Smoothing-Type Methods for Linear Programs

Dissertation
zur Erlangung des Doktorgrades
des Fachbereichs Mathematik
der Universität Hamburg

vorgelegt von
Stephan Engelke
aus Hamburg

Hamburg
2001

Als Dissertation angenommen vom Fachbereich
Mathematik der Universität Hamburg

auf Grund der Gutachten von PD Dr. Christian Kanzow
und Prof. Dr. Carl Geiger

Hamburg, 6. Juli 2001

Prof. Dr. Rainer Hass
Dekan des Fachbereichs Mathematik

Contents

1 Introduction

Linear optimization problems, also known as linear programs, play an important role in many different areas, such as mathematics, economics, finance and engineering. There probably is no other type of optimization problem which is solved as frequently as a linear program in these application. Even when the situation being modeled is actually non-linear, a linear model is favored in many cases, because of the highly sophisticated software available for the solution of such a model and also because uncertainties in the model and the data often make it impractical to construct a more elaborate non-linear model.

For a long time, the simplex method introduced by George Dantzig [18] in the mid 1940s was the only method available to solve linear programs; it usually performs very well in practice, even for large scale problems.

In 1972, however, Klee and Minty [38] showed that in theory the number of iterations required by the simplex method to solve a problem could depend exponentially on the dimension of the problem, even though the average performance is typically much better, see in particular Borgwardt [3].

This observation prompted many researchers to find an algorithm which has a better worst-case performance than the implex method; in fact one was interested in finding algorithms where the number of iterations required to achieve a solution depends polynomially on the problem's dimension.

The first polynomial-time algorithm was found by Khachiyan [37] in 1979. Even though this algorithm has a polynomial complexity bound, it turned out to be less successful in practice than the simplex method. In 1984 Karmarkar [36] published another algorithm for the solution of linear programs. Karmarkar showed that his method has a polynomial complexity bound and claimed that his method is also significantly faster than the simplex method for many practical problems.

Even though Karmarkar's statement regarding his algorithm's performance may have been rather enthusiastic, his article may be considered the starting point of the field of interior-point methods, these methods turned into the focus of many researchers in the field of mathematical programming.

A few years ago in the field of complementarity problems another idea was born. In this area it is a common technique to reformulate a complementarity problem's optimality conditions into a system of equations using so-called NCP-functions and then apply Newton's method to this system. The NCP-functions employed here often are non-smooth, requiring the application of non-smooth analysis and the use of a non-smooth version of Newton's method. Kanzow was the first to take a slightly different approach which, though not new in itself, was applied to complementarity problems for the first time, see, e. g., [32]. The so-called smoothing or continuation methods introduce a perturbation to the NCP-function in order to turn the formerly non-smooth function into a differentiable one. When applied to complementarity problems, these continuation methods perform quite well numerically

Smoothing methods typically perturb both sides of the system of equations to which Newton's method is applied. Chen, Qi and Sun [14] introduced another related approach to the non-differentiability problem of the NCP-functions. They used a variant of a smoothing method which is now called a *Jacobian smoothing method*, since only the Jacobian on the left hand side of the Newton equation is actually perturbed. This appears to be a quite reasonable approach, since it is only the Jacobian which poses a problem in this context of non-differentiability. The resulting linear system is actually more similar to the one that is to be solved in the first place.

In this thesis the smoothing and the Jacobian smoothing approach are applied to linear programs for the first time. Since a linear optimization problem is a special case of a linear complementarity problem, it appears to be a logical step to take in the area of linear optimization.

In specializing these methods and taking into account the special structure of the linear optimization problem, one obtains algorithms which perform quite well numerically. Also for the first time, combinations of Jacobian smoothing and regular smoothing methods in form of predictor-corrector algorithms are presented and examined in detail. The numerical results produced by these new methods are comparable to those of modern-day interior-point codes used to solve linear programs. The material on the new methods presented here has essentially been taken from the articles [20, 21, 22, 23].

This thesis is structured as follows: After presenting the notation which will be used here, some necessary background information is given in Chapter 3. In Chapter 4 a short overview over interior-point methods is followed by a presentation of Mehrotra's predictor-corrector Algorithm [43], the currently most successful primal-dual method for solving linear optimization problems. General ideas of smoothing-type methods are conveyed in Chapter 5 and a basic Jacobian smoothing method is presented in Chapter 6. Numerical results for this algorithm are given. Chapter 7 introduces two slightly different approaches to regular smoothing methods and gives global convergence results. Chapter 8 introduces the new locally quadratically convergent combinations of Jacobian smoothing and regular smoothing methods. Numerical results are also presented for these methods and compared to those produced by an interior-point code.

Acknowledgements

The thesis and the software which has been developed during the research phase have been influenced by many people along the way. In this place, I'd like to thank all of them for many comments and suggestions.

Special thanks go to Christian Kanzow for help, discussions, many valuable insights and comments during the preparation of the thesis and the implementation of the software and Carl Geiger for his interested and extremely speedy review of the thesis.

Additional thanks go to Helge Baumann for many helpful comments and proof-reading and to Roland Weber for some magic.

2 Notation

The notation used here is standard in most places.

Norms

All vector norms are Euclidian norms; for matrices the corresponding spectral norm is used, which is, in particular, consistent with the Euclidian vector norm in the sense that $\|Az\| \le \|A\| \cdot \|z\|$ for all matrices A and all vectors z of appropriate dimensions. For a matrix $A \in \mathbb{R}^{n\times n}$, $\|A\|_F := \left(\sum_{i,j=1}^n a_{ij}^2\right)^{1/2}$ denotes the Frobenius-norm of a matrix.

Eigenvalues and Matrices

The notations $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ are used for the minimal and maximal eigenvalue of a symmetric matrix $A \in \mathbb{R}^{n\times n}$. We denote the Euclidian distance of a vector z to the non-empty set S by $\operatorname{dist}_S(z) := \inf_{w\in S} \|w - z\|$.

As common in interior-point literature, a matrix $X := \operatorname{diag}(x_1, \dots, x_n) \in \mathbb{R}^{n\times n}$ denotes the diagonal matrix made up of the components of $x \in \mathbb{R}^n$.

Vectors and Index Sets

If $x \in \mathbb{R}^n$ then the subscript i in x_i will indicate the i-th component of x. The superscript k in x^k is used to indicate that this x is the k-th iterate of a sequence $\{x^k\} \subseteq \mathbb{R}^n$.

Quite often a triple of the form $w = (x^T, \lambda^T, s^T)^T$ will be considered, were $x \in \mathbb{R}^n$, $\lambda \in \mathbb{R}^m$, and $s \in \mathbb{R}^n$. Then, of course w is a vector in $\mathbb{R}^{n+m+n}$. In order to simplify the notation, we will write $w = (x, \lambda, s)$ instead of the mathematically correct formula $w = (x^T, \lambda^T, s^T)^T$.

If z is an r-dimensional vector and $I \subseteq \{1, 2, \dots, r\}$ any given subset, then r_I denotes the subvector of the components r_i for $i \in I$. Similarly, if A denotes an $r \times s$-matrix and $I \subseteq \{1, 2, \dots, r\}$, $J \subseteq \{1, 2, \dots, s\}$ are two given subsets, A_{IJ} denotes the submatrix with elements a_{ij} for $i \in I$ and $j \in J$. For any set I, $|I|$ denotes the number of elements in this set.

Furthermore, for any given vector z, we write z_+ for the projection of z on the non-negative orthant, i. e., the components of z_+ are given by $\max\{0, z_i\}$ for all $i \in \{1, \dots, n\}$.

If $x, y \in \mathbb{R}^n$ are any given vectors satisfying the inequality $x_i \ge y_i$, for all indices $i = 1, \dots, n$, we simply write $x \ge y$.

NCP- and Smoothing Functions

NCP-functions are denoted by $\varphi(a, b)$, where a and b are scalars. A vector of NCP-functions is denoted by $\phi(x, s) := (\dots, \varphi(x_i, s_i), \dots)^{\mathrm{T}} \in \mathbb{R}^n$, with $x, s \in \mathbb{R}^n$. The Matrix $D_a \in \mathbb{R}^{n\times n} := \mathrm{diag}(\dots, \partial\varphi(x_i, s_i)/\partial a, \dots)$ denotes the diagonal matrix whose components are the partial derivatives of $\phi_i(x, s)$ for the first variable, D_b is defined similarly.

A smoothing function will be either denoted by $\varphi_\tau(a, b)$ or by $\theta_i(a, b, \tau)$ where a, b, τ are scalars with $\tau \geq 0$. The use of φ or θ depends on the view taken of the smoothing parameter τ. Similar to $\phi(x, y)$, $\theta(x, y, \tau) = (\dots, \theta_i(x_i, y_i, \tau), \dots)$ and $\phi_\tau(x, s) = (\dots, \varphi_\tau(x_i, s_i), \dots)$ denote vectors of smoothing functions for $x, s \in \mathbb{R}^n$. The diagonal matrices $D_{a,\tau}$ and $D_{b,\tau}$ are defined analogously to the Matrices D_a and D_b, respectively.

Order notation

At certain times the following two varieties of order notation, $O(\cdot)$ and $o(\cdot)$, will be used. If $\{\eta_k\}$ and $\{\nu_k\}$ are two given, non-negative infinite sequences of scalars, we say that

$$\eta_k = O(\nu_k)$$

if there is a positive constant C such that

$$\eta_k \leq C\nu_k$$

for all k sufficiently large. That is, the elements of the first sequence $\{\eta_k\}$ do not become too much larger than the elements of the second sequence $\{\nu_k\}$. We write

$$\eta_k = o(\nu_k)$$

if there is another non-negative sequence $\{\beta_k\}$ such that

$$\eta_k \leq \beta_k \nu_k, \;\; \beta_k \longrightarrow 0.$$

3 Theoretical Background

3.1 Linear Programming

The fundamental properties of a linear programming problem are:

(1) a vector of real variables, whose optimal values are found by solving the problem,

(2) a linear objective function,

(3) linear constraints, both inequalities and equalities.

We will use one particular formulation of a linear program, the so called *canonical form*, to describe and analyze the algorithms presented in this thesis. This form is:

$$\min \; c^{\mathrm{T}}x \quad \text{subject to} \quad Ax = b,\; x \geq 0, \tag{3.1}$$

where c and x are vectors in $\mathbb{R}^n$, b is a vector in $\mathbb{R}^m$ and A is an $m \times n$ matrix with real entries. If x satisfies the constraints $Ax = b$ and $x \geq 0$, we call it a *feasible point*. The set of feasible points is called the *feasible set*.

Note that any linear program can be converted into the canonical form.

Associated with any linear program is a so-called *dual* program, which consists of the same data differently arranged. The dual problem for (3.1) is

$$\max \; b^{\mathrm{T}}\lambda \quad \text{subject to} \quad A^{\mathrm{T}}\lambda + s = c,\; s \geq 0, \tag{3.2}$$

where $\lambda \in \mathbb{R}^m$ and $s \in \mathbb{R}^m$ are the problem's variables and $A \in \mathbb{R}^{m\times n}$, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ are the given data. Components of the vector λ are called *dual variables*, components of s are named *dual slacks* or *dual slack variables*.

The dual problem (3.2) can of course be stated without the slack variables, but these variables will turn out to be convenient in the development of the algorithms. In this context, problem 3.1 is called the *primal* problem. Together the two problems are often referred to as the *primal-dual pair*.

3.2 Duality

The following results further define the relationship between the primal and the dual linear programs, (3.1) and (3.2), respectively. The application of these results leads us to optimality conditions for a linear program. Proofs for all of the results presented in this section can be found in the book by Wright [56].

There are two major duality results which will be needed, a weak and a strong one.

Proposition 3.1 (Weak Duality) *Let $x \in \mathbb{R}^n$ be a feasible point of the primal linear program* (3.1)*, and let $(\lambda, s) \in \mathbb{R}^m \times \mathbb{R}^n$ be a feasible point of the dual program* (3.2)*. Then*

$$b^{\mathrm{T}}\lambda \leq c^{\mathrm{T}}x.$$

Corollary 3.2 *Let $x \in \mathbb{R}^n$ be a feasible point of the primal program* (3.1)*, let $(\lambda, s) \in \mathbb{R}^m \times \mathbb{R}^n$ be a feasible point of the dual program* (3.2)*, and assume that $c^{\mathrm{T}}x = b^{\mathrm{T}}\lambda$. Then x is a solution of the primal problem* (3.1) *and (λ, s) is a solution of the dual problem* (3.2)

It is already known from the weak duality result in Proposition 3.1 that

$$c^{\mathrm{T}}x - b^{\mathrm{T}}\lambda \geq 0$$

holds for all primal feasible points $x \in \mathbb{R}^n$ and all dual feasible points $\lambda \in \mathbb{R}^m$. (The variable λ is called dual feasible if there exists a vector s of slack variables such that the pair (λ, s) is dual feasible.) The term

$$c^{\mathrm{T}}x - b^{\mathrm{T}}\lambda$$

is called the *duality gap*. We also know from Corollary 3.2 that we have a pair of primal and dual solutions if there is no duality gap, i. e., if $c^{\mathrm{T}}x - b^{\mathrm{T}}\lambda = 0$.

Next we show that the reverse is true as well, i. e., if we have a pair of primal and dual solutions, then there is no duality gap. Additionally, the result states that the primal problem has a solution if and only if the dual one has one, too.

Theorem 3.3 (Strong Duality) *The primal problem* (3.1) *has a solution x if and only if the dual problem* (3.2) *has an optimal solution (λ, s). In this case, there is no duality gap, i. e., $c^{\mathrm{T}}x = b^{\mathrm{T}}\lambda$.*

While Theorem 3.3 says that the primal problem has an optimal solution if and only if the dual program has an optimal solution, it is still unclear under which assumptions these problems have any solutions at all. Theorem 3.5 gives a simple condition, based on the following definition.

Definition 3.4 *Consider the primal-dual pair of linear programs* (3.1) *and* (3.2)*. Define the* primal-dual feasible set

$$\mathcal{F} := \left\{(x, \lambda, s) \;\middle|\; Ax = b,\;\; A^{\mathrm{T}}\lambda + s = c,\;\; x \geq 0,\;\; s \geq 0\right\}.$$

Also define the primal-dual strictly feasible set

$$\mathcal{F}^0 := \left\{(x, \lambda, s) \;\middle|\; Ax = b,\;\; A^{\mathrm{T}}\lambda + s = c,\;\; x > 0,\;\; s > 0\right\}.$$

Theorem 3.5 (Existence of a Solution) *If the primal and the dual linear programs are feasible, (i. e., if the primal-dual feasible set $\mathcal{F}$ is non-empty), then both have optimal solutions.*

3.3 Optimality conditions

The duality theory in the previous section allows us to derive optimality conditions for a linear program. These conditions must be satisfied by a solution of the problem. They can also be stated as special cases of the optimality conditions for general constrained optimization problems, known as the *Karush-Kuhn-Tucker conditions* (or *KKT conditions*).

