An Elastic Primal Active-Set Method for Structured QPs

Von der Fakultät für Mathematik und Physik
der Gottfried Wilhelm Leibniz Universität Hannover

zur Erlangung des Grades
Doktor der Naturwissenschaften
Dr. rer. nat.

genehmigte Dissertation von

M.Sc. Daniel Rose

geboren am 12.02.1986 in Gehrden

2018

Referent:	Prof. Dr. Marc C. Steinbach, Gottfried Wilhelm Leibniz Universität Hannover
Korreferent:	Prof. Dr. Christian Kirches, Technische Universität Braunschweig
Korreferent:	Prof. Dr. Thomas Wick, Gottfried Wilhelm Leibniz Universität Hannover (bis 30.09.2017: Maître de conférences, École Polytechnique Palaiseau)
Tag der Promotion:	14. Dezember 2017

To Jennifer, Theo and Otis.

I love you.

Abstract

Sequential quadratic programming (SQP) methods have proved to be very successful in solving large structured nonlinear optimization problems that may arise from several sources, e.g. nonlinear optimal control problems or branch-and-bound schemes for mixed integer nonlinear programs. In real-life applications nonlinear programs (NLP) may comprise several thousand variables and the solution by standard techniques may be impractical. Thus, the sparse structure from the NLP should be preserved in the QP subproblems and the linear systems therein. They are then exploitable by specialized solvers without affecting the basic structure of the algorithm.

This dissertation develops a generic SQP framework incorporating a primal active-set method for QP that employs an arbitrary, possibly "matrix-free" KKT solver. ℓ_1 and ℓ_2 slack relaxations of the quadratic subproblems are in use to allow efficient warm starts from infeasible starting points to avoid a phase 1. The approach involves Schur complement and projection techniques that preserve the NLP sparse structure in the KKT system.

Relevant aspects of the software design are discussed. Numerical tests including a comparison of relaxation schemes and algorithmic parametrizations for QPs from the CUTEst library document the robustness of the algorithm. The applicability of the SQP method is proved by solving highly complicated problems emerging in real-life applications, e.g. the computation of recombination parameters in the field of mathematical biology and the solution of a multistage optimization problem for dynamic processes.

Keywords: active-set methods, sequential quadratic programming, nonlinear optimization, quadratic programming, penalty formulation, slack relaxation, infeasible warm start

Zusammenfassung

Sequentielle quadratische Programmierungsverfahren (SQP) haben sich als besonders wirkungsvoll bei der Lösung von großen, strukturierten nichtlinearen Optimierungsproblemen erwiesen. Diese können sich aus mehreren Quellen ergeben, z.B. aus nichtlinearen Optimalsteuerungsproblemen oder verzweigungsbasierten Schemata für gemischt-ganzzahlige nichtlineare Programme. In realen Anwendungen können nichtlineare Programme (NLP) mehrere tausend Variablen umfassen, sodass die Lösung durch Standardtechniken nicht praktikabel sein kann. Aus diesem Grund sollte die Struktur aus dem NLP in den QP-Teilproblemen und den darin auftretenden linearen Systemen bewahrt werden. Sie können dann durch spezialisierte Löser ausgenutzt werden, ohne die Grundstruktur des Algorithmus zu beeinträchtigen.

In dieser Dissertation wird ein generisches SQP-Programmiergerüst mit einer primalen aktive Mengen Methode für quadratische Programme (QP) entwickelt, welches einen beliebigen, möglicherweise “Matrix-freien” KKT-Löser verwendet. Dabei kommen ℓ_1 and ℓ_2 Relaxationen mittels Schlupfvariablen der quadratischen Teilprobleme zum Einsatz, um effiziente Warmstarts von unzulässigen Startpunkten zu ermöglichen und eine Phase 1 zu vermeiden. Der Ansatz beinhaltet Schur-Komplement- und Projektionstechniken, welche die NLP-Struktur im KKT-System bewahren.

Des Weiteren werden die relevanten Aspekte des Softwaredesigns diskutiert und die Robustheit des Algorithmus durch numerische Tests dokumentiert, in denen Relaxationsschemata und algorithmische Parametrisierungen für QPs aus der CUTEst Bibliothek miteinander verglichen weden. Darüber hinaus wird die Anwendbarkeit des SQP-Verfahrens durch die Lösung komplexer, realer Anwendungsprobleme validiert. Dies beinhaltet die Berechnung von Rekombinationsparametern im Bereich der mathematischen Biologie sowie die Lösung eines mehrstufigen Optimierungsproblems für dynamische Prozesse.

Schlagworte: Aktive Mengen Methode, sequentielle quadratische Programmierung, nichtlineare Optimierung, quadratische Programme, Straffunktionen, Relaxierung, Warmstart

Acknowledgements

I want to thank my supervisor Marc Steinbach for giving me the opportunity to explore the interesting fields of applied mathematics and software development as well as for guiding me during my work for this thesis.

I am grateful for a lot of conductive discussions with my collegues. A special thank goes to Sebastian Probst for extensively testing the code and checking the notation with all its little details. Additionally, I want to thank my sister Sarah for proofreading the text.

Finally, I want to thank my love Jennifer. Without her support, encouragement and believe in me, this work would probably never have been completed.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Active-Set Methods for Nonlinear Programming

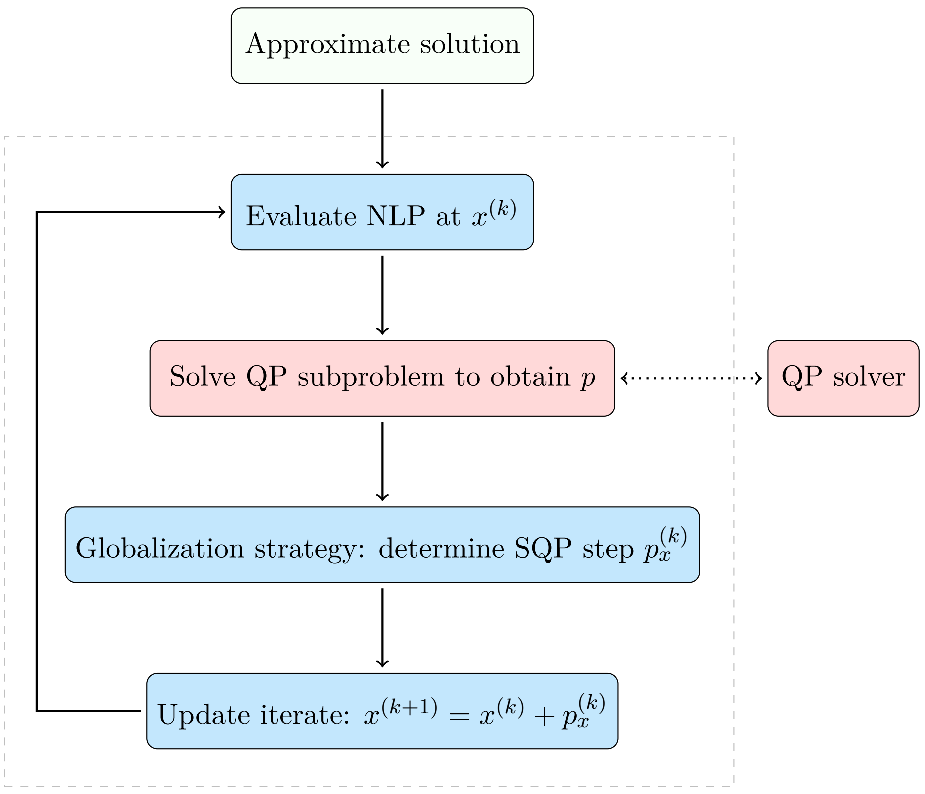
In 1963 Wilson [100] proposed the first sequential quadratic programming (SQP) method for solving constrained nonlinear optimization problems by modelling a sequence of quadratic subproblems. Ever since his basic idea never changed (see Figure 1.1) and SQP methods evolved into a powerful and effective class of optimization tools for a wide range of problems. Recent developments in methods for mixed integer nonlinear programming (MINLP) and minimization of functions subject to differential equation constraints prove that SQP methods are still a subject of active research. Especially, this is because of their capability of exploiting problem-specific structures and being "warm started" from approximate solutions [47].

Structured optimization problems arise from many sources like mechanics, control engineering or economics and finance. Using techniques of mathematical modelling for stating an objective function and involved constraints and dynamics leads to structured, possibly large-scale, nonlinear problems. Examples are optimal control problems or branch-and-bound schemes for MINLPs.

On the top level of the plain mathematical model a possibly infinite-dimensional optimization problem is stated in a certain standard form. Problem-specific data structures may be delegated from this level to the lower levels of nonlinear programs (NLP) and quadratic programs (QP).

A discretization of the dynamics in the system, e.g. by *direct Multiple Shooting* [20], leads to large structured NLPs. At this stage the sparsity of the problem data is set: Hessians and constraint matrices are obtained by the evaluation of the nonlinear model, defining the quadratic subproblems which inherit the sparsity. Now, if direct data access, like row- or column access in the matrices, is avoided by the QP solution algorithm the possibility of delegating the sparsity downwards is enabled. The result is, that the linear systems solved within the QP solution also reflect the top level structure[1] and spezialized sparse solvers can be employed.

[1]The top level states the fundamental structure of the problem by modelling the constraints and dynamics of the system. Also, the sparsity highly depends on the discretization in the NLP formulation.



Approximate solution
Evaluate NLP at $x^{(k)}$
Solve QP subproblem to obtain p
QP solver
Globalization strategy: determine SQP step $p_x^{(k)}$
Update iterate: $x^{(k+1)} = x^{(k)} + p_x^{(k)}$

Figure 1.1: The basic idea of sequential quadratic programming.

The performance and reliability of the QP solution algorithm is the linchpin in SQP methods in terms of efficiency and robustness. Warm starting SQP methods by supplying a good starting point and information about active constraints at the solution motivates the use of active-set methods for QP. Then, also the subsolver can be efficiently warm started by using the optimal active-set of the preceding subproblem. But, due to linearization errors in the formulation of the subproblems, the subalgorithm needs to be able to do so even from infeasible starting points. This is achieved by using a slack relaxation which avoids the splitting into a feasibility and an optimality phase as in standard methods.

1.2 Contributions and Organization

Structure exploitation, independence of data structures in algorithmic scopes and warm start techniques form the basis of the algorithms developed for this thesis. A carefully designed framework is developed that enables its user to easily modify and exchange certain parts, called *building blocks*, of the SQP and active-set method. This allows the user to extend the methods for solving challenging problems, easily instantiate different variants of the algorithms and make it accessible to user-defined data structures or even “matrix-free” optimization.

This dissertation is organized as follows. In Chapter 2 the required basic concepts of mathematical optimization are introduced. Based on this a SQP method for nonlinear programs implementing a filter line-search method is presented in Chapter 3. As a major task the independence between operations on the NLP and the QP level is guaranteed. This allows the implementation of a highly flexible optimization framework but also imposes higher standards on the sub-solver for QP. In particular, warm start from infeasible points is a topic which forms the focus of this thesis and is covered in Chapter 4. An elastic primal active-set method is presented that incorporates a convexification strategy for tackling nonconvex problems and relaxation schemes with structure-preserving projection techniques. The computational costs are mainly located in the solution of similar KKT systems with changing size. Thus, Chapter 5 concentrates on their efficient solution involving a generic Schur complement factorization update. Algorithmic design and the techniques of software engineering that are used to implement the optimization framework are stated in Chapter 6. Chapter 7 presents computational results for QPs from the CUTEst library to demonstrate the robustness and flexibility of the code. The applicability of the SQP method is proved by solving highly complicated problems emerging in real-life applications. This includes the computation of recombination parameters in the field of mathematical biology and the solution of a multistage optimization problem for dynamic processes. Finally, Chapter 8 concludes the thesis and gives some ideas for algorithmic extensions and applications.

Chapter 2

Basic concepts

In this chapter the basic concepts of nonlinear optimization and quadratic programming according to the topic of this thesis are presented. The chapter is organized as follows. Section 2.1 introduces the class of nonlinear optimization problems (NLP). Fundamental definitions and necessary and sufficient conditions for characterizing the solution of these problems are given. Section 2.2 deals with quadratic programs (QP) which play an important role as itself and as arising subproblems in several methods for general constrained optimization. The presented content is based on the textbooks [77, 26].

2.1 Theory of Nonlinear Optimization

In the following vectors $v \in \mathbb{R}^n$ are stated as column vectors. Vector components are symbolized by sub-indices, e.g. v_i is the i-th component of vector v. $\nabla f \in \mathbb{R}^n$ denotes the gradient of a sufficiently smooth function $f\colon \mathbb{R}^n \to \mathbb{R}$. For $c\colon \mathbb{R}^n \to \mathbb{R}^m$ the Jacobian $\nabla c \in \mathbb{R}^{m \times n}$ consist of the transposes of the component gradients $\nabla c_i \in \mathbb{R}^n$.

2.1.1 Statement of the Problem

Consider the general constrained (nonlinear) optimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) && (2.1a) \\ \text{s.t.} \quad & c_i(x) = 0, \quad i \in \mathcal{E}, && (2.1b) \\ & c_i(x) \geq 0, \quad i \in \mathcal{I} && (2.1c) \end{aligned}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ and $c_i\colon \mathbb{R}^n \to \mathbb{R}$ are sufficiently smooth. f is called the *objective function.* According to the finite and disjoint index sets $\mathcal{E}$ and $\mathcal{I}$ for *equality* and *inequality constraints* the vectors of equalities and inequalities are denoted as

$$c_{\mathcal{E}}\colon \mathbb{R}^n \to \mathbb{R}^m \quad \text{with} \quad c_{\mathcal{E}}(x) = 0 \quad \text{and} \quad c_{\mathcal{I}}\colon \mathbb{R}^n \to \mathbb{R}^k \quad \text{with} \quad c_{\mathcal{I}}(x) \geq 0, \qquad (2.2)$$

respectively. If not stated otherwise $|\mathcal{E}| = m$ and $|\mathcal{I}| = k$.

Every point x that satisfies (2.2) is called *feasible* for (2.1). The *feasible set* of a NLP consists of all these points and is stated as

$$\mathcal{F} = \{x \in \mathbb{R}^n \colon c_{\mathcal{E}}(x) = 0 \text{ and } c_{\mathcal{I}}(x) \geq 0\}. \tag{2.3}$$

Definition 1 (Active Set)**.** *The* active set $\mathcal{A}(x)$ *at any feasible point x consinsts of all constraint indices $i \in \mathcal{E}$ and the indices of the inequality constraints $i \in \mathcal{I}$ for which $c_i(x) = 0$ holds. For (2.1) it is*

$$\mathcal{A}(x) = \mathcal{E} \cup \{i \in \mathcal{I} \colon c_i(x) = 0\}. \tag{2.4}$$

An inequality constraint $c_i(x)$ for any $i \in \mathcal{I}$ is said to be *active* if $c_i(x) = 0$ and *inactive* if $c_i(x) > 0$ holds.

2.1.2 Local and Global Solutions

The goal of this section is to state proper optimality conditions for local solutions of NLPs of type (2.1). For these definitions of a constraint qualification and the lagrangian function are needed.

Definition 2 (Local Solution)**.** *A vector x^* is called a* local solution *of the problem (2.1) if $x^* \in \mathcal{F}$ and a neighborhood $\mathcal{N}$ of x^* exists such that $f(x) \geq f(x^*)$ for $x \in \mathcal{N} \cap \mathcal{F}$. A local solution is called* strict, *if $f(x) > f(x^*)$ for all $x \in \mathcal{N} \cap \mathcal{F}$ with $x \neq x^*$.*

Based on (strict) local solutions a point x^* is called an *isolated local solution* if there is a neighborhood $\mathcal{N}$ of x^* such that x^* is the only local solution in $\mathcal{N} \cap \mathcal{F}$. If both the objective function f and the feasible set are convex, (2.1) is said to be a *convex problem.*

The fastest algorithms are designed to find local minima of a NLP, but *global solutions* as the best of such points are desirable in some applications. It is usually more complicated to find global minima except for a few special cases, e.g. in convex programming in which all local solutions are global. The reader may get a proper overview on global optimization and software, e.g., in [31]. For convex programming see also [8].

Algorithms for the solution of (2.1) are usually designed subject to the definition of constraint qualifications. These conditions ensure that the linearized feasible set captures the essential features of $\mathcal{F}$ at x^*. Most often the qualification stated next is used.

Definition 3 (Linear Independence Constraint Qualification)**.** *Let $x \in \mathbb{R}^n$ be a feasible point of (2.1) and $\mathcal{A}(x)$ the associated active set. The* linear independence constraint qualification *(LICQ) holds if the set of active constraint gradients $\{\nabla c_i(x), i \in \mathcal{A}(x)\}$ is linear independent.*

Other constraint qualifications like the *Mangasarian-Fromovitz constraint qualification* (MFCQ) with associated optimality conditions are not discussed here. The interested reader is referred to Nocedal and Wright [77] and the references therein. Finally one way to characterize optimality conditions for the optimization problem under consideration is given by means of the Lagrangian function.

Definition 4 (Lagrangian Function)**.** *Let* $\lambda_{\mathcal{E}} = (\lambda_i)_{i\in\mathcal{E}} \in \mathbb{R}^m$ *and* $\lambda_{\mathcal{I}} = (\lambda_i)_{i\in\mathcal{I}} \in \mathbb{R}^k$ *be the vectors of so-called* Lagrange multipliers *(or* dual variables *or* dual multipliers*) for equality and inequality constraints. The function* $\mathcal{L}\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^k \to \mathbb{R}$ *given by*

$$\mathcal{L}(x, \lambda_{\mathcal{E}}, \lambda_{\mathcal{I}}) = f(x) - \sum_{i\in\mathcal{E}} \lambda_i c_i(x) - \sum_{i\in\mathcal{I}} \lambda_i c_i(x) \tag{2.5}$$

is then called the Lagrangian function *of (2.1).*

2.1.3 First-Order and Second-Order Optimality Conditions

The theorems presented in the following express necessary first-order and second-order optimality conditions for a solution of (2.1) by means of the Lagrangian function and Lagrange multipliers.

Theorem 1 (First-Order Necessary Conditions)**.** *Let* $x^* \in \mathbb{R}^n$ *be a local solution of (2.1) and let* f *and* c_i *for all* $i \in \mathcal{E} \cup \mathcal{I}$ *be continuously differentiable. In addition assume, that the LICQ holds at* x^**. Then there exist vectors* $\lambda^*_{\mathcal{E}} \in \mathbb{R}^m$ *and* $\lambda^*_{\mathcal{I}} \in \mathbb{R}^k$ *such that the following conditions are satisfied:*

$$\nabla f(x^*) - \sum_{i\in\mathcal{E}} \lambda^*_i \nabla c_i(x^*) - \sum_{i\in\mathcal{I}\cap\mathcal{A}(x^*)} \lambda^*_i \nabla c_i(x^*) = 0 \tag{2.6a}$$

$$c_{\mathcal{E}}(x^*) = 0, \tag{2.6b}$$

$$c_{\mathcal{I}}(x^*) \geq 0, \tag{2.6c}$$

$$\lambda^*_{\mathcal{I}} \geq 0, \tag{2.6d}$$

$$\lambda^*_i c_i(x^*) = 0, \quad i \in \mathcal{E} \cup \mathcal{I}. \tag{2.6e}$$

x^* *is then called a stationary point of problem (2.1).*

The conditions (2.6) are known as the *Karush-Kuhn-Tucker conditions*, or short *KKT conditions.* A stationary point x^* is called a *KKT point.* Condition (2.6a) forms the *dual feasibility* where equations (2.6b) and (2.6c) ensure the *primal feasibility.* Further more, (2.6d) adresses the nonnegativity of dual variables to inequality constraints. (2.6e) is the *complementarity condition* to dual multipliers and corresponding constraints.

For a given solution x^* of (2.1) and dual vectors $\lambda^*_{\mathcal{E}}$ and $\lambda^*_{\mathcal{I}}$ satisfying (2.6) one speaks of *strict complementarity*, if exactly one of λ^*_i and $c_i(x^*)$ is zero for each $i \in \mathcal{I}$, giving

$$\lambda^*_i > 0, \quad i \in \mathcal{I} \cap \mathcal{A}(x^*). \tag{2.7}$$

Even when the conditions of Theorem 1 are satisfied one can not determine from first derivative information only whether the objective function will increase or decrease along the directions w with $w^T \nabla f(x^*) = 0$. The next theorem ensures, that the curvature of the Hessian of the Lagrangian function must be nonnegative along those directions at a given KKT point $(x^*, \lambda_\mathcal{E}^*, \lambda_\mathcal{I}^*)$.

Theorem 2 (Second-Order Necessary Conditions). *Let $x^* \in \mathbb{R}^n$ be a local solution of (2.1) and suppose that the LICQ is satisfied. Further more let $\lambda_\mathcal{E}^* \in \mathbb{R}^m$ and $\lambda_\mathcal{I}^* \in \mathbb{R}^k$ be the Lagrangian multiplier for which the conditions (2.6) hold. Then*

$$w^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda_\mathcal{E}^*, \lambda_\mathcal{I}^*) w \geq 0 \tag{2.8}$$

holds for all directions that satisfy one of the following conditions:

$$\nabla c_i(x^*)^T w = 0, \quad i \in \mathcal{E}, \tag{2.9a}$$
$$\nabla c_i(x^*)^T w = 0, \quad i \in \mathcal{I} \cap \mathcal{A}(x^*) \text{ with } \lambda_i^* > 0, \tag{2.9b}$$
$$\nabla c_i(x^*)^T w \geq 0, \quad i \in \mathcal{I} \cap \mathcal{A}(x^*) \text{ with } \lambda_i^* = 0. \tag{2.9c}$$

The set of directions described by (2.9) is called the *critical cone*. If strictly satisfied condition (2.8) is sufficient for second-order optimality, even if the LICQ is neglected. This is stated in the next theorem.

Theorem 3 (Second-Order Sufficient Conditions). *Let x^* be a stationary point and $\lambda_\mathcal{E}^*, \lambda_\mathcal{I}^*$ the associated multipliers by the means of Theorem 1. Suppose that for all directions w defined by (2.9)*

$$w^T \nabla_{xx}^2 \mathcal{L}(x^*, \lambda_\mathcal{E}^*, \lambda_\mathcal{I}^*) w > 0 \tag{2.10}$$

holds, then x^* is a strict local solution of (2.1).

A detailed discussion of optimality conditions for NLP as well as proofs to the stated theorems can be found in [77].

2.1.4 Standard Solution Techniques for NLPs

Optimization algorithms for (2.1) seek for a KKT point satisfying (2.6). If no inequality constraints are incorporated, i.e. $|\mathcal{I}| = 0$, $\mathcal{I} = \emptyset$, the KKT conditions reduce to a system of nonlinear equations which can be solved using Newton's method.

Most established solution techniques for NLP include *interior-point methods* (IPM) and *sequential quadratic programming* (SQP). For IPM see the textbook of Nocedal and Wright [77]. The basic structure of the latter one involves *inner* and *outer* iterations. Associated with an approximate solution $x^{(k)}$ in the k-th outer iteration (with dual multipliers $\lambda^{(k)}$), new primal-dual estimates are found by the solution of a QP subproblem. Using a method for quadratic programming, the iterations in the QP solution constitute the inner SQP iterations.

SQP / active set algorithms tend to respond better to warm starts than IPM, especially when the active set does not change significally. But while IPMs do not warm-start as easily, they have a polynomial complexity bound (see e.g. [65]) since SQP methods can hit a combinatorial loop which has exponential complexity.

Since the focus of this thesis is on an active-set SQP framework, Section 2.2 gives a more detailed look into the topic of quadratic programming.

2.2 Quadratic Programming

Quadratic programming, the problem of optimizing a quadratic function subject to linear constraints, has been widely used since its development in the 1950s. It is a simple type of nonlinear programming that can accurately model many real world systems. Problems which are formulated this can be optimized in a straightforward way when the objective function is convex [75, 35].

The genearal *quadratic program* (QP) can be stated as

$$\min_{x\in\mathbb{R}^n} \quad q(x) = \tfrac{1}{2}x^T Hx + f^T x \tag{2.11a}$$

$$\text{s.t.} \quad a_i^T x = b_i, \quad i \in \mathcal{E}, \tag{2.11b}$$

$$a_i^T x \geq b_i, \quad i \in \mathcal{I}, \tag{2.11c}$$

where $H \in \mathbb{R}^{n\times n}$ is symmetric and f and a_i for $i \in \mathcal{E} \cup \mathcal{I}$ are vectors in $\mathbb{R}^n$. If the Hessian matrix H is positive semidefinit (2.11) is called a *convex QP* and a *strictly convex QP* if H is positive definit. Otherwise H is indefinit and the problem is called a *nonconvex QP*.

2.2.1 Applications and Methods for Quadratic Programming

Optimization problems of type (2.11) appear in a large variety of fields and applications, including

- mathematical finance, portfolio analysis [74],
- chemical engineering, optimal control [69] and model predictive control [99],
- signal and image processing

and also arise in several approaches for general nonlinear constrained optimization, e.g. in SQP methods, to solve more complex NLPs. The online collection of Gould and Toint [54] contains references to several applications of quadratic programming. In many applications an approximate solution may be known in advance, like for model predictive control.

Moreover, QPs are known to be NP-hard [95], this is why they are part of the most interesting and challenging classes of optimization problems. The computational effort for the solution of (2.11) highly depends on the objective function and the number of inequality constraints (2.11c) involed, but they can always be solved or shown to be infeasible. For a closer look, see [77, 32].

Methods for QP can be divided into interior-point and active-set methods. In both the major part of work relies on the solution of sparse systems of linear equations in the step computation. These systems correspond to the KKT conditions (2.6) and take the form

$$\begin{bmatrix} H^{(k)} & (A^{(k)})^T \\ A^{(k)} & \end{bmatrix} \begin{pmatrix} -p \\ \lambda^* \end{pmatrix} = \begin{pmatrix} g^{(k)} \\ h^{(k)} \end{pmatrix}, \tag{2.12}$$

where $H^{(k)}$ is a symmetric submatrix of the Hessian and $A^{(k)}$ is a submatrix of the constraints Jacobian. Solution techniques and update schemes for factorizations of the KKT system in a more flexible form are discussed in Chapter 5.

The computational cost in an IPM iteration is usually larger than in active-set algorithms. This is because in KKT systems all of the decision variables and inequality constraints are involved, not only those belonging to the active-set. The sparsity structure of these systems is fixed all over the solution process in primal-dual interior-point mehods, while numerical values vary. This results in the requirement of a new factorization of the KKT system in every step. In spite of that, because they generally perform few iterations, IPMs are powerful for the solution of even large–scale quadratic programs. In contrast active-set methods solve a KKT system defined by the set of active constraints in each iteration and typically perform many iterations. The sparsity structure of the linear system varies from one iteration to the next due to changes in the prediction of the active-set, but a factorization of the KKT matrix can be updated.

2.2.2 Optimality Conditions for Constrained Quadratic Problems

The Lagrangian function of problem (2.11) reads

$$\mathcal{L}(x,\lambda) = \tfrac{1}{2}x^T H x + f^T x - \sum_{i\in\mathcal{I}\cup\mathcal{E}} \lambda_i \left(a_i^T x - b_i\right). \tag{2.13}$$

Moreover, the active set $\mathcal{A}(x^*)$ as in Definition 1 can be written as

$$\mathcal{A}(x^*) = \left\{ i \in \mathcal{E}\cup\mathcal{I} \colon a_i^T x^* = b_i \right\}. \tag{2.14}$$

Using both, the first-order optimality conditions (2.6) are adjustable to the quadratic optimization problem stated above. For any solution x^* of (2.11) there exist some Lagrange multipliers $\lambda_i^*, i \in \mathcal{A}(x^*)$, such that the following conditions are satisfied.

$$Hx^* + f - \sum_{i\in\mathcal{A}(x^*)} \lambda_i^* a_i = 0 \tag{2.15a}$$

and

$$\begin{aligned} a_i^T x^* &= b_i, \quad && i \in \mathcal{A}(x^*), && \text{(2.15b)}\\ a_i^T x^* &\geq b_i, \quad && i \in \mathcal{I} \setminus \mathcal{A}(x^*), && \text{(2.15c)}\\ \lambda_{\mathcal{I}}^* &\geq 0, \quad && i \in \mathcal{I} \cap \mathcal{A}(x^*), && \text{(2.15d)}\\ \lambda_i^*(a_i^T x - b_i) &= 0, \quad && i \in \mathcal{E} \cup \mathcal{I}. && \text{(2.15e)} \end{aligned}$$

In contrast to Theorem 1 the LICQ is replaced by the qualification that all constraints are linear which is obviously satisfied in quadratic programming. Furthermore, if H is positive semidefinite, i.e. when the QP is convex, the conditions (2.15) are also sufficient for x^* and yield a unique global solution. This is stated in the following theorem.

Theorem 4. *Let x^* be a feasible point for (2.15) with suitable $\lambda_i^*, i \in \mathcal{A}(x^*)$. If H is positive semidefinite, then x^* is a global solution of (2.11).*

In addition second-order sufficient conditions for x^* hold, if H is positive definite on the kernel of the active constraints Jacobian $A = \{a_i^T\}_{i \in \mathcal{A}(x^*)}$, i.e.

$$Z^T H Z > 0 \tag{2.16}$$

for a null-space basis matrix Z of A holds. This is subsummed in the next theorem.

Theorem 5. *Assume that $A = \{a_i^T\}_{i \in \mathcal{A}(x^*)}$ has full row rank. Let Z denote a null-space basis matrix of A and let the reduced Hessian $Z^T H Z$ be positive definite, then x^* satisfying (2.15) is the strict local solution of problem (2.11).*

Complications in algorithms may accour in the nonconvex case. The problem may then have more than one strict local solution or is unbounded, cf. [77].

Chapter 3

Sequential Quadratic Programming

The evolution of *sequential quadratic programming* (SQP) methods started in June 1963, when R.B. Wilson introduced them as a generalized simplicial method in his thesis [100]. Ever since the basic idea remained unchanged: this was tackling the nonlinear optimization problem (2.1) by modelling a quadratic subproblem of type (2.11) in each iteration and defining the search direction in the nonlinear scope to be the solution of this subproblem. Until today, SQP methods have evolved into a powerful and effective class of methods for a wide range of optimization problems, cf. [47]. In general, they prove their strength when constraints inherit significant nonlinearities and the number of free variables is relatively small.

Providing the capability of beeing warm started from good approximate solutions, SQP methods are in focus of recent developments in methods for *mixed integer nonlinear programming* (MINLP) and minimization of functions subject to differential equation constraints. A detailed review of the most prominent developments as well as active research in this field of optimization is given by Gill and Wong [47].

SQP methods come along in combinations of *trust-region* and *line-search* methods and can use active-set and interior-point methods within the QP solution. Line-search SQP methods solve one QP in each iteration. Hereby trials are adjusted by reducing the step length depending on a selected globalization strategy. Trus-region frameworks set up quadratic subproblems and add a trust-region radius to it, which is adjusted depending on the acceptance or rejection of the step. For a detailed look on trus-region methods, the reader is referred to Conn et al. [17].

Focusing on active-set type methods for nonlinear programming, approaches can be categorized in two types: the EQP approach and the IQP approach. The EQP approach decouples the active set determination as a primary stage in each iteration and solves a, generally easier, equality-constrained QP to find the step. See a variant called *sequential linear-quadratic programming* (SLQP) presented in [27]. IQP approaches state a general inequality-constrained quadratic program and determine a step as well as an estimation of the optimal active-set by its solution. It is unknown at present whether the EQP or IQP approach will prove to be more effective.

A valuable property of SQP is that their fundamental framework is independent to the problem sparsity and data structure. The only requirement relies in the evaluation of the objective function and the constraints, which are naturally given, as well as the statement of local approximations of first- and second-order derivatives. Sparsity can be transmitted directly to the QP-solver and is possibly forwarded down to the KKT level therein. This motivates the development of a generic, structure delegating framework which can employ any suitable subsolver and specialized linear algebra.

The organization of this chapter is as follows. Introductory Section 3.1 gives an overwiev on relevant software for SQP in nonlinear programming. Section 3.2 describes a standard framework for nonlinear programming including the formulation of subproblems, globalization strategies, second-order corrections etc. This framework implements a filter line-search method which is motivated by the techniques presented by Fletcher and Leyffer [30]. As a major task the independence between operations done on the NLP and the QP level is guaranteed. This allows the implementation of a highly flexible and generic framework but may result in infeasible or nonconvex subproblems. Infeasibility has to be handled by the subsolver itself, nonconvexity can be tackled by the use of positiv definite quasi-Newton methods. Useful algorithmic aspects of other SQP methods like SNOPT by Gill et al. [45] are integrated into the framework developed for this thesis.

Notation: From now on the following notation is used. Vectors $v \in \mathbb{R}^n$ are denoted by small letters and vector components by sub-indices, e.g. v_i is the i-th component of vector v. Augmented vectors are stated as a sorted list of sub-vectors. For example $v = (x, y)$ stands for $v = (x^T, y^T)^T$. Iteration numbers are indexed by braced super-indices, e.g. $y^{(k)}$ denotes the vector y in iteration k.

3.1 Relevant Software for SQP

SQP methods have been implemented in many packages. They are specialized to problem sizes and the occurrence of nonlinearities in the objective and/or constraints.

Two established SQP frameworks for large-scale optimization are SNOPT [45] by Gill, Murray and Saunders and FilterSQP [28, 29] by Fletcher and Leyffer. The former code uses an augmented Lagrangian merit function reduced along search directions by a line-search approach and limited memory quasi-Newton approximations of the Hessian, cf. [43, 44]. Subproblems are solved by SQOPT [39] a reduced Hessian active-set method for convex QP. FilterSQP implements a trust-region filter SQP algorithm where QPs arising are solved by the provided bqpd-code. The code includes an automatic variable and constraint scaling as well as a feasibility restoration phase to promote convergence. Both codes are written in Fortran 77.

Small- to medium-scale problems (due to an explicit storage of the Hessian) with expensive function evaluations can be solved by KNITRO/ACTIVE [12]. The complete C/C++ code of KNITRO [13] combines interior-point and active-set strategies. All of the three packages ensure feasible subproblems and guard the constraints Jacobians against rank-deficiency.

The software package CONOPT [22] by Drud implements a generalized reduced gradient algorithm wherein a line-search SQP approach using exact second derivatives is used for the determination of better search directions, cf. [23]. The rSQP++ package [4] by Barlett and Biegler implements an Object-Oriented reduced space SQP framework written in C++ which maintains a quasi-Newton approximation of the reduced Hessian.

3.2 SQP for Nonlinear Optimization

Throughout this section a sequential quadratic programming framework for the solution of the continous nonlinear optimization problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & f(x) && \text{(3.1a)}\\ \text{s.t.} \quad & c_{\mathcal{E}}(x) = 0, && \text{(3.1b)}\\ & c_{\mathcal{R}}(x) \in [r_l, r_u], && \text{(3.1c)}\\ & x \in [b_l, b_u], && \text{(3.1d)} \end{aligned}$$

is presented. If not stated otherwise, the objective function $f\colon \mathbb{R}^n \to \mathbb{R}$, equality constraints

$$c_{\mathcal{E}}(x) = (c_1(x), \ldots, c_m(x))^T : \mathbb{R}^n \to \mathbb{R}^m \tag{3.2}$$

and range constraints

$$c_{\mathcal{R}}(x) = (c_{m+1}(x), \ldots, c_{m+k}(x))^T : \mathbb{R}^n \to \mathbb{R}^k \tag{3.3}$$

are smooth functions. In distinction to (2.1) inequality constraints are split up into lower and upper *range constraints* (3.1c) and lower and upper *variable bounds* (3.1d). The lower and upper limits are given by $r_l, r_u \in \mathbb{R}^k \cup \{\pm\infty\}$ and $b_l, b_u \in \mathbb{R}^n \cup \{\pm\infty\}$. Values $\pm\infty$ indicate absence limits. The Lagrangian function of (3.1) is denoted as

$$\begin{aligned} \mathcal{L}(x, z, v_l, v_u, u_l, u_u) = f(x) &- z^T c_{\mathcal{E}}(x)\\ &- v_l^T(c_{\mathcal{R}}(x) - r_l) - v_u^T(-c_{\mathcal{R}}(x) + r_u)\\ &- u_l^T(x - b_l) - u_u^T(-x + b_u). \end{aligned} \tag{3.4}$$

$z \in \mathbb{R}^m$ is the vector of dual multipliers of the equality constraints, $v_l, v_u \in \mathbb{R}^k$ are the vectors of dual multipliers of the lower and upper range constraints and $u_l, u_u \in \mathbb{R}^n$ are the vectors of dual multipliers of the lower and upper variable bounds. For better reading, $\lambda = (z, v_l, v_u, u_l, u_u) \in \mathbb{R}^{n_d}$, $n_d = m + 2k + 2n$ denotes the vector of dual multipliers.

SQP methods use the Lagrangian formalism aiming directly for a first-order critical point satisfying the KKT conditions. The conditions (2.6) applied to (3.1) consist of the dual feasibility

$$\nabla_x \mathcal{L}(x,\lambda): \quad \nabla f(x) - \nabla c_{\mathcal{E}}(x)^T z - \nabla c_{\mathcal{R}}(x)^T (v_l - v_u) - (u_l - u_u) = 0 \tag{3.5a}$$

and primal feasibility

$$\begin{aligned}
\nabla_z \mathcal{L}(x,\lambda): &\quad c_{\mathcal{E}}(x) = 0, && (3.5b)\\
\nabla_{v_l} \mathcal{L}(x,\lambda): &\quad c_{\mathcal{R}}(x) \geq r_l, && (3.5c)\\
\nabla_{v_u} \mathcal{L}(x,\lambda): &\quad -c_{\mathcal{R}}(x) \geq -r_u, && (3.5d)\\
\nabla_{u_l} \mathcal{L}(x,\lambda): &\quad x \geq b_l, && (3.5e)\\
\nabla_{u_u} \mathcal{L}(x,\lambda): &\quad -x \geq -b_u && (3.5f)
\end{aligned}$$

as well as the nonnegativity and complementarity conditions (for all $i \in \mathcal{E} \cup \mathcal{R} \cup \mathcal{B}$)

$$\begin{aligned}
v_l, -v_u, u_l, -u_u &\geq 0, && (3.5g)\\
\lambda_i c_i(x) &= 0. && (3.5h)
\end{aligned}$$

In (3.5a) $\nabla c_{\mathcal{E}}(x) \in \mathbb{R}^{m \times n}$ and $\nabla c_{\mathcal{R}}(x) \in \mathbb{R}^{k \times n}$ denote the constraint Jacobians of $c_{\mathcal{E}}(x)$ and $c_{\mathcal{R}}(x)$, with

$$\nabla c_{\mathcal{E}}(x) = \begin{bmatrix} \frac{\partial c_1(x)}{\partial x_1} & \cdots & \frac{\partial c_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial c_m(x)}{\partial x_1} & \cdots & \frac{\partial c_m(x)}{\partial x_n} \end{bmatrix}, \quad \nabla c_{\mathcal{R}}(x) = \begin{bmatrix} \frac{\partial c_{m+1}(x)}{\partial x_1} & \cdots & \frac{\partial c_{m+1}(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial c_{m+k}(x)}{\partial x_1} & \cdots & \frac{\partial c_{m+k}(x)}{\partial x_n} \end{bmatrix}. \tag{3.6}$$

3.2.1 Determination of the Search Direction

The basic structure of SQP methods involves *outer* and *inner* iterations, associated with the current solution estimate $(x^{(k)}, \lambda^{(k)})$, the k-th outer iterate, for (3.1). As already mentioned aboved, new estimates are found by the minimization of a quadratic model of the objective function subject to a linearization of the constraints[1] about $x^{(k)}$; defining the inner iterates.

At first, consider the simpliest derivation in constrained nonlinear optimization, i.e. when only equality constraints are involved ($\mathcal{R} = \mathcal{B} = \emptyset$). The NLP to solve then reads

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{s.t.} \quad c_{\mathcal{E}}(x) = 0 \tag{3.7}$$

[1]Constraints are replaced by a linear first order Taylor series approximation, the objective by a second order Taylor series approximation augmented by second order information from the constraints.

and a local SQP method obtains search directions $p_x^{(k)}$ as the solution of

$$\min_{p\in\mathbb{R}^n} \quad q^{(k)}(p) = \tfrac{1}{2}p^T H^{(k)} p + (g^{(k)})^T p \tag{3.8a}$$

$$\text{s.t.} \quad A_{\mathcal{E}}^{(k)} p + c_{\mathcal{E}}^{(k)} = 0, \tag{3.8b}$$

where $g^{(k)} = \nabla_x f(x^{(k)})$, $A_{\mathcal{E}}^{(k)} = \nabla c_{\mathcal{E}}(x^{(k)})$, $c_{\mathcal{E}}^{(k)} = c_{\mathcal{E}}(x^{(k)})$ and $H^{(k)}$ is (an approximation of) the Hessian of the Lagrangian, i.e. $H^{(k)} \approx \nabla^2_{xx}\mathcal{L}(x^{(k)}, \lambda^{(k)})$.

Another access to SQP is to explain it by the application of Newton's method to the KKT optimality conditions for problem (3.7). A step $(x^{(k+1)}, \lambda^{(k+1)}) = (x^{(k)}, \lambda^{(k)}) + (p_x^{(k)}, p_\lambda^{(k)})$ is then obtained by solving the Newton-KKT system

$$\begin{bmatrix} H^{(k)} & (A_{\mathcal{E}}^{(k)})^T \\ A_{\mathcal{E}}^{(k)} & \end{bmatrix} \begin{pmatrix} p_x^{(k)} \\ -p_\lambda^{(k)} \end{pmatrix} = \begin{pmatrix} -g^{(k)} + (A_{\mathcal{E}}^{(k)})^T \lambda^{(k)} \\ -c_{\mathcal{E}}^{(k)} \end{pmatrix} \tag{3.9}$$

supposing that $(x^{(k)}, \lambda^{(k)})$ is an estimate of the KKT point (x^*, λ^*). The conformity becomes clear, by the view on the equivalence of the solution of (3.9) and a stationary point of (3.8) with associated dual multipliers $\lambda_{\text{QP}}^{(k)} = \lambda^{(k)} + p_\lambda^{(k)}$. For a more detailed look see Nocedal and Wright [77] or Conn et al. [17].

