation time is small, and the potential field method can be executed at high sampling rates. In essence, it is as though the potential is minimized iteratively using a gradient descent method. However, local minima tend to occur in the potential which do not correspond to the goal position, due to the repulsive potentials of the obstacles. In fact, it is well known that the

potential field method suffers from such minima, as illustrated in Figure 2.6, and substantial research efforts have been directed at tackling this crucial issue.

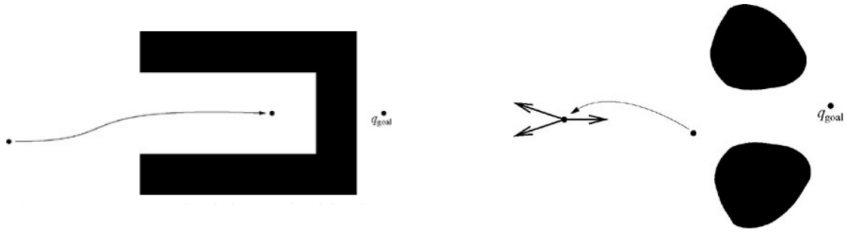



**Figure 2.6:** *Two examples of how local optima can hinder the applicability of the potential field method [28, §4].*

Both graph search methods and potential field methods compute a geometric path (or direction) to move along, without providing the required control inputs. It is worth mentioning that in some problems it is possible to take into account the kinematics in a decoupled approach, see for instance [25, 165]. However, to this end a case-dependent smoothing functionality to connect waypoints into a feasible path is required. Another approach with a similar spirit is that of motion primitives [58, 78, 54], which are essentially (short) maneuvers the vehicle can execute. A graph search method may then be used to search the space of maneuvers. However, the time complexity of this method scales exponentially in the number of maneuvers. As a result only very few maneuvers are typically considered. Given these difficulties and the recent advances in computing technology and optimization algorithms, the interest in coupled approaches, which compute both a path and the control inputs simultaneously, has been growing.

## 2.3 Coupled approaches

In a coupled approach for motion planning systems, an optimal control problem needs to be solved at every time instant, accounting for geometrical constraints, system dynamics and limitations. The first subsection outlines the structure of such an OCP. The remaining subsections discuss techniques to include obstacle avoidance constraints in the OCP.

### 2.3.1 Optimal control problem

In the setting of an autonomous motion system in an obstructed environment, the optimal control problem will typically have the following form:

$$\underset{(q,u)\in\mathbb{R}^{(N+1)n_q+Nn_u}}{\textbf{minimize}} \quad \ell(q,u) \tag{2.1a}$$

$$\textbf{subject to} \quad q_{k+1} = \varphi_k(q_k, u_k), \quad k = 0, \dots, N-1, \tag{2.1b}$$

$$q_0 = \bar{q}, \tag{2.1c}$$

$$u_k \in \mathcal{U}_k, \quad q_k \in \mathcal{Q}_k, \quad k = 0, \dots, N-1, \tag{2.1d}$$

$$q_k \notin O_i, \qquad k = 1, \dots, N, \quad i = 1, \dots, N_O. \tag{2.1e}$$

Here, $u$ and $q$ represent the vector of control inputs and states of the system respectively, $u_k$ and $q_k$ the input and state at time instant $k$, $\bar{q}$ the initial state (or an estimate thereof) and $\varphi_k$ the discrete time system dynamics. $N$ denotes the control horizon, that is the number of inputs which are computed, and $n_u$ and $n_q$ the number of inputs and states the system has. The objective $\ell$ depends on the problem setting, as mentioned in §1. In our application of motion planning, $\ell$ is given as the sum of stage costs and a terminal cost, that is

$$\ell(q,u) = \ell_N(q_N) + \sum_{k=0}^{N-1} \ell_k(q_k, u_k), \tag{2.2}$$

where the stage cost

$$\ell_k(q_k, u_k) = (q_k - q_{\text{ref}})^\intercal Q_k (q_k - q_{\text{ref}}) + (u_k - u_{\text{ref}})^\intercal R_k (u_k - u_{\text{ref}}),$$

and the terminal cost

$$\ell_N(q_N) = (q_N - q_{\text{ref}})^\intercal Q_N (q_N - q_{\text{ref}}).$$

Here, $q_{\text{ref}}$ and $u_{\text{ref}}$ are the reference state and input, $Q_k, Q_N \in \mathrm{Sym}_+(\mathbb{R}^{n_q})$ and $R_k \in \mathrm{Sym}_{++}(\mathbb{R}^{n_u})$. With this choice of objective, we will try to minimize the distance from all states to the destination state, taking into account also a small penalty for the use of the inputs and therefore striking a balance between a time- and energy-optimal trajectory.

The constraints (2.1d) represent general input and state constraints, as is common in MPC [131]. Finally, (2.1e) denotes general obstacle avoidance constraints, specifying that the states should be outside of certain sets $O_i$. The number of obstacles is denoted by $N_O$, and these are considered static. There is no problem in extending the given formulation to dynamic obstacles, since (2.1e) would then be replaced by $q_k \notin O_{ik}$.

OCP (2.1) has both the inputs and the states as decision variables. This formulation is known as the multiple shooting formulation, or as the simultaneous approach to optimal control. One can condense this problem by eliminating the states through recursive substitution of (2.1b), which leaves only the inputs as decision variables and

the initial state as a parameter. Let for instance $q = \Phi(u) = (\Phi_0(u), \dots, \Phi_N(u))$ with $\Phi_0(u) = \bar{q}$ and

$$q_{k+1} = \Phi_{k+1}(u) = \varphi_k(\Phi_k(u), u_k).$$

Using this notation, (2.1) can alternatively be cast into the single shooting formulation, or the sequential approach to optimal control:

$$\begin{aligned}
\underset{u \in \mathbb{R}^{N n_u}}{\textbf{minimize}} \quad & \ell(u) && && \text{(2.3a)}\\
\textbf{subject to} \quad & u_k \in \mathcal{U}_k, && k = 0, \dots, N-1, && \text{(2.3b)}\\
& \Phi_k(u) \in \mathcal{Q}_k, && k = 0, \dots, N-1, && \text{(2.3c)}\\
& \Phi_k(u) \notin O_i, && k = 1, \dots, N, \quad i = 1, \dots, N_O. && \text{(2.3d)}
\end{aligned}$$

Here, the objective $\ell(u)$ is obtained by eliminating the states from (2.2), as

$$\ell(u) = \ell_N(\Phi_N(u)) + \sum_{k=0}^{N-1} \ell_k(\Phi_k(u), u_k). \tag{2.4}$$

The sequential approach yields a smaller optimization problem than the simultaneous approach, but may not always be preferable. When the system dynamics $\varphi_k$ are linear, for example, this approach typically destroys the sparsity structure of the underlying matrices. Furthermore, in nonlinear MPC, the recursive evaluation of a nonlinear function in $\Phi$ may compound the "severity" of the nonlinearity. Moreover, it is often not straightforward to translate the state constraints (2.3c) into input constraints. Therefore, as a rule of thumb, the sequential approach is preferable if there is no significant sparsity or the solver cannot exploit it, and when the system is stable, whereas the simultaneous approach is preferable for unstable nonlinear systems and for problems with (complicated) state constraints [131, §8.1.3].

In this work, the sequential approach will be used. To simplify the notation and subsequent analysis of (2.3), we will collect the input constraints (2.3b) as $u \in \mathcal{U} = \mathcal{U}_0 \times \mathcal{U}_1 \times \cdots \times \mathcal{U}_{N-1}$, leave out state constraints, and assume that the mappings $\Phi_k$ in (2.3d) can somehow be incorporated in the obstacle formulations. Furthermore, we will denote the decision variable as $x \in \mathbb{R}^n$ (and replace $\mathcal{U}$ by $\mathcal{X}$), as is common in optimization literature. This results in the following simplified version of the problem

$$\begin{aligned}
\underset{x \in \mathbb{R}^n}{\textbf{minimize}} \quad & \ell(x) && \text{(2.5a)}\\
\textbf{subject to} \quad & x \in \mathcal{X}, && \text{(2.5b)}
\end{aligned}$$

$$x \notin O_i, \quad i = 1, \ldots, N_O. \tag{2.5c}$$

What remains now is the meaningful inclusion of obstacle avoidance constraints (2.5c), which is not trivial for general sets. Only specific types of obstacles can be incorporated straightforwardly. Another approach is to formulate state constraints that describe (a limited part of) the free space, effectively removing the need for (2.5c). This method starts from a feasible geometric path, usually computed by a decoupled approach, and expands polygons (or other shapes) around it to create a feasible tunnel [6, 104, 160]. Obviously, a disadvantage of such an approach is that the considered feasible region might be very limited. The remainder of this section gives a brief summary of obstacle avoidance formulations within optimization problems, restricted to 2D spaces for simplicity.

### 2.3.2 Simple inequalities

For circular obstacles, an obstacle avoidance constraint can be written down specifying that the distance between the motion system and the center of the circle is larger than its radius. If, furthermore, the motion system is circular with radius $R$, and the obstacle has a radius of $R_O$ and a center point $p$, the constraint becomes

$$\|q_k - p\| \geq R + R_O.$$

This formulation was for instance used in [162], where the authors considered multiple circular motion systems with collision avoidance amongst each other. Of course, ellipsoidal obstacles can be considered in a similar manner, with a weighted norm on the distance computation.

Another example of simple inequality constraints are obstacles with only one boundary, such as a wall. These constraints might as well have been incorporated in the general state constraints of (2.1d). A slightly less trivial example, perhaps, is lane keeping of cars on highroads [155]. There, the obstacles can be translated as simple bounds on the (lateral) position error with respect to the central lane.

### 2.3.3 Distance functions

Another commonly used approach is to impose via inequality constraints that the distance to the (convex) obstacle sets should be positive or larger than a small positive margin [65]. In fact, circular obstacles as mentioned above are a special case of this method. The disadvantage of such an approach is that calculating the distance of a point to a general convex set requires the solution of a minimization problem to obtain the projection of that point on the set.

A similar idea is to impose that the gauge function of every obstacle set is bounded below by 1. The gauge function of a set evaluated at a point represents the smallest

scaling factor applied to the set such that the point is included in the set. This approach is especially useful when the polar set of each obstacle is easily representable, such as for polytopic obstacles, where this method translates such constraints into bilinear inequality constraints [123].

### 2.3.4 Separating hyperplane theorem

For convex motion systems and convex obstacles, the separating hyperplane theorem [20, §2.5.1] provides another way of encoding obstacle avoidance constraints. This theorem states that for two disjoint convex sets $C$ and $D$, there exist vectors $a \neq 0$ and $b$, such that $a^\top x \leq b, \forall x \in C$ and $a^\top x \geq b, \forall x \in D$. The hyperplane $\{x : a^\top x = b\}$ is called a separating hyperplane of $C$ and $D$, since these sets lie on opposing sides thereof, as illustrated in Figure 2.7.

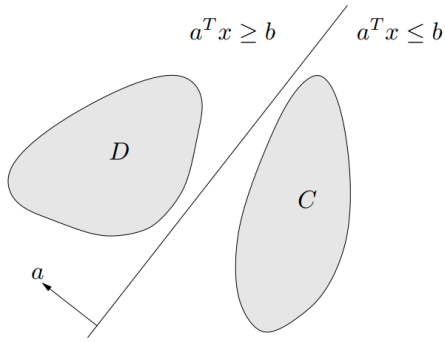



**Figure 2.7:** $\{x : a^\top x = b\}$ *is a separating hyperplane of the sets* $C$ *and* $D$ *[20].*

Supposing the obstacles and motion system are circular and/or polygonal, it is relatively straightforward to identify all the extreme points, either the circumference (given by a center point and radius) or the vertices of the polygon. Therefore, it suffices to impose the separating hyperplane inequalities for these extreme points of the obstacle and motion vehicle sets (as a function of $q_k$) in order to establish collision avoidance of the complete sets at all times. Collision avoidance using the separating hyperplane theorem for a rectangular motion system and possibly dynamic circular and rectangular obstacles is illustrated in [110].

### 2.3.5 General obstacle formulation

To deal with a more general framework for considering obstacles, we will use the formulation first introduced in [136]. In this work, an obstacle set $O_i$ is given by a set

of nonlinear inequalities:

$$O_i = \{x \in \mathbb{R}^n : h_{ij}(x) > 0, \quad j = 1, \dots, m_i\}. \tag{2.6}$$

Here, $h_{ij} : \mathbb{R}^n \to \mathbb{R}$ are continuously differentiable functions describing obstacles boundaries. The obstacle boundary is noted as $\partial O_i$ and the closure as $\overline{O}_i = O_i \cup \partial O_i$. In almost all cases, the boundary is given as the following set:

$$\partial O_i = \{x \in \mathbb{R}^n : h_i(x) \geq 0, \quad h_{ij}(x) = 0 \ \text{ for some } j\}. \tag{2.7}$$

**Remark 2.1.** In (2.7), instead of being equal, the boundary may only be a subset, for instance in the case where one of the boundary functions is artificially set to 0 outside the obstacle. For example, consider $O = \{x \in \mathbb{R} : h(x) > 0\}$, with $h(x) = \begin{cases} x^2 & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases}$, then $\partial O = \{0\}$ while the set in (2.7) would be $(-\infty, 0]$. In the remainder of this text it is assumed that (2.7) holds. Correctness of (2.7) can also be derived by assuming the Mangasarian-Fromovitz constraint qualification (MFCQ) on the active gradients $\nabla h_i(x)$ at boundary points, as shown in the following lemma. □

**Lemma 2.2.** *For an obstacle $O_i$ given in (2.6), let $\mathcal{A}_i(x) = \{j : h_{ij}(x) = 0\}$. If $\forall x : h_{ij}(x) \geq 0, j = 1, \dots, m_i$, there exists a vector $d_x$ such that $d_x^\top \nabla h_{ij}(x) > 0, \forall j \in \mathcal{A}_i(x)$, then the obstacle boundary is given by (2.7).*

*Proof.* Let $C = \{x \in \mathbb{R}^n : h_i(x) \geq 0, \quad h_{ij}(x) = 0 \ \text{ for some } j\}$ be the right-hand side of (2.7). Showing that $C = \partial O_i$ can be done by, for any $\bar{x} \in C$, demonstrating the existence of two sequences, $\{x_1^k\} \subset O_i$ and $\{x_2^k\} \subset O_i^c$, both converging to $\bar{x}$. It is straightforward to see that, given the continuity and differentiability of the functions $h_{ij}(x)$, the sequences $x_1^k = \bar{x} + \frac{1}{k} d_{\bar{x}}$ and $x_2^k = \bar{x} - \frac{1}{k} d_{\bar{x}}$, for $k$ large enough, satisfy exactly these properties. □

**Remark 2.3.** It is well known that the linear independence constraint qualification (LICQ) implies MFCQ, and so assuming LICQ, which we do in the theory of §4, is also sufficient to prove Lemma 2.2. □

To include an obstacle avoidance constraint of type (2.6) meaningfully in the OCP, we can replace (2.5c) by the following equality constraint:

$$\psi_i(x) := \prod_{j=1}^{m_i} [h_{ij}(x)]_+ = 0, \tag{2.8}$$

where the notation $[\,\cdot\,]_+ = \max(\,\cdot\,, 0)$. The *obstacle cost function* $\psi_i(x)$ is by definition strictly positive for $x \in O_i$, and exactly zero for $x \notin O_i$.

Note that each obstacle cost function $\psi_i$ is differentiable at points strictly inside and strictly outside the obstacle, but due to the presence of the $[\,\cdot\,]_+$ operators, it is not differentiable at points on the obstacle boundary. As shown in Section 3.1, a quadratic version $\psi_i^2$ is differentiable, but the constraint $\psi_i^2(x) = 0$ is similarly problematic

in that its gradient is zero for any feasible point, which violates every constraint qualification. How we overcome this theoretical challenge is the topic of §4. But first, the next chapter will discuss our strategy for solving the OCP with such obstacle avoidance constraints.

## 2.4 Summary

This chapter broadly covered the literature on autonomous navigation in an obstructed environment. The problem consists on the one hand of finding a feasible, and in some sense optimal, path that goes from start to destination while avoiding obstacles, and on the other hand of computing the control inputs to steer the vehicle along this path. Decoupled approaches, such as graph-search and potential field methods, execute this in two separate steps. As such, they are typically fast, but the vehicle actuation might not be compatible with the geometric path. Coupled approaches combine the two problems in one optimization program. It is not trivial, however, to incorporate obstacles in this formulation, except for some special cases. Finally, a more general framework was considered in which any obstacle could be included, as long as its boundary can be defined using smooth functions.

# Chapter 3

# Penalty Method for Mathematical Programs with Set Exclusion Constraints

This chapter considers a solution methodology for the OCP with general obstacle avoidance constraints, introduced in §2 as

$$\underset{x \in \mathcal{X}}{\textbf{minimize}} \quad \ell(x) \tag{3.1a}$$

$$\textbf{subject to} \quad \psi_i(x) = 0, \quad i = 1, \ldots, m. \tag{3.1b}$$

Here, $\ell \in C^{1,1}(\mathbb{R}^n)$, $\mathcal{X}$ is a nonempty, closed and convex set, and the obstacle cost function $\psi_i(x)$ is given by (2.8). Since the distinct feature of (3.1) is the set exclusion constraints, we denote this the mathematical program with set exclusion constraints (MPSEC). The key difficulty in solving this optimization problem is the nondifferentiable constraints (3.1b). It turns out, however, that a squared variant $\psi_i^2(x) = 0$ is differentiable, and as such amenable to a penalty formulation. Furthermore, given the nonconvex nature of the autonomous navigation problem, many local minima exist due to the presence of the obstacles. To overcome this issue, this chapter proposes some heuristics to supplement the penalty method in practice.

Section 3.1 presents the quadratic penalty method employed to deal with the obstacle avoidance constraints (3.1b). This method requires the solution of a series of optimization problems, which are unconstrained aside from the abstract set constraint $x \in \mathcal{X}$. Subproblems of this type can efficiently be solved using a projection based method. Section 3.2 outlines one suitable method for this purpose which exhibits high efficiency, namely the proximal averaged Newton-type method for optimal control (PANOC) [143]. Given the nonconvex nature of the autonomous navigation problem, many local minima exist due to the presence of the obstacles. Finally, Section 3.3 presents some heuristics developed in this work to complement the MPC algorithm in practice, such that it can avoid local minima and steer the vehicle safely.

The material in this chapter is based on the publications [82, 83].

## 3.1 Quadratic Penalty Method

As mentioned in Section 2.3.5, the obstacle cost function $\psi_i(x)$ is nonsmooth due to the $[\,\cdot\,]_+$ operators. The same, however, does not hold for $\psi_i^2(x)$, since its gradient is given by

$$\begin{aligned}
\nabla\psi_i^2(x) &= \nabla\left(\prod_{j=1}^{m_i}[h_{ij}(x)]_+^2\right) \\
&= 2\sum_{j=1}^{m_i}\nabla h_{ij}(x)[h_{ij}(x)]_+\prod_{k\neq j}[h_{ik}(x)]_+^2 \\
&= 2\sum_{j=1}^{m_i}\nabla h_{ij}(x)\underbrace{[h_{ij}(x)]_+\prod_{k\neq j}[h_{ik}(x)]_+}_{\psi_i(x)}h_{ik}(x) \\
&= 2\psi_i(x)\sum_{j=1}^{m_i}\nabla h_{ij}(x)\prod_{k\neq j}h_{ik}(x),
\end{aligned} \tag{3.2}$$

where the second equality follows from the product rule of differentiation, which can be used since $[\,\cdot\,]_+^2$ is differentiable, see Lemma A.1 in [50], with $\nabla([w]_+^2) = 2[w]_+\nabla w$, and the third from the fact that $[w]_+^2 = w[w]_+$. Note, however, that this gradient is again nonsmooth due to the $[\,\cdot\,]_+$ operators in $\psi_i(x)$.

### 3.1.1 The algorithm

Given that the obstacle cost function can be made smooth by squaring it, and that $\psi_i^2(x)$ is, like $\psi_i(x)$, only nonzero for $x \in O_i$, it seems intuitive to consider a penalty approach using this function. As such, we will solve a sequence of problems

$$\underset{x\in\mathcal{X}}{\textbf{minimize}} \quad \mathfrak{L}_\mu(x), \tag{3.3}$$

where the objective is composed of the original objective function $\ell(x)$ and a quadratic penalty on $\psi_i$ for each obstacle, that is

$$\mathfrak{L}_\mu(x) = \ell(x) + \frac{\mu}{2}\sum_{i=1}^{n}\psi_i^2(x). \tag{3.4}$$

Here, $\mu$ is a (positive) penalty parameter that indicates the relative weight given to avoiding the obstacles. Using a vector of penalty factors, one for each constraint, is

**Algorithm 3.1** Quadratic penalty method for problem (3.3)

REQUIRE $x^0 \in \mathbb{R}^n,\ \mu^1 > 0,\ \epsilon^\nu, \epsilon^*, \eta^* \geq 0,\ \{\epsilon^\nu\} \to \epsilon^*,\ \omega > 1$
PROVIDE Solution $x^*$ to (3.1)
1: **for** $\nu = 1, 2, \ldots$ **do**
2: Use inner optimization algorithm to minimize $\mathfrak{L}_{\mu^\nu}(x)$ as in (3.4) with starting point $x^{\nu-1}$ until we find $x^\nu \in \mathcal{X}$ and associated $e^\nu$ satisfying (3.6).
3: **if** $\|e^\nu\|_\infty \leq \epsilon^*$ **and** $\|\psi(x^\nu)\|_\infty \leq \eta^*$ **then**
4: Return $x^* \leftarrow x^\nu$.
5: $\mu^{\nu+1} \leftarrow \omega\mu^\nu$

typical in practice, but will not be considered in the analysis for the sake of keeping the presentation simple. A common extension to the quadratic penalty method is the augmented Lagrangian method, in which the objective of the subproblem is $\mathfrak{L}_\mu(x) + \sum_{i=1}^n \lambda_i \psi_i(x)$, with $\lambda$ a running estimate of the Lagrange multipliers of the problem. Although this method generally has favorable properties with respect to the quadratic penalty method, it is not applicable here since the last term is again nonsmooth, and this would severely complicate and limit the techniques applicable for the subproblem minimization. On the contrary, owing to (3.2), $\mathfrak{L}_\mu(x)$ is continuously differentiable, with gradient

$$\nabla\mathfrak{L}_\mu(x) = \nabla\ell(x) + \mu \sum_{i=1}^{n} \psi_i(x) \sum_{j=1}^{m_i} \nabla h_{ij}(x) \prod_{k \neq j} h_{ik}(x). \tag{3.5}$$

Hence, a smooth optimization technique can be used to perform the minimization in (3.3). Our choice for this inner algorithm will be outlined in Section 3.2. Solving each problem (3.3) for a fixed value $\mu^\nu$ is performed approximately, yielding $x^\nu \in \mathcal{X}$ as an $\epsilon^\nu$-approximate Karush-Kuhn-Tucker (KKT) point, that is, a feasible point $x^\nu$ satisfying

$$\exists e^\nu : e^\nu \in \nabla\mathfrak{L}_{\mu^\nu}(x^\nu) + N_{\mathcal{X}}(x^\nu), \quad \|e^\nu\|_\infty \leq \epsilon^\nu. \tag{3.6}$$

Here, $\{\epsilon^\nu\}$ is a sequence of positive tolerances tending towards some final tolerance $\epsilon^*$ as $\nu \to \infty$. The steps of the quadratic penalty method are given in Algorithm 3.1.

**Remark 3.1.** In Algorithm 3.1, let $\psi(x)$ denote the vector of all obstacle cost functions, i.e. $(\psi_1(x), \ldots, \psi_{N_O}(x))$, such that $\|\psi(x)\|_\infty \leq \eta^* \Leftrightarrow \forall i : \psi_i(x^\nu) \leq \eta^*$. Note that, in practice, the algorithm could also converge to a point of local infeasibility, where (at least) one $\psi_i(x)$ does not tend to zero. □

### 3.1.2 The parameters

In the penalty method, the penalty factors are raised until the optimization solver has converged to a solution for which the norm of the obstacle cost function is lower

than a certain tolerance $\eta^*$. Each subproblem (3.3) is a soft-constrained version of the original problem (3.1). Only for an infinite value of $\mu$ are these two problems equivalent, since the quadratic penalty then corresponds to an indicator function on the feasible set (outside of obstacles). Therefore, the quadratic penalty method is usually only exact ($\eta^* = 0$) if the penalty factor is raised to infinity [121]. With this in mind, a strictly positive tolerance is chosen, such that an infinite penalty is not required. In our algorithm, this is often on the order of $\eta^* = 10^{-2}$. Virtual enlargements of the obstacle furthermore complement this formulation, so that the real obstacles can in fact be completely avoided, even though the constraint tolerance is strictly positive. Such enlargements also allow the formulation to be used for a vehicle with a finite width.

In practice, the penalty parameter is also capped from above, say by $\mu^* > 0$, such that step 5 is replaced by

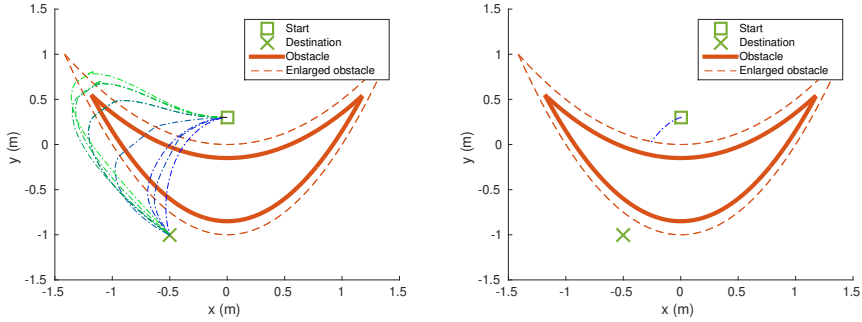
5: $\mu^{\nu+1} \leftarrow \min(\omega\mu^\nu, \mu^*)$.

Consequently, the termination criterion in step 3 should change accordingly to

3: **If** $\|e^\nu\|_\infty \leq \epsilon^*$ **and** $(\|\psi(x^\nu)\|_\infty \leq \eta^*$ **or** $\mu^\nu == \mu^*)$ **then**

The reason for upper bounding the penalty parameter is twofold. First, high values of this factor lead to ill conditioning of the subproblem (3.3), which can make it difficult for the inner solver to terminate quickly [15, §2.1]. Second, as mentioned in Remark 3.1, the sequence might have a limit point which is locally infeasible. Therefore, if $\|\psi(x^\nu)\|_\infty$ is still large for $\mu^\nu = \mu^*$, then the sequence may be converging to such a point, although the cap $\mu^*$ might also simply be too small. A good choice for the cap for our problems was $\mu^* = 10^4$, but this may well be problem dependent. For the theoretical results of §4, we require unbounded penalty parameters, as outlined in the original step 5. In practice, we also specify a maximum amount of iterations (or time), but again this is left out of Algorithm 3.1, and will be left out of any algorithm in the remainder of this text, for simplicity.

A penalty update factor $\omega$ is used to increase the penalty factor at each outer iteration. Typically, recommended values of this factor are somewhere in the interval $[1.5, 10]$ [121, §17.1] or $[4, 10]$ [15, §2.1]. However, there is always a trade-off in choosing this value: low values make the different optimization problems more similar and thus easier to warm start, but more problems will have to be solved in order to converge to a feasible solution. High values in contrast, render the consecutive optimization problems more difficult, but fewer of them are needed. It is observed that, for our motion planning problem, the optimization problems do not suffer from a high penalty update factor, thus $\omega$ is here chosen to be 10. In addition, after every update of the penalty factor, the solver is warm started with the solution from the previous iteration, as indicated in step 2.




**(a)** *Penalty method.* **(b)** *Fixed (high) penalty factor.*

**Figure 3.1:** *Illustration of the penalty method. The enlarged obstacle is defined by* $O = \{(x, y) : y > x^2 - 1, y < x^2/2\}$.

### 3.1.3 Practical benefit

Figure 3.1a shows an illustration of Algorithm 3.1 applied to a problem with a crescent-shaped obstacle. The trajectories ranging from blue to green correspond to approximate solutions of (3.3) for increasing penalty factors, which determine the balance between a feasible trajectory and one that is optimal for the least squares objective $\ell(x)$, as given in (2.4). The enlarged obstacle can never be completely avoided, but for high enough penalty factors, the original obstacle can be, as illustrated by the final two green trajectories. The combination of a virtual enlargement of the obstacle and finite values for the penalty factors is therefore indeed successful. Figure 3.1a also demonstrates that successive trajectories are usually similar in shape even though the penalty factors are updated somewhat aggressively in this work. Therefore, warm starting aids tremendously in the convergence of the subproblems.

The application of the penalty method may have an additional qualitative benefit for obstacle avoidance problems, illustrated by the difference between Figure 3.1a and Figure 3.1b. Assuming some obstacle is blocking the shortest path from start to destination, the initial trajectory calculated with low penalty factors is very likely to arrive at the destination while violating obstacle avoidance constraints. Subsequent iterations with higher and higher penalty factors tend to homotopically push the trajectory to the edge of the obstacle while remaining connected to the destination. In contrast, solving the problem only once with a high value for the penalty factors can impede convergence to a trajectory that reaches the destination since the vehicle is more likely to get stuck behind an obstacle, as illustrated in Figure 3.1b. Using the penalty method for autonomous navigation problems of this sort therefore aids in avoiding the local minimum.

What remains now is to discuss the most computationally expensive step in Algorithm 3.1, namely solving the subproblem in step 2. As mentioned, we can make use

of the recently introduced proximal averaged Newton-type method for optimal control (PANOC) [143].

## 3.2 PANOC

PANOC belongs to a class of proximal algorithms [122]. In essence, these algorithms use the proximal operator, as defined in Definition 1.1, one way or another. Proximal algorithms have recently received increased attention, since they are first-order methods and so do not require expensive matrix computations like in Newton-type methods. As such, their iterations are typically cheap, but at the price of some robustness against ill conditioning and potentially requiring a high number of iterations. As first-order methods, they may require a large amount of iterations to converge, especially for strict termination tolerances. Furthermore, proximal algorithms are typically only useful when the proximal mapping is easily computable. The reader is referred to [122] for an overview of many interesting applications which belong in this category. For our purpose here, it is important to recognize that the proximal mapping applied to an indicator function (1.3) of a set is equivalent to the projection operator for that set. Therefore, given that $\mathcal{X}$ is a set on which it is easy to project, as for instance for simple bound constraints on the control inputs, then $\operatorname{prox}_{\delta_{\mathcal{X}}}(x) = \Pi_{\mathcal{X}}(x)$ is tractable.

### 3.2.1 Proximal gradient method

One example of a popular proximal algorithm is the proximal gradient method, also known as forward-backward splitting. This method applies to a problem of type

$$\textbf{minimize} \quad f(x) + g(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is an $L_f$-smooth function, and $g : \mathbb{R}^n \to \overline{\mathbb{R}}$ a proper and lsc function. The proximal gradient method starts from an initial point $x^0$ and iteratively computes

$$x^{k+1} \in T_\gamma(x^k) \coloneqq \operatorname{prox}_{\gamma g}(x^k - \gamma \nabla f(x^k)),$$

where $\gamma > 0$ is a suitable step size. The proximal gradient mapping $T_\gamma$ consists of a gradient descent step on $f$, which is a forward step since it explicitly relies on current information, and the proximal mapping on $g$, which is a backward step since it is implicit as it involves a minimization, hence the names proximal gradient method and forward-backward splitting. Since problem (3.3) is equivalent to

$$\textbf{minimize} \quad \mathfrak{L}_\mu(x) + \delta_{\mathcal{X}}(x),$$

we can recognize $f = \mathfrak{L}_\mu$ and $g = \delta_{\mathcal{X}}$, such that the proximal gradient method for this problem amounts to

$$x^{k+1} = T_\gamma(x) = \Pi_{\mathcal{X}}(x^k - \gamma \nabla \mathfrak{L}_\mu(x^k)). \tag{3.7}$$

Here, it is assumed that $\mathcal{X}$ is convex, which makes $g = \delta_{\mathcal{X}}$ convex and $\Pi_{\mathcal{X}}$ have a unique value, hence the replacement of the $\in$ operator by the $=$ operator. Although extensively employed in practice, the proximal gradient algorithm, being a first-order method, has Q-linear convergence at best, and the Q-factor approaches one when the problem is more ill conditioned.

What makes PANOC an innovative proximal algorithm is that it combines proximal gradient steps with efficient Newton-type steps to improve the convergence rate.

### 3.2.2 Newton-type acceleration

Finding a stationary point for problem (3.3), since $g$ is convex, is equivalent to finding a point at which the fixed-point residual $R_\gamma$ is equal to zero [143], where

$$R_\gamma(x) = \frac{1}{\gamma}(x - T_\gamma(x)), \tag{3.8}$$

since $R_\gamma(x) = 0$ can easily be transformed into (3.6), with $e^\nu = 0$, by substituting (3.7) and the definition of the projection operator in (1.4). This motivates addressing the problem by applying a Newton-type step of the form

$$x^{k+1} = x^k - H^k R_\gamma(x^k),$$

where $H^k$ should capture curvature information. Typically, this is a running approximation of the (local) inverse Hessian of $R_\gamma(x^k)$. In quasi-Newton methods, for instance, $H^k$ starts typically from some multiple of the identity and is updated after every iteration using the secant condition

$$x^{k+1} - x^k = H^{k+1}(R_\gamma(x^{k+1}) - R_\gamma(x^k)). \tag{3.9}$$

There is still some freedom to choose a matrix $H^{k+1}$ satisfying this secant condition, and different quasi-Newton methods, such as the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) and symmetric rank-1 (SR1) methods, derive different updates based on this freedom [121, §6]. What they have in common, however, is that the update is of low-rank, respectively 2 and 1 for BFGS and SR1. Furthermore, there is a limited memory variant of BFGS (L-BFGS) which avoids storing $H^k$, typically a dense matrix, but instead stores this matrix implicitly by saving a finite number, say $m$, of vectors $x^{k+1} - x^k$ and $(R_\gamma(x^{k+1}) - R_\gamma(x^k))$ [121, §7.2]. The matrix vector product $H^{k+1}(R_\gamma(x^{k+1}) - R_\gamma(x^k))$ can then be carried out via a two-loop recursion of inner products and vector summations, see [121, Algorithm 7.4]. L-BFGS not only has the advantage of requiring little memory, but it typically also works well in practice since

**Algorithm 3.2** PANOC algorithm for problem (3.3)

REQUIRE $L_{\mathfrak{L}_\mu} > 0$, $\gamma \in (0, \frac{1}{L_{\mathfrak{L}_\mu}})$, $\sigma \in (0, \frac{\gamma}{2}(1 - \gamma\frac{L_{\mathfrak{L}_\mu}}{2}))$, $x^0 \in \mathbb{R}^n$, $\tau > 0$, L-BFGS memory $mem$.
PROVIDE Solution $x^*$.

1: **for** $k = 0, 1, 2, \ldots$ **do**
2: $\bar{x}^k \leftarrow \Pi_{\mathcal{X}}(x^k - \gamma\nabla\mathfrak{L}_\mu(x^k))$
3: $r^k \leftarrow \frac{x^k - \bar{x}^k}{\gamma}$
4: **if** $\|r^k\|_\infty < \tau$ **then**
5: Return $x^* \leftarrow \bar{x}^k$.
6: $d^k = -H^k r^k$ using L-BFGS
7: $x^{k+1} \leftarrow x^k - (1 - \alpha^k)\gamma r^k + \alpha^k d^k$, with $\alpha^k$ the largest in $\{\frac{1}{2^i} : i \in \mathbb{N}\}$ such that

$$\varphi_\gamma(x^{k+1}) \leq \varphi_\gamma(x^k) - \sigma\|r^k\|^2 \tag{3.10}$$

old curvature information, which is less likely to be relevant for the current (inverse) Hessian, is erased periodically. Therefore, L-BFGS will be our quasi-Newton method of choice to accelerate the proximal gradient method.

It is well known, however, that steps of type (3.9) are not guaranteed to converge for arbitrary starting points, and may in fact diverge if the initial point is not close to a solution. Therefore, a globalization strategy is necessary. The authors of [151] propose such a globalization technique based on the forward-backward envelope (FBE) $\varphi_\gamma^{\text{FB}}(x) \coloneqq f(x) - \frac{\gamma}{2}\|\nabla f(x)\|^2 + g^\gamma(x - \gamma\nabla f(x))$, where $g^\gamma$ is the Moreau-envelope of $g$ as defined in (1.8). The FBE for problem (3.3) can be worked out as

$$\varphi_\gamma^{\text{FB}}(x) = \mathfrak{L}_\mu(x) - \frac{\gamma}{2}\|\nabla\mathfrak{L}_\mu(x)\|^2 + \frac{1}{2\gamma}\operatorname{dist}^2_{\mathcal{X}}(x - \gamma\nabla\mathfrak{L}_\mu(x)).$$

It can be shown (for $\gamma < L_{\mathfrak{L}_\mu}^{-1}$) that $\varphi_\gamma^{\text{FB}}(x)$ and $\mathfrak{L}_\mu(x) + \delta_{\mathcal{X}}(x)$ have the same minimizers, and hence minimization techniques can be directly applied to the FBE. In fact, forward-backward splitting steps are equivalent to scaled gradient descent steps on the FBE. In order to combine proximal gradient and quasi-Newton steps, PANOC considers an update that is composed of a convex combination of both, such that a sufficient decrease is guaranteed on the FBE, ensuring global convergence. The resulting averaging strategy and other steps of PANOC are summarized in Algorithm 3.2.

In this algorithm, $L_{\mathfrak{L}_\mu}$ denotes the Lipschitz constant of the objective function. This might not be known in practice. In fact, for our quadratic penalty method, this Lipschitz constant is likely dependent on the value of the penalty parameter, and so may increase over the successive subproblems. The PANOC algorithm, however, can also run with an estimate for this constant which is then updated in between iterations, by adding another step after step 3

3bis: **if** $\mathfrak{L}_\mu(\bar{x}^k) > \mathfrak{L}_\mu(u^k) - \gamma(\nabla\mathfrak{L}_\mu(x^k))^\top r^k + \frac{L_{\mathfrak{L}_\mu}}{2}\|\gamma r^k\|^2$ **then**
$\gamma \leftarrow \frac{\gamma}{2}$, $L_{\mathfrak{L}_\mu} \leftarrow 2L_{\mathfrak{L}_\mu}$, $\sigma \leftarrow \frac{\sigma}{2}$, go to step 2.

The reliance of PANOC on this Lipschitz estimate is another reason to upper bound the penalty parameter as done in Section 3.1.2.

Under certain assumptions, the PANOC algorithm can be shown to acquire the same convergence rate as its quasi-Newton steps, which is superlinear convergence when the Dennis-Moré condition is satisfied [143, Theorem III.5]. The reason for this is that close to a solution, the candidate quasi-Newton steps are accepted with $\alpha^k = 1$. Although L-BFGS does not exhibit this superlinear convergence in theory, for the aforementioned reasons of requiring less memory and its splendid performance in practice, this method is chosen as the acceleration in our application of PANOC. For a more in-depth discussion on the theoretical properties of PANOC, the reader is referred to [143].

Note that, since $\mathcal{X}$ is convex, the termination criterion in step 4 is equivalent to (3.6), since

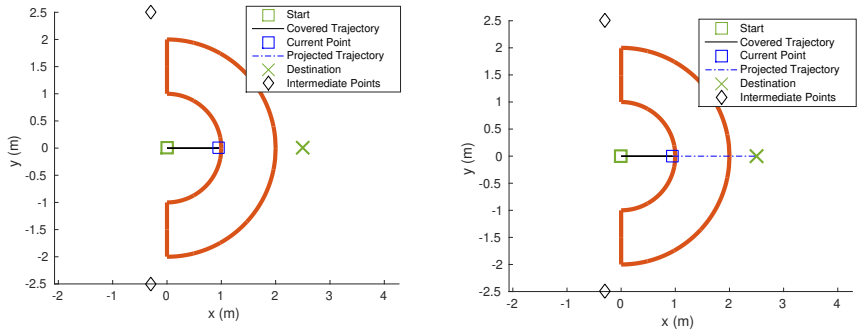
$$\begin{aligned} &\Pi_X(x - \gamma\nabla\mathfrak{L}_\mu(x)) = \bar{x} \\ \Leftrightarrow\ &x - \gamma\nabla\mathfrak{L}_\mu(x) - \bar{x} \in \mathcal{N}_\mathcal{X}(\bar{x}) \\ \Leftrightarrow\ &r = \frac{x - \bar{x}}{\gamma} \in \nabla\mathfrak{L}_\mu(x) + \mathcal{N}_\mathcal{X}(\bar{x}), \end{aligned}$$

where the first implication follows from the definition of the projection (1.4), and the second from the fact that $\gamma$ is a positive constant and $\mathcal{N}_\mathcal{X}$ is a cone, so $\frac{1}{\gamma}\mathcal{N}_\mathcal{X} = \mathcal{N}_\mathcal{X}$. Therefore, we can recognize that the final fixed-point residual is our error vector $e^\nu$ in the KKT condition, and we should set $\tau = \epsilon^\nu$ when calling PANOC to satisfy (3.6).

The approach for dealing with optimization problem (3.1) has been outlined. However, it is a general nonlinear, nonconvex problem, and the obstacles render the solution space nonconvex. Therefore, local minima often exist near obstacles, similarly to what happens in the potential field method of Section 2.2.2. The next section introduces heuristics to circumvent these local minima when needed.

## 3.3 Heuristics

Figure 3.2a illustrates an obstacle that creates an obvious local minimum, since the blue square is locally the closest point to the destination in the feasible region. To aid in the convergence to a feasible trajectory that reaches the destination, two additional heuristics have been developed for our MPC controller. The first heuristic is a fail-safe in case Algorithm 3.1 returns an infeasible solution, where the MPC controller will stop the vehicle (setting all velocities equal to zero) if the state becomes infeasible within the next three time steps. An infeasible solution may be returned either due to a low




**(a)** *High penalty factors cap.* **(b)** *Low penalty factors cap.*

**Figure 3.2:** *Illustration of the local minimum behind the obstacle, and the hold-in-place heuristic and choice of intermediate points. The half-disc shaped obstacle is defined by* $O = \{(x, y) : x^2 + y^2 > 1, x^2 + y^2 < 4, x > 0\}$.

penalty parameter cap, as illustrated in Figure 3.2b, due to convergence to a locally infeasible point, or due to hitting the maximum amount of iterations. The second heuristic consists of guiding the vehicle to one or more intermediate destinations whenever the vehicle remains in place for more than one time instant, due to the first heuristic or due to converging to a local minimum as in Figure 3.2a, before continuing on towards the final destination.

The intention behind this last heuristic is to guide the vehicle around the obstacle. A good intermediate destination is easy to reach from both the point where the vehicle was previously stuck at and the final destination. It will usually be close to a corner or edge point of the obstacle. This principle is also illustrated in Figure 3.2, where appropriate intermediate destinations are located near the black diamonds.

To avoid getting stuck in a local optimum near an obstacle, a suitable set of intermediate destinations must be available and a relevant choice from this set of points is necessary. The user may provide such a set, based on knowledge of the obstacle definitions (and therefore the locations of their corners). This approach, however, is hard to justify in an automated setup. Instead, in order to generate suitable intermediate points automatically, variants of Dijkstra's algorithm can be used to perform a simple graph search, see Section 2.2.1. This work utilizes the A* search algorithm [81], because of its simplicity and efficiency. The worst-case complexity of this algorithm in case a consistent heuristic cost is used, is $O(N)$ [109], with $N$ the number of nodes in the graph. The heurstic cost used here is the Euclidean distance between a node and the goal node, which is indeed consistent.

Figure 3.3 shows the division of the space into a grid of square cells, and how the graph search returns a feasible geometric trajectory from the current point to the destination. In an unobstructed space, a graph search would find a straight path.