In this section, two theorems are presented which give information on the optimality conditions for linearly constrained optimization problems and on the relationship between these conditions and so-called convex programming problems. We will obtain the optimality conditions for the linear program (3.1) by specializing these theorems.

Consider a not necessarily linear optimization problem with linear constraints of the form

$$\begin{array}{ll} \min & f(x) \\ \text{s.t.} & g_i(x) \geq 0, \quad i = 1, \dots, m, \\ & h_j(x) = 0, \quad j = 1, \dots, p, \end{array} \tag{3.3}$$

where $f\colon \mathbb{R}^n \longrightarrow \mathbb{R}$, and the constraints $g_i\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ and $h_j\colon \mathbb{R}^n \longrightarrow \mathbb{R}$ are linear functions.

The following result gives the necessary optimality conditions for the above optimization problem. A proof can be found in the book by Wright [56].

Theorem 3.6 *Consider the optimization problem* (3.3) *with a continuously differentiable objective function f and linear functions g_i and h_j. Assume that x is a solution of problem* (3.3). *Then there are vectors λ and μ such that the following conditions hold*

$$\begin{aligned} \nabla f(x) - \sum_{i=1}^{m} \lambda_i \nabla g_i(x) + \sum_{j=1}^{p} \mu_j \nabla h_j(x) &= 0 \\ h(x) &= 0 \\ \lambda \geq 0, \;\; g(x) \geq 0, \;\; \lambda^{\mathrm{T}} g(x) &= 0. \end{aligned} \tag{3.4}$$

Definition 3.7

(1) The conditions (3.4) *are called the* Karush-Kuhn-Tucker conditions (KKT conditions) *of the optimization problem* (3.3).

(2) Every vector $(x, \lambda, \mu) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p$, which satisfies the KKT conditions, is called a Karush-Kuhn-Tucker point (KKT point)*. The components of λ and μ are called* Lagrange multipliers*.*

To prove that a local minimizer of the non-linear problem (3.3) is a KKT point another condition, a so-called *constraint qualification*, has to be met. Since linear problems do not require any further qualifications (since the linearity is such a qualification already) and since non-linear problems are beyond the scope of this discussion, constraint qualifications will not be investigated further.

Note that problem (3.3) is a *convex programming* problem. Also note that a linear program is a special case of a convex programming problem. For such convex optimization problems, and thus linear programs, the local and global solutions are related by the following necessary optimality conditions:

Theorem 3.8

(1) If problem (3.3) *is a convex programming problem with a local solution x^*, then the KKT conditions are* sufficient *for x^* to be a global solution. That is, if Lagrange multipliers λ and μ exist, such that the conditions* (3.4) *are satisfied, then x^* is a global solution of the optimization problem* (3.3)*.*

(2) If, in addition, the function f is strictly convex on its feasible region, then any local solution is a uniquely defined global solution.

For a proof of this result see Wright [56].

The optimality conditions of problem (3.1) are obtained by specializing Theorems 3.6 and 3.8 and by taking into account that a linear program is also a convex program, since linear functions are convex (and also concave). Note that these conditions are both necessary *and* sufficient in the case of linear programs.

Corollary 3.9 (Optimality Conditions) *The vector x^* is a solution of the linear program* (3.1) *if and only if there exist vectors $s^* \in \mathbb{R}^n$ and $\lambda^* \in \mathbb{R}^m$ for which the following conditions hold for $(x, \lambda, s) = (x^*, \lambda^*, s^*)$:*

$$\begin{aligned} A^{\mathrm{T}}\lambda + s &= c \\ Ax &= b \\ x_i \geq 0,\ \ s_i \geq 0,\ \ x_i s_i &= 0 \quad i = 1, \dots, n. \end{aligned} \tag{3.5}$$

Note the choice of variable names: The Lagrange multipliers' names λ and s have deliberately been chosen equal to the variable names of the dual problem (3.2). Applying Theorems 3.6 and 3.8 to the dual problem (3.2) also yields the conditions stated in (3.5), i. e., the primal and the dual linear programs share the same optimality conditions. This is one characteristic of a primal-dual pair.

Closer examination of the conditions (3.5) derived from problems (3.1) and (3.2) shows that a vector (x^*, λ^*, s^*) solves the system (3.5) if and only if x^* solves the primal problem (3.1) and (λ^*, s^*) solves the dual problem (3.2). The vector (x^*, λ^*, s^*) is called the *primal-dual solution.*

Definition 3.10 *A solution (x^*, λ^*, s^*) of* (3.5) *is called* strictly complementary, *if $x^* + s^* > 0$, i. e., if there exists no index $i \in \{1, \dots, n\}$ such that $x_i^* = s_i^* = 0$.*

3.4 Solution Sets, Partitions and the Goldman-Tucker Theorem

Definition 3.11 *Denote the solution set of the optimality conditions* (3.5) *by S:*

$$S := \{(x,\lambda,s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \mid (x,\lambda,s) \text{ solves } (3.5)\}. \tag{3.6}$$

Furthermore, let S_P and S_D be the solution sets of the corresponding primal and dual linear programs:

$$S_P := \{x \mid x \text{ solves the primal problem } (3.1)\},$$
$$S_D := \{(\lambda,s) \mid (\lambda,s) \text{ solves the dual problem } (3.2)\}.$$

Note that

$$S = S_P \times S_D,$$

i. e., S_P and S_D are the projections of S onto the x- and (λ,s)-spaces, respectively.

Looking at the optimality conditions (3.5), we know that for every solution (x^*,λ^*,s^*) of these conditions $x_i^* = 0$ and/or $s_i^* = 0$ for all $i = 1,\dots,n$ holds. Now we can define two index sets $\mathcal{B}$ and $\mathcal{N}$ as follows:

$$\begin{aligned}\mathcal{B} &:= \{i \in \{1,\dots,n\} \mid x_i > 0 \text{ for some } x \in S_P\},\\ \mathcal{N} &:= \{i \in \{1,\dots,n\} \mid s_i > 0 \text{ for some } (\lambda,s) \in S_D\},\end{aligned} \tag{3.7}$$

In fact, $\mathcal{B}$ and $\mathcal{N}$ form a partition of the index set $\{1,\dots,n\}$; every $i = 1,\dots,n$ belongs to either $\mathcal{B}$ or $\mathcal{N}$, but not both. It is easy enough to show half of this result, namely, that $\mathcal{B}$ and $\mathcal{N}$ are disjoint. If there happened to be an index i that belonged to both sets, then from (3.7) there would be a primal solution x^* and a dual solution (λ^*,s^*) such that $x_i^* > 0$ and $s_i^* > 0$. But then we would have $x_i^* s_i^* > 0$, which contradicts the complementarity condition in (3.5). Hence, $\mathcal{B} \cap \mathcal{N} = \emptyset$.

The reverse result $\mathcal{B} \cup \mathcal{N} = \{1,\dots,n\}$ is known as the Goldman-Tucker Theorem [28].

Theorem 3.12 (Goldman-Tucker) *It holds that $\mathcal{B} \cup \mathcal{N} = \{1,\dots,n\}$. That is, there exists at least one primal solution $x^* \in S_P$ and one dual solution $(\lambda^*,s^*) \in S_D$ such that $x_i^* + s_i^* > 0$ for all $i = 1,\dots,n$.*

Proof. [28] or [56] ■

Next we state some simple properties of the two index sets defined in (3.7).

Lemma 3.13 *The following statements hold:*

(1) *We have $x_{\mathcal{N}} = 0$ for all $x \in S_P$ and $s_{\mathcal{B}} = 0$ for all $(\lambda,s) \in S_D$.*

(2) *We have $\mathcal{B} = \{i \in \{1,\dots,n\} \mid x_i^* > 0\}$ and $\mathcal{N} = \{i \in \{1,\dots,n\} \mid s_i^* > 0\}$, where $w^* = (x^*,\lambda^*,s^*)$ denotes a strictly complementary solution of the optimality conditions* (3.5)

Proof.

(1) Let $x \in S_P$ and $i \in \mathcal{N}$ be arbitrarily given. Then there exists a solution $(\lambda, s) \in S_D$ such that $s_i > 0$. Since $(x, \lambda, s) \in S_P \times S_D = S$, it follows from the complementarity condition that $x_i = 0$. This shows that $x_{\mathcal{N}} = 0$ for all $x \in S_P$. In a similar way, one can verify the statement that $s_{\mathcal{B}} = 0$ for all $(\lambda, s) \in S_D$.

(2) Let $w^* = (x^*, \lambda^*, s^*)$ be a strictly complementary solution of the optimality conditions (3.5). Define the corresponding index sets

$$\mathcal{B}^* := \{i \in \{1, \dots, n\} \mid x_i^* > 0\} \quad \text{and} \quad \mathcal{N}^* := \{i \in \{1, \dots, n\} \mid s_i^* > 0\}.$$

Since x^* and (λ^*, s^*) are particular solutions of the primal and dual linear programs (3.1) and (3.2), respectively, the definitions of the index sets $\mathcal{B}$ and $\mathcal{N}$ immediately give

$$\mathcal{B}^* \subseteq \mathcal{B} \quad \text{and} \quad \mathcal{N}^* \subseteq \mathcal{N}. \tag{3.8}$$

However, since $\mathcal{B}^* \cup \mathcal{N}^* = \{1, \dots, n\}$ by the strict complementarity assumption, this set has the same cardinality as the set $\mathcal{B} \cup \mathcal{N}$ (by the Goldman-Tucker Theorem 3.12). Hence equalities must hold in (3.8). ∎

3.5 Rank of the Matrix A

When discussing algorithms for linear programming, it is often convenient to assume that the matrix A in (3.1) hat full row rank m. There exist perfectly valid linear programs that do not satisfy this assumption, but such problems can always be transformed to equivalent problems that *do* satisfy it.

In practice this is not much of a problem. Implementations of the simplex algorithm typically introduce artificial variables into the formulation during the determination of an initial basis. In the augmented formulation, the constraint matrix A has full rank.

Interior-point codes as well as the implementations of the smoothing-type methods presented in this thesis do not perform such an augmentation. Instead, a presolver is run, which aims to eliminate the rank deficiencies by removing, among others, redundant constraints and empty rows, before the actual method is applied to the problem. Some presolvers used today employ some quite advanced linear algebra to reach this goal.

For the remainder of this thesis it is therefore always assumed that the matrix A has full row rank m, even if this is not explicitly specified.

3.6 Perturbation Results

This section covers two results which will be used later to prove local rate of convergence results for some of the algorithms.

The following result states that the non-singularity of the Jacobian $F'(w)$ in a neighborhood of x^* follows from the non-singularity of the Jacobian $F'(x^*)$. Additionally the inverse matrices are uniformly bounded. The proofs are based on Banach's Lemma, which is also known as Perturbation Lemma, [19, Theorem 3.1.4] or [25, Lemma B.8].

Lemma 3.14 *Let $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be continuously differentiable, $x^* \in \mathbb{R}^n$ and $F'(x^*)$ non-singular. Then there exists an $\varepsilon > 0$, such that $F'(w)$ is non-singular for all $w \in \mathcal{U}_\varepsilon(x^*) := \{x \in \mathbb{R}^n \mid \|x - x^*\| \le \varepsilon\}$. Furthermore there is a constant $c > 0$, such that*

$$\|F'(w)^{-1}\| \le c \quad \textit{for all } w \in \mathcal{U}_\varepsilon(x^*).$$

Proof. Geiger, Kanzow [25, Lemma 7.3]. ■

The following lemma is quite useful, since it gives an estimate of the distance of a given vector $x^k \in \mathbb{R}^n$ from the solution of an optimization problem such as (3.3).

Lemma 3.15 *Let $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be continuously differentiable and $\{x^k\} \subseteq \mathbb{R}^n$ a sequence converging to x^* with $F(x^*) = 0$ and $F'(x^*)$ non-singular. Then there is an index $k_0 \in \mathbb{N}$ and a constant $\gamma > 0$ such that*

$$\|F(x^k)\| \ge \gamma \|x^k - x^*\|$$

for all $k \ge k_0$.

Proof. Geiger, Kanzow [25, Lemma 7.4]. ■

3.7 Convergence Results

The following result is essentially an application of Taylor's Theorem. The proof can be found in any standard optimization text book.

Lemma 3.16 *Let $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be continuously differentiable, and let $\{x^k\} \subseteq \mathbb{R}^n$ be a sequence converging to x^*. Then the following assertions hold:*

(1) $\|F(x^k) - F(x^*) - F'(x^k)(x^k - x^*)\| = o(\|x^k - x^*\|)$.

(2) $\|F(x^k) - F(x^*) - F'(x^k)(x^k - x^*)\| = O(\|x^k - x^*\|^2)$, *if F' is locally Lipschitzian.*

Proof. Geiger, Kanzow [25, Lemma 7.2]. ■

Lemma 3.17 *Let $\{x^k\} \subseteq \mathbb{R}^n$ and $\{\Delta x^k\} \subseteq \mathbb{R}^n$ be two sequences such that*

$$\{x^k\} \longrightarrow x^* \quad \text{and} \quad \|x^k + \Delta x^k - x^*\| = o(\|x^k - x^*\|).$$

Let $F\colon \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be continuously differentiable and $F(x^*) = 0$. Let $F'(x^*)$ be non-singular. Then

$$\|F(x^k + \Delta x^k)\| = o(\|F(x^k)\|)$$

Proof. First note that F is locally Lipschitzian in x^* since it is continuously differentiable. Therefore

$$\begin{aligned}
\|F(x^k + \Delta x^k)\| &= \|F(x^k + \Delta x^k) - F(x^*)\| \\
&= O(\|x^k + \Delta x^k - x^*\|) \\
&= o(\|x^k - x^*\|) \\
&= o(\|F^{-1}(F(x^k)) - F^{-1}(F(x^*))\|) \\
&= o(\|F(x^k) - F(x^*)\|) \\
&= o(\|F(x^k)\|)
\end{aligned}$$

Here the last but second line follows from the Inverse Function Theorem (cf. [52]) and the last but one line follows from the fact that F^{-1} is locally Lipschitzian. ∎

The following result by Moré and Sorensen [46] gives conditions under which an entire sequence converges to an accumulation point if a subsequence does the same.

Proposition 3.18 *Assume that $x^* \in \mathbb{R}^n$ is an isolated accumulation point of a sequence $\{x^k\} \subseteq \mathbb{R}^n$ such that $\{\|x^{k+1} - x^k\|\} \longrightarrow 0$ for any subsequence $\{x^k\}_L$ converging to x^*. Then the whole sequence $\{x^k\}$ converges to x^*.*

Proof. Moré, Sorensen [46]. ∎

3.8 Multifunctions

For some of the convergence results, some more terminology is needed. The definitions and the following result are taken from Robinson [51].