Recalling the optimality conditions for quadratic programming (Section 2.2.2) applied to (3.8) or looking at the Newton iteration above, the iteration procedure is well defined, when the KKT matrix is nonsingular. That is if at $(x, \lambda) = (x^{(k)}, \lambda^{(k)})$ for $\nabla c(x) = \nabla c_{\mathcal{E}}(x)$ the LICQ is satisfied and the second-order sufficient condition, see Theorem 3, holds close to (x^*, λ^*). This leads to common regularity assumptions in SQP, which are subsummed in the following.

Assumption 1. *(A1) The constraints Jacobian $A^{(k)} = \nabla c(x)$ has full row rank;*

(A2) The Hessian of the Lagrangian $H^{(k)} = \nabla^2_{xx}\mathcal{L}(x, \lambda)$ is positive definit on the tangent space of $c(x)$, that is, $d^T \nabla^2_{xx}\mathcal{L}(x, \lambda) d > 0$ for all $d \neq 0$ satisfying $\nabla c(x) d = 0$.

At this point, a local SQP method can be stated (see Algorithm 1). Given user-specified primal and dual tolerances $\epsilon_{\text{pri}} > 0$, $\epsilon_{\text{dual}} > 0$ it generates a sequence $\{x^{(k)}\}$ until a local minimizer (or at least stationary point) of (3.7) is found, i.e. if it (nearly) satisfies the KKT conditions (3.5a) and (3.5b) for some $\lambda^{(k)} \in \mathbb{R}^m$. This goal is monitored by the overall NLP termination criterion

$$\|c_{\mathcal{E}}(x^{(k)})\| < \epsilon_{\text{pri}} \quad \text{and} \quad \|\nabla_x \mathcal{L}(x^{(k)}, \lambda^{(k)})\| < \epsilon_{\text{dual}}. \tag{3.10}$$

The framework presented so far can be extended easily to the general case of problem (3.1) by linearizing both the equality and inequality constraints (i.e. $A_{\mathcal{R}}^{(k)} = \nabla c_{\mathcal{R}}(x^{(k)})$ and $c_{\mathcal{R}}^{(k)} = c_{\mathcal{R}}(x^{(k)})$). In IQP approches, this leads to quadratic subproblems of the following

Algorithm 1: Local SQP Algorithm for (3.7)

```
Input: User provided initial pair $(x^{(0)}, \lambda^{(0)})$.
1 Set $k = 0$.
2 while the NLP termination criterion (3.12) is not satisfied do
3     Evaluate $H^{(k)}$, $g^{(k)}$ and $A_{\mathcal{E}}^{(k)}$, $c_{\mathcal{E}}^{(k)}$.
4     Solve (3.8) to obtain $p_x^{(k)}$ and $\lambda_{\text{QP}}^{(k)}$.
5     Update iterates by $x^{(k+1)} \leftarrow x^{(k)} + p_x^{(k)}$ and $\lambda^{(k+1)} \leftarrow \lambda_{\text{QP}}^{(k)}$.
6     Set $k \leftarrow k + 1$.
```

type:

$$\min_{p \in \mathbb{R}^n} \quad q^{(k)}(p) = \tfrac{1}{2} p^T H^{(k)} p + (g^{(k)})^T p \tag{3.11a}$$

$$\text{s.t.} \quad A_{\mathcal{E}}^{(k)} p + c_{\mathcal{E}}^{(k)} = 0, \tag{3.11b}$$

$$A_{\mathcal{R}}^{(k)} p + c_{\mathcal{R}}^{(k)} \in [r_l, r_u], \tag{3.11c}$$

$$p + x^{(k)} \in [b_l, b_u]. \tag{3.11d}$$

In those the termination criterion is extended to the KKT conditions (3.5) for some $\lambda^{(k)} \in \mathbb{R}^{n_d}$. The nonnegativity of dual multipliers and the complementarity condition need not to be monitored by the SQP framework itself - they are provided by the QP subsolver by returning $p_x^{(k)} = 0$ and suitable $\lambda^{(k)} = \lambda_{\text{QP}}^{(k)}$, whenever $x^{(k)}$ is a stationary point for (3.1). The general NLP termination criterion is given by

$$\max\left(\|\xi^{(k)}\|, \|\rho_l^{(k)}\|, \|\rho_u^{(k)}\|, \|\beta_l^{(k)}\|, \|\beta_u^{(k)}\|,\right) < \epsilon_{\text{pri}} \quad \text{and} \quad \|\nabla_x \mathcal{L}(x^{(k)}, \lambda^{(k)})\| < \epsilon_{\text{dual}} \tag{3.12}$$

where $\xi^{(k)} = c_{\mathcal{E}}(x^{(k)})$ and (using $[y]^+ = \max(y, 0)$)

$$\rho_l^{(k)} = [c_{\mathcal{R}}(x^{(k)}) - r_l]^+, \rho_u^{(k)} = [-c_{\mathcal{R}}(x^{(k)}) + r_u]^+, \beta_l^{(k)} = [x^{(k)} - b_l]^+, \beta_u^{(k)} = [-x^{(k)} + b_u]^+.$$

It is easy to see, that if the iterate $x^{(k)}$ is primal feasible and the constraints (3.11b) and (3.11c) are linear a feasible starting point for (3.11) is given by $p = 0$. Near the solution, when steps get small enough, $\mathcal{A}(x^{(k)}) \approx \mathcal{A}(x^{(k-1)})$ holds. This motivates the use of active-set QP-solvers which can be warm started by reusing the optimal active-set of the previous iteration. The other way round difficulties may arise if constraints are nonlinear, the guess of the active-set contains inactive constraints or the quadratic models may not be convex, i.e. $H^{(k)} \not\succeq 0$.

3.2.2 Globalization with a Filter Line-Search Algorithm

In order to ensure global convergence from remote starting points to stationary points, the optimization progress needs to be monitored and steps might be truncated to enforce the globalization strategy. Once a search direction $(p_x^{(k)}, p_\lambda^{(k)})$ is determined by the solution of (3.11), a step length α has to be chosen to compute the next iterate given by

$$x^{(k+1)} = x^{(k)} + \alpha_k p_x^{(k)} \quad \text{and} \quad \lambda^{(k+1)} = \lambda^{(k)} + \alpha_k p_\lambda^{(k)}. \tag{3.13}$$

There are three broad classes of strategies for testing the acceptability of trial steps (cf. [68]): *augmented Lagrangian methods*, *penalty* and *merrit-function methods* and *filter methods*.

The first class successively minimizes a shifted augmented Lagrangian

$$\bar{\mathcal{L}}(x, \lambda; \rho) = f(x) - \lambda^T p_k(x) + \tfrac{\rho}{2} \|p_k(x)\|_2^2 \tag{3.14}$$

with higher-order nonlinear terms $p_k(x)$ at $x^{(k)}$, subject to a linearization of the constraints:

$$\min_x \bar{\mathcal{L}}(x, \lambda; \rho) \quad \text{s.t.} \quad c(x^{(k)}) + \nabla c(x^{(k)})^T (x - x^{(k)}). \tag{3.15}$$

Penalty and merrit-function techniques combine the aim of minimizing the objective function and reaching feasibility in a *merrit function.* A standard merrit function for problem (3.7) is the ℓ_1 penalty function

$$\phi_1(x) = f(x) + \rho \|c_{\mathcal{E}}(x)\|_1 \tag{3.16}$$

for a *penalty parameter* $\rho > 0$.

Filter Methods

Filter methods are first proposed by Fletcher and Leyffer [30]. The idea of filter methods is to treat the goals of minimizing the objective function and constraint violation separately. Therefore they keep record of the objective function values $f^{(l)} = f(x^{(l)})$ and constraint violations $\theta^{(l)} = \theta(x^{(l)})$ with

$$\theta(x^{(l)}) = \max\left(\|\xi^{(l)}\|, \|\rho_l^{(l)}\|, \|\rho_u^{(l)}\|, \|\beta_l^{(l)}\|, \|\beta_u^{(l)}\|, \right) \tag{3.17}$$

for some previous iterates $x^{(l)}$ with $(\theta^{(l)}, f^{(l)}) \in \mathcal{F}$. This allows to accept step lengths that make progress only in the objective function or the reduction of infeasibility instead of requiring progress in a combination of both. As an outcome, filter methods can often take larger steps and tend to be more robust.

The definitions and concepts in the upcoming paragraphs are based on the textbooks of Nocedal and Wright [77] and Biegler [5]. The presented approach follows the guidelines of Wächter and Biegler [97, 102]. See also Chin [14, 15] and Chin and Fletcher [16] for a similar approach.

Definition 5 (Domination). *One says that a pair* $(\theta^{(i)}, f^{(i)})$ *is dominated by another pair* $(\theta^{(j)}, f^{(j)})$ *if* $f^{(j)} \leq f^{(i)}$ *and* $h^{(j)} \leq \theta^{(i)}$ *holds.*

Definition 6 (Filter). *A filter* $\mathcal{F}$ *is a list of pairs* $(\theta^{(i)}, f^{(i)})$ *in which no pair dominates any other pair.*

The basic idea of a backtracking filter line-search is the generation of a sequence of primal step lengths

$$\alpha_{k,l+1} = \kappa_b \alpha_{k,l}, \quad \kappa_b \in (0,1), \quad l = 0,1,\ldots, \tag{3.18}$$

which are used to compute a sequence of trial points

$$x^{(+)} = x^{(k)} + \alpha_{k,l} p_x^{(k)}, \quad l = 0,1,\ldots, \tag{3.19}$$

until a trial point is found such that

$$(\theta^{(+)}, f^{(+)}) = (f(x^{(+)}), \theta(x^{(+)})) \tag{3.20}$$

is *acceptable* for the filter. The idea of *acceptability* is formulated in the following definition.

Definition 7 (Acceptability). *A pair* $(\theta^{(+)}, f^{(+)})$ *is said to be acceptable for inclusion in the filter if it is not dominated by any pair in the filter, i.e. if*

$$f^{(+)} \leq f^{(j)} \quad \textit{or} \quad \theta^{(+)} \leq \theta^{(j)} \tag{3.21}$$

holds for all $(\theta^{(j)}, f^{(j)}) \in \mathcal{F}$.

This acceptance criterion can be strengthened to improve the practical performance of the algorithm by replacing (3.21) by

$$f^{(+)} \leq f^{(j)} - \kappa_{m_1}\theta^{(j)} \quad \text{or} \quad \theta^{(+)} \leq (1-\kappa_{m_2})\theta^{(j)} \tag{3.22}$$

for given filter margins $\kappa_{m_1}, \kappa_{m_2} \in (0,1)$ that ensure, that iterates cannot accumulate at infeasible limit points. A typical filter in the (θ, f)-plane is shown in Figure 3.1. The straight lines correspond to the region (filled area) that is dominated by the filter, the dotted line coresponds to the envelope defined by the filter margins $\kappa_{m_1}, \kappa_{m_2}$.

Sufficient Reduction Condition

To ensure convergence to a local minimum, filter methods ensure that the trial point provides sufficient reduction with respect to the current iterate provided by

$$f^{(+)} \leq f^{(k)} - \kappa_{m_1}\theta^{(k)} \quad \text{or} \quad \theta^{(+)} \leq (1-\kappa_{m_2})\theta^{(k)}. \tag{3.23}$$

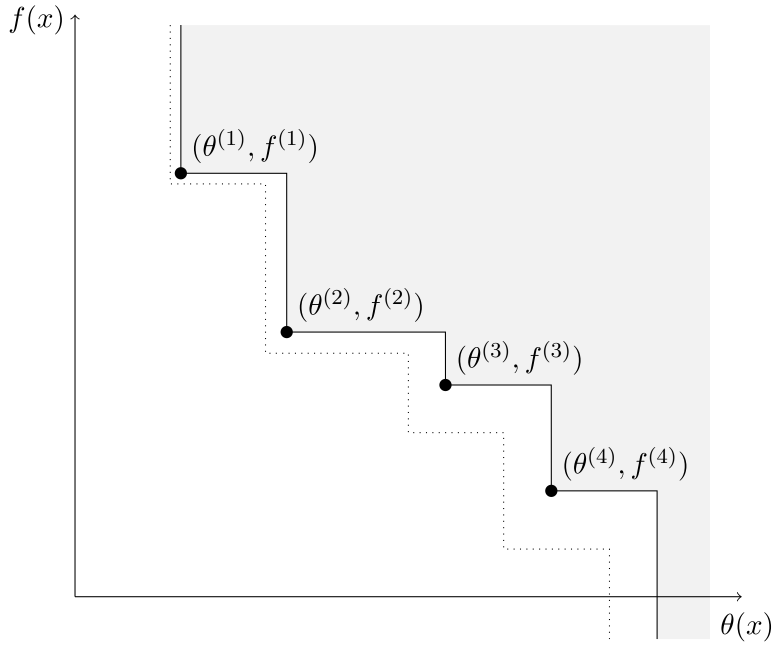

$f(x)$
$(\theta^{(1)}, f^{(1)})$
$(\theta^{(2)}, f^{(2)})$
$(\theta^{(3)}, f^{(3)})$
$(\theta^{(4)}, f^{(4)})$
$\theta(x)$

Figure 3.1: Example of a filter with five entries including $(-\infty, \theta_{\max})$.

These criteria provide sufficient decrease in the infeasibility but an additional condition indicating sufficient decrease in the objective function, in particular for near feasible trial points, i.e. $\theta^{(k)} \leq \theta_{\min}$, is needed. The above criterion is replaced by the *Armijo condition* for unconstrained optimization

$$f^{(+)} \leq f^{(k)} + \kappa_a \alpha_{k,l} (g^{(k)})^T p_x^{(k)}, \quad \kappa_a \in (0, \tfrac{1}{2}), \tag{3.24}$$

whenever the so called *switching condition*

$$(g^{(k)})^T p_x^{(k)} < 0 \quad \text{and} \quad \alpha_{k,l} \left[-(g^{(k)})^T p_x^{(k)} \right]^{\kappa_{s1}} > \kappa_\delta \left[\theta^{(k)} \right]^{\kappa_{s2}} \tag{3.25}$$

holds for fixed constants $\kappa_\delta > 0, \kappa_{s_2} > 1, \kappa_{s_1} > 2\kappa_{s_2}$. If the switching condition holds the step is called an *f-type step*. Otherwise, when (3.25) fails, an accepted step is said to be a *θ-type step*, mainly reducing the constraint violation.

Maximum Constraints Violation

In most cases, one wants to ensure that a trial point with a constraint violation larger than a given threshold $\theta_{\max}$ is never accepted by the filter. This can be achieved by initializing the filter as

$$\mathcal{F}_0 = \{(\theta_{\max}, -\infty)\} \tag{3.26}$$

Mostly, $\theta_{\max}$ is computed in dependence of the constraint violation at the starting point, e.g.

$$\theta_{\max} = \kappa_{i_1} \max(1, \kappa_{i_2}\theta^{(0)}), \quad \kappa_{i_1}, \kappa_{i_2} > 0. \tag{3.27}$$

Augmentation of the Filter

The basic motivation of filter techniques is to be less conservative than penalty methods. To achieve this goal, the filter is not augmented by $(\theta^{(k)}, f^{(k)})$ in every iteration. It is updated only, when the switching condition (3.25) or the Armijo condition (3.24) does not hold and the constraint violation at the trial point is larger than a given threshold $\theta_{\min} > 0$, defined by

$$\theta_{\min} = \kappa_{i_3}\theta(x^{(0)}), \quad \kappa_{i_3} > 0. \tag{3.28}$$

Second-Order Correction

The filter technique presented so far may suffer from the *Maratos effect*, see [77]. To overcome this problem, and to improve the overall robustness of the algorithm, *second-order correction* (SOC) is employed if a desired step is rejected (or truncated) by the filter. The method developed for this thesis follows the analysis of Wächter and Biegler [102], where an SOC step is computed if the first trial point is not acceptable to the filter.

There is a wide range of options for the computation of a second-order correction leading to slightly different SOC problems to solve, cf. [102]. One efficient approach in (active-set) SQP frameworks is to determine a composite step $\tilde{p}_x^{(k)} = p_x^{(k)} + p_{x,\text{soc}}^{(k)}$ (and associated dual variables $\lambda_{\text{soc}}^{(k)}$) obtained by the solution of

$$\min_{p \in \mathbb{R}^n} \quad q_{\text{soc}}^{(k)}(p) = \tfrac{1}{2} p^T H_{\text{soc}}^{(k)} p + (g_{\text{soc}}^{(k)})^T p \tag{3.29a}$$

$$\text{s.t.} \quad A_{\mathcal{E},\text{soc}}^{(k)} p + c_{\mathcal{E},\text{soc}}^{(k)} = 0, \tag{3.29b}$$

$$A_{\mathcal{R},\text{soc}}^{(k)} p + c_{\mathcal{R},\text{soc}}^{(k)} \in [r_l, r_u], \tag{3.29c}$$

$$p + x^{(k)} \in [b_l, b_u], \tag{3.29d}$$

where $H_{\text{soc}}^{(k)} = H^{(k)}, A_{\mathcal{E},\text{soc}}^{(k)} = A_{\mathcal{E}}^{(k)}, A_{\mathcal{R},\text{soc}}^{(k)} = A_{\mathcal{R}}^{(k)}, g_{\text{soc}}^{(k)} = g^{(k)}$ as in subproblem (3.11) and

$$c_{\mathcal{E},\text{soc}}^{(k)} = c_{\mathcal{E}}^{(k)} + c_{\mathcal{E}}(x^{(k)} + p_x^{(k)}), \quad c_{\mathcal{R},\text{soc}}^{(k)} = c_{\mathcal{R}}^{(k)} + c_{\mathcal{R}}(x^{(k)} + p_x^{(k)}).$$

This approach is inexpensive to compute and capable of reusing existing factorizations from the minimization of $q^{(k)}(p)$ in the solution of (3.29).

Another correction corresponds to the step determined in the next iteration, supposing that $x^{(k)} + p_x^{(k)}$ has been accepted. It is similar to the so called *nonmonotone (watchdog) strategies*, seeking for improved feasibility and optimality under the assumption that the reasons for the rejection by the globalization strategy will be temporary and subsequent steps will more than compensate for it, see [77]. The step correction $p_{x,\text{soc}}^{(k)}$ is obtained by the solution of (3.29) using the quantities

$$\begin{aligned} H_{\text{soc}}^{(k)} &\approx \nabla_{xx}^2 \mathcal{L}(x^{(k)} + p_x^{(k)}, \lambda_{\text{QP}}^{(k)}) & g_{\text{soc}}^{(k)} &= \nabla_x \mathcal{L}(x^{(k)} + p_x^{(k)}, \lambda_{\text{QP}}^{(k)}), \\ A_{\mathcal{E},\text{soc}}^{(k)} &= \nabla_x c^{\mathcal{E}}(x^{(k)} + p_x^{(k)}), & c_{\mathcal{E},\text{soc}}^{(k)} &= c^{\mathcal{E}}(x^{(k)} + p_x^{(k)}), \\ A_{\mathcal{R},\text{soc}}^{(k)} &= \nabla_x c^{\mathcal{R}}(x^{(k)} + p_x^{(k)}), & c_{\mathcal{R},\text{soc}}^{(k)} &= c^{\mathcal{R}}(x^{(k)} + p_x^{(k)}). \end{aligned}$$

and replace (3.29d) by $p + x^{(k)} + p_x^{(k)} \in [b_l, b_u]$. This alternative is more expansive to compute but may lead to better results on some instances with significant nonlinearities in the constraints.

The corrected trial and Lagrangian multipliers for the next iterate are then defined by

$$x^{(+)} = x^{(k)} + p_x^{(k)} + p_{x,\text{soc}}^{(k)} \quad \text{and} \quad \lambda^{(+)} = \lambda_{\text{soc}}^{(k)}. \tag{3.30}$$

However, if the corrected trial fails to be accepted by the filter, the step size is reduced via a backtracking strategy and the SOC step is rejected. Then, the backtracking procedure is repeated until $x^{(k)} + \alpha_{k,l} p_x^{(k)}$ satisfies the filter conditions.

For a detailed discussion of second-order correction the interested reader is referred to Conn et.al. [17], Nocedal and Wright [77] or Fletcher [25].

Algortihmic Details

Following the analysis of Wächter and Biegler [102] trial points with $\theta^{(+)} > \theta_{\min}$ are included into the filter only for θ-type iterations. Finally, it is not guaranteed that a step size $\alpha_{k,l}$ larger than a given threshold $\alpha_{\min}$ can be found, that is accepted by the filter. Using linear models of the involved functions, this situation may be indicated if $\alpha_{k,l}$ becomes smaller than

$$\alpha_{k,\min} = \kappa_\alpha \cdot \begin{cases} \min\left\{ \kappa_{m_2}, \frac{\kappa_{m_1}\theta^{(k)}}{-(g^{(k)})^T p_x^{(k)}}, \frac{\delta[\theta^{(k)}]^{\kappa_{s2}}}{[-(g^{(k)})^T p_x^{(k)}]^{\kappa_{s1}}}, \right\} & , \text{ if } (g^{(k)})^T p_x^{(k)} < 0 \\ \kappa_{m_2} & , \text{ otherwise,} \end{cases} \tag{3.31}$$

with a constant safty guard $\kappa_\alpha \in (0, 1]$. If this occurs the algorithm exits and the main SQP framework needs to restore feasibility, i.e. find a new iterate $x^{(k+1)}$ that is accaptable in the sense of (3.22).

For given algorithmic constants Algorithm 2 states the complete filter line-search method.

3.2.3 Feasibility Restoration Phase

In general it can neighter be guaranteed that (3.11) is sufficiently consistent and has a feasible solution nor that a step size $\alpha_{k,l} > \alpha_{k,\min}$ exists that is acceptable for the filter and

Algorithm 2: Filter Line-Search Algorithm

```
Input: Primal iterate $x^{(k)}$ and desired search direction $p_x^{(k)}$, vector of algorithmic
       constants $\kappa_{\text{ls}}$.
1  Initialize backtracking line-search counter $l = 0$, set $\alpha_{k,l} = 1$.
2  if $\alpha_{k,l} < \alpha_{k,\min}$ (see 3.31) then
3      Go to feasibility restoration phase (see Section 3.2.3).
4  Compute trial $x^{(+)} = x^{(k)} + \alpha_{k,l} p_x^{(k)}$.
5  if $x^{(+)}$ is acceptable to $\mathcal{F}$ (cf. (3.22)) then
6      if (3.25) holds then
           // $f$-type step.
7          if (3.24) holds then
8              Accept $x^{(+)}$, return step length $\alpha_{k,l}$ and search direction $p_x^{(k)}$.
9          else if $l = 0$ then
10             Go to SOC step computation, line 20.
11     else
           // $\theta$-type step.
12         if (3.23) holds then
13             Accept $x^{(+)}$, if $\theta^{(+)} > \theta_{\min}$, add $(\theta^{(k)}, f^{(k)})$ to $\mathcal{F}$ and remove all entries
               that are dominated by $(\theta^{(k)}, f^{(k)})$.
14             return step length $\alpha_{k,l}$ and search direction $p_x^{(k)}$.
15         else if $l = 0$ then
16             Go to SOC step computation, line 20.
17 else if $l = 0$ then
18     Go to SOC step computation, line 20.
   // Backtracking procedure
19 Set $\alpha_{k,l+1} = \kappa_b \alpha_{k,l}$, $l \leftarrow l + 1$ and go to line 2.
   // Second order correction
20 Determine a second-order correction $p_{\text{soc}}^{(k)}$ as the solution of (3.29). If the SOC
   subproblem is infeasible, go to line 29.
21 Compute trial $x^{(+)} = x^{(k)} + p_x^{(k)} + p_{\text{soc}}^{(k)}$.
22 if $x^{(+)}$ is acceptable to $\mathcal{F}$ (cf. (3.22)) then
23     if (3.25) holds for $\alpha_{k,0} = 1$, $d_k$ and $\theta^{(k)}$ then
           // $f$-type step.
24         if (3.24) holds with $\alpha_{k,0} = 1$ then
25             Accept $x^{(+)}$, return step length $\alpha_{k,l} = 1$ and search direction $p_x^{(k)} + p_{\text{soc}}^{(k)}$.
26     else if (3.23) holds then
           // $\theta$-type step.
27         Accept $x^{(+)}$, if $\theta^{(+)} > \theta_{\min}$, add $(\theta^{(k)}, f^{(k)})$ to $\mathcal{F}$ and remove all entries that
           are dominated by $(\theta^{(k)}, f^{(k)})$.
28         return step length $\alpha_{k,l}$ and corrected search direction $p_x^{(k)} + p_{\text{soc}}^{(k)}$.
29 Discard $p_{\text{soc}}^{(k)}$, set $\alpha_{k,l+1} = \kappa_b \alpha_{k,l}$, $l \leftarrow l + 1$ and go to line 2.
```

provides sufficient reduction in one of the filter's objectives. In these situations the SQP algorithm needs to switch to a *feasibility restoration phase*, whose purpose is to find a new iterate $x^{(k+1)}$ that is accaptable to the filter and satisfies (3.23) merely by the reduction of the constraint violation. The reduction of θ could be achieved by any iterative algorithm, for example by ignoring the objective function. Even different methods could be used at different stages of the optimization procedure, see [97].

It is reasonable to avoid that the solution of the feasibility restoration phase is too far away from the current iterate $x^{(k)}$ at which the procedure was invoked. To achieve this goal, one could, for example, compute the shortest vector satisfying the linearized constraints in (3.11). This can be achieved by solving the relaxed quadratic program

$$\min_{(p,s,t)\in\mathbb{R}^N} \quad \tfrac{1}{2}p^T p + \frac{\rho}{2}\|s\|_2^2 + \frac{\rho}{2}\|t\|_2^2 \tag{3.32a}$$

$$\text{s.t.} \quad A_{\mathcal{E}}^{(k)}p + s + c_{\mathcal{E}}^{(k)} = 0, \tag{3.32b}$$

$$A_{\mathcal{R}}^{(k)}p + t + c_{\mathcal{R}}^{(k)} \in [r_l, r_u], \tag{3.32c}$$

$$p + x^{(k)} \in [b_l, b_u] \tag{3.32d}$$

including slack variables $s \in \mathbb{R}^m, t \in \mathbb{R}^k$ absorbing the infeasibility and a fixed penalty parameter $\rho > 0$. It is obvious, that (3.32) is allways feasible and a solution point where s, t vanish is feasible to (3.1) w.r.t. the current linearization of the constraints.

Since feasible iterates are never included into the filter, the algorithm for the feasibility restoration phase should usually be able to find a new accaptable iterate unless it converges to a stationary point of θ - this would identify the problem to be (at least locally) infeasible. If it terminates successfully the filter is augmented by $(\theta^{(k)}, f^{(k)})$ to avoid cycling back to the problematic point $x^{(k)}$. More details on procedures for feasibility restoration in filter line-search methods can be found in [97, 30]. Algorithm 3 subsummes the procedure of the feasibility restoration.

Algorithm 3: Feasibility Restoration Phase

```
Input : Filter $\mathcal{F}$ and problematic pair $(\theta^{(k)}, f^{(k)})$.
1 Solve restoration problem (3.32) to obtain a (near) feasible point $x_{\text{frp}}^{(+)}$ (and
  multipliers $\lambda_{\text{frp}}^{(+)}$) that is acceptable to $\mathcal{F}$, see Section 3.2.3.
2 Augment the Filter $\mathcal{F}$ by $(\theta^{(k)}, f^{(k)})$.
3 return (near) feasible point $x_{\text{frp}}^{(+)}$ and multipliers $\lambda_{\text{frp}}^{(+)}$.
```

3.2.4 Algorithmic Details

The performance of SQP methods highly depends on the quality of the starting point and chosen algorithmic constants κ. These aspects are discussed in the following paragraphs.

Startingpoint Strategy

Three different situations can be distinguished depending on whether the user does not provide an initial estimate, provides an initial estimation for the primal variables $\bar{x}^{(0)}$ or provides primal and dual estimates $(\bar{x}^{(0)}, \bar{\lambda}^{(0)})$. If the user does not support any starting point $\bar{x}^{(0)} = 0$ is set.

Mainly, the aspects for the determination of the quality of the supplied starting point are its primal and dual feasibility. Obviously, it is helpful to start the algorithm with an almost feasible point which should also be not too far away from dual feasibility.

Algorithm 4 states the default initialization strategy of the implementation developed for this thesis. First the primal variables are shifted to ensure bound feasibility. Afterwards, if not given by the user, at least the dual multipliers $z^{(0)}$ corresponding to equality constraints are approximated by the solution of the linear least-squares problem

$$z^{(0)} = \arg\min_{z} \frac{1}{2} \left\| (\nabla c_{\mathcal{E}}(x^{(0)}))^T z - \nabla f(x^{(0)}) \right\|_2^2 . \tag{3.33}$$

Algorithm 4: Initialization Strategy

```
Input: Primal initial estimate $\bar{x}^{(0)}$ and optional multipliers $\bar{\lambda}^{(0)}$.
1  if $\bar{x}^{(0)}$ is not supplied then
2      Set $\bar{x}^{(0)} = 0$.
   // Feasibility to bound constraints
3  for $i = 1, \dots, n$ do
4      if $\bar{x}_i^{(0)} \in [b_{l,i}, b_{u,i}]$ then
5          Set $x_i^{(0)} = \bar{x}_i^{(0)}$.
6      else
7          Set $x_i^{(0)} = b_{l,i}$ if $\bar{x}_i^{(0)} < b_{l,i}$ and $\bar{x}_i^{(0)} = b_{u,i}$ otherwise.
   // Initialization of dual multipliers
8  if $\bar{\lambda}^{(0)}$ is supplied then
9      Set $\lambda^{(0)} = \bar{\lambda}^{(0)}$; initialize $\mathcal{W}_0$ in active-set QP solver.
10 else
11     Set $v_l, v_u, u_l, u_u = 1$.
12     Evaluate $\nabla f(x^{(0)})$, $A_{\mathcal{E}}^{(0)}$, $c_{\mathcal{E}}(x^{(0)})$.
13     Solve the linear least-squares problem (3.33) to initialize $z^{(0)}$.
14 return initial pair $(x^{(0)}, \lambda^{(0)})$.
```

Algorithmic Constants

Inspired by the interior-point filter line-search code Ipopt by Waechter and Biegler [98] and the generic iterior-point framework Clean::IPM by Martin Schmidt [82] Table 3.1

explanation	symbol	range	default
Small step margin	κ_s	≥ 0	10^{-8}
Filter margin f	κ_{m_1}	$(0,1)$	10^{-5}
Filter margin θ	κ_{m_2}	$(0,1)$	10^{-5}
Filter initialization factor	κ_{i_1}	> 0	10^4
Filter initialization factor	κ_{i_2}	> 0	10^1
Filter initialization factor	κ_{i_3}	> 0	10^{-4}
Switching condition factor	κ_{s_1}	$> \kappa_{s_2}$	2.3
Switching condition factor	κ_{s_2}	> 1	1.1
Switching condition factor	κ_δ	> 0	1
Safty guard for $\alpha_{\min}$	κ_α	$(0,1]$	1
Backtracking factor	κ_b	$(0,1)$	0.5

Table 3.1: Algorithmic constants used in the proposed Filter line-search method.

summarizes the ranges of inherited algorithmic constants and the chosen defaults for the implementation for this thesis.

3.2.5 Quasi-Newton Methods

Many methods for (un-)constrained optimization use an approximation of the Hessian of the Lagrangian when exact second-order derivatives are either unavailable or too expensive to compute. The computational study of Gill et al. [46] additionally shows, that in some situations, so called *quasi-Newton methods* are more efficient than competing methods based on using the exact Hessian of the Lagrangian.

Quasi-Newton methods refer to iterative approaches which only include first-order derivative information. The curvature of the objective function is estimated along the direction from $x^{(k)}$ to $x^{(k+1)}$ and is kept up to date in each iteration. Given an initial point $x^{(0)}$ and an initial approximation $\mathfrak{B}^{(0)}$, the subsequent approximations are obtained by using, for example, the symmetric *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) formula

$$\mathfrak{B}^{(k+1)} = \mathfrak{B}^{(k)} - \frac{\mathfrak{B}^{(k)}\mathfrak{s}^{(k)}(\mathfrak{s}^{(k)})^T\mathfrak{B}^{(k)}}{(\mathfrak{s}^{(k)})^T\mathfrak{B}^{(k)}\mathfrak{s}^{(k)}} + \frac{\mathfrak{g}^{(k)}(\mathfrak{g}^{(k)})^T}{(\mathfrak{s}^{(k)})^T\mathfrak{g}^{(k)}} \tag{3.34}$$

where

$$\mathfrak{s}^{(k)} = x^{(k+1)} - x^{(k)} \quad \text{and} \quad \mathfrak{g}^{(k)} = \nabla_x\mathcal{L}(x^{(k+1)}, \lambda^{(k+1)}) - \nabla_x\mathcal{L}(x^{(k)}, \lambda^{(k+1)}).$$

Formula (3.34) is a rank-two update that - under certain conditions - guarantees the sequence $\mathfrak{B}^{(k)}$ to be positive definite. If $\nabla^2_{xx}\mathcal{L} > 0$ in the region where the optimization takes place, BFGS quasi-Newton approximations $\mathfrak{B}^{(k)}$ tend to converge robustly and rapidly, but otherwise the BFGS approach may be problematic. Depending on the objective's curvature updates should then be skipped if the *curvature condition*

$$(\mathfrak{s}^{(k)})^T \mathfrak{g}^{(k)} \geq \theta (\mathfrak{s}^{(k)})^T \mathfrak{B}^{(k)} \mathfrak{s}^{(k)}, \qquad \theta > 0 \tag{3.35}$$

does not hold. In this context Powell [78] proposed a damping strategy for the constrained case (see also [77]) by replacing the definition of $\mathfrak{g}^{(k)}$ by

$$\mathfrak{r}^{(k)} = \theta_k \mathfrak{g}^{(k)} + (1 - \theta_k) \mathfrak{B}^{(k)} \mathfrak{s}^{(k)} \tag{3.36}$$

where the scalar value θ_k is defined as

$$\theta_k = \begin{cases} 1 & \text{, if (3.35) holds for } \theta = 0.2, \\ \frac{0.8 (\mathfrak{s}^{(k)})^T \mathfrak{B}^{(k)} \mathfrak{s}^{(k)}}{(\mathfrak{s}^{(k)})^T \mathfrak{B}^{(k)} \mathfrak{s}^{(k)} - (\mathfrak{s}^{(k)})^T \mathfrak{r}^{(k)}} & \text{, if (3.35) does not hold for } \theta = 0.2 \end{cases} \tag{3.37}$$

and $\mathfrak{B}^{(k)}$ is updated as follows:

$$\mathfrak{B}^{(k+1)} = \mathfrak{B}^{(k)} + \frac{\mathfrak{B}^{(k)} \mathfrak{s}^{(k)} (\mathfrak{s}^{(k)})^T \mathfrak{B}^{(k)}}{(\mathfrak{s}^{(k)})^T \mathfrak{B}^{(k)} \mathfrak{s}^{(k)}} + \frac{\mathfrak{r}^{(k)} (\mathfrak{r}^{(k)})^T}{(\mathfrak{s}^{(k)})^T \mathfrak{r}^{(k)}} \tag{3.38}$$

If one is using quasi-Newton approximations in SQP, the desire to obtain a descent direction by solving (3.11) motivates to use rank-two update formula like BFGS to obtain convex subproblems. Damped BFGS updating often works well in combination with line-search methods but it fails to address the underlying problem if the Hessian of the Lagrangian is not positive definite. In the nonconvex case one could use different approaches like the *symmetric rank-one* (SR1) method which will not guarantee the approximations to be positive definite and include appropriate safeguards in the globalization strategy.

Updating procedures like BFGS or SR1 result in dense matrices and therefore undermine sparsity in general. By this reason such methods are only useful when a more detailed sparsity with dense matrix blocks is used. Otherwise it may be profitable to use inexact update strategies.

An overview of quasi-Newton approximation for (un-)constrained optimization including convergence analysis can be found in [77] or in the often cited literature [19, 10].

3.2.6 The Complete Filter Line-Search SQP Framework

At this point, the complete filter line-search SQP framework can be stated (see Algorithm 5).

3.2.7 Convergence Analysis

The described SQP framework combines several aspects of a filter line-search method stated by Wächter and Biegler in [97, 102, 96]. The implementation follows the approaches

Algorithm 5: Filter Line-Search SQP Algorithm

```
Input: User provided starting point $\bar{x}^{(0)}$ and optional multiplier estimates $\bar{\lambda}^{(0)}$ for
       problem (3.1), vector of algorithmic constants $\kappa$ (see Table 3.1).
1  Set outer iteration counter $k = 0$.
2  Call Algorithm 4 with $\bar{x}^{(0)}$ (and $\bar{\lambda}^{(0)}$, if available) to obtain initial pair $(x^{(0)}, \lambda^{(0)})$.
3  Evaluate problem data $H^{(0)}$, $A_{\mathcal{E}}^{(0)}$, $A_{\mathcal{R}}^{(0)}$ and $g^{(0)}$, $c_{\mathcal{E}}(x^{(0)})$, $c_{\mathcal{R}}(x^{(0)})$ at $(x^{(0)}, \lambda^{(0)})$.
4  while the NLP termination criterion (3.12) does not hold do
       // Step determination
5      Solve (3.11) to obtain $p_x^{(k)}$ and $\lambda_{\text{QP}}^{(k)}$;
6      if (3.11) is insufficient consistent then
7          Go to feasibility restoration phase, i.e. call Algorithm 3 to obtain
           $(x_{\text{frp}}^{(+)}, \lambda_{\text{frp}}^{(+)})$ and set $x^{(k+1)} \leftarrow x_{\text{frp}}^{(+)}$, $\lambda^{(k+1)} \leftarrow \lambda_{\text{frp}}^{(+)}$.
8      else if $\|p_x^{(k)}\| < \kappa_s$ then
9          Accept small step; Update iterates by $x^{(k+1)} \leftarrow x^{(k)} + p_x^{(k)}$, $\lambda^{(k+1)} \leftarrow \lambda_{\text{QP}}^{(k)}$.
10     else
           // Determine stepsize by filter line-learch
11         Call Algorithm 2 with primal iterate $x^{(k)}$, search direction $p_x^{(k)}$ and maximal
           step size $\alpha_{\max} = 1$ to obtain $\alpha_k$;
12         if Algorithm 2 succeeds then
13             Compute new primal and dual iterates by $x^{(k+1)} \leftarrow x^{(k)} + \alpha_k p_x^{(k)}$ and
               $\lambda^{(k+1)} \leftarrow (1 - \alpha_k)\lambda^{(k)} + \alpha_k \lambda_{\text{QP}}^{(k)}$.
14         else
15             Go to feasibility restoration phase, i.e. call Algorithm 3 to obtain
               $(x_{\text{frp}}^{(+)}, \lambda_{\text{frp}}^{(+)})$ and set $x^{(k+1)} \leftarrow x_{\text{frp}}^{(+)}$, $\lambda^{(k+1)} \leftarrow \lambda_{\text{frp}}^{(+)}$.
       // Update iteration data
16     Compute/Update $H^{(k+1)}$, $A_{\mathcal{E}}^{(k+1)}$, $A_{\mathcal{R}}^{(k+1)}$.
17     Compute $g^{(k+1)}$, $c_{\mathcal{E}}(x^{(k+1)})$, $c_{\mathcal{R}}(x^{(k+1)})$.
18     Evaluate the KKT error at $(x^{(k+1)}, \lambda^{(k+1)})$ (see (3.12)).
19     Increase iteration counter $k \leftarrow k + 1$.
20 return optimal solution $(x^{(k)}, \lambda^{(k)})$.
```

stated therin and combines the techniques which ensure local and global convergence. For this reason, only the main assumptions and results are recapitulated.

The following notation is used: $\mathfrak{R}$ denotes the set of iteration indices k in which the feasibility restoration phase is invoked. Moreover, the set $\mathfrak{R}_{\text{inc}} \subseteq \mathfrak{R}$ denotes the set of iteration indices in which (3.11) is not (sufficiently) consistent.

Assumption 2 (Global convergence)**.** *Let $\{x^{(k)}\}$ be the sequence of iterates generated by Algorithm 5. Moreover, assume that the feasibility restoration phase allways terminates successfully and Algorithm 5 does not stop with a KKT point in line 4*

(G1) There exists an open set $\mathcal{O} \subset \mathbb{R}^n$ with $[x^{(k)}, x^{(k)} + p_x^{(k)}] \subset \mathcal{O}$ for all iterations $k \notin \mathfrak{R}_{inc}$ and $f, c_{\mathcal{E}}, c_{\mathcal{R}}$ are differentiable and bounded on $\mathcal{O}$ and their first derivatives are bounded and Lipschitz-continous over $\mathcal{O}$.

(G2) The Hessians of the Lagrangian $H^{(k)}$ or, if an approximation is used, its approximations of problem (3.1) are uniformly bounded for all $k \notin \mathfrak{R}_{inc}$.

(G3) There exists a constant $\theta_{inc} > 0$ such that Algorithm 5 does not enter the feasibility restoration phase if $\theta^{(k)} < \theta_{inc}$.

(G4) There exist a constants $C_p, C_\lambda, C_q > 0$, so that for all iterations $k \notin \mathfrak{R}_{inc}$

$$\|p_x^{(k)}\| \leq C_p, \quad \|\lambda_{QP}^{(k)}\| \leq C_\lambda, \quad \|q_x^{(k)}\| \leq C_q \theta^{(k)}$$

holds, where $q_x^{(k)}$ denotes the shortest vector satisfying the constraints in (3.11). Therefore the step is decomposed as

$$p_x^{(k)} = q_x^{(k)} + d_x^{(k)}. \tag{3.39}$$

(G5) There exists a constant $C_H > 0$, so that $(p_x^{(k)})^T H^{(k)} p_x^{(k)} > C_H (p_x^{(k)})^T p_x^{(k)}$ holds for all iterations $k \notin \mathfrak{R}_{inc}$.

Combining assumption (G3) and (G4) means, that the QP (3.11) is sufficiently consistent near to feasible points. Moreover, (G5) ensures descent in the objective function at sufficiently feasible points. Together these conditions coincide with (A1) and (A2).