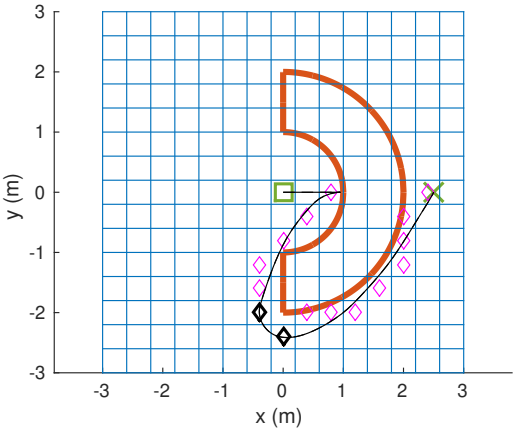



**Figure 3.3:** *Illustration of the graph search, represented by the magenta diamonds. At both of the black diamonds the direction changes (first left-right, then up-down), so these comprise the set of intermediate destinations. The black line denotes the final trajectory.*

Hence, (left-right or up-down) direction changes stem from the presence of an obstacle, and the points at which they occur are close to the corners of said obstacle. These points are therefore suitable candidates for intermediate destinations. Each element of the set of intermediate destinations is successively set as the reference state $q_{\text{ref}}$ in (2.2). Since now both the starting state and the reference state should lie somewhere on the same side of the obstacle, it is much less likely for Algorithm 3.1 to get stuck in a local minimum. The MPC controller changes the reference state whenever the distance to the current one is smaller than some tolerance. When the set of intermediate points is exhausted, the reference state is reset to the original destination. This concludes the proposed methodology for solving problem (3.1).

## 3.4 Summary

This chapter considered a solution strategy for the optimization problem with general obstacles. The core of this approach was the quadratic penalty method, which solves a sequence of subproblems with a once differentiable objective function. An effective proximal algorithm, PANOC, which combines proximal gradient and quasi-Newton steps, was employed to solve these subproblems efficiently. Furthermore, a heuristic involving a graph search was proposed to circumvent any locally optimal or infeasible solution the algorithm may encounter during MPC simulations.

# Chapter 4

## Penalty Method for MPSEC: Convergence Results

This chapter derives convergence results for Algorithm 3.1. In the literature, convergence of the quadratic penalty method and the augmented Lagrangian method applied to smooth problems is covered in the monograph of Bertsekas [15]. In [18, Chapter 6] convergence results are extended to problems with implicit set constraints on the decision variables. In particular, it is shown that every limit point is a stationary point of an infeasibility measure, but no conditions guaranteeing feasibility of limit points are provided.

Even though the quadratic penalty method is fairly simple conceptually and the textbooks mentioned above have outlined general convergence results, it is not trivial to derive such results for Algorithm 3.1 applied to problem (3.1). In fact, it is difficult to list the first-order necessary conditions for stationarity due to the nonsmoothness of the constraints in (3.1b). Moreover, the nonsmooth formulation using the normal cone of the constraint set [134, Theorem 6.12] is not usable, as in general the normal cone, or an outer approximation of this set, cannot be derived from the functions defining the set exclusion constraints. This is due to the fact that 0 belongs to the subdifferential of the max-operator (at the boundary), which violates the condition in [134, Corollary 10.50]. Since the original problem leaves us empty-handed with regards to stationarity conditions, our only option is to dig for gold elsewhere.

Section 4.1 constructs a mathematical program with vertical complementarity constraints (MPCC) which is equivalent to problem (3.1). For this type of program, different notions of stationarity exist [137]. Section 4.2 lists and slightly extends these conditions. Finally, Section 4.3 shows that the (transformed) iterates of Algorithm 3.1 satisfy stationarity conditions of the MPCC under certain assumptions.

The material in this chapter is based on the publication [83].

## 4.1 Vertical complementarity constraints

### 4.1.1 Introduction

Mathematical programs with vertical complementarity constraints (MPCCs) were first introduced in [38] and are a special class of optimization problems that often arise in practice. They extend mathematical programs with *regular* complementarity constraints. Such a constraint dictates that at least one of two nonnegative quantities should be equal to zero at all times. Mathematically, this translates to the constraint that for $f, g : \mathbb{R}^n \to \mathbb{R}$

$$f(x) \geq 0,\ g(x) \geq 0, \quad f(x) \cdot g(x) = 0,$$

or equivalently

$$\min(f(x), g(x)) = 0.$$

Multiple complementarity constraints can be succinctly listed in vector format using the scalar product

$$F(x) \geq 0, G(x) \geq 0, \quad F(x)^\top G(x) = 0,$$

with $F, G : \mathbb{R}^n \to \mathbb{R}^m$.

Although such problems present theoretical difficulties since no solution satisfies the Mangasarian-Fromovitz constraint qualification (MFCQ), they can nevertheless be solved fairly reliably by standard nonlinear optimization solvers [98]. KNITRO [24] is an interior-point solver that uses an $\ell_1-$penalty for complementarity constraints [99]. SQP methods with an elastic mode for dealing with infeasible constraints, such as SNOPT [69], have also been shown to converge locally for problems with complementarity constraints [7, 8]. Finally, the application of augmented Lagrangian approaches to problems with complementarity constraints was investigated in [89].

Vertical complementarity constraints simply extend this concept of complementarity to more than two functions. As such, a set of $m$ vertical complementarity constraints over (at most) $l$ functions can be written as

$$\min\{F_{i1}(z), \ldots, F_{il}(z)\} = 0, \quad i = 1, \ldots, m,$$

with $F : \mathbb{R}^n \to \mathbb{R}^{m \times l}$.

By including these constraints in a regular nonlinear program, the mathematical program with vertical complementarity constraints can be written in the following form [137]

$$\begin{aligned} \underset{z\in\mathbb{R}^{n_z}}{\textbf{minimize}} \quad & f(z), \\ \textbf{subject to} \quad & G(z) \leq 0, \\ & H(z) = 0, \\ & \min\{F_{i1}(z), \ldots, F_{il}(z)\} = 0, \quad i = 1, \ldots, m, \end{aligned} \tag{4.1}$$

where $f : \mathbb{R}^{n_z} \to \mathbb{R}, G : \mathbb{R}^{n_z} \to \mathbb{R}^p, H : \mathbb{R}^{n_z} \to \mathbb{R}^q, F : \mathbb{R}^{n_z} \to \mathbb{R}^{m\times l}$. Looking at our original problem (3.1), it is clear that we will additionally need implicit set constraints $z \in Z$, with $Z$ some closed, convex set. On the other hand, developments outlined below show that equality constraints are not needed, and so as MPCC we consider instead the following problem

$$\begin{aligned} \underset{z\in Z}{\textbf{minimize}} \quad & f(z), \\ \textbf{subject to} \quad & G(z) \leq 0, \\ & \min\{F_{i1}(z), \ldots, F_{il}(z)\} = 0, \quad i = 1, \ldots, m. \end{aligned} \tag{4.2}$$

### 4.1.2 The equivalent MPCC

Before introducing an MPCC which is equivalent to (3.1), we first require the concept of active constraints. In general optimization problems, equality constraints and inequality constraints that hold as equality at a feasible point are called active, and inequality constraints that hold strictly at this point are called inactive. Reusing this terminology, let us introduce the set of *active obstacles* $I_{\overline{O}}(x)$ and the set of *active obstacle boundaries* $I_a(x)$ as

$$\begin{aligned} I_{\overline{O}}(x) &= \{i : x \in \overline{O}_i\}. \\ I_a(x) &= \{(i,j) : i \in I_{\overline{O}}(x), h_{ij}(x) = 0\}. \end{aligned} \tag{4.3}$$

For every active obstacle, the state is either inside or on its boundary, and if there is an active obstacle boundary, then the state is on this (part of the) boundary.

Aside from active constraints, typically the constraint gradients are crucial in the analysis of constraint qualifications and stationarity conditions. However, $\psi_i(x)$ is nonsmooth, and unfortunately so at the interesting part, which is the obstacle boundary. For now, let us instead construct a *smooth* version of this obstacle cost function $\psi_i(x)$, created by dropping the $[\cdot]_+$ operators, and denote this *extended obstacle cost function*

as $\widetilde{\psi}_i(x)$, such that

$$\widetilde{\psi}_i(x) = \prod_{j=1}^{m_i} h_{ij}(x). \tag{4.4}$$

This function is equal to the obstacle cost function for points in $\overline{O}_i$. Its gradient at a point $\overline{x}$ on the obstacle boundary is therefore equal to $\nabla\widetilde{\psi}_i(\overline{x}) = \lim_{\substack{x \to \overline{x} \\ O_i}} \nabla\psi_i(x)$. As such, it serves as a meaningful placeholder for speaking about the constraint gradient of an active obstacle. The gradient of $\widetilde{\psi}_i(x)$ is given as

$$\nabla\widetilde{\psi}_i(x) = \sum_{j=1}^{m_i} \nabla h_{ij}(x) \prod_{k \neq j} h_{ik}(x). \tag{4.5}$$

Recalling (3.5) and using (4.5), $\nabla\mathfrak{L}_\mu(x)$ can now be written as

$$\nabla\mathfrak{L}_\mu(x) = \nabla\ell(x) + \mu \sum_{i=1}^{n} \psi_i(x)\nabla\widetilde{\psi}_i(x). \tag{4.6}$$

To arrive at an equivalent (smooth) MPCC reformulation of problem (3.1), slack variables $t_{ij}$ need to be introduced to eliminate the $[\,\cdot\,]_+$-operators. By letting $t_{ij}$ carry the meaning of $t_{ij} \geq [h_{ij}(x)]_+$, i.e. $t_{ij} \geq h_{ij}(x)$ and $t_{ij} \geq 0$, the equivalent MPCC is given as

$$\underset{(x,t)\in\mathcal{X}\times\mathbb{R}^{n_t}}{\textbf{minimize}} \quad \ell(x), \tag{4.7a}$$

$$\textbf{subject to} \quad t_{ij} \geq h_{ij}(x), \qquad i = 1,\ldots,m, \quad j = 1,\ldots,m_i, \tag{4.7b}$$

$$\min(t_{i1},\ldots,t_{im_i}) = 0, \quad i = 1,\ldots,m, \tag{4.7c}$$

with $n_t = \sum_{i=1}^{n} m_i$. This problem is indeed of type (4.2), with $z = (x,t)$, $Z = \mathcal{X}\times\mathbb{R}^{n_t}$, and smooth functions $f(z) = \ell(x)$, $G_k(z) = h_{ij}(x) - t_{ij}$, where $k = j + \sum_{l=1}^{i-1} m_l$, and $F_{ij}(z) = t_{ij}$.

Problems (3.1) and (4.7) are equivalent in the sense that they share the same cost function, which is independent of $t$, and that feasible points in one problem yield feasible points in the other. On the one hand, for any $(x,t)$ which is feasible for (4.7), $x$ is feasible also for (3.1), given the definition of the obstacle cost $\psi_i$ in (2.8). On the other hand, for any $x$ which is feasible for (3.1), it is trivial to show that for $t_{ij} = [h_{ij}(x)]_+$, the pair $(x,t)$ is also feasible for (4.7). For reasons mentioned in

Remark 4.2, however, we will use a slightly different formulation for $t$, namely

$$t_{ij} = \begin{cases} h_{ij}(x), & i \in I_{\overline{O}}(x), \\ 0, & i \notin I_{\overline{O}}(x), \quad j = \min \operatorname{argmin}_k \{h_{ik}(x)\}, \\ [h_{ij}(x)]_+ + \epsilon, & i \notin I_{\overline{O}}(x), \quad j \neq \min \operatorname{argmin}_k \{h_{ik}(x)\}, \end{cases} \tag{4.8}$$

with $\epsilon > 0$. Remarks 4.1 and 4.2 regarding this particular choice of the slack variables for respectively inactive and active obstacles show that the resulting point $(x, t)$ is also feasible for problem (4.7).

**Remark 4.1.** For active obstacles, the following reasoning can be made. Since $x$ is feasible, $\psi_i(x) = 0 \Leftrightarrow x \notin O_i$ for all $i$. For an active obstacle $i$ therefore, $x \in \overline{O}_i \setminus O_i = \partial O_i$ and recalling (2.7), this means that there is at least one $j$ for which $h_{ij}(x) = t_{ij} = 0$, according to the first case of (4.8). □

**Remark 4.2.** For inactive obstacles, the following reasoning can be made. For an obstacle $i$ to be inactive, at least one of the boundary functions has to be strictly negative, that is $\exists j : h_{ij}(x) < 0$. For one such $j$, the slack variable can be set to zero to satisfy the constraints in (4.7), as is done in the second case of (4.8). In the third case of (4.8), that is, for the other boundaries of the inactive obstacle, the slack variables can be chosen freely as long as they are strictly greater than $h_{ij}(x)$ and 0. This strict inequality, enforced using $\epsilon > 0$, is necessary to prevent obstacle boundaries $(i, j) : i \notin I_{\overline{O}}, h_{ij}(x) = 0$ from yielding active constraints in problem (4.7). □

## 4.2 MPCC stationarity

This section discusses two sets of stationarity conditions of problem (4.2). For a more thorough overview, the reader is referred to [137]. The relevant stationarity conditions for the convergence results of Section 4.3 are reproduced here, with a minor generalization to account for implicit set constraints, obtained from [134].

The first subsection discusses the definition of the linear independence constraint qualification (LICQ) for MPCCs (MPCC-LICQ), and two assumptions which can be shown to imply MPCC-LICQ at a certain point. The second subsection lists the strong and Clarke stationarity conditions for MPCCs, and their applications to problem (4.2).

### 4.2.1 Constraint qualifications

**Definition 4.3.** *A feasible point $z$ of problem (4.1) satisfies MPCC-LICQ if all the active constraint gradients*

$$\nabla G_r(z), \quad r : G_r(z) = 0$$

$$\nabla H_s(z), \quad s = 1, \ldots, q$$

$$\nabla F_{ij}(z),\ (i,j) : F_{ij}(z) = 0$$

*are linearly independent [137].*

By applying the MPCC-LICQ from [137] to problem (4.2), we obtain the condition that there exists no vector $y = (y^G, y^F) \neq 0$ that satisfies

$$-\sum_{r:G_r(z)=0} y_r^G \nabla G_r(z) - \sum_{(i,j):F_{ij}(z)=0} y_{ij}^F \nabla F_{ij}(z) \in N_Z(z). \tag{4.9}$$

To translate this condition for problem (4.7), let the index set $I_h(z)$ and $I_t(z)$ denote respectively the active inequality constraints and the active complementarity components at $z = (x, t)$:

$$I_h(z) = \{(i,j) : h_{ij}(x) - t_{ij} = 0\}, \tag{4.10}$$

$$I_t(z) = \{(i,j) : t_{ij} = 0\}. \tag{4.11}$$

Furthermore, from [134, Proposition 6.41], we have that $N_Z(z) = N_{\mathcal{X}}(x) \times \{0\}$. Given (4.9), (4.10) and (4.11), the following condition for MPCC-LICQ is obtained by splitting along $x$ and $t$. Note that the dependence of the index sets on $z$ is omitted for the sake of brevity in the mathematical expressions.

**Definition 4.4.** *A feasible point $z = (x, t)$ of problem (4.7) satisfies MPCC-LICQ iff there exists no vector $(y^h, y^t) \neq 0$ satisfying the following conditions*

$$\begin{cases} y_{ij}^h = 0 \quad \text{for } (i,j) \in I_h \setminus I_t, \\ y_{ij}^t = 0 \quad \text{for } (i,j) \in I_t \setminus I_h, \\ -\displaystyle\sum_{(i,j)\in I_h \cap I_t} y_{ij}^h \nabla h_{ij}(x) \in N_{\mathcal{X}}(x). \end{cases} \tag{4.12}$$

**Remark 4.5.** It is well known that LICQ implies the regular and strict Mangasarian-Fromovitz constraint qualifications (MCFQ and SMFCQ), [55, Chapter 2]. This also holds for the constraint qualifications defined for MPCCs, as these are the regular LICQ, MFCQ and SMFCQ applied to an associated NLP problem, see [137]. Furthermore, it is not as straightforward to extend MCFQ and SMFCQ in the presence of a normal cone as it is to extend LICQ. Therefore, only MPCC-LICQ will be used in this work. However, as shown in [138], this is not a stringent assumption for MPCCs. □

In Section 4.3, two assumptions will be used to prove that limit points of the generated sequences, if they exist, satisfy certain stationarity conditions. These assumptions can be thought of as mirroring linear independence of the active constraint gradients for the obstacle formulations. Before we introduce these assumptions and show that they imply MPCC-LICQ, we need the following result regarding the index set of the active obstacle boundaries:

**Lemma 4.6.** *Assume that $(x,t)$ is feasible for (4.7) and $t$ is given by (4.8). Then*

$$I_a(x) = I_h(x,t) \cap I_t(x,t).$$

*Proof.* This lemma can be proven by showing the set inclusion in both directions. In this proof the dependence of the index sets on the variables will be omitted. From (4.3) and the first case of (4.8) it follows that for $(i,j) \in I_a : h_{ij}(x) = t_{ij} = 0$. Therefore, we immediately have $I_a \subseteq I_h \cap I_t$.

The other way around, suppose that $(i,j) \in (I_h \cap I_t) \setminus I_a$. From $(i,j) \in I_h \cap I_t$ we have that $h_{ij}(x) = t_{ij} = 0$. Then, from $(i,j) \notin I_a$ and (4.3), it follows that $i \notin I_{\overline{O}}$. However, this implies that $(i,j)$ does not belong to the first case of (4.8). Neither does it belong to the second case as $h_{ij}(x) = 0$, cf. Remark 4.2. Therefore $(i,j)$ belongs to the third case of (4.8), and thus $t_{ij} = \epsilon > 0$. This contradiction disproves the assumption that $(i,j) \in (I_h \cap I_t)$, and therefore proves that $(I_h \cap I_t) \subseteq I_a$. □

The following two lemmas establish that certain assumptions imply the conditions (4.12).

**Lemma 4.7.** *Assume that at a point $x$ the following implication holds:*

$$-\sum_{i \in I_{\overline{O}}(x)} y_i \nabla \widetilde{\psi}_i(x) \in N_{\mathcal{X}}(x) \Rightarrow y = 0. \tag{4.13}$$

*If, furthermore, $(x,t)$ is feasible for problem (4.7), with $t$ given by (4.8), then (4.12) is satisfied at this point.*

*Proof.* Note that $0 \in N_{\mathcal{X}}(x)$ by definition (1.5). Therefore, (4.13) implies that the active obstacle cost gradients are linearly independent and thus not equal to zero. Given (4.5), this excludes the case where there is more than one active obstacle boundary. Moreover, because $x$ is feasible, $x \notin O_i$, and thus for $i \in I_{\overline{O}}(x)$, $x \in \overline{O}_i \setminus O_i = \partial O_i$, which implies that for each active obstacle, there is at least one active obstacle boundary. In conclusion, for each active obstacle, there is exactly one active obstacle boundary. Let $j_i$ denote the index of this active boundary, such that $h_{ij_i}(x) = 0$. From (4.5) it then follows that

$$\nabla \widetilde{\psi}_i(x) = \nabla h_{ij_i}(x) \prod_{k \neq j_i} h_{ik}(x), \quad (i,j_i) \in I_a(x). \tag{4.14}$$

From (4.14) and Lemma 4.6, (4.13) is now equivalent to (4.12). □

As mentioned in the proof of Lemma 4.7, equation (4.13) can only be satisfied if for every active obstacle there is only one active obstacle boundary, and if the gradient of this boundary function is nonzero at its boundary. Boundary functions with this latter property must therefore be chosen in order for the algorithm to work provably. Moreover, as the above assumption does not hold in case of multiple active boundaries

of the same obstacle, we provide also a second possible assumption that does cover this case.

**Lemma 4.8.** *If at a feasible point $(x,t)$, with $t$ given by (4.8), it holds that*

$$-\sum_{(i,j)\in I_a(x)} y_{ij}\nabla h_{ij}(x) \in N_{\mathcal{X}}(x) \Rightarrow y = 0, \tag{4.15}$$

*then (4.12) is satisfied at this point.*

*Proof.* The proof for this lemma is immediate as, according to Lemma 4.6, (4.15) and (4.12) are equivalent. □

**Remark 4.9.** The conditions (4.15) and (4.12) are only equivalent because of our choice of the slack variables (4.8), as this allows the application of Lemma 4.6. Had we chosen $\epsilon = 0$ in (4.8), (4.12) would be a set of stricter conditions than (4.15). However, as these points are neglected by choosing a strictly positive $\epsilon$, (4.15) and (4.12) can be considered as equivalent. Note also that if (4.13) holds at a feasible point, then (4.15) holds at this point because of (4.14). □

Figure 4.1 illustrates the gradient of the extended obstacle cost function along the (upper and left) boundaries of a rectangular obstacle set, and the cornerpoints where (4.13) is not satisfied, but (4.15) is. Note that assumption (4.15) is sufficient to guarantee the linear independence condition of Lemma 2.2. The same holds for (4.13) whenever there is only one active boundary constraint. Therefore, when working under either of these two assumptions, the obstacle boundary is defined as in (2.7).

### 4.2.2 Stationarity conditions

The stationarity conditions for problem (4.2) rely on the Lagrangian function $\mathcal{L}$ associated with this problem, given as [137, Eq. (1) with $\alpha = 1$],

$$\mathcal{L}(z,\Gamma,\lambda) = f(z) - \sum_{i=1}^{m}\sum_{j=1}^{l} F_{ij}(z)\Gamma_{ij} + \sum_{i=1}^{p} G_i(z)\lambda_i,$$

where $\Gamma \in \mathbb{R}^{m\times l}, \lambda \in \mathbb{R}^p$ are the multipliers corresponding to the complementarity and inequality constraints, respectively. In the following, two sets of stationarity conditions are derived, the first of which are the *strong stationarity conditions*. The strong stationarity conditions are the KKT-conditions applied to a *relaxed* NLP (RNLP) formulation of the MPCC [137]. This RNLP formulation at a feasible point $z$ of problem (4.2) is the following program

$$\begin{aligned} \underset{z'\in Z}{\textbf{minimize}} \quad & f(z') \\ \textbf{subject to} \quad & G(z') \leq 0, \end{aligned} \tag{4.16}$$

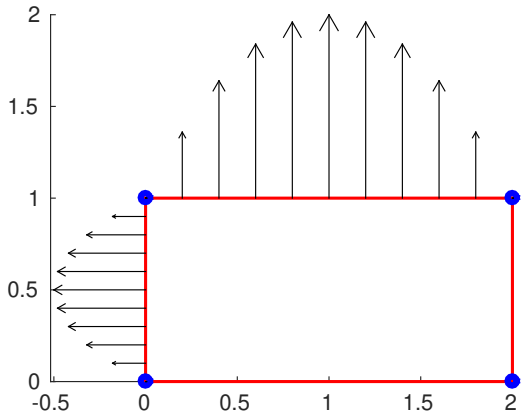



**Figure 4.1:** *The rectangular obstacle is given by* $O = \{(x, y) : x > 0,\ 2 - x > 0,\ y > 0,\ 1 - y > 0\}$. *The arrows represent* $-\nabla\widetilde{\psi}(x, y)$ *at several boundary points. More than one obstacle boundary is active at the corner, hence the gradient vanishes there such that* (4.13) *is not satisfied, whereas* (4.15) *is.*

$$F_{ij}(z') \begin{cases} = 0 \text{ if } F_{ij}(z) = 0 \text{ and } F_{ik}(z) > 0 \text{ for } k \neq j \\ \geq 0 \text{ otherwise.} \end{cases}$$

By applying [134, Corollary 6.15] to problem (4.16), the strong stationarity conditions are obtained as follows.

**Definition 4.10.** *For a feasible point* $z$ *of problem* (4.2)*, the strong stationarity conditions are the following conditions on* $z$ *and on the multipliers* $\Gamma$ *and* $\lambda$

$$0 \in \nabla_z \mathcal{L}(z, \Gamma, \lambda) + N_Z(z), \tag{4.17a}$$

$$F_{ij}(z)\Gamma_{ij} = 0, \quad i = 1, \ldots, m, \quad j = 1, \ldots, l, \tag{4.17b}$$

$$\lambda_i \geq 0, \quad i = 1, \ldots, p, \tag{4.17c}$$

$$G_i(z)\lambda_i = 0, \quad i = 1, \ldots, p, \tag{4.17d}$$

$$\Gamma_{ij} \geq 0, \quad \text{if } \exists k \neq j : t_{ij} = t_{ik} = 0. \tag{4.17e}$$

These conditions are only necessary in the general case when strict complementarity holds [137], stating that for a point $z$ of problem (4.2), there is for every $i$ only one $j_i$ such that $F_{ij_i}(z) = 0$. From Definition 4.10, with (4.17a) split according to $x$ and $t$, we can now derive the strong stationarity conditions for problem (4.7) as the following

conditions on $(x,t)$ and the unique multipliers $(\Gamma,\lambda)$

$$0 \in \nabla \ell(x) + \sum_{i=1}^{n}\sum_{j=1}^{m_i} \lambda_{ij}\nabla h_{ij}(x) + N_{\mathcal{X}}(x), \tag{4.18a}$$

$$\lambda_{ij} + \Gamma_{ij} = 0, \quad i = 1,\dots,m, \quad j = 1,\dots,m_i, \tag{4.18b}$$

$$t_{ij}\Gamma_{ij} = 0, \quad i = 1,\dots,m, \quad j = 1,\dots,m_i, \tag{4.18c}$$

$$\lambda_{ij} \geq 0, \quad i = 1,\dots,m, \quad j = 1,\dots,m_i, \tag{4.18d}$$

$$(h_{ij}(x) - t_{ij})\lambda_{ij} = 0, \quad i = 1,\dots,m, \quad j = 1,\dots,m_i, \tag{4.18e}$$

$$\Gamma_{ij} \geq 0, \quad \text{if } \exists k \neq j : t_{ij} = t_{ik} = 0. \tag{4.18f}$$

The second set of stationarity conditions used in this thesis are the *Clarke stationarity conditions*. In order to obtain their form with an inclusion for the implicit set constraints, we mimic here the proof for [137, Theorem 2].

**Theorem 4.11.** *If for a local minimizer $z$ of problem (4.2) there is no vector $(y^G, y^F) \neq 0$ for which (4.9) holds, i.e. MPCC-LICQ is satisfied, then there exist multipliers $\Gamma$ and $\lambda$ such that*

$$0 \in \nabla_z \mathcal{L}(z,\Gamma,\lambda) + N_Z(z), \tag{4.19a}$$

$$F_{ij}(z)\Gamma_{ij} = 0, \quad i = 1,\dots,m, \quad j = 1,\dots,l, \tag{4.19b}$$

$$\lambda_i \geq 0, \quad i = 1,\dots,p, \tag{4.19c}$$

$$G_i(z)\lambda_i = 0, \quad i = 1,\dots,p, \tag{4.19d}$$

$$\Gamma_{ij}\Gamma_{ik} \geq 0, \quad (j,k) : F_{ij}(z) = F_{ik}(z) = 0. \tag{4.19e}$$

*Proof.* By [29, Theorem 1] there exist nonzero multipliers $(\alpha, u, \lambda)$ such that

$$\alpha \geq 0,$$

$$\lambda \geq 0,$$

$$G_i(z)\lambda_i = 0, \quad i = 1,\dots,p,$$

$$0 \in \alpha\nabla f(z) + \sum_{i=1}^{p} \lambda_i \nabla G_i(z) + \sum_{i=1}^{m} u_i v_i + N_Z(z), \tag{4.20}$$

immediately showing (4.19c) - (4.19d). The vectors $v_i$ in (4.20) are given by $v_i \in \partial \min(F_{i1}(z), \dots, F_{il}(z)) = \text{conv}\{\nabla F_{ij}(z) | j : F_{ij}(z) = 0\}$, cf. [30]. Hence $v_i = \sum_{j=1}^{l} \beta_{ij} \nabla F_{ij}(z)$, with $\beta_{ij} \geq 0$, $\beta_{ij} F_{ij}(z) = 0$ and $\sum_{j=1}^{l} \beta_{ij} = 1$. Let $\Gamma_{ij} = u_i \beta_{ij}$, then (4.19b) is satisfied. Note also that $\Gamma_{ij}$ and $\Gamma_{ik}$ have the same sign (that of $u_i$), thus (4.19e) is also satisfied. Finally, for $\alpha = 0$, (4.20) would contradict (4.9). Therefore, $\alpha > 0$ and by scaling to obtain $\alpha = 1$ in (4.20), (4.19a) is also shown. □

The conditions (4.19a) - (4.19e) are the Clarke stationarity conditions. Clearly, they do not differ from the strong stationarity conditions (4.17a) - (4.17e), aside from (4.17e) which is now replaced by (4.19e). Hence, the Clarke conditions for problem (4.7) are the conditions (4.18a) - (4.18e), together with

$$\Gamma_{ij} \Gamma_{ik} \geq 0, \quad (j,k) : t_{ij} = t_{ik} = 0. \tag{4.18g}$$

The Clarke stationarity conditions are necessary if MFCQ holds. Other than their applicability under distinct constraint qualifications, it is unfortunately not obvious to qualify the difference between a point satisfying the Clarke versus the strong stationarity conditions. A toy problem is presented in [137, Example 3] in which the minimizer satisfies the Clarke but not the strong stationarity conditions.

## 4.3 Results

This section proves that if the sequence of iterates generated by Algorithm 3.1, with $\epsilon^*, \eta^* = 0$, has limit points, then such a limit point along with the corresponding slack variable $t$ from (4.8) satisfies either the strong or the Clarke stationarity conditions of problem (4.7), depending on the assumptions made. The first part of each proof is similar to the proof of [15, Proposition 2.3]. The second part derives the remaining conditions from the definitions of the slack and multiplier variables.

**Remark 4.12.** If the set $\mathcal{X}$ is compact, then the sequence has limit points. This condition is satisfied for instance in the simulation examples, where lower and upper bounds are applied to the control input. However, we do not wish to restrict the theoretical results here to this special case. □

**Theorem 4.13.** *Let $\{x^\nu\}$ be a sequence of iterates generated by Algorithm 3.1, with $\epsilon^*, \eta^* = 0$. Assume that a subsequence $\{x^\nu\}_{\nu \in K}$ converges to a vector $x^*$ such that (4.13) holds at $x^*$, with the set of slack variables $t^*$ defined as in (4.8). Then, point $(x^*, t^*)$ is feasible for (4.7) and unique multipliers can be derived so that $(x^*, t^*, \lambda^*, \Gamma^*)$ satisfies the strong stationarity conditions (4.18a) - (4.18f) of problem (4.7).*

*Proof.* Most of the constraints of problem (4.7) are straightforwardly satisfied in point $(x^*, t^*)$. Step 2 in Algorithm 3.1 gives us $x^\nu \in \mathcal{X}$, and thus because $\mathcal{X}$ is closed also $x^* \in \mathcal{X}$. Moreover, by construction, $t^*_{ij} \geq h_{ij}(x^*)$ is satisfied. The only constraints

remaining to be shown are the complementarity conditions $\min(t_{i1}, \dots, t_{im_i}) = 0$, $i = 1, \dots, m$. This corresponds to showing that for each $i$, $\psi_i(x^\nu) \to 0$, cf. Remark 4.1. Define for all $\nu$

$$\lambda_i^\nu = \mu^\nu \psi_i(x^\nu). \tag{4.21}$$

We have from (4.6) that

$$\begin{aligned}\nabla \mathfrak{L}_{\mu^\nu}(x^\nu) &= \nabla \ell(x^\nu) + \sum_{i=1}^{m} \mu^\nu \psi_i(x^\nu) \nabla \widetilde{\psi}_i(x^\nu) \\ &= \nabla \ell(x^\nu) + \sum_{i=1}^{m} \lambda_i^\nu \nabla \widetilde{\psi}_i(x^\nu) \\ &= \nabla \ell(x^\nu) + \sum_{i \in I_{\overline{O}}(x^\nu)} \lambda_i^\nu \nabla \widetilde{\psi}_i(x^\nu). \end{aligned} \tag{4.22}$$

For the last equality, note that for $i \notin I_{\overline{O}}(x^\nu)$, $\psi_i(x^\nu) = 0$ and thus $\lambda_i^\nu = 0$. Furthermore, there exists some $\nu_1 \in K$, such that for $K \ni \nu \geq \nu_1$, $x^\nu$ will be close enough to $x^*$ and therefore inactive obstacles at $x^*$ will also be inactive at $x^\nu$. Thus, for $K \ni \nu \geq \nu_1$, $I_{\overline{O}}^c(x^*) \subseteq I_{\overline{O}}^c(x^\nu)$, and by applying the complement we obtain $I_{\overline{O}}(x^\nu) \subseteq I_{\overline{O}}(x^*)$. Equation (4.22) for $K \ni \nu \geq \nu_1$ then becomes

$$\nabla \mathfrak{L}_{\mu^\nu}(x^\nu) = \nabla \ell(x^\nu) + \sum_{i \in I_{\overline{O}}(x^*)} \lambda_i^\nu \nabla \widetilde{\psi}_i(x^\nu). \tag{4.23}$$

Equation (4.23) can be substituted in (3.6) for $\nu \geq \nu_1$, yielding:

$$e^\nu \in \nabla \ell(x^\nu) + \sum_{i \in I_{\overline{O}}(x^*)} \lambda_i^\nu \nabla \widetilde{\psi}_i(x^\nu) + N_{\mathcal{X}}(x^\nu). \tag{4.24}$$

Since $\{x^\nu\}_{\nu \in K}$ converges, it is a bounded sequence, and so $\{\nabla \ell(x^\nu)\}_{\nu \in K}$ is also bounded. Given that $e^\nu$ is also bounded explicitly by (3.6), this results in boundedness of $\sum_{i \in I_{\overline{O}}(x^*)} \lambda_i^\nu \nabla \widetilde{\psi}_i(x^\nu)$ for $\nu \in K$. Indeed, if this sequence were not bounded, then the terms $e^\nu$ and $\nabla \ell(x^\nu)$ would become negligible in (4.24), which would then in the limit contradict (4.13). Furthermore, (4.13) also implies that for $\nu \in K$ large enough, the gradients $\nabla \widetilde{\psi}_i(x^\nu)$, $i \in I_{\overline{O}}(x^*)$ are linearly independent, thus we can solve (4.24) for the active multipliers

$$\lambda^\nu = [\nabla \widetilde{\psi}(x^\nu)^\top \nabla \widetilde{\psi}(x^\nu)]^{-1} \nabla \widetilde{\psi}(x^\nu)^\top (e^\nu - \nabla \ell(x^\nu) - w^\nu)),$$

with $w^\nu \in N_{\mathcal{X}}(x^\nu)$ satisfying (4.24). As $e^\nu \to 0$, it follows that

$$\{\lambda^\nu\}_K \to \overline{\lambda} = -[\nabla \widetilde{\psi}(x^*)^\top \nabla \widetilde{\psi}(x^*)]^{-1} \nabla \widetilde{\psi}(x^*)^\top (\nabla \ell(x^*) + w^*).$$

Hence, all $\overline{\lambda}_i$ have to be bounded and consequently, recalling (4.21) and because

$\mu^\nu \to \infty$, all $\psi_i(x^*)$ must be equal to 0. Therefore, the point $(x^*, t^*)$ is feasible for problem (4.7). Moreover, (4.24) can be written in the limit as

$$0 \in \nabla \ell(x^*) + \sum_{i \in I_{\overline{O}}(x^*)} \overline{\lambda}_i \nabla \widetilde{\psi}_i(x^*) + N_{\mathcal{X}}(x^*). \tag{4.25}$$

As (4.13) holds in $x^*$, it follows from Lemma 4.7 that the point $(x^*, t^*)$ satisfies MPCC-LICQ. Furthermore, from (4.14) and (4.8), strict complementarity holds at this point. Next, we show that the strong stationarity conditions are satisfied in this point. The multipliers for the reformulated problem can be retrieved as follows:

$$\lambda^*_{ij} = \overline{\lambda}_i \prod_{k \neq j} [h_{ik}(x^*)]_+, \tag{4.26}$$

$$\Gamma^*_{ij} = -\lambda^*_{ij} = -\overline{\lambda}_i \prod_{k \neq j} [h_{ik}(x^*)]_+. \tag{4.27}$$

Consider from the discussion above that for the inactive obstacles $i \notin I_{\overline{O}}(x^*)$, the multipliers $\overline{\lambda}_i = 0$. Thus, from the definition of the multipliers (4.26) and (4.27) it follows that

$$\lambda^*_{ij} = \Gamma^*_{ij} = 0, \quad i \notin I_{\overline{O}}(x^*). \tag{4.28}$$

Furthermore, as shown in the proof of Lemma 4.7, for each active obstacle there is exactly one active obstacle boundary. Let $j_i$ denote the index for which for each $i \in I_{\overline{O}}(x^*)$, $h_{ij_i}(x^*) = 0$. Then, from (4.26), for $k \neq j_i$,

$$\lambda^*_{ik} = \lambda^*_{ij_i} \frac{[h_{ij_i}(x^*)]_+}{[h_{ik}(x^*)]_+} = 0, \quad (i,k) \notin I_a(x^*). \tag{4.29}$$

Equations (4.14), (4.25), (4.26), (4.28) and (4.29) combined show that (4.18a) is satisfied at $x^*$. The remaining strong stationarity conditions follow from the definition of the multipliers (4.26), (4.27) and the slack variables (4.8). By choice of the multipliers $\Gamma^*_{ij}$ in (4.27), (4.18b) is trivially satisfied. Considering (4.28), condition (4.18c) is trivially satisfied for $i \notin I_{\overline{O}}(x^*)$. For $i \in I_{\overline{O}}(x^*)$, as $t^*_{ij} = h_{ij}(x^*)$ and using (4.27) and (4.26), condition (4.18c) is transformed to

$$t^*_{ij} \Gamma^*_{ij} = -\overline{\lambda}_i \psi_i(x^*), \quad i \in I_{\overline{O}}(x^*).$$

As shown above, $\psi_i(x^*) = 0$ for all $i$, thus condition (4.18c) is satisfied. By construction of the multipliers $\lambda^*_{ij}$ in (4.26) and given $\overline{\lambda}_i$ is positive due to (4.21), condition (4.18d) is also satisfied. Condition (4.18e) is trivially satisfied for $i \notin I_{\overline{O}}(x^*)$ because of (4.28). For $i \in I_{\overline{O}}(x^*)$, $t^*_{ij} = h_{ij}(x^*)$ and (4.18e) is thus also trivially satisfied for this case. Finally, condition (4.18f) does not apply in this case as strict complementarity holds because of (4.14) and definition (4.8). □

Limit points where more than one boundary of the same obstacle is active are excluded by the assumption used in Theorem 4.13. Such a limit point, however, can still be shown

to satisfy the Clarke conditions (4.18a) - (4.18e) and (4.18g) under the assumption that this point is feasible and (4.15) holds. Note that in this case strict complementarity does not (necessarily) hold, as multiple active obstacle boundaries translate into multiple indices $j$ for which $t_{ij} = 0$, according to the first case of (4.8).

**Theorem 4.14.** *Let $\{x^\nu\}$ be a sequence of iterates generated by Algorithm 3.1, with $\epsilon^*, \eta^* = 0$. Assume that a subsequence $\{x^\nu\}_{\nu \in K}$ converges to a vector $x^*$ feasible for problem (3.1), such that (4.15) holds at $x^*$. Define the set of slack variables $t^*$ as in (4.8). Then, point $(x^*, t^*)$ satisfies the Clarke stationarity conditions, (4.18a) - (4.18e) and (4.18g) of problem (4.7).*

*Proof.* Given the point $x^*$ is feasible for problem (3.1), it follows that $\forall i : \psi_i(x^*) = 0$, which implies feasibility of $(x^*, t^*)$ for problem (4.7), cf. Remark 4.1, and also that for every $i$, there is at least one index $j$ for which $[h_{ij}(x^*)]_+ = 0$. Define for all $\nu$:

$$\lambda_i^\nu = \mu^\nu \psi_i(x^\nu),$$

$$\lambda_{ij}^\nu = \lambda_i^\nu \prod_{l \neq j} [h_{il}(x^\nu)]_+.$$

Along the same reasoning as in the proof of Theorem 4.13, there exists some $\nu_1 \in K$, such that for $K \ni \nu \geq \nu_1$, $I_{\overline{O}}(x^\nu) \subseteq I_{\overline{O}}(x^*)$. The multipliers belonging to inactive obstacles will then be equal to zero for $K \ni \nu \geq \nu_1$. Furthermore, in the limit, the multipliers for the inactive obstacle boundaries will become negligible against the multipliers for the active ones. Let $j$ denote an index for which $[h_{ij}(x^*)]_+ = 0$, and $l$ an index for which $[h_{il}(x^*)]_+ > 0$. Then

$$\lim_{\substack{\nu \to \infty \\ K}} \frac{\lambda_{il}^\nu}{\lambda_{ij}^\nu} = \lim_{\substack{\nu \to \infty \\ K}} \frac{[h_{ij}(x^\nu)]_+}{[h_{il}(x^\nu)]_+} = 0. \tag{4.30}$$

The assumption (4.15) implies that the gradients $\nabla h_{ij}(x^*), (i,j) \in I_a(x^*)$ are linearly independent and thus different from the zero vector. Then, as a result of (4.30), terms in (4.24) with $(i,j) \notin I_a(x^*)$ will become negligible for large $\nu \in K$. Thus, (4.24) reduces in this case to

$$e^\nu \in \nabla \ell(x^\nu) + \sum_{(i,j) \in I_a(x^*)} \lambda_{ij}^\nu \nabla h_{ij}(x^\nu) + N_{\mathcal{X}}(x^\nu).$$

Since $e^\nu \to 0$, it follows that we can solve for the active multipliers and take the limit

$$\{\lambda^\nu\}_K \to \overline{\lambda} = -[\nabla h(x^*)^\top \nabla h(x^*)]^{-1} \nabla h(x^*)^\top (\nabla \ell(x^*) + w^*).$$

Thus the active multipliers are bounded. As a result, (4.30) shows that the inactive multipliers will be zero. By defining the optimal multipliers using (4.26) - (4.27), the remainder of the proof is analogous to the proof of Theorem 4.13, with $I_a(x^*)$ replacing $I_{\overline{O}}(x^*)$ and of course with the exception of showing condition (4.18g). This condition is however readily shown by recognizing that all multipliers $\Gamma_{ij}^*$ are nonpositive, so the product of two of them will always be nonnegative. □

**Remark 4.15.** The result here mirrors the second special case of Theorem 3.2 in [89]. The authors consider an augmented Lagrangian method applied to the problem with complementarity constraints, and find that the assumption of the accumulation point being feasible and MPCC-LICQ holding, for unbounded penalty parameters, results in this point satisfying the Clarke stationarity conditions. □

## 4.4 Summary

This chapter presented convergence results for the quadratic penalty method applied to problem (3.1). Given the presence of nonsmooth constraints and the resulting elusiveness of stationarity conditions for such a problem, the analysis constructed instead an equivalent mathematical program with vertical complementarity constraints. For the latter, the literature provides us with constraint qualifications and stationarity conditions, which were extended here to include abstract set constraints. The quadratic penalty method was then shown to produce iterates the limit points of which, if they exist, satisfy one of two stationarity conditions of the MPCC, depending on the assumptions made regarding the constraint gradients.

# Chapter 5

# Penalty Method for MPSEC: Numerical results

This chapter discusses numerical results for Algorithm 3.1 applied to a number of obstacle configurations. Two different nonlinear kinematic models are considered, a vehicle with a trailer and a bicycle model. Our method is shown to be versatile in finding trajectories from the start to the destination, especially so using the additional heuristics from Section 3.3. The numerical performance of Algorithm 3.1 is additionally compared against several state-of-the-art NLP solvers, IPOPT [161], SNOPT [69] and KNITRO [24].