Definition 3.19

(1) *A function whose values are sets instead of points, i. e., a multivalued function is called a* multifunction.

(2) *A multifunction whose graph is the union of finitely many polyhedral convex sets (called* components*) is called* polyhedral multifunction.

Definition 3.20 *A multifunction* $F \colon \mathbb{R}^n \longrightarrow \mathbb{R}^m$ *is called* locally upper Lipschitzian *at a point* v_0 *with modulus* ϑ*, if for some neighborhood* $\mathcal{U}(x_0)$ *of* x_0 *and all* $x \in \mathcal{U}$ *holds* $F(v) \subseteq F(v_0) + \vartheta\|v - v_0\|\mathcal{B}_1(0)$*, where* $\mathcal{B}_1(0) := \{v \mid \|v\| \leq 1\}$ *denotes the unit ball.*

Proposition 3.21 *Let* $F \colon \mathbb{R}^n \longrightarrow \mathbb{R}^m$ *be a polyhedral multifunction. Then there exists a constant* ϑ*, such that* F *is locally upper Lipschitzian at each* $v_0 \in \mathbb{R}^n$*.*

Proof. Robinson [51, Proposition 1]. ■

4 Interior-Point Methods

For a long time the simplex method had been state of the art in the field of linear programming. In recent years, however, a new class of methods, so-called interior-point methods, has become available and is now being widely used due to their excellent performance.

The interior-point methods described here are so-called primal-dual methods, i. e., no work is done on the the primal or dual problem directly but rather on the both problems' common optimality conditions.

In this chapter an overview will be given over interior-point methods, as they are closely related to the smoothing-type method approach demonstrated in later chapters. After general background information, we will give special attention to the most prominent member of the class of primal-dual interior-point methods, the predictor-corrector method published by Sanjay Mehrotra [43] in 1992.

4.1 The Central Path

The central path C is an arc of strictly feasible points that plays an important role in the theory of primal-dual algorithms. The arc is parameterized by a scalar $\tau > 0$; each point $(x_\tau, \lambda_\tau, s_\tau)$ solves the following system:

$$\begin{aligned} A^{\mathrm{T}}\lambda + s &= c, \\ Ax &= b, \\ x_i > 0, \;\; s_i > 0, \;\; x_i s_i &= \tau, \quad i = 1, \dots, n. \end{aligned} \tag{4.1}$$

These conditions differ from the KKT conditions (3.5) only in the term τ on the right-hand side in the third equation. Instead of the complementarity condition, we require that all pairwise products $x_i s_i$ have the same value for all indices i. From (4.1) we define the *central path* as

$$C = \{(x_\tau, \lambda_\tau, s_\tau) \mid \tau > 0\}\,. \tag{4.2}$$

This path plays a central role for the interior-point methods covered here; all these methods try to follow the central path in a more or less precise way.

However, it is not clear that the central path conditions (4.1) have a (unique) solution $(x_\tau, \lambda_\tau, s_\tau)$ for each $\tau > 0$. Our first task is to establish the existence and uniqueness of such a solution.

To this end we define the so-called *(logarithmic) barrier problem* corresponding to the primal linear program (3.1):

$$\min c^{\mathrm{T}}x - \tau \sum_{i=1}^{n} \log(x_i) \quad \text{subject to} \quad Ax = b, x > 0, \tag{4.3}$$

and the (logarithmic) barrier problem corresponding to the dual program (3.2):

$$\max b^{\mathrm{T}}\lambda - \tau \sum_{i=1}^{n} \log(s_i) \quad \text{subject to} \quad A^{\mathrm{T}}\lambda + s = c,\ s > 0, \tag{4.4}$$

The next result states that both barrier problems as well as the central path conditions indeed have a solution.

Proposition 4.1 *The following statements are equivalent for any given $\tau > 0$:*

(1) The primal barrier problem (4.3) *has a solution x_τ.*

(2) The dual barrier problem (4.4) *has a solution (λ_τ, s_τ).*

(3) The central path condition (4.1) *have a solution $(x_\tau, \lambda_\tau, s_\tau)$.*

Proof. Kanzow [34] ■

Having established this result, we can now turn to show that the primal barrier problems attains a solution.

Since the problem does not always have a solution, an additional condition is required to guarantee its solvability. Recall the definition of the *primal-dual strictly feasible set*

$$\mathcal{F}^0 := \left\{(x, \lambda, s) \;\middle|\; Ax = b,\ A^{\mathrm{T}}\lambda + s = c,\ x > 0,\ s > 0\right\}.$$

If $(x_\tau, \lambda_\tau, s_\tau)$ satisfies this central path conditions (4.1), this vector obviously belongs to the feasible set $\mathcal{F}^0$. Hence, if this set is empty, then the central path conditions (4.1) cannot have a solution. This, in turn, implies that the primal barrier problem (4.3) does not have a solution by Proposition 4.1. Therefore $\mathcal{F}^0 \neq \emptyset$ is a necessary condition for (4.3) to attain a minimum. The following result shows that this is also a sufficient condition.

Proposition 4.2 *Assume that $\mathcal{F}^0 \neq \emptyset$. Then the primal barrier problem* (4.3) *has a solution x_τ for any $\tau > 0$.*

Proof. Kanzow [34] ■

Theorem 4.3 *Assume that $\mathcal{F}^0 \neq \emptyset$. Then the central path conditions* (4.1) *have a solution $(x_\tau, \lambda_\tau, s_\tau)$ for any $\tau > 0$. Moreover, x_τ and s_τ are unique. If A has full row rank, λ_τ is also unique.*

Proof. Kanzow [34] ■

4.2 Path Following Methods

The class of primal-dual algorithms which will be described in the following sections consists essentially of Newton's method, applied to the central path conditions.

These methods restrict the iterates to a neighborhood of the central path C (as defined in (4.2)) and try to follow C numerically to a solution of the linear program in question. For this reason this class of algorithms is generally referred to as *path-following methods.*

4.3 A Basic Path-Following Method

Consider the linear program

$$\min \ c^{\mathrm{T}}x \quad \text{subject to} \quad Ax = b, \ x \geq 0,$$

where $A \in \mathbb{R}^{m \times n}$, $c \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$ and recall the corresponding central path conditions

$$\begin{aligned} A^{\mathrm{T}}\lambda + s &= c, \\ Ax &= b, \\ x_i > 0, \ s_i > 0, \ x_i s_i &= \tau \ \text{ for all } i = 1, \dots, n. \end{aligned}$$

An alternative way to write these conditions is to employ a mapping $F\colon \mathbb{R}^{n+m+n} \longrightarrow \mathbb{R}^{n+m+n}$,

$$F(x, \lambda, s) := \begin{bmatrix} A^{\mathrm{T}}\lambda + s - c, \\ Ax - b, \\ XSe \end{bmatrix}, \tag{4.5}$$

where $X := \operatorname{diag}(x_1, \dots, x_n) \in \mathbb{R}^{n \times n}$, $S := \operatorname{diag}(s_1, \dots, s_n) \in \mathbb{R}^{n \times n}$, and the vector $e := (1, \dots, 1)^{\mathrm{T}} \in \mathbb{R}^n$ is the vector of all ones. Now the central path conditions can be written as

$$\begin{gathered} F(x, \lambda, s) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}, \\ x, s > 0. \end{gathered} \tag{4.6}$$

All interior-point methods described in this thesis try to solve the linear program by finding a solution to the optimality conditions (3.5). Typically, this is done by trying to solve a sequence of non-linear problems of the type

$$F(x, \lambda, s) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}$$

with decreasing values of τ. Afterwards a suitable line search is performed in order to guarantee that x and s stay positive. Note that all interior-point methods generate iterates (x^k, λ^k, s^k) that satisfy the constraints $x^k > 0$ and $s^k > 0$ strictly. This property is the origin of the term *interior-point*, since the iterates x^k and s^k stay within the positive orthant at all times.

When applying Newton's method to $F(w) = 0$, with $w = (x, \lambda, s)$, we have to solve the linear system $F'(w)\Delta w = -F(w)$. We will start by showing that the Jacobian

$$F'(x, \lambda, s) = \begin{bmatrix} 0 & A^{\mathrm{T}} & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \tag{4.7}$$

is non-singular under suitable assumptions.

Proposition 4.4 *Let $(x, \lambda, s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ be any given point with $x, s > 0$ and assume that the matrix $A \in \mathbb{R}^{m \times n}$ has full rank. Then the Jacobian $F'(x, \lambda, s)$ of F is non-singular.*

Proof. Kanzow [34] ■

Note that in the last result the assumption $x, s > 0$ is not hard to satisfy, since all interior-point methods generate only positive iterates.

Proposition 4.4 now allows us to apply Newton's method to the system

$$F(x, \lambda, s) = \begin{bmatrix} 0 \\ 0 \\ \tau e \end{bmatrix}.$$

The following algorithm 4.5 presents a general framework for path-following methods which utilizes the strictly feasible set $\mathcal{F}^0$.

Algorithm 4.5 (General Path Following Method)

Step 0: Choose $(x^0, \lambda^0, s^0) \in \mathcal{F}^0$, $\varepsilon \in (0, 1)$ and set the iteration counter $k := 0$.

Step 1: If $(x^k)^{\mathrm{T}} s^k \le \varepsilon$: Stop.

Step 2: Choose $\sigma_k \in [0, 1]$ and set $\mu_k := (x^k)^{\mathrm{T}} s^k / n$.
Find a solution $(\Delta x^k, \Delta\lambda^k, \Delta s^k)$ of the linear system

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -X^k S^k e + \sigma_k \mu_k e \end{bmatrix} \tag{4.8}$$

Step 3: Set $(x^{k+1}, \lambda^{k+1}, s^{k+1}) := (x^k, \lambda^k, s^k) + t_k(\Delta x^k, \Delta\lambda^k, \Delta s^k)$, set $k \longleftarrow k+1$ and go to Step 1. Here $t_k > 0$ is a step length which guarantees that $x^{k+1} > 0$ and $s^{k+1} > 0$.

Note that the starting point (x^0, λ^0, x^0) is selected from the strictly feasible set $\mathcal{F}^0$, and that the choice of the step length t_k in Step 3 guarantees the positivity of all iterates x^k and s^k. Hence all iterates belong to the set $\mathcal{F}^0$. Therefore x^k is always primal feasible and (λ^k, s^k) is always dual feasible. Together with the optimality conditions (3.5) and

$$s^{\mathrm{T}}x = (c - A^{\mathrm{T}}\lambda)^{\mathrm{T}}x = c^{\mathrm{T}}x - \lambda^{\mathrm{T}}(Ax) = c^{\mathrm{T}}x - b^{\mathrm{T}}\lambda.$$

this yields

$$(x^k)^{\mathrm{T}}s^k = c^{\mathrm{T}}x^k - b^{\mathrm{T}}\lambda^k,$$

i. e. the expression $(x^k)^{\mathrm{T}}s^k$ describes the duality gap, which is the only part of the problem which needs to be driven to zero. This also explains the termination criterion used in Step 1 of Algorithm 4.5: The iteration is terminated if the duality gap is small enough, which indicates that a solution has been found, cf. Corollary 3.2. Furthermore, since the linear constraints are always satisfied, it becomes obvious why it is possible to use two zeros in the right-hand side of (4.8) in Step 2.

Since the system (4.8) can be solved uniquely if A has full row rank by Proposition 4.4, and since we can always find a positive step size t_k such that $x^{k+1} > 0$ and $s^{k+1} > 0$ in Step 3, it follows that Algorithm 4.5 is well-defined.

Algorithm 4.5 has two degrees of freedom: The choice of the so-called *centering parameter* σ_k and the choice of the step size t_k.

In (4.8) together with the weighted duality gap $\mu_k := (x^k)^{\mathrm{T}}s^k/n$, the centering parameter σ_k plays the role of the parameter τ in the central path conditions (4.1). While μ_k will always denote the weighted duality gap, we are relatively free in the choice of σ_k. The choice $\sigma_k = 0$ results in one Newton iteration being taken for the optimality conditions (3.5), which is the system that is to be solved in the first place. However, such a step may take us close to the boundary of the feasible set and thus result in a smaller step length in order to ensure the feasibility of the iterates x^k and s^k. On the other hand, choosing $\sigma_k = 1$ will take us close to the central path, but quite possibly not any closer to the solution set of (3.5), so one needs to be careful with the choice of a centering parameter. Mehrotra [43] addressed this problem when publishing his predictor-corrector method. This algorithm will be presented in the next section.

4.4 Mehrotra's Predictor-Corrector Approach

Most current implementations of primal-dual interior-point algorithms are based on a predictor-corrector algorithm which was first introduced by Mehrotra [43] in 1992.

Mehrotra's algorithm enhances the basic Newton-like search direction by a correction which is fairly inexpensive to compute. It also allows an adaptive choice of the centering parameter σ at each iteration.

Mehrotra's approach is largely based on ideas which had been developed earlier. One of the algorithm's key features, the use of higher-order approximations to the central path, is based on work by Monteiro, Adler, and Resende [45], who extended the initial research done by Megiddo [42]. Already an infeasible interior-point method had successfully been implemented by Lustig, Marsten and Shanno [40]. Such an infeasible interior-point method is characterized by the fact that the initial point is not required to be strictly feasible, but rather that its x- and s-components be strictly positive.

Mehrotra himself combined these ideas and added heuristics for choosing the centering parameter (adaptively), step lengths, and a starting point. The result is a highly efficient algorithm on which most existing interior-point codes are based.

The algorithm generates a sequence of infeasible iterates (x^k, λ^k, s^k) for which $(x^k, s^k) > 0$. The search direction at each iteration is computed in three steps:

(1) An affine-scaling "predictor" direction. This is the pure Newton direction for the function $F(x, \lambda, s)$ defined in (4.5).

(2) A centering term whose size is governed by the adaptively chosen centering parameter σ.

(3) A "corrector" direction that attempts to compensate for some of the nonlinearity in the affine-scaling direction.

The combination of the first two components, the affine-scaling step and the centering term, makes up the standard infeasible-interior-point method.

In Mehrotra's algorithm the affine-scaling direction and the centering term are computed separately from each other. Computing the scaling direction prior to the centering component results in the algorithm's main advantage: The centering parameter σ can be chosen adaptively rather than a priori. If the affine-scaling step makes good progress in reducing the duality measure μ while remaining inside the positive orthand defined by $(x, s) > 0$, one can conclude that little centering is needed in this iteration, so σ is assigned a small value. If, on the other hand, one can move only a short distance along the affine-scaling direction before violating $(x, s) > 0$, a larger amount of centering is required, so a larger value close to 1 is assigned to σ.