In [97], the following global convergence theorems are proved using the criticality measure $\chi(x^{(k)}) = \|d_x^{(k)}\|_2$ for iterations $k \notin \mathfrak{R}_{\text{inc}}$.

Theorem 6 (Feasibility)**.** *Suppose Assumption 2 holds. Then $\lim_{k\to\infty} \theta^{(k)} = 0$.*

Theorem 7 (Optimality)**.** *Suppose Assumption 2 holds. Then all limit points are feasible, and if $\{x^{(k)}\}$ is bounded, than there exists a limit point x^* of $\{x^{(k)}\}$ which is a first-order optimal point for the NLP (3.1), i.e.*

$$\lim_{k\to\infty} \theta^{(k)} = 0 \quad \textit{and} \quad \liminf_{k\to\infty} \chi(x^{(k)}) = 0. \tag{3.40}$$

The usage of second-order correction preserves Algorithm 5 from suffering from the Maratos effect. Under stronger assumptions superlinear convergence in the neighborhood of a local solution x^* of the NLP could be achieved. The following assumption and theorem are taken from [102].

Assumption 3 (Local convergence)**.** *Let $\{x^{(k)}\}$ be the sequence of iterates generated by Algorithm 5 that converges to a local solution x^* of the NLP (3.1), and the following assumptions hold.*

(L1) The functions f, $c_{\mathcal{E}}$, $c_{\mathcal{R}}$ are twice continously differentiable in a neighborhood of x^*.

(L2) x^* satisfies the KKT conditions (3.5) for some $\lambda^* \in \mathbb{R}^{n_d}$. The active constraints Jacobian $A^{(k)}_{\mathcal{A}(x^*)}$ has full row rank (see (A1)) and the Hessian of the Lagrangian $\nabla^2_{xx}\mathcal{L}(x^*, \lambda^*)$ is positive definite on the null-space of $A^{(k)}_{\mathcal{A}(x^*)}$ (see (A2)).

(L3) The matrices $H^{(k)}$ in QP (3.11) are bounded and uniformly positive definite on the null-space of the active constraints Jacobian $A^{(k)}_{\mathcal{A}(x^*)}$.

(L4) The matrices $H^{(k)}_{soc}$ in QP (3.29) are bounded and uniformly positive definite on the null-space of the active constraints Jacobian $A^{(k)}_{\mathcal{A}(x^*),soc}$, and

$$g^{(k)}_{soc} = o(\|p^{(k)}_x\|), \quad A^{(k)}_{\mathcal{E}} - A^{(k)}_{\mathcal{E},soc} = O(\|p^{(k)}_x\|), \quad A^{(k)}_{\mathcal{R}} - A^{(k)}_{\mathcal{R},soc} = O(\|p^{(k)}_x\|).$$

(L5) The (approximations of the) Hessian of the Lagrangian $H^{(k)}$ in QP (3.11) satisfy

$$(\nabla^2_{xx}\mathcal{L}(x^{(k)}, \lambda^*) - H^{(k)})p^{(k)}_x = o(\|p^{(k)}_x\|). \tag{3.41}$$

(L6) Assumption (G3) holds.

Theorem 8 (Local convergence)**.** *Suppose Assumption 3 holds. Then for k sufficiently large, full steps of the form $x^{(k)} + p^{(k)}_x$ or $x^{(k)} + p^{(k)}_x + p^{(k)}_{x,soc}$ are taken, and $x^{(k)}$ converges to x^* superlinearly.*

Chapter 4

Active-Set Methods for Quadratic Programming

Active-set methods (ASM) for QP have been widely used since the 1970s and are based on extending the simplex algorithm for linear programming. They find their strenght in the solution of small- and medium-sized problems and more recently as subsolvers in frameworks for nonlinear optimization. Because they are able to capitalize on a good estimate of the solution ASM are also effective when a series of related QPs must be solved. The solution of one problem may then be used to warm start the next. This feature makes this type of algorithms particulary strong in SQP methods. For instance, see Gill and Wong [47] for a recent survey.

There are three variants of active-set algorithms to be distinguished: *primal*, *dual* and *primal-dual*. Primal methods require a feasible starting point and maintain feasibility to equality and inequality constraints and drive dual infeasibility to zero. Vice-versa, dual approaches maintain feasibility of the dual inequalities requiering a dual feasible starting point, while moving to satisfy the primal inequalities. Finally, primal-dual methods combine the former ones: a QP is solved as a coupled pair of primal and dual quadratic programs. The interested reader is referred to Wong [101] and Boland [6] for primal and dual methods. A comparison of the mentioned approaches is, for example, given by Forgsen et al. [33]. For active-set methods for more general (nonconvex) QPs, see [50, 55, 53] and the references therein.

Existing implementations of active-set methods for quadratic programming include QPOPT [38], SQOPT [39] and QPBLUR [71]. QPBLUR implements an active-set convex QP solver based on regularized KKT systems. QPOPT and SQOPT use a two-phase active-set method employing a reduced Hessian strategy. The so called phase 1 generates a feasible starting point for the optimality phase.

Motivated by SQP methods and specialized sparse solvers available for the KKT systems arising in quadratic programming, the main goal of this thesis is to find an elastic active-set QP solver for structured QP. Ensuring genericity, the algorithm employs any custom KKT solver in a slack relaxation of the quadratic program to avoid a phase 1. This involves partial projection techniques that preserve the superordinate problem's sparse structure in the KKT system.

The outline of this chapter is as follows. Section 4.1 describes a formal (primal) active-set method for solving convex quadratic programs of the form (2.11). Step and step length computation, changes in the workingset, etc. are discussed. In Section 4.2 an elastic primal active-set method as described above is devoloped. Two different relaxation schemes with its structural and algorithmic details are presented.

Notation: As already used for the donation of vector components, row vectors a_i^T for $i \in \mathcal{M}$ of a matrix $A_{\mathcal{M}}$ are denoted by small letters with sub-indices, e.g. $a_i^T \in \mathbb{R}^n$ is a row of $A_{\mathcal{M}} = \{a_i\}_{i \in \mathcal{M}} \in \mathbb{R}^{|\mathcal{M}| \times n}$. e is the vector as follows in appropriate dimension: $e = (1, \ldots, 1)^T$.

4.1 Active-Set Methods for Convex QP

Active-set methods for convex quadratic programming are iterative algorithms. Starting at a feasible initial estimate a sequence of approximate solutions of (2.11) is generated and a prediction of the optimal set of active constraints is maintained and updated. This procedure is done with respect to the so called *workingset*.

Definition 8 (Workingset)**.** *The workingset $\mathcal{W}_k$ to the k-th iterate $x^{(k)}$ consists of all constraint indices from $\mathcal{E}$ and a selection of indices i from $\mathcal{I}$ whose constraints are active at $x^{(k)}$ and the gradients a_i of the constraints are linearly independent.*

Since the complete set of active constraints at a solution x^* of (2.11) determined by an active-set method may inherit linear dependent gradients, the workingset is not necessarily equivalent to the optimal active set. It holds

$$\mathcal{W}_k \subseteq \mathcal{A}(x^{(k)}). \tag{4.1}$$

4.1.1 Step Computation

Given an iterate $x^{(k)}$ primal active-set methods for convex QP determine a step $p^{(k)}$ by solving an equality-constrained quadratic subproblem. All constraints corresponding to the k-th workingset are regarded as equalities, the remaining constraints are temporary neglected. By the definition of

$$p = x - x^{(k)}, \quad g_k = Hx^{(k)} + f, \quad \rho_k = \tfrac{1}{2}(x^{(k)})^T H x^{(k)} + f^T x^{(k)} \tag{4.2}$$

the subproblem based on (2.11) can be written as

$$\min_{p \in \mathbb{R}^n} \quad q_k(p) = \tfrac{1}{2} p^T H p + (g^{(k)})^T p + \rho_k \tag{4.3a}$$

$$\text{s.t.} \quad a_i^T p = 0, \quad i \in \mathcal{W}_k. \tag{4.3b}$$

By the first-order necessary conditions (2.15) applied to (4.3) the existence of an appropriate vector of Lagrange multipliers λ^* to a solution p^* is guaranteed. According to Theorem 5, a KKT point can be computed by the solution of the linear system

$$\begin{bmatrix} H & \left(A^{(k)}_{\mathcal{W}_k}\right)^T \\ A^{(k)}_{\mathcal{W}_k} & 0 \end{bmatrix} \begin{pmatrix} -p \\ \lambda^* \end{pmatrix} = \begin{pmatrix} g^{(k)} \\ 0 \end{pmatrix}. \tag{4.4}$$

If $p^{(k)} = p$ denotes the solution of (4.3) (or equivalent of the system (4.4)) one finds, that for all $i \in \mathcal{W}_k$ and any step length $\alpha_k \geq 0$

$$a_i^T(x^{(k)} + \alpha_k p^{(k)}) = a_i^T x^{(k)} = b_i \tag{4.5}$$

holds. Supposing that $p^{(k)}$ is nonzero, α_k is chosen to be the largest value in the range $[0, 1]$. If $x^{(k)} + p^{(k)}$ is feasible w.r.t. $\mathcal{W}_k$, a full step is applied (i.e. $\alpha_k = 1$) and $x^{(k+1)} = x^{(k)} + p^{(k)}$. Otherwise it is

$$x^{(k+1)} = x^{(k)} + \alpha_k p^{(k)}. \tag{4.6}$$

4.1.2 Determination of the Step Length

Once a search direction $p^{(k)}$ has been determined α_k is chosen in the sense of maximal decrease of the objective function (2.11a) subject to retaining feasibility. There are three cases to be observed:

1. For $i \in \mathcal{W}_k$ every choice of α_k yields a next feasible iterate.
2. For $i \notin \mathcal{W}_k$ with $a_i^T p^{(k)} \geq 0$ constraint i is satisfied for all nonnegative choices of α_k, since

$$a_i^T(x^{(k)} + \alpha_k p^{(k)}) \geq a_i^T x^{(k)} \geq b_i. \tag{4.7}$$

3. For $i \notin \mathcal{W}_k$ with $a_i^T p^{(k)} < 0$ constraint i remains satisfied only if the step length is restricted to

$$\alpha_k \leq \frac{b_i - a_i^T x^{(k)}}{a_i^T p^{(k)}}. \tag{4.8}$$

Combining these aspects this yields an explicit definition of the maximal step length by

$$\alpha_k = \min(1, s) \quad \text{with} \quad s = \min_{i \notin \mathcal{W}_k, a_i^T p^{(k)} < 0} \frac{b_i - a_i^T x^{(k)}}{a_i^T p^{(k)}}. \tag{4.9}$$

If it exists, the constraint i for which (4.9) is achieved with $\alpha_k < 1$ is called the *blocking constraint*. It is mentioned, that if the desired step is not restricted, i.e. $\alpha_k = 1$, no such a constraint exists.

4.1.3 Updating the Workingset

As long as the search directions computed by solving (4.3) are nonzero, blocking constraint indices $j \notin \mathcal{W}_k$ are successively added to the workingset:

$$\mathcal{W}_{k+1} = \mathcal{W}_k \cup \{j\}. \tag{4.10}$$

If $p^{(k)} = 0$ is observed, $x^{(k)}$ minimizes (2.11a) over its current workingset $\mathcal{W}_k$. The first-order optimality condition (2.15a) for problem (4.3) then gives

$$\sum_{i \in \mathcal{W}_k} a_i \lambda_i^{(k)} = Hx^{(k)} + f. \tag{4.11}$$

for some Lagrange multipliers $\lambda_i^{(k)}, i \in \mathcal{W}_k$. By the definition of $\lambda_i^{(k)} = 0$ for $i \notin \mathcal{W}_k$, the pair $(x^{(k)}, \lambda^{(k)})$ satisfies the KKT conditions (2.15a), (2.15b) and (2.15c). If (2.15d) also holds, $x^{(k)}$ is a KKT point for the original problem (2.11).

If there are multipliers for which the nonnegativity condition is not satisfied, i.e. $\lambda_j^{(k)} < 0$ for $j \in \mathcal{W}_k \cap \mathcal{I}$, the objective function may be decreased by dropping one of these constraints. This implies

$$\mathcal{W}_{k+1} = \mathcal{W}_k \setminus \{j\}. \tag{4.12}$$

The index j corresponding to the most negative multiplier is often dropped in practice, since the rate of decrease in the objective function is proportional to the magnitude of $p_j^{(k)}$, even when this approach is susceptible to the scaling of the constraint (cf. [77]).

4.1.4 Basic Active-Set Algorithm for Convex QP

Algorithm 6 states the fundamentals of primal active-set frameworks in convex quadratic programming and forms the basis of all reifications presented in the following. It is stated under the assumption that (2.11a) is bounded by the means of (2.11b) and (2.11c). Aspects discussed are:

1. The choice of the feasible initial point and wokingset.
2. The solution of the step EQP in line 2.
3. The determination of Lagrange multipliers in line 4.

The choices for these algorithmic details are topic of Section 4.2. Previously, the determination of feasible starting points and possibilities of warm starts as well as issues arising with degeneracy are discussed.

Algorithm 6: Basic Active-Set Method for Convex QP

```
Input : User provided feasible initial estimate $x^{(0)}$ for (2.11) and a suitable workingset
        $\mathcal{W}_0 \subseteq \mathcal{A}(x^{(0)})$. If no workingset is submitted, $\mathcal{W}_0 = \emptyset$ is chosen.
1  for $k = 0, 1, 2, \dots$ do
2      Determine $p^{(k)}$ by solving subproblem (4.3).
3      if $p^{(k)} = 0$ then
4          Compute Lagrange multipliers $\lambda_i^{(k)}$ satisfying (4.11).
5          if $\lambda_i \geq 0$ for all $i \in \mathcal{W}_k \cap \mathcal{I}$ then
6              return solution $x^* = x^{(k)}$.
7          else
8              Identify $j = \arg\min_{j \in \mathcal{W}_k \cap \mathcal{I}} \lambda_j^{(k)}$.
9              Set $x^{(k+1)} \leftarrow x^{(k)}$ and $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$.
10     else
11         Compute step length $\alpha_k$ from (4.9).
12         Set $x^{(k+1)} \leftarrow x^{(k)} + \alpha_k p^{(k)}$.
13         if blocking constraints exist then
14             Choose one blocking constraint $j$ by means of (4.9).
15             Set $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \cup \{j\}$.
16         else
17             Set $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$.
```

4.1.5 Degeneracy, Stalling and Cycling

Difficulties in active-set algorithms occour in the case of *degeneracy*. It refers to situations in which, e.g. the strict complementary condition (2.7) does not hold or linear dependence of active constraint gradients takes place. The latter fact may lead to rank deficient matrices, which are needed to be factored in the step computations.

An algorithm based on the presented concepts is said to be at a *degenerate point*, when iterates move on and off weakly active constraints in a sequence of successive iterations without any decrease in the objective function. In other words, the algorithm returns to the same active-set after some $m > 0$ iterations, i.e. $\mathcal{W}_k = \mathcal{W}_{k+m}$. This behavior is called *cycling*. Procedures trying to avoid cycling in active-set methods, e.g. EXPAND [41], have been developed. However, cycling can occur in primal active-set methods. Most dual active-set methods for strictly convex quadratic programming cannot cycle. For more details on nondegeneracy see, for example, Gill and Wong [48, 47].

4.1.6 Two-Phase Active-Set Methods and Warm Start

The work of primal active-set methods is divided into two phases. In the so called *phase 1*, the *feasibility phase*, a feasible starting point is computed. Feasibility can be achieved by the solution of a *linear program* (LP) that sums up the constraint violation at a supplied initial guess $\bar{x}^{(0)}$. With $e = (1, \dots, 1)^T$, $\gamma_i = -\operatorname{sign}(a_i^T \bar{x}^{(0)} - b_i)$ for $i \in \mathcal{E}$ and $\gamma_i = 1$ for

$i \in \mathcal{I}$ the problem to solve reads

$$\min_{x,z} \quad e^T z \tag{4.13a}$$

$$\text{s.t.} \quad a_i^T x + \gamma_i z_i = b_i, \quad i \in \mathcal{E}, \tag{4.13b}$$

$$a_i^T x + \gamma_i z_i \geq b_i, \quad i \in \mathcal{I}, \tag{4.13c}$$

$$z \geq 0. \tag{4.13d}$$

Supposing the original problem has feasible points, the optimal objective value of problem (4.13) is zero. Any of such minimizers yield a feasible point for (2.11).

In the second phase, the *optimality phase*, primal feasibility is maintained and an optimal solution as well as an approximation to the optimal active-set is computed. The resulting procedure is subsummed in Algorithm 7.

```
Algorithm 7: Two-Phase Active-Set Method
Input: User provided initial estimate $\bar{x}^{(0)}$ for (2.11)
1 Solve (4.13) to identify a feasible initial value $x^{(0)}$ for (2.11).
2 Determine $\mathcal{W}_0$ by selecting a linearly independet subset of active constraints at $x^{(0)}$.
3 Call Algorithm 6 with $x^{(0)}, \mathcal{W}_0$ for the solution of (2.11).
```

When active-set methods are warm started in the case of good knowledge of $\mathcal{A}(x^*)$ they also need a suiting feasible starting point. If such a point is not available a phase 1 would deliver one but the information about the active-set gets lost. For example, this occours in SQP where the subproblems are related and the active-set does not change much near the NLP solution. But due to errors in the linearization of the constraints it may not be possible to provide a feasible initial estimate for the QP solver.

These issues motivate approaches which combine the two phases and avoid a phase 1 by relaxation. Variants for this technique are penalty (or big-M) methods which are topic of next section.

4.2 An Elastic Primal Active-Set Method for QP

This section describes an *elastic* active-set method which combines the feasibility and optimality phase (see Algorithm 7) by relaxation. Possible infeasibility in the constraints at $x^{(0)}$ or inconsistencies in the user supplied initial workingset is soaked up by slack variables which are penalized in the objective function. Since the feasibility to simple variable bounds is easy to verify, it is assumed in the following that the starting point satisfies theses constraints. The approach handles quadratic optimization problems of the form

$$\begin{aligned}
\min_{x\in\mathbb{R}^n} \quad & q(x) = \tfrac{1}{2}x^T Hx + f^T x && && (4.14a)\\
\text{s.t.} \quad & a_i^T x = b_i, && i \in \mathcal{E}, && (4.14b)\\
& a_i^T x \in [r_{l,i}, r_{u,i}], && i \in \mathcal{R}, && (4.14c)\\
& x_i \in [b_{l,i}, b_{u,i}], && i \in \mathcal{B}. && (4.14d)
\end{aligned}$$

In anology to (3.1), inequality constraints are splitted up into lower and upper range constraints (4.14c) as well as variable bounds (4.14d). Ranges and bounds are given by $r_l, r_u \in \mathbb{R}^k \cup \{\pm\infty\}$ and $b_l, b_u \in \mathbb{R}^n \cup \{\pm\infty\}$. Values $\pm\infty$ again indicate absent limits. The index sets $\mathcal{B}$, $\mathcal{E}$ and $\mathcal{R}$ correspond to the indices of variable bounds, equality and range constraints:

$$\mathcal{B} = \{1, \dots, n\}, \quad \mathcal{E} \;\; = \{n+1, \dots, n+m\}, \quad \mathcal{R} = \{n+m+1, \dots, n+m+k\}. \quad (4.15)$$

In the following, a more detailed look onto these index sets is required. The set of indices corresponding to equality constraints is splitted up into disjunct sets comprising the indices to feasible and infeasible constraints at a given point $\bar{x} \in \mathbb{R}^n$, by

$$\mathcal{E}^f(\bar{x}) = \{i \in \mathcal{E}\colon a_i^T \bar{x} = b_i\} \quad \text{and} \quad \mathcal{E}^r(\bar{x}) = \{i \in \mathcal{E}\colon a_i^T \bar{x} \neq b_i\}. \quad (4.16)$$

Clearly, $\mathcal{E} = \mathcal{E}^f(\bar{x}) \cup \mathcal{E}^r(\bar{x})$ and $\mathcal{E}^f(\bar{x}) \cap \mathcal{E}^r(\bar{x}) = \emptyset$ holds. The same partitioning is applied to the set of indices to range constraints, which yields

$$\mathcal{R}^f(\bar{x}) = \{i \in \mathcal{R}\colon a_i^T \bar{x} \in [r_{l,i}, r_{u,i}]\} \quad \text{and} \quad \mathcal{R}^r(\bar{x}) = \{i \in \mathcal{R}\colon a_i^T \bar{x} \notin [r_{l,i}, r_{u,i}]\}. \quad (4.17)$$

with $\mathcal{R} = \mathcal{R}^f(\bar{x}) \cup \mathcal{R}^r(\bar{x})$ and $\mathcal{R}^f(\bar{x}) \cap \mathcal{R}^r(\bar{x}) = \emptyset$. For simplicity, it is assumed that feasibility can be obtained by the addition of a nonnegative slack variable,[1] yielding

$$\begin{aligned}
i \in \mathcal{E}^r(\bar{x}): \quad & a_i^T \bar{x} < b_i && \rightsquigarrow && a_i^T \bar{x} + s_i = b_i, \quad s_i \geq 0 && (4.18)\\
i \in \mathcal{R}^r(\bar{x}): \quad & a_i^T \bar{x} < r_{l,i} && \rightsquigarrow && a_i^T \bar{x} + t_i \in [r_{l,i}, r_{u,i}], \quad t_i \geq 0. && (4.19)
\end{aligned}$$

Finally, for the statement of the relaxed QP index sets to the nonnegativity constraints for slack variables and a unique mapping between these sets and the corresponding equality and range constraints are needed.

For a fixed $\bar{x} \in \mathbb{R}^n$ and $N = n + m + k$ the required mapping can be stated as

$$\sigma_{\bar{x}}\colon \mathcal{E}^r(\bar{x}) \cup \mathcal{R}^r(\bar{x}) \to \{N+1, \dots, N + |\mathcal{E}^r(\bar{x})| + |\mathcal{R}^r(\bar{x})|\}, \quad (4.20)$$

[1] In the other case, i.e. $a_i^T \bar{x} > b_i$ for any $i \in \mathcal{E}^r(\bar{x})$ or $a_i^T \bar{x} > r_{u,i}$ for any $i \in \mathcal{R}^r(\bar{x})$, the signum of the constraint is changed.

where $\sigma_{\bar{x}}(\mathcal{M}_1) \subseteq \{N+1, \dots, N+|\mathcal{E}^r(\bar{x})|\}$ for any $\mathcal{M}_1 \subseteq \mathcal{E}^r(\bar{x})$ and $\sigma_{\bar{x}}(\mathcal{M}_2) \subseteq \{N+|\mathcal{E}^r(\bar{x})|+1, \dots, N+|\mathcal{E}^r(\bar{x})|+|\mathcal{R}^r(\bar{x})|\}$ for any $\mathcal{M}_2 \subseteq \mathcal{R}^r(\bar{x})$ holds. The index sets to nonnegativity constraints for slack variables are

$$\mathcal{S}(\bar{x}) = \sigma_{\bar{x}}(\mathcal{E}^r(\bar{x})) \quad \text{and} \quad \mathcal{T}(\bar{x}) = \sigma_{\bar{x}}(\mathcal{R}^r(\bar{x})). \tag{4.21}$$

The active-set and workingset are adjusted to the introduced sets. The partitioning into feasible and relaxed constraints is applied using $\bar{x} \in \mathbb{R}^n$, which yields

$$\mathcal{A}(\bar{x}) \subseteq \mathcal{E}^f(\bar{x}) \cup \mathcal{E}^r(\bar{x}) \cup \mathcal{R}^f(\bar{x}) \cup \mathcal{R}^r(\bar{x}) \cup \mathcal{B} \cup \mathcal{S}(\bar{x}) \cup \mathcal{T}(\bar{x}). \tag{4.22}$$

Subject to $\bar{x}$ and vectors of slack variables $s = (s_i)_{i \in \mathcal{E}^r(\bar{x})} \in \mathbb{R}^{|\mathcal{E}^r(\bar{x})|}$, $t = (t_i)_{i \in \mathcal{R}^r(\bar{x})} \in \mathbb{R}^{|\mathcal{R}^r(\bar{x})|}$ and a penalty function $\phi \colon \mathbb{R}^{|\mathcal{E}^r(\bar{x})|} \times \mathbb{R}^{|\mathcal{R}^r(\bar{x})|} \to \mathbb{R}_{\geq 0}$ the relaxed problem reads

$$\begin{aligned}
\min_{x,s,t} \quad & q(x) = \tfrac{1}{2}x^T H x + f^T x + \rho\phi(s,t) && && \text{(4.23a)}\\
\text{s.t.} \quad & a_i^T x = b_i, && i \in \mathcal{E}^f(\bar{x}), && \text{(4.23b)}\\
& a_i^T x + s_i = b_i, && i \in \mathcal{E}^r(\bar{x}), && \text{(4.23c)}\\
& a_i^T x \in [r_{l,i}, r_{u,i}], && i \in \mathcal{R}^f(\bar{x}), && \text{(4.23d)}\\
& a_i^T x + t_i \in [r_{l,i}, r_{u,i}], && i \in \mathcal{R}^r(\bar{x}), && \text{(4.23e)}\\
& x_i \in [b_{l,i}, b_{u,i}], && i \in \mathcal{B}, && \text{(4.23f)}\\
& s_i \geq 0, && \sigma_x(i) \in \mathcal{S}(\bar{x}), && \text{(4.23g)}\\
& t_i \geq 0, && \tau_x(i) \in \mathcal{T}(\bar{x}), && \text{(4.23h)}
\end{aligned}$$

where the penalty parameter $\rho > 0$ is fixed and sufficiently large.

Consider the vectors of dual multipliers $z^f \in \mathbb{R}^{|\mathcal{E}^f|}$, $z^r \in \mathbb{R}^{|\mathcal{E}^r|}$ for equality constraints, $v^f = v_l^f - v_u^f \in \mathbb{R}^{|\mathcal{R}^f|}$, $v^r = v_l^r - v_u^r \in \mathbb{R}^{|\mathcal{R}^r|}$ for range constraints and $u = u_l - u_u \in \mathbb{R}^n$, $u_s \in \mathbb{R}^{|\mathcal{S}|}$, $u_t \in \mathbb{R}^{|\mathcal{T}|}$ for variable bounds and nonnegativity conditions. The Lagrangian function for (4.23) then reads

$$\begin{aligned}
\mathcal{L}(x,s,t,z^f,z^r,v^r,u,u_s,u_t) =& \frac{1}{2}x^T H x + f^T x + \rho\phi(s,t)\\
&- \sum_{i \in \mathcal{E}^f(\bar{x})} z_i^f(a_i^T x - b_i) - \sum_{i \in \mathcal{E}^r(\bar{x})} z_i^r(a_i^T x + s_i - b_i)\\
&- \sum_{i \in \mathcal{R}^f(\bar{x})} v_{l,i}^f(a_i^T x - r_{l,i}) + \sum_{i \in \mathcal{R}^f(\bar{x})} v_{u,i}^f(a_i^T x - r_{u,i}) \qquad \text{(4.24)}\\
&- \sum_{i \in \mathcal{R}^r(\bar{x})} v_{l,i}^r(a_i^T x + t_i - r_{l,i}) + \sum_{i \in \mathcal{R}^r(\bar{x})} v_{u,i}^r(a_i^T x + t_i - r_{u,i})\\
&- \sum_{i \in \mathcal{B}} u_{l,i}(x_i - b_{l,i}) + \sum_{i \in \mathcal{B}} u_{u,i}(x_i - b_{u,i})\\
&- \sum_{\sigma_{\bar{x}}(i) \in \mathcal{S}(\bar{x})} u_{s,i}s_i - \sum_{\sigma_{\bar{x}}(i) \in \mathcal{T}(\bar{x})} u_{t,i}t_i.
\end{aligned}$$

type	level	name	description
QP	1	general QP	problem before relaxation, see (4.14)
	2	relaxed QP	relaxed problem, see (4.23)
EQP	3	relaxed EQP	including all constraints w.r.t. $\mathcal{W}_k$
	4	reduced EQP	projected onto null-space of active bounds for s
	5	reduced EQP	projected onto null-space of active bounds for t
	6	step EQP	projected onto null-space of active variabel bounds
LSE	7	KKT system	linear system of equations to EQP on level 6

Table 4.1: Subproblems on different levels in the elastic ASM.

Relaxation Schemes

The penalization in the objective function of (4.23) is chosen such that it remains quadratic and the elastic approach can follow the basic idea of primal active-set methods for QP; independent of the choice of $\phi(s,t)$. This motivates the usage of ℓ_1 and ℓ_2 penalty functions $\phi\colon \mathbb{R}^{|\mathcal{E}^r(\bar{x})|} \times \mathbb{R}^{|\mathcal{R}^r(\bar{x})|} \to \mathbb{R}_{\geq 0}$, i.e.

$$\phi_1(s,t) = (\|s\|_1 + \|t\|_1) = (e^T s + e^T t), \tag{4.25a}$$

$$\phi_2(s,t) = \tfrac{1}{2}(\|s\|_2^2 + \|t\|_2^2) = \tfrac{1}{2}(s^T s + t^T t). \tag{4.25b}$$

By a proper initialization of s, t (4.23) is always feasible and it is well known, that, if (4.14) is consistent with a solution x^*, it has a solution $x^*(\rho)$ which converges to x^* for $\rho \to \infty$ (see, for example, [77]).

The elastic approach includes several subproblems on different levels in each iteration; arising problems in decreasing order are described in Table 4.1.

Step Computation: Statement of the Relaxed EQP

To improve the readability, the arguments of the introduced index sets and active constraints Jacobians are dropped in the following, i.e. $\mathcal{R}^f$ is used instead of $\mathcal{R}^f(x^{(k)})$ and $A_{\mathcal{R}^r}$ instead of $A^{(k)}_{\mathcal{R}^r}$. It is applied since the partitioning changes during the overall solution process when relaxed constraints are identified to be feasible.

The upcoming discussion frequently uses selections of rows out of the constraint Jacobians w.r.t. the k-th workingset which motivates the following definition of orthogonal gather and scatter transformations.

Definition 9 (Row-selection matrix)**.** *For an index set $\mathcal{M}$ and workingset $\mathcal{W}_k$ the row-selection matrix $P^{(k)}_{\mathcal{M}} \in \mathbb{R}^{|\mathcal{M}\cap\mathcal{W}_k| \times l}$, $l \in \mathbb{N}$, consists of row vectors $(0,\dots,0,p_j,0,\dots,0) \in \mathbb{R}^l$ with a single nonzero component $p_j = 1$ for $j \in \mathcal{M}\cap\mathcal{W}_k$.*

By construction $P^{(k)}_{\mathcal{M}}(P^{(k)}_{\mathcal{M}})^T = I \in \mathbb{R}^{|\mathcal{M}\cap\mathcal{W}_k|\times|\mathcal{M}\cap\mathcal{W}_k|}$ and $P^{(k)}_{\mathcal{M}}(P^{(k)}_{\mathcal{M}'})^T = 0 \in \mathbb{R}^{|\mathcal{M}|\times|\mathcal{M}'|}$ holds for any suitable disjunct index set $\mathcal{M}'$, i.e. $\mathcal{M}\cap\mathcal{M}' = \emptyset$.

Corresponding to the current workingset and according to Definition 9 the constraint gradients to feasible and relaxed active range constraints are selected by

$$P_{\mathcal{R}^f}^{(k)} \in \mathbb{R}^{|\mathcal{R}^f \cap \mathcal{W}_k| \times |\mathcal{R}^f|} \quad \text{and} \quad P_{\mathcal{R}^r}^{(k)} \in \mathbb{R}^{|\mathcal{R}^r \cap \mathcal{W}_k| \times |\mathcal{R}^r|}. \tag{4.26}$$

In the same way, selections for active variable bounds and nonnegativity conditions for slack variables are achieved by left multiplication with

$$P_{\mathcal{B}}^{(k)} \in \mathbb{R}^{|\mathcal{B} \cap \mathcal{W}_k| \times n}, \quad P_{\mathcal{S}}^{(k)} \in \mathbb{R}^{|\mathcal{S} \cap \mathcal{W}_k| \times |\mathcal{E}^r|}, \quad P_{\mathcal{T}}^{(k)} \in \mathbb{R}^{|\mathcal{T} \cap \mathcal{W}_k| \times |\mathcal{R}^r|}. \tag{4.27}$$

For active bound or nonnegativity constraints the elements in (4.27) form null-space basis matrices to these constraints. This property will be used in the elimination of slack variables as well as in a projection of the KKT data onto the null-space of active variable bounds.

The relaxed EQP (Table 4.1, level 3) which has to be solved in the k-th iteration to obtain a primal step $(p_x^{(k)}, p_s^{(k)}, p_t^{(k)})$ (and dual multipliers to active constraints) can now be stated as

$$\begin{aligned}
\min_{p_x, p_s, p_t} \quad & \tfrac{1}{2} p_x^T H p_x + (g^{(k)})^T p_x + \rho\phi\left(s^{(k)} + p_s, t^{(k)} + p_t\right) && \text{(4.28a)}\\
\text{s.t.} \quad & A_{\mathcal{E}^f} p_x = 0, && \text{(4.28b)}\\
& A_{\mathcal{E}^r} p_x + p_s = 0, && \text{(4.28c)}\\
& (P_{\mathcal{R}^f}^{(k)} A_{\mathcal{R}^f}) p_x = 0, && \text{(4.28d)}\\
& (P_{\mathcal{R}^r}^{(k)} A_{\mathcal{R}^r}) p_x + P_{\mathcal{R}^r}^{(k)} p_t = 0, && \text{(4.28e)}\\
& P_{\mathcal{B}}^{(k)} p_x = 0, P_{\mathcal{S}}^{(k)} p_s = 0, P_{\mathcal{T}}^{(k)} p_t = 0. && \text{(4.28f)}
\end{aligned}$$

Dual multipliers to inactive constraints are set to zero (see Section 4.1.3). Reusing the same notation for multipliers corresponding to active constraints, the Lagrangian function to (4.28) reads

$$\begin{aligned}
\mathcal{L}(p_x, p_s, p_t, z^f, z^r, v^f, v^r, u, u_s, u_t) =& \tfrac{1}{2} p_x^T H p_x + (g^{(k)})^T p_x + \rho\phi\left(s^{(k)} + p_s, t^{(k)} + p_t\right)\\
&- (z^f)^T A_{\mathcal{E}^f} p_x\\
&- (z^r)^T (A_{\mathcal{E}^r} p_x + p_s)\\
&- (v^f)^T (P_{\mathcal{R}^f}^{(k)} A_{\mathcal{R}^f} p_x)\\
&- (v^r)^T (P_{\mathcal{R}^r}^{(k)} A_{\mathcal{R}^r} p_x + P_{\mathcal{R}^r}^{(k)} p_t)\\
&- u^T P_{\mathcal{B}}^{(k)} p_x - u_s^T P_{\mathcal{S}}^{(k)} p_s - u_t^T P_{\mathcal{T}}^{(k)} p_t.
\end{aligned} \tag{4.29}$$

The KKT optimality conditions are given by

$$\begin{aligned}
\nabla_{p_x}\mathcal{L}: \quad & H p_x + (A_{\mathcal{E}^f})^T(-z^f) + (A_{\mathcal{E}^r})^T(-z^r)\\
&+ (P_{\mathcal{R}^r}^{(k)} A_{\mathcal{R}^f})^T(-v^f) + (P_{\mathcal{R}^r}^{(k)} A_{\mathcal{R}^r})^T(-v^r) + (P_{\mathcal{B}}^{(k)})^T(-u_x) = -g^{(k)}, \quad \text{(4.30a)}
\end{aligned}$$

$$\nabla_{p_s}\mathcal{L}: \qquad \rho\nabla_{p_s}\phi\left(s^{(k)}+p_s, t^{(k)}+p_t\right) - z^r - (P_{\mathcal{S}}^{(k)})^T u_s = 0, \tag{4.30b}$$

$$\nabla_{p_t}\mathcal{L}: \qquad \rho\nabla_{p_t}\phi\left(s^{(k)}+p_s, t^{(k)}+p_t\right) - (P_{\mathcal{R}^r}^{(k)})^T v^r - (P_{\mathcal{T}}^{(k)})^T u_t = 0, \tag{4.30c}$$

and

$$\nabla_{z^f}\mathcal{L}: \qquad A_{\mathcal{E}^f} p_x = 0, \tag{4.30d}$$

$$\nabla_{z^r}\mathcal{L}: \qquad A_{\mathcal{E}^r} p_x + p_s = 0, \tag{4.30e}$$

$$\nabla_{v^f}\mathcal{L}: \qquad (P_{\mathcal{R}^f}^{(k)} A_{\mathcal{R}^f}) p_x = 0, \tag{4.30f}$$

$$\nabla_{v^r}\mathcal{L}: \qquad (P_{\mathcal{R}^r}^{(k)} A_{\mathcal{R}^r}) p_x + P_{\mathcal{R}^r}^{(k)} p_t = 0, \tag{4.30g}$$

$$\nabla_{u}\mathcal{L}: \qquad P_{\mathcal{B}}^{(k)} p_x = 0, \tag{4.30h}$$

$$\nabla_{u_s}\mathcal{L}: \qquad P_{\mathcal{S}}^{(k)} p_s = 0, \tag{4.30i}$$

$$\nabla_{u_t}\mathcal{L}: \qquad P_{\mathcal{T}}^{(k)} p_t = 0. \tag{4.30j}$$

4.2.1 The Quadratic Relaxation Scheme

This subsection describes the arising subproblems on different elimination stages when the quadratic penalty function (4.25b) is used. The gradients of $\phi_2(s,t)$ in (4.30b) and (4.30c) are given by

$$\nabla_{p_s}\phi_2\left(s^{(k)}+p_s, t^{(k)}+p_t\right) = s^{(k)}+p_s, \quad \nabla_{p_t}\phi_2\left(s^{(k)}+p_s, t^{(k)}+p_t\right) = t^{(k)}+p_t. \tag{4.31}$$

In the following paragraphs the KKT conditions (4.30) are projected onto the null-space of fixed slack variables according to $\mathcal{W}_k$ at $(x^{(k)}, s^{(k)}, t^{(k)})$.

Elimination of Slacks for Equality Constraints

The null-space basis $P_{\mathcal{S}}^{(k)}$ to fixed slack variables $s^{(k)}$ is expanded by a row-selection matrix $Q_{\mathcal{S}}^{(k)} \in \mathbb{R}^{|\mathcal{S}\setminus\mathcal{W}_k|\times|\mathcal{E}^r|}$ to

$$Z_{\mathcal{S}}^{(k)} = \begin{bmatrix} P_{\mathcal{S}}^{(k)} \\ Q_{\mathcal{S}}^{(k)} \end{bmatrix} \in \mathbb{R}^{|\mathcal{E}^r|\times|\mathcal{E}^r|}. \tag{4.32}$$

This yields a decomposition of the primal step p_s and duals z^r given by

$$Z_{\mathcal{S}}^{(k)} p_s = \begin{pmatrix} P_{\mathcal{S}}^{(k)} p_s \\ Q_{\mathcal{S}}^{(k)} p_s \end{pmatrix} = \begin{pmatrix} p_s^1 \\ p_s^2 \end{pmatrix}, \quad Z_{\mathcal{S}}^{(k)} z^r = \begin{pmatrix} z^{r,1} \\ z^{r,2} \end{pmatrix}. \tag{4.33}$$

The reduced vector of duals to active nonnegativity constraints $u_s \in \mathbb{R}^{|\mathcal{S}\cap\mathcal{W}_k|}$ is lifted up to the full space of this constraint type (including multipliers corresponding to inactive constraints) by transformation with $(P_{\mathcal{S}}^{(k)})^T \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}\cap\mathcal{W}_k|}$. This implies the decomposition of u_s in the global scope by

$$Z_{\mathcal{S}}^{(k)}(P_{\mathcal{S}}^{(k)})^T u_s = \begin{pmatrix} u_s^1 \\ u_s^2 \end{pmatrix} = \begin{pmatrix} u_s^1 \\ 0 \end{pmatrix}. \tag{4.34}$$

The dual vector $u_s^2 \in \mathbb{R}^{|\mathcal{S}\setminus\mathcal{W}_k|}$ corresponds to inactive nonnegativity constraints; this implies $u_s^2 = 0$. Equation (4.30i) determines $P_{\mathcal{S}}^{(k)} p_s = p_s^1 = 0$. If the slack variable is fixed, its value is supposed to be zero, i.e. $P_{\mathcal{S}}^{(k)} s^{(k)} = 0$. Transformation of (4.30b) using $Z_{\mathcal{S}}^{(k)}$ gives

$$-z^{r,1} - u_s^1 = 0 \tag{4.35a}$$

$$p_s^2 - \tfrac{1}{\rho} z^{r,2} = -Q_{\mathcal{S}}^{(k)} s^{(k)}. \tag{4.35b}$$

The upper equation determines u_s^1 depending on $z^{r,1}$, the lower one is used for the elimination of p_s^2. Depending on $z^{r,2}$ it is

$$\begin{pmatrix} p_s^1 \\ p_s^2 \end{pmatrix} = \begin{pmatrix} 0 \\ -Q_{\mathcal{S}}^{(k)} s^{(k)} + \tfrac{1}{\rho} z^{r,2} \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} u_s^1 \\ u_s^2 \end{pmatrix} = \begin{pmatrix} -z^{r,1} \\ 0 \end{pmatrix}. \tag{4.36}$$

Because the data involved in (4.30i) and (4.35) is not modified in the following the equations are dropped to improve readability. Decomposition of (4.30e) using $Z_{\mathcal{S}}^{(k)}$ and elimination of p_s^2 using (4.35b) gives

$$P_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x = 0, \tag{4.37a}$$

$$Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x + \tfrac{1}{\rho} z^{r,2} = Q_{\mathcal{S}}^{(k)} s^{(k)}. \tag{4.37b}$$

The remaining KKT conditions read

$$\begin{aligned}
\nabla_{p_x}\mathcal{L}: \quad & Hp_x - (A_{\mathcal{E}^f})^T z^f - (P_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r})^T z^{r,1} - (Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r})^T z^{r,2} \\
& -(P_{\mathcal{R}^f}^{(k)} A_{\mathcal{R}^f})^T v^f - (P_{\mathcal{R}^r}^{(k)} A_{\mathcal{R}^r})^T v^r - (P_{\mathcal{B}}^{(k)})^T u_x = -g^{(k)}, && (4.38a) \\
\nabla_{p_t}\mathcal{L}: \quad & \rho p_t - (P_{\mathcal{R}^r}^{(k)})^T v^r - (P_{\mathcal{T}}^{(k)})^T u_t = -\rho t^{(k)}, && (4.38b) \\
\nabla_{z^f}\mathcal{L}: \quad & A_{\mathcal{E}^f} p_x = 0, && (4.38c) \\
\nabla_{z^{r,1}}\mathcal{L}: \quad & P_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x = 0, && (4.38d) \\
\nabla_{z^{r,2}}\mathcal{L}: \quad & Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x + \tfrac{1}{\rho} z^{r,2} = Q_{\mathcal{S}}^{(k)} s^{(k)}, && (4.38e) \\
\nabla_{v^f}\mathcal{L}: \quad & (P_{\mathcal{R}^f}^{(k)} A_{\mathcal{R}^f}) p_x = 0, && (4.38f) \\
\nabla_{v^r}\mathcal{L}: \quad & (P_{\mathcal{R}^r}^{(k)} A_{\mathcal{R}^r}) p_x + P_{\mathcal{R}^r}^{(k)} p_t = 0, && (4.38g) \\
\nabla_{u}\mathcal{L}: \quad & P_{\mathcal{B}}^{(k)} p_x = 0, && (4.38h) \\
\nabla_{u_t}\mathcal{L}: \quad & P_{\mathcal{T}}^{(k)} p_t = 0. && (4.38i)
\end{aligned}$$

Elimination of Slacks for Range Constraints

With similar modifications like those applied above, the slack variables corresponding to range constraints as well as dual variables to active nonnegativity constraints (4.38i) are eliminated. The following cases have to be distinguished for the projection onto the null-space of active nonnegativity constraints for range constraint slack variables:

1. Range constraint i and nonnegativity condition $\sigma(i)$ are part of $\mathcal{W}_k$.
2. $\sigma(i)$ is part of $\mathcal{W}_k$, but the corresponding range constraint i is not active w.r.t. $\mathcal{W}_k$.
3. Range constraint i is part of $\mathcal{W}_k$, but $\sigma(i)$ is not.
4. Both, range constraint i and nonnegativity condition $\sigma(i)$ are inactive w.r.t. $\mathcal{W}_k$.