Section 5.1 derives the optimal control problem formulations that are considered in this work. For this purpose, the continuous-time vehicle dynamics of the two models are discretized, and different formulations regarding single shooting, multiple shooting, and the problem with complementarity constraints are derived. Section 5.2 presents the numerical simulation results, including a number of obstacle configuration examples and a performance comparison with state-of-the-art NLP solvers.

The material in this chapter is based on the publications [82, 83].

## 5.1 Optimal control problem

Let us recall the optimal control problem in multiple shooting form (2.1) (disregarding additional state constraints)

$$\underset{(q,u)\in\mathbb{R}^{(N+1)n_q+Nn_u}}{\textbf{minimize}} \quad \ell(q,u) \tag{5.1a}$$

$$\textbf{subject to} \quad q_{k+1} = \varphi_k(q_k, u_k), \quad k = 0, \dots, N-1, \tag{5.1b}$$

$$q_0 = \bar{q}, \tag{5.1c}$$

$$u_k \in \mathcal{U}_k, \qquad k = 0, \dots, N-1, \tag{5.1d}$$

$$q_k \notin O_j, \qquad k = 1, \dots, N, \quad j = 1, \dots, N_O. \tag{5.1e}$$

The objective is given as a quadratic penalty with respect to the reference state $q_{\text{ref}}$ and input $u_{\text{ref}}$, see (2.2). The input constraints are simple box constraints, representing an upper and a lower limit on the actuation, $\mathcal{U}_k = [u_{\min}, u_{\max}]$. What remains now is to discuss the dynamics (5.1b) and the obstacle avoidance constraints (5.1e).

### 5.1.1 Vehicle dynamics

This work considers MPC control for two types of vehicles: a kinematic bicycle model [129, p. 26] and a vehicle with a trailer, illustrated in Figure 5.1. Usually, we use the simplest model for the bicycle where slip of the wheels is neglected, which is defined by three states $q(t) = (q_x(t), q_y(t), \theta(t))$, the horizontal and vertical position of the center and its heading angle. It accepts two control inputs, $u(t) = (v(t), \delta(t))$, the velocity and the steering angle of the front wheel(s). The continuous kinematics relating $u(t)$ to $\dot{q}(t)$ are given by

$$\dot{q}_x = v \cdot \cos(\theta)$$

$$\dot{q}_y = v \cdot \sin(\theta)$$

$$\dot{\theta} = \frac{v}{L} \tan(\delta).$$

Here, $L = 0.5$m is the distance between the centers of mass of the wheels of the bicycle. For one scenario below, which was presented in a later paper, the bicycle is instead modeled as a system with 4 states, $q(t) = (q_x(t), q_y(t), v(t), \theta(t))$, a horizontal position, a vertical position, a velocity and a heading angle. This *extended* version of the bicycle accepts two control inputs, $u(t) = (a(t), \delta(t))$, the acceleration and the steering angle of the front wheel(s). The continuous kinematics relating $u(t)$ to $\dot{q}(t)$ are given by

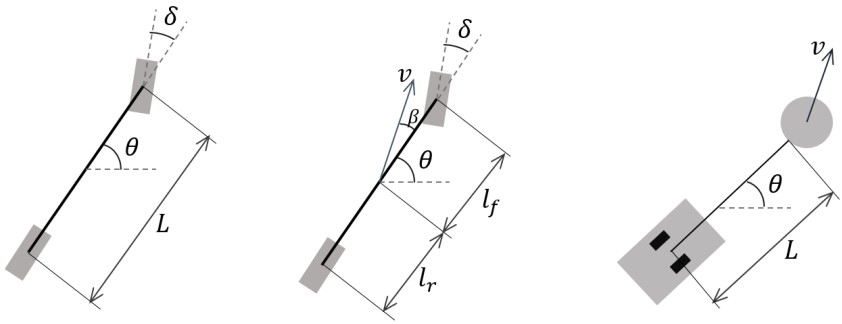
$$\dot{q}_x = v \cdot \cos(\theta + \beta)$$

$$\dot{q}_y = v \cdot \sin(\theta + \beta)$$

$$\dot{\theta} = \frac{v}{l_r} \sin(\beta)$$

$$\dot{v} = a,$$

where the slip angle $\beta = \tan^{-1}(\frac{l_r}{l_r + l_f} \tan(\delta))$, and $l_r = 1.17$m and $l_f = 1.77$m are the distances from the center of mass to the rear and front wheel respectively.




**(a)** *Simple and extended bicycle model.* **(b)** *Trailer model.*

**Figure 5.1:** *Vehicle models used in the simulations.*

The trailer is modeled as a system with 3 states, $q(t) = (q_x(t), q_y(t), \theta(t))$, a horizontal position, a vertical position and a heading angle. It is controlled via two control inputs, $u(t) = (v_x(t), v_y(t))$, the horizontal and vertical velocity of the towing vehicle. The continuous kinematics relating $u(t)$ to $\dot{q}(t)$ are given by

$$\dot{q}_x = v_x + L\sin(\theta) \cdot \dot{\theta}$$

$$\dot{q}_y = v_y - L\cos(\theta) \cdot \dot{\theta}$$

$$\dot{\theta} = \frac{1}{L}(v_y\cos(\theta) - v_x\sin(\theta)),$$

with $L = 0.5$m the distance between the trailer's center of mass and the fulcrum connecting to the towing vehicle.

In the above models, the continuous kinematics are described by a relationship between the derivative of the state with respect to time and the input and state of the vehicle. Let $f : \mathbb{R}^{n_u+n_q} \to \mathbb{R}^{n_q}$ denote this relationship, such that $\dot{q}(t) = f(u, q)$. The kinematics of both models are then discretized using an explicit fourth-order Runge Kutta method, chosen for its favorable tradeoff between accuracy and efficiency. As such, the equality of (5.1b) becomes

$$q_{k+1} = \varphi_k(q_k, u_k) = q_k + \frac{1}{6}(k_1 + k_2 + k_3 + k_4)$$

where

$$\begin{cases} k_1 = hf(u_k, q_k), \\ k_2 = hf(u_k, q_k + \frac{k_1}{2}), \\ k_3 = hf(u_k, q_k + \frac{k_2}{2}), \\ k_4 = hf(u_k, q_k + k_3), \end{cases}$$

with $h = \frac{T}{N}$ the sampling time.

### 5.1.2 OCP formulations

With the dynamics given by the previous subsection, the only remaining constraints to be clarified are the obstacle avoidance constraints (5.1e). Given the obstacle formulation of Section 2.3.5, the OCP in multiple shooting formulation can be written as

$$\begin{aligned}
\underset{(q,u)\in\mathbb{R}^{(N+1)n_q+Nn_u}}{\textbf{minimize}} \quad & \ell(q,u) && && (5.2a)\\
\textbf{subject to} \quad & q_{k+1} = \varphi_k(q_k,u_k), && k=0,\dots,N-1, && (5.2b)\\
& q_0 = \bar{q}, && && (5.2c)\\
& u_k \in \mathcal{U}_k, && k=0,\dots,N-1, && (5.2d)\\
& \psi_i(q_k) = 0, && k=1,\dots,N, \quad i=1,\dots,N_O. && (5.2e)
\end{aligned}$$

Substituting the dynamics as done in Section 2.3.1, the equivalent single shooting formulation, with $\ell(u) \leftarrow \ell(\Phi(u),u)$, is obtained as

$$\begin{aligned}
\underset{u\in\mathcal{U}}{\textbf{minimize}} \quad & \ell(u) && && (5.3a)\\
\textbf{subject to} \quad & \psi_i(\Phi_k(u)) = 0, && i=1,\dots,N_O, \quad k=1,\dots,N. && (5.3b)
\end{aligned}$$

This problem can readily be recognized as one of type (3.1), after renaming $u$ to $x$, and reshuffling the obstacle cost functions in (5.3b) as $\psi_i(\cdot) \leftarrow \psi_i(\Phi_k(\cdot))$, with $i \leftarrow N(i-1)+k$ and $m \leftarrow N \cdot N_O$. Since the $\Phi_k$ are smooth functions, the chain rule can be applied when computing derivatives. As such, we can use Algorithm 3.1 for this problem and the convergence results in Theorems 4.13 and 4.14 readily apply, given the appropriate assumptions, now on $\psi_i(\Phi_k(\cdot))$, hold. However, the other solvers are not straightforwardly able to deal with this problem, due to the nonsmooth constraints. Therefore, also a smooth variant with each obstacle cost function squared is considered,

$$\begin{aligned}
\underset{u\in\mathcal{U}}{\textbf{minimize}} \quad & \ell(u) && && (5.4a)\\
\textbf{subject to} \quad & \psi_i^2(\Phi_k(u)) = 0, && i=1,\dots,N_O, \quad k=1,\dots,N. && (5.4b)
\end{aligned}$$

This problem still poses theoretical difficulties, since according to (3.2) the gradient of $\psi_j^2$ will be zero at the boundary. Hence constraint qualifications are again a problem. Nevertheless, (5.4) can at least be provided to the NLP solvers. These solvers rely on

interior-point or sequential quadratic programming methods, and so have their unique ways of dealing with constraints. Unlike Algorithm 3.1, these solvers do not require the problem to be in the single shooting formulation. They can therefore also solve the multiple shooting formulation in the comparison

$$\begin{aligned}
\underset{(q,u)\in\mathbb{R}^{(N+1)n_q}\times\mathcal{U}}{\textbf{minimize}} \quad & \ell(q,u) && \text{(5.5a)}\\
\textbf{subject to} \quad & q_{k+1}=\varphi_k(q_k,u_k), \quad k=0,\dots,N-1, && \text{(5.5b)}\\
& q_0=\bar{q}, && \text{(5.5c)}\\
& \psi_i^2(q_k)=0, \qquad i=1,\dots,N_O, \quad k=1,\dots,N. && \text{(5.5d)}
\end{aligned}$$

Finally, to exhaust all options, also the equivalent problem with complementarity constraints is considered

$$\begin{aligned}
\underset{(u,q,t)\in\mathcal{U}\times\mathbb{R}^{(N+1)n_q+n_t}}{\textbf{minimize}} \quad & \ell(q,u), && \text{(5.6a)}\\
\textbf{subject to} \quad & q_{k+1}=\varphi_k(q_k,u_k), \qquad k=0,\dots,N-1, && \text{(5.6b)}\\
& q_0=\bar{q}, && \text{(5.6c)}\\
& t_{ijk}\geq h_{ij}(q_k), \quad i=1,\dots,N_O, \quad j=1,\dots,m_i, \quad k=1,\dots,N && \text{(5.6d)}\\
& \min(t_{i1},\cdots,t_{im_i})_k=0, \quad i=1,\dots,N_O, \quad k=1,\dots,N. && \text{(5.6e)}
\end{aligned}$$

As mentioned in Section 4.1.1, the complementarity constraints have an NLP version. In this case, (5.6e) would be transformed into $t_{ijk}\geq 0$ and $\sum_{i=1}^{N_O}\sum_{k=1}^{N}\prod_{j=1}^{m_i} t_{ijk}=0$. The latter equality constraint can also be replaced by $\sum_{i=1}^{N_O}\sum_{k=1}^{N}\prod_{j=1}^{m_i} t_{ijk}\leq 0$, since all the slack variables are constrained to be positive. The inequality formulation seems to work slightly better in practice, so it is adopted here. The problem with complementarity constraints, therefore, in multiple shooting formulation is the following

$$\begin{aligned}
\underset{(u,q,t)\in\mathcal{U}\times\mathbb{R}^{(N+1)n_q+n_t}}{\textbf{minimize}} \quad & \ell(q,u), && \text{(5.7a)}\\
\textbf{subject to} \quad & q_{k+1}=\varphi_k(q_k,u_k), \quad k=0,\dots,N-1, && \text{(5.7b)}\\
& q_0=\bar{q}, && \text{(5.7c)}
\end{aligned}$$

$$t_{ijk} \geq h_{ij}(q_k), \qquad i=1,\ldots,N_O, \;\; j=1,\ldots,m_i, \;\; k=1,\ldots,N \tag{5.7d}$$

$$t_{ijk} \geq 0, \qquad i=1,\ldots,N_O, \;\; j=1,\ldots,m_i, \;\; k=1,\ldots,N \tag{5.7e}$$

$$\sum_{i=1}^{N_O} \sum_{k=1}^{N} \prod_{j=1}^{m_i} t_{ijk} \leq 0. \tag{5.7f}$$

In the single shooting formulation, this becomes

$$\underset{(u,t)\in\mathcal{U}\times\mathbb{R}^{n_t}}{\textbf{minimize}} \quad \ell(u), \tag{5.8a}$$

$$\textbf{subject to} \quad t_{ijk} \geq h_{ij}(\Phi_k(u)), \quad i=1,\ldots,N_O, \;\; j=1,\ldots,m_i, \;\; k=1,\ldots,N \tag{5.8b}$$

$$t_{ijk} \geq 0 \qquad i=1,\ldots,N_O, \;\; j=1,\ldots,m_i, \;\; k=1,\ldots,N \tag{5.8c}$$

$$\sum_{i=1}^{N_O} \sum_{k=1}^{N} \prod_{j=1}^{m_i} t_{ijk} \leq 0. \tag{5.8d}$$

Problems (5.7) and (5.8) can immediately be fed into an NLP solver. Furthermore, KNITRO [24] has an additional feature to deal with complementarity constraints. However, this feature only accepts two lists of variables which are complementary to each other. Therefore, the complementarity constraints (5.6e) need to be reformed for any obstacle with more than two boundaries (and associated slack variables). A complementarity constraint with $m > 2$ decision variable components, $\min(t_1, \ldots, t_m) = 0$, can be transformed through the introduction of additional slack variables into complementarity constraints with two components and (linear) equality constraints. For instance, for three complementarity components we require three additional slack variables $(z_1, z_2, z_3)$ and end up with two complementarity and equality constraints:

$$\begin{aligned}
\min(t_1,t_2,t_3) = 0 &\Leftrightarrow \begin{cases} \min(t_1,z_1) = 0 \\ z_1 = \min(t_2,t_3) \end{cases} \\
&\Leftrightarrow \begin{cases} \min(t_1,z_1) = 0 \\ \min(t_2 - z_1, t_3 - z_1) = 0 \end{cases} \\
&\Leftrightarrow \begin{cases} \min(t_1,z_1) = 0 \\ \min(z_2,z_3) = 0 \\ z_2 = t_2 - z_1 \\ z_3 = t_3 - z_1. \end{cases}
\end{aligned}$$

Clearly, the first step in this procedure introduces a slack $z_1$ and sets it equal to $\min(t_2, \ldots, t_m)$. As such, the result is a constraint with two components, and one with $m-1$. The procedure is therefore recursive, requiring $m-2$ slacks to be introduced to bring the number of complementarity components down to 2, and $m-1$ slacks to transform these components again into decision variables. In total, therefore, $2m-3$ additional slack variables are required, and we end up with $m-1$ complementarity constraints with two components and linear equality constraints. Thus, the constraints of (5.6e) require a total of $N\sum_{i:m_i>2}(2m_i-3)$ additional slack variables. For obstacles with a high number of obstacle boundaries, therefore, this procedure will inevitably increase the number of decision variables significantly.

## 5.2 Simulation results

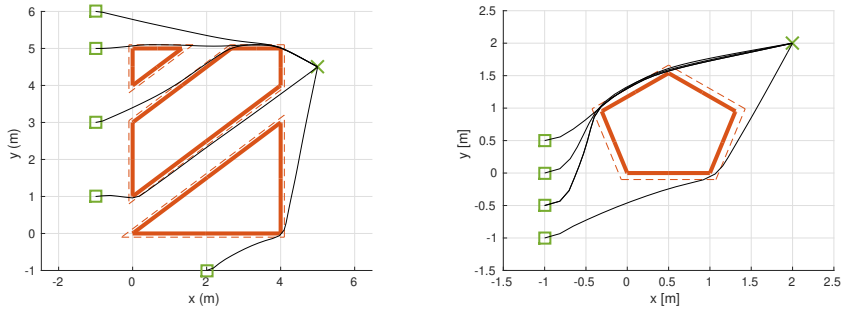
This section shows how an implementation of the proposed method in C, with an interface to MATLAB for the problem generation, is versatile in modeling complex obstacle shapes and efficiently calculating trajectories for the bicycle and trailer models. The first subsection provides a selection of obstacle configurations which are successfully avoided by the MPC controller. The second subsection provides a runtime comparison between Algorithm 3.1 and IPOPT, SNOPT and KNITRO applied to the OCP formulations of Section 5.1.2. All simulations were performed on a notebook with Intel(R) Core(TM) i7- 7600U CPU @ 2.80GHz x 2 processor and 16 GB of memory. For all simulations, the automatic differentiation (AD) package CasADi [4] was used to efficiently evaluate the objective and constraint functions and gradients.

### 5.2.1 Obstacle configurations

The legend of the figures in this section is always the same: the obstacles are given in red lines, their enlargements in red dashed lines (sometimes not visible because of scaling), the green cross represents the destination, the green squares denote different starting points, and the resulting trajectories are drawn in black. The optimal control problems are solved with a horizon length $N = 50$, and sampling time variably chosen such that the vehicle should be able to reach the destination given the actuator limits. The inputs of the bicycle are constrained by box constraints $-0.1\text{m/s} \leq v \leq 4\text{m/s}$ and $-\frac{\pi}{3} \leq \delta_f \leq \frac{\pi}{3}$, and those of the trailer by $-4\text{m/s} \leq v_x, v_y \leq 4\text{m/s}$, at every time instant. It is usually difficult for a first order method, such as PANOC, to find a solution for a strict tolerance, say $10^{-6}$. Therefore, the tolerance $\epsilon^*$ is set to $10^{-3}$. We observed in simulations that the closed-loop performance is not impacted by this choice, and that a stricter tolerance would be unnecessary. For the obstacle cost function, we set a tolerance $\eta^* = 10^{-2}$.

Figure 5.2 displays two polygonal obstacle configurations which are avoided by the bicycle. The sampling time for both of these examples is $t_s = 50\text{ms}$. The MATLAB interface allows the user to specify the vertices of a polygonal obstacle and computes

the boundary functions automatically using vert2con.[1] A polygonal obstacle is written as $O = \{Aq < b\}$, and to avoid the effects of ill conditioning, the rows of A are normalized so that they have unit norm.




**(a)** *Polygonal obstacles.* **(b)** *Pentagon obstacle.*

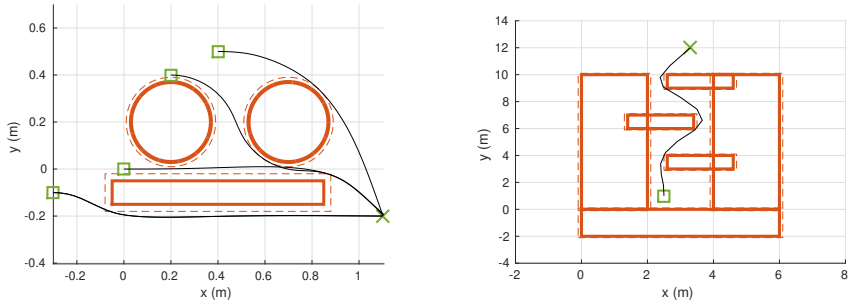
**Figure 5.2:** *The MPC controller steers the kinematic bicycle through two environments with polygonal obstacles.*

Figure 5.3 presents two configurations with circular and rectangular obstacles. The bicycle overcomes the arrangement of Figure 5.3a, with a sampling time $t_s = 30$ms, and the vehicle with trailer is applied to the situation in Figure 5.3b with $t_s = 200$ms. Circular and rectangular obstacles are fairly straightforward to incorporate in our obstacle formulation. They are given respectively by $O = \{(q_x - c_x)^2 + (q_y - c_y)^2 < r^2\}$, with $(c_x, c_y)$ the coordinates of the center and $r$ the radius of the circle, and by $O = \{l_x < q_x < u_x, l_y < q_y < u_y\}$.

Figure 5.4 provides instead some novelty in the form of obstacle with nonlinear boundary functions. As far as we know, it is impossible for any of the other approaches discussed in Section 2.3 to straightforwardly incorporate these obstacles, given their nonlinearity and furthermore their nonconvexity. The obstacles in Figure 5.4a are given by $O_1 = \{(q_x, q_y) : 0 < q_x < 5, -2 < q_y < 2 + \frac{3}{2}\sin(\frac{2\pi}{5}q_x)\}$ and $O_2 = \{(q_x, q_y) : 0 < q_x < 5, 4 + \frac{3}{2}\sin(\frac{2\pi}{5}q_x) < q_y < 8\}$, and are avoided by the extended bicycle with $t_s = 50$ms. The (enlarged) obstacle of Figure 5.4b is given by $O = \{(q_x, q_y) : q_y > q_x^2 - 1, q_y < \frac{q_x^2}{2}\}$ and is avoided by the trailer vehicle with $t_s = 30$ms.

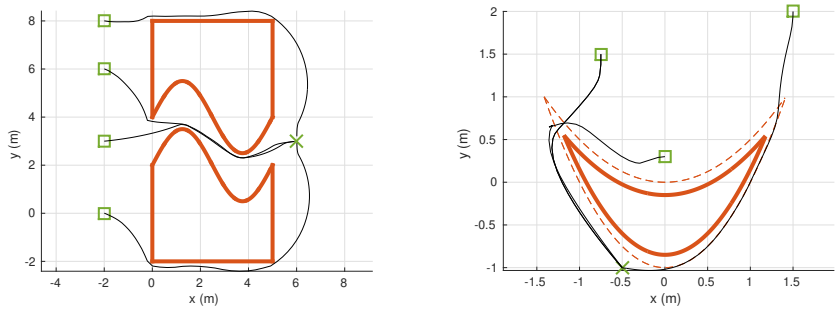
Figure 5.5 finally illustrates two obstacle configurations for which the graph search heuristic of Section 3.3 proved necessary to find a trajectory that reached the destination. These arrangements were overcome by the vehicle with trailer, both with a sampling time of $t_s = 30$ms. As in Figure 3.3, the magenta diamonds represent the solution of the graph search and the black diamonds are the intermediate destinations visited successfully by the MPC controlled trailer. The grids are of size 15 by 15, (manually) centered around the obstacle, and scaled such that the outer layer consists

[1] https://nl.mathworks.com/matlabcentral/fileexchange/7895-vert2con-vertices-to-constraints




**(a)** *Circular and rectangular obstacles.* **(b)** *Rectangular obstacles.*

**Figure 5.3:** *The MPC controller steers the kinematic bicycle (left) and the trailer (right) through an environment with rectangular and circular obstacles.*



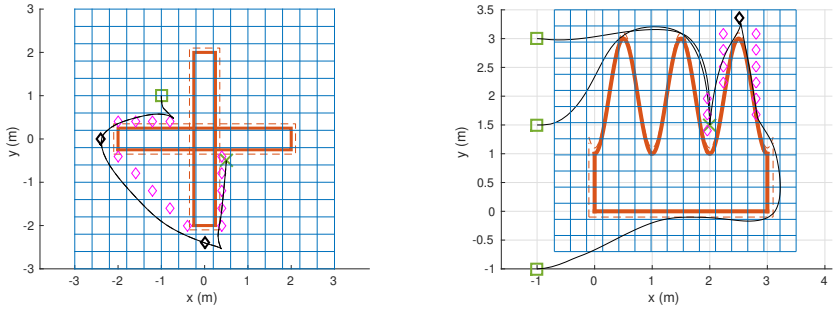

**(a)** *S-shaped corridor.* **(b)** *Crescent-shaped obstacle.*

**Figure 5.4:** *The MPC controller steers the kinematic bicycle (left) and the trailer (right) through an obstructed environment. The obstacles in these examples each include one or more nonlinear boundary functions.*

completely of free cells. Moreover, the starting and destination points are checked to be inside the grid. In this way, the graph search should always be able to find a solution, as confirmed in the figures.

### 5.2.2 Performance comparison

Table 5.1 shows a comparison in terms of runtime and objective value of the proposed method with solvers that address the equivalent problem with complementarity constraints, for 50 instances of the OCP, each with a random initial state. The horizon




**(a)** *Cross-shaped obstacle.* **(b)** *Rack-shaped obstacle.*

**Figure 5.5:** *The MPC controller steers the trailer through two environments with clear local minima, for which the heuristic from Section 3.3 proves effective.*

length was restricted to 30 for this example. The parameters used in Algorithm 3.1 were: $x^0 = u^0 = 0$, $\mu^1 = 100$, $\mu^* = 10^6$, $\omega = 10$, $\epsilon^0 = 1$, $\epsilon^{\nu+1} = 10^{-1}\epsilon^\nu$, $\epsilon^* = 10^{-3}$ and $\eta^* = 10^{-2}$. Aside from our method relying on PANOC, the solvers were tested for both a single-shooting (SS) and a multiple-shooting (MS) formulation. In the case of SNOPT and IPOPT, the problems solved are (5.7) with MS and (5.8) with SS. For KNITRO, these problems are reformulated to only contain a list of complementary variables, as outlined in Section 5.1.2.

Fail percentage indicates the frequency of not converging to an approximate local minimum, either not converging at all or converging to a point of local infeasibility. In those cases, the heuristics of Section 3.3 can be applied, but they are not included in the current simulations. The solvers are compared in terms of runtime and objective function value. No time was measured during problem construction, and only the reported runtimes by the solvers are presented. The performance in terms of objective values for each solver $s_i$ is reported as the relative difference between the objective value of the optimal input sequence for that solver $u^{*,s_i}$, and the best solution found by all solvers for each instance, i.e. $\frac{\ell(u^{*,s_i})-\ell^*}{\ell^*}$, with $\ell^* = \min_{s\in S}\{\ell(u^{*,s})\}$, with $S$ the set of all solvers. This measure indicates the quality of the solution, which is interesting as problems with set exclusion constraints are highly nonconvex and therefore many local minima exist. Typically, because of the large terminal cost, trajectories where the vehicle remains in place behind an obstacle are local minima with a higher objective value than trajectories that find their way to the destination. Hence, the proposed method, which exhibits the lowest average and maximum differences, is shown to be effective at finding meaningful local minima for these types of path planning problems.

Furthermore, Table 5.1 indicates that the proposed method outperforms the other methods in terms of success percentage and in terms of runtime. The problem with complementarity constraints is also inherently much larger as it contains more decision variables and constraints. Especially for the formulation used by KNITRO, a lot of

additional slack variables and constraints are introduced, as it requires two index lists denoting which variables are complementary to each other. The runtime discrepancies are (at least partially) due to the solvers working on different constraints.

**Table 5.1:** *Performance comparison for the MPC controlled bicycle of Figure 5.4a, with solvers other than Algorithm 3.1 working on the problem(s) with complementarity constraints.*

| Solver | Runtime (s) | | $\frac{\ell(u^{*,s_i})-\ell^*}{\ell^*}$ | | Fail (%) |
|---|---|---|---|---|---|
| | Avg | Max | Avg | Max | |
| Algorithm 3.1 (SS) | 0.017 | 0.115 | 0.19 | 1.12 | 8 |
| IPOPT (SS) | 1.731 | 7.776 | 0.25 | 1.24 | 14 |
| IPOPT (MS) | 1.604 | 4.448 | 0.38 | 8.55 | 16 |
| SNOPT (SS) | 9.320 | 4.335 | 0.29 | 2.02 | 16 |
| SNOPT (MS) | 1.153 | 3.054 | 0.42 | 1.43 | 78 |
| KNITRO (SS) | 4.391 | 13.73 | 0.64 | 10.9 | 38 |
| KNITRO (MS) | 14.13 | 19.78 | 0.60 | 1.42 | 74 |

It is, therefore, also interesting to consider another comparison between our method and SNOPT, IPOPT and KNITRO applied to the original problem with obstacle avoidance as equality constraints $\psi_i^2(x) = 0$, see problems (5.4) and (5.5). This squared version of the equality constraint is differentiable, but will not satisfy LICQ. The example considered here is that of Figure 5.4b, with a horizon length of 50 and sampling time of 30ms. The runtime, objective and fail percentage results are presented in Table 5.2. KNITRO is not applied to the single shooting formulation here as the generated files were too large to compile. The proposed method again outperforms or is at least competitive against other state-of-the-art solvers for this problem.

**Table 5.2:** *Performance comparison for the MPC controlled trailer of Figure 5.4b, with solvers other than Algorithm 3.1 working on problems (5.4) and (5.5).*

| Solver | Runtime (s) | | $\frac{\ell(u^{*,s_i})-\ell^*}{\ell^*}$ | | Fail (%) |
|---|---|---|---|---|---|
| | Avg | Max | Avg | Max | |
| Algorithm 3.1 (SS) | 0.067 | 0.447 | 0.30 | 1.87 | 0 |
| IPOPT (SS) | 0.776 | 2.821 | 0.45 | 3.45 | 0 |
| IPOPT (MS) | 1.183 | 2.659 | 0.58 | 3.93 | 2 |
| SNOPT (SS) | 0.098 | 0.608 | 0.37 | 2.38 | 2 |
| SNOPT (MS) | 0.366 | 3.462 | 0.82 | 3.86 | 8 |
| KNITRO (MS) | 4.398 | 11.42 | 0.47 | 4.26 | 14 |

Finally, Figure 5.6 shows the runtime for Algorithm 3.1, IPOPT and SNOPT run on the single-shooting formulation of (5.4). The figure clearly demonstrates that Algorithm 3.1 benefits from warm starting the next OCP using the (shifted) solution

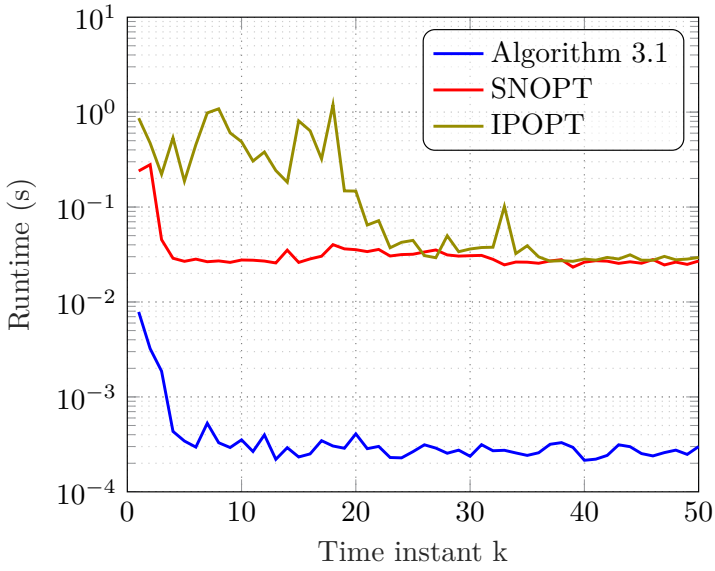



**Figure 5.6:** *Runtime comparison in an MPC simulation of Figure 5.4b.*

of the previous one. This is an essential aspect of a solver to be used in MPC, since subsequent OCPs are often very similar. It is well known that interior-point methods, the class to which IPOPT belongs, typically do not benefit much from a good initial guess. This is also confirmed in Figure 5.6. SNOPT, an SQP method, instead does benefit from warm starting, as expected. Note that around time instance 20, the vehicle had passed by the obstacle and so the subsequent OCPs were comparatively easier, hence the drop in runtime of IPOPT after that instant.

## 5.3 Summary

This chapter presented numerical simulation results for Algorithm 3.1 applied to various obstacle configurations. MPC control was applied to two different nonlinear kinematic models, a vehicle with a trailer and a simplified bicycle model. The versatility of our method when it comes to finding trajectories from the start to the destination was clearly established, especially when using the additional heuristics from Section 3.3. Furthermore, Algorithm 3.1 was shown to outperform several state-of-the-art NLP solvers, IPOPT, SNOPT and KNITRO, in both runtime and quality of the solution, when applied to this type of problem with obstacle constraints. Moreover, an MPC simulation illustrated that our approach benefits from warm starting, a crucial aspect of any numerical solver to be used in MPC.

# Part II

## Proximal Augmented Lagrangian Method for Nonconvex Quadratic Programs

# Chapter 6

# Quadratic Programming

This chapter discusses quadratic programs (QPs), which are optimization problems of the following form

$$\underset{x \in \mathbb{R}^n}{\textbf{minimize}} \quad \tfrac{1}{2} x^\top Q x + q^\top x \tag{6.1a}$$

$$\textbf{subject to} \quad \ell \leq Ax \leq u, \tag{6.1b}$$

where $Q \in \mathrm{Sym}(\mathbb{R}^n)$, $q \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $\ell_i, u_i \in \overline{\mathbb{R}} \cup \{-\infty\}, i = 1, \ldots, m$. Even though only inequality constraints are listed, it is trivial to include equality constraints in the above formulation by setting some $\ell_i = u_i$. Optimization methods for quadratic programming are almost always focused on convex QPs, in which $Q$ is positive semidefinite. Efficiently and reliably solving QPs is a key challenge in optimization [121, §16].

Section 6.1 covers some application domains in which (convex) QPs arise, such as finance and linear MPC. Moreover, it describes a popular nonlinear programming technique, sequential quadratic programming, which as the name implies, relies on the solution of many subsequent QPs. Section 6.2 provides a brief overview of the optimization techniques for solving QPs in the literature. Existing methods fall typically under the categories of active-set methods, interior-point methods, or proximal methods. Section 6.3 discusses briefly the difficulty in extending these methods to nonconvex QPs.

## 6.1 Application domains

Quadratic programs arise in a wide variety of applications. For example, in the domain of machine learning, support vector machines (SVM) [37], the least absolute shrinkage and selection operator (Lasso) [153] and Huber fitting [87, 106] all require

the solution of or can be formulated as a convex QP. This section discusses two other domains where QPs emerge, which are used in the benchmarks of Section 9.2. The first is the domain of finance, in which optimizing a portfolio of investments, given an expected return and accompanying risks can be formulated as a QP. The second is control engineering, in which both linear MPC and moving horizon estimation require the solution of (successive) QPs. Finally, the key idea behind a popular nonlinear programming technique, sequential quadratic programming, is also briefly explained.

### 6.1.1 Finance

One of the most well-known (optimization) problems in finance is that of portfolio selection [36, §6]. Say an investor wishes to invest his funds in a series of $n$ potential assets. Let $r_i$ denote the return for the $i$-th asset. Of course, in practice, these returns are unknown. It is often assumed that these are randomly distributed along a normal distribution, with expected values $\mu_i = \mathrm{E}[r_i]$ and variance $\sigma_i^2 = \mathrm{E}[(r_i - \mu_i)^2]$. Moreover, the returns of many assets are in general not independent. As such, we can define correlations between pairs of returns as $\rho_{ij} = \frac{1}{\sigma_i \sigma_j} \mathrm{E}[(r_i - \mu_i)(r_j - \mu_j)]$. Let $x_i$ denote the fraction of the investor's funds to be allocated in asset $i$. Then, the (normalized) return value is given as $R = \sum_{i=1}^{n} x_i r_i$. By using some simple statistical calculations, one obtains the mean and variance of $R$ as $\mathrm{E}[R] = x^\top \mu, \mathrm{Var}[R] = x^\top \Sigma \mathrm{x}$, where $\Sigma$ is the covariance matrix, with entries $\Sigma_{ii} = \rho_{ij} \sigma_i \sigma_j$.

Ideally, the investor would want to maximize the expected return and minimize the risk. In the model introduced by Markowitz [107], also known as the Markowitz mean-variance model, both of these measures are combined in an optimization problem, with the risk weighted by a risk-aversion coefficient $\kappa$. The ensuing quadratic program, assuming short selling is not allowed, is then

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\textbf{minimize}} \quad & \kappa x^\top \Sigma x - \mu^\top x \\ \textbf{subject to} \quad & x \geq 0 \\ & \sum_{i=1}^{n} x_i = 1. \end{aligned}$$

Investors who are very risk-averse, can solve this problem with a high value of $\kappa$, whereas investors who wish to make a higher return at the cost of a higher risk would use a low value. Alternatively, this problem can be reformulated to minimize the risk given a wanted return, say $\bar{\mu}$

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\textbf{minimize}} \quad & x^\top \Sigma x \\ \textbf{subject to} \quad & x \geq 0 \end{aligned}$$

$$\sum_{i=1}^{n} x_i = 1,$$

$$\mu^\top x \geq \bar{\mu},$$

or, to maximize the return given a certain tolerated risk level, $\bar{\sigma}$

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\textbf{minimize}} \quad & -\mu^\top x \\ \textbf{subject to} \quad & x \geq 0 \\ & \sum_{i=1}^{n} x_i = 1, \\ & x^\top \Sigma x \leq \bar{\sigma}^2. \end{aligned}$$

Notice however that this last program is no longer a QP according to our definition, since it also contains a quadratic constraint. Quadratically constrained QPs (QCQPs) are generally much harder to solve than regular QPs, and will not be treated in this work.

### 6.1.2 Control engineering

Model predictive control is a control technique that is becoming more and more popular, given its inherent capability to optimize a control strategy over a time horizon while taking into account constraints [131]. In linear MPC, the resulting optimization problems are quadratic programs. Consider a (discrete) linear system, with states $q$, inputs $u$ and outputs $y$, which is governed by the state-space equations

$$q_{k+1} = Aq_k + Bu_k,$$

$$y_k = Cq_k + Du_k,$$

with $(A, B, C, D)$ the (discrete) state-space matrices of appropriate size. Let $n_q$, $n_u$ and $n_y$ denote the number of states, inputs and outputs respectively. Furthermore, let us assume the goal of the MPC controller is to track a reference state and input over a control horizon $N$. The resulting OCP is then given as

$$\begin{aligned} \underset{z \in \mathbb{R}^{(N+1)n_q + Nn_u}}{\textbf{minimize}} \quad & \|q_N - q_{\text{ref}}\|_P^2 + \sum_{k=0}^{N-1} \|q_k - q_{\text{ref}}\|_Q^2 + \|u_k - u_{\text{ref}}\|_R^2 \\ \textbf{subject to} \quad & q_{k+1} = Aq_k + Bu_k, \qquad k = 0, \ldots, N-1, \\ & u_k \in \mathcal{U}, \qquad\qquad\quad k = 0, \ldots, N-1, \end{aligned}$$

$$
\begin{aligned}
&q_k \in \mathcal{Q}, && k = 0, \dots, N-1, \\
&q_N \in \mathcal{Q}_N, \\
&q_0 = \bar{x}.
\end{aligned}
$$

Here, the notation $\|v\|_M^2 = v^\top M v$ is used. The vector $z$ is the stacking of all the states and inputs, that is $z = (q_0, u_0, q_1, \dots, u_{N-1}, q_N)$. $Q \in \mathrm{Sym}_+(\mathbb{R}^{n_x})$ and $R \in \mathrm{Sym}_{++}(\mathbb{R}^{n_u})$ represent the stage cost matrices for the states and inputs respectively, while $P \in \mathrm{Sym}_+(\mathbb{R}^{n_x})$ represents a terminal penalty. Moreover, the sets $\mathcal{U}$, $\mathcal{Q}$ and $\mathcal{Q}_N$ describe constraints on the control inputs, states and terminal state respectively. In order for the above OCP to be a quadratic program, these sets should be polyhedral sets. Finally, $\bar{q}$ denotes the initial state, which is estimated or measured directly.

State estimation is another important aspect of control engineering. A very popular approach is Kalman filtering [90], also known as least squares estimation. An in-depth discussion on state estimation techniques is out of the scope of this thesis. However, a method that is gaining popularity nowadays is moving horizon estimation (MHE). MHE is becoming more popular since it uses the previous $N$ measurements to make accurate estimates, and can be combined seamlessly with an MPC approach to control the system. In the simplest form, let us imagine that the state evolves autonomously. Furthermore, assume we have measurements on all the past outputs, and we denote these measurements $\hat{y}_k$. No physical system nor sensor is free of noise, however. Hence, letting $w_k$ and $v_k$ denote respectively the process noise and the measurement noise at the $k$-th step, then the state space system equation becomes $q_{k+1} = Aq_k + w_k$ and the (measured) output equation $\hat{y}_k = Cq_k + v_k$. MHE computes estimates of the previous $N$ states, given we are now at time $T \geq N$, by solving the following optimization problem

$$
\underset{z \in \mathbb{R}^{N n_q}}{\textbf{minimize}} \quad \sum_{k=T-N}^{T-1} \|q_{k+1} - Aq_k\|_{Q^{-1}} + \sum_{k=T-N}^{T} \|\hat{y}_k - Cq_k\|_{R^{-1}},
$$

where $z = (q_{T-N}, \dots, q_T)$, and $Q$ and $R$ represent process and measurement disturbance covariances, i.e. we assume that $w_k \sim \mathcal{N}(0, Q)$ and $v_k \sim \mathcal{N}(0, R)$ respectively. For the sake of brevity, the problem for $T < N$ is not considered. Moreover, the MHE problem can of course also be considered in the presence of constraints [130]. As long as these are linear, the MHE problem remains a quadratic program.

### 6.1.3 Sequential quadratic programming

A popular method for nonlinear programming is sequential quadratic programming (SQP), which as the name implies, solves a sequence of QPs. The key idea is that these QPs are local approximations to the nonlinear program, in exactly the same way Newton's method operates in unconstrained optimization [113]. Given a nonlinear

program

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\textbf{minimize}} \quad & f(x) \\ \textbf{subject to} \quad & c(x) = 0, \\ & g(x) \leq 0, \end{aligned}$$

the Lagrangian function is given as

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda c(x) + \mu c(x),$$

with $\lambda$ and $\mu$ the multipliers for the equality and inequality constraints respectively. Let $Jh(x) = (\nabla h_1(x), \ldots, \nabla h_m(x))^\top$ denote the Jacobian of a mapping $h : R^n \to R^m$. At the $k$-th iteration, using a quadratic approximation of the Lagrangian for the objective and linear approximations of the constraints around the current iterate $x^k$, SQP methods will typically construct a quadratic program of the form

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\textbf{minimize}} \quad & \nabla f(x^k)^\top (x - x^k) + \tfrac{1}{2}(x - x^k)^\top H_k (x - x^k) \\ \textbf{subject to} \quad & c(x^k) + Jc(x^k)(x - x^k) = 0, \\ & g(x^k) + Jg(x^k)(x - x^k) \leq 0. \end{aligned}$$

Note that in the objective function, $\nabla f$ appeared instead of $\nabla_x \mathcal{L}$. The program with the latter gradient would be entirely equivalent to the one presented, aside from the interpretation of the multipliers. For the matrix $H_k$, either the exact Hessian of the Lagrangian $\nabla^2_{xx}\mathcal{L}$ or a (current) approximation of it can be used. On the one hand, the exact Hessian is often expensive to compute and may be indefinite. Therefore, it is typically only chosen when it is efficiently computable and in the context of trust-region methods which prevent the QP from being unbounded. Approximations, on the other hand, can be cheaply made and kept positive definite, such that the resulting QPs are convex, but have the disadvantage of losing some of the second-order information.

In SQP methods, just applying the basic step resulting from the QP subproblem may not lead to a converging algorithm. Instead, a globalization strategy needs to be added in order to ensure convergence. The main approaches for this purpose are linesearch methods which accept steps if they provide a decrease of the merit function, and trust-region methods which add a trust-region constraint such as $\|x - x^k\|_\infty \leq \Delta_k$ to the above QP and in which the step is accepted when the quadratic model and nonlinear program correspond sufficiently. The reader is referred to [70] for a detailed overview of the many variants and techniques used in sequential quadratic programming. Popular SQP solvers include SNOPT [69], filterSQP [57] and acados [159].