The drawback of calculating the centering component separately in this way is that now *two* linear systems have to be solved at each iterations instead of only one. However, this disadvantage is mitigated by the fact that both systems have the same coefficient matrix, thus we only need to perform one matrix factorization and two back-substitutions, one for each right-hand side.

The term *predictor-corrector* arose due to the analogy of this algorithm with a class of methods used in the field of ordinary differential equations.

The affine-scaling direction is obtained from a linear approximation of the equality conditions in the KKT system (3.5), the pure Newton directon for the function $F(x, \lambda, s)$ from (4.5). Since this direction is computed separately, it can be used to assess the error in the linear approximation. Knowledge of this error allows one to calculate the corrector component, the third component of the

Mehrotra-search direction. Since the centering and corrector components are obtained by solving linear systems with the same coefficient matrix as the affine-scaling direction, and since they are independent of each other, there is no need to compute them separately. We can simply merge them into a single direction by adding their corresponding right-hand sides and compute the combined centering-corrector direction with a single back-substitution.

This way Mehrotra's algorithm requires two back-substitutions at each iteration instead of only one, as earlier methods did. The extra cost of an additional back-substitution is easily justified by a significant reduction in the total number of iterations required to solve a problem. This way Mehrotra's method is highly efficient in solving a linear program.

The idea of different primal and dual step lengths has been put to practical use by numerous authors before. Mehrotra, however, added a quite successful heuristic to the step length computation.

4.4.1 Statement of Algorithm

The description of Mehrotra's predictor-corrector method given in this section differs slightly from the original one published by Mehrotra [43], but is similar to the variant implemented in codes such as LIPSOL [57, 58] and PCx [16, 17].

Algorithm 4.6 (Mehrotra's Predictor-Corrector Method)

Step 0: Choose $w^0 = (x^0, \lambda^0, s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ with $x^0 > 0$ and $s^0 > 0$ (see Section 4.4.3), $0 < \gamma_f \ll 1$, $\varepsilon > 0$ and set the iteration counter $k := 0$.

Step 1: If the termination criterion

$$\frac{\|Ax - b\|}{\max\{1, \|b\|\}} + \frac{\|A^{\mathrm{T}}\lambda + s - c\|}{\max\{1, \|c\|\}} + \frac{|c^{\mathrm{T}}x - b^{\mathrm{T}}y|}{\max\{1, |c^{\mathrm{T}}x|\}} \le \varepsilon$$

is satisfied: Stop.

Step 2 (Affine-Scaling Predictor Step):

Compute a solution $\Delta w_a^k = (\Delta x_a^k, \Delta\lambda_a^k, \Delta s_a^k)$ of the linear system

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = - \begin{bmatrix} A^{\mathrm{T}}\lambda^k + s^k - c \\ Ax^k - b \\ X^k S^k e \end{bmatrix}. \tag{4.9}$$

Compute

$$\begin{aligned} t_k^{a,P} &= \arg\max\left\{t \in [0,1] \;\middle|\; x^k + t\Delta x_a^k \ge 0\right\}, \\ t_k^{a,D} &= \arg\max\left\{t \in [0,1] \;\middle|\; s^k + t\Delta s_a^k \ge 0\right\}, \end{aligned}$$

and

$$\mu_k^a = \frac{\left(x^k + t_k^{a,P}\Delta x_a^k\right)^{\mathrm{T}}\left(s^k + t_k^{a,D}\Delta s_a^k\right)}{n},$$
$$\sigma_k = \left(\frac{\mu_k^a}{\mu}\right)^3.$$

Step 3 (Centering-Corrector Step):
Find a solution $\Delta w_c^k = (\Delta x_c^k, \Delta\lambda_c^k, \Delta s_c^k)$ of the linear system

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = -\begin{bmatrix} 0 \\ 0 \\ \Delta X_a^k \Delta S_a^k e - \sigma_k \mu_k^a e \end{bmatrix} \tag{4.10}$$

Step 4 (Search Direction):
Calculate the search direction: $\Delta w^k = \Delta w_a^k + \Delta w_c^k$.

Step 5 (Line Search):
Compute

$$t_k^{\max,P} = \arg\max\left\{t \geq 0 \;\middle|\; x^k + t\Delta x^k \geq 0\right\}, \tag{4.11a}$$
$$t_k^{\max,D} = \arg\max\left\{t \geq 0 \;\middle|\; s^k + t\Delta s^k \geq 0\right\}. \tag{4.11b}$$

Calculate

$$\mu_+ = \frac{\left(x^k + t_k^{\max,P}\Delta x^k\right)^{\mathrm{T}}\left(s^k + t_k^{\max,D}\Delta s^k\right)}{n}.$$

For one particular index i for which $x_i^k + t_k^{\max,P}\Delta x_i^k = 0$, define f_k^P by

$$(x_i^k + f_k^P t_k^{\max,P} + \Delta s_i^k)(s_i^k + t_k^{\max,D} + \Delta s_i^k) = \gamma_f \mu_+.$$

For one particular index i for which $s_i^k + t_k^{\max,P}\Delta s_i^k = 0$, define f_k^D by

$$(x_i^k + t_k^{\max,P}\Delta x_i^k)(s_i^k + f_k^D t_k^{\max,D}\Delta s_i^k) = \gamma_f \mu_+.$$

Set

$$t_k^P = \max\left\{1 - \gamma_f, f_k^P\right\} \cdot t_k^{\max,P},$$
$$t_k^D = \max\left\{1 - \gamma_f, f_k^D\right\} \cdot t_k^{\max,D}.$$

Step 6 (Update):
Compute the new iterate as

$$x^{k+1} = x^k + t_k^P \Delta x^k,$$
$$(\lambda^{k+1}, s^{k+1}) = (\lambda^k, s^k) + t_k^D(\Delta\lambda^k, \Delta s^k).$$

Set $k \leftarrow k + 1$ and go to Step 1.

General Comments

For a better understanding of this algorithm we will give some notes on specific parts of the method (we will leave out the iteration counter k throughout this section):

The linear system solved in Step 2 of the Algorithm is obtained by setting $\sigma = 0$ on the right-hand side of the generic infeasible-interior-point equations (4.8). The step lengths to the boundary along this direction can now be computed, performing separate calculations for the primal and dual components as follows:

$$\begin{aligned} t^{a,P} &= \arg\max \{t \in [0,1] \mid x + t\Delta x_a \geq 0\}, \\ t^{a,D} &= \arg\max \{t \in [0,1] \mid s + t\Delta s_a \geq 0\}, \end{aligned}$$

Now define μ^a as the hypothetical value of μ resulting from a full step to the boundary, i. e.,

$$\mu^a = \frac{\left(x + t^{a,P}\Delta x_a\right)^{\mathrm{T}} \left(s + t^{a,D}\Delta s_a\right)}{n}.$$

This gives us a measure of the affine-scaling direction's quality. If μ^a is significantly smaller than μ, the affine-scaling direction is a good search direction which permits significant progress to be made in reducing μ, so the centering parameter σ can be chosen close to zero. If μ^a is only a little smaller than μ, we choose σ closer to 1. This results in placing the iterate closer to the central path C, so that the algorithm is in a better position to achieve a substantial decrease in μ on the next iteration. The following heuristic for the computation of the centering parameter has proven to be quite effective in numerical tests:

$$\sigma = \left(\frac{\mu^a}{\mu}\right)^3.$$

One could compute a centering term by solving a linear system with the same coefficient matrix as in the affine-scaling step and the right-hand side $(0, 0, \sigma\mu e)^{\mathrm{T}}$, but as mentioned before, it is more efficient to compute this term in conjunction with the corrector term.

For a motivation of the corrector step, consider the i-th pairwise product $x_i s_i$. Let us examine how this product is affected by a full step in the affine-scaling direction. From (4.9) we have

$$\begin{aligned} (x_i + (\Delta x_a)_i)\,(s_i + (\Delta s_a)_i) &= x_i s_i + x_i(\Delta s_a)_i + s_i(\Delta x_a)_i + (\Delta x_a)_i(\Delta s_a)_i \\ &= (\Delta x_a)_i(\Delta s_a)_i. \end{aligned} \tag{4.12}$$

When a full step is taken, the pairwise product $x_i s_i$ transforms to $(\Delta x_a)_i(\Delta s_a)_i$, instead of the zero value predicted in the linearized model used in (4.9). It is the corrector component's task to compensate for the deviation from linearity. This is done by modifying the search direction so that the pairwise products come closer to their target value of zero.

In order to assess the effect the corrector component, examine the pairwise product obtained from a full step in the combined affine-scaling/corrector direction. From the linear systems (4.9) and (4.10) we obtain

$$\begin{aligned}(x_i + (\Delta x_a)_i + (\Delta x_c)_i)\,(s_i + (\Delta s_a)_i + (\Delta s_c)_i)\\ = (\Delta x_a)_i(\Delta s_c)_i + (\Delta x_c)_i(\Delta s_a)_i + (\Delta x_c)_i(\Delta s_c)_i \end{aligned} \tag{4.13}$$

The pairwise product in (4.13) is closer to zero than the one in (4.12) if the coefficient matrix of the linear systems approaches its limit. If the limiting matrix is singular the corrector step may no longer be smaller in norm than the affine-scaling step, but in practice the use of the corrector component usually still enhances the overall efficiency of the algorithm. The cost of this additional efficiency is quite small: For the corrector step one has to solve an additional linear system with the same coefficient matrix as in the corrector step. Therefore only a single back-substitution is required to obtain a solution of the linear system.

The step length computation described in Step 5 of Algorithm 4.6 is not commonly used. Most codes simply apply fixed factors such as 0.99 to the computed step length rather than computing the step factors as documented. The heuristic described in Step 5 was published by Mehrotra [43]. When using this heuristic the scaling factors f^P and f^D applied to t^P and t^D approach 1 in later iterations. The factors are chosen to be the largest value for which the "critical" pairwise products $x_i s_i$ are larger than $\gamma_f \mu_+$, for a small constant γ_f. A typical value for γ_f is 0.01.

4.4.2 Termination Criteria

Unlike the simplex method, primal-dual algorithms usually do not find an exact solution of the linear program. Termination criteria must therefore be used to decide when the current iterate is close enough to the solution set. In this case most primal-dual codes simply report an approximate solution for which the residuals and duality measure are sufficiently small. Relative measures of smallness are used to lessen the effect of scaling. The termination criterion in Step 1 of Algorithm 4.6 is taken from the code LIPSOL by Zhang [57, 58] and represents a typical case of such a criterion. The parameter ε is typically set to a small positive value, such as 10^{-8}.

4.4.3 The Initial Point

First we present Mehrotra's suggestion for choosing an initial point for his interior-point method, then its merits are discussed.

Computation of the Initial Point

Mehrotra suggests in [43] that the initial point for the predictor-corrector method should be computed as follows: First calculate

$$\begin{aligned}
\tilde{\lambda} &= \left(AA^{\mathrm{T}}\right)^{-1} Ac, \\
\tilde{s} &= c - A^{\mathrm{T}}\tilde{\lambda}, \\
\tilde{x} &= A^{\mathrm{T}} \left(AA^{\mathrm{T}}\right)^{-1} b,
\end{aligned}$$

by solving the appropriate linear systems, then compute

$$\delta_x = \max\{-1.5 \cdot \min\{\tilde{x}_i\}, 0\} \quad \text{and} \quad \delta_s = \max\{-1.5 \cdot \min\{\tilde{s}_i\}, 0\}.$$

Then one obtains

$$\tilde{\delta}_x = \delta_x + \frac{1}{2} \cdot \frac{(\tilde{x} + \delta_x e)^{\mathrm{T}}(\tilde{s} + \delta_s e)}{\sum_{i=1}^{n} (\tilde{s}_i + \delta_s)}, \tag{4.14a}$$

$$\tilde{\delta}_s = \delta_s + \frac{1}{2} \cdot \frac{(\tilde{x} + \delta_x e)^{\mathrm{T}}(\tilde{s} + \delta_s e)}{\sum_{i=1}^{n} (\tilde{x}_i + \delta_x)}. \tag{4.14b}$$

This allows the initial point to be generated as

$$\begin{aligned}
x_i^0 &= \tilde{x}_i + \tilde{\delta}_x, \quad i = 1, \ldots, n, \\
\lambda^0 &= \tilde{\lambda}, \\
s_i^0 &= \tilde{s}_i + \tilde{\delta}_s, \quad i = 1, \ldots, n
\end{aligned}$$

This approach does indeed generate $x^0 > 0$ and $s^0 > 0$. In the cases in which it fails to produce such a point, either the problem reduces to that of finding a feasible solution for the primal or dual problem (3.1) or (3.2), respectively, or an optimal solution is generated.

From the definitions of δ_x and δ_s, we know that $x^0 \geq 0$ and $s^0 \geq 0$. A positive point is always generated if $\delta_s > 0$ and $\delta_s > 0$. To show that $x^0 > 0$ and $s^0 > 0$, it is sufficient to show that $\tilde{\delta}_x > 0$ and $\tilde{\delta}_s > 0$.

First consider the case when $\delta_x = \delta_s = 0$. In this case $\tilde{x}$ is a feasible solution for the primal problem (3.1) and $(\tilde{\lambda}, \tilde{s})$ is a feasible solution for the dual problem (3.2). If $\tilde{x}^{\mathrm{T}}\tilde{s} = 0$, then these solutions are optimal for the respective problems. Otherwise, $\tilde{x}^{\mathrm{T}}\tilde{s} > 0$ and hence $\tilde{\delta}_x > 0$ and $\tilde{\delta}_s > 0$.

On the other hand, if $\tilde{x}_i = 0$ for all i, then $b = 0$ and the problem reduces to that of finding a feasible solution of the dual problem (3.2). This problem can then be solved separately or by generating a perturbed problem for which the right-hand side is δAe for any positive δ. The term δe can be used as a feasible interior solution of the perturbed problem, and (4.14) can be used to generate a feasible point of the perturbed problem. The case in which $\delta_x > 0$ and $\delta_s = 0$ can be argued in a similar manner.

The choice of the constants 1.5 and 1/2 in the computation of $\tilde{\delta}_x$ and $\tilde{\delta}_s$ is arbitrary. These constants can be replaced by any value larger than 1 and 1/2, respectively, without changing the validity of the approach and its properties.

Advantages of Mehrotra's Initial Point

The approach described above has two advantages.

First, it is not affected by scaling of the primal constraints or the cost vector c.

Second it is independent of a shift in the origin. To explain this, consider the dual problem

$$\max \ b^{\mathrm{T}}(\lambda + \Delta\lambda) \quad \text{subject to} \quad A^{\mathrm{T}}(\lambda + \Delta\lambda) + s = c, \ \ s \geq 0 \tag{4.15}$$

for any fixed choice of $\Delta\lambda$. The polytope defined by the constraints in (4.15) is the same as the polytope defined by the constraints in (3.2) except for a shift of the origin. It is desirable that an initial point be the same in relation to the respective polytopes. Note that $\tilde{s}$ in (4.4.3) is independent of the choice of $\Delta\lambda$.