Expressing the four cases in terms of the indices $\sigma(i) \in \mathcal{T}$ gives the disjoint subsets

$$\mathcal{T}_1 = (\mathcal{T} \cap \mathcal{W}_k) \cap \sigma(\mathcal{R}^r \cap \mathcal{W}_k), \tag{4.39a}$$

$$\mathcal{T}_2 = (\mathcal{T} \cap \mathcal{W}_k) \setminus \sigma(\mathcal{R}^r \cap \mathcal{W}_k), \tag{4.39b}$$

$$\mathcal{T}_3 = \sigma(\mathcal{R}^r \cap \mathcal{W}_k) \setminus (\mathcal{T} \cap \mathcal{W}_k), \tag{4.39c}$$

$$\mathcal{T}_4 = \mathcal{T} \setminus [(\mathcal{T} \cap \mathcal{W}_k) \cup \sigma(\mathcal{R}^r \cap \mathcal{W}_k)] \tag{4.39d}$$

of $\mathcal{T}$. The null-space basis $P_{\mathcal{T}}^{(k)} = \begin{bmatrix} P_{\mathcal{T}_1}^{(k)} & P_{\mathcal{T}_2}^{(k)} \end{bmatrix}^T$ to fixed slack variables $t^{(k)}$ is splitted up by the means of (4.39a) and (4.39b), and is expanded by row-selection matrices according to the index sets in (4.39c) and (4.39d), yielding

$$Z_{\mathcal{T}}^{(k)} = \begin{bmatrix} P_{\mathcal{T}_1}^{(k)} \\ P_{\mathcal{T}_2}^{(k)} \\ Q_{\mathcal{T}_3}^{(k)} \\ Q_{\mathcal{T}_4}^{(k)} \end{bmatrix} \in \mathbb{R}^{|\mathcal{R}^r| \times |\mathcal{R}^r|}, \quad Z_{\mathcal{T}}^{(k)} p_t = \begin{pmatrix} P_{\mathcal{T}_1}^{(k)} p_t \\ P_{\mathcal{T}_2}^{(k)} p_t \\ Q_{\mathcal{T}_3}^{(k)} p_t \\ Q_{\mathcal{T}_4}^{(k)} p_t \end{pmatrix} = \begin{pmatrix} p_t^1 \\ p_t^2 \\ p_t^3 \\ p_t^4 \end{pmatrix}. \tag{4.40}$$

In the same way as in the decomposition of $(P_{\mathcal{S}}^{(k)})^T u_s$ above (see (4.34)), $(P_{\mathcal{T}}^{(k)})^T \in \mathbb{R}^{|\mathcal{T}| \times |\mathcal{T} \cap \mathcal{W}_k|}$ lifts $u_t \in \mathbb{R}^{|\mathcal{T} \cap \mathcal{W}_k|}$ up to the full space $\mathbb{R}^{|\mathcal{T}|}$. In anology, $(P_{\mathcal{R}^r}^{(k)})^T \in \mathbb{R}^{|\mathcal{R}^r| \times |\mathcal{R}^r \cap \mathcal{W}_k|}$ lifts $v^r \in \mathbb{R}^{|\mathcal{R}^r \cap \mathcal{W}_k|}$ up to the full space $\mathbb{R}^{|\mathcal{R}^r|}$. The decomposition of the lifted vectors of dual multipliers can be stated as

$$Z_{\mathcal{T}}^{(k)} (P_{\mathcal{T}}^{(k)})^T u_t = \begin{pmatrix} u_t^1 \\ u_t^2 \\ u_t^3 \\ u_t^4 \end{pmatrix} = \begin{pmatrix} u_t^1 \\ u_t^2 \\ 0 \\ 0 \end{pmatrix}, \quad Z_{\mathcal{T}}^{(k)} (P_{\mathcal{R}^r}^{(k)})^T v^r = \begin{pmatrix} v^{r,1} \\ v^{r,2} \\ v^{r,3} \\ v^{r,4} \end{pmatrix} = \begin{pmatrix} v^{r,1} \\ 0 \\ v^{r,3} \\ 0 \end{pmatrix}. \tag{4.41}$$

The dual vectors u_t^3, u_t^4 correspond to inactive nonnegativity constraints which yields $u_t^3 = 0$, $u_t^4 = 0$. $v^{r,2}, v^{r,4}$ correspond to inactive range constraints implying $v^{r,2} = 0$ and $v^{r,4} = 0$.

Equation (4.38i) implies $p_t^1 = 0$ and $p_t^2 = 0$. This also indicates that the slack variables $P_{\mathcal{T}_1}^{(k)} t^{(k)}$ and $P_{\mathcal{T}_2}^{(k)} t^{(k)}$ vanish. Transformation of (4.38b) using the decompositions stated in (4.40) and (4.41) gives

$$
\begin{aligned}
-v^{r,1} - u_t^1 &= 0, && (4.42a)\\
-u_t^2 &= 0, && (4.42b)\\
p_t^3 - \tfrac{1}{\rho} v^{r,3} &= -Q_{\mathcal{T}_3}^{(k)} t^{(k)}, && (4.42c)\\
p_t^4 &= -Q_{\mathcal{T}_4}^{(k)} t^{(k)}. && (4.42d)
\end{aligned}
$$

Equation (4.42d) determines p_t^4 and (4.42c) determines p_t^3 depending on $v^{r,3}$. It is used for the elimination of p_t^3. In summary, the step vector for range slacks and duals u_t read

$$
\begin{pmatrix} p_t^1 \\ p_t^2 \\ p_t^3 \\ p_t^4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -Q_{\mathcal{T}_3}^{(k)} t^{(k)} + \frac{1}{\rho} v^{r,3} \\ -Q_{\mathcal{T}_4}^{(k)} t^{(k)} \end{pmatrix}, \quad \begin{pmatrix} u_t^1 \\ u_t^2 \\ u_t^3 \\ u_t^4 \end{pmatrix} = \begin{pmatrix} -v^{r,1} \\ 0 \\ 0 \\ 0 \end{pmatrix}, \qquad (4.43)
$$

The other equations in (4.42) determine the remaining parts of u_t. Again, the involved data is not modified in later projections and can be dropped to improve the readability.

Before the reduced KKT conditions can be stated, (4.38g) is splitted up according to $v^{r,1}$, $v^{r,3}$ using $P_{\mathcal{R}^r}^{(k)} = \begin{bmatrix} P_{\mathcal{T}_1}^{(k)} & Q_{\mathcal{T}_3}^{(k)} \end{bmatrix}^T$. After the elimination of p_t^3 the equations read

$$
\begin{aligned}
(P_{\mathcal{T}_1}^{(k)} A_{\mathcal{R}^r}) p_x &= 0, && (4.44)\\
(Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{R}^r}) p_x + \tfrac{1}{\rho} v^{r,3} &= Q_{\mathcal{T}_3}^{(k)} t^{(k)}. && (4.45)
\end{aligned}
$$

Now the reduced KKT conditions can be stated.

$$
\begin{aligned}
\nabla_{p_x}\mathcal{L}: \quad & H p_x - (A_{\mathcal{E}^f})^T z^f - (P_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r})^T z^{r,1} - (Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r})^T z^{r,2} \\
& -(P_{\mathcal{R}^f}^{(k)} A_{\mathcal{R}^f})^T v^f - (P_{\mathcal{T}_1}^{(k)} A_{\mathcal{R}^r})^T v^{r,1} - (Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{R}^r})^T v^{r,3} && (4.46a)\\
& \qquad -(P_{\mathcal{B}}^{(k)})^T u_x = -g^{(k)}, \\
\nabla_{z^f}\mathcal{L}: \quad & A_{\mathcal{E}^f} p_x = 0, && (4.46b)\\
\nabla_{z^{r,1}}\mathcal{L}: \quad & P_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x = 0, && (4.46c)\\
\nabla_{z^{r,2}}\mathcal{L}: \quad & Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x + \tfrac{1}{\rho} z^{r,2} = Q_{\mathcal{S}}^{(k)} s^{(k)}, && (4.46d)\\
\nabla_{v^f}\mathcal{L}: \quad & P_{\mathcal{R}^f}^{(k)} A_{\mathcal{R}^f} p_x = 0, && (4.46e)\\
\nabla_{v^{r,1}}\mathcal{L}: \quad & P_{\mathcal{T}_1}^{(k)} A_{\mathcal{R}^r} p_x = 0, && (4.46f)\\
\nabla_{v^{r,3}}\mathcal{L}: \quad & Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{R}^r} p_x + \tfrac{1}{\rho} v^{r,3} = Q_{\mathcal{T}_3}^{(k)} t^{(k)}, && (4.46g)\\
\nabla_{u}\mathcal{L}: \quad & P_{\mathcal{B}}^{(k)} p_x = 0, && (4.46h)
\end{aligned}
$$

Augmentation of the projected KKT Data

The primal feasibility conditions (4.46b) to (4.46h) consist of three types of equations wich can be augmented for easier reading. Equations only comprising p_x, with the special case of (4.46h) for active variable bounds and those depending on p_x and some dual variables $z^{r,2}$, $v^{r,3}$.

Consider the index sets which encapsulate the indices of feasible and relaxed active constraints, stated as

$$\mathcal{M}_k^f = \mathcal{E}^f \cup \sigma^{-1}(\mathcal{S} \cap \mathcal{W}_k) \cup (\mathcal{R}^f \cap \mathcal{W}_k) \cup \mathcal{T}_1, \tag{4.47a}$$

$$\mathcal{M}_k^r = \sigma^{-1}(\mathcal{S} \setminus \mathcal{W}_k) \cup \mathcal{T}_3. \tag{4.47b}$$

Dual variables corresponding to these sets are denoted by the augmented vectors

$$\lambda^f = \begin{pmatrix} z^f \\ z^{r,1} \\ v^f \\ v^{r,1} \end{pmatrix} \in \mathbb{R}^{|\mathcal{M}_k^f|} \quad \text{and} \quad \lambda^r = \begin{pmatrix} z^{r,2} \\ v^{r,3} \end{pmatrix} \in \mathbb{R}^{|\mathcal{M}_k^r|}. \tag{4.48}$$

Now, by augmentation of the KKT data by

$$A_{\mathcal{M}_k^f} = \begin{bmatrix} A_{\mathcal{E}^f} \\ P_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} \\ P_{\mathcal{R}^f}^{(k)} A_{\mathcal{R}^f} \\ P_{\mathcal{T}_1}^{(k)} A_{\mathcal{R}^r} \end{bmatrix}, \quad A_{\mathcal{M}_k^r} = \begin{bmatrix} Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} \\ Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{R}^r} \end{bmatrix}, \quad c_{\mathcal{M}_k^r}^{(k)} = \begin{pmatrix} Q_{\mathcal{S}}^{(k)} s^{(k)} \\ Q_{\mathcal{T}_3}^{(k)} t^{(k)} \end{pmatrix} \tag{4.49}$$

the projected KKT conditions can be rewritten as

$$\nabla_{p_x}\mathcal{L}: \qquad H p_x - A_{\mathcal{M}_k^f}^T \lambda^f - A_{\mathcal{M}_k^f}^T \lambda^r - (P_{\mathcal{B}}^{(k)})^T u_x = -g^{(k)}, \tag{4.50a}$$

$$\nabla_{\lambda^f}\mathcal{L}: \qquad A_{\mathcal{M}_k^f} p_x = 0, \tag{4.50b}$$

$$\nabla_{\lambda^r}\mathcal{L}: \qquad A_{\mathcal{M}_k^r} p_x + \tfrac{1}{\rho}\lambda^r = c_{\mathcal{M}_k^r}^{(k)}, \tag{4.50c}$$

$$\nabla_{u}\mathcal{L}: \qquad P_{\mathcal{B}}^{(k)} p_x = 0. \tag{4.50d}$$

4.2.2 The Linear Relaxation Scheme

This subsection summarizes the elimination of slack variables in the step computation when the linear penalty function (4.25a) is used. The gradients of $\phi_1(s,t)$ in (4.30b) and (4.30c) are given by

$$\nabla_{p_s}\phi_1\left(s^{(k)} + p_s, t^{(k)} + p_t\right) = e \in \mathbb{R}^{|\mathcal{E}^r|}, \quad \nabla_{p_t}\phi_1\left(s^{(k)} + p_s, t^{(k)} + p_t\right) = e \in \mathbb{R}^{|\mathcal{R}^r|}. \tag{4.51}$$

As seen above, the KKT conditions (4.30) are projected onto the null-space of fixed slack variables. The porposed projections and decompositions stated in Section 4.2.1 do not change and are not repeated in the following.

The KKT conditions change in (4.30b) and (4.30c). Using (4.51) the equations read

$$\nabla_{p_s}\mathcal{L}: \qquad -z^r - (P_{\mathcal{S}}^{(k)})^T u_s = -\rho e, \tag{4.52a}$$

$$\nabla_{p_t}\mathcal{L}: \qquad -(P_{\mathcal{R}^r}^{(k)})^T v^r - (P_{\mathcal{T}}^{(k)})^T u_t = -\rho e. \tag{4.52b}$$

The upcoming paragraphs state the differences to the quadratic penalty approach.

Projection onto the Null-Space of Slack Bounds for Equality Constraints

The dual vector $u_s^2 \in \mathbb{R}^{|\mathcal{S}\setminus\mathcal{W}_k|}$ vanishes, since it corresponds to inactive nonnegativity constraints. Decomposition of (4.52a) using $Z_{\mathcal{S}}^{(k)}$ gives

$$-z^{r,1} - u_s^1 = -\rho P_{\mathcal{S}}^{(k)} e, \tag{4.53a}$$

$$-z^{r,2} = -\rho Q_{\mathcal{S}}^{(k)} e \tag{4.53b}$$

which determines u_s^1 depending on $z^{r,1}$ as well as $z^{r,2}$. (4.53b) is used to eliminate the dual multipliers $z^{r,2}$ in (4.30a). The constant term $\rho(Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r})^T Q_{\mathcal{S}}^{(k)} e$ is absorbed into the KKT right-hand side. Since $p_s^1 = 0$ is fixed by (4.30i), splitting up the relaxed equality constraints yields

$$P_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x = 0, \tag{4.54a}$$

$$Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x + p_s^2 = 0. \tag{4.54b}$$

The latter equation determines p_s^2 and is dropped to improve the readability, while equation (4.54b) remains unchanged in the following. In summary, the step vector for slacks to equality and dual multipliers u_s (depending on $z^{r,1}$) are

$$\begin{pmatrix} p_s^1 \\ p_s^2 \end{pmatrix} = \begin{pmatrix} 0 \\ -Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x \end{pmatrix}, \quad \begin{pmatrix} u_s^1 \\ u_s^2 \end{pmatrix} = \begin{pmatrix} \rho P_{\mathcal{S}}^{(k)} e - z^{r,1} \\ 0 \end{pmatrix}. \tag{4.55}$$

Projection onto the Null-Space of Slack Bounds for Range Constraints

Transformation of the dual feasibility constraint (4.30c) using the representations

$$P_{\mathcal{T}}^{(k)} = \begin{bmatrix} P_{\mathcal{T}_1}^{(k)} \\ P_{\mathcal{T}_2}^{(k)} \end{bmatrix}, \quad P_{\mathcal{R}^r}^{(k)} = \begin{bmatrix} P_{\mathcal{T}_1}^{(k)} \\ Q_{\mathcal{T}_3}^{(k)} \end{bmatrix} \tag{4.56}$$

gives the following decomposition of $\nabla_{p_t}\mathcal{L}(p_t, \lambda^{(k)})$:

$$-v^{r,1} - u_t^1 = -\rho P_{\mathcal{T}_1}^{(k)} e, \tag{4.57a}$$

$$-u_t^2 = -\rho P_{\mathcal{T}_2}^{(k)} e, \tag{4.57b}$$

$$-v^{r,3} = -\rho Q_{\mathcal{T}_3}^{(k)} e, \tag{4.57c}$$

$$0 = -\rho Q_{\mathcal{T}_4}^{(k)} e. \tag{4.57d}$$

Equation (4.57a) determines u_t^1 depending on $v^{r,1}$. The dual multipliers to the pairs of relaxed range constraint where whether i or $\sigma(i)$ is part of the current workingset are fixed to the size of the penalty parameter ρ by (4.57b) and (4.57c). The latter equation is also used to eliminate $v^{r,3}$ in (4.30a); the constant term $\rho(Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{E}^r})^T Q_{\mathcal{T}_3}^{(k)} e$ is absorbed into the KKT right-hand side.

More attention has to be paid to equation (4.57d). It enforces that the fourth case never appears, i.e. $\mathcal{T}_4 = \emptyset$. Otherwise the KKT conditions suffer from the loss of regularity and can not be solved. The effect can be avoided by convexification of the KKT system; adding a small fraction ν_i to $(p_t^4)_i$ for $i \in \mathcal{T}_4$ leading to search directions $(p_t^4)_i = \frac{\rho}{\nu_i}$. It has shown to be effective in numerical computations to choose $\nu_i = -(Q_{\mathcal{T}_4}^{(k)} A_{\mathcal{R}^r} p_x)_i$ such that a full step, i.e. $\alpha_k = 1$, provides feasibility in the corresponding constraint. This procedure is comparable with a combination of the linear and quadratic penalty approach.

Transformation of (4.30g) using $p_t^1 = 0$ from (4.38i), which also states $p_t^2 = 0$, results in

$$P_{\mathcal{T}_1}^{(k)} A_{\mathcal{R}^r} p_x = 0, \tag{4.58a}$$

$$Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{R}^r} p_x + p_t^3 = 0. \tag{4.58b}$$

The latter equation is used for the determination of p_t^3 depending on p_x. If $\mathcal{T}_4 = \emptyset$ is ensured, the step data and dual multipliers read

$$\begin{pmatrix} p_t^1 \\ p_t^2 \\ p_t^3 \\ p_t^4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{R}^r} p_x \\ 0 \end{pmatrix}, \quad \begin{pmatrix} u_t^1 \\ u_t^2 \\ u_t^3 \\ u_t^4 \end{pmatrix} = \begin{pmatrix} \rho P_{\mathcal{T}_1}^{(k)} e - v^{r,1} \\ \rho P_{\mathcal{T}_2}^{(k)} e \\ 0 \\ 0 \end{pmatrix}. \tag{4.59}$$

By stating the KKT right-hand side vector including the absorbed data by

$$\tilde{g}^{(k)} = g^{(k)} - \rho(Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r})^T Q_{\mathcal{S}}^{(k)} e - \rho(Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{E}^r})^T Q_{\mathcal{T}_3}^{(k)} e \tag{4.60}$$

the projected KKT conditions read

$$\begin{aligned} \nabla_{p_x}\mathcal{L} : \quad & H p_x - (A_{\mathcal{E}^f})^T z^f - (P_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r})^T z^{r,1} \\ & -(P_{\mathcal{R}^r}^{(k)} A_{\mathcal{R}^f})^T v^f - (P_{\mathcal{T}_1}^{(k)} A_{\mathcal{R}^r})^T v^{r,1} - (P_{\mathcal{B}}^{(k)})^T u_x = -\tilde{g}^{(k)} \end{aligned} \tag{4.61a}$$

and

$$\begin{aligned}
\nabla_{z^f}\mathcal{L}: && A_{\mathcal{E}^f} p_x &= 0, && \text{(4.61b)}\\
\nabla_{z^{r,1}}\mathcal{L}: && P_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r} p_x &= 0, && \text{(4.61c)}\\
\nabla_{v^f}\mathcal{L}: && P_{\mathcal{R}^f}^{(k)} A_{\mathcal{R}^f} p_x &= 0, && \text{(4.61d)}\\
\nabla_{v^{r,1}}\mathcal{L}: && P_{\mathcal{T}_1}^{(k)} A_{\mathcal{R}^r} p_x &= 0, && \text{(4.61e)}\\
\nabla_{u}\mathcal{L}: && P_{\mathcal{B}}^{(k)} p_x &= 0. && \text{(4.61f)}
\end{aligned}$$

Augmentation of the projected KKT Data

Using the same notation for augmented KKT data as in (4.47) to (4.49) the KKT optimality conditions stated above can be rewritten as

$$\begin{aligned}
\nabla_{p_x}\mathcal{L}: && Hp_x - A_{\mathcal{M}_k^f}^T \lambda^f - (P_{\mathcal{B}}^{(k)})^T u_x &= -\tilde{g}^{(k)}, && \text{(4.62a)}\\
\nabla_{\lambda^f}\mathcal{L}: && A_{\mathcal{M}_k^f} p_x &= 0, && \text{(4.62b)}\\
\nabla_{u}\mathcal{L}: && P_{\mathcal{B}}^{(k)} p_x &= 0. && \text{(4.62c)}
\end{aligned}$$

The same projections used in the linear realaxation scheme are not used for the quadratic relaxation scheme, because the superstructure of the resulting linear system is more common to exploitation of structure. Direct elimination of range duals $t^{r,3}$ results in a dense Hessian due to the modification

$$\begin{aligned}
\tilde{H} &= H + \rho(Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r})^T (Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r}) + \rho(Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{R}^r})^T (Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{R}^r}), && \text{(4.63a)}\\
\tilde{g}^{(k)} &= g^{(k)} + \rho(Q_{\mathcal{S}}^{(k)} A_{\mathcal{E}^r})^T Q_{\mathcal{S}}^{(k)} s^{(k)} + \rho(Q_{\mathcal{T}_3}^{(k)} A_{\mathcal{R}^r})^T Q_{\mathcal{T}_3}^{(k)} t^{(k)}. && \text{(4.63b)}
\end{aligned}$$

If the penalization parameter is large the resulting KKT system may also get ill conditioned and hard to solve. This directly effects the quality of the obtained search direction and may result in failure in the determination of the optimal active-set. But, for small choices of ρ the linear relaxation scheme enables the employability of specialized KKT algorithms wich are designed to solve linear systems of type (2.12).

4.2.3 Projection onto the Null-Space of Variable Bounds

In comparison, the projected KKT conditions (4.62) in the linear relaxation scheme results in a linear system equivalent to the one in the step computation in a standard primal active-set method, see (4.4). The quadratic scheme yields a more detailed set of linear equations, see (4.50). Because of that, the projection onto the null-space of active variable bounds is presented for the more general latter case.

The null-space basis $P_{\mathcal{B}}^{(k)} \in \mathbb{R}^{|\mathcal{B}\cap\mathcal{W}_k|\times n}$ to fixed variables $x^{(k)}$ is expanded by a row-selection matrix $Q_{\mathcal{B}}^{(k)} \in \mathbb{R}^{|\mathcal{B}\setminus\mathcal{W}_k|\times n}$ to the basis matrix

$$Z_{\mathcal{S}}^{(k)} = \begin{bmatrix} P_{\mathcal{B}}^{(k)} \\ Q_{\mathcal{B}}^{(k)} \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{4.64}$$

This yields a decomposition of the primal step p_x and duals u_x given by

$$Z_{\mathcal{B}}^{(k)} p_x = \begin{pmatrix} P_{\mathcal{B}}^{(k)} p_x \\ Q_{\mathcal{B}}^{(k)} p_x \end{pmatrix} = \begin{pmatrix} p_x^1 \\ p_x^2 \end{pmatrix}, \quad A_{\mathcal{M}_k^f} p_x = 0, \quad (P_{\mathcal{B}}^{(k)})^T u_x = \begin{pmatrix} u_x^1 \\ u_x^2 \end{pmatrix}. \tag{4.65}$$

The reduced vector of duals to active variable bounds $u_x \in \mathbb{R}^{|\mathcal{B}\cap\mathcal{W}_k|}$ is therby lifted up to the full space of this constraint type (including multipliers corresponding to active (u_x^1) and inactive $(u_x^2 = 0)$ bounds) by transformation with $(P_{\mathcal{B}}^{(k)})^T \in \mathbb{R}^{n\times|\mathcal{B}\cap\mathcal{W}_k|}$.

Consider the KKT conditions (4.50). Equation (4.50d) determines $p_x^1 = 0$. Transformation of (4.50b) using the decomposition stated above gives

$$A_{\mathcal{M}_k^f} (Z_{\mathcal{B}}^{(k)})^T Z_{\mathcal{B}}^{(k)} p_x = A_{\mathcal{M}_k^f} (Q_{\mathcal{B}}^{(k)})^T p_x^2 = 0. \tag{4.66}$$

In analogy, equation (4.50c) is rewritten as

$$A_{\mathcal{M}_k^r} (Q_{\mathcal{B}}^{(k)})^T p_x^2 + \tfrac{1}{\rho}\lambda^r = c_{\mathcal{M}_k^r}^{(k)}. \tag{4.67}$$

Modifications in $\nabla_{p_x}\mathcal{L}$ incorporate an additional multiplication with $Z_{\mathcal{B}}^{(k)}$, decomposing (4.50a) into

$$P_{\mathcal{B}}^{(k)} H (Q_{\mathcal{B}}^{(k)})^T p_x^2 - P_{\mathcal{B}}^{(k)} A_{\mathcal{M}_k^f}^T \lambda^f - P_{\mathcal{B}}^{(k)} A_{\mathcal{M}_k^f}^T \lambda^r - u_x^1 = -P_{\mathcal{B}}^{(k)} g^{(k)}, \tag{4.68a}$$

$$Q_{\mathcal{B}}^{(k)} H (Q_{\mathcal{B}}^{(k)})^T p_x^2 - Q_{\mathcal{B}}^{(k)} A_{\mathcal{M}_k^f}^T \lambda^f - Q_{\mathcal{B}}^{(k)} A_{\mathcal{M}_k^f}^T \lambda^r = -Q_{\mathcal{B}}^{(k)} g^{(k)}. \tag{4.68b}$$

Equation (4.68a) determines u_x^1 bepending on p_x^2 and multipliers λ^f, λ^r. The data is not modified in the following and is dropped to improve the readability. By redefinition of the modified data with

$$\bar{H} = Q_{\mathcal{B}}^{(k)} H (Q_{\mathcal{B}}^{(k)})^T, \quad \bar{A}_{\mathcal{M}_k^f} = A_{\mathcal{M}_k^f} (Q_{\mathcal{B}}^{(k)})^T, \quad \bar{A}_{\mathcal{M}_k^r} = A_{\mathcal{M}_k^r} (Q_{\mathcal{B}}^{(k)})^T, \quad \bar{g}^{(k)} = Q_{\mathcal{B}}^{(k)} g^{(k)}$$

the KKT conditions of the step EQP (Table 4.1, level 6) read

$$\nabla_{p_x}\mathcal{L}: \qquad \bar{H} p_x^2 - \bar{A}_{\mathcal{M}_k^f}^T \lambda^f - \bar{A}_{\mathcal{M}_k^f}^T \lambda^r = -\bar{g}^{(k)}, \tag{4.69a}$$

$$\nabla_{\lambda^f}\mathcal{L}: \qquad \bar{A}_{\mathcal{M}_k^f} p_x^2 = 0, \tag{4.69b}$$

$$\nabla_{\lambda^r}\mathcal{L}: \qquad \bar{A}_{\mathcal{M}_k^r} p_x^2 + \tfrac{1}{\rho}\lambda^r = c_{\mathcal{M}_k^r}^{(k)}. \tag{4.69c}$$

4.2.4 Projected KKT System and ASM Expansion

The optimality conditions (4.69) can be written as the *projected KKT system* $K^{(k)}y^{(k)} = r^{(k)}$

$$\begin{bmatrix} \bar{H} & (\bar{A}_{\mathcal{M}_k^f})^T & (\bar{A}_{\mathcal{M}_k^r})^T \\ \bar{A}_{\mathcal{M}_k^f} & & \\ \bar{A}_{\mathcal{M}_k^r} & & -\frac{1}{\rho}I \end{bmatrix} \begin{pmatrix} p_x^2 \\ -\lambda^f \\ -\lambda^r \end{pmatrix} = \begin{pmatrix} -\bar{g}^{(k)} \\ 0 \\ c_{\mathcal{M}_k^r}^{(k)} \end{pmatrix} \tag{4.70}$$

with $K^{(k)} \in \mathbb{R}^{N_k \times N_k}$, $N_k = |\mathcal{B} \setminus \mathcal{W}_k| + |\mathcal{E}| + |\mathcal{R} \cap \mathcal{W}_k|$. Upon solving (4.70) the solution is expanded with respect to the applied projections. Expansion 1 lifts the primal part $p_x^2 \in \mathbb{R}^{|\mathcal{B} \setminus \mathcal{W}_k|}$ up onto the full space $\mathbb{R}^n$ by

$$p_x^{(k)} = (Q_{\mathcal{B}}^{(k)})^T p_x^2 \in \mathbb{R}^n. \tag{4.71}$$

Expansion 2 determines the stepdata for slack variables using the projections presented above and reorders the components of λ^f, λ^r into the correct components of the vectors v_l, v_u and u_l, u_u.

In summary, the original set of KKT conditions is first projected due to the elimination of slack variables and afterwards projected onto the null-space of active variable bounds. The result is system (4.70). The smaller system is solved and its solution is then expanded to a solution of the complete set of KKT conditions.

4.2.5 Determination of the Step Lenght and Updating the Workingset

Once a search direction $p_x^2 \in \mathbb{R}^{|\mathcal{B} \setminus \mathcal{W}_k|}$ has been determined by the solution of (4.70) it is projected onto the full space, giving $(Q_{\mathcal{B}}^{(k)})^T p_x^2 = p_x^{(k)} \in \mathbb{R}^n$. Stepvectors for slack variables $p_s^{(k)}$, $p_t^{(k)}$ are determined as stated in the preceding paragraphs. The step lenght α_k is mostly chosen in analogy to the formalism of the basic active-set algorithm presented in Section 4.1.2 extended to lower and upper limits. It is chosen in the sense of maximal decrease of the penalized objective function in the view of some special aspects according to the relaxation which is discussed next.

Consider an inactive relaxed range constraint $i \in \mathcal{R}^r \setminus \mathcal{W}_k$ with a lower and upper limit where the lower one is injured. Without modification of the step determination a desired step would be truncated whenever i becomes feasible, because it implies that i itself and the nonegativity constraint $\sigma(i)$ of the corresponding slack variable are active. Since the nonegativity condition is linearly independent to all constraints $i \in \mathcal{E} \cup \mathcal{R}$ an unnecessary slow down of the algorithm can be avoided by multiple changes in the workingset when it comes to active bounds for slack variables. $\sigma(i)$ can be added to the workingset whenever $\alpha_k \geq \alpha_k^{\text{low}}$. The step is only blocked by the upper limit yielding $\alpha_k \leq \alpha_k^{\text{up}}$. This is illustrated in Figure 4.1. By propper scaling of $p_{t,i}^{(k)}$ the next iterate will comprise $t_i^{(k+1)} = 0$. Furthermore, if additional information about the constraints is supplied, e.g. full row

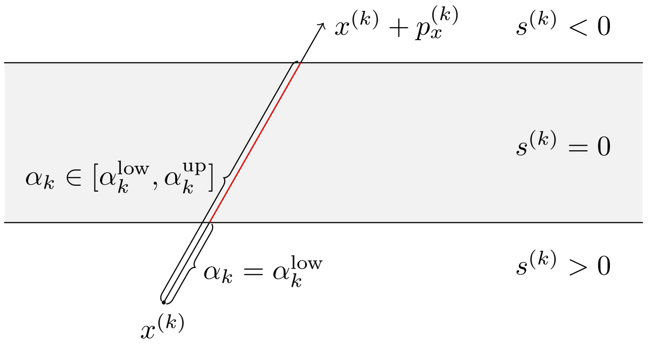

$x^{(k)} + p_x^{(k)}$
$s^{(k)} < 0$
$s^{(k)} = 0$
$\alpha_k \in [\alpha_k^{\text{low}}, \alpha_k^{\text{up}}]$
$\alpha_k = \alpha_k^{\text{low}}$
$s^{(k)} > 0$
$x^{(k)}$

Figure 4.1: Determination of the step length α_k for relaxed range constraints.

rank of $A_{\mathcal{E}}$ and absence of range constraints and variable bounds, multiple changes in the working set can be allowed to speed up the determination of the optimal active-set.

If constraint i is relaxed and active, i.e. $i \in \mathcal{M}_k^r$, the step is truncated by choosing $\alpha_k = \alpha_k^{\text{low}}$. In this case $\sigma(i)$ is added to the workingset and i has to be dropped from $\mathcal{W}_k$ to ensure the LICQ.

The elastic active-set method may also be run in an aggressive-mode wherein multiple changes in the workingset are enforced. By this relaxed constraints which are active in iteration k remain active even when they become feasible at the next iterate. The user has to provide, that the LICQ holds during the complete optimization process.

The cases to be observed in the determination of the step lenght are:

1. For all $i \in \mathcal{M}_k^f$ every choice of α_k yields a next iterate which remains feasible w.r.t. constraint i. See Section 4.1.2.

2. For $i \in \mathcal{R}^f \cup \mathcal{B}$ and $i \notin \mathcal{W}_k$ with a lower limit and $a_i^T p_x^{(k)} \geq 0$ constraint i is satisfied for all nonnegative choices of α_k, because $p_{t,i}^{(k)} = t_i^{(k)} = 0$ and

$$a_i^T(x^{(k)} + \alpha_k p_x^{(k)}) + t_i^{(k)} + \alpha_k p_{t,i}^{(k)} \geq a_i^T x^{(k)} \geq b_i, \quad b_i \in \{b_{l,i}, r_{l,i}\}.$$

   In analogy upper variable bounds or range constraints $i \in (\mathcal{R}^f \cup \mathcal{B}) \setminus \mathcal{W}_k$ where $a_i^T p_x^{(k)} \leq 0$ holds are satisfied for every $\alpha_k \geq 0$, since

$$a_i^T(x^{(k)} + \alpha_k p_x^{(k)}) + t_i^{(k)} + \alpha_k p_{t,i}^{(k)} \leq a_i^T x^{(k)} \leq b_i, \quad b_i \in \{b_{u,i}, r_{u,i}\}.$$

3. For $i \in \mathcal{R}^f \cup \mathcal{B}$ and $i \notin \mathcal{W}_k$ the step lenght is chosen such that every variable bound or range constraint i remains satisfied w.r.t. its lower and upper limits. If $a_i^T p_x^{(k)} < 0$ holds for lower limits and $a_i^T p_x^{(k)} > 0$ for upper ones, this yields

$$\alpha_k \leq \frac{b_i - a_i^T x^{(k)}}{a_i^T p_x^{(k)}}, \quad b_i \in \{r_{l,i}, r_{u,i}, b_{l,i}, b_{u,i}\}. \tag{4.72}$$

4. For all $i \in \mathcal{E}^r \cup \mathcal{R}^r$ (active or inactive) every choice of α_k yields a next iterate which does not increase the infeasibility, i.e. $s_i^{(k+1)} \leq s_i^{(k)}$ and $t_i^{(k+1)} \leq t_i^{(k)}$. This enforces that for any $\alpha_k > 0$ it holds

$$\begin{aligned} p_{s,i}^{(k)}, p_{t,i}^{(k)} &\leq 0, \quad \text{if the lower limit of } i \text{ is violated,} \\ p_{s,i}^{(k)}, p_{t,i}^{(k)} &\geq 0, \quad \text{if the upper limit of } i \text{ is violated.} \end{aligned}$$

5. For $i \in \mathcal{R}^r$ and $i \notin \mathcal{W}_k$ constraint i remains satisfied. If i has a violated lower limit, i.e. $t_i^{(k)} \geq 0$, and $a_i^T p_x^{(k)} + p_{t,i}^{(k)} \geq 0$ holds, it is

$$a_i^T (x^{(k)} + \alpha_k p_x^{(k)}) + t_i^{(k)} + \alpha_k p_{t,i}^{(k)} \geq a_i^T x^{(k)} + t_i^{(k)} \geq r_{l,i}$$

for every nonnegative choice of α_k. If i has a violated upper limit, i.e. $t_i^{(k)} \leq 0$, and $a_i^T p_x^{(k)} + p_{t,i}^{(k)} \leq 0$ holds, it is

$$a_i^T (x^{(k)} + \alpha_k p_x^{(k)}) + t_i^{(k)} + \alpha_k p_{t,i}^{(k)} \leq a_i^T x^{(k)} + t_i^{(k)} \leq r_{u,i}.$$

6. For $i \in \mathcal{R}^r \setminus \mathcal{W}_k$ the relaxed constraint i remains satisfied. The step length is chosen in analogy to 3. but distincting two cases:

   a) Whenever the lower limit of constraint i is violated its upper limit is obviously not and vice versa. If $a_i^T x^{(k)} < r_{l,i}$, i.e. $t_i^{(k)} > 0$, and $a_i^T p_x^{(k)} > 0$ and $p_{t,i}^{(k)} < 0$ holds the step is only truncated by reaching the upper limit. Therefor the step length is determined by (4.72). The same formula is used if $a_i^T x^{(k)} > r_{u,i}$, i.e. $t_i^{(k)} < 0$, and $a_i^T p_x^{(k)} < 0$ and $p_{t,i}^{(k)} > 0$. In these cases i is a blocking constraint and i and $\sigma(i)$ are included into $\mathcal{W}_{k+1}$.

   b) If the lower limit of constraint i is violated and $a_i^T p_x^{(k)} + p_{t,i}^{(k)} < 0$ holds, but $a_i^T p_x^{(k)} \leq 0$ or $p_{t,i}^{(k)} \geq 0$ the step is truncated by choosing

$$\alpha_k \leq \frac{b_i - (a_i^T x^{(k)} + t^{(k)})}{a_i^T p_x^{(k)} + p_{t,i}^{(k)}} = 0, \quad b_i \in \{r_{l,i}, r_{u,i}\}. \tag{4.73}$$

   The same is applied if the upper limit of i is violated and it is $a_i^T p_x^{(k)} + p_{t,i}^{(k)} > 0$, but $a_i^T p_x^{(k)} \geq 0$ or $p_{t,i}^{(k)} \leq 0$. In these cases i is added to the workingset.

Combining these aspects this leads to an explicit definition of the maximal step length $\alpha_k \in [0, 1]$. The determination is achieved by calling Algorithm 8.

Updates in the workingset $\mathcal{W}_k$ are nearly identical to the presented procedure in Section 4.1.3 extended to the index sets used in the elastic approach. The differences rely on that active variable bounds for slack variables are never dropped from the workingset (see item 4.) and several indices of those constraints can be added in a single iteration (see Algorithm 8, lines 5 to 9).

Algorithm 8: Determination of the Step Lenght in the Elastic Approach

```
   // Determine blocking constraint; if no blocking constraint exists,
      the index remains empty and the step length is set to 1.
 1 Choose $i_f \in (\mathcal{R}^f \cup \mathcal{B}) \setminus \mathcal{W}_k$ with minimal $\alpha_f \leq 1$ using (4.72), cf. item 3.
 2 Choose $i_s \in (\mathcal{S} \cup \mathcal{T}) \setminus \mathcal{W}_k$ with $\sigma(i_s) \in \mathcal{W}_k$ implying minimal step lenght
   $\alpha_s = \min(-s_{i_s}^{(k)}/p_{s_{i_s}}^{(k)}, -t_{i_s}^{(k)}/p_{t_{i_s}}^{(k)}) \leq 1$, cf. item 4.
 3 Check if a blocking constraint $i_r \in \mathcal{R}^r \setminus \mathcal{W}_k$ exists using (4.73). If it exists set
   $\alpha_r = 0$, cf. item 6b.
 4 Choose an index $i \in \{i_f, i_s, i_r\}$ corresponding to the smalles allowed step lenght
   $\alpha_k = \min(\alpha_f, \alpha_s, \alpha_r)$.
   // Fix independent variable bounds for slacks.
 5 Add all indices $i_t \in \mathcal{T}$ with $\sigma(i_t) \notin \mathcal{W}_k$ and $a_{\sigma(i_t)}^T(x^{(k)} + \alpha_k p_x^{(k)}) \in [r_{l,\sigma(i_t)}, r_{u,\sigma(i_t)}]$
   to $\mathcal{W}_k$, cf. item 6a.
 6 if aggressive-mode then
 7     Add all indices $i_e \in \mathcal{S}$ with $a_{\sigma(i_e)}^T(x^{(k)} + \alpha_k p_x^{(k)}) - b_{\sigma(i_e)} = 0$ as well as every
       $i_t \in \mathcal{T}$ with $\sigma(i_t) \in \mathcal{W}_k$ and $a_{\sigma(i_t)}^T(x^{(k)} + \alpha_k p_x^{(k)}) \in [r_{l,\sigma(i_t)}, r_{u,\sigma(i_t)}]$ to $\mathcal{W}_k$, cf.
       item 6a.
 8 else if $\mathcal{B} \cap \mathcal{W}_k = \emptyset$ and $\mathcal{R} \cap \mathcal{W}_k = \emptyset$ then
 9     Add all indices $i_e \in \mathcal{S}$ with $a_{\sigma(i_e)}^T(x^{(k)} + \alpha_k p_x^{(k)}) - b_{\sigma(i_e)} = 0$ to $\mathcal{W}_k$, cf. item 6a.
10 return step lenght $\alpha_k$ and blocking index $i$.
```

Case 6b especially takes place, whenever $p_x^{(k)}$ vanishes but $p_{t,i}^{(k)} = t_i^{(k)} \neq 0$ for any $i \in \mathcal{R}^r \setminus \mathcal{W}_k$. In this case i is a blocking constraint and is added to the workingset. Further more, to avoid numerical difficulties, $t_i^{(k+1)} = 0$ is explicitly set and the step information $p_{t,i}^{(k)}$ is dropped whenever $\sigma(i)$ is added to the workingset.