## 6.2 Optimization methods

Given the emergence of quadratic programs in many interesting application domains, a substantial amount of research has naturally been devoted to the development of efficient and robust methods for solving them. Robustness is here, and in the remainder of this thesis, understood as a measure for how unaffected the solver is with respect to ill conditioning in the problem data. What makes a QP fundamentally difficult is the presence of inequality constraints, since the KKT conditions of an equality constrained quadratic program (ECQP) boil down to a set of linear equations. Specifically, given the ECQP

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\textbf{minimize}} \quad & \tfrac{1}{2} x^\top Q x + q^\top x \\ \textbf{subject to} \quad & Ax = b, \end{aligned}$$

the KKT conditions on the primal and dual solution pair $(x^*, y^*)$ are given as

$$\begin{bmatrix} Q & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ y^* \end{bmatrix} = \begin{bmatrix} -q \\ b \end{bmatrix}. \tag{6.2}$$

Hence, solving such an ECQP amounts to solving a linear system, which can be considered a done deal with current numerical linear algebra techniques. Assuming the equality constraints are linearly independent, possibly after removing redundant constraints, either the above system is solved using direct methods or, typically in the case of very large problems, using a conjugate gradient method [73]. Therefore, the main challenge lies in solving the more general QP with inequality constraints. To deal with these inequality constraints, several approaches have been invented, which are discussed in the remainder of this section. For simplicity in the following discussion, let us denote the inequality constrained QP as

$$\begin{aligned} \underset{x \in \mathbb{R}^n}{\textbf{minimize}} \quad & \tfrac{1}{2} x^\top Q x + q^\top x && \text{(6.3a)} \\ \textbf{subject to} \quad & A_{\mathcal{E}} x = b_{\mathcal{E}}, && \text{(6.3b)} \\ & A_{\mathcal{I}} x \leq b_{\mathcal{I}}. && \text{(6.3c)} \end{aligned}$$

Here $\mathcal{E}$ and $\mathcal{I}$ denote the index set of equalities and inequalities respectively. The KKT conditions of problem (6.3) on a primal-dual solution pair $(x^*, y^*)$ are given as

$$\begin{aligned} Qx^* + q + A_{\mathcal{E}}^\top y_{\mathcal{E}}^* + A_{\mathcal{I}}^\top y_{\mathcal{I}}^* = 0, && \text{(6.4a)} \\ A_{\mathcal{E}} x^* = b_{\mathcal{E}}, && \text{(6.4b)} \end{aligned}$$

$$A_{\mathcal{I}} x^* \leq b_{\mathcal{I}}, \tag{6.4c}$$

$$y^* \geq 0, \tag{6.4d}$$

$$y_i^* (Ax^* - b)_i = 0, \quad \forall i \in \mathcal{I}. \tag{6.4e}$$

Condition (6.4e) is known as the complementarity condition. It dictates that for each $i \in \mathcal{I}$ either the inequality constraint is satisfied as equality, i.e. $(Ax^* - b)_i = 0$, or the corresponding multiplier is zero, or both. In the first case, the inequality is considered to be active. Vice-versa, for an inactive inequality $i$, it holds that $(Ax^* - b)_i < 0$. Such constraints do not influence the solution. Equality constraints are always active, since they have to be satisfied as equality by definition. Let $\mathcal{A}(x) = \{i | (Ax - b)_i = 0\}$ denote the set of active constraints at $x$. It is clear that if we somehow knew $\mathcal{A}(x^*)$, problem (6.3) can be solved by solving the corresponding ECQP. This is the key idea behind one of the state-of-the-art methods, namely that of the accordingly named active-set methods. Although active-set methods may belong to primal, dual or primal-dual types, the following discussion is restricted to primal active-set methods.

### 6.2.1 Active-set methods

Active-set methods intend to find the set of active constraints at the solution. For this purpose, they keep track of a *working set*, i.e. a running estimate of the final active set. Assuming the initial point $x^0$ is feasible, the remaining iterates will retain feasibility. A common strategy to determine an initial feasible point is to first solve a feasibility linear program, which, given a user-supplied initial point $\bar{x}$, is given by

$$\begin{aligned} \underset{(x,z)\in\mathbb{R}^{n+n_z}}{\textbf{minimize}} \quad & \sum_{j=1}^{n_z} z_j \\ \textbf{subject to} \quad & A_{\mathcal{E}} x + \gamma_{\mathcal{E}} z_{\mathcal{E}} = b_{\mathcal{E}}, \\ & A_{\mathcal{I}} x + \gamma_{\mathcal{I}} z_{\mathcal{I}} \leq b_{\mathcal{I}}, \\ & z \geq 0, \end{aligned}$$

where $n_z = |\mathcal{E}| + |\mathcal{I}|$ is the total number of constraints, and $\gamma_i = -\operatorname{sgn}((A\bar{x} - b)_i)$ for $i \in \mathcal{E}$, and $\gamma_i = -1$ for $i \in \mathcal{I}$. It is easy to verify that $(\bar{x}, \bar{z})$, with $\bar{z}_i = |(A\bar{x} - b)_i|, i \in \mathcal{E}$ and $\bar{z}_i = \max(0, (b - A\bar{x})_i), i \in \mathcal{I}$, is a feasible point for the above linear program. Moreover, if $\bar{x}$ is feasible for the original problem, then $(\bar{x}, 0)$ can be shown to be optimal for the feasibility problem. Therefore, the resulting vector from solving this problem, denoted by $x^0$, can be thought of as a sort of minimal perturbation to $\bar{x}$ which is feasible for the original QP. Given $x^0$, the initial working set can be chosen as a linearly independent subset of the active constraints at $x^0$.

From then on, the (primal) active-set method will iteratively update the trial point $x^k$ and the working set $\mathcal{W}^k$ by solving first the ECQP

$$\underset{x\in\mathbb{R}^n}{\textbf{minimize}} \quad \tfrac{1}{2}x^\top Qx + q^\top x \tag{6.5a}$$

$$\textbf{subject to} \quad (Ax)_i = b_i, \quad \forall i \in \mathcal{W}^k, \tag{6.5b}$$

to obtain $\hat{x}^k$ and derive from it the trial step $p^k = \hat{x}^k - x^k$. In addition, we obtain $\hat{y}_i^k, i \in \mathcal{W}^k$ from the solution and set the remaining $\hat{y}_i^k = 0, i \notin \mathcal{W}^k$.

Now, if $p^k = 0$ and $\hat{y}_i^k \geq 0, i \in \mathcal{W}^k$, it can be shown that the KKT conditions (6.4) are satisfied at $\hat{x}^k, \hat{y}^k$ and the solver terminates. If $p^k = 0$ but some $\hat{y}_i^k < 0, i \in \mathcal{W}^k$, then the trial point can be further improved by dropping some combination of these constraints from the working set. Since it is uncertain which combination needs to be dropped, active-set methods typically drop only one constraint, and usually the one with the smallest multiplier. Hence, $x^{k+1} \leftarrow x^k$ and $W^{k+1} \leftarrow W^k \setminus \{j\}$ for some $j \in \arg\min \hat{y}_i^k$ in this scenario.

On the other hand, if $p^k \neq 0$, then the question becomes one of determining the step size to be taken in this direction. If $x^k + p^k$ is feasible for all constraints, then this full step is taken, $x^{k+1} \leftarrow x^k + p^k$. Otherwise, one needs to determine the blocking constraints, i.e. those which are infeasible at $x^k + p^k$. Since $x^k$ is feasible with respect to all constraints, the only possibility for a blocking constraint $i$ is to have $(Ap^k)_i > 0$. To determine the step size $\alpha^k$, which we want to have as close as possible to 1, the following equation needs to be solved

$$\alpha^k = \min\left(1, \min_{(Ap^k)_i > 0} \frac{b_i - (Ax^k)_i}{(Ap^k)_i}\right).$$

This equation can also be used to determine whether there were blocking constraints in the first place. After all, if $\alpha^k = 1$ no constraint blocked the full step. However, if $\alpha^k < 1$, then the constraint $j$ which achieves the minimum in the above equation is identified, upon which the trial point and working set are updated using $x^{k+1} = x^k + \alpha^k p^k$ and $\mathcal{W}^{k+1} = \mathcal{W}^k \cup \{j\}$.

From this discussion, it is clear the the working set changes only by one element per iteration. Some active-set methods may drop and add one constraint in the same iteration but, evidently, the successive ECQPs (6.5) are very similar. This fact can be exploited to avoid having to solve the linear system (6.2) from scratch every time. For example, if the linear system is solved using a null-space method, which requires the computation of the null-space of $A_{W^k}$ via a QR factorization [13, §6], then this factorization may be updated efficiently when one row is added or deleted, see [67, 39, 80].

An advantage of active-set methods is that they can easily make use of an initial guess,

also known as a warm start, both for the initial point and possibly for the initial working set. This aspect is very useful when solving a series of related QPs, such as in SQP or in MPC. The biggest drawback of active-set methods, however, is that a large number of iterations can be required to converge to the right working set, as the number of possible working sets grows exponentially with the number of constraints. Popular active-set based QP solvers include the open-source solver qpOASES [53], the QPA module in the open-source software GALAHAD [74], and QPNNLS [12], the latter being tailored for embedded MPC applications.

### 6.2.2 Interior-point methods

The key idea in interior-point methods is to try and solve the KKT conditions (6.4) more directly, albeit iteratively and with a perturbation [164]. For this reason, let us first denote a vector of slack variables $\mathbb{R}^{|\mathcal{I}|} \ni s = b_{\mathcal{I}} - A_{\mathcal{I}}x$, with which the KKT conditions can be reformulated as

$$Qx^* + q + A_{\mathcal{E}}^\top y_{\mathcal{E}}^* + A_{\mathcal{I}}^\top y_{\mathcal{I}}^* = 0, \tag{6.6a}$$

$$A_{\mathcal{E}}x^* = b_{\mathcal{E}}, \tag{6.6b}$$

$$A_{\mathcal{I}}x^* + s^* = b_{\mathcal{I}}, \tag{6.6c}$$

$$s^*, y^* \geq 0, \tag{6.6d}$$

$$y_i^* s_i^* = 0, \quad \forall i \in \mathcal{I}. \tag{6.6e}$$

Moreover, let $\mu = \frac{s^\top y}{|\mathcal{I}|}$ denote the complementarity measure, which quantifies how closely (6.6e) is satisfied. Then, the perturbed KKT conditions in $(x, s, t, \tau\mu)$ are given by

$$\begin{bmatrix} Qx + q + A^\top y \\ A_{\mathcal{I}}x + s - b_{\mathcal{I}} \\ SY\mathbf{1} - \tau\mu\mathbf{1} \\ A_{\mathcal{E}}x - b_{\mathcal{E}} \end{bmatrix} = 0, \tag{6.7}$$

where $S = \operatorname{diag}(s)$, $Y = \operatorname{diag}(y_{\mathcal{I}})$, and $\tau \in [0, 1]$ is a parameter varied over the iterations. The solutions of (6.7) for positive values of $\tau$ and $\mu$ define the so-called central path, a trajectory which leads to the solution of the QP as $\tau\mu$ tends to zero. For a given value of $(x, s, y)$, a Newton step on the system (6.7) can be computed by

solving

$$\begin{bmatrix} Q & 0 & A_{\mathcal{I}}^\top & A_{\mathcal{E}}^\top \\ A_{\mathcal{I}} & I & 0 & 0 \\ 0 & Y & S & 0 \\ A_{\mathcal{E}} & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y_{\mathcal{I}} \\ \Delta y_{\mathcal{E}} \end{bmatrix} = \begin{bmatrix} -(Qx + q + A^\top y) \\ -(A_{\mathcal{I}} x + s - b_{\mathcal{I}}) \\ -(SY\mathbf{1} - \tau\mu\mathbf{1}) \\ b_{\mathcal{E}} \end{bmatrix}. \tag{6.8}$$

Using the third equation of (6.8), $\Delta s$ can be eliminated and substituted into the second equation, leading to the equivalent system

$$\begin{bmatrix} Q & A_{\mathcal{I}}^\top & A_{\mathcal{E}}^\top \\ A_{\mathcal{I}} & -Y^{-1}S & 0 \\ A_{\mathcal{E}} & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y_{\mathcal{I}} \\ \Delta y_{\mathcal{E}} \end{bmatrix} = \begin{bmatrix} -(Qx + q + A^\top y) \\ -(A_{\mathcal{I}} x + s - b_{\mathcal{I}} - s + \tau\mu Y^{-1}\mathbf{1}) \\ b_{\mathcal{E}} \end{bmatrix}.$$

This symmetric indefinite system, assuming $\mathcal{A}_{\mathcal{E}}$ is full rank, can be solved by using standard methods, either direct or iterative [13]. After obtaining the step direction, a suitable step size needs to be determined. Typically, a distinction is made between primal and dual steps, such that $(x^{k+1}, s^{k+1}) = (x^k, s^k) + \alpha_{\mathrm{p}}^k(\Delta x, \Delta s)$ and $\mathrm{Y}^{k+1} = y^k + \alpha_{\mathrm{d}}^k \Delta y$. Obtaining suitable step sizes is a crucial aspect that influences the convergence of the methods, and relies typically on a filter or a linesearch on a merit function. A comprehensive overview is given in [161]. The centering parameter $\tau$ is typically updated using a heuristic, such as $\tau = \frac{\mu_{\mathrm{aff}}}{\mu}$, with $\mu_{\mathrm{aff}}$ obtained from an affine scaling step, which solves (6.8) for $\tau = 0$.

Interior point methods usually require fewer but more expensive iterations than active-set methods. Their iterations involve the solution of a fresh linear system at every iteration. Interior-point methods are generally efficient, but suffer from not having warm starting capabilities. Examples of state-of-the-art QP solvers using an interior-point method are the commercial solvers Gurobi [79] and MOSEK [114], the closed-source solver BPMPD [111] and the open-source solvers OOQP [63] and HPIPM [60], the last one being tailored to small-to-medium dense and OCP-structured problems. Moreover, general nonlinear interior-point methods, such as IPOPT [161] and KNITRO [24], also work relatively well for convex and nonconvex QPs. Finally, [59] uses an interior-point method after adding a proximal-point type regularization on the primal and dual problems. This regularization is similar to the one used in the proximal augmented Lagrangian method (P-ALM), as we will see in §7, and provides an advantage in solving the linear systems, since for instance rank-deficient constraint Jacobians no longer pose an issue.

### 6.2.3 Proximal methods

Recently, proximal algorithms, also known as operator splitting methods [122], have experienced a resurgence in popularity. Relying only on first-order information of the problems, such methods have as their advantage operational simplicity and cheap iterations, but they may exhibit slow asymptotic convergence for poorly scaled problems. Many of the algorithms, however, may be accelerated by Nesterov-type [119],[120, §2.2]

methods [71, 126, 101], or by Newton-type methods [142, 152]. Moreover, proximal algorithms are only interesting when the involved proximal steps are computationally tractable. Fortunately, this is very often the case, as demonstrated in the many examples in [122]. The proximal operator and the proximal point algorithm (PPA) [133] with its relation to P-ALM are discussed in Section 7.1. This subsection provides instead a brief overview on the most common proximal algorithms that can be applied to QPs. Moreover, these algorithms are only outlined qualitatively, since there is no central method.

One of the simplest proximal algorithms is the proximal gradient method, also known as forward-backward splitting (FBS), which interleaves a gradient descent step and a proximal step, as mentioned in Section 3.2.1. Typically, the proximal part applies to the indicator of a constraints set, in which case the step becomes a projection on that set. The proximal gradient method is then also known as the projected gradient method. Such a method is for instance interesting for a box constrained QP

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\textbf{minimize}} \quad & \tfrac{1}{2}x^\top Q x + q^\top x \\ \textbf{subject to} \quad & \ell \leq x \leq u. \end{aligned}$$

The proximal gradient method may be accelerated using a Nesterov-type scheme [101]. A more sophisticated variant also exists, known as the gradient projection method [33]. This method looks along the piecewise-linear path obtained by projecting the steepest descent step on the box to find an (approximately) optimal point along this path every iteration, instead of just the end-point as in the normal projected gradient method. The gradient projection method applied to the dual QP has known some success in MPC applications [9], and can be further accelerated with a Nesterov-type method [124].

The application of the proximal gradient method to the dual problem leads to another proximal algorithm, known as the alternating minimization algorithm (AMA), first introduced in [154]. AMA is useful for problems with a separable structure, such as those arising in distributed MPC. Like the proximal gradient method, there exist accelerated versions of AMA, based on Nesterov-type acceleration [71], or on a combination with Newton-type methods [142].

Another popular proximal algorithm was first introduced in [48] to solve heat differential equations and is known as Douglas-Rachford splitting (DRS), which can be shown to be equivalent to the alternating direction method of multipliers (ADMM). Like AMA, ADMM is also suited for composite optimization problems, such as those arising in distributed MPC [52, 157]. ADMM-type schemes can also be accelerated using Nesterov-type [71, 126] or Newton-type methods [152]. Recently, a full-fledged (convex) QP solver based on ADMM was proposed in [144], the operator-splitting QP (OSQP) solver. It partially overcomes the weakness of first order methods against ill conditioning by means of a tailored offline scaling and a small amount of online parameter adjustments, although a more thorough benchmarking, see §9, confirms the known limitations of proximal algorithms. Indeed, since the parameters are typically

set before the execution, with only minor online adjustments, the operator splitting iterations remain simple and computationally cheap, but mostly fail to take into account curvature information about the problem that can greatly speed up convergence.

## 6.3 Nonconvex QPs

Nonconvex QPs arise in several application domains, such as in a reformulation of mixed integer quadratic programs, in the solution of partial differential equations and in VLSI chip design [163]. Furthermore, a (potentially) indefinite QP has to be solved at every iteration of a sequential quadratic programming method applied to a nonconvex optimization problem if the exact Hessian of the Lagrangian is used, as mentioned in Section 6.1.3.

It is generally difficult to extend the aforementioned methods to be able to find stationary points of nonconvex QPs, without additional assumptions. Augmented Lagrangian based algorithms and ADMM, for example, would require surjectivity of the constraint matrix $A$ [100, 19, 21, 150]. Some proposals have been made for interior-point methods to solve nonconvex QPs [167, 1] and for active-set methods [68, 56, 77], but these methods were either not available or found to often exhibit numerical issues in our benchmarks. Finally, global optimization of nonconvex QPs has been the topic of a large amount of research, see for example [139, 22, 26], but this is a separate issue and will not be discussed further here, since finding the global optimum of a nonconvex QP is an NP-hard problem [118] and we are satisfied with finding just a stationary point.

One method that has not been discussed until now is the proximal augmented Lagrangian method. In the next chapter, a detailed overview of this method will be given, alongside a convergence proof when applied to nonconvex QPs, albeit after a slight modification, without any surjectivity assumptions.

## 6.4 Summary

This chapter discussed the prevalence of quadratic programs in various application domains. Well-known examples are support vector machines, lasso and Huber-fitting in machine learning, portfolio optimization in finance and linear MPC and MHE in control engineering. Quadratic programming solvers have therefore been the topic of decades of development. State-of-the-art algorithms for inequality constrained QPs rely typically on active-set methods, interior-point methods or proximal algorithms. However, it is generally difficult to extend these methods for nonconvex QPs, and little focus has been put on them. In contrast, the proximal augmented Lagrangian method, which will be discussed in the next chapter, may naturally deal with such nonconvexity.

# Chapter 7

## Proximal Augmented Lagrangian Method

This chapter discusses the application of the augmented Lagrangian method (ALM), first introduced by Hestenes [86] and Powell [128], and in particular proximal ALM, originally known as the proximal method of multipliers [132], to quadratic programs. The augmented Lagrangian method is described in detail in [15], and has been applied in nonlinear optimization codes, such as MINOS [116, 117, 115], LANCELOT [34], PENNON [92], ALGENCAN [5] and GRAMPC [49], the last one being tailored to embedded MPC applications. However, research on the application of ALM to QPs is rather limited compared to the other state-of-the-art methods of Section 6.2. The authors of [66] presented an ALM solver for QPs, which solves the inner minimization problems using a combination of active-set and gradient projection methods, although this research was not peer-reviewed and seems to be discontinued. FBstab [102] was developed independently and around the same time as QPALM, relying also on P-ALM, but restricted to convex QPs. Our inner minimization routine, presented in §8, differs from FBstab, however. The latter focuses particularly on small dense and OCP-structured problems, considers penalty parameter values instead of vectors, and uses a backtracking linesearch.

This chapter considers the application of P-ALM to quadratic programs, and establishes its theoretical ground in the form of convergence proofs. Section 7.1 outlines the basic P-ALM iteration and its origin, the proximal point algorithm (PPA). It is shown how P-ALM and PPA applied to the KKT operator are equivalent, given that the KKT operator is monotone. Section 7.2 shows the theoretical convergence of the PPA applied to the KKT conditions of a convex QP, drawing heavily from existing literature on monotone operator theory. Section 7.3 looks into application of P-ALM to nonconvex QPs, where despite retaining its equivalence to PPA applied to the (hypomonotone) KKT conditions, convergence is not readily obtained. Small modifications need to be made to the basic P-ALM iteration to make it instead equivalent to inexact proximal point iterations applied to the extended cost function. A convergence proof for the proposed algorithm is also given at the end of this section.

The material in this chapter is based on the publications [84, 85].

## 7.1 Proximal Method of Multipliers

This section discusses the proximal ALM iteration and the proximal point algorithm. Both of these require the solution of a sequence of (perturbed) optimization problems. For now, inexactness is left out of the discussion for simplicity, but it is treated in the convergence proofs of Sections 7.2 and 7.3.

### 7.1.1 Proximal ALM

To outline the proximal augmented Lagrangian method, first a reformulation of the QP in (6.1) is needed, given by introducing slack variables $z = Ax$ as

$$\underset{(x,z)\in\mathbb{R}^{n+m}}{\textbf{minimize}} \quad \tfrac{1}{2}x^\top Qx + q^\top x \tag{7.1a}$$

$$\textbf{subject to} \quad Ax = z, \tag{7.1b}$$

$$\ell \leq z \leq u. \tag{7.1c}$$

Obviously, problem (7.1) is equal to (6.1) after eliminating the slack variable $z$, and so the two problems are equivalent. Problem (7.1) has the (theoretical) advantage, however, of having no inequality constraints aside from bounds on the variables. Furthermore, let the set $C = [\ell, u]$ denote the set of bounds, and the functions

$$f(x) \coloneqq \tfrac{1}{2}x^\top Qx + q^\top x, \tag{7.2a}$$

$$g(z) \coloneqq \delta_C(z) = \begin{cases} 0 & \text{if } z \in C, \\ \infty & \text{otherwise,} \end{cases} \tag{7.2b}$$

denote the objective function and the constraint function on $z$. Then, (7.1) is equivalent to

$$\underset{(x,z)\in\mathbb{R}^{n+m}}{\textbf{minimize}} \quad f(x) + g(z) \tag{7.3a}$$

$$\textbf{subject to} \quad Ax = z. \tag{7.3b}$$

The Lagrangian function of (7.3) with respect to the equality constraints (7.3b) is given by

$$\mathcal{L}(x, z, y) \coloneqq f(x) + g(z) + y^\top(Ax - z), \tag{7.4}$$

with $y$ the corresponding Lagrange multiplier. The augmented Lagrangian is obtained by augmenting this regular Lagrangian with an additional (quadratic) penalty on the constraint violation. Given a positive definite (diagonal) matrix of penalty parameters $\Sigma_Y$, the augmented Lagrangian function is thus given as

$$\mathcal{L}_{\Sigma_Y}(x, z, y) \coloneqq f(x) + g(z) + y^\top(Ax - z) + \tfrac{1}{2}\|Ax - z\|^2_{\Sigma_Y}. \tag{7.5}$$

The proximal augmented Lagrangian, furthermore, adds a proximal penalty on the distance between the new $x$ iterate and the regularization point $\hat{x}$. In regular P-ALM, $\hat{x}^k = x^k$, but a distinction is made here already since it will be necessary once modifications for P-ALM are discussed in the nonconvex case. Given a positive definite (diagonal) matrix of proximal penalty parameters $\Sigma_{\mathrm{X}}$ and a regularization point $\hat{x}$, the proximal augmented Lagrangian function is given as

$$\mathcal{L}_{\hat{x},\Sigma_{\mathrm{X}},\Sigma_{\mathrm{Y}}}(x,z,y) = f(x) + g(z) + y^\top(Ax - z) + \tfrac{1}{2}\|Ax - z\|^2_{\Sigma_{\mathrm{Y}}} + \tfrac{1}{2}\|x - \hat{x}\|^2_{\Sigma_{\mathrm{X}}^{-1}}. \quad (7.6)$$

P-ALM iteratively updates $(x, z)$ by minimizing the proximal augmented Lagrangian function and (typically) using a first-order update for $y$ as follows

$$\begin{cases} (x^{k+1}, z^{k+1}) = \arg\min_{x,z} \mathcal{L}_{\hat{x}^k,\Sigma_{\mathrm{X},k},\Sigma_{\mathrm{Y},k}}(x,z,y^k) \\ y^{k+1} = y^k + \Sigma_{\mathrm{Y},k}\big(Ax^{k+1} - z^{k+1}\big) \\ \hat{x}^{k+1} = x^{k+1}. \end{cases} \quad (7.7)$$

Apparently, by first minimizing (7.6) with respect to $z$, $z^{k+1}$ is given by

$$\begin{aligned} z^{k+1} &= \arg\min_z \mathcal{L}_{\hat{x}^k,\Sigma_{\mathrm{X},k},\Sigma_{\mathrm{Y},k}}(x,z,y^k) \\ &= \arg\min_z g(z) + y^\top(Ax - z) + \tfrac{1}{2}\|Ax - z\|^2_{\Sigma_{\mathrm{Y},k}} \\ &= \arg\min_z g(z) + \tfrac{1}{2}\|Ax + \Sigma^{-1}_{\mathrm{Y},k}y^k - z\|^2_{\Sigma_{\mathrm{Y},k}} \\ &= \mathrm{prox}_g^{\Sigma^{-1}_{\mathrm{Y},k}}(Ax + \Sigma^{-1}_{\mathrm{Y},k}y^k) = \Pi_C(Ax + \Sigma^{-1}_{\mathrm{Y},k}y^k), \end{aligned} \quad (7.8)$$

with the notation $\mathrm{prox}_g^{\Sigma^{-1}_{\mathrm{Y},k}}$ as defined in (1.9). Note that in the above derivation, terms independent of $z$, such as $f(x)$, $\frac{1}{2}\|x-\hat{x}\|^2_{\Sigma_{\mathrm{X}}^{-1}}$ and $-\Sigma^{-1}_{\mathrm{Y},k}\|y^k\|^2$ have been dropped since they do not influence the minimizer. Moreover, the last equality follows from the fact that $\Sigma_{\mathrm{Y},k}$ is diagonal and the set $C$ is separable, implying that the projection on $C$ with respect to the Euclidean norm and that induced by $\Sigma_{\mathrm{Y},k}$ coincide. Let $Z_k(x)$ denote the right-hand side of (7.8), i.e.

$$Z_k(x) = \Pi_C(Ax + \Sigma^{-1}_{\mathrm{Y},k}y^k), \quad (7.9)$$

which is a Lipschitz-continuous mapping. Then, it is possible to work out the minimization with respect to $x$ as

$$\begin{aligned} x^{k+1} &= \arg\min_x \mathcal{L}_{\hat{x}^k,\Sigma_{\mathrm{X},k},\Sigma_{\mathrm{Y},k}}(x, Z_k(x), y^k) \\ &= \arg\min_x f(x) + \tfrac{1}{2}\|x - \hat{x}^k\|^2_{\Sigma^{-1}_{\mathrm{X},k}} + \tfrac{1}{2}\|Ax + \Sigma^{-1}_{\mathrm{Y},k}y^k - Z_k(x)\|^2_{\Sigma_{\mathrm{Y},k}} \\ &= \arg\min_x f(x) + \tfrac{1}{2}\|x - \hat{x}^k\|^2_{\Sigma^{-1}_{\mathrm{X},k}} + \tfrac{1}{2}\,\mathrm{dist}^2_{\Sigma_{\mathrm{Y},k}}(Ax + \Sigma^{-1}_{\mathrm{Y},k}y^k, C), \end{aligned} \quad (7.10)$$

where $\operatorname{dist}_{\Sigma_{\mathrm{Y},k}}(z, C)$ is the distance from $z$ to $C$ induced by the norm in $\Sigma_{\mathrm{Y},k}$.

Let $\hat{\varphi}_k(x)$ denote the objective function in the last minimization problem, i.e.

$$\hat{\varphi}_k(x) \coloneqq f(x) + \tfrac{1}{2} \operatorname{dist}^2_{\Sigma_{\mathrm{Y},k}}(Ax + \Sigma_{\mathrm{Y},k}^{-1} y^k, C) + \tfrac{1}{2}\|x - \hat{x}^k\|^2_{\Sigma_{\mathrm{X},k}^{-1}}. \tag{7.11}$$

This function is (Lipschitz) differentiable and, through an appropriate choice of $\Sigma_{\mathrm{X},k}$ as discussed later, strongly convex with gradient

$$\nabla\hat{\varphi}_k(x) = \nabla f(x) + A^\top\big(y^k + \Sigma_{\mathrm{Y},k}(Ax - Z_k(x))\big) + \Sigma_{\mathrm{X},k}^{-1}(x - \hat{x}^k). \tag{7.12}$$

For convenience of notation, let us also denote a version $\varphi_k(x)$ which is equal to $\hat{\varphi}_k(x)$ with the exception that $\hat{x}^k$ is replaced by $x^k$, namely

$$\varphi_k(x) \coloneqq f(x) + \tfrac{1}{2} \operatorname{dist}^2_{\Sigma_{\mathrm{Y},k}}(Ax + \Sigma_{\mathrm{Y},k}^{-1} y^k, C) + \tfrac{1}{2}\|x - x^k\|^2_{\Sigma_{\mathrm{X},k}^{-1}}. \tag{7.13}$$

By combining (7.7), (7.8), (7.10) and (7.11), it is clear that the P-ALM iteration amounts to the update

$$\begin{cases} x^{k+1} = \operatorname{arg\,min}_{x\in\mathbb{R}^n} \hat{\varphi}_k(x) \\ z^{k+1} = \Pi_C(Ax^{k+1} + \Sigma_{\mathrm{Y},k}^{-1} y^k) \\ y^{k+1} = y^k + \Sigma_{\mathrm{Y},k}\big(Ax^{k+1} - z^{k+1}\big) \\ \hat{x}^{k+1} = x^{k+1}. \end{cases} \tag{7.14}$$

Regular ALM replaces the proximal augmented Lagrangian in (7.7) with the augmented Lagrangian function in (7.5). Although the two methods differ only in the addition of the proximal penalty term, P-ALM has some theoretical advantages with respect to plain ALM, at seemingly no additional cost [132]. In particular, when considering nonconvex problems, the proximal penalty can be used to guarantee the strong convexity of the subproblems, and as a result the existence and uniqueness of their solutions, as illustrated in Remark 7.1.

**Remark 7.1** (Proximal ALM vs plain ALM)**.** A major advantage of proximal ALM over plain ALM when applied to a nonconvex QP is that by suitably selecting the proximal weights each subproblem is guaranteed to have solutions. An illustrative example showing how ALM may not be applicable is given by the nonconvex QP

$$\begin{aligned} \underset{x\in\mathbb{R}^2}{\textbf{minimize}} \quad & x_1 x_2 \\ \textbf{subject to} \quad & x_1 = 0, \end{aligned}$$

which is clearly lower bounded and with minimizers given by $\big\{x \in \mathbb{R}^2 \mid x_1 = 0\big\}$. For a fixed penalty $\beta > 0$ and a Lagrangian multiplier $y \in \mathbb{R}$, the $x$-minimization step prescribed by ALM is

$$x^+_{\mathrm{ALM}} \in \operatorname*{arg\,min}_{w\in\mathbb{R}^2} \big\{w_1 w_2 + y^\top w_1 + \tfrac{\beta}{2}\|w_1\|^2\big\} = \emptyset,$$

owing to lower unboundedness of the augmented Lagrangian (take, e.g., $w^k = (1, -k)$ for $k \to \infty$). The problem is readily solved by proximal ALM, as long as the proximal weight $\Sigma_X \in \mathrm{Sym}_{++}(\mathbb{R}^2)$ satisfies $\Sigma_X \prec \mathrm{I}$ (the $2 \times 2$-identity matrix). In fact, the P-ALM update step results in

$$x^+_{\text{P-ALM}} \in \operatorname*{argmin}_{w \in \mathbb{R}^2} \left\{ w_1 w_2 + y w_1 + \tfrac{\beta}{2} \|w_1\|^2 + \tfrac{1}{2} \|w - x\|^2_{\Sigma_X^{-1}} \right\}$$

$$= \left\{ \left[ \Sigma_X^{-1} + \begin{pmatrix} \beta & 1 \\ 1 & \end{pmatrix} \right]^{-1} \left( \Sigma_X^{-1} x - \begin{pmatrix} y \\ 0 \end{pmatrix} \right) \right\},$$

which is well defined regardless of what the Lagrange multiplier $y$ and the penalty $\beta$ are. □

The choice and update strategy of the penalty weights $\Sigma_{Y,k}, \Sigma_{X,k}$ and the tolerances for the subproblem minimization are crucial for the efficiency of the method in practice. For the theory however, we only need some simple conditions on these parameters, as outlined in Sections 7.2 and 7.3. The practical choices for these parameters are discussed §8. In the next section, the origin of P-ALM, namely the proximal point algorithm, is investigated, since this will allow us to derive convergence in the convex case straightforwardly.

### 7.1.2 Proximal point algorithm

One rather general algorithm that can be applied to optimization problems is the proximal point algorithm (PPA) [133]. The PPA generally considers the problem of finding $0 \in T(x)$, and consists of applying a sequence of resolvent operations on $T$, that is

$$x^{k+1} \in (\mathrm{id} + \Sigma_{X,k} T)^{-1}(x^k). \tag{7.15}$$

The key ingredient of the proximal point algorithm is the proximal operator. As defined in Definition 1.1, the proximal operator applied to a function $h$ and for a step size $\gamma > 0$ is the set-valued mapping

$$\mathrm{prox}_{\gamma h}(x) = \operatorname*{argmin}_{w \in \mathbb{R}^n} \left\{ h(w) + \tfrac{1}{2\gamma} \|w - x\|^2 \right\}.$$

If $\gamma$ can be and is chosen such that $h + \frac{1}{2\gamma} \|\cdot\|^2$ is strongly convex, then the minimizer above is unique and $\mathrm{prox}_{\gamma h}(x)$ would therefore be single-valued. An important property of the proximal operator when $h$ is convex is the (extended) Moreau decomposition [11, Thm. 14.3(ii)], which states that

$$x = \mathrm{prox}_{\gamma h}(x) + \gamma \, \mathrm{prox}_{\gamma^{-1} h^*}(x/\gamma). \tag{7.16}$$

Here, $h^*$ is the conjugate function of $h$, as defined in (1.1).

Recall from (1.9) that the definition of the proximal operator can be extended for a matrix $\Sigma \in \mathrm{Sym}_{++}(\mathbb{R}^n)$ of "step sizes",

$$\mathrm{prox}_h^{\Sigma}(x) = \underset{w \in \mathbb{R}^n}{\mathrm{argmin}} \left\{ h(w) + \tfrac{1}{2}\|w - x\|_{\Sigma^{-1}}^2 \right\}. \tag{7.17}$$

Typically, we will only consider a diagonal matrix for $\Sigma$, in which case each diagonal element represents the step size in the corresponding component of $x$. Assuming the objective in (7.17) is convex, the minimization in (7.17) leads to the following optimality condition on $\mathrm{prox}_h^{\Sigma}(x)$,

$$\hat{x} = \mathrm{prox}_h^{\Sigma}(x) \quad \Leftrightarrow \quad 0 \in \partial h(\hat{x}) + \Sigma^{-1}(\hat{x} - x). \tag{7.18}$$

Depending on the definition of $T$ in (7.15), different algorithms may be derived from the PPA. The most intuitive and straightforward application of the PPA is to a primal optimality condition, and the result is then known as the proximal minimization algorithm.

## Primal PPA

Given a minimization problem

$$\underset{x \in \mathbb{R}^n}{\textbf{minimize}} \quad \varphi(x), \tag{7.19}$$

the proximal minimization algorithm is obtained by setting $T = \partial\varphi$ in the PPA. Notice that (7.3) may be formulated equivalently as a problem of type (7.19), by setting

$$\varphi(x) \coloneqq f(x) + g(Ax). \tag{7.20}$$

Assuming that $\Sigma_{\mathrm{x},k}$ is chosen such that $\varphi + \frac{1}{2}\|\cdot\|_{\Sigma_{\mathrm{x},k}^{-1}}^2$ is convex, it is possible to work out (7.15) for this case as follows

$$\begin{aligned}
& x^{k+1} = (\mathrm{id} + \Sigma_{\mathrm{x},k}\partial\varphi)^{-1}(x^k) \\
\Rightarrow\ & x^{k+1} + \Sigma_{\mathrm{x},k}\partial\varphi(x^{k+1}) \ni x^k \\
\Rightarrow\ & \partial\varphi(x^{k+1}) + \Sigma_{\mathrm{x},k}^{-1}(x^{k+1} - x^k) \ni 0 \\
\Rightarrow\ & x^{k+1} = \mathrm{prox}_{\varphi}^{\Sigma_{\mathrm{x},k}}(x^k),
\end{aligned} \tag{7.21}$$

where the last implication follows from (7.18). Therefore, it is clear that primal PPA iteratively applies the proximal operator to $\varphi$. The resulting minimization (7.21) can be seen as a trade-off between minimizing the original function $\varphi$ and remaining close to $x^k$. The smaller the diagonal elements of $\Sigma_{\mathrm{x},k}$, the more emphasis is placed on staying close to $x^k$. Vice-versa, the larger these elements, the more emphasis is placed

on minimizing $\varphi$. Clearly, it is typically advisable to choose higher values for $\Sigma_{\mathrm{x},k}$, since the original goal is to minimize $\varphi$. Yet, these elements can not be chosen too high, lest the favorable regularization properties such as strong convexity are lost. An effective choice for $\Sigma_{\mathrm{x},k}$ will be discussed in §8.

The second application of the PPA is to the dual problem of (7.19), which is similar to the derivation of the proximal minimization algorithm. It can be shown that PPA applied to the dual problem, at least in the convex setting, yields the same iterations as the augmented Lagrangian method [132]. More interestingly for us, is the third application of PPA, which considers $T$ to be the KKT operator of problem (7.3), resulting in primal-dual PPA.

### Primal-dual PPA

The KKT-operator of the QP can be derived from the stationarity condition $0 \in \partial\mathcal{L}(x, z, y)$, which given (7.4) can be worked out to give primal-dual conditions on $(x, y)$ by eliminating the slack variable $z$ and using a property of the conjugate function, $y \in \partial g(Ax) \Leftrightarrow Ax \in \partial g^*(y)$, resulting in

$$0 \in \partial\mathcal{L}(x, z, y) = \begin{pmatrix} \nabla f(x) + A^\top y \\ \partial g(z) - y \\ Ax - z \end{pmatrix}$$

$$\Rightarrow\ 0 \in \mathcal{M}(x, y) = \begin{pmatrix} \nabla f(x) + A^\top y \\ -Ax + \partial g^*(y) \end{pmatrix}.$$

The KKT-operator $\mathcal{M}(x, y)$ is maximally monotone if the given QP is convex. PPA applied to the KKT-operator of a convex QP therefore leads to primal-dual updates of the form

$$(x^{k+1}, y^{k+1}) = (\mathrm{id} + \Sigma_k \mathcal{M})^{-1}(x^k, y^k), \tag{7.22}$$

with $\Sigma_k = \begin{bmatrix} \Sigma_{\mathrm{x},k} & 0 \\ 0 & \Sigma_{\mathrm{y},k} \end{bmatrix}$ a positive-definite diagonal matrix for all $k$. Equation (7.22) can be worked out as follows

$$0 = \Sigma_{\mathrm{x},k}^{-1}(x^{k+1} - x^k) + \nabla f(x^{k+1}) + A^\top y^{k+1}, \tag{7.23a}$$

$$0 \in y^{k+1} - y^k + \Sigma_{\mathrm{y},k}(\partial g^*(y^{k+1}) - Ax^{k+1}). \tag{7.23b}$$

Recognizing a condition of type (7.18) in (7.23b), the latter is equivalent to

$$\begin{aligned} y^{k+1} &= \mathrm{prox}_{\Sigma_{\mathrm{y},k} g^*}(y^k + \Sigma_{\mathrm{y},k} Ax^{k+1}) \\ &= y^k + \Sigma_{\mathrm{y},k} Ax^{k+1} - \Sigma_{\mathrm{y},k}\, \mathrm{prox}_{\Sigma_{\mathrm{y},k}^{-1} g}(Ax^{k+1} + \Sigma_{\mathrm{y},k}^{-1} y^k) \end{aligned}$$

$$= \Sigma_{\mathrm{Y},k}\big(Ax^{k+1} + \Sigma_{\mathrm{Y},k}^{-1}y^k - \Pi_C(Ax^{k+1} + \Sigma_{\mathrm{Y},k}^{-1}y^k)\big)$$

$$= y^k + \Sigma_{\mathrm{Y},k}(Ax^{k+1} - z^{k+1}),$$

where the second equality follows from the Moreau decomposition (7.16), the third one from the fact that $\Sigma_{\mathrm{Y},k}$ is diagonal and the set $C$ is separable, implying that the projection on $C$ with respect to the Euclidean norm and that induced by $\Sigma_{\mathrm{Y},k}$ coincide, and the last one by introducing the auxiliary variable $z^{k+1} = \Pi_C(Ax^{k+1} + \Sigma_{\mathrm{Y},k}^{-1}y^k) = Z_k(x^{k+1})$, as defined in (7.9). Notice that $A^\top y^{k+1}$ is the gradient of $\frac{1}{2}\operatorname{dist}^2_{\Sigma_{\mathrm{Y},k}}(A\cdot + \Sigma_{\mathrm{Y},k}^{-1}y^k, C)$ at $x^{k+1}$. Therefore, the right-hand side of (7.23a) is equal to the gradient of $\varphi_k(x)$, defined in (7.13), such that $x^{k+1} = \arg\min_x \varphi_k(x)$. As such, a resolvent step (7.22) amounts to

$$\begin{cases} x^{k+1} = \arg\min_x \varphi_k(x) \\ z^{k+1} = \Pi_C(Ax^{k+1} + \Sigma_{\mathrm{Y},k}^{-1}y^k) \\ y^{k+1} = y^k + \Sigma_{\mathrm{Y},k}\big(Ax^{k+1} - z^{k+1}\big). \end{cases} \tag{7.24}$$

Clearly, comparing (7.14) and (7.24), it is shown that P-ALM and primal-dual PPA are equivalent, at least for convex QPs. In the next section, it is shown that, even with inexactness, iterations of type (7.22) applied to a convex QP converge to a KKT-optimal pair.

## 7.2 Convex QP

This section derives convergence guarantees in the convex case for the resolvent iterations (7.22) with errors, following the original analysis in [132]. That analysis is here generalized to account for scaling matrices (as opposed to scalars) and by including the possibility of having different Lagrangian and proximal weights.

Let $\mathcal{V}_\star \coloneqq \operatorname{zer}\mathcal{M}$ be the set of primal-dual solutions. Since $\mathcal{M}$ is maximally monotone, as first observed in [132] one can find KKT-optimal primal-dual pairs by recursively applying the resolvent of $c_k\mathcal{M}$, where $\{c_k\}_{k\in\mathbb{N}}$ is an increasing sequence of strictly positive scalars, which is recognized as primal-dual PPA [132]. We now show that these scalars can in fact be replaced by positive definite matrices.