To demonstrate how similar arguments hold for the primal problem (3.1), we consider an equivalent formulation of this problem. Let $Z \in \mathbb{R}^{n\times m}$ be a matrix whose columns form the basis for the null space of A, and let x_0 be any point satisfying $Ax_0 = b$. It is easy to see that the primal problem (3.1) is equivalent to

$$\max \ (c^{\mathrm{T}}Z)(y + \Delta y) \quad \text{subject to} \quad Z^{\mathrm{T}}(y + \Delta y) \geq -x_0 \tag{4.16}$$

for $y \in \mathbb{R}^n$ and any fixed choice of Δy. An approach analogous to that of finding $\tilde{s}$ computes

$$\tilde{y} = -(Z^{\mathrm{T}}Z)^{-1}z^{\mathrm{T}}x_0.$$

The slacks in the constraints of (4.15) are given by

$$\begin{aligned} x_0 - Z(Z^{\mathrm{T}}Z)^{-1}Z^{\mathrm{T}}x_0 &= \left(I - Z(Z^{\mathrm{T}}Z)^{-1}Z^{\mathrm{T}}\right)x_0 \\ &= A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}Ax_0 \\ &= A^{\mathrm{T}}(AA^{\mathrm{T}})^{-1}b \\ &= \tilde{x}. \end{aligned}$$

Since $\tilde{x}$ and $\tilde{s}$ are independent of Δy and $\Delta\lambda$, respectively, it is obvious that x^0 and s^0 are independent of this as well.

Disadvantages of Mehrotra's Initial Point

The initial point generated by the above mentioned approach is affected by the presence of redundant constraints. This can pose a problem, for example, if redundant dual constraints with large slacks (primal variables with huge costs) are present. To alleviate this problem Mehrotra [43] suggests to let the user give relative importance to various primal variables (dual constraints). Redundant variables could then be assigned a zero weight and could be removed while generating the initial point.

Column scaling also affects the Mehrotra's initial point. For numerical examples the reader is referred to Mehrotra's paper [43], in which he compares initial points generated with and without the previous application of a scaling function.

4.4.4 Differences and Enhancements with Regard to Existing Implementation

Step length

In Step 5 of Algorithm 4.6 scaling factors are applied to the step-to-boundary values $t_k^{\max,P}$ and $t_k^{\max,D}$. These factors are computed by using an advanced heuristic. In a number of implementations, these step factors are either a fixed constant, ranging from conservative values such as 0.9 or 0.995 to aggressive values such as 0.999. The package LIPSOL by Zhang [57, 58] uses an adaptive strategy to prevent the pairwise products $x_i s_i$ from becoming too unbalanced. It first tries a multiplier very close to 1 and checks that the new iterate stays inside a neighborhood

$$\mathcal{N}_{-\infty}(\gamma) := \left\{(x,\lambda,s) \in \mathcal{F}^0 \;\middle|\; x_i s_i \geq \gamma\mu \quad \text{for all } i = 1,\dots,n\right\}$$

for some $\gamma \in (0,1)$. If not, it backs off along the chosen direction, trying smaller and smaller values for the step length until the neighborhood condition is satisfied.

A typical value for γ_f in Step 5 of Algorithm 4.6 is 0.01. This heuristic speeds up asymptotic convergence in addition to improving the robustness of the algorithm. Wright [56] also states, that in a number of cases, the use of the heuristic allows the algorithm to solve a problem, while the use of a step factor bounded away from 1 causes failure.

Search Direction

In the original algorithm, Mehrotra calculates a little different search direction from the one given in Algorithm (4.6). In [43] he defines the parameters ε_x and ε_s as the maximum steps to the boundary along a curved path:

$$\begin{aligned}\varepsilon_x &= \arg\max\left\{\varepsilon \in [0,1] \;\middle|\; x + \varepsilon\Delta x^a + \varepsilon^2\Delta x^c\right\},\\ \varepsilon_s &= \arg\max\left\{\varepsilon \in [0,1] \;\middle|\; s + \varepsilon\Delta s^a + \varepsilon^2\Delta s^c\right\}.\end{aligned}$$

He then chooses the search direction to be

$$(\Delta x, \Delta\lambda, \Delta s) = (\varepsilon_x\Delta x_a, \varepsilon_s\Delta\lambda_a, \varepsilon_s\Delta s_a) + \left(\varepsilon_x^2\Delta x_c, \varepsilon_s^2\Delta\lambda_c, \varepsilon_s^2\Delta s_c\right).$$

Note that the direction given in Algorithm 4.6 is slightly different, but is the one implemented in codes such as LIPSOL [57, 58] and PCx [16, 17].

Numerical performance

In [43] Mehrotra uses the cubic centering multiplyer σ_k as stated in Algorithm 4.6. Lustig, Marsten and Shanno [41] describe an implementation of Mehrotra's method which only uses a quadratic centering parameter. They state that it apparently makes little difference when comparing the iteration counts in [43] and [41]. The authors of [41] also cite better numerical stability if the centering parameter is computed as

$$\sigma_k = \frac{(x^k)^{\mathrm{T}} s^k}{\phi(n)}, \quad \text{where} \quad \phi(n) = \begin{cases} n^2, & \text{if } n \le 5000, \\ n^{2/3}, & \text{if } n > 5000. \end{cases}$$

For some numerical results produced by different codes, see e.g. [57] and [16] For a broader benchmark of different codes see [44].

5 Smoothing and Smoothing-Type Methods — Background

This chapter will cover the background required for the work with smoothing and smoothing-type methods. The methods introduced in the following chapters aim to solve a linear program's KKT conditions (3.5) by reformulating these conditions into a system of equations. To do this, a special class of functions, so-called *NCP-functions*, is employed.

5.1 NCP-Functions

Let us begin by defining one of the basic tools which we will use for the remainder of this thesis, the so-called *NCP-function.*

Definition 5.1 *A mapping $\varphi\colon \mathbb{R}^2 \longrightarrow \mathbb{R}$ is called an* NCP-function *if it has the following property:*

$$\varphi(a,b) = 0 \quad \Longleftrightarrow \quad a \geq 0,\ b \geq 0,\ ab = 0,$$

i. e., an NCP-function is zero exactly on the two non-negative half-axes.

Here *NCP* stands for *Non-linear Complementarity Problem.* This class of functions has originally been used in connection with non-linear and linear complementarity problems, hence the name. For an in-depth discussion of the possible uses of NCP-functions to solve complementarity and constrained optimization problems, see Kanzow [31] and references therein.

Recall a linear program's optimality conditions: A vector $x^* \in \mathbb{R}^n$ is a solution of the linear program (3.1) if and only if there exist vectors $\lambda^* \in \mathbb{R}^m$ and $s^* \in \mathbb{R}^n$ such that $(x,\lambda,s) = (x^*,\lambda^*,s^*)$ satisfies the following conditions:

$$\begin{aligned} A^{\mathrm{T}}\lambda + s - c &= 0, \\ Ax - b &= 0, \\ x_i \geq 0,\ s_i \geq 0,\ x_i s_i &= 0 \quad i = 1,\dots,n. \end{aligned}$$

Now let φ be any NCP-function and define the operator $\Phi\colon \mathbb{R}^{n+m+n} \longrightarrow \mathbb{R}^{n+m+n}$ by

$$\Phi(x,\lambda,s) := \begin{bmatrix} A^{\mathrm{T}}\lambda + s - c \\ Ax - b \\ \phi(x,s) \end{bmatrix}, \tag{5.1}$$

where $\phi(x,s) := (\varphi(x_1,s_1),\dots,\varphi(x_n,s_n))^{\mathrm{T}}$.

Obviously, (x^*,λ^*,s^*) solves (3.5) if and only if (x^*,λ^*,s^*) solves the system $\Phi(x,\lambda,s) = 0$. The solution of the optimality conditions (3.5) can therefore be reduced to the solution of a non-linear system of equations. Of course, the system of equations $\Phi(x, \lambda, s) = 0$ depends heavily on the choice of the NCP-function φ.

The following example lists a number of NCP-functions.

Example 5.2 The following functions are NCP-functions:

(1) The Fischer-Burmeister function, see [24]:

$$\varphi(a, b) := a + b - \sqrt{a^2 + b^2}. \tag{5.2}$$

(2) The minimum function, see [48] or [49]:

$$\varphi(a, b) := 2 \min\{a, b\} = a + b - |a - b| = a + b - \sqrt{(a - b)^2} \tag{5.3}$$

(3) The penalized Fischer-Burmeister function, cf. [10]:

$$\varphi(a, b) := \lambda(a + b - \sqrt{a^2 + b^2}) + (1 - \lambda)a_+ b_+ \text{ with } \lambda \in (0, 1) \text{ fixed.} \tag{5.4}$$

(4) The penalized minimum function:

$$\varphi(a, b) := \lambda 2 \min\{a, b\} + (1 - \lambda)a_+ b_+, \text{ with } \lambda \in (0, 1) \text{ fixed.} \tag{5.5}$$

This function seems to be new. It is, however, constructed in a way similar to the penalized Fischer-Burmeister function, therefore it is called the penalized minimum function. Its first apparent use has been in the course of [20] and [23].

In the last two examples the notation z_+ for any given scalar $z \in \mathbb{R}$ denotes its projection on the non-negative orthand, i. e., $\max\{0, z\}$.

For the remainder of this thesis, φ will always denote an NCP-function.

5.2 Smoothing functions

All of the functions given in example 5.2 are obviously non-smooth, which leads to the corresponding equation operator Φ also being non-smooth. This poses a problem if one intends to solve a system of equations such as $\Phi(x, \lambda, s) = 0$ by applying Newton's method. The primary reason why one does not use differentiable NCP-functions is the fact that simple proofs show that the Jacobian $\Phi'(x^*, \lambda^*, s^*)$ may be singular in the solution vector (x^*, λ^*, s^*) of $\Phi(x, \lambda, s) = 0$. For further details see [31]. To avoid this problem we use smooth approximations of the non-differentiable NCP-functions. These smooth approximations of non-smooth mappings are called *smoothing functions*. They are constructed by adding a so-called *smoothing parameter* to the function. This smoothing parameter will be called τ for the remainder of this thesis. In most smoothing and Jacobian-smoothing related literature this parameter is commonly called μ, which conflicts with the naming convention used here for the duality measure in interior-point literature, see

Chapter 4. Looking ahead at the system (5.21) one will see that this system is identical to the central path conditions (4.1), in which the perturbation parameter is also called τ. Hence the naming of the parameter is not misleading as it might seem at first.

The following example will name some smoothing functions which will be used throughout this thesis.

Example 5.3 The following functions are the smoothed counterparts of the respective NCP-functions in example 5.2. Note that for the approximation given here, we have $\varphi = \varphi_\tau$ for $\tau = 0$, and that φ_τ is continuously differentiable for any $\tau > 0$.

(1) The smoothed Fischer-Burmeister function:

$$\varphi_\tau(a,b) := a + b - \sqrt{a^2 + b^2 + 2\tau^2} \tag{5.6}$$

This mapping was first introduced by Kanzow [32].

(2) The smoothed minimum function, also known as the *Chen-Harker-Kanzow-Smale smoothing function*:

$$\varphi_\tau(a,b) := a + b - \sqrt{(a-b)^2 + 4\tau^2}. \tag{5.7}$$

See also [11, 32, 53]. There are several other suitable approximations of the minimum function, see e. g. Chen and Mangasarian [12].

(3) The smoothed penalized Fischer-Burmeister function:

$$\varphi_\tau(a,b) := \lambda\left(a + b - \sqrt{a^2 + b^2 + 2\tau^2}\right) + (1-\lambda)\sqrt{a_+^2 b_+^2 + 2\tau^2}, \tag{5.8}$$

with $\lambda \in (0,1)$ fixed.

(4) The smoothed penalized minimum function:

$$\varphi_\tau(a,b) := \lambda\left(a + b - \sqrt{(a-b)^2 + 4\tau^2}\right) + (1-\lambda)\sqrt{a_+^2 b_+^2 + 4\tau^2}, \tag{5.9}$$

with $\lambda \in (0,1)$ fixed.

The last two functions have not been used extensively before, they first appear in [10] and [20, 23], respectively

For the remainder of this thesis, φ_τ will always denote a smoothing function.

The next result shows that the four smoothing functions φ_τ defined in example 5.3 are indeed good approximations of the corresponding non-smooth NCP-functions from example 5.2 (see also [33]).

Lemma 5.4 *For each of the four NCP-functions φ defined in Example 5.2 and the corresponding smoothing functions φ_τ defined in Example 5.3, there exists a constant $c = c_\varphi > 0$ (independent of τ and (a,b)) such that*

$$|\varphi(a,b) - \varphi_\tau(a,b)| \le c\tau$$

for all $(a,b) \in \mathbb{R}^2$ and all $\tau > 0$.

For the four functions from examples 5.2 and 5.3 this constant has the following values:

(1) Fischer-Burmeister function: $c = \sqrt{2}$.

(2) Minimum function: $c = 2$.

(3) Penalized Fischer-Burmeister function: $c = \sqrt{2}$.

(4) Penalized minimum function: $c = 2$.

Proof. In case of the minimum function, we obtain

$$\begin{aligned}
|\varphi(a,b) - \varphi_\tau(a,b)| &= \left| a + b - \sqrt{(a-b)^2} - \left(a + b - \sqrt{(a-b)^2 + 4\tau^2}\right)\right| \\
&= \sqrt{(a-b)^2 + 4\tau^2} - \sqrt{(a-b)^2} \\
&= \frac{(a-b)^2 + 4\tau^2 - (a-b)^2}{\sqrt{(a-b)^2 + 4\tau^2} + \sqrt{(a-b)^2}} \\
&\le \frac{4\tau^2}{2\tau} \\
&= 2\tau,
\end{aligned}$$

for all $(a,b) \in \mathbb{R}^2$ and all $\tau > 0$, i. e., the statement holds with $c := 2$. Similar calculations verify that the statement holds for the other NCP-functions from example 5.2 with the values stated above. ∎

5.3 Smoothing a System of Equations

As the equation operator Φ defined in (5.1) is non-smooth, it is not possible to employ Newton's method to solve the non-linear system $\Phi(w) = 0$ directly. Instead, we define a continuously differentiable approximation of this operator. To achieve this goal, we will use smoothing functions instead of NCP-functions. This will give us a (perturbed) system which we will then solve while trying to lessen the perturbation at each iteration.