The next paragraph explains how slack variables can be used not only for constraints which are infeasible at a supplied initial value $\bar{x}$ but for the satisfaction of a supplied initial workingset $\mathcal{W}_0$ also.

4.2.6 Starting Point Techniques and Warm Start

The presented elastic approach is capable of handling infeasiblility in equality and range constraints by relaxation while feasibility to variable bounds is allways supposed to hold. If no starting point $x^{(0)}$ is given the algorithm starts in the origin of $\mathbb{R}^n$. At a first stage the bound feasibility is ensured, in order that

$$x^{(0)} \ni \mathcal{F}_\mathcal{B} = \{x \in \mathbb{R}^n \colon b_{l,i} \leq x_i \leq b_{u,i}, i \in \mathcal{B}\}.$$

Based on a bound feasible iterate $x^{(0)}$, slack variables for injured equality and range constraints are set such that $(x^{(0)}, s^{(0)}, t^{(0)})$ is feasible for the relaxed QP (4.23).

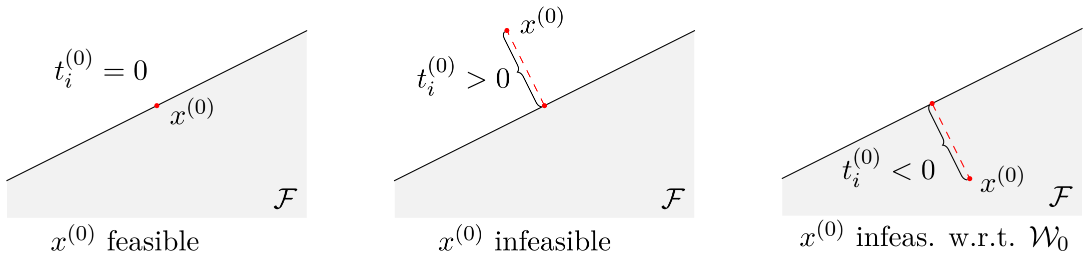


Figure 4.2: Initialization of slack variables.

If an initial estimation of the optimal active-set is given (warm start), slack variables are not only used to obtain a feasible starting point but also to make it feasible w.r.t. the workingset $\mathcal{W}_0$. For $i \in \mathcal{W}_0 \cap \mathcal{R}$ the relaxation chooses $t_i \neq 0$, such that

$$a_i^T x^{(0)} + t_i - b_i = 0. \tag{4.74}$$

Subsummed the initialization strategy follows two key points:

1. Ensure bound feasibility in the supplied starting point $x^{(0)}$.
2. Determine $s^{(0)}, t^{(0)}$ such that $(x^{(0)}, s^{(0)}, t^{(0)})$ is a feasible starting point to the relaxed QP and (4.74) holds for all $i \in \mathcal{W}_0 \cap \mathcal{R}$.

Figure 4.2 shows the three possible cases: a feasible starting point where no relaxation is needed, i.e. $t_i = 0$ (left), an infeasible point $x^{(0)} \notin \mathcal{F}$ yielding $t_i > 0$ (middle) and an initial estimate $x^{(0)} \in \mathcal{F}$ which is infeasible w.r.t. $\mathcal{W}_0$ yielding $t_i < 0$ (right).[2] Slack variables corresponding to the latter case have to be dropped whenever the constraint is removed from the workingset. This is achieved by setting $t_i = 0$ and $\mathcal{W}_k \cup \{i\}$.

Depending on the starting point it may lead to better results in some instances, if all relaxed constraints are added to the workingset during the initialization of the algorithm. This is the feasible-mode in the implementation to this thesis. The computed steps then inherit more information about the feasible set and the iterates tend to become feasible faster. However, the optimization progress may be slowed down in other instances or starting points. Finally, all relaxed constraints have to be part of the initial working set whenever the linear penalty approach is used, see Section 4.2.2. The initialization strategy is stated in Algorithm 9.

4.2.7 Heuristics and Algorithmic details

This section describes a couple of enhancements of the presented active-set method. These enhancements are designed to improve the algorithm for difficult instances, indicate degeneracy and optimize the memory consumption.

[2]The stated signums of t_i hold for injured lower limits, they change in the case of injured upper limits.

Algorithm 9: Initialization Strategy in the Elastic Approach

```
Input: Initial estimate $x^{(0)}$ (default: $0 \in \mathbb{R}^n$) and workingset $\mathcal{W}_0$ (default: $\mathcal{E}$).
1 Initialize $s^{(0)}$ such that $a_i^T x^{(0)} + s_i^{(0)} - b_i = 0$ holds for all $i \in \mathcal{E}$.
2 if feasible-mode then
3     Add all constraints $i \in \mathcal{R}^r$ to $\mathcal{W}_0$.
4 Initialize $t^{(0)}$ such that $a_i^T x^{(0)} + t_i^{(0)} \in [r_{l,i}, r_{u,i}]$ holds for all $i \in \mathcal{R}$ and (4.74) holds
  for all $i \in \mathcal{W}_0 \cap \mathcal{R}$.
```

Convexification Strategy

When insufficient information about the active-set is given the presented algorithm can not guarantee that the projected Hessian is positive definite. The result is that the projected KKT system (4.70) can not be solved. A remedy for nonconvex Hessians is to add a positive multiple of the idetity matrix yielding $\bar{H} + \nu_c I$, such that the linear system can be solved. In this case a slightly different but strictly convex QP is solved but it makes it possible to find a descent direction as long as the workingset does not differ too much from the optimal active-set. Steps $p_x^2 \in \mathbb{R}^{|\mathcal{B}\setminus\mathcal{W}_k|}$ are then determined by the solution of the equality constrained subproblems

$$\min_{p_x, p_{\tilde{s}}} \quad \frac{1}{2} p_x^T (\bar{H} + \nu_c I) p_x + (\bar{g}^{(k)} + \nu_c x^{(k)})^T p_x + \rho\phi(\tilde{s}^{(k)} + p_{\tilde{s}}) \tag{4.75a}$$

$$\text{s.t.} \quad a_i^T p_x = 0, \quad i \in \mathcal{M}_k^f, \tag{4.75b}$$

$$a_i^T p_x + p_{\tilde{s}} = 0, \quad i \in \mathcal{M}_k^r \tag{4.75c}$$

with augmented slack variables $\tilde{s}$ corresponding to active constraints. The projected KKT system to problem (4.75) (after $p_{\tilde{s}}$ is eliminated) differs from (4.70). The first block row reads

$$(\bar{H} + \nu_c I) p_x - \sum_{i \in \mathcal{M}_k^f} a_i \lambda_i^f - \sum_{i \in \mathcal{M}_k^r} a_i \lambda_i^r = -\bar{g}^{(k)} - \nu_c x^{(k)}. \tag{4.76}$$

It is obvious that an optimal solution of the convexified problem yielding $p_x = 0$ does not coinside with a solution of the original one. This is due to the disturbance of the dual multipliers in the magnitude of $\nu_c x^{(k)}$.

The presented algorithm switches to the solution of a convexified QP and drives it to optimality whenever it is needed. When the convexified solution is found, it returns to the solution of the orginal problem and reduces the maximal allowed convexification by a fixed reduction parameter $\kappa_c > 0$. A suitable convexification parameter can be suggested either by the user before solving the problem or by the subsolver used for the KKT solution within the solution process.

An access to avoid indefinite projected Hessians is used in *inertia-controlling QP methods*

(ICQP) which can be applied to the relaxed QP. Such methods as presented by Gill et al. [42] use the workingset to control the inertia of the projected Hessian, which is never allowed to have more than one nonpositive eigenvalue. For example, see a null-space method for ASM using inertia-control and updates of the search direction by Gómez [49].

Incrementation of the Penalty Parameter

The presented approach determines a feasible solution, i.e. $\|(s^{(k)}, t^{(k)})\| < \kappa_{\text{tol}}$, whenever it exists and the penalization is chosen large enough. But a too large penalty parameter may infect the quality of the step vectors obtained by the solution of the KKT system (4.70). To avoid numerical difficulties the penalty parameter is chosen to be small in the beginning and increased later on. In the implementation to this thesis the initial penalty parameter is chosen to be

$$\rho = \max\left(\kappa_{\rho_1}, \sqrt{\|(s^{(0)}, t^{(0)})\|}, \|\nabla_x q(x^{(0)})\|, q(x^{(0)})\right) \tag{4.77}$$

with a minimal penalization $\kappa_{\rho_1} \geq 0$. The penalty parameter is enlarged whenever the algorithm discovers a stationary point w.r.t. the current relaxed objective function. It is enlarged by a constant factor $\kappa_{\rho_2} > 1$, yielding

$$\rho \leftarrow \kappa_{\rho_2} \cdot \rho. \tag{4.78}$$

QP Termination Criterion

The elastic approach terminates whenever the first-order optimality for the unrelaxed QP holds and the primal infeasibility is smaller than a given threshold, i.e.

$$\max(\nabla_x \mathcal{L}^{(k)}, \|s^{(k)}\|, \|t^{(k)}\|) < \kappa_{\text{tol}}, \quad \kappa_{\text{tol}} > 0. \tag{4.79}$$

This speeds up the algorithm since not all indices $i \in \mathcal{S} \cup \mathcal{T}$ need to be added to the workingset. Even if the determination of the optimal active-set $\mathcal{A}^r(x^*)$ for the relaxed QP is incomplete, it is not for the one of the original problem. It holds

$$\mathcal{A}(x^*) = \mathcal{A}^r(x^*) \setminus (\mathcal{S} \cup \mathcal{T}). \tag{4.80}$$

The algorithm is also terminated when neighter the objective function nor the primal infeasibility is reduced over a fixed number $\kappa_{\text{np}} \geq 1$ of iterations. In this case it is either affected by stalling or cycling. Whenever $\nabla_x \mathcal{L}^{(k)} < \kappa_{\text{tol}}$ but (4.79) does not hold, i.e. $s^{(k)}, t^{(k)} > \kappa_{\text{tol}}$, and the penalization parameter reached its upper limit $\kappa_{\max}$, the algorithm terminates and returns an infeasible solution. Additionally, the number of iterations and the overall solution time may be limited.

Size	Initialization	Reduction	KKT Solution	Expansion 1
n	$x^{(k)} \longrightarrow x^{(k)}$	$* \longrightarrow *$	$* \longrightarrow *$	$* \longrightarrow *$
m	$s^{(k)} \longrightarrow s^{(k)}$	$* \longrightarrow *$	$* \longrightarrow *$	$* \longrightarrow *$
k	$t^{(k)} \longrightarrow t^{(k)}$	$* \longrightarrow *$	$* \longrightarrow *$	$* \longrightarrow *$
n	$* \longrightarrow g^{(k)}$	$* \longrightarrow *$	$* \longrightarrow *$	$* \longrightarrow *$
n	$g^{(k)} \longrightarrow g^{(k)}$	$g^{(k)} \longrightarrow X_1$	$X_1 \longrightarrow Y_1$	$Y_1 \longrightarrow p_x^{(k)}$
m	$* \longrightarrow s^{(k)}$	$s^{(k)} \longrightarrow X_2$	$X_2 \longrightarrow Y_2$	$* \longrightarrow *$
k	$* \longrightarrow t^{(k)}$	$t^{(k)} \longrightarrow X_3$	$X_3 \longrightarrow Y_3$	$* \longrightarrow *$
k	$* \longrightarrow *$	$* \longrightarrow *$	$* \longrightarrow *$	$* \longrightarrow *$

Table 4.2: Vector management: initialization, reduction and expansion of the primal search direction (part 1). Memory blocks marked with X_i, Y_i, $i = 1, 2, 3$, are coherent and are used for the projected KKT right-hand side.

Size	Expansion 2.a	Step computation
n	$* \longrightarrow *$	$x^{(k)} \longrightarrow x^{(k)} + \alpha_k p_x^{(k)}$
m	$* \longrightarrow *$	$s^{(k)} \longrightarrow s^{(k)} + \alpha_k p_s^{(k)}$
k	$* \longrightarrow *$	$t^{(k)} \longrightarrow t^{(k)} + \alpha_k p_t^{(k)}$
n	$* \longrightarrow *$	$* \longrightarrow *$
n	$p_x^{(k)} \longrightarrow p_x^{(k)}$	$* \longrightarrow *$
m	$* \longrightarrow p_s^{(k)}$	$* \longrightarrow *$
k	$* \longrightarrow p_t^{(k)}$	$* \longrightarrow *$
k	$* \longrightarrow *$	$* \longrightarrow A_{\mathcal{R}}^{(k)} x^{(k)} + t^{(k)}$

Table 4.3: Vector management: expansion of the primal search direction for slack variables (part 2.a) for nonzero steps and memory usage in step computation.

Vector Management

The complete iteration data, including iterates, search directions, KKT right-hand side vectors, etc., is stored in a vector $y \in \mathbb{R}^{3n+2m+3k}$. Since the elastic approach relies on the basic active-set method presented in Section 4.1, dual multipliers are only computed if the solution of (4.70) gives $p_x^2 = 0$. If $p_x^2 \neq 0$ is encountered, the dual part of the KKT system needs not be solved. Instead, the stepdata for slack variables is determined without using dual multipliers by

$$p_{s,i}^{(k)} = \begin{cases} -a_i^T p_x^{(k)}, & i \in \mathcal{E}^r, \\ 0, & i \in \mathcal{E}^f \end{cases} \quad \text{and} \quad p_{t,i}^{(k)} = \begin{cases} -a_i^T p_x^{(k)}, & i \in \mathcal{R}^r \cap \mathcal{W}_k, \\ -t_i^{(k)}, & i \in \mathcal{R}^r \setminus \mathcal{W}_k, \\ 0, & i \in \mathcal{R}^f. \end{cases} \tag{4.81}$$

Memory delegated to range duals is then used as workspace in the step computation. The memory management in each iteration including the initialization, reduction and solution of the primal part of the KKT system is visualized in Table 4.2. Tables 4.3 and 4.4 show

Size	Expansion 2.b			Expansion 3		
n	$*$	$\longrightarrow$	$*$	$x^{(k)}$	$\longrightarrow$	$x^{(k)}$
m	$*$	$\longrightarrow$	$*$	$s^{(k)}$	$\longrightarrow$	$s^{(k)}$
k	$*$	$\longrightarrow$	$*$	$t^{(k)}$	$\longrightarrow$	$t^{(k)}$
n	$g^{(k)}$	$\longrightarrow$	$u_l - u_u$	$u_l - u_u$	$\longrightarrow$	u_l
n	$p_x^{(k)}$	$\longrightarrow$	$p_x^{(k)}$	$*$	$\longrightarrow$	u_u
m	$*$	$\longrightarrow$	z	z	$\longrightarrow$	z
k	$*$	$\longrightarrow$	$v_l - v_u$	$v_l - v_u$	$\longrightarrow$	v_l
k	$*$	$\longrightarrow$	$*$	$*$	$\longrightarrow$	v_u

Table 4.4: Vector management: expansion of primal step data (part 2.b) for zero steps and determination of dual multipliers by expansion (part 3).

the memory usage depending on the determined step vector p_x. Memory blocks which are starred out are either not changed ($* \longrightarrow *$) or the stored input data is unused and overwritten (e.g. $* \longrightarrow g^{(k)}$).

4.2.8 Example: Behavior of the Elastic Approach

This section demonstrates the behavior of the presented approach. Consider the problem

$$\min_{x \in \mathbb{R}^2} \quad \tfrac{1}{2}\, x^T x \tag{4.82a}$$

$$\text{s.t.} \quad x_1 + x_2 \geq \tfrac{3}{2}, \tag{4.82b}$$

$$x_1 \in [1, 3], \tag{4.82c}$$

$$x_2 \geq 0. \tag{4.82d}$$

The classification of the constraints is as follows. By construction it is $\mathcal{B} = \{1, 2\}$. Only (4.82d) is interpreted as a variable bound, i.e. $b_{l,1} = -\infty$ and $b_{u,1} = b_{u,2} = +\infty$. The problem does not contain equality constraints. This gives $\mathcal{E} = \emptyset$. Equations (4.82b) and (4.82c) are interpreted as range constraints, yielding $\mathcal{R} = \{3, 4\}$ and $\sigma(\mathcal{R}) = \mathcal{T} \subseteq \{5, 6\}$ whenever constraints are relaxed.

The optimal solution of (4.82) is given by $x^* = (1, \frac{1}{2})^T$ with dual multipliers $v_l = (\frac{1}{2}, \frac{1}{2})^T$ and $v_u = 0 \in \mathbb{R}^2$, $u_l = u_u = 0 \in \mathbb{R}^2$. The optimal active-set is given by $\mathcal{A}(x^*) = \{3, 4\}$ with active lower limits. It is obvious that the origin of $\mathbb{R}^2$ is the minimizer of the unconstrained optimization problem.

The QP above is solved starting at four different starting points but with the same information about the active-set, namely $\mathcal{W}_0 = \emptyset$. The initial estimates are chosen by the following means:

1. Demonstrating the equivalence of the elastic approach to the basic active-set method when the starting point is feasible, i.e. no relaxation is needed.

	run 1 (red)		run 2 (orange)		run 3 (green)		run 4 (blue)	
k	$x^{(k)}$	$\mathcal{W}_k$	$x^{(k)}$	$\mathcal{W}_k$	$x^{(k)}$	$\mathcal{W}_k$	$x^{(k)}$	$\mathcal{W}_k$
0	$(\frac{3}{2}, \frac{3}{2})$	$\{\ \}$	$(4, 2)$	$\{\ \}$	$(\frac{1}{2}, \frac{3}{2})$	$\{\ \}$	$(-1, \frac{1}{2})$	$\{\ \}$
1	$(1, 1)$	$\{3\}$	$(1, \frac{1}{2})$	$\{3, 5\}$	$(\frac{1}{2}, \frac{3}{2})$	$\{3\}$	$(0, 0)$	$\{\ \}$
2	$(1, \frac{1}{2})$	$\{3, 4\}$	$(1, \frac{1}{2})$	$\{3, 4, 5\}$	$(\frac{3}{4}, \frac{3}{4})$	$\{3, 4\}$	$(0, 0)$	$\{4\}$
3					$(1, \frac{1}{2})$	$\{3, 4\}$	$(\frac{3}{4}, \frac{3}{4})$	$\{4\}$
4							$(\frac{3}{4}, \frac{3}{4})$	$\{3, 4\}$
5							$(1, \frac{1}{2})$	$\{3, 4\}$

Table 4.5: Iteration data of the solution progress to the solution of (4.82) using $\mathcal{W}_0 = \emptyset$.

2. Multiple changes in the workingset when an inactive relaxed constraint with a violated upper limit becomes active at the lower limit and vice versa. This corresponds to item 6a in Section 4.2.5. See also the introducing issue discussed at the beginning of the same section.

3. The starting point is located outside the feasible set and at least once the computed step needs to be rejected to ensure the reduction of the primal infeasibility. Violated constraints then need to be introduced into the workingset to push the iterates into the feasible set. This corresponds to item 6b in Section 4.2.5.

4. The initial estimate is located near the unconstrained minimizer outside the feasible set. Changes in the workingset enforce the algorithm to leave the stationary point to obtain feasible iterates. This again corresponds to item 6b in Section 4.2.5.

The algorithm is run using the quadratic relaxation scheme ($\rho_0 = 10^8$) in the default-mode, i.e. neither the feasible-mode is used in the initialization nor the aggressive-mode is used during the optimization. It is terminated as soon as the termination criterion (cf. Section 4.2.7) holds for $\kappa_{\text{tol}} \approx 1.5 \cdot 10^{-8}$. The iterative progress to the solution of (4.82) for the stated choices of the starting point $x^{(0)}$ using $\mathcal{W}_0 = \emptyset$ is summed up in Table 4.5. Figure 4.3 visualizes the movement of the iterates during the solution progress. The path of the iterates to case 1 is plotted with a red, case 2 in an orange, case 3 in a green and case 4 in a blue dashed line. The curvature of the objective function is outlined in red dotted circles. Constraint gradients and the feasible set are sketched in light grey.

The computed step in the first iteration only depends on the objective gradient. Since the algorithm is cold started it seeks for the unconstrained minimizer. Depending on the choice of the starting point, it is

$$Hp_x^{(0)} = -g^{(0)} \qquad \Longleftrightarrow \qquad p_x^{(0)} = -x^{(0)}. \tag{4.83}$$

Starting at $x^{(0)} = (\frac{3}{2}, \frac{3}{2})^T$ the step is blocked by $3 \in \mathcal{R}^f$ yielding $\alpha_0 = \frac{1}{3}$ and $x^{(1)} = (1, 1)^T$. $p_x^{(1)} = (0, -1)^T$ is afterwards blocked by $4 \in \mathcal{R}^f$, with $\alpha_1 = \frac{1}{2}$, resulting in the optimal solution $x^{(2)} = (1, \frac{1}{2})^T$ with $\mathcal{W}_2 = \{3, 4\}$.

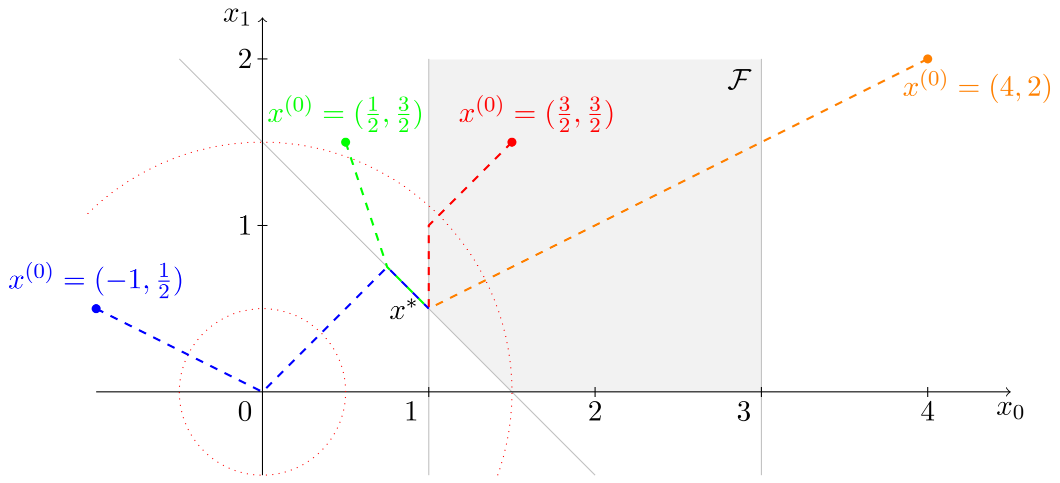

x_1
2
1
0
1
2
3
4
x_0
$\mathcal{F}$
$x^{(0)} = (\frac{1}{2}, \frac{3}{2})$
$x^{(0)} = (\frac{3}{2}, \frac{3}{2})$
$x^{(0)} = (4, 2)$
$x^{(0)} = (-1, \frac{1}{2})$
x^*

Figure 4.3: Movement of the iterates using $\mathcal{W}_0 = \emptyset$.

Observing case 2 above with $x^{(0)} = (4, 2)^T$ constraint $3 \in \mathcal{R}$ is relaxed, i.e. $3 \in \mathcal{R}^r$ and $t_3 = -1$. Since the upper limit is violated at the starting point every step lenght $\alpha_0 \in [\frac{1}{4}, \frac{3}{4}]$ yields a feasible iterate. $p_x^{(0)} = (-4, -2)^T$ is blocked by the lower limit of $3 \in \mathcal{R}^r$, which gives $\alpha_0 = \frac{3}{4}$. The next iterate is then feasible and the slack variable t_3 is set to zero and $\sigma(3) = 5 \in \mathcal{T}$ is simultaneously added to the workingset. Iteration 2 is then blocked by $4 \in \mathcal{R}^f$ and the algorithm identifies the optimal solution and active-set.

When the algorithm is run starting at $x^{(0)} = (\frac{1}{2}, \frac{3}{2})^T$ the lower limit of $3 \in \mathcal{R}^r$ is violated and it is $t_3 = \frac{1}{2}$. In the first iteration every nonzero step towards the origin of $\mathbb{R}^2$ would increase the infeasibility of $3 \in \mathcal{R}^r$. This is easy to see, since $p_{t,3}^{(0)} = \frac{1}{2} > 0$. It is $\alpha_0 = 0$ and the blocking index is included in $\mathcal{W}_1 = \{3\}$. The upcoming step in iteration 2 is blocked by $4 \in \mathcal{R}^f$ at $x^{(2)} = (1, \frac{1}{2})^T$ with $t_3 = 0$. The termination criterion (4.79) is already satisfied at this point, such that $5 \in \mathcal{T}$ is not included in $\mathcal{W}_2$. But the optimal active-set is determined by the means of (4.80).

Starting at $x^{(0)} = (-1, \frac{1}{2})^T$ implies $t_3 = 2$ and $t_4 = 1$ as well as $3, 4 \in \mathcal{R}^r$. Applying the Newton step $p_x^{(0)} = -x^{(0)}$ reduces the infeasibility in both relaxed constraints and minimizes the unconstrained objective function leading to $x^{(1)} = (0, 0)^T$. The determination of $p_x^{(1)}$ follows (4.83) with $-g^{(1)} = 0 \in \mathbb{R}^2$ giving $p_x^{(1)} = 0 \in \mathbb{R}^2$. Criterion (4.79) is not satisfied at $(x^{(1)}, t^{(1)}) = ((0, 0)^T, (1, 1.5)^T)$ and therefore constraint $4 \in \mathcal{R}^r$ (due to the most absolut violation) is added to $\mathcal{W}_1 = \{\ \}$. In the following iteration $x^{(3)} = (\frac{3}{4}, \frac{3}{4})^T$ is obtained which is the constrained minimizer w.r.t. $\mathcal{W}_3 = \{4\}$, i.e. $p_x^{(4)}$ vanishes. Since $3 \notin \mathcal{W}_3$ and $t_3 > 0$ the constraint is included in $\mathcal{W}_4 = \{3, 4\}$ to enforce feasibility which is achieved in $x^{(5)} = (1, \frac{1}{2})^T$.

Warm Start

When it comes to warm starts, the presented relaxation is also used to preserve prior knowledge of the optimal active-set. As stated in Section 4.2.6 the initialization of slack

	run 1 (red)		run 2 (orange)		run 3 (green)		run 4 (blue)	
k	$x^{(k)}$	$\mathcal{W}_k$	$x^{(k)}$	$\mathcal{W}_k$	$x^{(k)}$	$\mathcal{W}_k$	$x^{(k)}$	$\mathcal{W}_k$
0	$(\frac{3}{2},\frac{3}{2})$	$\{3,4\}$	$(4,2)$	$\{3,4\}$	$(\frac{1}{2},\frac{3}{2})$	$\{3,4\}$	$(-1,\frac{1}{2})$	$\{3,4\}$
1	$(1,\frac{1}{2})$	$\{3,4,5,6\}$	$(1,\frac{1}{2})$	$\{3,4,5,6\}$	$(1,\frac{1}{2})$	$\{3,4,6\}$	$(1,\frac{1}{2})$	$\{3,4\}$

Table 4.6: Iteration data of the solution progress to the solution of (4.82) using $\mathcal{W}_0 = \mathcal{A}(x^*)$.

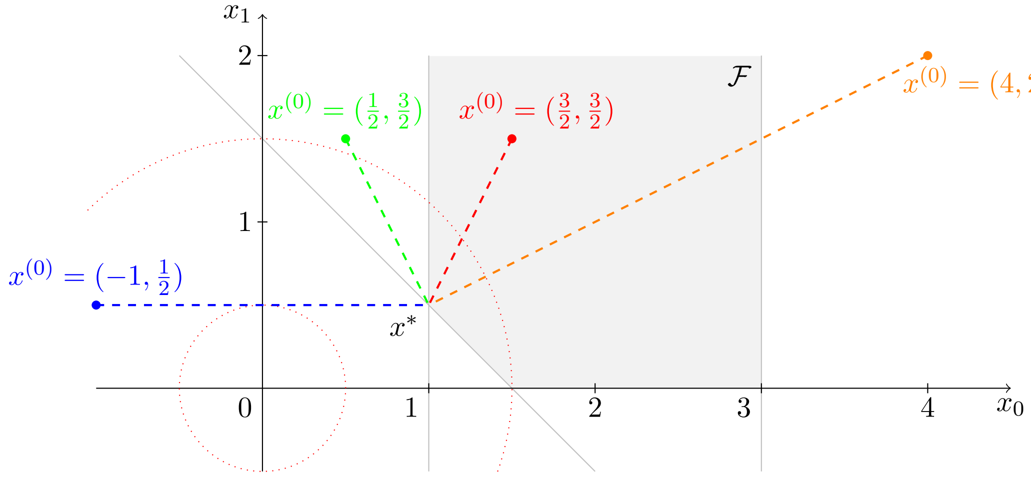


Figure 4.4: Movement of the iterates using $\mathcal{W}_0 = \{3,4\}$.

variables is dedicated to the initial workingset: slack variables do not only ensure the feasibility of equality and range constraints, they also compensate infeasibility w.r.t. $\mathcal{W}_0$.

The solution of problem (4.82) starting at any initial estimate but with $\mathcal{W}_0 = \mathcal{A}(x^*)$ should be achieved in a single iteration. This is demonstrated for the starting points stated in the preceeding paragraph as a proof of concept.

Considering case 1 above no relaxation would be needed in the first place since $x^{(0)} \in \mathcal{F}$ but the supplied initial estimate would violate the supplied workingset information since constraints (4.82b) and (4.82c) are supposed to be active. In case 2, $t_3^{(0)}$ does not only soak up the infeasibility corresponding to the upper limit of (4.82c) which would be achieved by defining $t_3^{(0)} = -1$. Instead $t_3^{(0)} = -3$ is chosen to ensure $a_3^T x^{(0)} + t_3^{(0)} - r_{l,3} = 0$.

The iteration data of the solution processes for the stated initial starting points is summed up in Table 4.6. The movement of the iterates is visualized in Figure 4.4.

Interpretation of the Final Workingset

It is easy to observe that the optimal active-set $\mathcal{A}^r(x^*) = \{3,4,5,6\} = \mathcal{R} \cup \mathcal{T}$ for the relaxed QP is not completely determined in most of the cases above. But the correlation (4.80) always holds yielding the correct optimal active-set $\mathcal{A}(x^*)$ for QP (4.14).

4.2.9 The Complete Elastic Active-Set Method

By now, the complete elastic active-set method can be stated (see Algorithm 10).

If supported by the KKT solution algorithm the solution of the reduced system may be splitted up into the computation of the primal step p_x^2 in line 10 and the determination of the dual multipliers only if $p_x^2 \neq 0$ in line 18. The returned solution in line 25 may be infeasible, i.e. $\|(s^{(k)}, t^{(k)})\| > \kappa_{\mathrm{tol}}$, whenever the penalty parameter ρ reached its maximum by the incrementation in line 21. Using exact arithmetic, an infeasible solution identifies either that the QP does not have a feasible solution or the inconsistency it. But numerically, it can also be an indicator for badly scaled problem data. If the KKT system can not be solved without convexification of the Hessian, a convexified solution is returned in line 25 whenever the suggested convexification parameter is not too large. The upper limit of the allowed convexification is successively reduced in line 23. A convexified solution delivers a good estimation of the optimal active-set, but the final iterate and objective value differ from the original ones.

The bottleneck in time and memory consumption of Algorithm 10 is the construction and solution of the linear system $K^{(k)}y^{(k)} = r^{(k)}$. On the one hand, it is reasonable to employ efficient projection techniques and avoid uneccessary (re-)evaluations of the problem data. On the other hand, the solution should include partial solution techniques and updating strategies for existing factorizations whenever direct solvers are used.

Algorithm 10: Complete Elastic Primal Active-Set Method for QP

Input : User provided initial estimate $x^{(0)}$ and a suitable workingset $\mathcal{W}_0$ (optional), vector of algorithmic constants κ.

```
   // Initialization
1  if $x^{(0)} \notin \mathcal{F}_\mathcal{B}$ then
2      return unchanged $x^{(0)}$ and $\mathcal{W}_0$ and report failure.
3  Evaluate constraints at $x^{(0)}$.
4  Call Algorithm 9 to initialize slack vectors $s^{(0)}$, $t^{(0)}$ and adjust $\mathcal{W}_0$, cf. Section 4.2.6.
5  Initialize the penalty parameter $\rho$, cf. (4.77), and convexification parameter $\nu_c$.
   // Iteration loop
6  Set iteration counter $k = 0$.
7  while the QP termination criterion does not hold do
8      Evaluate the objective gradient $g^{(k)} = Hx^{(k)} + f$.
9      Set up the KKT system (4.70) w.r.t. $\nu_c$.
10     Call KKT solution algorithm; request primal solution to obtain $p_x^2$.
11     if convexification required then
12         Adjust $\nu_c$ and update the KKT right-hand side vector.
13         Call KKT solution algorithm; request primal solution to obtain $p_x^2$.
14     Lift $p_x^2$ up onto the full space, cf. Expansion 1 in (4.71).
15     Compute $p_s^{(k)}, p_t^{(k)}$ using (4.81), cf. Expansion 2.a in Section 4.2.3.
16     if $\|(p_x^{(k)}, p_s^{(k)}, p_t^{(k)})\| < \kappa_{step}$ then                /* zero step */
17         Call KKT solution algorithm; request dual solution to obtain $\lambda^f, \lambda^r$.
18         Determine dual multipliers $v_l, v_u, u_l, u_u$, cf. Expansion 2.b in Section 4.2.3.
19         if $v_{l,i} - v_{u,i} \geq 0$ and $u_{l,i} - u_{u,i} \geq 0$ for all $i \in \mathcal{W}_k$ then
20             if $\|(s^{(k)}, t^{(k)})\| \not< \kappa_{tol}$ and $\rho < \rho_{\max}$ then    /* inf. solution */
21                 Increase penalty parameter $\rho \leftarrow \kappa_{\rho_2} \cdot \rho$.
22             if $\nu_c > 0$ then                                /* conv. solution */
23                 Reduce max convexification $\kappa_{c_2} \leftarrow \kappa_{c_1} \cdot \kappa_{c_2}$ and set $\nu_c = 0$.
24             else                                               /* solution found */
25                 return solution $x^{(k)}$ and $\mathcal{W}_k$.
26         else
27             Identify $j = \arg\min_j (\min_{j \in \mathcal{W}_k \cap \mathcal{B}}(u_{l,j} - u_{u,j}), \min_{j \in \mathcal{W}_k \cap \mathcal{R}}(v_{l,j} - v_{u,j}))$.
28             Set $(x^{(k+1)}, s^{(k+1)}, t^{(k+1)}) \leftarrow (x^{(k)}, s^{(k)}, t^{(k)})$ and $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$.
29     else                                                       /* nonzero step */
30         Call Algorithm 8 to obtain step lenght $\alpha_k$ and blocking constraint $i$.
31         Set $(x^{(k+1)}, s^{(k+1)}, t^{(k+1)}) \leftarrow (x^{(k)}, s^{(k)}, t^{(k)}) + \alpha_k(p_x^{(k)}, p_s^{(k)}, p_t^{(k)})$.
32         if blocking constraints exist then
33             Set $\mathcal{W}_{k+1} = \mathcal{W}_k \cup \{i\}$.
34     Increase iteration counter $k \leftarrow k + 1$.
```

Chapter 5

KKT Solution Algorithms

This chapter concentrates on the solution of KKT systems arising as subproblems in optimization algorithms. They share a widespread superstructure and can be stated as follows:

$$Ky = r \qquad \Longleftrightarrow \qquad \begin{bmatrix} H & (A^f)^T & (A^r)^T \\ A^f & & \\ A^r & & -M^{-1} \end{bmatrix} \begin{pmatrix} p \\ -\lambda^f \\ -\lambda^r \end{pmatrix} = \begin{pmatrix} -g \\ c^f \\ c^r \end{pmatrix} \tag{5.1}$$

with $K \in \mathbb{R}^{N\times N}$ for $N = n+m+k$. Futhermore, $H \in \mathbb{R}^{n\times n}$ is symmetric and $A^f \in \mathbb{R}^{m\times n}$, $A^r \in \mathbb{R}^{k\times n}$. The lower right block is diagonal of size $k \times k$, i.e. $-M^{-1} \in \mathbb{R}^{k\times k}$. The vectors $y, r \in \mathbb{R}^N$ are partitioned according to the block structure of the KKT matrix.

Linear systems of this type are solved, e.g. in the step determination of active-set and interior-point methods. In the latter approach the system is of a fixed size, where in the former approach the size changes depending on the workingset, see Section 4.2.4. Efficient solution algorithms should not only exploit the superstructure of the system, but should also do so for substructures in the nonzero matrix blocks.

The chapter is organized as follows. Section 5.1 describes a strategy to tackle nonconvex Hessians, which for example arise in the early iterations of cold started ASM. A structure exploiting direct solution algorithm for block dense matrices adapting this strategy is the topic of Section 5.2. The focus of Section 5.3 is on the exploitation of substructure which is understood as an extension of the procedure presented for the dense case. Finally, an ASM specific update scheme for factorizations of the KKT system is presented in Section 5.4.

5.1 Convexification Strategy

One difficulty in solving (5.1) is that the upper left block may not be positive definite on the null-space of $A = [(A^f)^T, (A^r)^T]^T$. Since it has usually to be ensured that p exists - and is a descent direction in line search methods, a *convexification strategy* can be incorporated. Modifying (5.1) by adding a nonnegative multiple of the identity matrix $\nu_c I \in \mathbb{R}^{n\times n}$ to the upper left block yields the *convexified KKT system* $K_{\nu_c} y = r$,

$$\begin{bmatrix} H+\nu_c I & (A^f)^T & (A^r)^T \\ A^f & & \\ A^r & & -M^{-1} \end{bmatrix} \begin{pmatrix} p \\ -\lambda^f \\ -\lambda^r \end{pmatrix} = \begin{pmatrix} -g \\ c^f \\ c^r \end{pmatrix}. \tag{5.2}$$

It can be shown that if the *convexification parameter* $\nu_c \geq 0$ is chosen large enough the convexified block $H + \nu_c I$ projected onto the null-space of A is positive definite.

5.2 The Dense Case: Null-Space Method

This section presents a structure exploiting direct solution algorithm using an adaptive convexification strategy for the linear system (5.2). The so called *null-space method* is tailored for dense blocks of the (convexified) KKT matrix.

The solution starts with the determination of a LQ factorization

$$A^f = LQ = \begin{bmatrix} L_1 & 0 \end{bmatrix} \begin{bmatrix} Y^T \\ Z^T \end{bmatrix} = L_1 Y^T. \tag{5.3}$$

Here, $L \in \mathbb{R}^{m\times n}$, $Q \in \mathbb{R}^{n\times n}$, $L_1 \in \mathbb{R}^{m\times m}$, $Y \in \mathbb{R}^{n\times m}$, $Z \in \mathbb{R}^{n\times(n-m)}$ holds and the zero block in L is of size $m\times(n-m)$. Thereby, Z denotes an orthonormal basis of the null-space $\ker(A^f)$ of A^f and Y is chosen such that $[Y\ Z]$ is an orthonormal basis or $\mathbb{R}^n$.

Equation (5.3) yields a decomposition of the primal vector p, given by

$$p = Yp_Y + Zp_Z, \quad p_Y \in \mathbb{R}^m, \quad p_Y \in \mathbb{R}^{n-m}. \tag{5.4}$$

Substitution of p in the second block row of (5.2) and usage of the previously determined LQ factorization (5.3) yields

$$A^f p = A^f(Yp_Y + Zp_Z) = A^f Y p_Y = L_1 p_Y = c^f. \tag{5.5}$$

Upon this, $-\lambda^r$ is eliminated using the third block row of (5.2), equation (5.4) and $p_Y = 0$;

$$-\lambda^r = M(c^r - A^r(Yp_Y + Zp_Z)) \tag{5.6a}$$
$$= M(c^r - A^r Y p_Y) - MA^r Z p_Z. \tag{5.6b}$$

Substitution of this in the first block row of (5.2) and left-multiplication using the transposed null-space basis matrix Z^T results in the symmetric system

$$Z^T(H + \nu_c I + (A^r)^T M A^r) Z p_Z = -Z^T(g + (A^r)^T M(c^r - A^r Y p_Y)) \tag{5.7}$$

since $ZA^f = 0$. Re-writing this using the *(convexified) reduced Hessian*

$$\hat{H}_{\nu_c} = Z^T(H + \nu_c I + (A^r)^T M A^r)Z \in \mathbb{R}^{(n-m)\times(n-m)} \tag{5.8}$$

and the modified right-hand side

$$\hat{g} = Z^T(g + (A^r)^T M(c^r - A^r Y p_Y)) \in \mathbb{R}^{(n-m)} \tag{5.9}$$

yields the reduced symmetric linear system

$$\hat{H}_{\nu_c} p_Z = -\hat{g}. \tag{5.10}$$

(5.10) is solved by using a Cholesky factorization. If $\hat{H}_{\nu_c}$ is not positive definite for $\nu_c = 0$ the convexification parameter is successively enlarged and the involved components $\hat{H}_{\nu_c}$ and $\hat{g}$ are re-computed until the factorization succeeds. After p_Z is obtained by the solution of (5.10), the primal part of the KKT solution is computed using (5.4).