**Theorem 7.2.** *Suppose that (6.1) has a solution. Starting from $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$, let $\{x^k, y^k\}_{k\in\mathbb{N}}$ be recursively defined as*

$$(x^{k+1}, y^{k+1}) = (\mathrm{id} + \Sigma_k\mathcal{M})^{-1}(x^k, y^k) + e^k \tag{7.25}$$

*for a summable sequence $\{e^k\}_{k\in\mathbb{N}}$, and where $\Sigma_k \coloneqq \left(\begin{smallmatrix}\Sigma_{\mathrm{X},k} & \\ & \Sigma_{\mathrm{Y},k}\end{smallmatrix}\right)$ for some $\Sigma_{\mathrm{X},k} \in \mathrm{Sym}_{++}(\mathbb{R}^n)$ and $\Sigma_{\mathrm{Y},k} \in \mathrm{Sym}_{++}(\mathbb{R}^m)$. If $\Sigma_k \preceq \Sigma_{k+1} \preceq \Sigma_\infty \in \mathrm{Sym}_{++}(\mathbb{R}^n \times \mathbb{R}^m)$ holds for all $k$, then $\{x^k, y^k\}_{k\in\mathbb{N}}$ converges to a KKT-optimal pair for (6.1).*

*Proof.* We start by observing that for all $k$ it holds that $\operatorname{zer}(\Sigma_k\mathcal{M}) = \operatorname{zer}(\mathcal{M}) = \mathcal{V}_\star$ and that $\Sigma_k\mathcal{M}$ is maximally monotone with respect to the scalar product induced by $\Sigma_k^{-1}$. The resolvent $(\mathrm{id} + \Sigma_k\mathcal{M})^{-1}$ is thus firmly nonexpansive in that metric (see [11, Prop. 23.8 and Def. 4.1]): that is, denoting $v^k := (x^k, y^k)$ and $\tilde{v}^{k+1} := v^{k+1} - e^k = (\mathrm{id} + \Sigma_k\mathcal{M})^{-1}(v^k)$,

$$\|\tilde{v}^{k+1} - v_\star\|^2_{\Sigma_k^{-1}} \leq \|v^k - v_\star\|^2_{\Sigma_k^{-1}} - \|\tilde{v}^{k+1} - v^k\|^2_{\Sigma_k^{-1}} \tag{7.26}$$

holds for every $v_\star \in \mathcal{V}_\star$. Therefore, since $\|v^{k+1} - v_\star\|_{\Sigma_{k+1}^{-1}} \leq \|v^{k+1} - v_\star\|_{\Sigma_k^{-1}} \leq \|\tilde{v}^{k+1} - v_\star\|_{\Sigma_k^{-1}} + \|e^k\|_{\Sigma_k^{-1}} \leq \|v^k - v_\star\|_{\Sigma_k^{-1}} + \|e^k\|_{\Sigma_k^{-1}}$ (where the first inequality owes to the fact that $\Sigma_{k+1}^{-1} \preceq \Sigma_k^{-1}$), it follows from [32, Thm. 3.3] that the proof reduces to showing that any limit point of $\{v^k\}_{k\in\mathbb{N}}$ belongs to $\mathcal{V}_\star$.

From [32, Prop. 3.2(i)] it follows that the sequence is bounded and that $v^{k+1} - v^k \to 0$ as $k \to \infty$. Suppose that a subsequence $\{v^{k_j}\}_{j\in\mathbb{N}}$ converges to $v$; then, so do $v^{k_j+1}$ and $\tilde{v}^{k_j+1} = v^{k_j+1} - e^{k_j} = (\mathrm{id} + \Sigma_{k_j}\mathcal{M})^{-1}v^{k_j}$. We have

$$\Sigma_{k_j}^{-1}(v^{k_j} - \tilde{v}^{k_j+1}) \in \mathcal{M}(\tilde{v}^{k_j+1}),$$

since $\{\Sigma_k^{-1}\}_{k\in\mathbb{N}}$ is upper bounded, the left-hand side converges to 0, and from outer semicontinuity of $\mathcal{M}$ [134, Ex. 12.8(b)] it then follows that $0 \in \mathcal{M}(v)$, proving the claim. □

The above result was not found in literature, but is arguably only a very minor extension to the analysis of [132]. Monotone operator theory is now a relatively well established and mature theory. However, literature on nonmonotone operators is much sparser. The next section considers the nonconvex QP, for which the KKT-operator is no longer monotone.

## 7.3 Nonconvex QP

P-ALM when applied to convex problems has been shown in Section 7.1 to be equivalent to resolvent iterations on the monotone operator encoding the KKT optimality conditions. While this interpretation is still valid for (6.1) when $Q \in \mathrm{Sym}(\mathbb{R}^n)$ is no longer a positive semidefinite matrix, the resulting KKT system lacks the monotonicity requirement that is needed for guaranteeing convergence of the iterates along the lines of Theorem 7.2. Nonmonotone operators have not been covered anywhere near as extensively as monotone operators in the literature. In fact, $\mathcal{M}$ is hypomonotone, meaning that it can be made monotone by adding a suitably large multiple of the identity mapping. However, the same cannot be said about its inverse $\mathcal{M}^{-1}$, which means that $\mathcal{M}$ is not cohypomonotone. As far as we know, only convergence of the PPA applied to cohypomonotone operators has been proven [88, 31] and no result for hypomonotone operators is readily available. For this reason, in this section we

will instead try to relate a modified version of P-ALM to the inexact primal PPA, outlined in Section 7.1.2. As noted before in (7.21), this last algorithm, when allowing for inexactness, boils down to

$$\hat{x}^{k+1} \approx \operatorname{prox}_{\varphi}^{\Sigma_{\mathrm{x},k}}(\hat{x}^k) \coloneqq \operatorname*{arg\,min}_{x\in\mathbb{R}^n} \left\{ \varphi(x) + \tfrac{1}{2}\|x - \hat{x}^k\|^2_{\Sigma^{-1}_{\mathrm{x},k}} \right\}, \tag{7.27}$$

where $\varphi(x)$ was defined in (7.20). Updates of (7.27) will be shown to converge to KKT-optimal points of (6.1). However, the choice of the primal penalty parameters $\Sigma_{\mathrm{x},k}$ is now crucial. For simplicity, we will assume $\forall k : \Sigma_{\mathrm{x},k} = \Sigma_{\mathrm{x}}$, although the analysis may be extended for a sequence of increasing and bounded penalties. Similarly to what is suggested in [14], by selecting a suitably small weight $\Sigma_{\mathrm{x}} \in \mathrm{Sym}_{++}(\mathbb{R}^n)$, the objective function of the minimization subproblem defining $\operatorname{prox}_{\varphi}^{\Sigma_{\mathrm{x}}}(\hat{x}^k)$ in (7.27) can be made strongly convex. In fact, this is achieved by setting $\forall i : (\Sigma_{\mathrm{x}})_{ii} < \frac{1}{|\lambda_{\min}(Q)|}$, where $\lambda_{\min}(Q)$ is the minimum eigenvalue of $Q$. Of course, when this eigenvalue is larger than zero, the problem is already strongly convex, and so no restrictions are necessary on $\Sigma_{\mathrm{x}}$. How to efficiently find the minimum eigenvalue of a symmetric matrix is discussed later on in Section 8.2.2.

The proximal point iterations of (7.27), after introducing slack variables $z$, amount to solving a nonsmooth composite minimization problem of the form

$$\underset{(x,z)\in\mathbb{R}^{n+m}}{\textbf{minimize}} \quad \tfrac{1}{2}x^\top Q x + \delta_C(z) + \tfrac{1}{2}\|x - \hat{x}^k\|^2_{\Sigma_{\mathrm{x}}^{-1}} \tag{7.28a}$$

$$\textbf{subject to} \quad Ax - z = 0, \tag{7.28b}$$

which itself requires an iterative procedure. Although differing from the original problem (7.3) only by a quadratic term, nevertheless this subproblem has the major advantage of being (strongly) convex and thus amenable to be addressed by means of convex optimization algorithms such as the ALM. The augmented Lagrangian function of problem (7.28) is exactly the proximal augmented Lagrangian of (7.3), since the objective (7.28a) already contains the relevant proximal penalty term. Therefore, starting from a vector $y^k \in \mathbb{R}^m$ and for a given dual weight matrix $\Sigma_{\mathrm{y},k} \in \mathrm{Sym}_{++}(\mathbb{R}^m)$, one iteration of ALM applied to (7.28) produces a triplet $(x^{k+1}, z^{k+1}, y^{k+1})$ according to the following update rule:

$$\begin{cases} (x^{k+1}, z^{k+1}) = \arg\min_{x,z} \mathcal{L}_{\hat{x}^k,\Sigma_{\mathrm{x}},\Sigma_{\mathrm{y},k}}(x, z, y^k) \\ y^{k+1} = y^k + \Sigma_{\mathrm{y},k}(Ax^{k+1} - z^{k+1}). \end{cases} \tag{7.29}$$

Obviously, this iteration is entirely equivalent to the P-ALM iteration proposed in (7.7), except the $\hat{x}$ update. As a result, the same reasoning can be followed as the one to derive (7.14), such that (7.29) can be written as

$$\begin{cases} x^{k+1} = \arg\min_{x\in\mathbb{R}^n} \hat{\varphi}_k(x) \\ z^{k+1} = \Pi_C(Ax^{k+1} + \Sigma_{\mathrm{y},k}^{-1} y^k) \\ y^{k+1} = y^k + \Sigma_{\mathrm{y},k}\big(Ax^{k+1} - z^{k+1}\big), \end{cases} \tag{7.30}$$

with $\hat{\varphi}_k(x)$ defined in (7.11). The update on $\hat{x}$ in this case is only carried out whenever $x^{k+1}, z^{k+1}$ is deemed sufficiently optimal for the subproblem (7.28). The resulting algorithm is outlined in Algorithm 7.1. It is clear that $x^{k+1}$ already satisfies the dual optimality condition from step 2 up to a tolerance, i.e. $\|\nabla\hat{\varphi}_k(x^{k+1})\| \leq \delta_k$, so what remains is to check the primal optimality condition $\|Ax^{k+1} - z^{k+1}\|_{\Sigma_{\mathrm{Y},k}} \leq \varepsilon_k$, as is done in step 5. In this manner, inexact proximal point iterations applied to the original problem may be viewed as modified P-ALM iterations, where the modification lies solely in the $\hat{x}$-update strategy.

**Algorithm 7.1** Modified P-ALM for nonconvex QPs

REQUIRE $(\hat{x}^0, y^0) \in \mathbb{R}^{n+m}$; $\delta_0, \varepsilon_0 > 0$; $\rho \in (0,1)$; $\Sigma_{\mathrm{Y},0} \succeq \Sigma_{\mathrm{Y},\min} \in \mathrm{Sym}_{++}(\mathbb{R}^m)$
$\Sigma_{\mathrm{X}} \in \mathrm{Sym}_{++}(\mathbb{R}^n)$ such that $f + \frac{1}{2}\|\cdot\|^2_{\Sigma_{\mathrm{X}}^{-1}}$ is strongly convex

1: **for** $k = 0, 1, \ldots$ **do**
2: Find $x^{k+1}$ such that $\|\nabla\hat{\varphi}_k(x^{k+1})\| \leq \delta_k$
3: $z^{k+1} = \Pi_C\big(Ax^{k+1} + \Sigma_{\mathrm{Y},k}^{-1} y^k\big)$
4: $y^{k+1} = y^k + \Sigma_{\mathrm{Y},k}(Ax^{k+1} - z^{k+1})$
5: **if** $\|Ax^{k+1} - z^{k+1}\|_{\Sigma_{\mathrm{Y},k}} \leq \varepsilon_k$ **then** ▷ [Quit ALM inner loop]
6: Update $\hat{x}^{k+1} = x^{k+1}$ and choose $\Sigma_{\mathrm{Y},k+1} \succeq \Sigma_{\mathrm{Y},\min}$ and $\varepsilon_{k+1} \leq \rho\varepsilon_k$
7: **else**
8: Set $\hat{x}^{k+1} = \hat{x}^k$ and $\varepsilon_{k+1} = \varepsilon_k$, and choose $\Sigma_{\mathrm{Y},k+1} \succeq \Sigma_{\mathrm{Y},k}$
9: Choose $\delta_{k+1} \leq \rho\delta_k$

Some recent papers [93, 103] developed and analyzed the iteration complexity of a closely related three-layer algorithm, not specifically for QPs, which involves solving a series of quadratically penalized subproblems using inexact proximal point iterations, which are in turn computed using an accelerated composite gradient method. This algorithm also deals with nonconvexity in the objective through the proximal penalty, but is quite different from ours in that it uses a pure quadratic penalty instead of ALM and therefore requires the penalty parameters to go to infinity and is prone to exhibit a slower convergence rate [15, §2.2.5]. The next section discusses the convergence of inexact proximal point iterations in a nonconvex setting.

### 7.3.1 Convergence of inexact PP

We now summarize a key result that was shown in [146, §4.1] in the more general setting of proximal gradient iterations. Given that [146] has not yet been peer-reviewed and also for the sake of self-containedness, we provide a proof tailored to our simplified setting here. The following theorem considers the convergence of inexact proximal point iterations applied to a problem of type (7.19), not restricted to (nonconvex) quadratic programs.

**Theorem 7.3** (Inexact nonconvex PP [146, §4.1])**.** *Let $\varphi : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper, lsc and lower bounded function. Starting from $x^0 \in \mathbb{R}^n$ and given a sequence $\{e^k\}_{k\in\mathbb{N}} \subset \mathbb{R}^n$*

*such that* $\sum_{k\in\mathbb{N}} \|e^k\| < \infty$*, consider the inexact PP iterations*

$$x^{k+1} \in \operatorname{prox}_{\varphi}^{\Sigma_x}(x^k + e^k) \tag{7.31}$$

*for some* $\Sigma_x \in \operatorname{Sym}_{++}(\mathbb{R}^n)$*. Then, the following hold:*

*(i) the real-valued sequence* $\{\varphi(x^{k+1})\}_{k\in\mathbb{N}}$ *converges to a finite value;*

*(ii) the sequence* $\{\|x^{k+1}-x^k\|^2\}_{k\in\mathbb{N}}$ *has finite sum, and in particular* $\min_{i\le k}\|x^{i+1}-x^i\| \le o(1/\sqrt{k})$*;*

*(iii)* $\varphi$ *is constant and equals the limit of* $\{\varphi(x^{k+1})\}_{k\in\mathbb{N}}$ *on the set of cluster points of* $\{x^k\}_{k\in\mathbb{N}}$*, which is made of stationary points for* $\varphi$*;*

*(iv) if* $\varphi$ *is coercive, then* $\{x^k\}_{k\in\mathbb{N}}$ *is bounded.*

*Proof.* The proximal inequality (1.10) with $\bar{x} = x^{k+1}$ and $x' = x^k + e^k$, yields

$$\varphi(x^{k+1}) + \tfrac{1}{2}\|x^{k+1} - x^k - e^k\|^2_{\Sigma_x^{-1}} \le \varphi(x^k) + \tfrac{1}{2}\|e^k\|^2_{\Sigma_x^{-1}}. \tag{7.32}$$

Then, it is possible to bound

$$\begin{aligned}\varphi(x^{k+1}) + \tfrac{1}{4}\|x^{k+1} - x^k\|^2_{\Sigma_x^{-1}} &\le \varphi(x^{k+1}) + \tfrac{1}{2}\|x^{k+1} - x^k - e^k\|^2_{\Sigma_x^{-1}} + \tfrac{1}{2}\|e^k\|^2_{\Sigma_x^{-1}} \\ &\le \varphi(x^k) + \|e^k\|^2_{\Sigma_x^{-1}},\end{aligned} \tag{7.33}$$

where the first inequality follows from the (squared) triangle inequality and the second from (7.32). After bringing $\varphi(x^{k+1})$ to the right-hand side and telescoping (7.33) from $k_1$ to $k_2 \ge k_1$, we obtain the following bound

$$\begin{aligned}\tfrac{1}{4}\sum_{k=k_1}^{k_2} \|x^{k+1} - x^k\|^2_{\Sigma_x^{-1}} &\le \varphi(x^{k_1}) - \varphi(x^{k_2+1}) + \sum_{k=k_1}^{k_2} \|e^k\|^2_{\Sigma_x^{-1}}. \\ &\le \varphi(x^{k_1}) - \inf\varphi + c,\end{aligned} \tag{7.34}$$

where $c = \sum_{k=0}^{\infty} \|e^k\|^2_{\Sigma_x^{-1}} < \infty$. Moreover, since $\varphi$ is proper, $\varphi(x^{k_1})$ (for $k_1 \ge 1$) is finite, such that the right-hand side of (7.34) is finite for any $k_2 \ge k_1$, proving assertion 7.3*(ii)*. Rearranging (7.34) to

$$\varphi(x^{k_2+1}) \le \varphi(x^{k_1}) - \tfrac{1}{4}\sum_{k=k_1}^{k_2} \|x^{k+1} - x^k\|^2_{\Sigma_x^{-1}} + \sum_{k=k_1}^{k_2} \|e^k\|^2_{\Sigma_x^{-1}}, \tag{7.35}$$

it is clear that also $\varphi(x^{k_2+1})$ is bounded by a finite quantity, such that the coerciveness of $\varphi$ immediately implies the boundedness of $x^{k_2+1}$, proving assertion 7.3*(iv)*. Similarly rearranging (7.33) to bound $\varphi(x^{k+1})$, assertion 7.3*(i)* follows by invoking [127, Lem. 2.2.2]. Next, let $\{x^k\}_{k\in K}$ be a subsequence converging to a point $x^\star$; then, it also holds that $\{x^{k+1}\}_{k\in K}$ converges to $x^\star$ owing to assertion 7.3*(ii)*. From the proximal

inequality (1.10) with $\bar{x} = x^{k+1}$ and $x' = x^\star$, we have

$$\varphi(x^{k+1}) + \tfrac{1}{2}\|x^{k+1} - x^k - e^k\|^2_{\Sigma_{\mathrm{x}}^{-1}} \leq \varphi(x^\star) + \tfrac{1}{2}\|x^\star - x^k - e^k\|^2_{\Sigma_{\mathrm{x}}^{-1}},$$

so that passing to the limit for $K \ni k \to \infty$, we get that $\limsup_{k\in K} \varphi(x^{k+1}) \leq \varphi(x^\star)$. In fact, equality holds since $\varphi$ is lsc. Hence from assertion 7.3*(i)* we conclude that $\varphi(x^{k+1}) \to \varphi(x^\star)$ as $k \to \infty$, and in turn from the arbitrarity of $x^\star$ it follows that $\varphi$ is constantly equal to this limit on the whole set of cluster points. To conclude the proof of assertion 7.3*(iii)*, observe that the inclusion $\Sigma_{\mathrm{x}}^{-1}(x^k + e^k - x^{k+1}) \in \hat{\partial}\varphi(x^{k+1})$, cf. (1.11) with $\bar{x} = x^{k+1}$, implies that

$$\begin{aligned}\operatorname{dist}\big(0, \partial\varphi(x^{k+1})\big) &\leq \operatorname{dist}\big(0, \hat{\partial}\varphi(x^{k+1})\big) \\ &\leq \|\Sigma_{\mathrm{x}}^{-1}\|\big(\|x^k - x^{k+1}\| + \|e^k\|\big),\end{aligned} \tag{7.36}$$

and with limiting arguments (recall that $\lim_{k\in K} \varphi(x^k) = \varphi(\lim_{k\in K} x^k)$) the claimed stationarity of the cluster points is obtained. □

Note that with trivial modifications of the proof the arguments also apply to time-varying proximal weights $\Sigma_{\mathrm{x},k}$, $k \in \mathbb{N}$, as long as there exist $\Sigma_{\mathrm{x,min}}, \Sigma_{\mathrm{x,max}} \in \mathrm{Sym}_{++}(\mathbb{R}^n)$ such that $\Sigma_{\mathrm{x,min}} \preceq \Sigma_{\mathrm{x},k} \preceq \Sigma_{\mathrm{x,max}}$ holds for all $k$. Theorem 7.3*(iv)* indicates that coerciveness of the cost function is a sufficient condition for inferring boundedness of the iterates. In (6.1), however, the cost function $\varphi$ may fail to be coercive even if lower bounded on the feasible set $\{x \mid Ax \in C\}$. This happens when there is a feasible direction for which the objective is constant, i.e., when for some $x, d \in \mathbb{R}^n$ with $d \neq 0$ it holds that $\lim_{\tau\to\infty} \delta_C(A(x + \tau d)) = 0$, $Qd = 0$ and $q^\top d = 0$. Nevertheless, it has been shown in [105] that (exact) proximal point iterations on a lower bounded nonconvex quadratic program remain bounded and, in fact, converge to a stationary point.

The next section proves that this remains true even for inexact proximal point iterations, provided that the inexactness vanishes at linear rate. Thereafter, this linear convergence rate is carried over to Algorithm 7.1 by showing how the inexactness used there is related to that in proximal point iterations.

### 7.3.2 Linear convergence for QPs

Before showing the linear convergence rate for inexact proximal point iterations in Theorem 7.6, we present a simple technical lemma regarding the Lipschitz constant of a function on every level set, that will be needed in the proof.

**Lemma 7.4.** *Let $h$ be a lower bounded and $L_h$-smooth function. Then, for every $\alpha > 0$ it holds that*

$$\operatorname{lip}_{\mathrm{lev}_{\leq \inf h + \alpha} h} h \leq \tfrac{1+\sqrt{2}}{2}\sqrt{2\alpha L_h}.$$

*Proof.* Without loss of generality we may assume that $\inf h = 0$. Let $\alpha > 0$ be fixed, and consider $x, y \in \operatorname{lev}_{\leq \alpha} h$ with $x \neq y$. From the quadratic upper bound of Lipschitz differentiable functions (1.2) and the fact that $0 \leq h(x), h(y) \leq \alpha$, we have that

$$\begin{aligned} \frac{|h(y) - h(x)|}{\|y - x\|} &\leq \min\left\{ \tfrac{\alpha}{\|y-x\|}, \|\nabla h(x)\| + \tfrac{L_h}{2}\|y - x\| \right\} \qquad (7.37) \\ &\leq \min\left\{ \tfrac{\alpha}{\|y-x\|}, \sqrt{2\alpha L_h} + \tfrac{L_h}{2}\|y - x\| \right\}, \end{aligned}$$

where the last inequality follows from the fact that

$$\alpha \geq h(x) - h(x - \tfrac{1}{L_h}\nabla h(x)) \geq \tfrac{1}{2L_h}\|\nabla h(x)\|^2,$$

where the last inequality again uses the quadratic lower bound of (1.2). By solving a second-order equation in $\|y - x\|$, we see that

$$\begin{aligned} &\min\left\{ \tfrac{\alpha}{\|y-x\|}, \sqrt{2\alpha L_h} + \tfrac{L_h}{2}\|y - x\| \right\} \\ &\qquad = \begin{cases} \frac{\alpha}{\|y-x\|} & \text{if } \|y - x\| \geq (2 - \sqrt{2})\sqrt{\frac{\alpha}{L_h}}, \\ \sqrt{2\alpha L_h} + \frac{L_h}{2}\|y - x\| & \text{otherwise,} \end{cases} \\ &\qquad \leq \tfrac{\alpha}{2-\sqrt{2}}\sqrt{\tfrac{L_h}{\alpha}} = \tfrac{1+\sqrt{2}}{2}\sqrt{2\alpha L_h}, \end{aligned}$$

resulting in the claimed bound. □

**Remark 7.5.** By discarding the term $\frac{\alpha}{\|y-x\|}$ in (7.37) and letting $(y, x) \to (\bar{x}, \bar{x})$ with $x \neq y$, one obtains that the *pointwise* Lipschitz constant of a lower bounded and $L_h$-smooth function $h$ can be estimated as $\operatorname{lip} h(\bar{x}) \leq \sqrt{2(h(\bar{x}) - \inf h)L_h}$. Therefore, if $h$ is also (quasi-)convex, Lemma 7.4 can be tightened to $\operatorname{lip}_{\operatorname{lev}_{\leq \inf h + \alpha} h} h \leq \sqrt{2\alpha L_h}$, owing to convexity of the sublevel set together with [134, Thm. 9.2]. □

The following proof hinges on the (exact) proximal gradient error bound analysis of [105] and on the close relation existing among proximal point and proximal gradient iterations for this kind of problems.

**Theorem 7.6** (Linear convergence of inexact PP on nonconvex quadratic programs)**.** *Let $\varphi = f + \delta_\Omega$, where $\Omega \subseteq \mathbb{R}^n$ is a nonempty polyhedral set and $f : \mathbb{R}^n \to \mathbb{R}$ is a (possibly nonconvex) quadratic function which is lower bounded on $\Omega$. Starting from $x^0 \in \mathbb{R}^n$ and given a sequence $\{e^k\}_{k\in\mathbb{N}} \subset \mathbb{R}^n$ such that $\|e^k\| \in O(\rho^k)$ for some $\rho \in (0, 1)$, consider the inexact PP iterations (7.31) for some $\Sigma_x \in \operatorname{Sym}_{++}(\mathbb{R}^n)$. Then, the sequence $\{x^k\}_{k\in\mathbb{N}}$ converges at R-linear rate to a stationary point of $\varphi$.*

*Proof.* Let $x^k_\star$ be a projection of $x^k$ onto the set $\operatorname{zer} \partial\varphi$ of stationary points for $\varphi$. Such a point exists for every $k$ owing to nonemptiness and closedness of $\operatorname{zer} \partial\varphi$, the former condition holding by assumption and the latter holding because of closedness of

$\operatorname{gph}\partial\varphi$, cf. [134, Prop. 8.7 and Thm. 5.7(a)]. From [105, Eq.s (2.1) and (A.3)], which can be invoked owing to [105, Thm. 2.1(b)], it follows that there exists $\tau > 0$ such that

$$\varphi(x_\star^k) = \varphi_\star \text{ and } \operatorname{dist}(x^k, \operatorname{zer}\partial\varphi) \le \tau\|x^k - \Pi_\Omega^{\Sigma_X}(x^k - \Sigma_X\nabla f(x^k))\|_{\Sigma_X^{-1}} \tag{7.38}$$

hold for $k$ large enough, where $\Pi_\Omega^{\Sigma_X} = \operatorname{prox}_{\delta_\Omega}^{\Sigma_X}$ is the projection with respect to the distance $\|\cdot\|_{\Sigma_X^{-1}}$. Let $L_f$ and $L_{\varphi^{\Sigma_X}}$ be Lipschitz constants for $\nabla f$ and $\nabla\varphi^{\Sigma_X}$, respectively. Note that stationarity of $x_\star^k$ implies that $\varphi(x_\star^k) = \varphi^{\Sigma_X}(x_\star^k)$ and $\nabla\varphi^{\Sigma_X}(x_\star^k) = 0$. We have

$$\begin{aligned}\varphi^{\Sigma_X}(x^k) - \varphi_\star = \varphi^{\Sigma_X}(x^k) - \varphi^{\Sigma_X}(x_\star^k) &\le \tfrac{L_{\varphi^{\Sigma_X}}}{2}\|x^k - x_\star^k\|^2 = \tfrac{L_{\varphi^{\Sigma_X}}}{2}\operatorname{dist}(x^k, \operatorname{zer}\partial\varphi)^2 \\ &\overset{(7.38)}{\le} \tfrac{L_{\varphi^{\Sigma_X}}\tau^2}{2}\|x^k - \Pi_\Omega^{\Sigma_X}(x^k - \Sigma_X\nabla f(x^k))\|_{\Sigma_X^{-1}}^2. \end{aligned} \tag{7.39}$$

Next, observe that

$$\begin{aligned} x^{k+1} = \operatorname{prox}_\varphi^{\Sigma_X}(x^k + e^k) &\Leftrightarrow \Sigma_X^{-1}(x^k + e^k - x^{k+1}) \in \overbrace{\nabla f(x^{k+1}) + \partial\delta_\Omega(x^{k+1})}^{\partial\varphi(x^{k+1})} \\ &\Leftrightarrow \Sigma_X^{-1}(x^k + e^k - \Sigma_X\nabla f(x^{k+1}) - x^{k+1}) \in \partial\delta_\Omega(x^{k+1}) \\ &\Leftrightarrow x^{k+1} = \operatorname{prox}_{\delta_\Omega}^{\Sigma_X}[x^k + e^k - \Sigma_X\nabla f(x^{k+1})] \\ &\Leftrightarrow x^{k+1} = \Pi_\Omega^{\Sigma_X}[x^k + e^k - \Sigma_X\nabla f(x^{k+1})]. \end{aligned} \tag{7.40}$$

Denoting $c \coloneqq L_{\varphi^{\Sigma_X}}\tau^2$, we obtain through (7.39) and the squared triangle inequality that

$$\begin{aligned}\varphi^{\Sigma_X}(x^k) - \varphi_\star &\le c\|x^k - x^{k+1}\|_{\Sigma_X^{-1}}^2 \\ &\quad + c\Big\|\overbrace{\Pi_\Omega^{\Sigma_X}[x^k + e^k - \Sigma_X\nabla f(x^{k+1})]}^{x^{k+1}} - \Pi_\Omega^{\Sigma_X}[x^k - \Sigma_X\nabla f(x^k)]\Big\|_{\Sigma_X^{-1}}^2, \end{aligned}$$

which, using 1-Lipschitz continuity of $\Pi^{\Sigma_X}$ in the norm $\|\cdot\|_{\Sigma_X^{-1}}$,

$$\begin{aligned} &\le c\|x^k - x^{k+1}\|_{\Sigma_X^{-1}}^2 + c\Big\|x^k + e^k - \Sigma_X\nabla f(x^{k+1}) - x^k + \Sigma_X\nabla f(x^k)\Big\|_{\Sigma_X^{-1}}^2 \\ &\le c\|x^k - x^{k+1}\|_{\Sigma_X^{-1}}^2 + 2c\|e^k\|_{\Sigma_X^{-1}}^2 + 2c\Big\|\nabla f(x^k) - \nabla f(x^{k+1})\Big\|_{\Sigma_X}^2 \\ &\le (c + 2cL_f^2\|\Sigma_X\|)\|x^k - x^{k+1}\|_{\Sigma_X^{-1}}^2 + 2c\|e_k\|_{\Sigma_X^{-1}}^2 \\ &\le c_1\|x^k - x^{k+1}\|_{\Sigma_X^{-1}}^2 + c_2\rho^{2k} \end{aligned} \tag{7.41}$$

for some constants $c_1, c_2 > 0$. Observe that

$$\varphi^{\Sigma_X}(x^{k+1}) \le \varphi(x^{k+1}) = \varphi^{\Sigma_X}(x^k + e^k) - \tfrac{1}{2}\|x^{k+1} - x^k - e^k\|_{\Sigma_X^{-1}}^2$$

$$\leq \varphi^{\Sigma_x}(x^k) + L\|e^k\| - \tfrac{1}{4}\|x^{k+1} - x^k\|^2_{\Sigma_x^{-1}} + \tfrac{1}{2}\|e^k\|^2_{\Sigma_x^{-1}}$$

$$\leq \varphi^{\Sigma_x}(x^k) - \tfrac{1}{4}\|x^{k+1} - x^k\|^2_{\Sigma_x^{-1}} + c_3\rho^k \tag{7.42}$$

for some constant $c_3 > 0$, where in the second inequality $L$ denotes a Lipschitz constant of the smooth function $\varphi^{\Sigma_x}$ on a sublevel set that contains all iterates; the existence of such an $L$ is guaranteed by Theorem 7.3*(i)* and Lemma 7.4, since $-\infty < \inf\varphi \leq \varphi^{\Sigma_x} \leq \varphi$. Therefore,

$$\big(\varphi^{\Sigma_x}(x^k) - \varphi_\star\big) - \big(\varphi^{\Sigma_x}(x^{k+1}) - \varphi_\star\big) \overset{(7.42)}{\geq} \tfrac{1}{4}\|x^{k+1} - x^k\|^2_{\Sigma_x^{-1}} - c_3\rho^k$$

$$\overset{(7.41)}{\geq} \tfrac{1}{4c_1}\big(\varphi^{\Sigma_x}(x^k) - \varphi_\star\big) - c_4\rho^k$$

holds for some constant $c_4 > 0$. By possibly enlarging $c_1$ we may assume without loss of generality that $\rho \geq 1 - 1/4c_1$, so that

$$\big(\varphi^{\Sigma_x}(x^{k+1}) - \varphi_\star\big) \leq \rho\big(\varphi^{\Sigma_x}(x^k) - \varphi_\star\big) + c_4\rho^k$$

$$\leq \rho^{k+1}\big(\varphi^{\Sigma_x}(x^0) - \varphi_\star\big) + c_4\sum_{j=0}^{k}\rho^{k-j}\rho^j$$

$$= \big(\rho(\varphi^{\Sigma_x}(x^0) - \varphi_\star) + c_4(k+1)\big)\rho^k \leq c_5(\sqrt{\rho})^k, \tag{7.43}$$

where $c_5$ is any such that $\big(\rho(\varphi^{\Sigma_x}(x^0) - \varphi_\star) + c_4(k+1)\big)(\sqrt{\rho})^k \leq c_5$ holds for every $k \in \mathbb{N}$. Next, denoting $\varphi_k := \varphi^{\Sigma_x}(x^k) + \frac{c_3}{1-\rho}\rho^k$ observe that

$$\varphi_{k+1} \leq \varphi_k - \tfrac{1}{4}\|x^{k+1} - x^k\|^2_{\Sigma_x^{-1}} \tag{7.44}$$

as it follows from (7.42), and that $\varphi_\star < \varphi_k \to \varphi_\star$ as $k \to \infty$. In fact, (7.43) implies that

$$0 \leq \varphi_k - \varphi_\star \leq c_6\rho^{k/2} \tag{7.45}$$

holds for some $c_6 > 0$ and all $k \in \mathbb{N}$. Therefore,

$$\sum_{j\geq k}\|x^{j+1} - x^j\|_{\Sigma_x^{-1}} \overset{(7.44)}{\leq} 2\sum_{j\geq k}\sqrt{\varphi_j - \varphi_{j+1}} \overset{(7.45)}{\leq} 2\sum_{j\geq k}\sqrt{\varphi_j - \varphi_\star}$$

$$\leq 2c_6^{1/2}\sum_{j\geq k}\rho^{(j-1)/4} \leq c_7\rho^{k/4}$$

for some constant $c_7 > 0$. In particular, the sequence $\{x^k\}_{k\in\mathbb{N}}$ has finite length and thus converges to a point $x^\star$, which is stationary for $\varphi$ owing to Theorem 7.3*(iii)*. In turn, the claimed $R$-linear convergence follows from the inequality $\|x^k - x^\star\|_{\Sigma_x^{-1}} \leq \sum_{j\geq k}\|x^{j+1} - x^j\|_{\Sigma_x^{-1}}$. □

The following theorem relates the inexactness in steps 2 and 5 of Algorithm 7.1 to the inexactness in the proximal point iterations of (7.31), such that the prior convergence claims immediately apply to Algorithm 7.1.

**Theorem 7.7.** *Suppose that problem (7.3) is lower bounded, and consider the iterates generated by Algorithm 7.1 with $f(x) = \frac{1}{2}x^\top Qx + q^\top x$ and $g(z) = \delta_C(z)$. Then, the following hold:*

*(i) The triplet $(x^{k+1}, y^{k+1}, z^{k+1})$ produced at the $k$-th iteration satisfies*

$$\|\nabla f(x^{k+1}) + A^\top y^{k+1}\| \leq \delta_k + \|\Sigma_{\mathrm{x}}^{-1}(x^{k+1} - \hat{x}^k)\| \quad \text{and} \quad y^{k+1} \in \partial g(z^{k+1}).$$

*(ii) The condition at step 5 is satisfied infinitely often, and $\|\hat{x}^{k+1} - \hat{x}^k\| \to 0$ as $k \to \infty$. In particular, for any primal-dual tolerances $\epsilon_{\mathrm{p}}, \epsilon_{\mathrm{d}} > 0$, the termination criteria*

$$\|\nabla f(x^{k+1}) + A^\top y^{k+1}\| \leq \epsilon_{\mathrm{d}} \qquad y^{k+1} \in \partial g(z^{k+1}) \qquad \|Ax^{k+1} - z^{k+1}\| \leq \epsilon_{\mathrm{p}}$$

*are satisfied in a finite number of iterations.*

*(iii) The sequence $\{\hat{x}^k\}_{k\in\mathbb{N}}$ converges to a stationary point of problem (6.1); in fact, denoting $\{k_i\}_{i\in\mathbb{N}}$ as the (infinite) set of those indices at which the condition at step 5 is satisfied, the sequence $\{x^{k_i+1}\}_{i\in\mathbb{N}}$ converges at R-linear rate.*

*Proof.* The definition of $z^{k+1}$ at step 3 and (7.8), together with the characterization of $\operatorname{prox}_g^{\Sigma_{\mathrm{Y},k}^{-1}}$ yield

$$\partial g(z^{k+1}) \ni \nabla g^{\Sigma_{\mathrm{Y},k}^{-1}}(Ax^{k+1} + \Sigma_{\mathrm{Y},k}^{-1}y^k) = \Sigma_{\mathrm{Y},k}\left(Ax^{k+1} + \Sigma_{\mathrm{Y},k}^{-1}y^k - z^{k+1}\right) = y^{k+1}.$$

By expanding the gradient appearing in the norm at step 2 via (7.12), we thus have

$$\delta_k \geq \|\nabla f(x^{k+1}) + A^\top y^{k+1} + \Sigma_{\mathrm{x}}(x^{k+1} - \hat{x}^k)\|, \tag{7.46}$$

and assertion 7.7*(i)* follows from the triangular inequality. Next, observe that whenever the condition at step 5 is not satisfied the variable $\hat{x}^{k+1}$ is not updated (cf. step 8), and thus steps 2–4 amount to ALM iterations applied to the strongly convex problem

$$\begin{aligned} \underset{(x,z)\in\mathbb{R}^{n+m}}{\textbf{minimize}} \quad & f(x) + \tfrac{1}{2}\|x - \hat{x}^k\|^2_{\Sigma_{\mathrm{x}}^{-1}} + g(z) \\ \textbf{subject to} \quad & Ax = z \end{aligned}$$

with a summable inexactness in the computation of the $x$-minimization step. The existence of dual solutions entailed by the strong duality of convex QPs guarantees through [132, Thm. 4 and §6] that the feasibility residual vanishes, hence that eventually step 5 holds.

Let $d^k \coloneqq \nabla f(x^{k+1}) + A^\top y^{k+1} + \Sigma_{\mathrm{x}}(x^{k+1} - \hat{x}^k)$ be the gradient appearing in the norm at step 2 and let $e^k \coloneqq Ax^{k+1} - z^{k+1}$. Let $\{k_i\}_{i\in\mathbb{N}}$ be the (infinite) set of all indices at

which the condition at step 5 is satisfied, so that $\hat{x}^{k_i+1} = x^{k_i+1}$ and $\|e^{k_i}\| \leq \varepsilon_{k_i} \leq \rho^i \varepsilon_0$. Then, for every $i \in \mathbb{N}$

$$\begin{cases} 0 = \nabla f(x^{k_i+1}) + \Sigma_{\mathrm{X}}^{-1}\big(x^{k_i+1} - (x^{k_{i-1}+1} + \Sigma_{\mathrm{X}} d^{k_i})\big) + A^\top y^{k_i+1} \\ 0 \in \partial g(z^{k_i+1}) - y^{k_i+1} \\ 0 = Ax^{k_i+1} - z^{k_i+1} - e^{k_i}. \end{cases}$$

In particular, $(x^{k_i+1}, z^{k_i+1}, y^{k_i+1})$ is a primal-dual solution of

$$\begin{aligned} \underset{(x,z)\in\mathbb{R}^{n+m}}{\textbf{minimize}} \quad & f(x) + g(z) + \tfrac{1}{2}\|x - (x^{k_{i-1}+1} + \Sigma_{\mathrm{X}} d^{k_i})\|^2_{\Sigma_{\mathrm{X}}^{-1}} \\ \textbf{subject to} \quad & Ax - z = e^{k_i}. \end{aligned}$$

Therefore, denoting $\mathcal{X} : \operatorname{dom} \mathcal{X} \subseteq \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ as the operator

$$\mathcal{X}(u, v) = \operatorname*{arg\,min}_{x\in\mathbb{R}^n} \left\{ \tfrac{1}{2} x^\top Q x + q^\top x + \tfrac{1}{2}\|x - u\|^2_{\Sigma_{\mathrm{X}}^{-1}} \mid Ax - v \in C \right\},$$

we have that $x^{k_i+1} = \mathcal{X}(\hat{x}^{k_{i-1}+1} + \Sigma_{\mathrm{X}} d^{k_i}, e^{k_i})$. Notice further that $\mathcal{X}(u, 0) = \operatorname{prox}_{\varphi}^{\Sigma_{\mathrm{X}}}(u)$ for the QP function $\varphi(x) = \frac{1}{2} x^\top Q x + q^\top x + \delta_C(Ax)$. As shown in [125, Thm. 1], $\mathcal{X}$ is a polyhedral mapping, and as it is at most single valued (owing to strong convexity of the QP) we deduce from [47, Cor. 3D.5] that it is globally Lipschitz continuous on its (polyhedral) domain with constant, say, $L$. Therefore,

$$\begin{aligned} \|x^{k_i+1} - \operatorname{prox}_{\varphi}^{\Sigma_{\mathrm{X}}}(x^{k_{i-1}+1})\|^2 &= \|\mathcal{X}(x^{k_{i-1}+1} + \Sigma_{\mathrm{X}} d^{k_i}, e^{k_i}) - \mathcal{X}(\hat{x}^{k_{i-1}+1}, 0)\|^2 \\ &\leq L^2\big(\|\Sigma_{\mathrm{X}} d^{k_i}\|^2 + \|e^{k_i}\|^2\big) \\ &\leq L^2 \|\Sigma_{\mathrm{X}}\|^2 \delta_{k_i}^2 + L^2 \|\Sigma_{\mathrm{Y,min}}\|^{-1} \varepsilon_{k_i}^2 \\ &\leq c\rho^i \end{aligned}$$

for some constant $c > 0$ that only depends on the problem and on the algorithm initialization. Denoting $\xi^i := x^{k_i+1}$ as the $i$-th "outer" iterate, this shows that $\{\xi^i\}_{i\in\mathbb{N}}$ is generated by an inexact proximal point algorithm on function $\varphi$ with error $\|e^i\| \leq O(\rho^i)$, namely,

$$\xi^{i+1} = \operatorname{prox}_{\varphi}^{\Sigma_{\mathrm{X}}}(\xi^i + e^i).$$

In particular, all the assertions follow from Theorems 7.3 and 7.6. □

## 7.4 Summary

This chapter discussed the theoretical background of the (inexact) proximal augmented Lagrangian method and its origin, the proximal point algorithm. A distinction was made in the application to either convex or nonconvex QPs. For the former, the

equivalence between P-ALM and PPA applied to the KKT-operator, along with some monotone operator theory, sufficed to prove the convergence of P-ALM. For the latter, however, although the equivalence still holds, no result was readily available for the hypomonotone KKT-operator. Instead, existing error analyses were used and tailored to the QP setting in order to derive convergence for the slightly modified version of P-ALM that was proposed. The next chapter will focus on the innards of Algorithm 7.1, in particular the minimization strategy used in step 2, parameter settings and updates, and additional features to robustify the method, such as preconditioning of the problem data and infeasibility detection.

# Chapter 8

# The QPALM Algorithm

The previous chapter outlined the overall strategy employed by QPALM, the proximal augmented Lagrangian method, or a slightly modified version for nonconvex QPs. This chapter inspects instead the inner minimization procedure to solve the subproblems. In order to create an effective QP solver, this minimization strategy is constructed to exploit the features of the QP. For example, when using a semismooth Newton method to compute step directions, the linear system does not need to be factorized from scratch every time. Since this linear system changes only slightly between iterations, the factorization may instead be updated using dedicated factorization update routines. Another example is the linesearch. Rather than a typical backtracking linesearch which is common in nonlinear optimization, it is here possible to compute the optimal step size efficiently, since the merit function is a convex piecewise quadratic function. Aside from the inner minimization procedure, key features of a full-fledged solver, such as parameter settings, preconditioning of the problem data, and infeasibility detection are also discussed.

Section 8.1 outlines the inner minimization procedure, consisting of the semismooth Newton direction and exact linesearch. Section 8.2 presents the factorization and update routines to efficiently compute the Newton direction, which were implemented in a C package, LADEL. It mentions also the computation of the minimum eigenvalue of the Hessian matrix, which is performed using the locally optimal block preconditioned conjugate gradient (LOBPCG) method, introduced by Knyazev [91]. Section 8.3 examines effective parameter selection and update routines, as well as a useful preconditioning strategy to minimize the effect of ill conditioning in the problem data. Furthermore, it considers the termination criteria, both for stationary points and infeasibility criteria. Finally, Section 8.4 mixes all the aforementioned ingredients together, presenting the full QPALM algorithm, as well as its default parameter values.