Define the operator $\Phi_\tau \colon \mathbb{R}^{n+m+n} \longrightarrow \mathbb{R}^{n+m+n}$ by

$$\Phi_\tau(x,\lambda,s) := \begin{bmatrix} A^{\mathrm{T}}\lambda + s - c \\ Ax - b \\ \phi_\tau(x,s) \end{bmatrix}, \tag{5.10}$$

where $\phi_\tau(x,s) := (\varphi_\tau(x_1,s_1),\dots,\varphi_\tau(x_n,s_n))^{\mathrm{T}} \in \mathbb{R}^n$.

As a direct consequence we obtain the following result, where Φ and Φ_τ denote the mappings defined in (5.1) and (5.10).

Lemma 5.5 *There exists a constant $\kappa > 0$ (independent of τ and $w = (x,\lambda,s)$), such that*

$$\|\Phi(w) - \Phi_\tau(w)\| \le \kappa\tau$$

For equation operators defined by the four functions from examples 5.2 and 5.3 this constant has the following values:

(1) Fischer-Burmeister function: $\kappa = \sqrt{2n}$.

(2) Minimum function: $\kappa = 2\sqrt{n}$.

(3) Penalized Fischer-Burmeister function: $\kappa = \sqrt{2n}$.

(4) Penalized minimum function: $\kappa = 2\sqrt{n}$.

Proof. Using Lemma 5.4 and the definition of the mappings Φ and Φ_τ, we obtain for the minimum function

$$\begin{aligned}\|\Phi(w) - \Phi_\tau(w)\| &= \sqrt{\sum_{i=1}^{n} (\varphi(x_i,s_i) - \varphi_\tau(x_i,s_i))^2} \\ &\le \sqrt{\sum_{i=1}^{n} c^2\tau^2} \\ &= \sqrt{nc^2\tau^2} \\ &= c\tau\sqrt{n},\end{aligned}$$

i. e., the statement holds with $\kappa := c\sqrt{n} = 2\sqrt{n}$. In a similar way, one can show that the assertion holds for the other NCP-functions with the values stated above. ■

Note that the constant κ introduced in Lemma 5.5 is actually known. This fact will play an important role in the design of the algorithms where this constant will be used explicitly.

In later chapters Newton's method will be used to solve the system $\Phi_\tau(w) = 0$. This yields the Newton equation

$$\Phi'_\tau(w)\Delta w = -\Phi_\tau(w). \tag{5.11}$$

The next result states that the Jacobian Matrices $\Phi'_\tau(w)$ are non-singular for all w. Hence the linear system (5.11) can always be solved uniquely.

Proposition 5.6 *Let Φ_τ denote any of the mappings defined by any of the four smoothing functions from example 5.3. Then the Jacobian matrices $\Phi'_\tau(w)$ are non-singular for all $w = (x, \lambda, s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ and all $\tau > 0$.*

Proof. It is easy to see that Φ_τ is differentiable with

$$\Phi'_\tau(x, \lambda, s) = \begin{bmatrix} 0 & A^{\mathrm{T}} & I \\ A & 0 & 0 \\ D_{a,\tau} & 0 & D_{b,\tau} \end{bmatrix} \tag{5.12}$$

with diagonal matrices

$$D_{a,\tau} := \operatorname{diag}\left(\ldots, \frac{\partial \varphi_\tau}{\partial a}(x_i, s_i), \ldots\right) \in \mathbb{R}^{n\times n}, \tag{5.13a}$$

and

$$D_{b,\tau} := \operatorname{diag}\left(\ldots, \frac{\partial \varphi_\tau}{\partial b}(x_i, s_i), \ldots\right) \in \mathbb{R}^{n\times n}. \tag{5.13b}$$

For all functions φ_τ defined in example 5.3 simple calculations show that the corresponding diagonal matrices $D_{a,\tau}$ and $D_{b,\tau}$ are positive definite

Now let $q = (q^{(1)}, q^{(2)}, q^{(3)}) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ be an appropriately partitioned vector with $\Phi'_\tau(w)q = 0$. Then (5.12) implies

$$A^{\mathrm{T}} q^{(2)} + q^{(3)} = 0, \tag{5.14a}$$

$$A q^{(1)} = 0, \tag{5.14b}$$

$$D_{a,\tau} q^{(1)} + D_{b,\tau} q^{(3)} = 0. \tag{5.14c}$$

Premultiplying (5.14a) by $(q^{(1)})^{\mathrm{T}}$ and taking into account (5.14b) gives

$$(q^{(1)})^{\mathrm{T}} q^{(3)} = 0. \tag{5.15}$$

On the other hand, solving (5.14c) for $q^{(1)}$ and substituting into (5.15) yields that

$$(q^{(3)})^{\mathrm{T}} D_{a,\tau}^{-1} D_{b,\tau} q^{(3)} = 0. \tag{5.16}$$

Since $D_{a,\tau}$ and $D_{b,\tau}$ are positive definite, we obtain $q^{(3)} = 0$ from (5.16). This implies $q^{(1)} = 0$ because of (5.14c). Hence we also get $q^{(2)} = 0$ from (5.14a) and the full rank assumption of A. ■

The following results will guarantee the existence of an accumulation point, when using one of the penalized functions (5.5) and (5.4) and their smooth approximations (5.9) and (5.8), respectively. It is not true, in general, for the minimum and the Fischer-Burmeister function.

Proposition 5.7 *Let φ be the penalized minimum or penalized Fischer-Burmeister function, and consider the set*

$$\mathcal{L}_c := \left\{ w = (x, \lambda, s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \;\middle|\; A^T\lambda + s = c,\;\; Ax = b,\;\; \|\Phi(w)\| \le c \right\},$$

where $c \in \mathbb{R}$ is any given constant. Assume that there exists a strictly feasible point for the optimality conditions (3.5)*. Then the set $\mathcal{L}_c$ is compact.*

Proof. We only consider the case where φ denotes the penalized minimum function since the proof for the penalized Fischer-Burmeister function is similar.

Assume that $\mathcal{L}_c$ is unbounded for some $c \in \mathbb{R}$. Then there is a sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\} \subseteq \mathcal{L}_c$ with $\{\|w^k\|\} \longrightarrow +\infty$ for $k \longrightarrow +\infty$. The definition of the penalized minimum function immediately implies that there is no index $i \in \{1, \dots, n\}$ such that $x_i^k \longrightarrow -\infty$ or $s_i^k \longrightarrow -\infty$ on a subsequence, since otherwise we would have $\varphi(x_i^k, s_i^k) \longrightarrow -\infty$ which, in turn, would imply $\|\Phi(w^k)\| \longrightarrow +\infty$ on a subsequence in contrast to $\|\Phi(w^k)\| \le c$ for all $k \in \mathbb{N}$. Hence all components of the two sequences $\{x^k\}$ and $\{s^k\}$ are bounded from below, i. e.,

$$x_i^k \ge \gamma \quad \text{and} \quad s_i^k \ge \gamma \quad \forall i = 1, \dots, n, \;\; \forall k \in \mathbb{N}, \tag{5.17}$$

where $\gamma \in \mathbb{R}$ denotes a suitable (possibly negative) constant. On the other hand, the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ is unbounded by assumption. This implies that there is at least one component $i \in \{1, \dots, n\}$ such that $x_i^k \longrightarrow +\infty$ or $s_i^k \longrightarrow +\infty$ on a subsequence, since otherwise the two sequences $\{x^k\}$ and $\{s^k\}$ would be bounded. This, in turn, would imply the boundedness of the sequence $\{\lambda^k\}$ as well because we have $A^T\lambda^k + s^k = c$ for all $k \in \mathbb{N}$ and because A is assumed to have full rank.

Now let $\hat{w} = (\hat{x}, \hat{\lambda}, \hat{s}) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ be a strictly feasible point for (3.5) whose existence is guaranteed by our assumptions. Then, in particular, we have

$$A^T\hat{\lambda} + \hat{s} = c \quad \text{and} \quad A\hat{x} = b.$$

Since we also have

$$A^T\lambda^k + s^k = c \quad \text{and} \quad Ax^k = b$$

for all $k \in \mathbb{N}$, we get

$$A^T(\hat{\lambda} - \lambda^k) + (\hat{s} - s^k) = 0 \tag{5.18a}$$

and

$$A(\hat{x} - x^k) = 0 \tag{5.18b}$$

by subtracting these equations. Premultiplying (5.18a) with $(\hat{x} - x^k)^T$ and taking into account (5.18b) gives

$$(\hat{x} - x^k)^T(\hat{s} - s^k) = 0.$$

Reordering this equation, we obtain

$$\hat{s}^{\mathrm{T}} x^k + \hat{x}^{\mathrm{T}} s^k = (x^k)^{\mathrm{T}} s^k + \hat{x}^{\mathrm{T}} \hat{s} \tag{5.19}$$

for all $k \in \mathbb{N}$. Using (5.17) as well as $\hat{x} > 0$ and $\hat{s} > 0$ in view of the strict feasibility of the vector $\hat{w} = (\hat{x}, \hat{\lambda}, \hat{s})$, it follows from (5.19) and the fact that $x_i^k \longrightarrow +\infty$ or $s_i^k \longrightarrow +\infty$ on a subsequence for at least one index $i \in \{1, \dots, n\}$ that

$$(x^k)^{\mathrm{T}} s^k \longrightarrow +\infty.$$

Hence there exists a component $j \in \{1, \dots, n\}$ (independent of k) such that

$$x_j^k s_j^k \longrightarrow +\infty$$

on a suitable subsequence. Using (5.17), this implies

$$\max\left\{0, x_j^k\right\} \max\left\{0, s_j^k\right\} \longrightarrow +\infty$$

and therefore

$$\varphi(x_j^k, s_j^k) \longrightarrow +\infty$$

on this subsequence. This, in turn, gives $\|\Phi(w^k)\| \longrightarrow +\infty$, a contradiction to $w^k \in \mathcal{L}_c$ for all $k \in \mathbb{N}$. ■

We next state a simple consequence of Proposition 5.7.

Corollary 5.8 *Let φ be the penalized minimum or penalized Fischer-Burmeister function, and consider the set*

$$\mathcal{L}_{c,\tau} := \left\{ w = (x, \lambda, s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \;\middle|\; A^{\mathrm{T}} \lambda + s = c,\ Ax = b,\ \|\Phi_\tau(w)\| \le c \right\},$$

where $c \in \mathbb{R}$ and $\tau \ge 0$ are any given constants. Assume that there exists a strictly feasible point for the optimality conditions (3.5)*. Then the set $\mathcal{L}_{c,\tau}$ is bounded.*

Proof. Let $w \in \mathcal{L}_{c,\tau}$ be given. Using Lemma 5.5 and the triangle inequality, it follows that

$$\|\Phi(w)\| \le \|\Phi(w) - \Phi_\tau(w)\| + \|\Phi_\tau(w)\| \le \kappa\tau + c.$$

Hence we have $w \in \mathcal{L}_{\bar{c}}$, where $\bar{c} := \kappa\tau + c$ and $\mathcal{L}_{\bar{c}}$ denotes the set defined in Proposition 5.7. This shows that the inclusion

$$\mathcal{L}_{c,\tau} \subseteq \mathcal{L}_{\bar{c}}$$

holds. Since the set on the right-hand side is compact by Proposition 5.7, the statement follows. ■

The proof of the previous result actually shows that the following uniform boundedness result holds.

Corollary 5.9 *Let φ be the penalized minimum function or the penalized Fischer-Burmeister function, and let $\mathcal{L}_{c,\tau}$ be the set defined in Corollary 5.8. Then the set*

$$\bigcup_{\tau\in[0,\bar{\tau}]} \mathcal{L}_{c,\tau}$$

is bounded for any $c \in \mathbb{R}$ and any $\bar{\tau} \geq 0$.

As mentioned above of this section, Proposition 5.7 and its corollaries will guarantee the existence of accumulation points when using one of the penalized functions.

For the remaining part of this section, we now investigate the question whether the smooth equation

$$\Phi_\tau(w) = 0 \tag{5.20}$$

has a solution for each $\tau > 0$.

The answer is relatively simple if φ denotes an NCP-function. For this class of functions, Kanzow [32] noted the following:

Theorem 5.10 *Let Φ_τ be defined using the Chen-Harker-Kanzow-Smale smoothing function (smoothed minimum function) or the smoothed Fischer-Burmeister function. Then the vector $w_\tau := (x_\tau, \lambda_\tau, s_\tau)$ solves the non-linear system of equations $\Phi_\tau(w) = 0$ if and only if w_τ satisfies the conditions*

$$\begin{aligned} A^{\mathrm{T}}\lambda + s &= c \\ Ax &= b, \\ x_i > 0,\ s_i > 0,\ x_i s_i &= \tau^2 \ \textit{for all } i = 1,\dots,n. \end{aligned} \tag{5.21}$$

Proof. It is easily verified that both the Chen-Harker-Kanzow-Smale smoothing function and the smoothed Fischer-Burmeister function ϕ_τ have the following property:

$$\phi_\tau(a,b) = 0 \quad \Longleftrightarrow \quad a > 0,\ b > 0,\ ab = \tau^2.$$

The proposition then follows directly from the definition of Φ_τ. ■

These conditions, however, are precisely the *central path conditions* used by basically all interior-point methods. By Theorem 4.3 it is known that they have a unique solution for every $\tau > 0$ if there exists a strictly feasible vector $\hat{w} = (\hat{x}, \hat{\lambda}, \hat{s})$ for the optimality conditions (3.5). Hence we now turn our attention to the two penalized functions. The next result states that the system (5.20) has a solution in this case, too.

Proposition 5.11 *Let φ be the penalized minimum or penalized Fischer-Burmeister function. Assume that there is a strictly feasible vector for the optimality conditions* (3.5)*. Then the system* (5.20) *has a solution for every $\tau > 0$.*

Proof. Let $\tau > 0$ be fixed, and consider the optimization problem

$$\min \Psi_\tau(w) \quad \text{subject to} \quad A^{\mathrm{T}}\lambda + s = c, \;\; Ax = b, \tag{5.22}$$

where $\Psi_\tau \colon \mathbb{R}^n \longrightarrow \mathbb{R}$ denotes the function

$$\Psi_\tau(w) := \frac{1}{2}\|\Phi_\tau(w)\|^2.$$

The set $\mathcal{L}_{c,\tau}$ defined in Corollary 5.8 is non-empty (for sufficiently large $c > 0$) and compact. Hence the above optimization problem has a global minimum $w_\tau = (x_\tau, \lambda_\tau, s_\tau)$. We will show that w_τ is a solution of the non-linear system (5.20).