If the dual part of the solution is required $-\lambda^r$ is determined by (5.6). Left-multiplication of the first block row of (5.2) with Y^T gives

$$Y^T(A^f)^T(-\lambda^f) = -Y^T\left(g + (H + \nu_c I)p + (A^r)^T(-\lambda^r)\right). \tag{5.11}$$

The latter equation can be solved using the initially calculated LQ factorization, by

$$L_1^T(-\lambda^f) = -\bar{g}, \quad \bar{g} = Y^T\left(g + (H + \nu_c I)p + (A^r)^T(-\lambda^r)\right). \tag{5.12}$$

The presented solution algorithm for block dense KKT matrices is stated in Algorithm 11.

5.3 The Sparse Case

The null-space method presented in the preceeding section only exploits the superstructure of (5.2) but does not take any sub-structure of the matrix blocks into account. This may be problematic, for instance in real-world applications where the problem size often enforces the exploitation of structure on a very detailed level because standard solution techniques are impractical. So, whenever the user has a good knowlege of the problem data and the sparsity it is recommended to design a specialized solver.

For a good example the interested reader is referred to Steinbach [83] who developed a structure exploiting recursive solution algorithm for multistage KKT systems with an underliying chain structure arising in nonlinear optimal control problems. See also the thesis of Huebner [63] who has implemented distributed algorithms for the solution of KKT systems arising in interior-point methods for tree-structured nonlinear programs.

Algorithm 11: Null-Space Method for Block Dense KKT Systems

```
Input: KKT matrix $K$, right-hand side $r$ and solution request.
1  if $K$ is factorized then
2      Go to primal solution in line 14.
   // Factorization
3  Compute $LQ$ factorization $A^f = LQ$, see (5.3).
4  if $L_1$ has zero columns then
5      LICQ does not hold; return $r$ and report error.
6  Initialize convexification parameter $\nu_c = 0$ and counter $k_c = 0$.
7  repeat
8      Compute the (convexified) reduced Hessian $\hat{H}_{\nu_c}$, see (5.8).
9      Try to compute the Cholesky factorization $L_{\hat{H}_{\nu_c}} L^T_{\hat{H}_{\nu_c}}$.
10     On failure update $\nu_c$ and set $k_c \leftarrow k_c + 1$.
11 until Cholesky factorization was successful.
12 if $k_c > 0$ then
13     return Convexification counter $k_c$ and parameter $\nu_c$.
   // Solution of the primal part
14 if primal solution requested then
15     Solve $L_1 p_Y = c^f$
16     Compute modified right-hand side $\hat{g} = Z^T(g + (A^r)^T M(c^r - A^r Y p_Y))$.
17     Solve $\hat{H}_{\nu_c} p_Z = -\hat{g}$ and compute $p = Y p_Y + Z p_Z$.
   // Solution of the dual part
18 if dual solution requested then
19     Compute $-\lambda^r = M(c^r - A^r Y p_Y) - M A^r Z p_Z$.
20     Compute $\bar{g} = Y^T \left(g + (H + \nu_c I)p + (A^r)^T(-\lambda^r)\right)$ and solve $L_1^T(-\lambda^f) = -\bar{g}$.
21 return solution $y = (p, -\lambda^f, -\lambda^r)$.
```

5.4 Generic Factorization Updates in Active-Set Methods

Consider the linear system (5.2) appearing as the KKT system to solve within the l-th iteration of the elastic active-set method presented in Chapter 4. Assume that a valid factorization of it is known and the workingset changes in the transition from $\mathcal{W}_l$ to $\mathcal{W}_{l+1}$. Then, the KKT sizes change, i.e.

$$n^{(l)} + m^{(l)} + k^{(l)} = N^{(l)} \neq N^{(l+1)} = n^{(l+1)} + m^{(l+1)} + k^{(l+1)} \tag{5.13}$$

holds, where the sizes are given by $n^{(j)} = n - |\mathcal{B} \cap \mathcal{W}_j|$ and $m^{(j)} = |\mathcal{E}^f| + |\mathcal{R}^f \cap \mathcal{W}_j|$ and $k^{(j)} = |\mathcal{E}^r| + |\mathcal{R}^r \cap \mathcal{W}_j|$ for $j \in \{l, l+1\}$. In other words, the subblocks change, whenever

- constraints are added to the workingset,
- constraints are removed from the workingset or
- relaxed active constraints become feasible and remain in the workingset.

In detail, adding $i \in \mathcal{B}$ to $\mathcal{W}_l$ decreases the size of the Hessian, the upper left block in (5.2). Including $i \in \mathcal{R}$ in $\mathcal{W}_l$ enlarges either $m^{(l)}$ or $k^{(l)}$, dropping it reduces the size. And when a relaxed active constraint is identified to be feasible and remains in the workingset $k^{(l)}$ is reduced while $m^{(l)}$ is enlarged.

It is well-known that Cholesky factorizations for dense matrices can be recovered after modifications as rank one updates $K = K + \gamma y y^T$, $\gamma \in \mathbb{R}$ in $\mathcal{O}(N^2)$. Furthermore, they can be maintained after appending a row and column or removing the last ones. The latter is also true for QR factorizations. Appending rows and columns also works for $P^T LU$ factorizations. See Gill et al. [36] and the references therein for an overwiev of relevant algorithms. Established software which should be mentioned is LUMOD [81], a Fortran code for updating dense LU factors when rows and columns are added, deleted or replaced and the LUSOL [40] software package which implements a sparse LU decomposition for square and rectangular matrices. The latter one is the basis factorization package of SQOPT and SNOPT.

The fill-in produced by common update techniques is dense in general thus any sparsity in the factors gets lost. In the following paragraphs a generic update technique for factorizations using the *Schur complement method* is discussed. The main focus relies in the preservation of sparsity. It is achieved by re-using existing factors without modifications by an additional back-solve and the solution of a smaller dense symmetric linear system.

Appending Rows and Columns to Valid Factorzations

Assume, that a sparse factorization of a structured KKT matrix $K \in \mathbb{R}^{N \times N}$ for the solution of $Ky = r$ exists and let $a \in \mathbb{R}^N$ be appended to it as a row and column. The linear system then reads

$$K^+ y^+ = r^+ \quad \Longleftrightarrow \quad \begin{bmatrix} K & a \\ a^T & 0 \end{bmatrix} \begin{pmatrix} y_1^+ \\ -\lambda_1^+ \end{pmatrix} = \begin{pmatrix} r_1^+ \\ 0 \end{pmatrix} \tag{5.14}$$

where $K^+ \in \mathbb{R}^{(N+1)\times(N+1)}$ $y^+, r^+ \in \mathbb{R}^{(N+1)}$ and $y_1^+, r_1^+ \in \mathbb{R}^N$, $-\lambda_1^+ \in \mathbb{R}$. (5.14) is solved using the *Schur complement* $S^+ = -a^T K^{-1} a \in \mathbb{R}$. The inverse of K^+ is determined by

$$(K^+)^{-1} = \begin{bmatrix} 1 & -K^{-1}a \\ & 1 \end{bmatrix} \begin{bmatrix} K^{-1} & \\ & (S^+)^{-1} \end{bmatrix} \begin{bmatrix} 1 & \\ -(K^{-1}a)^T & 1 \end{bmatrix}. \tag{5.15}$$

Applied to the KKT right-hand side $(r_1^+, 0) \in \mathbb{R}^{N+1}$ this gives

$$y_1^+ = K^{-1}\left(r_1^+ - a(-\lambda_1^+)\right) \quad \text{with} \quad -\lambda_1^+ = \frac{a^T K^{-1} r_1^+}{a^T K^{-1} a}. \tag{5.16}$$

The solution of the enlarged system (5.14) is obtained by solving the linear systems

$$K\gamma_r^+ = r_1^+ \quad \text{and} \quad K\gamma_a^+ = a \tag{5.17}$$

of size $N \times N$ using the existing factorization of K. Upon this, the corresponding terms in (5.16) are substituted, yielding

$$-\lambda_1^+ = \frac{a^T \gamma_r^+}{a^T \gamma_a^+} \quad \text{and} \quad y_1^+ = \gamma_r^+ - \gamma_a^+(-\lambda_1^+). \tag{5.18}$$

In summary, the solution requires one additional back-solve with the existing factorization instead of computing a new factorization of $K^+ \in \mathbb{R}^{(N+1)\times(N+1)}$.

The proposed method is easily extended to a set of linearly independent vectors a_j, $j = 1, \dots, s$. Re-using the notation above and appending

$$A^+ := [a_1 \dots a_s] \in \mathbb{R}^{N\times s},$$

to the factorized matrix K results in the solution of an enlarged linear system of size $(N+s)\times(N+s)$, reading

$$K^+ y^+ = r^+ \quad \Longleftrightarrow \quad \begin{bmatrix} K & A^+ \\ (A^+)^T & 0 \end{bmatrix} \begin{pmatrix} y_s^+ \\ -\lambda_s^+ \end{pmatrix} = \begin{pmatrix} r_s^+ \\ 0 \end{pmatrix} \tag{5.19}$$

with $y_s^+ \in \mathbb{R}^N$ and $\lambda_s^+ \in \mathbb{R}^s$. In analogy, the Schur complement is defined by $S^+ = -(A^+)^T K^{-1} A^+ \in \mathbb{R}^{s\times s}$ giving

$$(K^+)^{-1} = \begin{bmatrix} I_N & -K^{-1}A^+ \\ & I_s \end{bmatrix} \begin{bmatrix} K^{-1} & \\ & (S^+)^{-1} \end{bmatrix} \begin{bmatrix} I_N & \\ -(K^{-1}A^+)^T & I_s \end{bmatrix}. \tag{5.20}$$

The resulting equations to solve for y_s^+ and λ_s^+ then read

$$y_s^+ = K^{-1}\left(r_s^+ - A^+(-\lambda_s^+)\right) \quad \text{and} \quad \left((A^+)^T K^{-1} A^+\right)(-\lambda_s^+) = (A^+)^T K^{-1} r_s^+. \tag{5.21}$$

The computation of the latter vector is prepared by the solution of the linear systems

$$K\gamma_r^+ = r_s^+ \quad \text{and} \quad K\gamma_{a,j}^+ = a_j, \quad j = 1, \dots, s, \quad \gamma_{a,j}^+ \in \mathbb{R}^N \tag{5.22}$$

and is completed by solving the dense symmetric system

$$\Gamma_s(-\lambda_s^+) = (A^+)^T \gamma_r^+ \quad \text{with} \quad \Gamma_s = (A^+)^T \left[\gamma_{a,1}^+ \dots \gamma_{a,s}^+\right] \in \mathbb{R}^{s \times s}. \tag{5.23}$$

As mentioned in the introduction direct factorizations that can be updated find their application at this point, when the columns of A^+ are appended one after the other while r_s^+ changes after every appendage. Vector $y_s^+ \in \mathbb{R}^N$ is finally determined by

$$y_s^+ = \gamma_r^+ - \sum_{j=1}^{s} (-\lambda_{s,j}^+)\gamma_{a,j}^+. \tag{5.24}$$

In total, the solution of (5.19) requires $s+1$ back-solves using the existing factorization of K, one for the upper part with right-hand side r_s^+ and one per appended vector a_j, $j = 1, \dots, s$, and the solution of the dense symmetric system (5.23) of size $s \times s$.

The lower right block of K^+ and the lower part of r^+ in (5.19) do not necessarily need be zero. For $-M^+ \in \mathbb{R}^{s\times s}$ and $c_s^+ \in \mathbb{R}^s$ the system changes to

$$\begin{bmatrix} K & A^+ \\ (A^+)^T & -M^+ \end{bmatrix} \begin{pmatrix} y_s^+ \\ -\lambda_s^+ \end{pmatrix} = \begin{pmatrix} r_s^+ \\ c_s^+ \end{pmatrix}. \tag{5.25}$$

The only modification of the solution algorithm relies in the changed Schur complement $S^+ = -(A^+)^T K^{-1} A^+ - M^+ \in \mathbb{R}^{s\times s}$ leading to the modified version of (5.23), reading

$$\left(\Gamma_s + M^+\right)(-\lambda_s^+) = (A^+)^T \gamma_r^+ - c_s^+. \tag{5.26}$$

Removing Rows and Columns from Valid Factorzations

If a set of vectors $\{a_j\}$ is successively appended and intermediate solutions $\gamma_{a,j}^+$ are stored after each appendage, once appended vectors can easily be removed. The removal is achieved by skipping the corresponding elements in the construction of Γ_s and the computation of the extended solution, see (5.24). Against that, removing a row and column of K itself enforces a refactorization. But a positiv side effect is that the rows and columns of A^+ which will not be dropped again can be comprised in K^+ such that the solution of (5.23) becomes cheaper in later solutions.

Special Handling of Workingset Changes for Variable Bounds

The presented update scheme is independent of the detailed sparsity of the KKT matrix and speeds up the solution algorithm as long as system (5.23) is small. But, appending single rows and columns requires direct data access to the corresponding vectors which can not be guaranteed – imagine data free or sparse problem implementations which underliy a special data structure.

Since only changes in the status of simple variable bounds are completely structure independent, the implementation to this thesis only incorporates factorization updates in this case. The appended or removed vectors are then given by $e_j = (0, \dots, 0, 1, 0, \dots, 0) \in \mathbb{R}^N$ where only the j-th component is nonzero with $1 \leq j \leq n$.

The Generic Factorization Update Algorithm

Algorithm 12 states the presented generic Schur complement update. It can be included in Algorithm 10 at line 10, before the sub-algorithm for the KKT solution is called.

Algorithm 12: Generic KKT Factorization Update for Active-Set Methods

```
Input: Index $j$ and optional vector $a \neq 0 \in \mathbb{R}^N$.
1  Initialize empty set $\mathcal{SC} \leftarrow \{\ \}$.
   // Update factorization
2  if $a \neq 0$ then                                              /* append vector */
3      Solve $K\gamma^+_{a,j} = a$; store $\gamma^+_{a,j}$ and set $\mathcal{SC} \leftarrow \mathcal{SC} \cup \{j\}$.
4  else                                                             /* drop vector */
5      if $j \in \mathcal{SC}$ then
6          Drop index $j$ by setting $\mathcal{SC} \leftarrow \mathcal{SC} \setminus \{j\}$.
7      else
8          Re-compute $K$ and factorize $K$ without row/column $j$ and set $\mathcal{SC} \leftarrow \{\ \}$.
   // Solve
9  Obtain $\gamma^+_r$ by solving $K\gamma^+_r = r^+_{\mathcal{SC}}$. Update $\Gamma_{\mathcal{SC}}$ and solve $\Gamma_{\mathcal{SC}}(-\lambda^+_{\mathcal{SC}}) = (A^+)^T\gamma^+_r$.
10 Determine $y^+_{\mathcal{SC}} = \gamma^+_r - \sum_{i\in\mathcal{SC}}(-\lambda^+_{\mathcal{SC},i})\gamma^+_{a,i}$ and return $(y^+_{\mathcal{SC}}, -\lambda^+_{\mathcal{SC}})$.
```

Chapter 6

Software Design

In mathematical optimization many different algorithmic approaches are considered for tackling an enormous amount of different problem classes. But not every algorithm works well on every class of problem. Especially when it comes to real-world applications, efficient numerical solution schemes require specialized software fitted to the application and data structure at hand. Therefore, algorithms have been reimplemented over and over again by just changing few algorithmic strategies or the supported data structure they operate on.

For example, consider the step computation in a Filter Line-Search SQP, where a QP subproblem has to be solved, a feasibility restoration phase in the case of failure or even the complete line-search technique has to be chosen - all with respect to the overall robustness and efficiency. The main algorithm, i.e. the SQP framework, only requires the propper solution of the self-contained subproblems, which is independent of the chosen sub-solvers in detail. Even more, the usage of special data structures requires the independence of the data structure in use for all incorporated (sub-)algorithms.

The idea that comes along is to stop the need of reimplementations by giving flexible skeletons of the basic algorithms for numerical optimization and delegate the solution of self-contained subproblems to sub-solvers encapsulated in exchangable building blocks. In the view of a main algorithm those building blocks only have to suit a fixed interface to make it independent from the precise implementations. If these aspects hold, the way for the realization of a highly flexible and maintainable software framework is paved.

According to the authors colleague Martin Schmidt[1],the goal of the software framework developed for this thesis is to make a reimplementation of the presented algorithmic frameworks unnecessary. This is achieved by using the generic implementation properties of C++. Its current version C++14 offers a wide range of techniques allowing the implementation of efficient and generic optimization frameworks satisfying the conceptional ideas stated above. It comprises the presented Filter Line-Search SQP method in Chapter 3 referred to as Clean::SQP in the following and the elastic primal active-set method for QP in Chapter 4 referred to as Clean::ASM. Both are part of the generic software library Clean, which is an acronym for *A C++ Library for Efficient Algorithms in Numerics*, which is developed in the

[1]JProf. Dr. Martin Schmidt, `mar.schmidt@fau.de`, Department Mathematik, Friedrich-Alexander-Universität Erlangen-Nürnberg

working group *Algorithmic Optimization* of Marc C. Steinbach[2]at the Leibniz Universität Hannover. Clean is not yet sufficiently mature but it is intended to become public domain as soon as it is considered to be [82, 63]. The application of most of the discussed software concepts in the context of numerics goes back to Marc C. Steinbach and Clean.

The Chapter is organized as follows. Section 6.1 summarizes general design concepts from the field of software engineering and reviews some modern C++ design aspects, which are extensively used in the implementation to this thesis. Based on this, the software architecture of Clean is presented in Section 6.2 with special attention to the architcetures of Clean::SQP in Sect. 6.2.1 and Clean::ASM in Sect. 6.2.2.

6.1 General Design Concepts

In order to fulfill the introductory stated characteristics, a well designed software framework has to satisfy some of the most important concepts from the field of software engeneering: *orthogonality*, *cohesion* and *coupling.*

Orthogonality names the independence of modules of a software. According to [64], an orthogonal software design satisfies the property that their modules are as independent as possible. This is achieved, when the functionality or implementation details of a module can change without harming the functionality of others.

Cohesion in software engeneering is the "degree to which the elements of a module belong together", cf. [103]. High cohesion proves the maintainability of modules and enhances the readability of the code. The user does not need to know how implemented modules work in detail. This makes it easier to exchange a certain one.

Coupling describes the degree of independence between software modules. It is a measure of how closely two modules are connected, cf. [103]. Low coupling correlates with high cohesion and viece versa. In the scope of orthogonality, modules need to be loosely coupled to be independent.

In summary, a well structured software framework of good design shows low coupling, and when it is combined with high cohesion it supports the general goals of high readability and maintainability. Subsequent generic programming techniques in C++ are described which are extensively used in Clean.

Generic Programming Techniques

The realization of the concepts of software design stated above in Clean::ASM and Clean::SQP is achieved by using techniques of generic programming with C++ templates. Most classes

[2]Prof. Dr. Marc C. Steinbach, `steinbach@ifam.uni-hannover.de`, Institute of Applied Mathematics, Leibniz Universität Hannover

are implemented following a *policy-based class design* introduced by Alexandrescu [1], which can be seen as a compile-time variant of the *strategy pattern* (see [34]).

The fundamental idea of policy-based class design is to design classes that take several template parameters as input which are instantiated in dependence on types given by the user that specify the behavior of the class. They use *type traits* - a programming technique allowing compile-time decisions based on types instead of runtime decisions based on values - to define all the types required in its scope. Type traits are used extensively in some generic libraries, e.g. the *C++ Standard Template Library* (STL) [90] and Boost [91]. An example of the policy-traits design is shown in Listing 6.1.

Listing 6.1: Example for the policy-traits design.

```
template<class Example_Policy>
struct Example_Traits
{
  using Dummy = Policy::T_Dummy;
};

template<class Example_Policy>
class Example
{
  using Traits = Example_Traits<Example_Policy>;
  using Dummy  = typename Traits::Dummy;
};
```

Another frequently used technique to be mentioned is the one of *tag dispatching*. This is a way of using function overloading to dispatch based on properties of a type [7] which is often used hand in hand with traits classes. For example see Listing 6.2.

Listing 6.2: Example for tag dispatching.

```
struct Foo {};
struct Bar {};
struct Use_Foo { using Tag = Foo; };
struct Use_Bar { using Tag = Bar; };

template<class T>
void action(T& t) { action(t, typename T::Tag()); }

template<class T>
void action(T& t, Foo tag) { foo_action(); }

template<class T>
void action(T& t, Bar tag) { bar_action(); }
```

To avoid code duplication, basic functionalities shared by various classes are encapsulated in *base classes* such that inheriting from a base class provides the derived class with its functionalities. Base classes may not be parameterized by a policy but by the traits of the inheriting class.

Some of the base classes employ the *Curiously Recurring Template Pattern* (CRTP) [18]. The design combines templates and inheritance in such a way that the template hierarchy is directed opposite to the inheritance hierarchy. Thus, a class inherits a CRTP base class and passes itself in complete instantiation as a template parameter. Methods within the base class can therefore use the supplied template parameter to access members of derived classes, but unlike standard inheritance CRTP base classes are not allowed to hold any data members. An abstract example of the CRTP design is shown in Listing 6.3.

Listing 6.3: Example for the CRTP design.

```
template<class T>
class CRTP_Base
{
  // ...
};

template<class Policy>
class Derived
  : public CRTP_Base<Derived<Policy>>
{
  // ...
};
```

6.2 The Software Architecture of Clean

Clean is a generic C++ library whose conceptional idea relies in the provision of numerical algorithms that are not dependent on specific data types. The user may be able to employ any problem-tailored data type (e.g. sparse matrices) and implement suitable (sub-)algorithms which work on the data structure in use. Plugging in new implementations is required to not affect the numerical logic of an algorithm, thus the provided ones need not to be reimplemented.

C++ and External Libraries

As the long form of its name already tells, the software library Clean is written in C++. At the time of publication, it mostly satisfies the C++11 standard but also employs some features of the current C++14 standard [92]. It requires the C++ standard library and the Boost C++ libraries (version 1.59.0 or higher) [91]. Linear algebra operations on dense data (matrices and vectors) are based on BLAS [67] and LAPACK [2]. Those for sparse matrices in *triplet sparse* (TS) format are either self-implemented (e.g. matrix-vector products) or use routines from the HSL Mathematical Software Library [61, 62]. The code documentation is done using Doxygen [94].

Clean Algorithms

The strict independence between *main algorithms*, their *sub-algorithms* and *data structures* requires a strict separation of responsibilities. Algorithms implement the basic numerical logic of itselves. They employ *data objects* of the given data structure on which mathematical operations are executed using fixed interfaces. These objects represent mathematical objects like a vector or matrix and provide certain mathematical operations, e.g. matrix-vector pruducts. Required operations are delegated to the algorithms data objects using *servers*, which provide the interface to the data objects and ensure that they harmonize with each other.

In detail, the main components of a numerical algorithm in Clean are distinguished into the following groups.

Main Algorithms encapsulate the basic algorithmic logic of itselves. As, for example, see the basic frameworks for SQP (Algorithm 1) and ASM (Algorithm 6).

Sub-Algorithms are the exchangeable building-blocks of a main algorithm which are responsible for the solution of self-contained subproblems. Thus, every self-contained task of a superordinate algorithm, like the determination of a stepsize or the solution of a KKT system, is done by a sub-algorithm. Since there are a lot of possibilities in which way a subproblem can be solved the user can either use an existing sub-algorithm or implement a specialized one. In the latter case the new implementation only has to suit the interface between the sub-algorithm and the superordinate algorithm and their server.

Servers are the heart of every algorithm. A server collects all relevant data structures and delegates the operations on these data structures which are requested by the main algorithm. It is also possible that a superordinate algorithm and its sub-algorithms share the same server if they operate on the same data, e.g. vector data like iterates and search directions.

Data Structures mainly comprise vectors for the iterates (x, λ) and model evaluation data, like the objective gradient and constraint values and problem matrices $H, A_{\mathcal{E}}, A_{\mathcal{R}}$. Except the iterates and stepdata, the data structures in SQP are encapsulated in the definition of the QP subproblem. In ASM it is the KKT right-hand side vector, the projected problem matrices as well as diagonal blocks for the convexification of the projected Hessian $(\nu_c I)$ and the relaxtion block $(-M)$. The user can employ existing data structures[3] or implement new ones. For example, Huebner [63] employed a highly efficient parallel tree-sparse KKT solver and distributed data structures to the code Clean::IPM by Schmidt [82] in his thesis.

[3] At the time of publication Clean offers implementations using dense matrices and matrices in TS format. KKT matrix implementations fulfill the 3×3-block superstructure. KKT vectors are available fitted to the block structure of KKT matrices, i.e. comprising variable-, equality- and range-vector blocks.

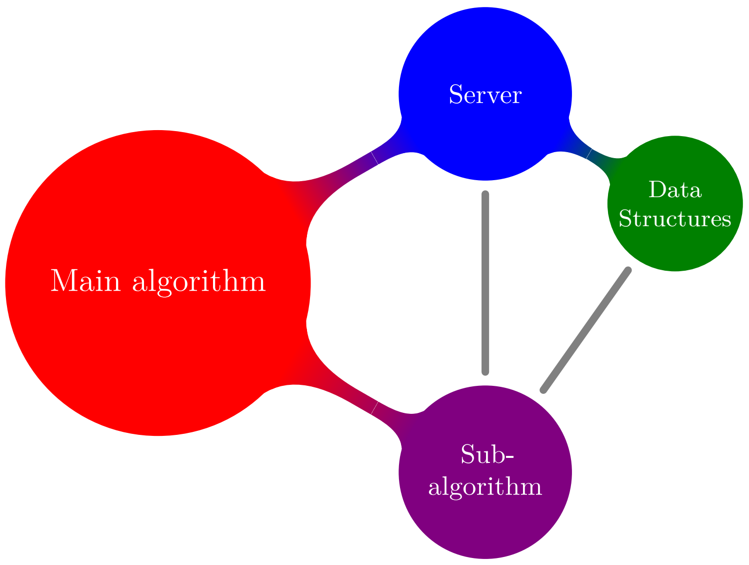


Figure 6.1: A schematic overview of the software architecture of algorithms of Clean.

> According to [82], the aspect of strong but desired coupling has to be mentioned. The data structures are not determined by the problem itself. But, they are appointed by the sub-algorithm that solves the subproblem in the step determination, i.e. the QP solver in SQP and the KKT solver in active-set methods. For this reason, this sub-algorithm determines the data structures on which it operates.

The described components and their correlation are illustrated in Figure 6.1. Recapitulating the concept of coupling, most of the nodes are not directly connected representing loose coupling of the components. This also visualizes the design aspect of orthogonality in the code. Special aspects introducing stronger coupling to the design, like the dependencies between sub-solvers and the data structures they operate on, are represented by the grey connections.

6.2.1 The Architecture of Clean::SQP

The code Clean::SQP comes along with independent building blocks implementing the filter line-search SQP framework presented in Chapter 3. It preserves the NLP sparse structure in a native way since the algorithm only requires the evaluation of the problem data (e.g. objective function, constraitns, etc.) at certain points.

If the user implements a new problem-tailored data structure he has to provide the correct filling of the local QP in the step determination. The QP data is held by the QP server which has to suit the data structure of the employed QP solver which defines the data structure.

Figure 6.2 illustrates the described components and their relationships.

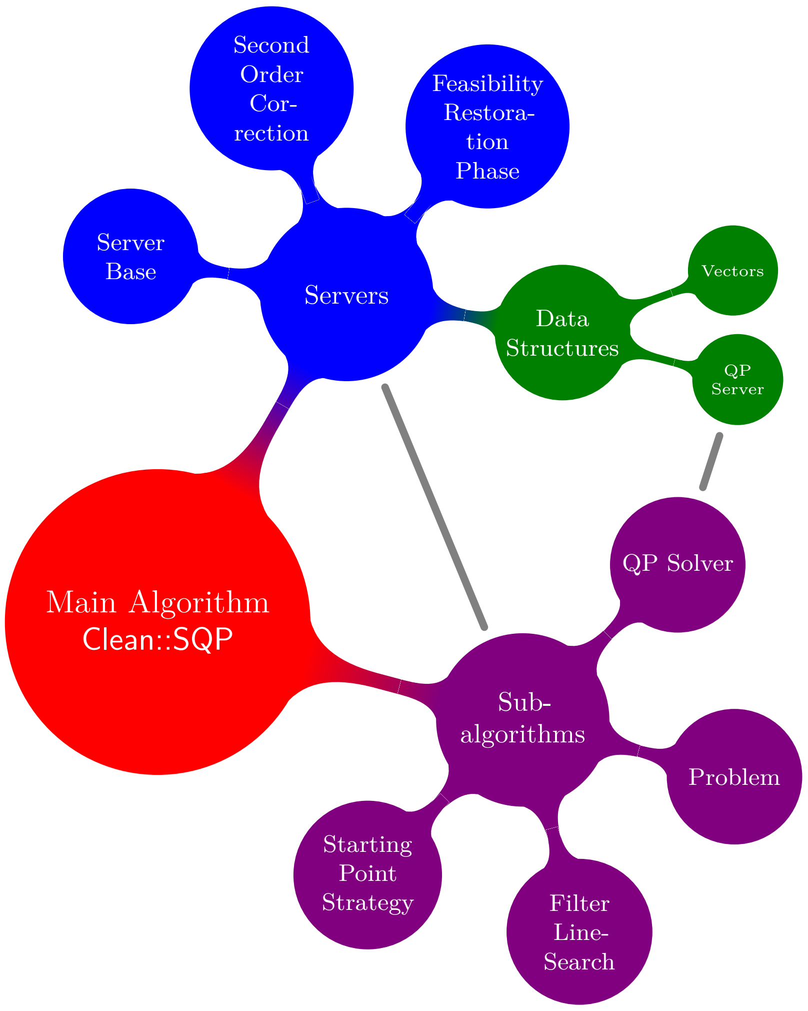


Figure 6.2: A schematic overview of the software architecture of Clean::SQP.

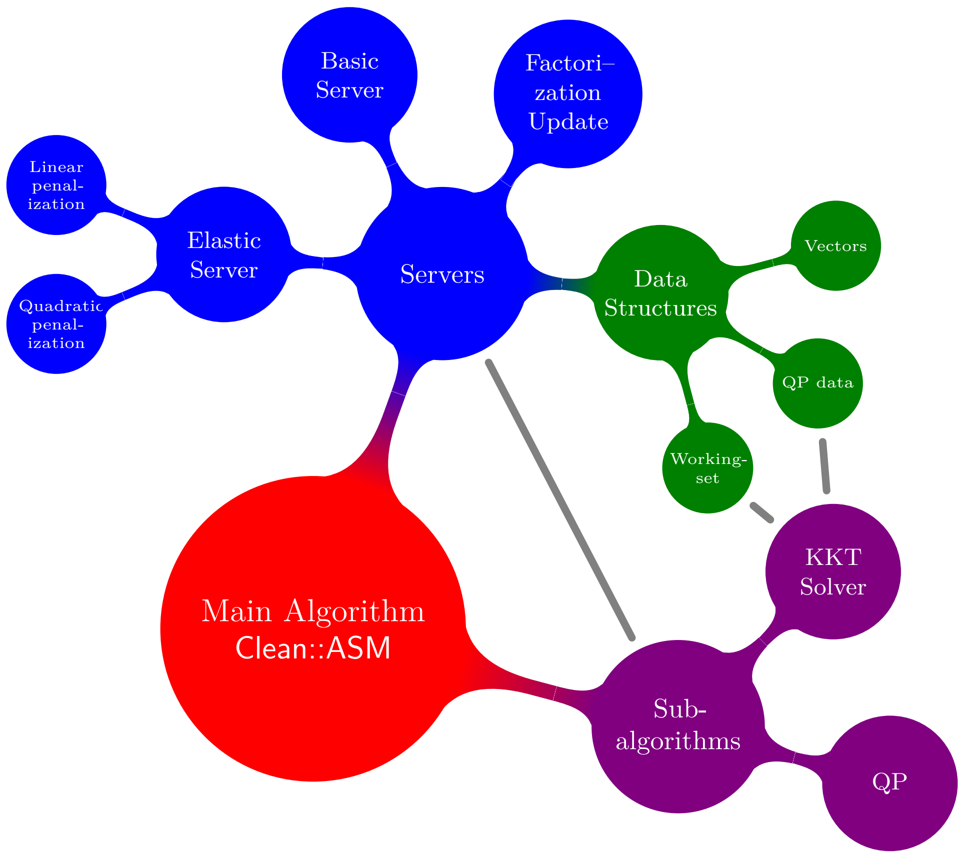


Figure 6.3: A schematic overview of the software architecture of Clean::ASM

6.2.2 The Architecture of Clean::ASM

The code Clean::ASM implements the elastic primal active-set approach presented in Section 4.2. It comes along with a set of three generic servers: a basic one for feasible problems and one for each presented relaxation scheme. The generic update scheme for factorizations is also provided.

Troughout the complete implementation no direct element or row access to the QP matrix data is needed. Data driven operations only require linear mappings using the block structure of the KKT matrix. For the solution of the KKT system any sufficiently accurate KKT solver can be employed which defines the data structure. Solvers for the dense case, implementing the direct solution method presented in Section 5.2, and the sparse case using matrices in triplet-sparse format are available. For the latter one an interface to the sparse solvers MA27/57 [61, 62] of the HSL library is provided. For any self implemented data structure the user must provide a projector for the KKT system. This building block is used to set up the reduced KKT system out of the structured QP w.r.t. the current workingset.

Figure 6.3 illustrates the correlation of the components of the presented algorithm in Chapter 4.

Chapter 7

Computational Results

This chapter presents computational results obtained with Clean::ASM and Clean::SQP on a variety of test problems. Its main goal is to document the generality of the software framework. First, Section 7.1 examines the load capacity of Clean::ASM by solving a set of established convex quadratic programming examples. The relaxation schemes of the elastic QP solver are compared and the effort gained by using the generic procedure for updating KKT factors is presented. The applicability of Clean::SQP and Clean::ASM is evaluated by the solution of a real-life application from the field of mathematical biology in Section 7.2. Finally, Section 7.3 presents results on a multistage optimization problem for dynamic processes. Specialized data structures and software for the solution of the KKT systems are in use which proves the orthogonality of the code and the capability of the framework to adapt non-standard techniques.

Computational results were obtained using a Fujitsu Primergy RX4770 M2 with an Intel Xeon e7-8867 v3 CPU @ 2.50 GHz (4 socket x 16 cores x 2 smt) and 2 TB RAM. The operating system is CentOS 7.3 (x86_64). The software was compiled with g++ (GCC) and GNU Fortran (GCC) in version 6.3.0. If not stated otherwise, matrices are stored in TS format and KKT systems are solved using MA57 Version 3.7.0 from the HSL Mathematical Software Library. All CPU times are in seconds and do not include the time required to load the problem data.

7.1 Results for Convex Quadratic Programming

This section evaluates the ability of Clean::ASM to solve convex QPs of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & q(x) = \tfrac{1}{2}x^T H x + f^T x && && \text{(7.1a)}\\ \text{s.t.} \quad & a_i^T x = b_i, && i \in \mathcal{E}, && \text{(7.1b)}\\ & a_i^T x \in [r_{l,i}, r_{u,i}], && i \in \mathcal{R}, && \text{(7.1c)}\\ & x_i \in [b_{l,i}, b_{u,i}], && i \in \mathcal{B}. && \text{(7.1d)} \end{aligned}$$

The problems arise from the Maros and Mészáros convex QP test set [73, 72]. It is a collection of 138 convex quadratic programming examples from a variety of sources: Subset

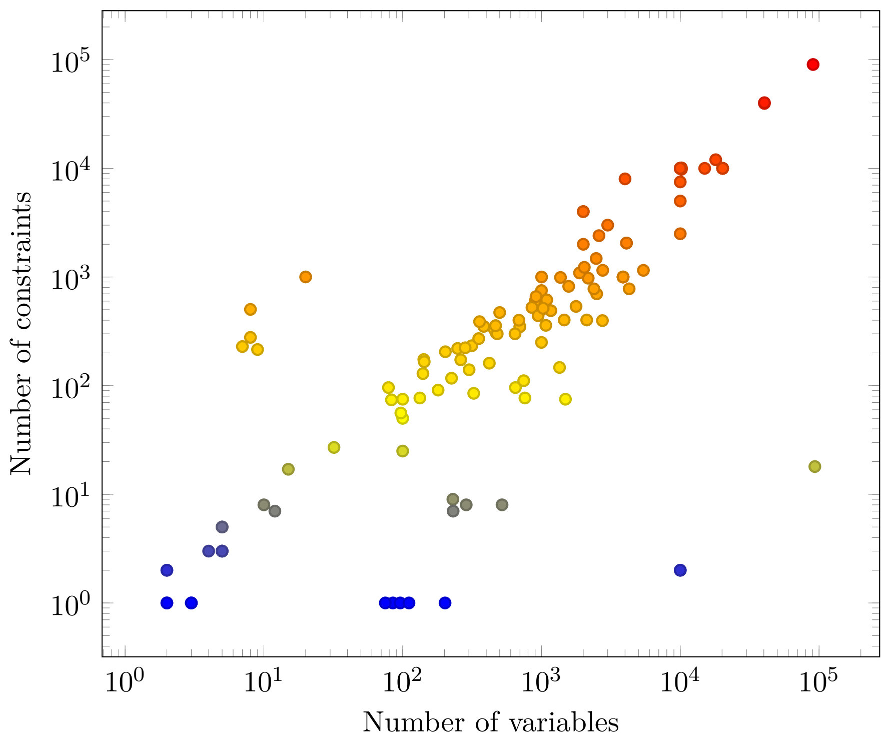


Figure 7.1: Distribution of the number of variables and constraints in the Maros and Mészáros convex QP test set.

QPDATA1 bundles 76 problems coming from the CUTE library (the predecessor to CUTEr/st [51, 52]). 46 problems, in QPDATA2, are provided by the Brunel optimization group and the remaining 16 of subset QPDATA3 come from miscellaneous sources. The complete test set includes both separable and nonseparable problems, and a reference optimal objective value computed by the interior-point QP solver BPMPD [9, 76]. The problems are available in QPS format, which is a subset of the SIF format used by CUTEr/st.

The distribution of problems within the Maros and Mészáros convex QP test set according to the number of variables and constraints is visualized in Figure 7.1. Less than 20 of the 138 problems in this test set have more than 1000 degrees of freedom, and less than 10 have more than 3000 degrees of freedom. For a more detailed look onto the set of problems or the data format the reader is referred to the technical report of Maros and Mészáros [73].

7.1.1 General Mean Scaling of QP Data

Due to the sensitivity of active-set methods to problem scaling it is often useful to scale the QP data to prevent the algorithm from numerical difficulties. Based on the *General-Mean Scaling* code gmscale.m by Saunders [80] the implementation to this thesis comprises a scaling method for matrices in TS format. Model scaling is not part of the presented framework but can be applied in a preprocessing step.

The scaling method is an iterative procedure based on geometric means. Several passes are made through the columns and rows of a supplied matrix $A \in \mathbb{R}^{m\times n}$. To dampen the effect of small matrix entries, on each pass through A, a new column scale $c_j \in \mathbb{R}$ or row scale $r_i \in \mathbb{R}$ will not be smaller than $\sqrt{\kappa_\text{d}}$ times the largest (scaled) entry in that column or row. The main steps of the procedure are stated in Algorithm 13.

Algorithm 13: Geometric-Mean Scaling

```
Input: (Un-)scaled sparse matrix $A \in \mathbb{R}^{m\times n}$, algorithmic constants $\kappa_\text{d}, \kappa_\text{scl}$,.
1  Compute $a_\text{rat} = \max_j (\max_i a_{ij} / \min_i a_{ij})$.
2  repeat
3      for $j = 1, \dots, n$ do
           // column scaling
4          Determine $c_\text{min} = \min_i a_{ij}$ and $c_\text{max} = \max_i a_{ij}$.
5          Divide column $j$ by $c_j = \sqrt{\max(c_\text{min}, \kappa_\text{d} \cdot c_\text{max}) \cdot c_\text{max}}$.
6      for $i = 1, \dots, m$ do
           // row scaling
7          Determine $r_\text{min} = \min_j a_{ij}$ and $r_\text{max} = \max_j a_{ij}$.
8          Divide row $i$ by $r_i = \sqrt{\max(r_\text{min}, \kappa_\text{d} \cdot r_\text{max}) \cdot r_\text{max}}$.
9      Compute $s_\text{rat} = \max_j (\max_i a_{ij} / \min_i a_{ij})$.
10 until $s_{rat} \geq \kappa_{scl} \cdot a_{rat}$.
```

The computed column- and row-scales determined by scaling $A = [A_\mathcal{E}^T, A_\mathcal{R}^T]^T \in \mathbb{R}^{(m+k)\times n}$ are applied to the data of the quadratic program by a transformation using the diagonal matrices $C = \operatorname{diag}(c_j)_{j\in\mathcal{B}} \in \mathbb{R}^{n\times n}$, and $R_\mathcal{E} = \operatorname{diag}(r_i)_{i\in\mathcal{E}} \in \mathbb{R}^{m\times m}$ and $R_\mathcal{R} = \operatorname{diag}(r_i)_{i\in\mathcal{R}} \in \mathbb{R}^{k\times k}$. With scaled variables $\bar{x} = Cx$ the scaled data is defined by

$$\bar{H} = C^{-1}HC^{-1}, \quad \bar{f} = C^{-1}f, \quad \bar{A_\mathcal{E}} = R^{-1}A_\mathcal{E}C^{-1}, \quad \bar{A_\mathcal{R}} = R^{-1}A_\mathcal{E}C^{-1},$$
$$\bar{b} = R_\mathcal{E}^{-1}b, \quad \bar{r_l} = R_\mathcal{R}^{-1}r_l, \quad \bar{r_u} = R_\mathcal{R}^{-1}r_u, \quad \bar{b_l} = Cb_l, \quad \bar{b_u} = Cb_u.$$

7.1.2 Performance Profiles

The visualization of the numerical results in the following is done using *performance profiles* in the form proposed by Dolan and Moré in [21]. For a set of problems P and a set of solvers (or solver options) S, the *performance measure*

$$t_{p,s} = \text{time required to solve problem } p \in P \text{ by solver } s \in S \tag{7.2}$$

is used. P is a significant subset of a set of test problems and S is the set of different solvers according to chosen parametrizations of the algorithm. The *performance ratio* is defined by

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}. \tag{7.3}$$

instant.	mode	multiple changes $\mathcal{W}$	KKT factor update	scaled QP data	relaxed cons. incl. in $\mathcal{W}_0$	κ_{ρ_1} – min. penalty
`L*base`	default	✓	–	–	–	10^2
`L*gfup`	default	✓	✓	–	–	10^2
`L*scal`	default	✓	–	✓	–	10^2
`L*feas`	feasible	–	–	–	✓	10^6

Table 7.1: Parametrization of Clean::ASM. Instances use either the ℓ_1 (`*=1`) or ℓ_2 (`*=2`) relaxation scheme.