The material in this chapter is based on the publications [84, 85].

## 8.1 Subproblem minimization

This section describes our approach to the inner minimization in step 2, which is clearly the most computationally expensive step of Algorithm 7.1. QPALM uses an iterative method to solve this convex unconstrained optimization problem, $\arg\min_x \hat{\varphi}_k(x)$, computing a semismooth Newton direction and the optimal step size at every iteration.

### 8.1.1 Semismooth Newton method

Recall the function to be minimized, $\hat{\varphi}_k(x)$ and its gradient from (7.11) and (7.12) respectively,

$$\hat{\varphi}_k(x) \coloneqq f(x) + \tfrac{1}{2}\operatorname{dist}^2_{\Sigma_{\mathrm{Y},k}}(Ax + \Sigma_{\mathrm{Y},k}^{-1}y^k, C) + \tfrac{1}{2}\|x - \hat{x}^k\|^2_{\Sigma_{\mathrm{X},k}^{-1}},$$

$$\nabla\hat{\varphi}_k(x) = \nabla f(x) + A^\top\big(y^k + \Sigma_{\mathrm{Y},k}(Ax - Z_k(x))\big) + \Sigma_{\mathrm{X},k}^{-1}(x - x^k),$$

with $Z_k(x)$ from (7.9), which can also be written as

$$\begin{aligned} Z_k(x) &= \Pi_C(Ax + \Sigma_{\mathrm{Y},k}^{-1}y^k) \\ &= Ax + \Sigma_{\mathrm{Y},k}^{-1}y^k + [\ell - Ax - \Sigma_{\mathrm{Y},k}^{-1}y^k]_+ - [Ax + \Sigma_{\mathrm{Y},k}^{-1}y^k - u]_+. \end{aligned} \tag{8.1}$$

Note that $\nabla\hat{\varphi}_k$ also appears in (7.46), with trial point $\tilde{y}^{k+1} = y^k + \Sigma_{\mathrm{Y},k}(Ax - Z_k(x))$. Furthermore, because of the projection operator in $Z_k$, this gradient is not continuously differentiable. However, we can use the generalized Jacobian [51, §7.1] of $\Pi_C$ at $Ax + \Sigma_{\mathrm{Y},k}^{-1}y^k$, one element of which is the diagonal matrix $P_k(x)$ with entries

$$(P_k(x))_{ii} = \begin{cases} 1 & \text{if } \ell_i < (Ax + \Sigma_{\mathrm{Y},k}^{-1}y^k)_i < u_i, \\ 0 & \text{otherwise,} \end{cases}$$

see e.g., [148, §6.2.d]. Therefore, one element of the generalized Hessian of $\varphi_k$ is

$$H_k(x) = Q + A^\top\Sigma_{\mathrm{Y},k}(\mathcal{I} - P_k(x))A + \Sigma_{\mathrm{X}}^{-1}.$$

Denoting the set of *active* constraints as

$$\mathcal{J}_k(x) \coloneqq \big\{i \mid (Ax + \Sigma_{\mathrm{Y},k}^{-1}y^k)_i \notin (\ell_i, u_i)\big\}, \tag{8.2}$$

one has that $(\mathcal{I} - P_k(x))_{ii}$ is equal to 1 if $i \in \mathcal{J}_k(x)$ and 0 otherwise. In the remainder of the text, whenever $\mathcal{J}_k(x)$ is used to indicate a submatrix (in subscript), its dependency on $k$ and $x$ will be omitted for the sake of brevity of notation. $H_k(x)$ can now be written as

$$H_k(x) = Q + A_{\mathcal{J}\cdot}^\top(\Sigma_{\mathrm{Y},k})_{\mathcal{J}\mathcal{J}}A_{\mathcal{J}\cdot} + \Sigma_{\mathrm{X}}^{-1}. \tag{8.3}$$

The semismooth Newton direction $d$ at $x$ satisfies

$$H_k(x)d = -\nabla\varphi_k(x). \tag{8.4}$$

Denoting $\lambda \coloneqq (\Sigma_{\mathrm{Y},k})_{\mathcal{J}\mathcal{J}} A_{\mathcal{J}\cdot} d$, the computation of $d$ is equivalent to solving the following extended linear system

$$\mathcal{K}_k(x) \begin{bmatrix} d \\ \lambda \end{bmatrix} = \begin{bmatrix} Q + \Sigma_{\mathrm{X}}^{-1} & A_{\mathcal{J}\cdot}^\top \\ A_{\mathcal{J}\cdot} & -(\Sigma_{\mathrm{Y},k})_{\mathcal{J}\mathcal{J}}^{-1} \end{bmatrix} \begin{bmatrix} d \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla\varphi_k(x) \\ 0 \end{bmatrix}. \tag{8.5}$$

Finding the solution of either linear system (8.4) or (8.5) in an efficient manner is discussed in further detail in Section 8.2.1.

### 8.1.2 Exact linesearch

Once a suitable direction $d$ has been found, a step size $\tau$ needs to be computed. This is typically done via a linesearch on a suitable merit function. QPALM can compute the optimal step size when using the piecewise quadratic function $\psi(\tau) = \varphi_k(x + \tau d)$ as the merit function. Therefore, using the notation $\langle a, b\rangle = a^\top b$ to denote the scalar product between two vectors $a$ and $b$, finding the optimal step size is equivalent to finding a zero of

$$\begin{aligned}
\psi'(\tau) &= \langle \nabla\varphi_k(x+\tau d), d\rangle \\
&= \langle d, \nabla f(x+\tau d) + \Sigma_{\mathrm{X}}^{-1}(x+\tau d - \hat{x}^k)\rangle + \langle Ad, y^k + \Sigma_{\mathrm{Y},k}\big(A(x+\tau d) - Z_k(x+\tau d)\big)\rangle \\
&= \tau\langle d, (Q + \Sigma_{\mathrm{X}}^{-1})d\rangle + \langle d, Qx + \Sigma_{\mathrm{X}}^{-1}(x - \hat{x}^k) + q\rangle \\
&\quad + \langle \Sigma_{\mathrm{Y},k}Ad, \left[Ax + \Sigma_{\mathrm{Y},k}^{-1}y^k - u + \tau Ad\right]_+\rangle - \langle \Sigma_{\mathrm{Y},k}Ad, \left[\ell - Ax - \Sigma_{\mathrm{Y},k}^{-1}y^k - \tau Ad\right]_+\rangle \\
&= \eta\tau + \beta + \langle \delta, [\delta\tau - \alpha]_+\rangle,
\end{aligned} \tag{8.6}$$

where

$$\begin{cases}
\mathbb{R} \ni \eta \coloneqq \langle d, (Q + \Sigma_{\mathrm{X}}^{-1})d\rangle, \\
\mathbb{R} \ni \beta \coloneqq \langle d, Qx + \Sigma_{\mathrm{X}}^{-1}(x - \hat{x}^k) + q\rangle, \\
\mathbb{R}^{2m} \ni \delta \coloneqq \begin{bmatrix} -\Sigma_{\mathrm{Y},k}^{1/2}Ad & \Sigma_{\mathrm{Y},k}^{1/2}Ad \end{bmatrix}, \\
\mathbb{R}^{2m} \ni \alpha \coloneqq \Sigma_{\mathrm{Y},k}^{-1/2}\begin{bmatrix} y^k + \Sigma_{\mathrm{Y},k}(Ax - \ell) & \Sigma_{\mathrm{Y},k}(u - Ax) - y^k \end{bmatrix}.
\end{cases} \tag{8.7}$$

Here, the notation $\begin{bmatrix} a & b \end{bmatrix}$ is used to denote the vertical stacking of two vectors $a$ and $b$. Note that $\psi'$ is a monotonically increasing piecewise affine function. The root of this function can be found by sorting all the breakpoints, and starting from the origin going through these points $t_i$ until $\psi'(t_i) > 0$. The optimal step size is then in between this and the previous breakpoint and can easily be retrieved by means of interpolation. This procedure is outlined in Algorithm 8.1.

**Algorithm 8.1** Exact linesearch

REQUIRE $x, d \in \mathbb{R}^n$, diagonal $\Sigma \in \mathrm{Sym}_{++}(\mathbb{R}^n)$
PROVIDE optimal stepsize $\tau_\star \in \mathbb{R}$
1: Let $\psi' : \mathbb{R} \to \mathbb{R}$, $\alpha, \beta \in \mathbb{R}$ and $\delta, \eta \in \mathbb{R}^{2m}$ be as in (8.6) and (8.7)
2: Define the set of breakpoints of $\psi'$
$T = \left\{ \frac{\alpha_i}{\delta_i} \mid i = 1, \ldots, 2m,\ \delta_i \neq 0 \right\}$
3: Sort $T = \{t_1, t_2, \ldots\}$ such that $t_i < t_{i+1}$ for all $i$
4: Let $t_i \in T$ be the smallest such that $\psi'(t_i) \geq 0$
5: **return** $\tau_\star = t_{i-1} - \frac{t_i - t_{i-1}}{\psi'(t_i) - \psi'(t_{i-1})} \psi'(t_{i-1})$ (if $i = 1$ then $t_{i-1} = 0$)

The following section discusses the linear algebra ingredients needed in order to efficiently compute solutions to the linear systems in (8.4) or (8.5). Moreover it outlines an efficient algorithm to compute the minimum eigenvalue of $Q$, which is necessary to set the proximal penalty parameter in the nonconvex case.

## 8.2 Linear algebra code

The QPALM algorithm is fully implemented in open-source C code,[2] licensed under the GNU Lesser General Public License version 3 (LGPL v3). The code is standalone, aside from a dependency on LAPACK [3] for computing the minimum eigenvalue. QPALM also provides interfaces to MATLAB, Python and Julia.

This section further discusses the relevant linear algebra used in QPALM, which is implemented in a standalone C package LADEL,[3] also licensed under LGPL v3, and the routine that is used to compute the minimum eigenvalue of a symmetric matrix.

### 8.2.1 Solving linear systems

The most computationally expensive step in one iteration of QPALM is solving the semismooth Newton system (8.4) or (8.5). The matrix $\mathcal{K}_k(x)$ in (8.5), without penalty parameters, is readily recognized as the system of equations that represent the first-order necessary conditions of equality-constrained QPs [121, §16.1], as also given in (6.2), and is therefore dubbed the Karush-Kuhn-Tucker (KKT) matrix. The matrix $H_k(x)$ is the Schur complement of $\mathcal{K}_k(x)$ with respect to the $-(\Sigma_{Y,k})^{-1}_{\mathcal{J}\mathcal{J}}$ block, and is therefore dubbed the Schur matrix. Solving either of the two systems results in a direction along which we can update the primal iterate $x$. The reader is referred to [13] for a broad overview of solution methods for such systems, including direct and

[2] https://github.com/Benny44/QPALM_vLADEL
[3] https://github.com/Benny44/LADEL

iterative methods. In the case of QPALM, the matrix $\mathcal{K}_k(x)$ or $H_k(x)$ is decomposed as the product of a unit lower diagonal matrix $L$, a diagonal matrix $D$ and the transpose of $L$. This is more commonly known as an $LDL^\top$ factorization. The factorization and updates are slightly different for $\mathcal{K}_k(x)$ and $H_k(x)$, so these cases will be discussed separately.

### KKT system

It is not guaranteed that an $LDL^\top$ factorization, with $D$ diagonal, can be found for every matrix. However, because $\Sigma_{\text{Y},k}$ and $Q + \Sigma_{\text{X}}^{-1}$ are symmetric positive definite by construction, $\mathcal{K}_k(x)$ can readily be recognized as a symmetric quasidefinite matrix, which is strongly factorizable [158, Theorem 2.1]. A symmetric matrix $K$ is strongly factorizable if for any symmetric permutation $P$, there exists a unit lower diagonal matrix $L$ and a diagonal matrix $D$ such that $PKP^\top = LDL^\top$. In other words, it should always be possible to find an $LDL^\top$ factorization of $\mathcal{K}_k(x)$ with $D$ diagonal. For this purpose, LADEL has implemented a simple uplooking Cholesky method with separation of the diagonal elements, see [40].

A crucial step in maintaining sparsity during the factorization is to find an effective permutation. Moreover, permutations are sometimes used to prevent ill conditioning. However, finding the optimal permutation is an NP-hard problem [166]. Various heuristics have been developed, an overview of which can be found in [41, §7.7]. LADEL uses the open-source (BSD-3 licensed) implementation[4] of the approximate minimum degree (AMD) ordering algorithm discussed in [2], which promotes sparsity but does not take into account ill conditioning. Nevertheless, no significant numerical issues have been encountered for ill-conditioned QPs, in part due to a careful tuning of the penalty parameters and preconditioning of the problem data, outlined in Section 8.3.

In QPALM, a fill-reducing ordering of the full KKT system, i.e. with $\mathcal{J} = \{1, \dots, m\}$, is computed using AMD once before the first factorization and is used during the remainder of the solve routine. Hence, this permutation minimizes the fill-in of the worst case, that is with all constraints active. In fact, when solving the KKT system, we will not consider $\mathcal{K}_k(x)$ directly, but rather an augmented version

$$\widetilde{\mathcal{K}}_k(x) = \begin{bmatrix} Q + \Sigma_{\text{X}}^{-1} & (A^{\mathcal{J}\cdot})^\top \\ A^{\mathcal{J}\cdot} & -(\Sigma_{\text{Y},k})^{-1} \end{bmatrix}.$$

Note that, as mentioned in (1.2.2), $A^{\mathcal{J}}$ is the $m \times n$ matrix, with $A^{\mathcal{J}}_{j\cdot} = A_{j\cdot}$ if $j \in \mathcal{J}$ and zero otherwise. The size of $\widetilde{\mathcal{K}}_k(x)$ is therefore always $(m+n) \times (m+n)$, but due to (1.2.2) all the inactive constraints give rise to rows and columns that are 0 apart from the diagonal element. Combined with (8.5), it immediately follows that $\lambda_j = 0$ for $j \notin \mathcal{J}_k(x)$.

[4] https://github.com/DrTimothyAldenDavis/SuiteSparse/tree/master/AMD

**Algorithm 8.2** Row addition ([44], with modifications in step 4 and step 5)

REQUIRE $LDL^\top$ factors $L$ and $D$ of $C \in \mathrm{Sym}(\mathbb{R}^n)$ with $C_{\beta\cdot} = C_{\cdot\beta} = 0$ except for $C_{\beta\beta} = \epsilon$. Let $\alpha = 1:\beta-1$ and $\gamma = \beta+1:n$, then

$$LDL^\top = \begin{bmatrix} L_{\alpha\alpha} & & \\ 0 & 1 & \\ L_{\gamma\alpha} & 0 & L_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} D_{\alpha\alpha} & & \\ & d_{\beta\beta} & \\ & & D_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} L^\top_{\alpha\alpha} & & L^\top_{\gamma\alpha} \\ & 1 & 0 \\ & & L^\top_{\gamma\gamma} \end{bmatrix} = \begin{bmatrix} C_{\alpha\alpha} & 0 & C^\top_{\gamma\alpha} \\ 0 & \epsilon & 0 \\ C_{\gamma\alpha} & 0 & C_{\gamma\gamma} \end{bmatrix}.$$

PROVIDE updated $LDL^\top$ factors $\bar{L}$ and $\bar{D}$ of $\bar{C}$ which is $C$ except with the $\beta$-th row and column replaced by $\bar{c}^\top_{\cdot\beta}$ and $\bar{c}_{\cdot\beta}$ respectively, i.e.

$$\bar{L}\bar{D}\bar{L}^\top = \begin{bmatrix} L_{\alpha\alpha} & & \\ \bar{l}^\top_{\alpha\beta} & 1 & \\ L_{\gamma\alpha} & \bar{l}_{\gamma\beta} & \bar{L}_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} D_{\alpha\alpha} & & \\ & \bar{d}_{\beta\beta} & \\ & & \bar{D}_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} L^\top_{\alpha\alpha} & \bar{l}_{\alpha\beta} & L^\top_{\gamma\alpha} \\ & 1 & \bar{l}^\top_{\gamma\beta} \\ & & \bar{L}^\top_{\gamma\gamma} \end{bmatrix} = \begin{bmatrix} C_{\alpha\alpha} & \bar{c}_{\alpha\beta} & C^\top_{\gamma\alpha} \\ \bar{c}^\top_{\alpha\beta} & \bar{c}_{\beta\beta} & \bar{c}^\top_{\gamma\beta} \\ C_{\gamma\alpha} & \bar{c}_{\gamma\beta} & C_{\gamma\gamma} \end{bmatrix}.$$

1: Solve the lower triangular system $L_{\alpha\alpha} D_{\alpha\alpha} \bar{l}_{\alpha\beta} = \bar{c}_{\alpha\beta}$ to find $\bar{l}_{\alpha\beta}$.
2: $\bar{d}_{\beta\beta} = \bar{c}_{\beta\beta} - \bar{l}^\top_{\alpha\beta} D_{\alpha\alpha} \bar{l}_{\alpha\beta}$.
3: $\bar{l}_{\gamma\beta} = \bar{d}^{-1}_{\beta\beta} (\bar{c}_{\gamma\beta} - L_{\gamma\alpha} D_{\alpha\alpha} \bar{l}_{\alpha\beta})$.
4: $w = \bar{l}_{\gamma\beta} \sqrt{|\bar{d}_{\beta\beta}|}$
5: Perform the rank-1 update or downdate $\bar{L}_{\gamma\gamma} \bar{D}_{\gamma\gamma} \bar{L}^\top_{\gamma\gamma} = L_{\gamma\gamma} D_{\gamma\gamma} L^\top_{\gamma\gamma} - \mathrm{sgn}(d_{\beta\beta}) w w^\top$.

Before the condition of step 2 is satisfied, several Newton steps may be required. However, during these iterations $k$ remains constant, and so does $\Sigma_{\mathrm{Y},k}$. Therefore, the only manner in which $\widetilde{\mathcal{K}}_k(x)$ changes is as a result of the change in active constraints when $x$ is updated. Instead of refactorizing the matrix $\widetilde{\mathcal{K}}_k(x)$, we can instead use sparse factorization update routines to update the existing factorization matrices $L$ and $D$. In particular, LADEL has implemented the row addition and row deletion algorithms of [44], with minor modifications to allow for negative diagonal elements (indefinite systems), as outlined in Algorithm 8.2 and Algorithm 8.3, hence the name LDL with Add and DELete update routines (LADEL).

### Schur system

The Schur matrix $H_k(x)$ is symmetric positive definite, since it is the sum of a positive definite matrix $(Q + \Sigma_{\mathrm{X}}^{-1})$ and a positive semidefinite matrix. Therefore, a Cholesky factorization of $H_k(x)$ exists. Furhtermore, when $k$ remains constant and $x$ changes to $x^+$, the difference between $H_k(x^+)$ and $H_k(x)$ is given by

$$H_k(x^+) - H_k(x) = A^\top_{\mathcal{J}^e\cdot} (\Sigma_{\mathrm{Y},k})_{\mathcal{J}^e\mathcal{J}^e} A_{\mathcal{J}^e\cdot} - A^\top_{\mathcal{J}^l\cdot} (\Sigma_{\mathrm{Y},k})_{\mathcal{J}^l\mathcal{J}^l} A_{\mathcal{J}^l\cdot},$$

where $\mathcal{J}^e = \mathcal{J}_k(x^+) \setminus \mathcal{J}_k(x)$ and $\mathcal{J}^l = \mathcal{J}_k(x) \setminus \mathcal{J}_k(x^+)$ are the sets of constraints respectively *entering* and *leaving* the active set. Therefore, two low rank Cholesky factorization updates can be performed [42, 43]. LADEL has implemented the one-rank

**Algorithm 8.3** Row deletion ([44], with modifications in step 4 and step 5)

REQUIRE $LDL^\top$ factors $L$ and $D$ of $C \in \mathrm{Sym}(\mathbb{R}^n)$. Let $\alpha = 1 : \beta - 1$ and $\gamma = \beta + 1 : n$, then

$$LDL^\top = \begin{bmatrix} L_{\alpha\alpha} & & \\ l_{\alpha\beta}^\top & 1 & \\ L_{\gamma\alpha} & l_{\gamma\beta} & L_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} D_{\alpha\alpha} & & \\ & d_{\beta\beta} & \\ & & D_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} L^\top_{\alpha\alpha} & l_{\alpha\beta} & L^\top_{\gamma\alpha} \\ & 1 & l^\top_{\gamma\beta} \\ & & L^\top_{\gamma\gamma} \end{bmatrix} = \begin{bmatrix} C_{\alpha\alpha} & c_{\alpha\beta} & C^\top_{\gamma\alpha} \\ c^\top_{\alpha\beta} & c_{\beta\beta} & c^\top_{\gamma\beta} \\ C_{\gamma\alpha} & c_{\gamma\beta} & C_{\gamma\gamma} \end{bmatrix}.$$

PROVIDE updated $LDL^\top$ factors $\bar{L}$ and $\bar{D}$ of $\bar{C}$ which is equal to $C$ except with the $\beta$-th row and column deleted and the diagonal element $c_{\beta\beta}$ replaced by $\epsilon$, i.e.

$$\bar{L}\bar{D}\bar{L}^\top = \begin{bmatrix} L_{\alpha\alpha} & & \\ 0 & 1 & \\ L_{\gamma\alpha} & 0 & \bar{L}_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} D_{\alpha\alpha} & & \\ & \bar{d}_{\beta\beta} & \\ & & \bar{D}_{\gamma\gamma} \end{bmatrix} \begin{bmatrix} L^\top_{\alpha\alpha} & & L^\top_{\gamma\alpha} \\ & 1 & 0 \\ & & \bar{L}^\top_{\gamma\gamma} \end{bmatrix} = \begin{bmatrix} C_{\alpha\alpha} & 0 & C^\top_{\gamma\alpha} \\ 0 & \epsilon & 0 \\ C_{\gamma\alpha} & 0 & C_{\gamma\gamma} \end{bmatrix}.$$

1: $\bar{l}_{\alpha\beta} = 0$.
2: $\bar{d}_{\beta\beta} = \epsilon$.
3: $\bar{l}_{\gamma\beta} = 0$.
4: $w = l_{\gamma\beta}\sqrt{|d_{\beta\beta}|}$
5: Perform the rank-1 update or downdate $\bar{L}_{\gamma\gamma}\bar{D}_{\gamma\gamma}\bar{L}^\top_{\gamma\gamma} = L_{\gamma\gamma}D_{\gamma\gamma}L^\top_{\gamma\gamma} + \mathrm{sgn}(d_{\beta\beta})ww^\top$.

update routines in [42], which are slightly less efficient than the multiple-rank routines outlined in [43], implemented in CHOLMOD [27].

### Choosing a system

Let $H$ and $\mathcal{K}$ denote the "full" Schur and KKT matrices, that is with $\mathcal{J} = \{1, \ldots, m\}$. In QPALM, it is automatically determined which of these systems to factorize, depending on an estimate of the floating point operations required for each. The work required to compute an $LDL^\top$ factorization is $\sum |L_{\cdot i}|^2$. However, we do not have access to the column counts of the factors before the symbolic factorization. Therefore, a rough estimate of the column counts of the factor is computed using the column counts of the matrices themselves. Moreover, an average column count for each matrix is considered rather than counting the nonzeros in each individual column. As such, QPALM uses the following quantity to determine the choice of linear system:

$$\frac{\sum_{i=1}^{n+m} |L^{\mathcal{K}}_{\cdot i}|^2}{\sum_{i=1}^{n} |L^{H}_{\cdot i}|^2} \approx \frac{\sum_{i=1}^{n+m} |\mathcal{K}_{\cdot i}|^2}{\sum_{i=1}^{n} |H_{\cdot i}|^2} \approx \frac{\sum_{i=1}^{n+m} \left(\frac{|\mathcal{K}|}{n+m}\right)^2}{\sum_{i=1}^{n} \left(\frac{|H|}{n}\right)^2} = \frac{n}{n+m}\frac{|\mathcal{K}|^2}{|H|^2} \approx \frac{n}{n+m}\frac{|\mathcal{K}|^2}{|\widetilde{H}|^2},$$

with $L^{\mathcal{K}}$ and $L^{H}$ the lower diagonal factors of the corresponding matrix. Computing $|H|$ exactly requires the same order of work as computing $H$ itself. Depending on the sparsity pattern of $Q$ and $A$, $H$ can be much denser than $\mathcal{K}$. Hence, we do not want to compute $H$ before choosing between the two systems. Instead, $|H|$ can be further (over)estimated by $|\widetilde{H}|$, which is determined by considering separate contributions from $Q + \Sigma_{\mathrm{x}}^{-1}$ and from $A^{\top}\Sigma_{\mathrm{y},k}A$ to the nonzero pattern. Note that a row in $A$ with $|A_{i\cdot}|$ nonzero elements contributes a block in $A^{\top}A$ with $|A_{i\cdot}|^2$ elements. After discounting the diagonal elements, which are already present in $\Sigma_{\mathrm{x}}^{-1}$, this amount becomes $|A_{i\cdot}|^2 - |A_{i\cdot}|$. The overlap of different elements from different rows of $A$, however, cannot be accounted for (cheaply). Therefore, in our estimate we deduct the minimum (possible) amount of overlap of each block with the biggest block. Let $\hat{\imath}$ denote the row of $A$ with the most nonzero elements, and $\hat{A} = |A_{\hat{\imath}\cdot}| = \max_i(|A_{i\cdot}|)$ this column count. Then, the overlap, again discounting diagonal elements, is given as $[\hat{A} + |A_{i\cdot}| - n]_+^2 - [\hat{A} + |A_{i\cdot}| - n]_+$, and so our estimate for $|H|$ is

$$|\widetilde{H}| = |Q + \Sigma_{\mathrm{x}}^{-1}| + \hat{A}^2 - \hat{A} + \sum_{i \neq \hat{\imath}} |A_{i\cdot}|^2 - |A_{i\cdot}| - [\hat{A} + |A_{i\cdot}| - n]_+^2 + [\hat{A} + |A_{i\cdot}| - n]_+.$$

Finally, Figure 8.1 compares the runtimes of QPALM solving either the KKT or the Schur system applied to the Maros Meszaros test set. Note that the runtime of QPALM using the KKT system can still be much lower than that using the Schur system for an estimated nonzero ratio of 1. This is why a heuristic threshold value of 2, indicated by the red dashed line, is chosen for this ratio such that QPALM by default solves the KKT system for values less than 2 and the Schur system for values higher than 2. The user also has the option to specify beforehand which system to solve.

### 8.2.2 Computing the minimum eigenvalue

Finding the solution of a large symmetric eigenvalue problem has been the topic of a substantial body of research, and many methods exist. They are typically divided into two categories: direct methods, which find all eigenvalues, and iterative methods, which find some (or all) eigenvalues. The reader is referred to [72, §8] for a detailed overview of (the origin) of these methods. In our case, since we only need the minimum eigenvalue of $Q$, iterative methods seem more promising. Of these, the locally optimal block preconditioned conjugate gradient (LOBPCG) method, developed by Knyazev [91], demonstrated the best performance regarding robustness and speed of convergence in our tests. A dedicated implementation of LO(B)PCG to find only the minimum eigenvalue and its corresponding eigenvector was added in QPALM. This method iteratively minimizes the Rayleigh quotient $\frac{x^{\top}Qx}{x^{\top}x}$ in a Krylov subspace spanned by three vectors: the current eigenvector estimate $x^k$, the current residual $w^k = Qx^k - \lambda^k x^k$, and a conjugate gradient direction $p^k$. The details (of the implementation in QPALM) can be found in Algorithm 8.4. The computational cost of this algorithm per iteration is essentially a matrix-vector product and solving a 3-by-3 generalized eigenvalue

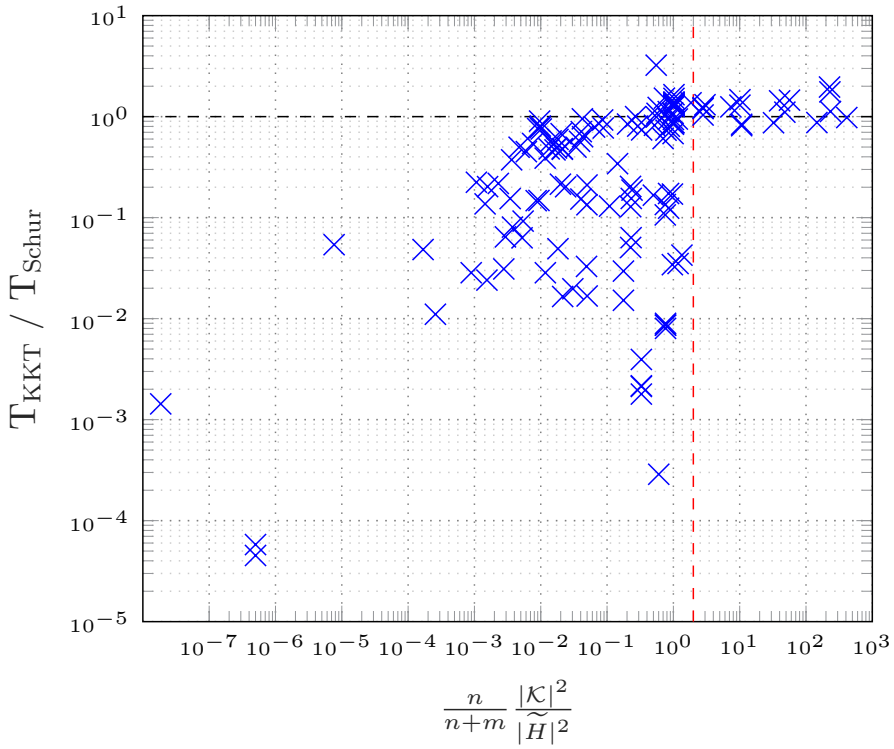



**Figure 8.1:** *Runtime comparison of KKT and Schur complement methods when applying QPALM to the Maros Meszaros test set.*

system. The latter is performed by the relevant routine in LAPACK [3]. Note that Algorithm 8.4 is very similar to [91, Algorithm 4.1], aside from some scaling.

## 8.3 Parameter selection

The technicalities can make or break the practical performance of an algorithm. This section discusses aspects that make QPALM more robust, such as preconditioning of the problem data, and the most important parameter settings. Some of these parameters and parameter update criteria have been tuned manually and some are computed automatically based on the problem data or current iterates. The last subsection also lays out in detail the termination criteria employed by QPALM.

**Algorithm 8.4** LO(B)PCG

REQUIRE $x^0 \in \mathbb{R}^n, \varepsilon > 0$ and $Q \in \mathrm{Sym}(\mathbb{R}^n)$.
PROVIDE Lower bound on $\lambda^* = \lambda_{\min}(Q)$ and estimate of the corresponding eigenvector $x^*$ of $Q$.

Initialize $\lambda^0 = (x^0)^\top A x^0$, and $w^0 = Qx^0 - \lambda^0 x^0$, and let $S = [x^0, w^0]$.
Solve the eigenvalue system $S^\top Q S y = \mu S y$ to find $(\mu, y)$, set $\lambda^1 = \min(\mu)$ and let $\tilde{y}$ denote the corresponding eigenvector.
$x^1 = \tilde{y}_1 x^0 + \tilde{y}_2 w^0$.
$p^1 = \tilde{y}_2 w^0$.
**for** $k = 1, 2, \ldots$ **do**
    $w^k = Qx^k - \lambda^k x^k$.
    **if** $\|w^k\|_2 \leq \varepsilon$ **then**
        Return $\lambda^* = \lambda^k - \|w^k\|_2$ and $x^* = x^k$.
    Let $S = [x^k, w^k, p^k]$.
    Solve the eigenvalue system $S^\top Q S y = \mu S y$ to find $(\mu, y)$, set $\lambda^k = \min(\mu)$ and let $\tilde{y}$ denote the corresponding eigenvector.
    $x^{k+1} = \tilde{y}_1 x^k + \tilde{y}_2 w^k + \tilde{y}_3 p^k$.
    $p^{k+1} = \tilde{y}_2 w^k + \tilde{y}_3 p^k$.

### 8.3.1 Factorization updates

As mentioned in Section 8.2.1, in between Newton iterations the factorization can be updated instead of being recomputed from scratch. However, in practice, the factorization update routines will only be more efficient if the number of constraints entering and leaving the active set is relatively low. Hence, when the active set changes significantly, it is more efficient to recompute the factorization instead. After some experimental tuning, the following rule of thumb to do an update was found

$$|\mathcal{J}^e| + |\mathcal{J}^l| \leq \min(\texttt{max\_rank\_update}, \texttt{max\_rank\_update\_fraction} \cdot (n+m)), \quad (8.8)$$

with `max_rank_update` $= 160$ and `max_rank_update_fraction` $= 0.1$ respectively an absolute and a relative limit on the number of changing active constraints. Both of these parameters can also be set by the user.

### 8.3.2 Preconditioning

Most optimization solvers will perform a scaling of the problem in order to prevent too much ill conditioning. A standard idea in nonlinear optimization is to evaluate the objective and constraints and/or the norm of their gradients at a representative point and scale them respectively with their inverse, see for example [18, §12.5]. However, the quality of this scaling depends on the degree to which the initial point is representative

**Algorithm 8.5** Ruiz equilibration[135]

REQUIRE $A \in \mathbb{R}^{m \times n}$.
PROVIDE $D \in \mathbb{R}^n, E \in \mathbb{R}^m$ and $\bar{A} = EAD$ .
1: Initialize $\bar{A} = A, D = I_n, E = I_m$.
2: **for** $k = 1, \ldots,$ `scaling` **do**
3: **for** $i = 1, \ldots, m$ **do**
4: $\bar{E}_{ii} = \sqrt{\|\hat{A}_{i\cdot}\|_\infty}$
5: **for** $j = 1, \ldots, n$ **do**
6: $\bar{D}_{jj} = \sqrt{\|\bar{A}_{\cdot j}\|_\infty}$
7: $\bar{A} = \bar{E}^{-1}\bar{A}\bar{D}^{-1}$
8: $D = D\bar{D}^{-1}$
9: $E = E\bar{E}^{-1}$

for the iterates, and by extension the solution. Furthermore, since we are dealing with a QP, the constraints and objective are all determined by matrices. Therefore, it makes sense to equilibrate these matrices directly, as is done in OSQP for example [144, §5.1]. OSQP applies a modified Ruiz equilibration [135] to the KKT matrix. This equilibration routine iteratively scales the rows and columns in order to make their infinity norms go to 1. OSQP adds an additional step that scales the objective to take into account also the linear part $q$. We have found in our benchmarks, however, that instead of this scaling it is better to apply Ruiz equilibration to the constraints only, and to scale the objective by a single constant. Why exactly this is the better strategy is unknown to us, but we suspect that the constraints are more sensitive to the scaling, so it might be better to deal with them separately.

In QPALM, the Ruiz equilibration outlined in Algorithm 8.5 is applied to the constraint matrix $A$, yielding $\bar{A} = EAD$. The setting `scaling` denotes the number of scaling iterations which can be set by the user and defaults to 10. The objective is furthermore scaled by $c = \max(1.0, \|D(Qx^0 + q)\|_\infty)^{-1}$. In conclusion, we arrive at a scaled version of (6.1)

$$\begin{aligned} \underset{\bar{x} \in \mathbb{R}^n}{\textbf{minimize}} \quad & \tfrac{1}{2}\bar{x}^\top \bar{Q}\bar{x} + \bar{q}^\top \bar{x} \\ \textbf{subject to} \quad & \bar{A}\bar{x} \in \bar{C}, \end{aligned}$$

with $\bar{x} = D^{-1}x$, $\bar{Q} = cDQD$, $\bar{q} = cDq$, $\bar{A} = EAD$, $\bar{C} = \left\{ z \in \mathbb{R}^m \mid \bar{\ell} \leq z \leq \bar{u} \right\}$, $\bar{\ell} = E\ell$ and $\bar{u} = Eu$. The Lagrange multipliers in this problem are $\bar{y} = cE^{-1}y$. This is the problem that is actually solved in QPALM, although the termination criteria are unscaled, i.e. they apply to the original problem (6.1), see Section 8.3.4.

### 8.3.3 Penalty parameters

The choice of the penalty parameters, and the rules used to update them have been found to be a decisive factor in the performance of QPALM. This section discusses both the traditional penalty parameters arising in the augmented Lagrangian formulation $\Sigma_{\mathrm{Y}}$, and the proximal penalty parameters $\Sigma_{\mathrm{X}}$.

#### Dual penalty parameters

The dual penalty parameters $\Sigma_{\mathrm{Y}}$ play an integral role in slowly but surely enforcing feasibility over the iterations. Because the inner subproblems solved by QPALM are strongly convex, there is no theoretical requirement on the penalty parameters, other than the obvious one of them being positive. However, experience with augmented Lagrangian methods suggests that high values can inhibit convergence initially, as they then introduce ill conditioning in the problem, whereas high values near a solution are very useful to enforce feasibility. As such, these penalty parameters are typically increased during the solve routine depending on the constraint violations of the current iterate. A standard rule is to (only) increase the parameters when the respective constraint violations have not decreased sufficiently, see [121, §17.4]. Furthermore, an added rule in QPALM that is observed to work well is to increase the penalties proportional to their corresponding constraint violation. Hence, we employ the following strategy to find $\Sigma_{\mathrm{Y},k+1}$ in steps 6 and 8 (based on scaled quantities).

$$\frac{(\Sigma_{\mathrm{Y},k+1})_{ii}}{(\Sigma_{\mathrm{Y},k})_{ii}} = \begin{cases} 1.0 & \text{if } i \in \mathcal{D}_k, \\ \min\left[\frac{\sigma_{\max}}{(\Sigma_{\mathrm{Y},k})_{ii}}, \max\left(\boldsymbol{\Delta}\frac{|(\bar{A}\bar{x}^{k+1}-\bar{z}^{k+1})_i|}{\|\bar{A}\bar{x}^{k+1}-\bar{z}^{k+1}\|_\infty}, 1.0\right)\right] & \text{otherwise,} \end{cases} \tag{8.9}$$

with $\mathcal{D}_k = \{i : |(\bar{A}\bar{x}^{k+1} - \bar{z}^{k+1})_i| < \theta|(\bar{A}\bar{x}^k - \bar{z}^k)_i|\}$, the set of indices for which the constraint violation decreased sufficiently. The usage of this rule, in particular letting the factor depend on the constraint violation itself, has been a crucial step in making the performance of QPALM more robust. The default parameters here are $\theta = 0.25$, $\boldsymbol{\Delta} = 100$, and $\sigma_{\max} = 10^9$ and can all be set by the user. Note that in case only a few penalties are modified, the factorization of either $\widetilde{\mathcal{K}}$ or $H$ may be updated using low-rank update routines. In practice, we set the limit on the amount of changing penalties a bit lower than in (8.8) as we expect an additional update to be required for the change in active constraints.

As with regards to an initial choice of penalty parameters, the formula proposed in [18, §12.4] was found to be effective after some tweaking of the parameters inside. As such, the following rule is used in QPALM to determine initial values of the penalties

$$(\Sigma_{\mathrm{Y},0})_{ii} = \max\left[10^{-4}, \min\left(\sigma_{\mathrm{init}}\frac{\max(1.0, |\frac{1}{2}(\bar{x}^0)^\top\bar{Q}\bar{x}^0 + \bar{q}^\top\bar{x}^0|)}{\max(1.0, \frac{1}{2}\|\bar{A}\bar{x}^0 - \Pi_{\bar{C}}(\bar{A}\bar{x}^0)\|^2)}, 10^4\right)\right], \tag{8.10}$$

with $\sigma_{\mathrm{init}}$ a parameter with a default value of 20 and which can also be set by the user. Setting the initial penalty parameters to a high value can be very beneficial when

provided with a good (feasible) initial guess, as therefore feasibility will not be lost. A deeper investigation into this and warm starting QPALM in general is a topic for future work.

### Primal penalty parameters

The primal, or proximal, penalty parameters $\Sigma_{\mathrm{x}}$ serve to regularize the QP around the "current" point $\hat{x}^k$. An appropriate choice makes it so the subproblems are strongly convex, as discussed before. In many problems, the user knows whether the QP will be convex or not. Therefore, QPALM allows the user to indicate which case is dealt with. If the user indicates the problem is (or might be) nonconvex, i.e. that $Q$ is not necessarily positive semidefinite, QPALM uses Algorithm 8.4 to obtain a tight lower bound $\lambda^*$ on the minimum eigenvalue. If this value is negative, we set $\forall i : \Sigma_{\mathrm{x},ii} = \frac{1}{|\lambda^* - 10^{-6}|}$. Otherwise, or in case the user indicates the problem is convex, the default for $\Sigma_{\mathrm{x},ii} = 10^7$, a reasonably low value to not interfere with the convergence speed while guaranteeing that $H_k(x)$ or $\widetilde{\mathcal{K}}_k(x)$ is positive definite or quasidefinite respectively. Furthermore, in the convex case, if the convergence is slow but the primal termination criterion (8.11b) is already satisfied, $\Sigma_{\mathrm{x},ii}$ may be increased even further to $10^{12}$ depending on an estimate of the (machine accuracy) errors that would be accumulated in $H_k(x)$. Finally, QPALM also allows the selection of an initial $(\Sigma_{\mathrm{x},0})_{ii} = \gamma_{\mathrm{init}}$, and an update rule $(\Sigma_{\mathrm{x},k+1})_{ii} = \min(\gamma_{\mathrm{upd}}(\Sigma_{\mathrm{x},k+1})_{ii}, \gamma_{\max})$, but this has not proven particularly beneficial in practice. Not only does it not seem to speed up convergence on average, but every change in $\Sigma_{\mathrm{x}}$ also forces QPALM to refactorize the system.

## 8.3.4 Termination

This section discusses the termination criteria used in QPALM. In addition to the conditions for a stationary point, QPALM may also check certain criteria to determine whether the problem is primal or dual infeasible.

### Stationarity

Termination is based on the unscaled residuals, that is the residuals pertaining to the original problem (6.1). In QPALM, a twofold, absolute and relative, tolerance is used for both the primal and dual residual. As such, the solver is terminated if on an approximate stationary primal-dual pair $(\bar{x}, \bar{y})$, with associated $\bar{z}^k = \Pi_C(\bar{A}\bar{x} + \Sigma_{\mathrm{Y},k}^{-1}\bar{y})$

$$\tfrac{1}{c}\|D^{-1}(\bar{Q}\bar{x} + \bar{q} + \bar{A}^\top\bar{y})\|_\infty \leq \varepsilon_{\mathrm{a}} + \frac{\varepsilon_{\mathrm{r}}}{c}\|\begin{bmatrix} D^{-1}\bar{Q}\bar{x} & D^{-1}\bar{q} & D^{-1}\bar{A}^\top\bar{y}\end{bmatrix}\|_\infty \tag{8.11a}$$

$$\|E^{-1}(\bar{A}\bar{x} - \bar{z}^k)\|_\infty \leq \varepsilon_{\mathrm{a}} + \varepsilon_{\mathrm{r}}\|\begin{bmatrix} E^{-1}\bar{A}\bar{x} & E^{-1}\bar{z}^k\end{bmatrix}\|_\infty. \tag{8.11b}$$

Here, the notation $\mathbb{R}^{2n} \ni z = \begin{bmatrix} x & y \end{bmatrix}$ is again used to indicate the concatenation of two column vectors $x, y \in \mathbb{R}^n$. The tolerances $\varepsilon_{\mathrm{a}}$ and $\varepsilon_{\mathrm{r}}$ are by default $10^{-4}$ and can be chosen by the user. In the simulations of §9, these tolerances were always set to $10^{-6}$.