To this end, note that (5.22) is a linearly constrained optimization problem. Hence there exist Lagrange multipliers $\eta \in \mathbb{R}^m$ and $\mu \in \mathbb{R}^n$ such that (w_τ, η, μ) satisfies the KKT conditions of (5.22):

$$\nabla\Psi_\tau(w_\tau) + \begin{bmatrix} A^{\mathrm{T}}\eta \\ A\mu \\ \mu \end{bmatrix} = 0.$$

Since $\nabla\Psi_\tau(w_\tau) = \Phi_\tau'(w_\tau)^{\mathrm{T}}\Phi_\tau(w_\tau)$, we can use the expression (5.12) for the Jacobian in order to rewrite the above equation as

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & D_{a,\tau} \\ A & 0 & 0 \\ I & 0 & D_{b,\tau} \end{bmatrix} \begin{bmatrix} A^{\mathrm{T}}\lambda_\tau + s_\tau - c \\ Ax_\tau - b \\ \phi_\tau(x_\tau, s_\tau) \end{bmatrix} + \begin{bmatrix} A^{\mathrm{T}}\eta \\ A\mu \\ \mu \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

where the diagonal matrices $D_{a,\tau}$ and $D_{b,\tau}$ are defined in a way similar to (5.13). In addition, w_τ is feasible for (5.22). Consequently, the previous conditions can be simplified to

$$D_{a,\tau}\phi_\tau(x_\tau, s_\tau) = -A^{\mathrm{T}}\eta, \;\; A\mu = 0, \;\; D_{b,\tau}\phi_\tau(x_\tau, s_\tau) + \mu = 0.$$

Multiplying the third equation with A, taking into account the second equation, and replacing $\phi_\tau(x_\tau, s_\tau)$ from the first equation gives

$$AD_{a,\tau}^{-1}D_{b,\tau}A^{\mathrm{T}}\eta = 0.$$

Since A has full rank and $D_{a,\tau}, D_{b,\tau} \in \mathbb{R}^{n\times n}$ are positive definite diagonal matrices, it follows that $\eta = 0$. This, in turn, implies $\phi_\tau(x_\tau, s_\tau) = 0$. Using once again the fact that $w_\tau = (x_\tau, \lambda_\tau, s_\tau)$ satisfies the linear constraints in (5.22), we see that the vector w_τ is indeed a solution of the non-linear system of equations (5.20). ■

Even though Proposition 5.11 guarantees the existence of a solution of the system (5.20) also for the two penalized functions, it is currently not clear whether this solution is always unique. In view of Proposition 5.6, any such solution must be an isolated solution, however, this does not necessarily exclude multiple solutions. In any case, the existence of such a solution is used here only for motivational purposes since our smoothing-type methods try to follow the corresponding *smoothing path*

$$\tau \mapsto w_\tau,$$

where w_τ denotes a (unique) solution of (5.20). However, the convergence theory for our methods still works even if this system may not have a solution (since, in any case, we follow this smoothing path only inexactly, i. e., we require only inexact solutions of the systems (5.20).

6 Jacobian Smoothing Methods

We are now in a position to state a Jacobian smoothing method for the solution of a linear program's optimality conditions (3.5). Basically, this method is a Newton-type method for the solution of the system $\Phi(x, \lambda, s) = 0$. However, instead of solving the corresponding Newton equation

$$\Phi'(x^k, \lambda^k, s^k) \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = -\Phi(x^k, \lambda^k, s^k)$$

at each iteration with a possibly singular or not existing Jacobian $\Phi'(x^k, \lambda^k, s^k)$, we solve a linear system of the form

$$\Phi'_{\tau_k}(x^k, \lambda^k, s^k) \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = -\Phi(x^k, \lambda^k, s^k)$$

for some $\tau_k > 0$ and the operator Φ_τ introduced in (5.10) . This guarantees that the Jacobian of Φ_{τ_k} exists; moreover, as we will see later, this matrix is always non-singular. The search direction computed in this way is, generally, not a descent direction for the natural merit function

$$\Psi(x, \lambda, s) := \frac{1}{2}\Phi(x, \lambda, s)^{\mathrm{T}}\Phi(x, \lambda, s) = \frac{1}{2}\|\Phi(x, \lambda, s)\|^2,$$

but it turns out to be a descent direction for the smoothed merit function

$$\Psi_\tau(x, \lambda, s) := \frac{1}{2}\Phi_\tau(x, \lambda, s)^{\mathrm{T}}\Phi_\tau(x, \lambda, s) = \frac{1}{2}\|\Phi_\tau(x, \lambda, s)\|^2, \tag{6.1}$$

with $\tau = \tau_k$.

The precise algorithm is as follows:

Algorithm 6.1 (Jacobian Smoothing Method)
Step 0 (Initialization):

Choose $\varrho, \alpha, \eta \in (0, 1), \varepsilon > 0, \sigma \in (0, \frac{1}{2}(1 - \alpha))$ and $w^0 := (x^0, \lambda^0, s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$. Set $\beta_0 := \|\Phi(w^0)\|, \tau_0 := \frac{\alpha}{2\kappa}\beta_0$ and set the iteration counter $k := 0$.

Step 1 (Termination Criterion):

If $\|\Phi(w^k)\| \le \varepsilon$: Stop.

Step 2 (Search Direction):

Compute a solution $\Delta w^k = (\Delta x^k, \Delta\lambda^k, \Delta s^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ of the linear system

$$\Phi'_{\tau_k}(w^k)\Delta w = -\Phi(w^k). \tag{6.2}$$

Step 3 (Line Search):
Compute $t_k = \max\left\{\varrho^\ell \;\middle|\; \ell = 0,1,2,\dots\right\}$ such that

$$\Psi_{\tau_k}(w^k + t_k\Delta w^k) \le \Psi_{\tau_k}(w^k) - 2\sigma t_k \Psi(w^k) \tag{6.3}$$

and set $w^{k+1} := w^k + t_k\Delta w^k$.

Step 4 (Update of Smoothing Parameter):
If

$$\|\Phi(w^{k+1})\| \le \max\left\{\eta\beta_k, \frac{\|\Phi(w^{k+1}) - \Phi_{\tau_k}(w^{k+1})\|}{\alpha}\right\}, \tag{6.4}$$

then set

$$\beta_{k+1} := \|\Phi(w^{k+1})\| \tag{6.5}$$

and choose τ_{k+1} such that

$$\tau_{k+1} \in \left(0, \min\left\{\frac{\alpha}{2\kappa}\beta_{k+1}, \frac{\tau_k}{2}\right\}\right). \tag{6.6}$$

Otherwise (i. e., if (6.4) is not satisfied) set $\beta_{k+1} := \beta_k$ and $\tau_{k+1} := \tau_k$.

Step 5: Set $k \longleftarrow k+1$ and go to Step 1.

For a better understanding, let us add some comments on the algorithm. Step 0 of Algorithm 6.1 is the initialization (with κ being the constant from Lemma 5.5), while Step 1 contains the termination criterion. Step 2 computes the Newton-type search direction which is the main computational effort of Algorithm 6.1. Step 3 then calculates a step size by using an Armijo-type condition for the smoothed merit function Ψ_{τ_k}. Step 4 contains the updating rule for the smoothing parameter τ_k. This updating rule looks somewhat complicated, however, it is exactly the rule that is required for the theoretical analysis. More precisely, the current updating rule for τ_k will be used in order to establish a global convergence result for Algorithm 6.1, see Section 6.1 for further details. To guarantee fast local convergence, however, one has to impose some further conditions on the choice of τ_k. Loosely speaking, τ_k has to go to zero sufficiently fast. This issue will be discussed in more detail in Section 6.2.

6.1 Global Convergence

Throughout this section, we assume that the termination parameter ε is equal to zero and that Algorithm 6.1 does not terminate in a finite number of iterations. Under this assumption, we will show that Algorithm 6.1 is well-defined and globally convergent in the sense that any accumulation point of a sequence generated by the Jacobian smoothing method is a solution of the optimality conditions (3.5).

We next state a technical inequality which, however, will turn out to be very helpful in our subsequent analysis.

Lemma 6.2 *Let $\{w^k\}$ be a sequence generated by Algorithm 6.1, and let α be the parameter defined in the initialization phase of Algorithm 6.1. Then*

$$\|\Phi(w^k) - \Phi_{\tau_k}(w^k)\| \le \alpha\|\Phi(w^k)\|$$

holds for all $k \in \mathbb{N}$.

Proof. Define the index set

$$K := \{0\} \cup \left\{k \in \mathbb{N} \;\middle|\; \|\Phi(w^k)\| \le \max\left\{\eta\beta_{k-1},\ \frac{1}{\alpha}\|\Phi(w^k) - \Phi_{\tau_{k-1}}(w^k)\|\right\}\right\} \tag{6.7}$$

It is necessary to distinguish two cases:

Case 1. $k \in K$:
We obtain from (6.6) and Lemma 5.5

$$\|\Phi(w^k) - \Phi_{\tau_k}(w^k)\| \le \kappa\tau_k \le \frac{\alpha}{2}\beta_k \le \alpha\beta_k = \alpha\|\Phi(w^k)\|$$

Case 2. $k \notin K$:
In this case, we have $\tau_k = \tau_{k-1}$, so that we obtain from (6.4)

$$\|\Phi(w^k) - \Phi_{\tau_k}(w^k)\| = \|\Phi(w^k) - \Phi_{\tau_{k-1}}(w^k)\| < \alpha\|\Phi(w^k)\|,$$

and this completes the proof. ■

The following result guarantees that the line search in Step 3 of Algorithm 6.1 is well-defined. Its proof is quite similar to the ones used in Chen, Qi, and Sun [14] or Kanzow and Pieper [35] and is based on Lemma 6.2.

Proposition 6.3 *At each iteration k, there exists a finite exponent ℓ_k such that the step size $t_k = \varrho^{\ell_k}$ satisfies the line search criterion* (6.3).

Proof. The continuous differentiability of $\Phi_\tau(\cdot)$ implies that Ψ_τ is also continuously differentiable and that $\nabla\Psi(w^k) = \Phi'_{\tau_k}(w^k)^T\Phi_{\tau_k}(w^k)$. By the construction of Algorithm 6.1 $\Phi'_{\tau_k}(w^k)\Delta w^k = -\Phi(w^k)$. Taking into account Lemma 6.2 we have

$$\begin{aligned}
\Psi_{\tau_k}(w^k + t_k\Delta w^k) - \Psi_{\tau_k}(w^k) &= t_k\nabla\Psi_{\tau_k}(w^k)^T\Delta w^k + o(t_k)\\
&= t_k\Phi_{\tau_k}(w^k)^T\Phi'_{\tau_k}(w^k)\Delta w + o(t_k)\\
&= -t_k\Phi_{\tau_k}(w^k)^T\Phi(w^k) + o(t_k)\\
&= -t_k\Phi(w^k)^T\Phi_{\tau_k}(w^k) + o(t_k)\\
&= -2t_k\Psi(w^k)\\
&\qquad + t_k\Phi(w^k)^T\left[\Phi(w^k) - \Phi_{\tau_k}(w^k)\right] + o(t_k)\\
&\le -2t_k\Psi(w^k) + 2t_k\alpha\Psi(w^k) + o(t_k)\\
&= -2t_k(1-\alpha)\Psi(w^k) + o(t_k)
\end{aligned}$$

Since $\sigma < \frac{1}{2}(1-\alpha) < 1-\alpha$ there exists a finite non-negative integer ℓ_k such that (6.3) holds. ∎

Propositions 5.6 and 6.3 together imply that Algorithm 6.1 is well-defined. For the proof of its global convergence we will use the index set K from (6.7) again.

Our next result says that, although $\|\Phi(w^k)\|$ does not necessarily decrease monotonically (since our line search is based on $\|\Phi_{\tau_k}(\cdot)\|$), a possible increase cannot be too dramatic.

Proposition 6.4 *All iterates $w^k = (x^k, \lambda^k, s^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ generated by Algorithm 6.1 belong to the level set*

$$\mathcal{L} := \left\{ w \;\middle|\; \|\Phi(w)\| \le (1+\alpha)\|\Phi(w^0)\| \right\}.$$

Proof. The proof is essentially the same as the ones given in [14] and [35] for the corresponding results in that papers. Nevertheless, it is included here not only for the sake of completeness, but also, because we will derive an important inequality which will be used in the subsequent analysis.

Let us partition the index set K from (6.7) into $K = \{0\} \cup K_1 \cup K_2$, where

$$K_1 := \left\{ k \in K \;\middle|\; \eta\beta_{k-1} \ge \frac{\|\Phi_{\tau_{k-1}}(w^k) - \Phi(w^k)\|}{\alpha} \right\}$$

and

$$K_2 := \left\{ k \in K \;\middle|\; \eta\beta_{k-1} < \frac{\|\Phi_{\tau_{k-1}}(w^k) - \Phi(w^k)\|}{\alpha} \right\}.$$

Assume that K consists of $k_0 = 0 < k_1 < k_2 < \cdots$ (note that K might be finite or infinite). Let $k \in \mathbb{N}$ be arbitrarily given and k_j be the largest number in K such that $k_j \le k$. Then we have

$$\tau_k = \tau_{k_j} \quad \text{and} \quad \beta_k = \beta_{k_j} = \|\Phi(w^{k_j})\|$$

in view of our updating rules in Step 4 of Algorithm 6.1. Using the line search criterion (6.3), we have

$$\|\Phi_{\tau_{k_j}}(w^k)\| \le \|\Phi_{\tau_{k_j}}(w^{k_j})\|.$$

Hence we obtain from Lemma 5.5

$$\begin{aligned}
\|\Phi(w^k)\| &\le \|\Phi_{\tau_k}(w^k)\| + \|\Phi(w^k) - \Phi_{\tau_k}(w^k)\| \\
&= \|\Phi_{\tau_{k_j}}(w^k)\| + \|\Phi(w^k) - \Phi_{\tau_{k_j}}(w^k)\| \\
&\le \|\Phi_{\tau_{k_j}}(w^k)\| + \kappa\tau_{k_j} \\
&\le \|\Phi_{\tau_{k_j}}(w^{k_j})\| + \kappa\tau_{k_j} \qquad\qquad (6.8) \\
&\le \|\Phi(w^{k_j})\| + \|\Phi_{\tau_{k_j}}(w^{k_j}) - \Phi(w^{k_j})\| + \kappa\tau_{k_j} \\
&\le \|\Phi(w^{k_j})\| + \kappa\tau_{k_j} + \kappa\tau_{k_j} \\
&= \beta_{k_j} + 2\kappa\tau_{k_j}.
\end{aligned}$$

Now, if $j = 0$, we have $\beta_{k_j} = \beta_0$ and $\tau_{k_j} = \tau_0$ and therefore

$$\|\Phi(w^k)\| \le \beta_0 + 2\kappa\tau_0 = (1+\alpha)\|\Phi(w^0)\|$$

from (6.8) and Step 0 of Algorithm 6.1. On the other hand, if $j \ge 1$, we get from Step 4 of Algorithm 6.1 that

$$\tau_{k_j} \le \frac{1}{2}\tau_{k_j-1} = \frac{1}{2}\tau_{k_{j-1}}$$

and either

$$\beta_{k_j} \le \eta\beta_{k_j-1} = \eta\beta_{k_{j-1}} \quad \text{if } k_j \in K_1$$

or, using Lemma 5.5 again,

$$\beta_{k_j} \le \frac{\|\Phi_{\tau_{k_j-1}}(w^{k_j}) - \Phi(w^{k_j})\|}{\alpha} \le \frac{\kappa}{\alpha}\tau_{k_j-1} = \frac{\kappa}{\alpha}\tau_{k_{j-1}} \le \frac{1}{2}\beta_{k_{j-1}} \quad \text{if } k_j \in K_2.$$

Let us define

$$r := \max\left\{\frac{1}{2}, \eta\right\}.$$

Then it follows from the definitions of τ_0 and β_0 that, for $j \ge 1$, we have

$$\tau_{k_j} \le \frac{1}{2^{j-1}}\tau_0 = \frac{1}{2^j}\frac{\alpha}{\kappa}\|\Phi(w^0)\| \tag{6.9}$$

and

$$\beta_{k_j} \le r^{j-1}\beta_0 = r^{j-1}\|\Phi(w^0)\|. \tag{6.10}$$

Therefore, using (6.8) and $r \ge 1/2$, we obtain for $j \ge 1$

$$\|\Phi(w^k)\| \le \left(r^{j-1} + \frac{\alpha}{2^{j-1}}\right)\|\Phi(w^0)\| \le r^{j-1}(1+\alpha)\|\Phi(w^0)\|. \tag{6.11}$$

This shows that the inequality

$$\|\Phi(w^k)\| \le (1+\alpha)\|\Phi(w^0)\|$$

holds in any case. ■

We are now in the position to prove our main global convergence result.