Then, the fraction of problems that are solved by solver s within a factor $\tau \geq 1$ of the performance of the best solver for problem p can be expressed using

$$\rho_s(\tau) = \frac{1}{|P|}|\{p \in P\colon r_{p,s} \leq \tau\}|. \tag{7.4}$$

The profile $\rho_s(\tau)\colon \mathbb{R} \to [0,1]$ is non-decreasing and piecewise constant. Its basic interpretation is as follows: The value of $\rho_s(1)$ is the probability that s is the best of all solvers with respect to the performance measure $t_{p,s}$. Furthermore, by choosing $r_M \geq r_{p,s}$ for all p and all s and $r_{p,s} = r_M$ if and only if solver s does not solve problem p, it is $r_{p,s} \in [1, r_M]$ and

$$\rho_s^* = \lim_{\tau \nearrow r_M} \rho_s(\tau) \tag{7.5}$$

can be interpreted as the probability that s solves p. In other words, $\rho_s(1)$ is a measure of efficiency whereas ρ_s^* is a measure of robustness.

From now on, different parametrizations of Clean::ASM are interpreted as different solvers when results are visualized using performance profiles.

7.1.3 Results on The Maros and Mészáros Convex QP Test Set

In the following results of running Clean::ASM on the Maros and Mészáros convex QP test set are presented. To varify the flexibility and robustness of the code both relaxation schemes each in four different parametrizations are tested. An overview of the algorithmic differences is given in Table 7.1. Fixed constants are the overall tolerance κ_{tol}, the variable bound satisfaction tolerance κ_{bnd} and the step tolerance κ_{step} indicating zero steps as well as the lower limit of the penalization parameter κ_{ρ_1}. The values are:

$$\kappa_{\text{tol}} = 10^{-6}, \quad \kappa_{\text{step}} = 10^{-6}, \quad \kappa_{\text{bnd}} = 10^{-16}, \quad \kappa_{\rho_1} = 100. \tag{7.6}$$

Motivated by the usage of Clean::ASM within SQP the algorithm is started in the origin $0 \in \mathbb{R}^n$ for most of the problems in the test set. Starting points are available for problems provided by the CUTE library by calling csetup of the interface to CUTEr/st. Nevertheless, the supplied points may not necessarily be bound feasible. In this case the starting point is shifted into the feasible set $\mathcal{F}_{\mathcal{B}}$ setting $x_i \in \{b_{l,i}, b_{u,i}\}$ for all $i = 1$ depending on which limit is not satisfied. Since no information about the active-set is provided, the algorithm is cold started, i.e. $\mathcal{W}_0 = \mathcal{E}$, except for instances `L1feas` and `L2feas`, where all relaxed range constraints are included in the initial workingset, i.e. $\mathcal{W}_0 = \mathcal{E} \cup \mathcal{R}^r$.

Due to rank deficiency in the equality constraints Jacobians nine problems, namely QBORE3D, QBRANDY, QSCORPIO, QSHIP04L, QSHIP04S, QSHIP08L, QSHIP08S, QSHIP12L, QSHIP12S, are excluded. Here the LICQ is not satisfied for $\mathcal{W}_0 = \mathcal{E}$.[1] The factorization of the KKT matrices for problems BOYD1, BOYD2 and CVXQP3_L is prohibitively expensive with the result that the QP solution exceeds the time limit of 10 hours. The problems are not solved at the time of publication and are excluded, too.

Tables 7.2, 7.3 and 7.4 present the results of the fastest instantiation on each of the 126 problems in the test set. The columns are as follows: n is the number of variables, $|\mathcal{E}|$ is the number of equality constraints, $|\mathcal{R}_{l,u}|$ and $|\mathcal{B}_{l,u}|$ are the total number of lower and upper limits of range constraints and variable bounds. $|\mathcal{S} \cup \mathcal{T}|$ is the number of slack variables needed depending on the supplied starting point, ρ_0 is the initial penalty parameter determined by equation (4.77). $q(x^*)$ is the computed objective value at the final iterate. `inst` is the most efficient parametrization of Clean::ASM (cf. Table 7.1), `iter` is the number of iterations, `res` is the infinity norm of the primal infeasibility, `sec` is the solution time in seconds, and `rc` is the Clean::ASM return code. An optimal solution is found if `rc` $= 0$. `rc` $= 1$ indicates an optimal solution of the convexified QP. If `rc` $= 2$ the required feasibility tolerance is not achieved, i.e. the deliverd solution is not feasible.

In total, 121 out of 126 problems are solved (96%). Clean::ASM failed in five problems, where all parametrizations produce singular KKT systems or end up in stalling or cycling. This is caused by wrong decisions made in the workingset evoked by rounding errors in the constraint evaluation. The problems are namely QGFRDXPN, QPILOTNO, QSHELL, CONT-201 and STADAT1. Figures 7.2 and 7.3 show the performance of Clean::ASM using performance profiles. `L1best` and `L2best` denote the best choice of parameters per problem, visualizing the robustness of the relaxation scheme.

For the linear relaxation scheme, `L1gfup` is the fastest instantiation on most of the problems. It is not as robust as `L1base` and `L1scal` due to rounding errors provoked by the generic factorization update. `L2feas` outperforms all other parametrizations using the quadratic relaxation scheme in terms of efficiency and robustness. It benefits from the inclusion of relaxed range constraints in the initial workingset againts all other parametrizations for the problems in the test set. Feasibility is obtained earlier and additional iterations where indices switch from $\mathcal{R}^r \cap \mathcal{W}_k$ to $\mathcal{R}^f \cap \mathcal{W}_k$ are avoided (see Figure 7.4). Figure 7.5 presents the performance of the code if relaxed range constraints are included for all parametrizations, i.e. $\mathcal{W}_0 = \mathcal{E} \cup \mathcal{R}^r$. This also enables a direct comparison of the ℓ_1 and ℓ_2 relaxation scheme given in Figure 7.6. Therein `L*best` denotes the overall best choice of the relaxation scheme and parametrization.

[1]Compare also Assumption 1 (regualrity) in Section 3.2.

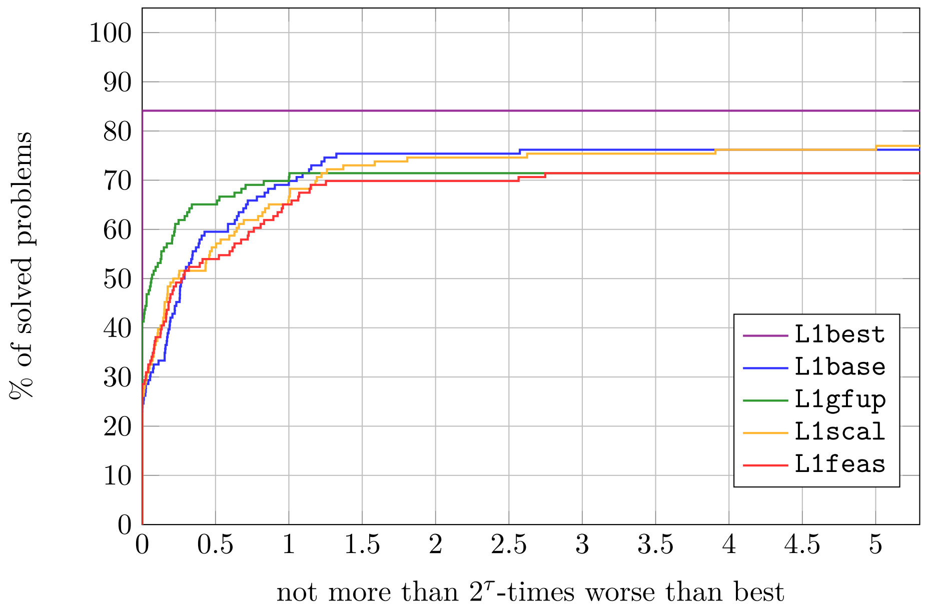


Figure 7.2: Performance profile for CPU time for Clean::ASM using ϕ_1.

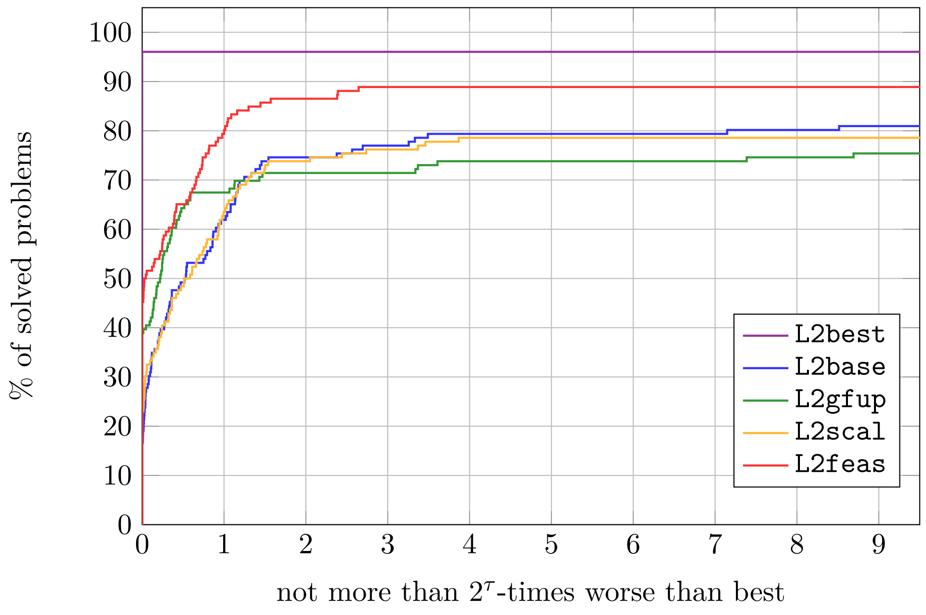


Figure 7.3: Performance profile for CPU time for Clean::ASM using ϕ_2.

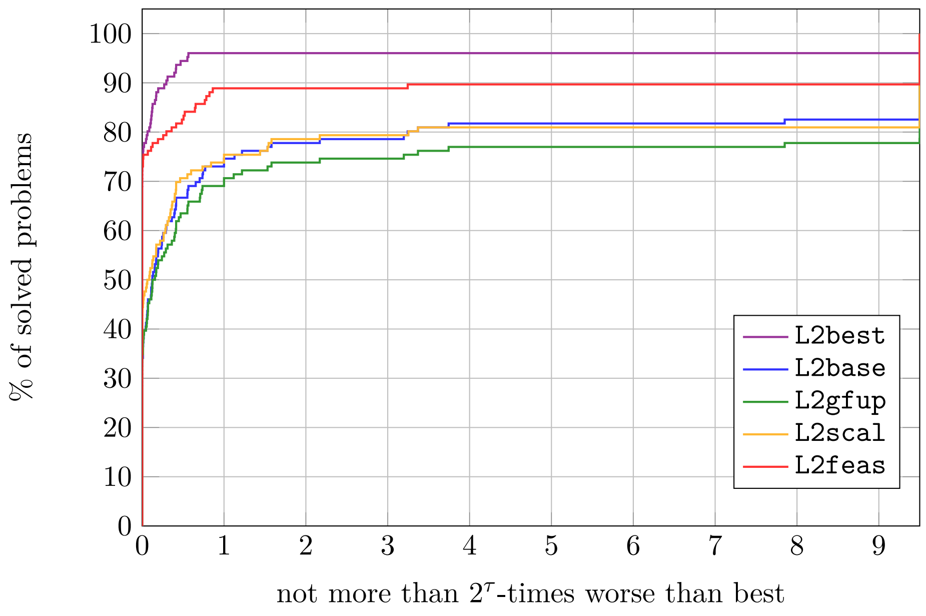


Figure 7.4: Performance profile for the number of iterations for Clean::ASM using ϕ_2.

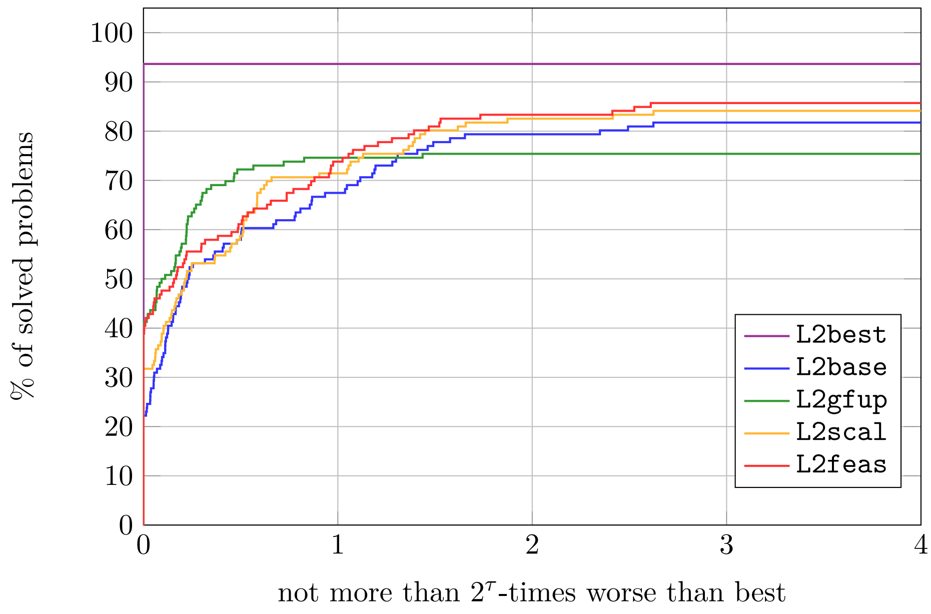


Figure 7.5: Performance profile for CPU time for Clean::ASM using ϕ_2 where relaxed constraints are included in $\mathcal{W}_0$.

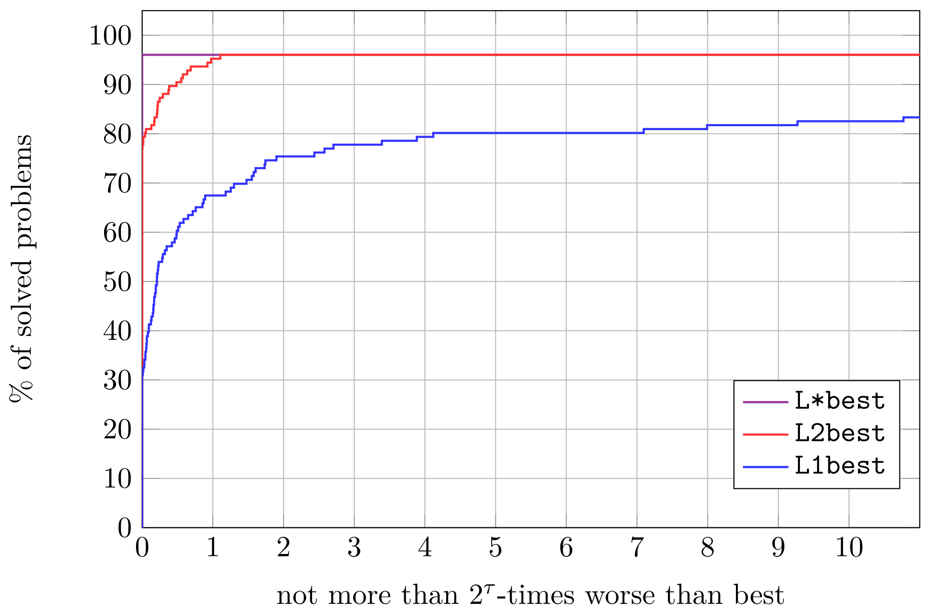


Figure 7.6: Performance profile for CPU time for Clean::ASM using the most efficient parametrization.

Problem Specific Parametrizaton and Warm Start

The QP solution algorithm is not designed for the solution of various test problems which comprise artificial difficulties using a single set of parameters. As a part of Clean, it is focused on the solution of difficult problems where the user has a good knowledge of the problem and is able to warm start. Example given, when Clean::ASM is used as the subsolver in Clean::SQP where it is warm started in a natural way. In the following, a more detailed view on the solution of a selection of difficult problems in the test set is given.

Some problems are solved (or feasibility is obtained) only when the starting point is shifted away from the boundary of the feasible set. In the tables below these problems are marked by `*`. The components of the starting point are centered between the limits if boxed and shifted into the interior of $\mathcal{F}_\mathcal{B}$ by $x_i = b_{l,i} + 1$ or $x_i = b_{u,i} - 1$ for all i. Some more are solved using adjusted algorithmic constants and tolerances: `LISWET12` needs a more loose step tolerance of $\kappa_{step} = 3.5 \cdot 10^{-6}$. `QPCBOEI1` requires scaled QP data. `QSCFXM2` has to be penalized with a penalty parameter of at least 10^{12}. And `QSCFXM3` is only solved by using $\kappa_{\rho_1} \geq 10^8$, $\kappa_{step} = 1.5 \cdot 10^{-5}$ and an initial convexification of $\nu_c = 1$.

An essential speed up is achieved, when the algorithmic choices and starting point suit the optimization problem or the code is warm started. E.g. cold starting on `EXDATA` at least takes 1502 iterations and 341.00 seconds (`L1gfup`) to determine the optimal solution. In contrast it is identified after 3 iterations and 11.41 seconds when all variables that do not appear in the equality constraint are fixed to their lower limit, i.e. $x_i^{(0)} = 0$ for all $i = 1, \ldots, n$ with $a_i = 0$. 1500 variables can be eliminated which reduces the KKT size significantly without infacting the LICQ.

Name	n	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}_{l,u}\rvert$	$\lvert\mathcal{B}_{l,u}\rvert$	$\lvert\mathcal{S}\cup\mathcal{T}\rvert$	ρ_0	$q(x^*)$	inst	iter	res	sec	rc
AUG2D	20200	10000	0	0	10000	1.00e+06	1.6874e+06	L2feas	9	7.22e-08	1.61	1
AUG2DC	20200	10000	0	0	10000	1.00e+06	1.8183e+06	L2feas	6	7.50e-08	0.56	0
AUG2DCQP	20200	10000	0	20200	9999	1.00e+06	6.4981e+06	L2feas	10210	2.78e-07	1283.86	0
AUG2DQP	20200	10000	0	20200	9998	9.80e+03	6.2370e+06	L2base	9729	2.78e-07	1211.71	1
AUG3D	3873	1000	0	0	1000	1.00e+06	5.5406e+02	L2feas	7	4.35e-08	0.79	1
AUG3DC	3873	1000	0	0	1000	1.00e+06	7.7126e+02	L2feas	30	4.64e-08	0.82	0
AUG3DCQP	3873	1000	0	3873	952	1.69e+03	9.9336e+02	L2gfup	866	4.21e-07	9.94	0
AUG3DQP	3873	1000	0	3873	952	1.09e+03	6.7523e+02	L2gfup	847	5.73e-07	6.39	1
CVXQP1_L	10000	5000	0	20000	5000	5.62e+07	1.0870e+08	L2gfup	7402	1.24e-06	3341.99	2
CVXQP1_M	1000	500	0	2000	500	5.63e+05	1.0875e+06	L2gfup	715	8.89e-07	11.25	0
CVXQP1_S	100	50	0	200	50	5.68e+03	1.1590e+04	L2base	63	2.04e-07	0.05	0
CVXQP2_L	10000	2500	0	20000	2500	5.62e+07	8.1842e+07	L2gfup	5750	1.25e-07	665.09	0
CVXQP2_M	1000	250	0	2000	250	5.63e+05	8.2015e+05	L2gfup	579	5.83e-07	1.83	0
CVXQP2_S	100	25	0	200	25	5.68e+03	8.1209e+03	L1gfup	88	2.03e-14	0.03	0
CVXQP3_M	1000	750	0	2000	750	1.00e+06	1.3628e+06	L2feas	1786	7.15e-07	90.05	0
CVXQP3_S	100	75	0	200	75	5.68e+03	1.1943e+04	L2gfup	71	1.48e-07	0.05	0
DTOC3	14999	9998	0	4	9960	1.00e+06	2.3524e+02	L2feas	2147	2.76e-07	172.31	0
DUAL1	85	1	0	170	1	1.00e+02	3.5012e-02	L1gfup	28	0.00e+00	0.03	0
DUAL2	96	1	0	192	1	1.00e+02	3.3733e-02	L1gfup	7	0.00e+00	0.01	0
DUAL3	111	1	0	222	1	1.00e+02	1.3575e-01	L1gfup	17	0.00e+00	0.03	0
DUAL4	75	1	0	150	1	1.00e+02	7.4609e-01	L1gfup	16	0.00e+00	0.01	0
DUALC1	9	1	214	18	1	3.36e+06	6.1552e+03	L1gfup	12	0.00e+00	0.01	0
DUALC2	7	1	228	14	1	1.00e+06	3.5513e+03	L2feas	11	4.72e-07	0.00	0
DUALC5	8	1	277	16	1	1.00e+06	4.2723e+02	L2feas	9	4.60e-08	0.00	0
DUALC8	8	1	502	16	1	1.00e+06	1.8309e+04	L2feas	14	3.32e-08	0.01	1
GENHS28	10	8	0	0	7	1.00e+06	9.2717e-01	L2feas	2	1.34e-07	0.00	0
GOULDQP2	699	349	0	1398	349	1.00e+06	1.8427e-04	L2feas	1007	1.01e-10	2.49	0
GOULDQP3	699	349	0	1398	349	1.00e+06	2.0627e+00	L2feas	185	2.97e-07	0.75	0
HS118	15	0	29	30	0	9.42e+02	6.6482e+02	L2gfup	18	0.00e+00	0.00	0
HS21	2	0	1	4	0	1.00e+02	-9.9960e+01	L2gfup	3	0.00e+00	0.00	0
HS268	5	0	5	0	0	3.09e+04	7.2759e-12	L2gfup	2	0.00e+00	0.00	0
HS35	3	0	1	3	0	1.00e+02	1.1111e-01	L2gfup	3	0.00e+00	0.00	0
HS35MOD	3	0	1	4	0	1.00e+02	2.5000e-01	L2gfup	3	0.00e+00	0.00	0
HS51	5	3	0	0	0	1.00e+02	0.0000e+00	L2gfup	2	0.00e+00	0.00	0
HS52	5	3	0	0	1	1.00e+06	5.3266e+00	L2feas	4	3.27e-08	0.00	0
HS53	5	3	0	10	1	1.00e+06	4.0930e+00	L2feas	4	2.04e-08	0.00	0
HS76	4	0	3	4	0	1.00e+06	-4.6818e+00	L1feas	7	0.00e+00	0.00	0
HUES-MOD	10000	2	0	10000	2	1.00e+02	3.4824e+07	L1gfup	561	0.00e+00	9.49	0
HUESTIS	10000	2	0	10000	2	5.08e+03	3.4824e+11	L1gfup	563	3.46e-18	8.65	0
KSIP	20	0	1001	0	0	1.00e+02	5.7579e-01	L2base	363	0.00e+00	1.77	0
LISWET1	10002	0	10000	0	0	1.00e+06	3.6122e+01	L2feas	10002	0.00e+00	532.19	0
LISWET10	10002	0	10000	0	0	1.00e+06	4.9487e+01	L2feas	10173	0.00e+00	540.61	0
LISWET11	10002	0	10000	0	0	1.00e+06	4.9524e+01	L2feas	10257	0.00e+00	540.37	0
LISWET12	10002	0	10000	0	0	1.00e+06	1.7369e+03	L2feas	10043	0.00e+00	501.23	0
LISWET2	10002	0	10000	0	0	1.00e+06	2.4998e+01	L2feas	10003	0.00e+00	534.64	0
LISWET3	10002	0	10000	0	0	1.00e+06	2.5001e+01	L2feas	10699	0.00e+00	563.81	0
LISWET4	10002	0	10000	0	0	1.00e+06	2.5000e+01	L2feas	10758	0.00e+00	558.69	0
LISWET5	10002	0	10000	0	0	1.60e+04	2.5034e+01	L2scal	10654	0.00e+00	570.11	0
LISWET6	10002	0	10000	0	1	1.00e+06	2.4995e+01	L2feas	10562	0.00e+00	560.23	0
LISWET7	10002	0	10000	0	1	1.93e+03	4.9845e+02	L2scal	10089	1.45e-07	535.08	0
LISWET8	10002	0	10000	0	2	1.00e+06	7.1455e+02	L2feas	11577	1.59e-08	615.59	0
LISWET9	10002	0	10000	0	2	3.15e+03	1.9632e+03	L1base	10793	0.00e+00	584.09	0
LOTSCHD	12	7	0	12	7	1.00e+02	2.3984e+03	L2gfup	21	2.95e-07	0.00	0
MOSARQP1	2500	0	700	2500	0	5.36e+02	-9.5287e+02	L2gfup	1523	0.00e+00	7.11	0
MOSARQP2	900	0	600	900	0	1.00e+02	-1.5974e+03	L2gfup	334	0.00e+00	0.98	0
POWELL20	10000	0	10000	0	5000	1.00e+06	5.2089e+10	L2feas	5009	6.28e-08	277.86	0
PRIMAL1	325	0	85	1	0	1.00e+02	-3.5012e-02	L2gfup	76	0.00e+00	0.88	0
PRIMAL2	649	0	96	1	0	1.00e+02	-3.3733e-02	L2scal	102	0.00e+00	2.37	0
PRIMAL3	745	0	111	1	0	1.00e+02	-1.3575e-01	L2scal	107	0.00e+00	12.47	0
PRIMAL4	1489	0	75	1	0	1.00e+02	-7.4609e-01	L2scal	66	0.00e+00	3.58	0
PRIMALC1	230	0	9	215	0	1.00e+02	-6.1552e+03	L2gfup	220	0.00e+00	0.12	0
PRIMALC2	231	0	7	229	0	1.00e+02	-3.5513e+03	L2gfup	236	0.00e+00	0.11	0
PRIMALC5	287	0	8	278	0	1.00e+02	-4.2723e+02	L2gfup	286	0.00e+00	0.16	0
PRIMALC8	520	0	8	503	0	1.00e+02	-1.8309e+04	L2gfup	515	0.00e+00	0.52	0
QPCBLEND	83	43	31	83	0	1.00e+02	-7.8425e-03	L2gfup	241	0.00e+00	0.12	0
QPCBOEI1	384	9	431	540	113	1.00e+06	1.1503e+07	L2feas	1755	4.44e-08	4.10	0
QPCBOEI2	143	4	181	197	36	1.00e+02	8.1719e+06	L2scal	702	1.62e-07	0.76	0
QPCSTAIR	467	209	147	549	127	2.04e+05	6.2043e+06	L2gfup	1766	4.71e-07	9.80	0
S268	5	0	5	0	0	3.09e+04	7.2759e-12	L2gfup	2	0.00e+00	0.00	0
STCQP1	4097	2052	0	8194	2052	5.64e+04	1.5514e+05	L2gfup	519	3.91e-07	21.98	0
STCQP2	4097	2052	0	8194	2052	1.00e+06	2.2327e+04	L2feas	377	2.32e-08	63.05	0
TAME	2	1	0	2	1	1.00e+02	2.0707e-30	L2gfup	2	0.00e+00	0.00	0
UBH1	18009	12000	0	12030	6	1.00e+02	1.1160e+00	L2base	3979	9.19e-12	267.7	0
YAO	2002	0	2000	5	0	1.00e+06	1.9770e+02	L2feas	2005	0.00e+00	19.62	0
ZECEVIC2	2	0	2	4	0	1.00e+02	-4.1250e+00	L2scal	5	0.00e+00	0.00	0

Table 7.2: Results of Clean::ASM on 76 problems of the the Maros and Mészáros QP Test Problem Set provided by the CUTE library (QPDATA1).

Name	n	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}_{l,u}\rvert$	$\lvert\mathcal{B}_{l,u}\rvert$	$\lvert\mathcal{S}\cup\mathcal{T}\rvert$	ρ_0	$q(x^*)$	inst	iter	res	sec	rc
Q25FV47*	1571	515	305	1571	586	6.67e+04	1.3745e+07	L2scal	10137	5.96e-07	327.36	1
QADLITTL	97	15	41	97	8	3.31e+03	4.8031e+05	L1base	199	1.13e-13	0.09	1
QAFIRO*	32	8	19	32	12	1.00e+02	-1.5907e+00	L1scal	40	0.00e+00	0.01	0
QBANDM*	472	305	0	472	301	1.00e+02	1.6730e+04	L2scal	1087	2.12e-07	4.62	1
QBEACONF	262	140	33	262	34	1.70e+02	-5.4662e+05	L2scal	142	1.45e-07	0.76	1
QCAPRI	353	142	129	486	123	1.00e+02	6.6792e+07	L1gfup	1157	8.05e-12	2.65	1
QE226	282	33	190	282	35	1.00e+02	-5.9239e+00	L2gfup	1524	3.28e-07	3.23	1
QETAMACR	688	272	128	905	94	9.99e+02	7.0297e+04	L2base	2062	6.15e-08	12.45	1
QFFFFF80	854	350	174	854	153	2.56e+02	8.9153e+05	L1scal	1819	2.27e-13	11.97	1
QFORPLAN*	421	90	72	445	112	6.71e+07	7.4566e+09	L2feas	1150	6.70e-07	6.01	1
QGFRDXPN	1092	548	68	1350	2	6.71e+07	5.1843e+10	L1scal	848	0.00e+00	2.89	7
QGROW15	645	300	0	1245	0	1.00e+02	-1.1066e+01	L2scal	1168	0.00e+00	7.28	1
QGROW22	946	440	0	1826	0	1.00e+02	-1.6000e+01	L2scal	1335	0.00e+00	12.21	1
QGROW7	301	140	0	581	0	1.00e+02	-4.7538e+00	L2scal	869	0.00e+00	2.47	1
QISRAEL	142	0	174	142	8	3.00e+03	2.5359e+07	L2gfup	252	9.69e-08	0.25	1
QPILOTNO*	2172	701	274	2716	775	2.89e+06	2.9025e+06	L1feas	2488	0.00e+00	23.53	7
QRECIPE	180	67	24	275	15	1.00e+02	6.0172e+01	L1gfup	71	0.00e+00	0.04	1
QSC205	203	91	114	203	0	1.00e+02	-5.4752e-03	L2gfup	84	0.00e+00	0.08	1
QSCAGR25	500	300	171	500	57	5.10e+02	2.0176e+08	L2scal	1200	2.11e-12	4.04	1
QSCAGR7	140	84	45	140	21	5.10e+02	2.6867e+07	L2scal	235	2.84e-14	0.23	1
QSCFXM1*	457	187	143	457	232	1.00e+06	1.6883e+07	L2feas	1306	5.21e-08	4.53	1
QSCFXM2	914	374	286	914	464	1.00e+12	2.7785e+07	L2feas	3023	3.92e-09	18.41	1
QSCFXM3*	1371	561	429	1371	696	1.00e+08	3.0835e+07	L2feas	3827	3.42e-08	34.64	1
QSCRS8	1169	384	106	1169	38	5.30e+03	1.1397e+03	L2base	1938	1.14e-07	7.92	1
QSCSD1	760	77	0	760	1	1.00e+02	8.6666e+00	L1gfup	829	0.00e+00	1.37	1
QSCSD6	1350	147	0	1350	9	1.00e+02	5.0808e+01	L1gfup	1904	2.77e-17	5.98	1
QSCSD8*	2750	397	0	2750	15	1.04e+04	9.4076e+02	L1gfup	2652	2.11e-12	18.91	1
QSCTAP1	480	120	180	480	154	1.00e+02	1.4158e+03	L1base	1002	6.66e-16	2.11	1
QSCTAP2	1880	470	620	1880	521	1.00e+02	1.7350e+03	L2scal	2215	2.56e-07	14.02	1
QSCTAP3	2480	620	860	2480	682	1.00e+02	1.4387e+03	L2scal	2988	4.84e-07	25.51	1
QSEBA	1028	507	15	1535	224	4.40e+02	8.1470e+07	L2gfup	1132	7.53e-07	5.00	1
QSHARE1B	225	89	28	225	75	1.00e+02	8.2252e+05	L2gfup	792	1.19e-08	0.90	1
QSHARE2B	79	13	83	79	5	1.00e+06	1.1703e+04	L2feas	280	2.63e-07	0.17	1
QSHELL	1775	534	2	2142	257	6.71e+07	1.0446e+12	L1feas	1894	0.00e+00	33.23	7
QSIERRA	2036	528	699	4072	148	6.31e+04	2.3751e+07	L2base	2665	1.37e-07	18.96	1
QSTAIR*	467	209	147	549	292	2.42e+05	7.9854e+06	L2gfup	1640	2.88e-08	10.66	1
QSTANDAT	1075	160	199	1195	12	1.00e+06	-8.5760e+01	L2feas	1198	2.25e-08	6.01	1

Table 7.3: Results of Clean::ASM on 46 problems of the the Maros and Mészáros QP Test Problem Set provided by the Mathematical Programming Group, Brunel University, London (QPDATA2).

Name	n	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}_{l,u}\rvert$	$\lvert\mathcal{B}_{l,u}\rvert$	$\lvert\mathcal{S}\cup\mathcal{T}\rvert$	ρ_0	$q(x^*)$	inst	iter	res	sec	rc
CONT-050	2597	2401	0	5194	2401	1.00e+06	-4.5638e+00	L2feas	3	1.75e-07	0.43	0
CONT-100	10197	9801	0	20394	9801	1.00e+06	-4.6444e+00	L2feas	4	2.13e-07	4.68	0
CONT-101*	10197	10098	0	20394	689	1.00e+02	1.9551e-01	L2scal	438	6.52e-07	442.17	0
CONT-200	40397	39601	0	80794	39601	1.00e+06	-4.6848e+00	L2feas	4	2.47e-07	34.50	0
CONT-201	40397	40198	0	80794	398	1.00e+02	0.0000e+00	L1gfup	12	2.50e-02	29.53	3
CONT-300*	90597	90298	0	181194	598	1.00e+06	1.9150e-01	L2feas	5	4.33e-07	158.43	0
DPKLO1	133	77	0	0	49	1.00e+06	3.7009e-01	L2feas	3	3.50e-07	0.01	0
EXDATA	3000	1	3000	4500	0	1.00e+02	0.0000e+00	L1gfup	1502	0.00e+00	341.00	0
LASER	1002	0	2000	0	1000	1.00e+06	2.4096e+06	L2feas	1575	2.87e-07	16.47	0
QPTEST	2	0	2	3	1	1.00e+06	4.3718e+00	L2feas	4	4.27e-08	0.00	0
STADAT1	2001	0	5999	0	0	1.00e+06	-2.1408e+07	L2feas	2004	0.00e+00	27.33	3
STADAT2	2001	0	5999	0	0	1.00e+06	-3.2626e+01	L2feas	2074	0.00e+00	29.06	0
STADAT3	4001	0	11999	0	0	1.00e+06	-3.5779e+01	L2feas	4041	0.00e+00	109.77	0
VALUES	202	1	0	404	0	1.00e+06	0.0000e+00	L2feas	4	0.00e+00	0.01	1

Table 7.4: Results of Clean::ASM on 16 problems of the the Maros and Mészáros QP Test Problem Set provided by Groeneboom, Mittelmann, Chalimourda, Boyd, McNames and Wolkowitz (QPDATA3).

7.2 Computation of Recombination Parameters in Mathematical Biology

In the context of his diploma thesis [79] and PhD thesis Probst investigates the usability of numerical optimization techniques as an alternative to stochastic simulation in the field of mathematical biology. He observes a population of a fixed size and tries to predict the evolution in time by the characterization of allelic states at a number loci at the same chromosome. For the determination of recombination probabilities he uses a Moran model with single crossovers – which yields a Markov chain in continous time – allowing changes only in between two arbitrary loci at time t. The multilocus Moran model is illustrated in Figure 7.7.

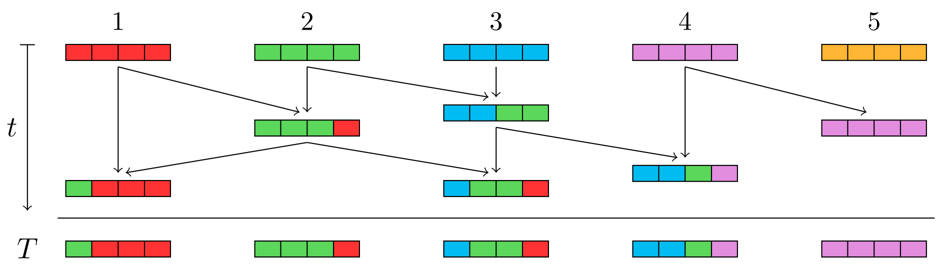


Figure 7.7: Snapshot of a Moran Model realization with $N = 5$ individuals and $n = 4$ loci. For example, in the first eventpoint, individual 3 dies and is replaced by a copy of individuals 2 and 3. The last line shows the composition of the population at $t = T$, see [24].

In the following, N denotes the size of the population and n is the number of loci. The probability of recombination after a certain locus i is stated as r_i for $i = 1, \ldots, n$ and, by convention, r_1 soaks up the event of no recombination.

The dynamics of the recombination process come from a duality relation between the Moran model and the ancestral partitioning process proved by Esser et al. [24]. The usage of a Markov process model leads to an initial value problem for the dynamics. It results in a separated multistage boundary value problem of infinit dimension, reading

$$\min_{(h,r)\in\mathcal{H}\times\mathbb{R}^n} \; \frac{1}{2}\sum_{j=0}^{m} \|h(t_j,r) - \hat{h}_j\|^2 \tag{7.7a}$$

$$\text{s.t.} \quad \tfrac{\mathrm{d}}{\mathrm{d}t}h(t) - \Theta(N,r)h(t) = 0, \qquad t \in [0,T] \tag{7.7b}$$

$$1 - \sum_{i=1}^{n} r_i = 0, \tag{7.7c}$$

$$r_i \geq 0, \qquad i = 1,\ldots,n, \tag{7.7d}$$

where $t_j \leq T$ defines the time points of observation for $j = 0, \ldots, m$. Since the model is highly complicated, only a brief description is given. The vector $r \in \mathbb{R}^n$ contains the recombination parameters. With respect to the *Bell number*[2] ω_n, $h \in \mathcal{H} \subset C^0(I, \mathbb{R}^{\omega_n})$ represents the expectation of type-distribution with respect to all possible partitions of $\{1, \dots, n\}$ matching simulated data points $\hat{h}_j = h(t_j)$ for all time points t_j. Finally, the partitioning process at time t_j is described by infitesimal generator $\Theta(N, r)$.

Probst uses direct Multiple Shooting [20] for the discretization of (7.7b) to obtain a multistage boundary value problem (MSBVP) of finite dimension. More precisely, he uses the observation time points t_j as the shooting points for the discretization. The resulting NLP has $\omega_n(m+1)+n$ variables and $\omega_n m$ equality constraints corresponding to the dynamics (7.7b), one equality constraint for equation (7.7c) and n variable bounds (lower limits). Numerical solutions of the initial value problems on $[t_{j-1}, t_j]$, $j = 1, \dots, m$, are computed using the integrators difsys [11, 56] and Metanb [3]. In his computational experiments the discretized problem is solved using Clean::SQP and Clean::ASM. Matrix data is stored in triplet sparse format, and linear systems are solved using the sparse solver MA57 of the HSL library.

Figure 7.8 presents results of an inverse parameter estimation obtained by solving (7.7) with a fixed population size of 10.000 individuals, three parameters and five partitions. The discretized NLP has 108 variables, 101 equality constraints and three variable bounds. The problem is relatively small, but it turns out that the initial value problem (7.7b) has to be solved with high accuracy in every iteration. Unfortunately, this is the most expensive computation in the solution of the problem.

The computed parameter vector $r^*_{\text{opt}} = (r_1, r_2, r_3)^T$ at the determined solution is

$$r^*_{\text{opt}} = (0.997367, 0.001460, 0.001173)^T.$$

Expected parameters given by simulation are $r^*_{\text{sim}} = (0.997500, 0.001250, 0.001250)^T$, yielding an absolute error of

$$\|r^*_{\text{opt}} - r^*_{\text{sim}}\|_2 = 0.000260.$$

Enlarging the problem, by expanding the model to four parameters, i.e. $\omega_4 = 15$ partitions, yields a NLP comprising 319 variables, 301 equalities and four lower limits of variables. The obtained solution is

$$r^*_{\text{opt}} = (0.998170, 0.000920, 0.000883, 0.000000)^T \quad \text{with} \quad \|r^*_{\text{opt}} - r^*_{\text{sim}}\|_2 = 0.0011.$$

The size of the NLP is scaled linearly by the size of m, the number of time points in the observation, and exponentially by n, the number of loci, due to the definition of the Bell number ω_n. The size of the population only infects the continuity conditions of discretization and affects the computational costs in the solution of (7.7b). In applied

[2]The first 6 Bell numbers are $\omega_1 = 1$, $\omega_2 = 2$, $\omega_3 = 5$, $\omega_4 = 15$, $\omega_5 = 52$, $\omega_6 = 203$, $\omega_7 = 877$. For details, see [79] and the thesis of Probst.