To determine termination of the subproblem in step 2 of Algorithm 7.1, following (7.46), the termination criterion

$$\frac{1}{c}\|D^{-1}(\bar{Q}\bar{x} + \bar{q} + \Sigma_{\mathrm{x}}^{-1}(\bar{x} - \hat{\bar{x}}^k) + \bar{A}^\top \bar{y})\|_\infty \leq \delta_{\mathrm{a},k} + \frac{\delta_{\mathrm{r},k}}{c}\|\begin{bmatrix} D^{-1}\bar{Q}\bar{x} & D^{-1}\bar{q} & D^{-1}\bar{A}^\top\bar{y} \end{bmatrix}\|_\infty \tag{8.12}$$

is used. Here, the absolute and relative intermediate tolerances $\delta_{\mathrm{a},k}$ and $\delta_{\mathrm{r},k}$ start out from $\delta_{\mathrm{a},0}$ and $\delta_{\mathrm{r},0}$, which can be set by the user and default to $10^0$. In step 9 they are updated using the following rule

$$\delta_{\mathrm{a},k+1} = \max(\rho\delta_{\mathrm{a},k+1}, \varepsilon_{\mathrm{a}}),$$

$$\delta_{\mathrm{r},k+1} = \max(\rho\delta_{\mathrm{r},k+1}, \varepsilon_{\mathrm{r}}),$$

with $\rho$ being the tolerance update factor, which can be set by the user and defaults to $10^{-1}$. Note that, in theory, these intermediate tolerances should not be lower bounded but instead go to zero. In practice, this is however not possible due to machine accuracy errors. Furthermore, we have not perceived any inhibition on the convergence as a result of this lower bound. This makes sense as the inner subproblems are solved up to machine accuracy by the semismooth Newton method as soon as the correct active set is identified.

### Infeasibility detection

Detecting infeasibility of a (convex) QP from the primal and dual iterates has been discussed in the literature [10]. The relevant criteria have also been implemented in QPALM, with a minor modification of the dual infeasibility criterion for a nonconvex QP. As such, it is determined that a problem is (approximately) primal infeasible if for a $\delta\bar{y} \neq 0$ the following two conditions hold

$$\|D^{-1}\bar{A}^\top\delta\bar{y}\|_\infty \leq \varepsilon_{\mathrm{pinf}}\|E\delta\bar{y}\|_\infty, \tag{8.13a}$$

$$\bar{u}^\top[\delta\bar{y}]_+ - \bar{l}^\top[-\delta\bar{y}]_+ \leq -\varepsilon_{\mathrm{pinf}}\|E\delta\bar{y}\|_\infty, \tag{8.13b}$$

with the certificate of primal infeasibility being $\frac{1}{c}E\delta\bar{y}$.

The problem is determined to be (approximately) dual infeasible if for a $\delta\bar{x} \neq 0$

$$(E^{-1}\bar{A}\delta\bar{x})_i \begin{cases} \in [-\varepsilon_{\mathrm{dinf}}, \varepsilon_{\mathrm{dinf}}]\|D\delta\bar{x}\|_\infty & \text{if } \bar{u}_i, \bar{\ell}_i \in \mathbb{R}, \\ \geq -\varepsilon_{\mathrm{dinf}}\|D\delta\bar{x}\|_\infty & \text{if } \bar{u}_i = +\infty, \\ \leq \varepsilon_{\mathrm{dinf}}\|D\delta\bar{x}\|_\infty & \text{if } \bar{\ell}_i = -\infty, \end{cases} \tag{8.14a}$$

holds for all $i \in [1, m] \cap \mathbb{N}$, and either

$$\begin{cases} \|D^{-1}\bar{Q}\delta\bar{x}\|_\infty \leq c\varepsilon_{\text{dinf}}\|D\delta\bar{x}\|_\infty, & \text{(8.14b)} \\ \bar{q}^\top\delta\bar{x} \leq -c\varepsilon_{\text{dinf}}\|D\delta\bar{x}\|_\infty & \text{(8.14c)} \end{cases}$$

or

$$(\delta\bar{x})^\top\bar{Q}\delta\bar{x} \leq -c\varepsilon_{\text{dinf}}^2\|\delta\bar{x}\|^2 \tag{8.14d}$$

hold. Equations (8.14b) and (8.14c) express the original dual infeasibility for convex QPs, that is the conditions that $\delta\bar{x}$ is a direction of zero curvature and negative slope, whereas (8.14d) is added in the nonconvex case to determine whether the obtained $\delta\bar{x}$ is a direction of negative curvature. In the second case, the objective would go to $-\infty$ quadratically along $\delta\bar{x}$, and in the first case only linearly. Therefore, the square of the tolerance, assumed to be smaller than one, is used in (8.14d), so as to allow for earlier detection of this case. Note that minus signs were added in equations (8.13b) and (8.14c) in comparison to [10]. The reason for this is that the interpretation of our tolerance is different. In essence, [10] may declare some problems infeasible even though they are feasible. Our version prevents such false positives at the cost of requiring sufficient infeasibility and possibly a slower detection. We prefer this, however, over incorrectly terminating a problem, as many interesting problems in practice may be close to infeasible. When the tolerances are very close to zero, of course both versions converge to the same criterion. The tolerances $\varepsilon_{\text{pinf}}$ and $\varepsilon_{\text{dinf}}$ can be set by the user and have a default value of $10^{-5}$.

## 8.4 The full QPALM algorithm

Algorithm 8.6 synopsizes all steps and details that make up the QPALM algorithm. Herein we set $\varepsilon_{\text{a},0} = \delta_{\text{a},0}$, $\varepsilon_{\text{r},0} = \delta_{\text{r},0}$. For brevity, the details on factorizations and updates necessary for step 22, which have been discussed prior in Section 8.2.1, have been omitted here.

It is interesting to note that QPALM algorithm presented here differs from its antecedent convex counterpart [84] only by the addition of the lines marked with a star "$\star$", namely for the setting of $\Sigma_{\text{x}}$ and the inner termination criteria. In the convex case, the starred lines are ignored and step 7 and step 28 will always activate. It is clear that the routines in QPALM require minimal changes when extended to nonconvex QPs. Furthermore, in numerical experience with nonconvex QPs the criterion of step 27 seemed to be satisfied most of the time. Therefore, aside from the computation of a lower bound of the minimum eigenvalue of $Q$, QPALM behaves in a very similar manner for convex and for nonconvex QPs. Nevertheless, in practice convergence can be quite a bit slower due to the (necessary) heavy regularization induced by $\Sigma_{\text{x}}$ if $Q$ has a negative eigenvalue with a relatively large magnitude.

Table 8.1 lists the main user settable parameters used in QPALM alongside their default values.

| Name | Default | Description |
|---|---|---|
| $\varepsilon_{\text{a}}$ | $10^{-4}$ | Absolute tolerance on termination criteria |
| $\varepsilon_{\text{r}}$ | $10^{-4}$ | Relative tolerance on termination criteria |
| $\delta_{\text{a},0}$ | $10^{0}$ | Starting value of the absolute intermediate tolerance |
| $\delta_{\text{r},0}$ | $10^{0}$ | Starting value of the relative intermediate tolerance |
| $\rho$ | $10^{-1}$ | Update factor for the intermediate tolerances |
| $\sigma_{\text{init}}$ | 20 | Used in the starting penalty parameters (cf. (8.10)) |
| $\sigma_{\max}$ | $10^{9}$ | Cap on the penalty parameters |
| $\boldsymbol{\Delta}$ | 100 | Factor used in updating the penalties (cf. (8.9)) |
| $\theta$ | 0.25 | Used in penalty update criterion (cf. (8.9)) |
| $\gamma_{\text{init}}$ | $10^{7}$ | Initial proximal penalties (convex) |
| $\gamma_{\text{upd}}$ | 10 | Update factor for the proximal penalties (convex) |
| $\gamma_{\max}$ | $10^{7}$ | Cap on the proximal penalties (convex) |
| `scaling` | 10 | Number of Ruiz scaling iterations applied to $A$ |

**Table 8.1:** *Main parameters used in QPALM and their default values.*

## 8.5 Summary

This chapter presented the innards of the QPALM algorithm. The subproblem minimization was tackled using semismooth Newton directions and optimal step sizes. For the former, efficient $LDL^{\top}$ factorization and update routines were worked out and implemented in LADEL, a C package. For the latter, it sufficed to find the root of a piecewise affine function, which could be done by looping over the breakpoints and interpolating once a zero-crossing has been detected. To optimize the performance of QPALM, a careful investigation was made regarding parameter settings and update routines. The penalty parameters are updated proportionally to the constraint violation, and the proximal penalty parameter was set based on the minimum eigenvalue of $Q$ if it is smaller than 0, and to a small value otherwise. To find this eigenvalue, a simplified version of the LOBPCG method was presented. Furthermore, a tailored preconditioning of the problem data, with a focus on the constraint matrix, was used to prevent ill conditioning. Finally, the termination criteria for approximate stationarity points as well as for the detection of primal and dual infeasibility were laid out. The complete QPALM algorithm is summarized in Algorithm 8.6. The next chapter is dedicated to the performance comparison of QPALM with state-of-the-art QP methods.

**Algorithm 8.6** QPALM for the nonconvex problem (6.1)

REQUIRE Problem data: $Q \in \mathrm{Sym}(\mathbb{R}^n)$; $q \in \mathbb{R}^n$; $A \in \mathbb{R}^{m\times n}$; $\ell, u \in R^m$; $\varepsilon_{\mathrm{a}}, \varepsilon_{\mathrm{r}}, \delta_{\mathrm{a},0}, \delta_{\mathrm{r},0}, \varepsilon_{\mathrm{pinf}}, \varepsilon_{\mathrm{dinf}}, \varepsilon_{\mathrm{a},0}, \varepsilon_{\mathrm{r},0}, \sigma_{\mathrm{init}}, \sigma_{\max}, \gamma > 0$; $\rho, \theta \in (0,1)$; $(x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m$; $\boldsymbol{\Delta} > 1$; `scaling` $\in \mathbb{N}$

Use Algorithm 8.5 to find $D$, $E$, and $c = \max(1.0, \|D(Qx^0+q)\|_\infty)^{-1}$. Convert the data using the scaling factors: $\bar{x}^0 = D^{-1}x^0$, $\bar{y}^0 = cE^{-1}y^0$, $\bar{Q} = cDQD$, $\bar{q} = cDq$, $\bar{A} = EAD$, $\bar{\ell} = E\ell$ and $\bar{u} = Eu$.

Initialize $\hat{\bar{x}}^0 = \bar{x}^0$, $\Sigma_{\mathrm{Y},0}$ from (8.10) and $\delta\bar{x} = 0$.

Compute $\lambda^*$ using Algorithm 8.4.

**if** $\lambda^* < 0$ **then**

  $\Sigma_{\mathrm{X},ii} = \frac{1}{|\lambda^* - 10^{-6}|}, \quad i = 1, \dots, n$

**else**

  $\Sigma_{\mathrm{X},ii} = \gamma, \quad i = 1, \dots, n$

**for** $k = 0, 1, \dots$ **do**

  Set $\bar{x}^{k,0} = \bar{x}^k$.

  **for** $\nu = 0, 1, \dots$ **do**

    $\bar{z}^{k,\nu} = \Pi_{\bar{C}}(\bar{A}\bar{x}^{k,\nu} + \Sigma_{\mathrm{Y},k}^{-1}\bar{y}^k)$.

    $\delta\bar{y} = \Sigma_{\mathrm{Y},k}(\bar{A}\bar{x}^{k,\nu} - \bar{z}^{k,\nu})$

    **if** (8.11) is satisfied at $(\bar{x}^{k,\nu}, \bar{y}^k + \delta\bar{y})$ **then**

      **return** $(\bar{x}^{k,\nu}, \bar{y}^k + \delta\bar{y})$

    **else if** (8.13) is satisfied at $\delta\bar{y}$ **then**

      **return** $c^{-1}E\delta\bar{y}$ as the certificate of primal infeasibility.

    **else if** (8.14) is satisfied at $\delta\bar{x}$ **then**

      **return** $D\delta\bar{x}$ as the certificate of dual infeasibility.

    **else if** (8.12) is satisfied at $(\bar{x}^{k,\nu}, \bar{y}^k + \delta\bar{y})$ **then**

      **break**

    **else**

      Find $d$ by solving either (8.4) or (8.5).

      Find $\tau$ using Algorithm 8.1.

      $\delta\bar{x} = \tau d$.

      $\bar{x}^{k,\nu+1} = \bar{x}^{k,\nu} + \delta\bar{x}$.

  Set $\bar{x}^{k+1} = \bar{x}^{k,\nu}$, $\bar{z}^{k+1} = \bar{z}^{k,\nu}$ and $\bar{y}^{k+1} = \bar{y}^k + \delta\bar{y}$.

  **if** $\|E^{-1}(\bar{A}\bar{x}^{k+1} - \bar{z}^{k+1})\|_\infty \le \varepsilon_{\mathrm{a},k} + \varepsilon_{\mathrm{r},k}\|E^{-1}\begin{bmatrix}\bar{A}\bar{x}^{k+1} & \bar{z}^{k+1}\end{bmatrix}\|_\infty$ **then**

    Update $\hat{\bar{x}}^{k+1} = \bar{x}^{k+1}$

    $\varepsilon_{\mathrm{a},k+1} = \max(\rho\varepsilon_{\mathrm{a},k}, \varepsilon_{\mathrm{a}})$ and $\varepsilon_{\mathrm{r},k+1} = \max(\rho\varepsilon_{\mathrm{r},k}, \varepsilon_{\mathrm{r}})$.

  **else**

    Set $\hat{\bar{x}}^{k+1} = \hat{\bar{x}}^k$, $\varepsilon_{\mathrm{a},k+1} = \varepsilon_{\mathrm{a},k}$ and $\varepsilon_{\mathrm{r},k+1} = \varepsilon_{\mathrm{r},k}$

  Update $\Sigma_{\mathrm{Y},k+1}$ according to (8.9).

  $\delta_{\mathrm{a},k+1} = \max(\rho\delta_{\mathrm{a},k}, \varepsilon_{\mathrm{a}})$ and $\delta_{\mathrm{r},k+1} = \max(\rho\delta_{\mathrm{r},k}, \varepsilon_{\mathrm{r}})$

# Chapter 9

# QPALM: Numerical results

In this chapter, the performance of QPALM is benchmarked against other state-of-the-art solvers. For convex QPs, we chose the interior-point solver Gurobi [79], the active-set solver qpOASES [53], and the operator splitting based solver OSQP [144]. In addition, for MPC problems, the interior-point solver HPIPM [60] was included. There are many other solvers available, but the aforementioned ones provide a good sample of the main methods used for convex QPs. For nonconvex QPs, however, no state-of-the-art open-source (local) optimization solver exists to our knowledge. Some indefinite QP algorithms have been proposed, such as in [1]. However, their solver was found to run into numerical issues very often. Hence, we did not compare against a QP solver specifically, but rather against a state-of-the-art nonlinear optimization solver, IPOPT [161], when dealing with nonconvex QPs. All simulations were performed on a notebook with Intel(R) Core(TM) i7-7600U CPU @ 2.80GHz x 2 processor and 16 GB of memory. The problems are solved to medium-high accuracy, with the termination tolerances $\varepsilon_{\mathrm{a}}, \varepsilon_{\mathrm{r}}$ set to $10^{-6}$ for QPALM. In other solvers, the corresponding termination tolerances were similarly set to $10^{-6}$. Furthermore, for all solvers and all problems, the maximum number of iterations was set to infinity, and a time limit of 3600 seconds was specified.

Section 9.1 discusses some preliminary material on common statistics used to compare performances on large data sets, such as the shifted geometric means and performance profiles. Section 9.2 benchmarks QPALM's performance on convex QPs. Both a popular and comprehensive test set, the Maros-Meszaros collection [108], as well as two specific application examples, portfolio optimization and linear MPC, are considered in this section. Finally, Section 9.3 presents a comparison between QPALM and IPOPT on the nonconvex QPs of the Cutest test set [75].

The material in this chapter is based on the publication [85].

## 9.1 Comparing runtimes

Comparing the performance of solvers on a benchmark test set is not straightforward, and the exact statistics used may influence the resulting conclusions greatly. In this work, runtimes of different solvers on a set of QPs are compared using two measures, the shifted geometric means (sgm) and the performance profiles. When dealing with specific problem classes, such as in Section 9.2.2 and Section 9.2.3, we will not use these statistics but instead make a simple plot of the runtime of the various solvers as a function of the problem dimension.

### 9.1.1 Shifted geometric means

Let $t_{\mathtt{s},\mathtt{p}}$ denote the time required for solver $\mathtt{s}$ to solve problem $\mathtt{p}$. Then the shifted geometric means $\bar{t}_{\mathtt{s}}$ of the runtimes for solver $\mathtt{s}$ on problem set $\mathtt{P}$ is defined as

$$\bar{t}_{\mathtt{s}} = \sqrt[|\mathtt{P}|]{\prod_{\mathtt{p}\in\mathtt{P}}(t_{\mathtt{s},\mathtt{p}}+\zeta)} - \zeta = e^{\frac{1}{|\mathtt{P}|}\sum_{\mathtt{p}\in\mathtt{P}}\ln(t_{\mathtt{s},\mathtt{p}}+\zeta)} - \zeta,$$

where the second formulation is used in practice to prevent overflow when computing the product. In this paper, runtimes are expressed in seconds, and a shift of $\zeta = 1$ is used. Also note that we adopt the convention that when a solver $\mathtt{s}$ fails to solve a problem $\mathtt{p}$ (within the time limit), the corresponding $t_{\mathtt{s},\mathtt{p}}$ is set to the time limit for the computation of the sgm.

### 9.1.2 Performance profile

To compare the runtime performance in more detail, also performance profiles [46] are used. Such a performance profile plots the fraction of problems solved within a runtime of $f$ times the runtime of the fastest solver for that problem. Let $\mathtt{S}$ be the set of solvers tested, then

$$r_{\mathtt{s},\mathtt{p}} = \frac{t_{\mathtt{s},\mathtt{p}}}{\min_{\bar{\mathtt{s}}\in\mathtt{S}} t_{\bar{\mathtt{s}},\mathtt{p}}},$$

denotes the performance ratio of solver $\mathtt{s}$ with respect to problem $\mathtt{p}$. By convention $r_{\mathtt{s},\mathtt{p}}$ is set to $\infty$ when $\mathtt{s}$ fails to solve $\mathtt{p}$ (within the time limit). The fraction of problems $q_{\mathtt{s}}(f)$ solved by $\mathtt{s}$ to within a multiple $f$ of the best runtime, is then given as

$$q_{\mathtt{s}}(f) = \frac{1}{|\mathtt{P}|}\sum_{\mathtt{P}\ni\mathtt{p}:r_{\mathtt{s},\mathtt{p}}\leq f} 1.$$

Performance profiles have been found to misrepresent the performance when more than two solvers were compared at the same time [76]. As such, we will construct only

the performance profile of each other solver and QPALM, and abstain from comparing the other solvers amongst each other.

In the next section, these tools are used to benchmark the performance of QPALM against state-of-the-art convex QP solvers on the Maros-Meszaros test set, alongside other benchmarks for specific application examples.

## 9.2 Convex QPs

As mentioned in Section 6.2, convex quadratic programming solvers typically rely on active-set, interior-point or proximal methods, and many solvers exist. In this section, QPALM is compared against one representative from each branch, in particular the interior-point solver Gurobi [79], the active-set solver qpOASES [53] and the operator splitting based solver OSQP [144]. First and foremost, all solvers are compared on the Maros-Meszaros benchmark test set [108], a well-known collection of convex QPs drawn from a broad range of application domains. However, qpOASES is excluded in this comparison as it tends to fail on larger problems which are ubiquitous in this set. Thereafter, the performance of all solvers is also compared for some structured quadratic problems arising from two specific application domains, portfolio optimization and MPC. For this last problem, the performance is compared also to HPIPM [60], an interior-point solver tailored to small-to-medium dense and OCP-structured problems. This solver was run in its `balanced` mode, since the `speed` mode version often did not converge.

### 9.2.1 Maros Meszaros

The Maros-Meszaros test set contains 138 convex quadratic programs, and is often used to benchmark convex QP solvers. Table 9.1 lists the shifted geometric mean of the runtime and failure rate of QPALM, OSQP and Gurobi applied to this set. The Maros-Meszaros set includes many large-scale and ill conditioned QPs, and the fact that QPALM succeeds in solving all of them within one hour is a clear indication that it is very robust with respect to ill conditioning of the problem data. In runtime it is also faster on average than the other solvers. However, Gurobi is faster more often, as is shown in the performance profiles in Figure 9.1. The high shifted geometric mean runtime of Gurobi is mostly due to its relatively high failure rate. OSQP also has a high failure rate, and is also slower than QPALM, both on average and in frequency. As a first-order method, in spite of employing a similar preconditioning routine to ours, it seems to still exhibit a lack of robustness with respect to ill conditioning and to somewhat stricter tolerance requirements.

| | QPALM | OSQP | Gurobi |
|---|---|---|---|
| Runtime (sgm) | 0.8189 | 7.1098 | 1.2018 |
| Failure rate [%] | 0.0000 | 9.4203 | 7.9710 |

**Table 9.1:** *Shifted geometric mean runtime and failure rate for QPALM, OSQP and Gurobi on the Maros Meszaros problem set.*

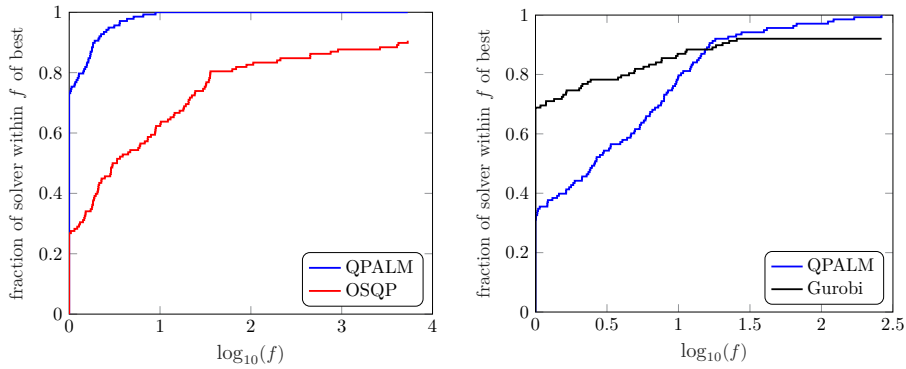



**Figure 9.1:** *Performance profiles comparing QPALM with OSQP and Gurobi respectively on the Maros Meszaros problem set.*

### 9.2.2 Portfolio

As mentioned in Section 6.1.1, the goal in portfolio optimization is to select a portfolio of assets to invest in to maximize profit taking into account risk levels. Given a vector $x$ denoting the (relative) investment in each asset, the resulting quadratic program is the following

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\textbf{minimize}} \quad & \kappa x^\top \Sigma x - \mu^\top x \\ \textbf{subject to} \quad & x \geq 0, \\ & \sum_{i=1}^{n} x_i = 1, \end{aligned}$$

with $\mu \in \mathbb{R}^n$ a vector of expected returns, $\Sigma \in \mathrm{Sym}(\mathbb{R}^n)$ a covariance matrix representing the risk and $\kappa > 0$ a parameter to adjust the aversion to risk. Typically, $\Sigma = FF^\top + D$, with $F \in \mathbb{R}^{n\times r}$ a low rank matrix and $D \in \mathbb{R}^{n\times n}$ a diagonal matrix. In order not to form the matrix $\Sigma$, the following reformulated problem can be solved

instead in $(x, y)$

$$\begin{aligned} \underset{(x,y)\in\mathbb{R}^{n+r}}{\textbf{minimize}} \quad & \begin{bmatrix} x \\ y \end{bmatrix}^\top \begin{bmatrix} D & \\ & I_r \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \kappa^{-1}\mu^\top x \\ \textbf{subject to} \quad & x \geq 0, \\ & \sum_{i=1}^{n} x_i = 1, \\ & y = F^\top x. \end{aligned}$$

We solved this problem for values of $n$ ranging from 100 to 1000, with $r = \lceil \frac{n}{10} \rceil$. We choose the elements of $\mu$ uniformly on $[0, 1]$, the diagonal elements $D_{ii}$ uniformly on the interval $[0, \sqrt{r}]$, and the matrix $F$ has 50% nonzeros drawn from $\mathcal{N}(0, 1)$. For each value of $n$, we solve the problem for five values of $\kappa$, $(10^{-2}, 10^{-1}, 1, 10^1, 10^2)$, and compute the arithmetic mean of the runtimes. The runtimes of QPALM, OSQP, Gurobi and qpOASES solving these problems as such for different values of $n$ are shown in Figure 9.2. The structure of the portfolio optimization problem is quite specific: the Hessian of the objective is diagonal, and the only inequality constraints are bound constraints. It is clear from the figure that Gurobi exhibits the lowest runtimes for this type of problem, followed closely by QPALM and OSQP. The latter performs well especially for the small problems and has some robustness issues for larger ones. It seems that qpOASES exhibits quite a high runtime when compared to the other solvers.

### 9.2.3 MPC

As explained in Section 6.1.2, model predictive control is a strategy wherein one solves an optimal control problem (OCP) at every sample time to determine the optimal control inputs that need to be applied to a system. The OCP considers a control horizon $N$, that is, it computes a series of $N$ inputs, of which only the first is applied to the system. Given a discrete linear system with $n_q$ states $q$ and $n_u$ inputs $u$, and its corresponding system dynamics in state-space form, $q_{k+1} = Aq_k + Bu_k$, the OCP considered here is one where we control the system from an initial state $\bar{q}$ to the reference state, for simplicity assumed to be at the origin, which can be formulated as

$$\begin{aligned} \underset{z\in\mathbb{R}^{(N+1)n_q+Nn_u}}{\textbf{minimize}} \quad & q_N^\top Q_N q_N + \sum_{k=0}^{N-1} q_k^\top Q\, q_k + u_k^\top R u_k \\ \textbf{subject to} \quad & q_0 = \bar{q}, \\ & q_{k+1} = Aq_k + Bu_k, \quad k = 0, \ldots, N-1, \end{aligned}$$

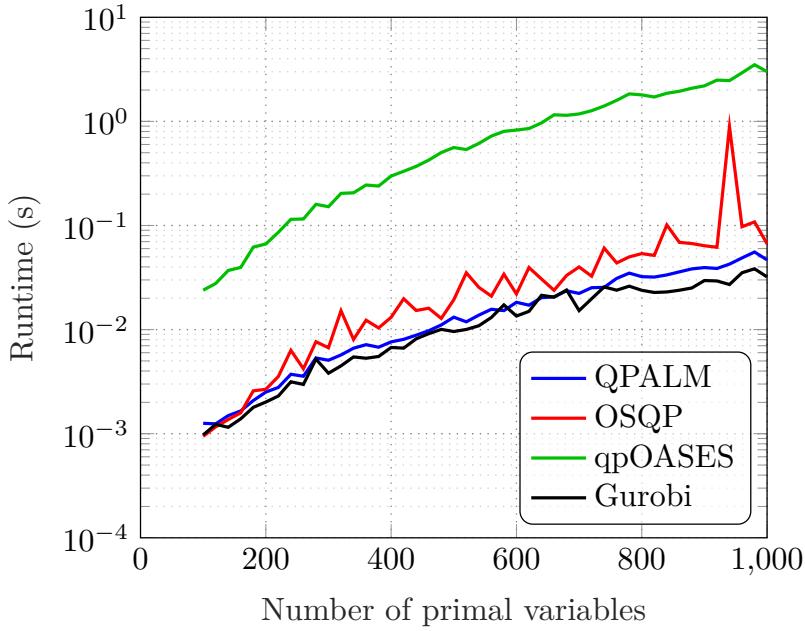



**Figure 9.2:** *Runtimes of QPALM, OSQP, qpOASES and Gurobi when solving portfolio optimization problems of varying sizes.*

$$
\begin{aligned}
&q_k \in \mathcal{Q}, && k = 0, \dots, N-1, \\
&q_N \in \mathcal{Q}_N, && \\
&u_k \in \mathcal{U}, && k = 0, \dots, N-1.
\end{aligned}
$$

Here, the decision variable is the collection of $N+1$ state samples and $N$ input samples, $z = (q_0, u_0, q_1, \dots, u_{N-1}, q_N)$. The stage and terminal state cost matrices are positive definite matrices, $Q, Q_N \in \mathrm{Sym}_{++}(\mathbb{R}^{n_q})$ and $R \in \mathrm{Sym}_{++}(\mathbb{R}^{n_u})$. $\mathcal{Q}$, $\mathcal{Q}_N$ and $\mathcal{U}$ represent polyhedral constraints on the states, terminal state and inputs respectively. In our example, these are taken as box constraints $\mathcal{Q} = [-q_b, q_b]$ and $\mathcal{U} = [-u_b, u_b]$ and the terminal constraint is determined as the maximal control invariant set [94] of the system. Furthermore, the terminal cost is computed from the discrete-time algebraic Riccati equations.

We solved this problem for a system with 10 states and 5 inputs for different values of the time horizon. The state cost matrix is set as $Q = MM^\top$, with $M \in \mathbb{R}^{n_q \times n_q}$ consisting of 50% nonzeros drawn from the normal distribution $\mathcal{N}(0, 5)$. The input cost matrix is chosen to be a small diagonal matrix with $R_{ii} = 0.01$. The system considered is slightly unstable, with the elements of $A$ drawn from $\mathcal{N}(0, 2)$ and those

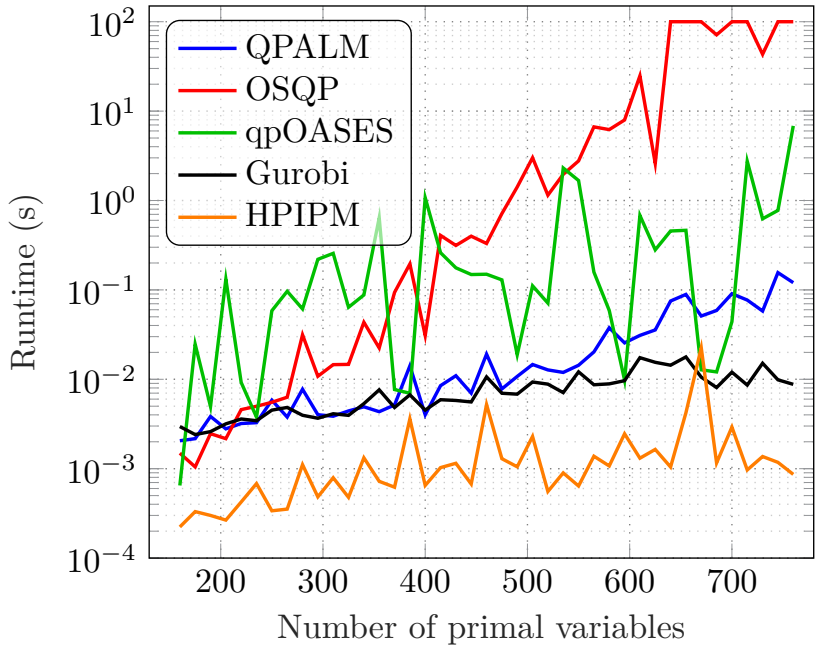



**Figure 9.3:** *Runtimes of QPALM, OSQP, qpOASES, Gurobi and HPIPM when solving OCPs for varying time horizons.*

of $B$ from $\mathcal{N}(0, 1)$. The state and input limits $q_b$ and $u_b$ are drawn from $\mathcal{N}(10, 2)$. Finally, the initial state is chosen such that it is possible but difficult to satisfy all the constraints, in order to represent a challenging MPC problem. The resulting runtimes of solving one such OCP for varying time horizons are shown in Figure 9.3. HPIPM performs best, as expected from a tailored solver, followed by Gurobi and QPALM. OSQP and qpOASES both have issues with robustness given the challenging nature of the problem, although the latter also exhibits fast convergence in some cases.

An important aspect to consider when choosing a QP solver for MPC is the degree to which it can work with an initial guess. This is of great import due to the fact that subsequent OCPs are very similar. The solution of the previous OCP can therefore be shifted by one sample time and supplied as an initial guess. This procedure is also called warm starting. Figure 9.4 shows the result of warm starting subsequent OCPs in this manner. Here, we solved 30 subsequent OCPs for a fixed time horizon of 30, corresponding to 460 primal variables. Furthermore, a small disturbance, drawn from the normal distribution $\mathcal{N}(0, 0.01)$, is applied when computing the next initial state. It is clear that qpOASES, QPALM and OSQP all benefit greatly from this warm starting. However, Gurobi, as is typical of an interior-point method, does not have this advantage. For this reason, interior-point methods are typically not considered as

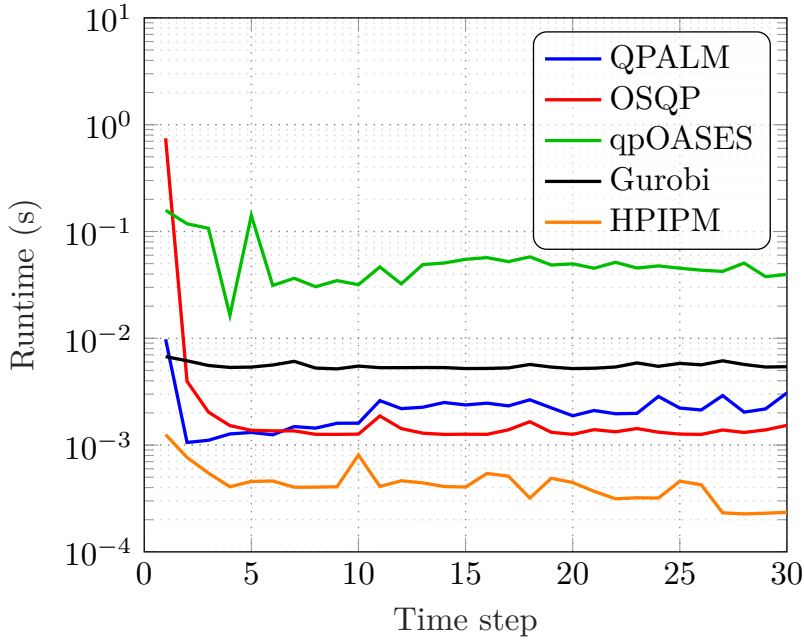



**Figure 9.4:** *Runtimes of QPALM, OSQP, qpOASES, Gurobi and HPIPM when solving sequential OCPs in an MPC setting, with* $N = 30$.

solvers for MPC problems. HPIPM, however, has incredibly low runtimes regardless, and so may be an excellent choice for optimal control problems.

## 9.3 Nonconvex QPs

As mentioned in Section 6.3, nonconvex QPs arise in several application domains, such as in mixed integer quadratic programs, partial differential equations and VLSI chip design [163]. Successive (potentially) indefinite QPs have to be solved in an SQP method applied to a nonconvex optimization problem if the exact Hessian of the Lagrangian is used. To have a broad range of sample QPs, this work considers the set of nonconvex QPs included in the Cutest test framework [75], since these problems span various application domains and also include many ill conditioned and/or large-scale problems that may be used to validate the robustness of the solvers.

Table 9.2 lists for each of those QPs the number of primal variables $n$ and the number of constraints $m$, excluding bound constraints. In addition, it lists a comparison of the runtime and final objective value for both QPALM and IPOPT. Given that both

solvers only produce an (approximate) stationary point, and not necessarily the same, these results have been further analyzed to produce Table 9.3. Here, the problems have been divided according to whether both solvers converged to the same point or not, the criterion of which was set to a relative error on the primal solutions of $10^{-6}$.

On the one hand, the runtimes of the problems where the same solution was found have been listed as shifted geometric means. It is clear that QPALM was on average a bit faster for these problems. These runtimes were further compared in the performance profile of Figure 9.5. This shows that QPALM was not only faster on average, but also more often the fastest of the two solvers. On the other hand, for the problems with different solutions, the objective value of the solution was compared and the number of times either QPALM or IPOPT had the lowest objective was counted. The resulting tally of 45 against 40 in favour of QPALM suggests there is no clear winner in this case. This was to be expected as both solvers report on the first stationary point obtained, and neither uses globalization or restarting procedures to obtain a better one.

Finally, also the failure rate was reported. It is clear that QPALM outperforms IPOPT by a small margin. Furthermore, the six problems that QPALM failed to solve within the time limit, that is NCVXQP{1-3,7-9}, IPOPT also failed to solve in time. IPOPT reported two of the problems, A2NNDNIL and A5NNDNIL, as primal infeasible, whereas for these problems QPALM found a point satisfying the approximate stationary conditions. In fact, the problems are primal infeasible, and QPALM also reports this once slightly stricter termination tolerances are enforced. The difference in result stems from a slightly different interpretation of the termination criteria and corresponding tolerances. For these reasons, both solvers are considered to have succeeded for these two cases.

| | | | Runtime | | Objective | |
|---|---|---|---|---|---|---|
| Problem | n | m | QPALM | IPOPT | QPALM | IPOPT |
| A0ENDNDL | 45006 | 15002 | 1.14e+01 | 1.75e+00 | -3.87e-05 | 1.84e-04 |
| A0ENINDL | 45006 | 15002 | 9.63e+00 | 1.58e+00 | -1.63e-04 | 1.84e-04 |
| A0ENSNDL | 45006 | 15002 | 5.56e+00 | 2.54e+01 | -2.55e-07 | 1.48e-04 |
| A0ESDNDL | 45006 | 15002 | 1.14e+01 | 1.57e+00 | -7.29e-05 | 1.84e-04 |
| A0ESINDL | 45006 | 15002 | 1.05e+01 | 1.63e+00 | -7.08e-07 | 1.84e-04 |
| A0ESSNDL | 45006 | 15002 | 4.67e+00 | 2.54e+01 | -5.82e-06 | 1.48e-04 |
| A0NNDNDL | 60012 | 20004 | 2.09e+02 | 4.65e+00 | -3.17e-05 | 1.84e-04 |
| A0NNDNIL | 60012 | 20004 | 1.95e+03 | 2.38e+01 | 7.08e-01 | 1.98e-04 |
| A0NNDNSL | 60012 | 20004 | 6.73e+01 | 1.81e+01 | 5.35e-04 | 1.68e-04 |
| A0NNSNSL | 60012 | 20004 | 1.86e+01 | 2.96e+01 | -1.25e-04 | 1.54e-04 |
| A0NSDSDL | 60012 | 20004 | 3.22e+01 | 4.12e+00 | -1.89e-05 | 1.84e-04 |
| A0NSDSDS | 6012 | 2004 | 1.50e+00 | 9.12e-01 | 2.35e-06 | 4.91e-04 |
| A0NSDSIL | 60012 | 20004 | 1.00e+03 | 2.52e+01 | 6.47e-05 | 1.97e-04 |
| A0NSDSSL | 60012 | 20004 | 3.21e+01 | 1.23e+01 | -2.86e-06 | 1.67e-04 |
| A0NSSSSL | 60012 | 20004 | 1.82e+01 | 2.70e+01 | -6.44e-05 | 1.49e-04 |
| A2ENDNDL | 45006 | 15002 | 2.77e+01 | 2.32e+00 | 9.96e-07 | 9.88e-04 |
| A2ENINDL | 45006 | 15002 | 2.65e+01 | 2.38e+00 | 8.50e-07 | 9.73e-04 |
| A2ENSNDL | 45006 | 15002 | 4.98e+00 | 4.29e+01 | 1.91e-06 | 2.10e-02 |
| A2ESDNDL | 45006 | 15002 | 2.86e+01 | 2.29e+00 | 8.47e-07 | 9.88e-04 |
| A2ESINDL | 45006 | 15002 | 2.58e+01 | 2.28e+00 | 2.45e-07 | 9.73e-04 |

| | | | | | | |
|---|---|---|---|---|---|---|
| A2ESSNDL | 45006 | 15002 | 4.78e+00 | 4.33e+01 | 1.02e-06 | 2.10e-02 |
| A2NNDNDL | 60012 | 20004 | 1.52e+03 | 6.34e+00 | 3.46e-04 | 3.03e-04 |
| A2NNDNIL | 60012 | 20004 | 3.51e+01 | PI | 6.45e+01 | / |
| A2NNDNSL | 60012 | 20004 | 3.81e+02 | 5.38e+01 | -5.54e-06 | 2.02e-04 |
| A2NNSNSL | 60012 | 20004 | 3.17e+01 | 5.16e+01 | 1.99e-05 | 2.01e-02 |
| A2NSDSDL | 60012 | 20004 | 6.19e+01 | 5.14e+00 | 3.57e-06 | 7.76e-04 |
| A2NSDSIL | 60012 | 20004 | 4.08e+01 | 3.76e+01 | 1.07e+01 | 1.57e+00 |
| A2NSDSSL | 60012 | 20004 | 5.23e+01 | 1.85e+01 | -5.97e-07 | 2.51e-02 |
| A2NSSSSL | 60012 | 20004 | 3.41e+01 | 3.04e+01 | -5.79e-06 | 4.70e-04 |
| A5ENDNDL | 45006 | 15002 | 6.85e+01 | 2.29e+00 | 8.14e-07 | 2.19e-03 |
| A5ENINDL | 45006 | 15002 | 7.33e+01 | 2.33e+00 | -5.88e-07 | 2.25e-03 |
| A5ENSNDL | 45006 | 15002 | 6.62e+00 | 4.97e+01 | 1.74e-05 | 5.12e-02 |
| A5ESDNDL | 45006 | 15002 | 6.94e+01 | 3.23e+00 | 1.10e-06 | 2.19e-03 |
| A5ESINDL | 45006 | 15002 | 6.97e+01 | 2.70e+00 | 5.33e-07 | 2.25e-03 |
| A5ESSNDL | 45006 | 15002 | 8.21e+00 | 4.95e+01 | 3.30e-05 | 5.12e-02 |
| A5NNDNDL | 60012 | 20004 | 2.56e+03 | 8.07e+00 | 5.10e-04 | 1.86e-03 |
| A5NNDNIL | 60012 | 20004 | 3.50e+01 | PI | 1.02e+02 | / |
| A5NNDNSL | 60012 | 20004 | 2.43e+02 | 2.08e+01 | 3.32e-06 | 7.44e-02 |
| A5NNSNSL | 60012 | 20004 | 3.47e+01 | 1.53e+02 | 5.11e-06 | 2.47e-02 |
| A5NSDSDL | 60012 | 20004 | 9.59e+01 | 4.84e+00 | 3.53e-06 | 1.86e-03 |
| A5NSDSDM | 6012 | 2004 | 1.58e+00 | 8.17e-01 | 5.28e-07 | 4.91e-04 |
| A5NSDSIL | 60012 | 20004 | 3.43e+01 | 1.18e+02 | 8.43e+00 | 1.27e+00 |
| A5NSDSSL | 60012 | 20004 | 4.85e+01 | 1.84e+01 | 3.69e-05 | 1.00e-02 |
| A5NSSNSM | 6012 | 2004 | 1.56e+00 | 8.29e-01 | 1.30e-06 | 4.91e-04 |
| A5NSSSSL | 60012 | 20004 | 3.34e+01 | 1.11e+02 | -3.29e-06 | 1.73e-02 |
| BIGGSC4 | 4 | 7 | 3.52e-04 | 5.67e-02 | -2.45e+01 | -2.45e+01 |
| BLOCKQP1 | 10010 | 5001 | 1.72e-01 | 1.22e+01 | -4.99e+03 | -4.99e+03 |
| BLOCKQP2 | 10010 | 5001 | 1.55e-01 | 1.88e+00 | -4.99e+03 | -4.99e+03 |
| BLOCKQP3 | 10010 | 5001 | 2.55e+02 | 3.60e+02 | -2.49e+03 | -2.49e+03 |
| BLOCKQP4 | 10010 | 5001 | 4.09e-01 | 1.55e+00 | -2.50e+03 | -2.50e+03 |
| BLOCKQP5 | 10010 | 5001 | 1.73e+02 | 3.48e+02 | -2.49e+03 | -2.49e+03 |
| BLOWEYA | 4002 | 2002 | 1.96e+00 | 3.37e+03 | -8.01e-04 | -2.28e-02 |
| BLOWEYB | 4002 | 2002 | 5.03e-02 | 3.04e+03 | -3.74e-05 | -1.52e-02 |
| BLOWEYC | 4002 | 2002 | 7.44e-01 | F | -2.92e-03 | / |
| CLEUVEN3 | 1200 | 2973 | 6.69e+00 | 3.14e+01 | 3.76e+05 | 2.86e+05 |
| CLEUVEN4 | 1200 | 2973 | 6.59e+02 | 6.57e+01 | 5.37e+06 | 2.86e+05 |
| CLEUVEN5 | 1200 | 2973 | 6.54e+00 | 3.13e+01 | 3.76e+05 | 2.86e+05 |
| CLEUVEN6 | 1200 | 3091 | 6.05e+00 | 2.86e+01 | 2.21e+07 | 2.21e+07 |
| FERRISDC | 2200 | 210 | 2.43e+00 | 3.06e+00 | -1.02e-10 | -2.13e-04 |
| GOULDQP1 | 32 | 17 | 3.44e-03 | 7.34e-02 | -3.49e+03 | -3.49e+03 |
| HATFLDH | 4 | 7 | 1.30e-04 | 5.08e-02 | -2.45e+01 | -2.45e+01 |
| HS44 | 4 | 6 | 1.69e-04 | 3.49e-02 | -1.50e+01 | -1.30e+01 |
| HS44NEW | 4 | 6 | 1.22e-04 | 3.34e-02 | -1.50e+01 | -1.30e+01 |
| LEUVEN2 | 1530 | 2329 | 3.16e+00 | 2.05e+00 | -1.41e+07 | -1.41e+07 |
| LEUVEN3 | 1200 | 2973 | 1.16e+03 | 3.20e+02 | -1.38e+09 | -1.99e+09 |
| LEUVEN4 | 1200 | 2973 | 1.75e+01 | 6.17e+02 | -4.78e+08 | -1.83e+09 |
| LEUVEN5 | 1200 | 2973 | 1.20e+03 | 3.16e+02 | -1.38e+09 | -1.99e+09 |
| LEUVEN6 | 1200 | 3091 | 2.75e+03 | 1.21e+02 | -1.17e+09 | -1.19e+09 |
| LEUVEN7 | 360 | 946 | 8.95e-02 | 4.19e-01 | 6.95e+02 | 6.95e+02 |
| LINCONT | 1257 | 419 | PI | PI | / | / |
| MPC1 | 2550 | 3833 | 3.73e+00 | 5.74e+00 | -2.33e+07 | -2.33e+07 |
| MPC10 | 1530 | 2351 | 5.01e+00 | 7.78e-01 | -1.50e+07 | -1.50e+07 |
| MPC11 | 1530 | 2351 | 4.47e+00 | 8.80e-01 | -1.50e+07 | -1.50e+07 |

| | | | | | | |
|---|---|---|---|---|---|---|
| MPC12 | 1530 | 2351 | 4.31e+00 | 7.96e-01 | -1.50e+07 | -1.50e+07 |
| MPC13 | 1530 | 2351 | 3.53e+00 | 8.09e-01 | -1.50e+07 | -1.50e+07 |
| MPC14 | 1530 | 2351 | 3.79e+00 | 7.94e-01 | -1.50e+07 | -1.50e+07 |
| MPC15 | 1530 | 2351 | 3.62e+00 | 8.35e-01 | -1.50e+07 | -1.50e+07 |
| MPC16 | 1530 | 2351 | 3.26e+00 | 7.86e-01 | -1.50e+07 | -1.50e+07 |
| MPC2 | 1530 | 2351 | 3.99e+00 | 7.26e-01 | -1.50e+07 | -1.50e+07 |
| MPC3 | 1530 | 2351 | 3.06e+00 | 7.99e-01 | -1.50e+07 | -1.50e+07 |
| MPC4 | 1530 | 2351 | 4.72e+00 | 6.56e-01 | -1.50e+07 | -1.50e+07 |
| MPC5 | 1530 | 2351 | 4.72e+00 | 6.54e-01 | -1.50e+07 | -1.50e+07 |
| MPC6 | 1530 | 2351 | 4.45e+00 | 7.59e-01 | -1.50e+07 | -1.50e+07 |
| MPC7 | 1530 | 2351 | 4.45e+00 | 8.10e-01 | -1.50e+07 | -1.50e+07 |
| MPC8 | 1530 | 2351 | 5.40e+00 | 7.78e-01 | -1.50e+07 | -1.50e+07 |
| MPC9 | 1530 | 2351 | 5.11e+00 | 7.70e-01 | -1.50e+07 | -1.50e+07 |
| NASH | 72 | 24 | PI | PI | / | / |
| NCVXQP1 | 10000 | 5000 | F | F | / | / |
| NCVXQP2 | 10000 | 5000 | F | F | / | / |
| NCVXQP3 | 10000 | 5000 | F | F | / | / |
| NCVXQP4 | 10000 | 2500 | 1.08e+03 | 3.11e+03 | -9.38e+09 | -9.38e+09 |
| NCVXQP5 | 10000 | 2500 | 1.38e+03 | 3.49e+03 | -6.63e+09 | -6.63e+09 |
| NCVXQP6 | 10000 | 2500 | 2.20e+03 | F | -3.40e+09 | / |
| NCVXQP7 | 10000 | 7500 | F | F | / | / |
| NCVXQP8 | 10000 | 7500 | F | F | / | / |
| NCVXQP9 | 10000 | 7500 | F | F | / | / |
| PORTSNQP | 100000 | 2 | 3.66e+02 | 1.28e+00 | -1.56e+00 | -1.00e+00 |
| QPNBAND | 50000 | 25000 | 5.18e+00 | F | -2.50e+05 | / |
| QPNBLEND | 83 | 74 | 5.60e-03 | 4.20e-02 | -9.14e-03 | -9.13e-03 |
| QPNBOEI1 | 384 | 351 | 2.54e-01 | 1.36e+00 | 6.78e+06 | 6.75e+06 |
| QPNBOEI2 | 143 | 166 | 3.16e-02 | 7.76e-01 | 1.37e+06 | 1.37e+06 |
| QPNSTAIR | 467 | 356 | 7.57e-02 | 9.30e-01 | 5.15e+06 | 5.15e+06 |
| SOSQP1 | 5000 | 2501 | 4.61e-02 | 1.35e-01 | 4.24e-07 | -1.03e-10 |
| SOSQP2 | 5000 | 2501 | 1.48e-01 | 1.41e-01 | -1.25e+03 | -1.25e+03 |
| STATIC3 | 434 | 96 | DI | DI | / | / |
| STNQP1 | 8193 | 4095 | 1.66e+01 | 1.07e+03 | -3.12e+05 | -3.12e+05 |
| STNQP2 | 8193 | 4095 | 3.07e+01 | 8.94e+00 | -5.75e+05 | -5.75e+05 |

**Table 9.2:** *Runtime and final objective value comparison for QPALM and IPOPT applied to the nonconvex QPs of the Cutest test set. Failure codes: PI = primal infeasible, DI = dual infeasible and F = time limit exceeded (or numerical issues).*

| | QPALM | IPOPT |
|---|---|---|
| Runtime (sgm) | 4.0045 | 8.2645 |
| Optimal | 45 | 40 |
| Failure rate [%] | 5.6075 | 8.4112 |

**Table 9.3:** *Statistics of QPALM and IPOPT applied to the nonconvex QPs of the Cutest test set. The runtime reported is the mean over the 11 problems which converged to the same stationary point, whereas optimal denotes the number of times the solver found the stationary point with the lowest objective in problems where different stationary points were found.*

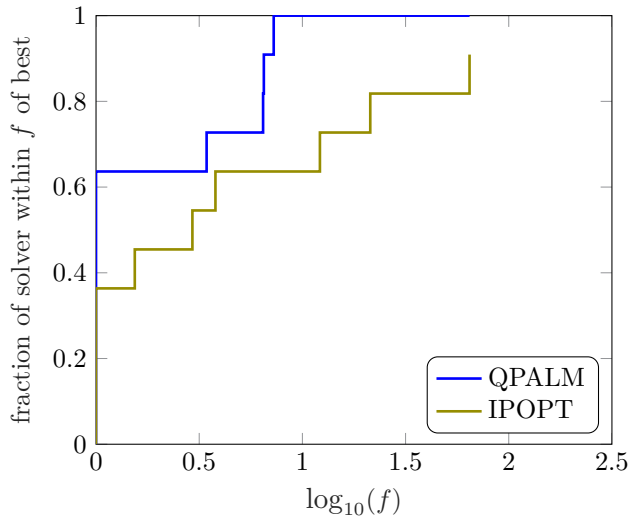



**Figure 9.5:** *Performance profile for QPALM and IPOPT on the nonconvex QPs of the Cutest test set where both converged to the same approximate stationary point.*

## 9.4 Summary

This chapter presented numerical simulations in which the performance of QPALM, implemented in open-source C code, was compared against state-of-the-art QP solvers. Our solver was shown to strike a unique balance between robustness when faced with hard problems, such as those in the Maros-Meszaros and Cutest test sets, and efficiency when faced with easy problems, such as portfolio optimization. Given a time limit of one hour, QPALM found an approximate stationary point or correctly identified infeasibility for 94.39% of the nonconvex QPs in the Cutest test set, whereas IPOPT did this only for 91.95% of them. Moreover, QPALM was able to solve all of the convex QPs in the Maros-Meszaros set, whereas Gurobi and OSQP exhibited a fail rate of 7.97% and 9.42% respectively. These results are significant in demonstrating QPALM's robustness since the Cutest and Maros-Meszaros test sets contain some very large-scale and ill conditioned QPs. Furthermore, QPALM was shown to benefit from warm starting, unlike interior-point methods.

# Conclusions and Future Research

## Conclusions

In this thesis, we considered some interesting applications of penalty and augmented Lagrangian methods to the optimal control problems arising in model predictive control. The main contribution of this work was twofold:

- A quadratic penalty method was introduced that could handle general obstacle avoidance constraints. These obstacles are described by smooth, nonlinear boundary functions. On the theoretical side, the stationarity conditions of the resulting nonlinear program proved elusive. Therefore, an equivalent problem with vertical complementarity constraints was derived, and the limit points of the proposed approach, after transformation, were shown to satisfy stationarity conditions of this latter problem. On the practical side, the method was extended by some simple heuristics, the main one involving a graph search to circumvent local minima, which arise inevitably in these kinds of problems. It was then validated in numerical simulations of autonomous navigation of two kinematic vehicle models in various obstacle scenarios. The proposed approach was shown to be more robust and runtime efficient than other state-of-the-art solvers applied to either the original problem or the problem with complementarity constraints.
- A novel quadratic programming solver, QPALM, was worked out based on the proximal augmented Lagrangian method. The novelty here lied in its extension also to nonconvex QPs, which are passed over by typical QP solvers. For convex QPs, inexact P-ALM iterations with different penalty parameters were shown, by relying on monotone operator theory, to converge to limit points satisfying the KKT conditions. For nonconvex QPs, the method was interpreted differently, relating rather to proximal point iterations on the extended cost function. The necessary modifications were shown to be minor, pertaining only to the primal penalties and the update of the regularization point, after which a linear convergence rate (of the outer iterates) to stationary points was proven. The inner minimization routine consisted of semismooth Newton steps and optimal step sizes, both of which can be implemented efficiently for QPs. QPALM was implemented in open source C code, and demonstrated in numerical simulations

to be very robust against ill conditioned problem data, as well as competitively efficient in runtime when compared against state-of-the-art QP solvers. Moreover, it is receptive to warm starting, and may therefore be a favorable choice in difficult linear MPC problems.

Both algorithms were shown to be founded on solid theoretical ground and to reach notable heights in numerical simulations. Both are also available to users for their own research projects. The penalty method was implemented independently from our work in the open-source solver optimization engine [140].[5] QPALM has been implemented in house in open-source C code.[6] These algorithms are of course not the end all be all, and multiple avenues for future research are still open. Such is the nature of most research. Much like the generation of a Koch curve, whenever one frontier is explored, multiple new segments tend to take its place.

# Avenues for Future Research

This thesis constructed two algorithms based on penalty and augmented Lagrangian methods that are useful in MPC. Although both algorithms have been extensively studied in theory and in numerical simulations, one shortcoming of the thesis might be a lack of experimental validation. While this was perhaps not necessarily as crucial a component to the development of both algorithms as the theoretical and simulation aspects, it would have nevertheless allowed also for more pragmatic considerations and lead to a more significant end result for the purposes of embedded MPC. Here, merely a side note is made that this extension to embedded applications should, at least conceptually, have been relatively easy since most of the provided code is in C. However, conclusions with regard to the applicability of the algorithms on experimental setups should only be made after corresponding results have been achieved. As they say, the proof of the pudding is in the eating. Therefore, experimental validation of the proposed algorithms, and of their extensions as discussed below, is left to the next generation of researchers. Aside from this experimental direction, several avenues regarding the development itself may be considered for further research, some of which are currently already being actualized.

## Penalty method for autonomous navigation

**Extension for additional constraints** Problem (3.1) may be extended to account for additional equality and inequality constraints. A priori, no particular challenges are foreseen for the corresponding extension of the theoretical results of §4, after appropriate modifications of the assumptions in Lemmas 4.7 and 4.8. In practice, however the approach of §3 needs to be extended, since the inner solver, PANOC,

[5] https://github.com/alphaville/optimization-engine
[6] https://github.com/Benny44/QPALM_vLADEL/

can not inherently deal with additional constraints. The authors of [140] present optimization engine to do exactly that, by relying on an augmented Lagrangian framework for general constraints and on quadratic penalties for obstacle avoidance constraints similar to ours, with PANOC as the inner solver. The theory for the combination of the quadratic penalty and an overarching augmented Lagrangian framework, however, has not yet been presented. Their solver is written in Rust. In addition, Pieter Pas, a master thesis student, is currently working in parallel on a C++ implementation of the augmented Lagrangian method with PANOC as the inner solver.

**Heuristic development** The graph-search heuristic presented in Section 3.3 is effective in the presented simulations, but still somewhat unsophisticated. For example, the current cell decomposition only takes into account the distance between the current state and the goal position, but not the size of the obstacles. In the simulations, the grid was fixed manually, since position and size of the obstacles were assumed to be known a priori. If this is not the case, it may be possible that large obstacles prevent A* from finding any feasible path on the generated grid. Adaptive grid generation strategies may therefore be considered to robustify this approach. In addition, approximate dynamic programming [16, §6] techniques may be considered to compartmentalize the space around the obstacle and avoid the curse of dimensionality while still resulting in approximate intermediate destinations.

## QPALM

**Equality constraints** In our consideration of a QP (6.1), no distinction was made for equality constraints. These were simply treated as inequality constraints with equal lower and upper bounds. However, it may be useful to consider them separately. In OSQP, for example, the penalty parameters corresponding to equality constraints are set to a higher value, reasoning that these constraints will always be active and so may be (more) enforced from the beginning [144, §5.2]. Aside from this approach, it is probably worthwhile to consider them separately, not including them in the augmented Lagrangian, but instead solving every time equality constrained subproblems of the form

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\textbf{minimize}} \quad & \hat{\varphi}_k(x) \\ \textbf{subject to} \quad & A_{\mathcal{E}}x = b_{\mathcal{E}}. \end{aligned}$$

The (semismooth) Newton direction of Section 8.1.1 is easily amenable for this equality constrainted problem, although the resulting KKT matrix is no longer quasidefinite. Therefore, modifications need to be made to the $LDL^\top$ scheme. Kemal Mahmuljin, a master thesis student, is currently working on extending QPALM for explicit equality constraints as described here.

**Exploiting OCP structure** In QPALM, sparse linear algebra factorization and update routines are used. As such, the sparsity present in structured optimization problems can typically be successfully exploited. However, in some scenarios, such as in optimal control problems, the sparsity structure is of such a type that specialized routines may be used to solve the emerging linear systems in a more efficient manner. In particular, for the systems arising in OCPs, a backward Riccati recursion technique was introduced in [141]. This method has been used in the interior-point methods HPMPC [61] and its successor HPIPM [60], as well as in the P-ALM based solver FBstab [102]. Therefore, it makes sense to incorporate it also in QPALM as an option for when the QP originated from linear MPC.

**Second-order cone constraints** As a more fundamental augmentation of the range of applicability of QPALM, also second-order cone (SOC) constraints, of the form

$$\|Ax - b\|_2 \leq c^\top x + d,$$

may be considered. With such an addition, (6.1) would of course no longer be a QP. A special case of an SOC constraint is one where $c = 0$, such that it may be reformulated as a quadratic constraint, after squaring both sides. In this case, (6.1) is then a quadratically constrained quadratic program (QCQP), a variant of QPs which is considered to be much harder than linearly constrained QPs. A possible approach to quadratic constraints in QPALM is to consider a linearization at every iteration, much like SQP methods. This methods seems appealing, at least if there are few such quadratic constraints, since it can be incorporated efficiently using the factorization update routines discussed in Section 8.2.1. After all, the old linearization may be considered as leaving the active set, while the new one as entering it. The extension of QPALM to QCQPs and possibly further to SOCPs needs to be further investigated in practice.

**Sequential quadratic programming** Briefly introduced in Section 6.1.3 and known as one of the standard techniques for nonlinear programming, sequential quadratic programming (SQP) solves, as the name implies, a sequence of quadratic programs obtained by approximating the original problem. The main component, therefore, is the underlying QP solver, and QPALM may be a prime candidate for this purpose for three reasons. First, nonconvex QPs may arise when exact second-order information, which is readily available from an AD package such as CasADi [4], is utilized. Second, the QPALM algorithm can be warm started in between SQP iterations, resulting in reduced runtimes on subsequent QPs. Third, also the eigenvalue computation routine LOBPCG of Algorithm 8.4 is amenable to warm starting, effectively removing much of the time spent on computing this eigenvalue while retaining a strict bound in every QP. Currently, a trust-region SQP solver based on QPALM is under development by David Kiessling, a fellow PhD researcher, and myself.

# Bibliography


[1] Absil, P.-A., and Tits, A. L. Newton-KKT interior-point methods for indefinite quadratic programming. *Computational Optimization and Applications 36*, 1 (2007), 5–41.

[2] Amestoy, P. R., Davis, T. A., and Duff, I. S. Algorithm 837: AMD, an approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software (TOMS) 30*, 3 (2004), 381–388.

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

[4] Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., and Diehl, M. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation* (In Press, 2018).

[5] Andreani, R., Birgin, E. G., Martínez, J. M., and Schuverdt, M. L. On augmented Lagrangian methods with general lower-level constraints. *SIAM Journal on Optimization 18*, 4 (2007), 1286–1309.

[6] Andreasson, H., Saarinen, J., Cirillo, M., Stoyanov, T., and Lilienthal, A. J. Fast, continuous state path smoothing to improve navigation accuracy. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (2015), IEEE, pp. 662–669.

[7] Anitescu, M. On solving mathematical programs with complementarity constraints as nonlinear programs. *Preprint ANL/MCS-P864-1200, Argonne National Laboratory, Argonne, IL 3* (2000).

[8] Anitescu, M. Global convergence of an elastic mode approach for a class of mathematical programs with complementarity constraints. *SIAM Journal on Optimization 16*, 1 (2005), 120–145.

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

[10] Banjac, G., Goulart, P., Stellato, B., and Boyd, S. Infeasibility detection in the alternating direction method of multipliers for convex optimization. *Journal of Optimization Theory and Applications 183*, 2 (2019), 490–519.

[11] Bauschke, H. H., and Combettes, P. L. *Convex analysis and monotone operator theory in Hilbert spaces*. CMS Books in Mathematics. Springer, 2017.

[12] Bemporad, A. A numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares. *IEEE Transactions on Automatic Control 63*, 2 (2017), 525–531.

[13] Benzi, M., Golub, G. H., and Liesen, J. Numerical solution of saddle point problems. *Acta numerica 14* (2005), 1.

[14] Bertsekas, D. P. Convexification procedures and decomposition methods for nonconvex optimization problems. *Journal of Optimization Theory and Applications 29*, 2 (Oct 1979), 169–197.

[15] Bertsekas, D. P. Constrained optimization and Lagrange multiplier methods. *Computer Science and Applied Mathematics, Boston: Academic Press, 1982* (1982).

[16] Bertsekas, D. P. Dynamic programming and optimal control 3rd edition, volume II. *Belmont, MA: Athena Scientific* (2011).

[17] Bertsekas, D. P. *Nonlinear Programming*. Athena Scientific, 2016.

[18] Birgin, E. G., and Martínez, J. M. *Practical Augmented Lagrangian Methods for Constrained Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[19] Bolte, J., Sabach, S., and Teboulle, M. Nonconvex Lagrangian-based optimization: Monitoring schemes and global convergence. *Mathematics of Operations Research 43*, 4 (2018), 1210–1232.

[20] Boyd, S., and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

[21] Boţ, R. I., and Nguyen, D.-K. The proximal alternating direction method of multipliers in the nonconvex setting: Convergence analysis and rates. *Mathematics of Operations Research 45*, 2 (2020), 682–712.

[22] Burer, S., and Vandenbussche, D. A finite branch-and-bound algorithm for nonconvex quadratic programming via semidefinite relaxations. *Mathematical Programming 113*, 2 (2008), 259–282.

[23] Burgard, W., Stachniss, C., Bennewitz, M., and Arras, K. Introduction to mobile robotics: Robot motion planning, 2011. http://ais.informatik.uni-freiburg.de/teaching/ss11/robotics/slides/18-robot-motion-planning.pdf.

[24] Byrd, R. H., Nocedal, J., and Waltz, R. A. KNITRO: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*. Springer, 2006, pp. 35–59.

[25] Candeloro, M., Lekkas, A. M., and Sørensen, A. J. A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels. *Control Engineering Practice 61* (2017), 41–54.

[26] Chen, J., and Burer, S. Globally solving nonconvex quadratic programming problems via completely positive programming. *Mathematical Programming Computation 4*, 1 (2012), 33–52.

[27] Chen, Y., Davis, T. A., Hager, W. W., and Rajamanickam, S. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS) 35*, 3 (2008), 1–14.

[28] Choset, H. M., Hutchinson, S., Lynch, K. M., Kantor, G., Burgard, W., Kavraki, L. E., Thrun, S., and Arkin, R. C. *Principles of robot motion: theory, algorithms, and implementation.* MIT press, 2005.

[29] Clarke, F. H. A new approach to Lagrange multipliers. *Mathematics of Operations Research 1*, 2 (1976), 165–174.

[30] Clarke, F. H. *Optimization and nonsmooth analysis*, vol. 5. Society for Industrial and Applied Mathematics, 1990.

[31] Combettes, P. L., and Pennanen, T. Proximal methods for cohypomonotone operators. *SIAM journal on control and optimization 43*, 2 (2004), 731–742.

[32] Combettes, P. L., and Vũ, B. C. Variable metric quasi-Fejér monotonicity. *Nonlinear Analysis, Theory, Methods and Applications 78*, 1 (2013), 17–31.

[33] Conn, A. R., Gould, N. I., and Toint, P. L. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of computation 50*, 182 (1988), 399–430.

[34] Conn, A. R., Gould, N. I., and Toint, P. L. *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*, vol. 17. Springer Science & Business Media, 2013.

[35] Coppens, P., Hermans, B., Vandersteen, J., Pipeleers, G., and Patrinos, P. A new heuristic approach for low-thrust spacecraft trajectory optimization. In *IFAC World Congress 2020* (2020). To be published.

[36] Cornuejols, G., and Tütüncü, R. *Optimization methods in finance*, vol. 5. Cambridge University Press, 2006.

[37] Cortes, C., and Vapnik, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[38] Cottle, R. W., and Dantzig, G. B. A generalization of the linear complementarity problem. *Journal of Combinatorial Theory 2* (1970), 79–90.

[39] Daniel, J. W., Gragg, W. B., Kaufman, L., and Stewart, G. W. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Mathematics of Computation 30*, 136 (1976), 772–795.

[40] Davis, T. A. Algorithm 849: A concise sparse Cholesky factorization package. *ACM Transactions on Mathematical Software (TOMS) 31*, 4 (2005), 587–591.

[41] Davis, T. A. *Direct Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, 2006.

[42] Davis, T. A., and Hager, W. W. Modifying a sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications 20*, 3 (1999), 606–627.

[43] Davis, T. A., and Hager, W. W. Multiple-rank modifications of a sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications 22*, 4 (2001), 997–1013.

[44] Davis, T. A., and Hager, W. W. Row modifications of a sparse Cholesky factorization. *SIAM Journal on Matrix Analysis and Applications 26*, 3 (2005), 621–639.

[45] Dijkstra, E. W. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271.

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

[47] Dontchev, A. L., and Rockafellar, R. T. *Implicit functions and solution mappings*, vol. 208. Springer, 2009.

[48] Douglas, J., and Rachford, H. H. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society 82*, 2 (1956), 421–439.

[49] Englert, T., Völz, A., Mesmer, F., Rhein, S., and Graichen, K. A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC). *Optimization and Engineering 20*, 3 (2019), 769–809.

[50] Evens, B., Latafat, P., Themelis, A., Suykens, J., and Patrinos, P. Neural Network Training as an Optimal Control Problem: An Augmented Lagrangian Approach. *arXiv preprint arXiv:2103.14343* (2021).

[51] Facchinei, F., and Pang, J.-S. *Finite-dimensional variational inequalities and complementarity problems*, vol. II. Springer, 2003.

[52] Farokhi, F., Shames, I., and Johansson, K. H. Distributed MPC via dual decomposition and alternative direction method of multipliers. In *Distributed model predictive control made easy*. Springer, 2014, pp. 115–131.

[53] Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., and Diehl, M. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation 6*, 4 (2014), 327–363.

[54] Flasskamp, K., Ober-Blöbaum, S., and Worthmann, K. Symmetry and motion primitives in model predictive control. *Mathematics of Control, Signals, and Systems 31*, 4 (2019), 455–485.

[55] Flegel, M. L. *Constraint qualifications and stationarity concepts for mathematical programs with equilibrium constraints*. PhD thesis, Universität Würzburg, 2005.

[56] Fletcher, R. Resolving degeneracy in quadratic programming. *Annals of Operations Research 46*, 2 (1993), 307–334.

[57] Fletcher, R., and Leyffer, S. Nonlinear programming without a penalty function. *Mathematical programming 91*, 2 (2002), 239–269.

[58] Frazzoli, E., Dahleh, M. A., and Feron, E. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE transactions on robotics 21*, 6 (2005), 1077–1091.

[59] Friedlander, M. P., and Orban, D. A primal–dual regularized interior-point method for convex quadratic programs. *Mathematical Programming Computation 4*, 1 (2012), 71–107.

[60] Frison, G., and Diehl, M. HPIPM: a high-performance quadratic programming framework for model predictive control. *arXiv preprint arXiv:2003.02547* (2020).

[61] Frison, G., Sørensen, H. H. B., Dammann, B., and Jørgensen, J. B. High-performance small-scale solvers for linear model predictive control. In *2014 European Control Conference (ECC)* (2014), IEEE, pp. 128–133.

[62] Ge, S. S., and Cui, Y. J. Dynamic motion planning for mobile robots using potential field method. *Autonomous robots 13*, 3 (2002), 207–222.

[63] Gertz, E. M., and Wright, S. J. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software (TOMS) 29*, 1 (2003), 58–81.

[64] Ghosh, S. K. *Visibility algorithms in the plane*. Cambridge university press, 2007.

[65] Gilbert, E., and Johnson, D. Distance functions and their application to robot path planning in the presence of obstacles. *IEEE Journal on Robotics and Automation 1*, 1 (1985), 21–30.

[66] Gilbert, J. C., and Joannopoulos, É. OQLA/QPALM–Convex quadratic optimization solvers using the augmented Lagrangian approach, with an appropriate behavior on infeasible or unbounded problems.

[67] Gill, P. E., Golub, G. H., Murray, W., and Saunders, M. A. Methods for modifying matrix factorizations. *Mathematics of computation 28*, 126 (1974), 505–535.

[68] Gill, P. E., and Murray, W. Numerically stable methods for quadratic programming. *Mathematical programming 14*, 1 (1978), 349–372.

[69] Gill, P. E., Murray, W., and Saunders, M. A. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM review 47*, 1 (2005), 99–131.

[70] Gill, P. E., and Wong, E. Sequential quadratic programming methods. In *Mixed integer nonlinear programming*. Springer, 2012, pp. 147–224.

[71] Goldstein, T., O'Donoghue, B., Setzer, S., and Baraniuk, R. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences 7*, 3 (2014), 1588–1623.

[72] Golub, G. H., and Van Loan, C. F. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013.

[73] Gould, N. I., Hribar, M. E., and Nocedal, J. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM Journal on Scientific Computing 23*, 4 (2001), 1376–1395.

[74] Gould, N. I., Orban, D., and Toint, P. L. GALAHAD, a library of thread-safe fortran 90 packages for large-scale nonlinear optimization. *ACM Transactions on Mathematical Software (TOMS) 29*, 4 (2003), 353–372.

[75] Gould, N. I., Orban, D., and Toint, P. L. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications 60*, 3 (2015), 545–557.

[76] Gould, N. I., and Scott, J. A note on performance profiles for benchmarking software. *ACM Transactions on Mathematical Software (TOMS) 43*, 2 (2016), 1–5.

[77] Gould, N. I., and Toint, P. L. An iterative working-set method for large-scale nonconvex quadratic programming. *Applied Numerical Mathematics 43*, 1-2 (2002), 109–128.

[78] Gray, A., Gao, Y., Lin, T., Hedrick, J. K., Tseng, H. E., and Borrelli, F. Predictive control for agile semi-autonomous ground vehicles using motion primitives. In *American Control Conference (ACC)* (2012), IEEE, pp. 4239–4244.

[79] Gurobi Optimization, L. Gurobi optimizer reference manual, 2018.

[80] Hammarling, S., and Lucas, C. Updating the QR factorization and the least squares problem.

[81] Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics 4*, 2 (1968), 100–107.

[82] Hermans, B., Patrinos, P., and Pipeleers, G. A penalty method based approach for autonomous navigation using nonlinear model predictive control. In *6th IFAC Conference on Nonlinear Model Predictive Control (NMPC)* (2018), pp. 268–274.

[83] Hermans, B., Pipeleers, G., and Patrinos, P. A penalty method for nonlinear programs with set exclusion constraints. *Automatica 127* (2021). Article. 109500.

[84] Hermans, B., Themelis, A., and Patrinos, P. QPALM: A Newton-type proximal augmented Lagrangian method for quadratic programs. In *58th IEEE Conference on Decision and Control (CDC)* (2019), pp. 4325–4330.

[85] Hermans, B., Themelis, A., and Patrinos, P. QPALM: A proximal augmented Lagrangian method for nonconvex quadratic programs. *Mathematical Programming Computation* (2021). Submitted.

[86] Hestenes, M. R. Multiplier and gradient methods. *Journal of optimization theory and applications 4*, 5 (1969), 303–320.

[87] Huber, P. J. Robust estimation of a location parameter. *The Annals of Mathematical Statistics* (1964), 73–101.

[88] Iusem, A. N., Pennanen, T., and Svaiter, B. F. Inexact variants of the proximal point algorithm without monotonicity. *SIAM Journal on Optimization 13*, 4 (2003), 1080–1097.

[89] Izmailov, A. F., Solodov, M. V., and Uskov, E. I. Global convergence of augmented Lagrangian methods applied to optimization problems with degenerate constraints, including problems with complementarity constraints. *SIAM Journal on Optimization 22*, 4 (2012), 1579–1606.

[90] Kalman, R. E. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* (1960), 35–45.

[91] Knyazev, A. V. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing 23*, 2 (2001), 517–541.

[92] Kočvara, M., and Stingl, M. PENNON: A code for convex nonlinear and semidefinite programming. *Optimization methods and software 18*, 3 (2003), 317–333.

[93] Kong, W., Melo, J. G., and Monteiro, R. D. Complexity of a quadratic penalty accelerated inexact proximal point method for solving linearly constrained nonconvex composite programs. *SIAM Journal on Optimization 29*, 4 (2019), 2566–2593.

[94] Korda, M., Henrion, D., and Jones, C. N. Convex computation of the maximum controlled invariant set for polynomial control systems. *SIAM Journal on Control and Optimization 52*, 5 (2014), 2944–2969.

[95] Latombe, J.-C. *Robot motion planning*, vol. 124. Springer Science & Business Media, 1991.

[96] LaValle, S. M. *Planning algorithms*. Cambridge university press, 2006.

[97] Lekić, A., Hermans, B., Jovičić, N., and Patrinos, P. Microsecond nonlinear model predictive control for DC-DC converters. *International Journal of Circuit Theory and Applications 48*, 3 (2020), 406–419.

[98] Leyffer, S. Complementarity constraints as nonlinear equations: Theory and numerical experience. In *Optimization with Multivalued Mappings*. Springer, 2006, pp. 169–208.

[99] Leyffer, S., López-Calva, G., and Nocedal, J. Interior methods for mathematical programs with complementarity constraints. *SIAM Journal on Optimization 17*, 1 (2006), 52–77.

[100] Li, G., and Pong, T. K. Global convergence of splitting methods for nonconvex composite optimization. *SIAM Journal on Optimization 25*, 4 (2015), 2434–2460.

[101] Li, H., and Lin, Z. Accelerated proximal gradient methods for nonconvex programming. *Advances in neural information processing systems 28* (2015), 379–387.

[102] Liao-McPherson, D., and Kolmanovsky, I. FBstab: A proximally stabilized semismooth algorithm for convex quadratic programming. *Automatica 113* (2020), 108801.

[103] Lin, Q., Ma, R., and Xu, Y. Inexact proximal-point penalty methods for non-convex optimization with non-convex constraints. *arXiv preprint arXiv:1908.11518* (2019).

[104] Liu, S., Watterson, M., Mohta, K., Sun, K., Bhattacharya, S., Taylor, C. J., and Kumar, V. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments. *IEEE Robotics and Automation Letters 2*, 3 (2017), 1688–1695.

[105] Luo, Z.-Q., and Tseng, P. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research 46*, 1 (1993), 157–178.

[106] Mangasarian, O. L., and Musicant, D. R. Robust linear and support vector regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 9 (2000), 950–955.

[107] Markowitz, H. Portfolio selection. *Journal of Finance 7* (1952), 77–91.

[108] Maros, I., and Mészáros, C. A repository of convex quadratic programming problems. *Optimization Methods and Software 11*, 1-4 (1999), 671–681.

[109] Martelli, A. On the complexity of admissible search algorithms. *Artificial Intelligence 8*, 1 (1977), 1–13.

[110] Mercy, T., Van Parys, R., and Pipeleers, G. Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology* (2017).

[111] Mészáros, C. The BPMPD interior point solver for convex quadratic problems. *Optimization Methods and Software 11*, 1-4 (1999), 431–449.

[112] Montiel, O., Orozco-Rosas, U., and Sepúlveda, R. Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. *Expert Systems with Applications 42*, 12 (2015), 5177–5191.

[113] Moré, J. J., and Sorensen, D. C. Newton's method. Tech. rep., Argonne National Lab., IL (USA), 1982.

[114] MOSEK, A. Mosek optimization toolbox for matlab. *User's Guide and Reference Manual, Version 9.2.22* (2019).

[115] Murtagh, B. A. Minos 5.5 user's guide. *Technical Report* (1998).

[116] Murtagh, B. A., and Saunders, M. A. Large-scale linearly constrained optimization. *Mathematical programming 14*, 1 (1978), 41–72.

[117] Murtagh, B. A., and Saunders, M. A. A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints. In *Algorithms for constrained minimization of smooth nonlinear functions*. Springer, 1982, pp. 84–117.

[118] Murty, K. G., and Kabadi, S. N. Some NP-complete problems in quadratic and nonlinear programming. *Mathematical programming 39*, 2 (1987), 117–129.

[119] Nesterov, Y. A method of solving a convex programming problem with convergence rate $O(\frac{1}{k^2})$. In *Sov. Math. Dokl* (1983), vol. 27, no. 2.

[120] Nesterov, Y. *Lectures on convex optimization*, vol. 137. Springer, 2018.

[121] Nocedal, J., and Wright, S. *Numerical optimization*. Springer Science & Business Media, 2006.

[122] Parikh, N., and Boyd, S. Proximal algorithms. *Foundations and Trends in Optimization 1*, 3 (2014), 127–239.

[123] Patel, R. B., and Goulart, P. J. Trajectory generation for aircraft avoidance maneuvers using online optimization. *Journal of guidance, control, and dynamics 34*, 1 (2011), 218–230.

[124] Patrinos, P., and Bemporad, A. An accelerated dual gradient-projection algorithm for embedded linear model predictive control. *IEEE Transactions on Automatic Control 59*, 1 (2014), 18–33.

[125] Patrinos, P., and Sarimveis, H. A new algorithm for solving convex parametric quadratic programs based on graphical derivatives of solution mappings. *Automatica 46*, 9 (2010), 1405 – 1418.

[126] Patrinos, P., Stella, L., and Bemporad, A. Douglas-Rachford splitting: Complexity estimates and accelerated variants. In *53rd IEEE Conference on Decision and Control* (2014), IEEE, pp. 4234–4239.

[127] Polyak, B. T. Introduction to optimization. *Inc., Publications Division, New York 1* (1987).

[128] Powell, M. J. D. A method for nonlinear constraints in minimization problems. *Optimization* (1969), 83–298.

[129] Rajamani, R. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[130] Rao, C. V., Rawlings, J. B., and Mayne, D. Q. Constrained state estimation for nonlinear discrete-time systems: Stability and moving horizon approximations. *IEEE transactions on automatic control 48*, 2 (2003), 246–258.

[131] Rawlings, J. B., Mayne, D. Q., and Diehl, M. *Model predictive control: theory, computation, and design*, vol. 2. Nob Hill Publishing Madison, WI, 2017.

[132] Rockafellar, R. T. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research 1*, 2 (1976), 97–116.

[133] Rockafellar, R. T. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization 14*, 5 (1976), 877–898.

[134] Rockafellar, R. T., and Wets, R. J.-B. *Variational analysis*, vol. 317. Springer Science & Business Media, 2011.

[135] Ruiz, D. A scaling algorithm to equilibrate both rows and columns norms in matrices. Tech. rep., Rutherford Appleton Laboratorie, 2001.

[136] Sathya, A. S., Sopasakis, P., Van Parys, R., Themelis, A., Pipeleers, G., and Patrinos, P. Embedded nonlinear model predictive control for obstacle avoidance using PANOC. In *Proceedings of the 2018 European Control Conference* (2018).

[137] Scheel, H., and Scholtes, S. Mathematical programs with complementarity constraints: Stationarity, optimality, and sensitivity. *Mathematics of Operations Research 25*, 1 (2000), 1–22.

[138] Scholtes, S., and Stöhr, M. How stringent is the linear independence assumption for mathematical programs with complementarity constraints? *Mathematics of Operations Research 26*, 4 (2001), 851–863.

[139] Sherali, H. D., and Tuncbilek, C. H. A reformulation-convexification approach for solving nonconvex quadratic programming problems. *Journal of Global Optimization 7*, 1 (1995), 1–31.

[140] Sopasakis, P., Fresk, E., and Patrinos, P. OpEn: Code generation for embedded nonconvex optimization. In *IFAC World Congress* (Berlin, Germany, 2020).

[141] Steinbach, M. C. A structured interior point SQP method for nonlinear optimal control problems. In *Computational Optimal Control*. Springer, 1994, pp. 213–222.

[142] Stella, L., Themelis, A., and Patrinos, P. Newton-type alternating minimization algorithm for convex optimization. *IEEE Transactions on Automatic Control 64*, 2 (2018), 697–711.

[143] Stella, L., Themelis, A., Sopasakis, P., and Patrinos, P. A simple and efficient algorithm for nonlinear model predictive control. In *56th IEEE Conference on Decision and Control* (2017), pp. 1939–1944.

[144] Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation* (2020), 1–36.

[145] Stentz, A. Optimal and efficient path planning for partially-known environments. In *Intelligent unmanned ground vehicles* (1997), pp. 203–220.

[146] Sun, T., Jiang, H., Cheng, L., and Zhu, W. A convergence framework for inexact nonconvex and nonsmooth algorithms and its applications to several iterations. *arXiv preprint arXiv:1709.04072* (2017).

[147] Takahashi, O., and Schilling, R. J. Motion planning in a plane using generalized Voronoi diagrams. *IEEE Transactions on robotics and automation 5*, 2 (1989), 143–150.

[148] Themelis, A., Ahookhosh, M., and Patrinos, P. On the acceleration of forward-backward splitting via an inexact Newton method. In *Splitting Algorithms, Modern Operator Theory, and Applications*, R. Luke, H. Bauschke, and R. Burachik, Eds. Springer, 2019.

[149] Themelis, A., Hermans, B., and Patrinos, P. A new envelope function for nonsmooth DC optimization. In *59th IEEE Conference on Decision and Control (CDC)* (2020), IEEE, pp. 4697–4702.

[150] Themelis, A., and Patrinos, P. Douglas–Rachford splitting and ADMM for nonconvex optimization: Tight convergence results. *SIAM Journal on Optimization 30*, 1 (2020), 149–181.

[151] Themelis, A., Stella, L., and Patrinos, P. Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone linesearch algorithms. *SIAM Journal on Optimization 28*, 3 (2018), 2274–2303.

[152] Themelis, A., Stella, L., and Patrinos, P. Douglas-Rachford splitting and ADMM for nonconvex optimization: Accelerated and Newton-type algorithms. *arXiv preprint arXiv:2005.10230* (2020).

[153] Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*, 1 (1996), 267–288.

[154] Tseng, P. Applications of a splitting algorithm to decomposition in convex programming and variational inequalities. *SIAM Journal on Control and Optimization 29*, 1 (1991), 119–138.

[155] Turri, V., Carvalho, A., Tseng, H. E., Johansson, K. H., and Borrelli, F. Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. In *16th international IEEE conference on intelligent transportation systems (ITSC 2013)* (2013), IEEE, pp. 378–383.

[156] Ullrich, G. Automated guided vehicle systems. *Information Systems Frontiers 17* (2015).

[157] Van Parys, R., and Pipeleers, G. Distributed MPC for multi-vehicle systems moving in formation. *Robotics and Autonomous Systems 97* (2017), 144–152.

[158] Vanderbei, R. J. Symmetric quasidefinite matrices. *SIAM Journal on Optimization 5*, 1 (1995), 100–113.

[159] Verschueren, R., Frison, G., Kouzoupis, D., van Duijkeren, N., Zanelli, A., Quirynen, R., and Diehl, M. Towards a modular software package for embedded optimization. *IFAC-PapersOnLine 51*, 20 (2018), 374–380.

[160] Vitus, M., Pradeep, V., Hoffmann, G., Waslander, S., and Tomlin, C. Tunnel-MILP: Path planning with sequential convex polytopes. In *AIAA guidance, navigation and control conference and exhibit* (2008), p. 7132.

[161] Wächter, A., and Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming 106*, 1 (2006), 25–57.

[162] Wang, P., and Ding, B. A synthesis approach of distributed model predictive control for homogeneous multi-agent system with collision avoidance. *International Journal of Control 87*, 1 (2014), 52–63.

[163] Watanabe, H. IC layout generation and compaction using mathematical optimization. Tech. rep., University of Rochester, Computer Science Department, 1984.

[164] Wright, S. J. *Primal-dual interior-point methods*. SIAM, 1997.

[165] Yang, K., Gan, S. K., and Sukkarieh, S. An efficient path planning and control algorithm for RUAV's in unknown and cluttered environments. In *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009* (2009), Springer, pp. 101–122.

[166] Yannakakis, M. Computing the minimum fill-in is NP-complete. *SIAM Journal on Algebraic Discrete Methods 2*, 1 (1981), 77–79.

[167] Ye, Y. On affine scaling algorithms for nonconvex quadratic programming. *Mathematical Programming 56*, 1-3 (1992), 285–300.

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF MECHANICAL ENGINEERING
MECO RESEARCH TEAM
Celestijnenlaan 300C
B-3001 Leuven
ben.hermans2@kuleuven.be
https://www.mech.kuleuven.be/en/pma/research/meco/