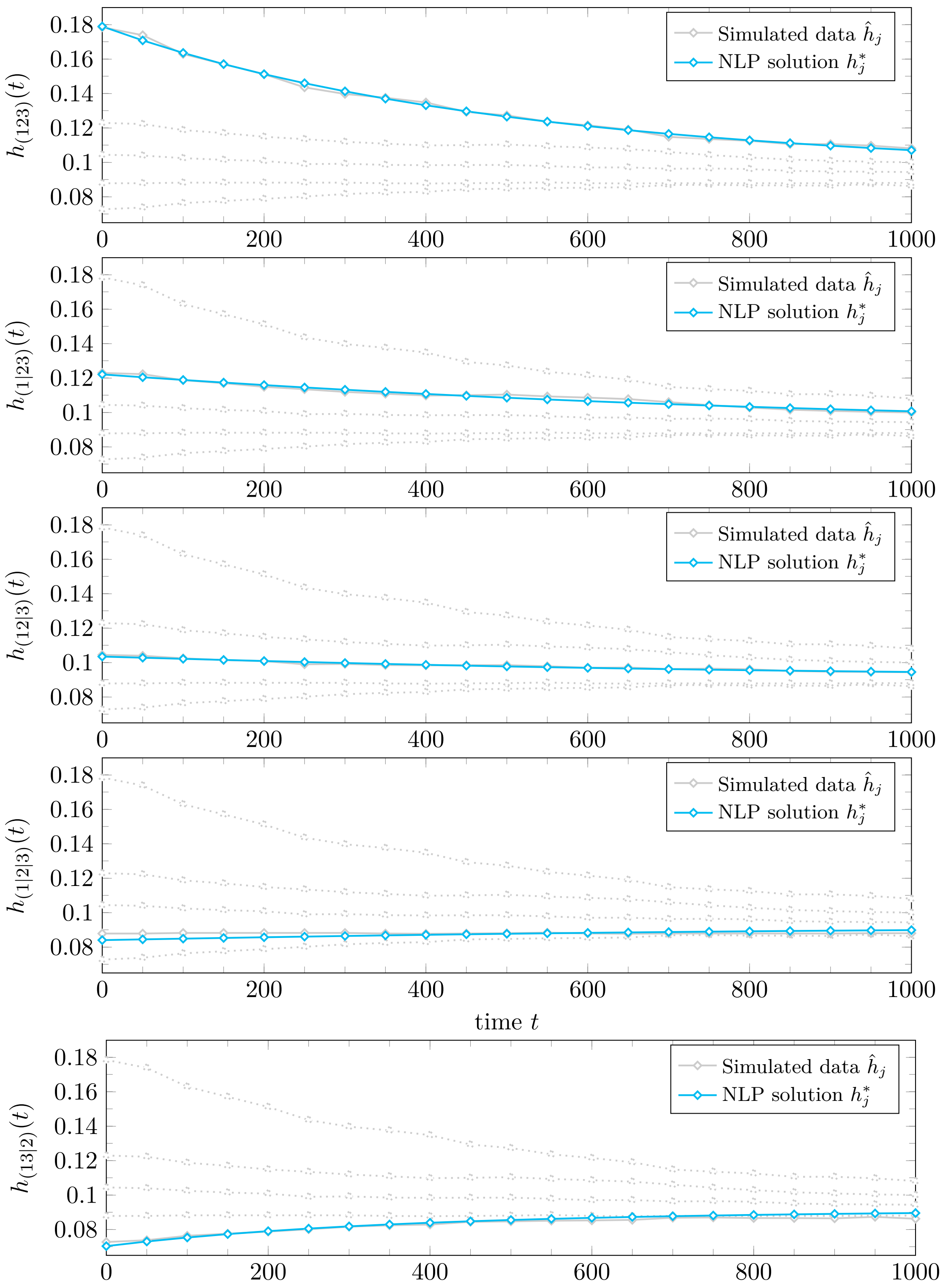


Figure 7.8: Expected type-distributions with respect to all possible partitions of $\{1, 2, 3\}$. Allelic states at $n = 3$ loci ($\Rightarrow \omega_3 = 5$) for $m = 20$ time points are observed. $h(t) = (h_{(123)}(t), \ldots, h_{(1|2|3)}(t))^T$ denotes the time-course vector for a (single) fixed type. The popultion comprises 10.000 individuals.

mathematical biology evolutions in time are observed for around seven loci at a larger number of observations. This, combined with a finer discretization used for multiple shooting to improve the accuracy, leads to large-scale optimization problems with several thousand variables and constraints.

The presented results are obtained within seconds and deal as a proove of concept for the usage of numerical optimization in the field of mathematical biology. Since the structure of the problem remains the same for large values of m, n and N it confirms the applicability of Clean::SQP of Probst's approach. Results for large-scale problems are not veryfied at the time of publication.

7.3 Results for Multistage Boundary Value Problems with Local Constraints

In this section the employability of specialized subsolvers within Clean::SQP is prooved by the use of a structure exploiting KKT solver for the solution of a multistage optimization problem for dynamic processes modeled by ordinary differential equations (ODE).

Assuming separability, i.e. variables on different stages $j = 0, \dots, N$ are at most linearly coupled, the representing NLP belongs to the class of *ODE-constrained multistage boundary value problems* (ODE-MSBVP), cf. [88]. For $I = [0, T]$ and $0 = t_0 < \dots < t_N = T$ let

$$I_j = (t_{j-1}, t_j), \quad \mathcal{X}_j = C^1(I_j, \mathbb{R}^{n_x}), \quad \mathcal{U}_j = C^0(I_j, \mathbb{R}^{n_x}) \tag{7.8}$$

as well as $\mathcal{X} = \{x \in C^0(I, \mathbb{R}^{n_x}) \colon x_{|I_j} \in \mathcal{X}_j\}$ and $\mathcal{U} = \{u \colon I \to \mathbb{R}^{n_u} \colon u_{|I_j} \in \mathcal{U}_j\}$. Then, problems of the described class read

$$\min_{x \in \mathcal{X}, u \in \mathcal{U}} \quad \sum_{j=0}^{N} f_j(x(t_j)) \tag{7.9a}$$

$$\text{s.t.} \quad 0 = \sum_{j=0}^{N} r_j(x(t_j)), \tag{7.9b}$$

$$0 = \dot{x}(t) - g(x(t), u(t)), \quad t \in I, \tag{7.9c}$$

$$b(x(t), u(t)) \geq 0, \qquad t \in I. \tag{7.9d}$$

Choosing a proper finite dimensional subspace $\bar{\mathcal{U}} \subset \mathcal{U}$, e.g. piecewise constant functions, and control $\bar{u}_0$ the initial value $x(t_0) = \bar{x}_0$ allows the representation of $x(t)$ as a function of $t_0, \bar{x}_0, \bar{u}_0$ (Picard-Lindelöf) yielding

$$x(t) = G(t, t_0, \bar{x}_0, \bar{u}_0), \quad t \in I_1. \tag{7.10}$$

The same applies to I_j, $j > 1$ such that integration over I_j gives

$$G_j(\bar{x}_{j-1}, \bar{u}_{j-1}) = G(t, t_{j-1}, \bar{x}_{j-1}, \bar{u}_{j-1}). \tag{7.11}$$

If the inequality constraints (7.9d) are only supposed to hold for time points t_j finite dimensional MSBVPs in a multiple shooting approach can be stated as

$$\begin{aligned}
\min_{\bar{x}\in\bar{\mathcal{X}},\bar{u}\in\bar{\mathcal{U}}} \quad & \sum_{j=0}^{N} f_j(\bar{x}_j) && \text{(7.12a)}\\
\text{s.t.} \quad & 0 = \sum_{j=0}^{N} r_j(\bar{x}_j), && \text{(7.12b)}\\
& 0 = G_j(\bar{x}_{j-1}, \bar{u}_{j-1}) - \bar{x}_j, \quad j = 1, \dots, N, && \text{(7.12c)}\\
& b(\bar{x}_j, \bar{u}_j) \geq 0, \qquad\qquad\qquad j = 0, \dots, N. && \text{(7.12d)}
\end{aligned}$$

Problem (7.12) has a chain structure defined by the underlying discrete dynamic system (7.12c). It is delegated to the quadratic subproblems and KKT systems in active-set SQP methods, cf. Chapter 3 and 4. The arising KKT systems

$$\begin{bmatrix} H & G^T & F^T \\ G & & \\ F & & \end{bmatrix} \begin{pmatrix} y \\ -\lambda \\ -\mu \end{pmatrix} = \begin{pmatrix} -f \\ h \\ e \end{pmatrix}. \tag{7.13}$$

with its full multistage block-sparse substructure are stated as follows: partitionings of the block-diagonal Hessian, state increment and the objective gradient gives

$$H = \begin{bmatrix} H_0 & J_0^T & & & \\ J_0 & K_0 & & & \\ & & \ddots & & \\ & & & H_N & J_N^T \\ & & & J_N & K_N \end{bmatrix}, \quad y = \begin{pmatrix} x_0 \\ u_0 \\ \vdots \\ x_N \\ u_N \end{pmatrix}, \quad f = \begin{pmatrix} f_0 \\ d_0 \\ \vdots \\ f_N \\ d_N \end{pmatrix}. \tag{7.14}$$

Partitioning of the matrix, multiplier and vector of continuity conditions yields

$$G = \begin{bmatrix} G_1 & E_1 & -I & 0 & & & \\ & & \ddots & & & & \\ & & & G_N & E_N & -I & 0 \end{bmatrix}, \quad \lambda = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_N \end{pmatrix}, \quad h = \begin{pmatrix} h_1 \\ \vdots \\ h_N \end{pmatrix}. \tag{7.15}$$

Matrix, multiplier and vector of the boundary conditions are partitioned according to

$$F = \begin{bmatrix} F_0^x & & & & \\ & D_0^x & & & \\ F_0^c & D_0^c & & & \\ & & \ddots & & \\ & & & F_N^x & \\ & & & & D_N^x \\ & & & F_N^c & D_N^c \\ F_0 & D_0 & \cdots & F_N & D_N \end{bmatrix}, \quad \mu = \begin{pmatrix} \mu_0^x \\ \mu_0^u \\ \mu_0^c \\ \vdots \\ \mu_N^x \\ \mu_N^u \\ \mu_N^c \end{pmatrix}, \quad e = \begin{pmatrix} e_0^x \\ e_0^u \\ e_0^c \\ \vdots \\ e_N^x \\ e_N^u \\ e_N^c \end{pmatrix}. \tag{7.16}$$

Augmentation of continuity and boundary conditions yields the reduced KKT system

$$\begin{bmatrix} H & C^T \\ C & \end{bmatrix} \begin{pmatrix} y \\ -\eta \end{pmatrix} = \begin{pmatrix} -g \\ c \end{pmatrix} \tag{7.17}$$

which is compatible to the presented linear regularization scheme of Clean::ASM in Section 4.2.2. For a detailed look onto the sparse structure of (7.13) and a recursive $O(N)$ algorithm for its solution see the thesis of Steinbach [83].

Interfaces

Steinbach et. al. presented the C-code MSTOP [89] for tackling optimization problems of type (7.9). It includes a classical Runge–Kutta method (rk4) for the solution of (7.9c) and incorporates a stage-wise BFGS update scheme for approximate Hessians of type (7.14). Both tools are wrapped into a NLP problem class written in C++ suiting the required interface for the problem evaluation and are plugged into Clean::SQP. This shows, that the implementation of powerful problem classes - like the one at hand - is easy to achieve due to the simple interface to Clean which outlines the quality of the code and software concept.

As a part of MSTOP, the code MSKKT is used to solve arising linear systems of type (7.17) in the presented linear relaxation scheme of Clean::ASM. Again, the interface is simple; it only requires a proper initialization and projection of the KKT data with respect to the workingset.

In the following the one-dimensional frictionless motion of a high velocity magnetic levitation vehicle, which is known as the *rocket car* [70] in the literature, is modeled as a MSBVP. Computational results are presented as a proof of concept for the flexibility of the code developed for this thesis.

7.3.1 Example: Rocket Car

The motion of the rocket car follows Newton's third law of motion. It is $\ddot{s}(t) = \dot{v}(t) = F/m$, where s denotes the position of the vehicle, v its velocity and F is the driving power and m its mass. Starting in $s(0) = s_0$ with $v(0) = v_0$, the task is to move the vehicle in minimal time T to position $s(T) = s_e$ with a specified velocity $v(T) = v_e$ at the destination. For this purpose, it can be controlled by adjusting the acceleration $u(t) = F/m \in [-\hat{u}, +\hat{u}]$. The initial value problem for the equation of motion can be written as a first-order system, yielding

$$\begin{pmatrix} \dot{s}(t) \\ \dot{v}(t) \end{pmatrix} = \begin{pmatrix} v(t) \\ u(t) \end{pmatrix} \quad \text{with} \quad \begin{pmatrix} s(0) \\ v(0) \end{pmatrix} = \begin{pmatrix} s_0 \\ v_0 \end{pmatrix}. \tag{7.18}$$

Assuming a constant acceleration $\bar{u}$, the analytical solution of (7.18) is given by a polynomial of second degree. It reads

$$s(t; \bar{u}, s_0, v_0) = \tfrac{\bar{u}}{2} t^2 + v_0 t + s_0 \quad \text{and} \quad v(t; \bar{u}, v_0) = \bar{u} t + v_0. \tag{7.19}$$

The optimal control of the rocket car when minimizing time T follows the *bang-bang* control strategy for the acceleration u. At first the vehicle accelerates with maximum power $|\hat{u}|$ towards the destination and until it reaches the switching point $\hat{t} \in [0, T]$ from where on it accelerates with the same power into the opposite direction. For a closer look see, e.g. Macki and Strauss [70] and Steinbach [87, 88].

Multistage NLP Formulation of Finite Dimension

In a multiple shooting approach the time interval $[0, T]$ is split into $N - 1$ subintervals $[t_j, t_{j+1}]$ with $t_j = \frac{jT}{N}$. The state and control variables for stages $j \in \{0, \dots, N\}$ are $x_j = (s(t_j), v(t_j), T)^T$ and $u_j = u(t_j)$, respectively. Stage functions

$$g_j(x_{j-1}, u_{j-1}) = \begin{pmatrix} g_{j,1}(x_{j-1}, u_{j-1}) \\ g_{j,2}(x_{j-1}, u_{j-1}) \\ x_{j-1,3} \end{pmatrix}, \quad \text{for } j = 1, \dots, N, \tag{7.20}$$

describe the dynamics of the system. The solution of the IVP (7.18) on stage $j - 1$ for initial values s_{j-1}, v_{j-1} is represented by

$$\begin{pmatrix} g_{j,1}(x_{j-1}, u_{j-1}) \\ g_{j,2}(x_{j-1}, u_{j-1}) \end{pmatrix} = \begin{pmatrix} s(t_j; u_{j-1}, x_{j-1,1}, x_{j-1,2}) \\ v(t_j; u_{j-1}, x_{j-1,1}) \end{pmatrix}. \tag{7.21}$$

Incorporation boundary values at the last stage ($j = N$) the problem to solve reads

$$\min_{x \in \mathbb{R}^{3N}} \quad x_{N,3} = T \tag{7.22a}$$

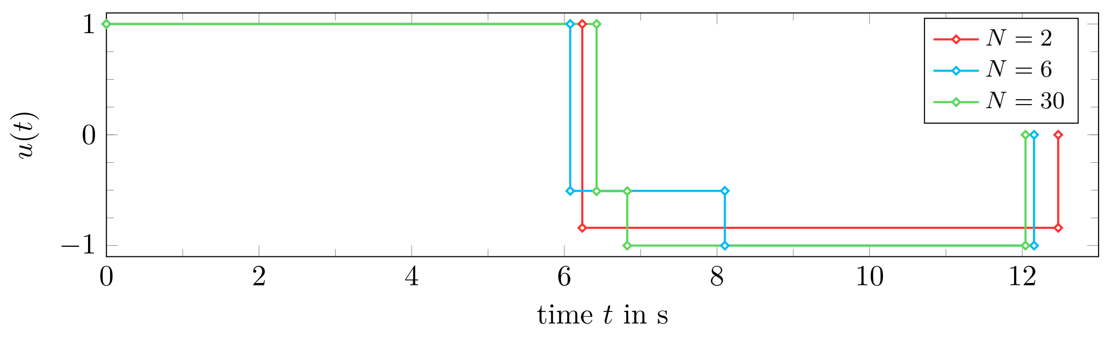


Figure 7.9: Computed optimal control trajectory of problem (7.22) for $s_0 = 0$, $v_0 = 0$ and $s_N = 42$, $v_N = 1$ for $N = 2, 6, 30$.

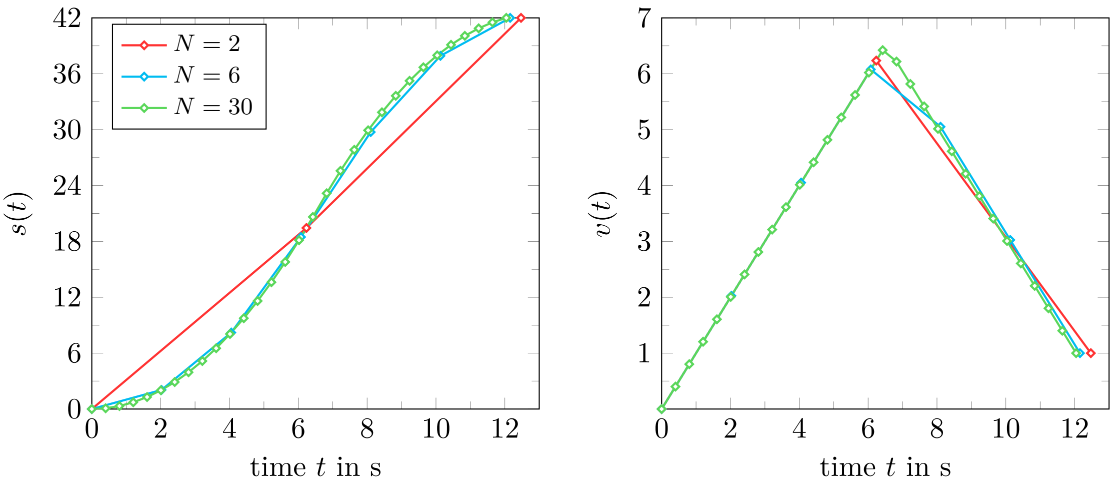


Figure 7.10: Computed optimal state trajectories of the car's position (left) and velocity (right) of problem (7.22) for $s_0 = v_0 = 0$ and $s_N = 42$, $v_N = 1$ for $N = 2, 6, 30$.

$$\begin{aligned} \text{s.t.} \quad & 0 = g_j(x_{j-1}, u_{j-1}) - x_j, && \text{for } j = 1, \ldots, N, && \text{(7.22b)} \\ & 0 = x_{0,0} - s_0, \quad 0 = x_{0,1} - v_0, && && \text{(7.22c)} \\ & 0 = x_{N,0} - s_e, \quad 0 = x_{N,1} - v_e, && && \text{(7.22d)} \\ & u_j \in [-1, 1], && \text{for } j = 0, \ldots, N. && \text{(7.22e)} \end{aligned}$$

Computational Results

Exemplary (7.22) is solved for $N = 2, 6, 30$. The rocket car starts standing, i.e. $v_0 = 0$, at $s_0 = 0$ and is driven to $s_e = 42$ with a final velocity of $v_e = 1$ in minimal time T_N. The computed optimal driving times depend on the number of stages. More precisely, a finer discretization offers a higher accuracy in the representation of the switching point $\hat{t}$. Solutions obtained are $T_2 = 12.471\,12\,\text{s}$, $T_6 = 12.092\,69\,\text{s}$ and $T_{30} = 12.042\,98\,\text{s}$. Figure 7.9 and 7.10 show the computed control and state trajectories.

Chapter 8

Conclusion and Outlook

In this thesis the generic seqential quadratic programming framework Clean::SQP and the elastic primal active-set method Clean::ASM are presented. It is in the main focus to allow warm start and preserve problem-specific structures in the formulation of independent subproblems on several levels. Maintaining sparsity the solution of self-contained problems is delegated to exchangeable subalgorithms. This allows the user to employ any specialized sub-solvers, e.g. for solving quadratic subproblems or KKT systems therein.

Filter SQP methods require the solution of closely related QPs in every iteration. The QP solver should be warm started using the information determined in the preceeding iteration. Because of that, Clean::ASM is predestined to be used within SQP because of its relaxation scheme that avoids a phase 1. Furthermore, it is capable to compute approximate solutions of inconsistent QPs making it usable in a feasibility restoration phase.

The presented active-set method uses projection techniques that preserve the NLP sparse structure leading to KKT systems that share a superstructure common in numerical optimization. Due to modifications in the workingset the KKT size changes when constraints are declared active or inactive. To avoid the refactorization of the changed KKT matrix in this context a Schur complement method is used to update already computed factorizations.

In order to illustrate the robustness and performance of Clean::ASM computational results on an established test set for convex quadratic programming by Maros and Mészáros are presented. The code of Clean::SQP is proved to be applicable and strong on the solution of hard to solve NLPs. This has been validated by solving optimization problems arising in real-life applications incorporating differential equation constraints. This also proofs the flexibility of Clean::ASM to be warm started from infeasible points as well as the algorithmic concept and design of the code.

Nevertheless, a number of improvements are possible. The presented slack relaxation scheme, especially ℓ_1, is sensitive to the magnitude of penalization of infeasibility. On the one hand feasibility may not be obtained if it is chosen too small but on the other hand large values for ρ imply KKT systems that are hard to solve with the required accuracy. As a remedy a constraint specific penalization could be implemented by introducing a vector of penalty parameters corresponding to relaxed constraints.

It is promising, that Clean::SQP is an efficient choice as the optimizer in *(nonlinear) model predictive control* ((N)MPC). NMPC is an optimization based method for feedback control of nonlinear systems with primary applications in stabilization and tracking problems. The fundamental idea is to utilize a model of the process in order to predict and optimize the future system behavior of a plant [59]. Figure 8.1 illustrates an open loop controller for iterative online optimization in MPC.

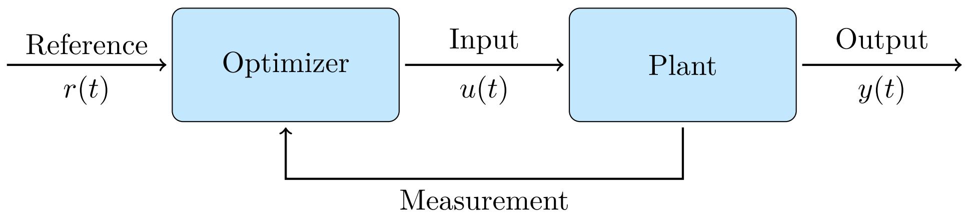


Figure 8.1: Open loop controller in model predictive control.

In online (N)MPC the sampling interval is short compared to the controller and plant dynamics such that the NLPs are closely related. Solutions in terms of future input and state trajectories can be time shifted one sampling period offering a good initialization for warm start purposes [57, 59]. Especially active-set information at time t can be forwarded to the QP solution algorithm to initialize the workingset of the primary QP subproblem in SQP of the NLP at time $t+1$. From the prediction horizon point of view, even complete control sequences and state trajectories can be used in moving horizon estimations (see Figure 8.2).

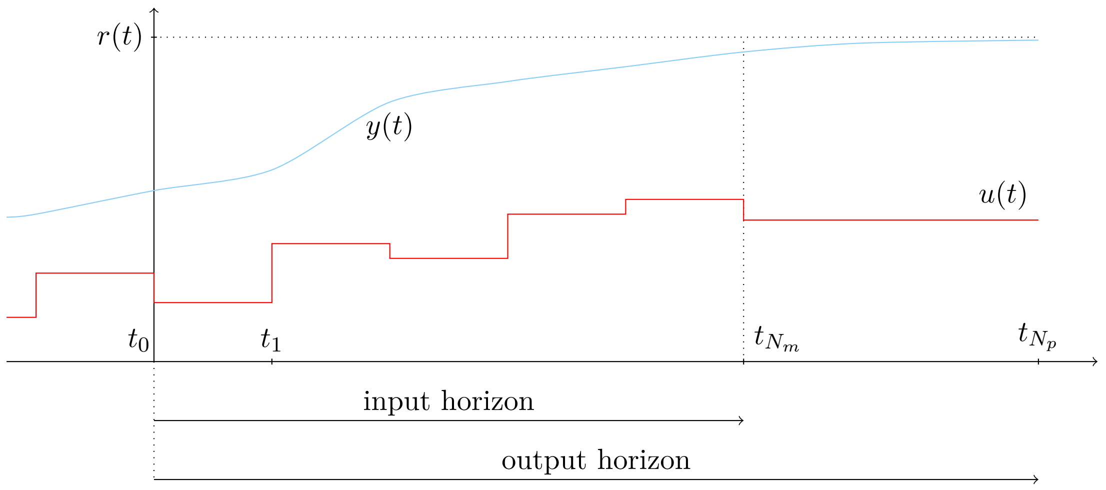


Figure 8.2: Illustration of an NMPC step at time t_0 in a moving horizont estimation.

Bibliography

[1] A. Alexandrescu. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM, Philadelphia, PA, 1992. `http://www.netlib.org/lapack`.

[3] G. Bader and P. Deuflhard. A semi-implicit midpoint rule for stiff systems of ordinary differential equations. *Numer. Math.*, 41:373–398, 1983.

[4] R. A. Bartlett and L. T. Biegler. rSQP++: an object-oriented framework for successive quadratic programming. In *Large-scale PDE-constrained optimization (Santa Fe, NM, 2001)*, volume 30 of *Lect. Notes Comput. Sci. Eng.*, pages 316–330. Springer, Berlin, 2003.

[5] L. Biegler. *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. Society for Industrial and Applied Mathematics, 2010.

[6] N. L. Boland. A dual-active-set algorithm for positive semi-definite quadratic programming. *Math. Programming*, 78(1, Ser. A):1–27, 1997.

[7] Boost. Generic Programming Techniques. `http://www.boost.org/community/generic_programming.html`.

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

[9] Bpmpd interior point solver. `http://www.sztaki.hu/~meszaros/bpmpd/`.

[10] C. G. Broyden, J. E. Dennis, and J. J. More. On the local and superlinear convergence of quasi-Newton methods. *J. Inst. Math. Appl.*, 12:223–245, 1973.

[11] R. Bulirsch and J. Stoer. Numerical treatment of ordinary differential equations by extrapolation methods. *Numer. Math.*, 8:1–13, 1966.

[12] R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. An algorithm for nonlinear optimization using linear programming and equality constrained subproblems. *Math. Program., Ser. B*, 100(1):27–48, May 2004.

[13] R. H. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In *Large Scale Nonlinear Optimization, 35–59, 2006*, pages 35–59. Springer Verlag, 2006.

[14] C. M. Chin. A global convergence theory of a filter line search method for nonlinear programming. *Numerical Optimization Report*, 2002.

[15] C. M. Chin. A local convergence theory of a filter line search method for nonlinear programming. *Numerical Optimization Report*, 2002.

[16] C. M. Chin and R. Fletcher. On the global convergence of an SLP-filter algorithm that takes EQP steps. *Math. Program.*, 96(1):161–177, 2003.

[17] A. R. Conn, N. I. M. Gould, and P. Toint. *Trust-Region Methods*. MPS-SIAM series on optimization. SIAM, 2000.

[18] J. O. Coplien. Curiously recurring template patterns. *C++ Rep.*, 7(2):24–27, Feb. 1995.

[19] J. E. Dennis and J. J. Moré. Quasi-Newton methods, motivation and theory. *SIAM Rev.*, 19(1):46–89, 1977.

[20] H.-J. Diekhoff, P. Lory, H.-J. Oberle, H. J. Pesch, P. Rentrop, and R. Seydel. Comparing routines for the numerical solution of initial value problems of ordinary differential equations in multiple shooting. *Numer. Math.*, 27:449–469, 1977.

[21] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2):201–213, 2002.

[22] A. S. Drud. CONOPT – a large-scale GRG code. *INFORMS J. Comput.*, 6(2):207–216, 1994.

[23] A. S. Drud. CONOPT: A system for large scale nonlinear optimization, reference manual for CONOPT subroutine library. Technical report, ARKI Consulting and Development A/S, Bagsvaerd, Denmark, 1996.

[24] M. Esser, S. Probst, and E. Baake. Partitioning, duality, and linkage disequilibria in the moran model with recombination. *Journal of Mathematical Biology*, 73(1):161–197, 2016.

[25] R. Fletcher. *Practical Methods of Optimization, Vol 2: Constrained Optimization*. Wiley, New York, 1981.

[26] R. Fletcher. *Practical methods of optimization*. A Wiley-Interscience Publication. John Wiley & Sons, Ltd., Chichester, second edition, 1987.

[27] R. Fletcher and E. S. de la Maza. Nonlinear programming and nonsmooth optimization by successive linear programming. *Math. Program.*, 43(3):235–256, 1989.

[28] R. Fletcher and S. Leyffer. A bundle filter method for nonsmooth nonlinear optimization. Numerical Analysis Report NA/195, University of Dundee, December 1999.

[29] R. Fletcher and S. Leyffer. *User manual for FilterSQP.* Dundee, DD1 4HN, Scotland, U.K., 1999.

[30] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Math. Program.*, 91:239–269, 2000.

[31] C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *J. Global Optim.*, 45(1):3–38, 2009.

[32] C. A. Floudas and V. Visweswaran. Quadratic optimization. In *Handbook of global optimization*, volume 2 of *Nonconvex Optim. Appl.*, pages 217–269. Kluwer Acad. Publ., Dordrecht, 1995.

[33] A. Forsgren, P. E. Gill, and E. Wong. Primal and dual active-set methods for convex quadratic programming. *Mathematical Programming*, 159(1):469–508, 2016.

[34] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[35] A. Geletu. Quadratic programming problems. Technical report, Ilmenau University of Technology, 2015.

[36] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Math. Comp.*, 28:505–535, 1974.

[37] P. E. Gill, L. O. Jay, M. W. Leonard, L. R. Petzold, and V. Sharma. An SQP method for the optimal control of large-scale dynamical systems. *J. Comput. Appl. Math.*, 120(1-2):197–213, 2000. SQP-based direct discretization methods for practical optimal control problems.

[38] P. E. Gill, W. Murray, and M. A. Saunders. *User's guide for QPOPT 1.0: a Fortran package for quadratic programming.* Stanford, CA, 1995.

[39] P. E. Gill, W. Murray, and M. A. Saunders. *User's guide for SQOPT Version 7: Software for large-scale linear and quadratic programming.* San Diego, La Jolla, CA, numerical analysis report 06-1 edition, 2006.

[40] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Maintaining *LU* factors of a general sparse matrix. *Linear Algebra Appl.*, 88/89:239–270, 1987.

[41] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. A practical anti-cycling procedure for linearly constrained optimization. *Math. Programming*, 45(3, (Ser. B)):437–474, 1989.

[42] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Inertia-controlling methods for general quadratic programming. *SIAM Rev.*, 33(1):1–36, 1991.

[43] P. E. Gill, W. Murray, and M. S. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. Technical Report NA 97-2, Department of Mathematics, University of California, 1997.

[44] P. E. Gill, W. Murray, and M. S. Saunders. User's guide for SNOPT 5.3: A Fortran package for large-scale nonlinear programming. Technical Report NA 97-5, Department of Mathematics, University of California, 1997.

[45] P. E. Gill, W. Murray, and M. S. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM J. Optim.*, 12(4):979–1006, 2002.

[46] P. E. Gill, M. A. Saunders, and E. Wong. On the performance of SQP methods for nonlinear optimization. In *Modeling and optimization: theory and applications*, volume 147 of *Springer Proc. Math. Stat.*, pages 95–123. Springer, Cham, 2015.

[47] P. E. Gill and E. Wong. Sequential quadratic programming methods. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 147–224. Springer New York, 2012.

[48] P. E. Gill and E. Wong. Methods for convex and general quadratic programming. *Math. Program. Comput.*, 7(1):71–112, 2015.

[49] M. A. Gómez. A null-space method for computing the search direction in the general inertia-controlling method for dense quadratic programming. *European J. Oper. Res.*, 161(3):655–662, 2005.

[50] N. Gould, D. Orban, and P. Toint. Numerical methods for large-scale nonlinear optimization. *Acta Numer.*, 14:299–361, 2005.

[51] N. I. M. Gould, D. Orban, and P. L. Toint. CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Trans. Math. Softw.*, 29(4):373–394, Dec. 2003.

[52] N. I. M. Gould, D. Orban, and P. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.*, 60(3):545–557, 2015.

[53] N. I. M. Gould and P. L. Toint. An iterative working-set method for large-scale nonconvex quadratic programming. *Appl. Numer. Math.*, 43(1-2):109–128, 2002. 19th Dundee Biennial Conference on Numerical Analysis (2001).

[54] N. I. M. Gould and P. L. Toint. A quadratic programming bibliography. Technical report, RAL Numerical Analysis Group Internal Report 2000–1, 2012.

[55] N. I. M. Gould and P. T. Toint. *Numerical Methods for Large-Scale Non-Convex Quadratic Programming*, chapter 149-179, pages 149–179. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2002.

[56] W. B. Gragg. On extrapolation algorithms for ordinary initial value problems. *J. Soc. Indust. Appl. Math. Ser. B Numer. Anal.*, 2:384–403, 1965.

[57] A. Grancharova and T. A. Johansen. *Explicit nonlinear model predictive control*, volume 429 of *Lecture Notes in Control and Information Sciences*. Springer, Heidelberg, 2012. Theory and applications.

[58] M. Grötschel, S. O. Krumke, and J. Rambau, editors. *Online Optimization of Large Scale Systems*. Springer, Berlin, 2001.

[59] L. Grüne and J. Pannek. *Nonlinear model predictive control*. Communications and Control Engineering Series. Springer, London, 2011. Theory and algorithms.

[60] I. G. Hernández and M. C. Steinbach. A multistage stochastic programming approach in real-time process control. ZIB Report ZR-01-05, Zuse Institute Berlin, Mar. 2001. Appeared in [58].

[61] The HSL Mathematical Software Library. *HSL MA27 Package Specification*, Mar. 2003.

[62] The HSL Mathematical Software Library. *HSL MA57 Package Specification*, Sept. 2016.

[63] J. Hübner. *Distributed Algorithms for Nonlinear Tree-Sparse Problems*. PhD thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2016.

[64] A. Hunt and D. Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[65] L. G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[66] C. Kirches, H. G. Bock, J. P. Schlöder, and S. Sager. Block-structured quadratic programming for the direct multiple shooting method for optimal control. *Optim. Methods Softw.*, 26(2):239–257, 2011.

[67] C. Lawson, R. Hanson, D. Kincaid, and F. Krough. Basic linear algebra subprograms for FORTRAN usage. *ACM Trans. Math. Software*, 5:308–325, 1979.

[68] S. Leyffer and A. Mahajan. Foundations of constrained optimization. *Wiley Encyclopedia of Operations Research and Management Science*, 2011.

[69] X. Liu, Y. X. Sun, and W. H. Wang. Stabilizing control of robustness for systems with maximum uncertain parameters—a quadratic programming approach. *Control Theory Appl.*, 16(5):729–732, 1999.

[70] J. W. Macki and A. Strauss. *Introduction to optimal control theory.* Springer-Verlag, New York-Berlin, 1982. Undergraduate Texts in Mathematics.

[71] C. Maes. *A Regularized Active-set Method for Sparse Convex Quadratic Programming.* PhD thesis, Stanford University, 2010.

[72] I. Maros and C. Mészáros. CUTEr version of the maros and mészáros quadratic programming test problem set. `http://www.numerical.rl.ac.uk/cuter-www/Problems/marmes.html`.

[73] I. Maros and C. Mészáros. A repository of convex quadratic programming problems. *Optim. Methods Softw.*, 11/12(1-4):671–681, 1999. Interior point methods.

[74] A. D. Martin. Mathematical programming of portfolio selections. *Management Science*, 1(2):152–166, 1955.

[75] B. A. McCarl, H. Moskowitz, and H. Furtan. Quadratic programming applications. *The International Journal of Management Science*, 5:43–55, 1977.

[76] C. Mészáros. The BPMPD interior point solver for convex quadratic problems. *Optim. Methods Softw.*, 11/12(1-4):431–449, 1999. Interior point methods.

[77] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer, Berlin, 2nd edition, 2006.

[78] M. J. D. Powell. The convergence of variable metric methods for nonlinearly constrained optimization calculations. In O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, editors, *Nonlinear Programming 3*, pages 27–61. Academic Press, 1978.

[79] S. Probst. Numerische berechnung genetischer rekombinationswahrscheinlichkeiten. Master's thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2012.

[80] M. Saunders. Geometric-mean scaling for sparse matrices. `https://github.com/mxsaunders/pdco/blob/master/code/gmscale.m`, 1996-2009.

[81] M. A. Saunders. Fortran software for updating dense lu factors, 2000. LUMOD.

[82] M. Schmidt. *A Generic Interior-Point Framework for Nonsmooth and Complementarity Constrained Nonlinear Optimization.* PhD thesis, Gottfried Wilhelm Leibniz Universität Hannover, 2013.

[83] M. C. Steinbach. *Fast Recursive SQP Methods for Large-Scale Optimal Control Problems.* Ph. D. dissertation, Universität Heidelberg, 1995.

[84] M. C. Steinbach. Recursive direct algorithms for multistage stochastic programs in financial engineering. ZIB Preprint SC-98-23, Zuse Institute Berlin, 1998.

[85] M. C. Steinbach. Recursive direct optimization and successive refinement in multistage stochastic programs. ZIB Preprint SC-98-27, Zuse Institute Berlin, 1998.

[86] M. C. Steinbach. Hierarchical sparsity in multistage convex stochastic programs. ZIB Report ZR-00-15, Zuse Institute Berlin, May 2000. Appeared in [93].

[87] M. C. Steinbach. Optimierung bei Differentialgleichungen, 2009. Lecture notes.

[88] M. C. Steinbach. Dynamische Optimierung, 2017. Lecture notes.

[89] M. C. Steinbach, H. G. Bock, G. V. Kostin, and R. W. Longman. Mathematical optimization in robotics: Towards automated high speed motion planning. *Surv. Math. Ind.*, 7(4):303–340, 1998.

[90] A. Stepanov and M. Lee. The standard template library. Technical report, WG21/N0482, ISO Programming Language C++ Project, 1994.

[91] Boost C++ Libraries. `http://www.boost.org/`, 1998-2018.

[92] ISO C++ Standard. `https://isocpp.org/std`.

[93] S. P. Uryasev and P. M. Pardalos, editors. *Stochastic Optimization: Algorithms and Applications.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.

[94] D. van Heesch. Doxygen. `http://www.stack.nl/~dimitri/doxygen/`, 1997-2018.

[95] S. A. Vavasis. Quadratic programming is in NP. *Inform. Process. Lett.*, 36(2):73–77, 1990.

[96] A. Wächter and L. T. Biegler. Global and local convergence of line search filter methods for nonlinear programming. Technical Report CAPD B-01-09, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pittsburgh, PA 015213, Philadelphia, USA, Aug. 2001.

[97] A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM J. on Optimization*, 16(1):1–31, May 2005.

[98] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106(1):25–57, 2006.

[99] Y. Wang and S. Boyd. Fast Model Predictive Control Using Online Optimization. *Control Systems Technology, IEEE Transactions on*, 18(2):267–278, 2010.

[100] R. Wilson. *A Simplicial Algorithm for Concave Programming.* PhD thesis, Harvard University, 1963.

[101] E. Wong. *Active-Set Methods for Quadratic Programming.* PhD thesis, Department of Mathematics, University of California, San Diego, June 2011.

[102] A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization*, 16(1):32–48, 2005.

[103] E. Yourdon and L. L. Constantine. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1st edition, 1979.

Lebenslauf – M.Sc. Daniel Rose

Angaben zur Person

Geburtsdatum:	12. Februar 1986
Geburtsort:	Gehrden
Staatsangehörigkeit:	deutsch
Adresse:	Köllerweg 22, 32760 Detmold
E-Mail:	rose@ifam.uni-hannover.de

Bildungsweg

07/1996 – 06/2003	Sophie Scholl Gesamtschule, *Wennigsen*, Gymnasialzweig
07/2003 – 06/2006	Marie-Curie-Schule, *Ronnenberg*, Oberstufe
06/2006	Abitur, *Marie-Curie-Schule Ronnenberg*
10/2006 – 09/2009	Bachelorstudium Mathematik, *Leibniz Universität Hannover*, Schwerpunkt Angewandte Mathematik Anwendungsfach Betriebswirtschaftslehre
06/2006	Bachelorarbeit: Modellierung und Approximation von Überläufen mit Rückfluss in Abwassersystemen, *Institut für Angewandte Mathematik* Unter der Betreuung von Prof. Dr. Marc C. Steinbach
10/2009 – 03/2012	Masterstudium Mathematik, *Leibniz Universität Hannover*, Schwerpunkt Angewandte Mathematik Anwendungsfach Meteorologie
12/2011	Masterarbeit: Numerische Berechnung akustischer Eigenschwingungen, *Institut für Angewandte Mathematik* Unter der Betreuung von Prof. Dr. Marc C. Steinbach
04/2012 – 12/2017	Promotionsstudium, *Leibniz Universität Hannover, Institut für Angewandte Mathematik*, Arbeitsgruppe Algorithmische Optimierung Unter der Betreuung von Prof. Dr. Marc C. Steinbach

Beruflicher Werdegang

03/2008 – 03/2012	Studentische Hilfskraft, *Leibniz Universität Hannover, Institut für Angewandte Mathematik*, Mitarbeit in Forschung und Lehre Programmierarbeiten sowie Tutorien zu den Lehrveranstaltungen Algorithmisches Programmieren und Mathematik für Ingenieure
09/2012 – 06/2013	Wissenschaftlicher Mitarbeiter, *Leibniz Universität Hannover, Institut für Angewandte Mathematik*, Projektarbeit, ForNe: Optimization of Gas Transport Networks Optimierung von Verdichertstationen in Gastransportnetzen
04/2012 – 03/2018	Wissenschaftlicher Mitarbeiter, *Leibniz Universität Hannover, Institut für Angewandte Mathematik*, Forschung und Lehre Betreuung der Veranstaltungen Algorithmisches Programmieren, Nichtlineare Optimierung, Mathematik für Wirtschaftswissenschaftler und Ingenieure

Publikationen

1. Daniel Rose, Martin Schmidt, Marc C. Steinbach, Bernhard M. Willert, Computational Optimization of Gas Compressor Stations: MINLP Models vs. Continuous Reformulations, Mathematical Methods of Operations Research, 2016.