Theorem 6.5 *Every accumulation point of a sequence* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ *generated by Algorithm 6.1 is a solution of the optimality conditions* (3.5).

Proof. If the index set K is infinite, then the statement follows immediately from (6.11). So consider the case where K is finite. Let $\hat{k}$ be the largest number in K. Then it follows from our updating rules in Step 4 of Algorithm 6.1 that the following relations hold for all $k \geq \hat{k}$:

$$\tau_k = \tau_{\hat{k}}, \tag{6.12a}$$

$$\beta_k = \beta_{\hat{k}} = \|\Phi(w^{\hat{k}})\|, \tag{6.12b}$$

$$\|\Phi(w^k)\| > \eta\beta_k = \eta\|\Phi(w^{\hat{k}})\| > 0, \tag{6.12c}$$

$$\alpha\|\Phi(w^k)\| > \|\Phi_{\tau_{\hat{k}}}(w^k) - \Phi(w^k)\|. \tag{6.12d}$$

Let w^* be an accumulation point of $\{w^k\}$ and $\{w^k\}_L$ be a subsequence converging to w^*. Without loss of generality, we assume that $k \geq \hat{k}$ for all $k \in L$. Let

$$t_* := \liminf_{k \in L} t_k$$

(note that we take the limes inferior on the subset L only). We now distinguish two cases.

Case 1: $t_* > 0$.
Since $t_k \geq t_*/2$ for all $k \in L$ sufficiently large, we then get from the line search rule that

$$\Psi_{\tau_{\hat{k}}}(w^{k+1}) - \Psi_{\tau_{\hat{k}}}(w^k) \leq -2\sigma t_k \Psi(w^k) \leq -\sigma t_* \Psi(w^k) < 0. \tag{6.13}$$

On the other hand (since $\tau_{\hat{k}}$ is fixed), the merit function $\Psi_{\tau_{\hat{k}}}$ is bounded from below (by zero) and $\{\Psi_{\tau_{\hat{k}}}(w^k)\}_{k\in\mathbb{N}}$ is a monotonically decreasing sequence. Hence $\{\Psi_{\tau_{\hat{k}}}(w^k)\}$ is convergent. In particular, we therefore have

$$\Psi_{\tau_{\hat{k}}}(w^{k+1}) - \Psi_{\tau_{\hat{k}}}(w^k) \longrightarrow 0 \quad \text{for } k \longrightarrow \infty.$$

Hence, taking the limit $k \longrightarrow \infty$ for $k \in L$ in (6.13), we get

$$0 \leq -\sigma t_* \Psi(w^*) \leq 0$$

and therefore $\Psi(w^*) = 0$.

Case 2: $t_* = 0$.
Since $\tau_k = \tau_{\hat{k}} > 0$ for all $k \in L$, $\Phi'_{\tau_{\hat{k}}}(w^*)$ is non-singular by Proposition 5.6, and $\{w^k\}_{k\in L} \longrightarrow w^*$, it follows from Lemma 3.14 that $\Phi'_{\tau_k}(w^k)$ is non-singular with

$$\|\Phi'_{\tau_k}(w^k)^{-1}\| \leq \kappa_1$$

for all $k \in L$ sufficiently large and a suitable constant $\kappa_1 > 0$. This implies

$$\begin{aligned}\|\Delta w^k\| &\le \|\Phi'_{\tau_k}(w^k)^{-1}\Phi(w^k)\| \\ &\le \kappa_1 \|\Phi(w^k)\| \\ &\le \kappa_1(1+\alpha)\|\Phi(w^0)\| \\ &=: \kappa_2\end{aligned}$$

for all large enough $k \in L$ by Step 2 of Algorithm 6.1 and Proposition 6.4. Hence we can assume without loss of generality that

$$\{\Delta w^k\}_{k\in L} \longrightarrow \Delta w^*$$

for some vector $\Delta w^* \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$. On the other hand, subsequencing if necessary, we have $\{t_k\}_{k\in L} \longrightarrow 0$. Therefore the step size t_k/ϱ does not satisfy the Armijo-like condition (6.3), i. e., we have

$$-2\sigma\frac{t_k}{\varrho}\Psi(w^k) < \Psi_{\tau_{\hat{k}}}(w^k + \frac{t_k}{\varrho}\Delta w^k) - \Psi_{\tau_{\hat{k}}}(w^k)$$

for all $k \in L$ sufficiently large. Using the Cauchy-Schwarz inequality and Lemma 6.2, this implies

$$\begin{aligned}
&-2\sigma\Psi(w^k) \\
&\quad< \frac{\Psi_{\tau_{\hat{k}}}(w^k + \frac{t_k}{\varrho}\Delta w^k) - \Psi_{\tau_{\hat{k}}}(w^k)}{\frac{t_k}{\varrho}} \\
&\quad= \nabla\Psi_{\tau_{\hat{k}}}(w^k)^{\mathrm{T}}\Delta w^k \\
&\qquad\quad+ \left(\frac{\Psi_{\tau_{\hat{k}}}(w^k + \frac{t_k}{\varrho}\Delta w^k) - \Psi_{\tau_{\hat{k}}}(w^k)}{\frac{t_k}{\varrho}} - \nabla\Psi_{\tau_{\hat{k}}}(w^k)^{\mathrm{T}}\Delta w^k\right) \\
&\quad= -\Phi_{\tau_{\hat{k}}}(w^k)^{\mathrm{T}}\Phi(w^k) \\
&\qquad\quad+ \left(\frac{\Psi_{\tau_{\hat{k}}}(w^k + \frac{t_k}{\varrho}\Delta w^k) - \Psi_{\tau_{\hat{k}}}(w^k)}{\frac{t_k}{\varrho}} - \nabla\Psi_{\tau_{\hat{k}}}(w^k)^{\mathrm{T}}\Delta w^k\right) \\
&\quad= -2\Psi(w^k) + \Phi(w^k)^{\mathrm{T}}\left(\Phi(w^k) - \Phi_{\tau_{\hat{k}}}(w^k)\right) \\
&\qquad\quad+ \left(\frac{\Psi_{\tau_{\hat{k}}}(w^k + \frac{t_k}{\varrho}\Delta w^k) - \Psi_{\tau_{\hat{k}}}(w^k)}{\frac{t_k}{\varrho}} - \nabla\Psi_{\tau_{\hat{k}}}(w^k)^{\mathrm{T}}\Delta w^k\right) \\
&\quad\le -2\Psi(w^k) + 2\alpha\Psi(w^k) \\
&\qquad\quad+ \left(\frac{\Psi_{\tau_{\hat{k}}}(w^k + \frac{t_k}{\varrho}\Delta w^k) - \Psi_{\tau_{\hat{k}}}(w^k)}{\frac{t_k}{\varrho}} - \nabla\Psi_{\tau_{\hat{k}}}(w^k)^{\mathrm{T}}\Delta w^k\right).
\end{aligned}$$

Taking the limit $k \longrightarrow \infty$ for $k \in L$ and using $w^k \longrightarrow w^*$, $\Delta w^k \longrightarrow \Delta w^*$, and $t_k \longrightarrow 0$ on this subsequence, we obtain from the continuous differentiability of $\Psi_{\tau_{\hat{k}}}$ that

$$\begin{aligned} &-2\sigma\Psi(w^*) \\ &\qquad \le -2\Psi(w^*) + 2\alpha\Psi(w^*) + \nabla\Psi_{\tau_{\hat{k}}}(w^*)^{\mathrm{T}}\Delta w^* - \nabla\Psi_{\tau_{\hat{k}}}(w^*)^{\mathrm{T}}\Delta w^* \end{aligned}$$

and therefore

$$\sigma\Psi(w^*) \ge (1-\alpha)\Psi(w^*).$$

By our choice of σ in Step 0 of Algorithm 6.1, this implies $\Psi(w^*) = 0$.

We therefore obtain in both cases that $\Psi(w^*) = 0$ and therefore also $\Phi(w^*) = 0$. However, by (6.12c) and continuity, we have $\|\Phi(w^*)\| > 0$. This contradiction shows that K is infinite, so the proof is complete. ∎

Note that the previous proof showed the following: Whenever the sequence $\{w^k\}$ generated by Algorithm 6.1 has an accumulation point, then the index set K is infinite. Hence K can be finite only if $\{w^k\}$ is unbounded.

6.2 Rate of Convergence

The aim of this section is to prove local quadratic convergence of the Jacobian smoothing method from Algorithm 6.1 under suitable assumptions. To this end, we first establish the following result which is also of interest by its own.

Theorem 6.6 *Let $w^* = (x^*, \lambda^*, s^*)$ be a solution of the optimality conditions* (3.5) *and let Φ be defined by the minimum or the Fischer-Burmeister function. Then the following statements are equivalent*

(1) The operator Φ is continuously differentiable at w^*, and the Jacobian $\Phi'(w^*)$ is non-singular.

(2) w^* is the unique solution of the optimality conditions (3.5).

Proof.

(1) $\Longrightarrow$ (2):

Let Φ be continuously differentiable at w^* with $\Phi'(w^*)$ being non-singular and let $w = (x, \lambda, s)$. Then, by Lemma 3.15

$$\|\Phi(w)\| \ge c\|w^* - w\|$$

for all w sufficiently close to w^*. This inequality shows that, locally, w^* is the unique solution of the optimality conditions (3.5). However, since it is easy to see that the solution set of (3.5) is convex, it follows that w^* is the unique solution of (3.5) also from a global point of view.

(2) $\Longrightarrow$ (1):

By the Goldman-Tucker Theorem 3.12, the unique solution w^* of the optimality conditions (3.5) satisfies the strict complementarity condition

$$x_i^* + s_i^* > 0 \quad \text{for } i = 1, \dots, n.$$

This implies that Φ is continuously differentiable at w^*.

In order to see that the Jacobian of Φ is non-singular at the point w^*, we assume throughout this proof that φ denotes the Fischer-Burmeister function from (5.2). The proof for the minimum function from (5.3) is very similar and therefore omitted here.

Consider the two index sets $\mathcal{B}$ and $\mathcal{N}$ defined in (3.7). By Lemma 3.13 (2) we have

$$\mathcal{N} = \{1, \dots, n\} \setminus \mathcal{B}.$$

Note that the Jacobian of $\Phi(w^*)$ has the form

$$\Phi'(w^*) = \begin{bmatrix} 0 & A^{\mathrm{T}} & I \\ A & 0 & 0 \\ D_a & 0 & D_b \end{bmatrix}$$

with

$$D_a := \operatorname{diag}\left(\dots, \frac{\partial \varphi}{\partial a}(x_i^*, s_i^*), \dots\right),$$
$$D_b := \operatorname{diag}\left(\dots, \frac{\partial \varphi}{\partial b}(x_i^*, s_i^*), \dots\right).$$

The definitions of the index sets $\mathcal{B}$ and $\mathcal{N}$ together with the definition of the Fischer-Burmeister function shows that

$$\frac{\partial \varphi}{\partial a}(x_i^*, s_i^*) = 1 - \frac{x_i^*}{\sqrt{(x_i^*)^2 + (s_i^*)^2}} = \begin{cases} 0 & \text{if } i \in \mathcal{B}, \\ 1 & \text{if } i \in \mathcal{N}, \end{cases} \tag{6.14a}$$

$$\frac{\partial \varphi}{\partial b}(x_i^*, s_i^*) = 1 - \frac{s_i^*}{\sqrt{(x_i^*)^2 + (s_i^*)^2}} = \begin{cases} 1 & \text{if } i \in \mathcal{B}, \\ 0 & \text{if } i \in \mathcal{N}. \end{cases} \tag{6.14b}$$

Then $\Phi'(x^*, \lambda^*, s^*)q = 0$ for an appropriately partitioned vector $q = (q^{(1)}, q^{(2)}, q^{(3)})$ implies

$$A^{\mathrm{T}} q^{(2)} + q^{(3)} = 0, \tag{6.15a}$$
$$A q^{(1)} = 0, \tag{6.15b}$$
$$(D_a)_{\mathcal{B}} q_{\mathcal{B}}^{(1)} + (D_b)_{\mathcal{B}} q_{\mathcal{B}}^{(3)} = 0, \tag{6.15c}$$
$$(D_a)_{\mathcal{N}} q_{\mathcal{N}}^{(1)} + (D_b)_{\mathcal{N}} q_{\mathcal{N}}^{(3)} = 0, \tag{6.15d}$$

where $q_{\mathcal{B}}^{(1)}$ denotes the $|\mathcal{B}|$-dimensional subvector of $q^{(1)}$ consisting of the components $q_i^{(1)}$ $(i \in \mathcal{B})$; similarly, $(D_a)_{\mathcal{B}}$ denotes the $|\mathcal{B}| \times |\mathcal{B}|$-diagonal matrix containing the diagonal entries a_{ii} $(i \in \mathcal{B})$ from the matrix D_a. The other subvectors and submatrices occurring in the above formulas are defined analogously.

Using (6.14a), (6.14b), (6.15c), and (6.15d), we obtain

$$q_{\mathcal{B}}^{(3)} = 0 \quad \text{and} \quad q_{\mathcal{N}}^{(1)} = 0. \tag{6.16}$$

We will use (6.16) in order to show that the vector

$$(x^*(t), \lambda^*(t), s^*(t)) := (x^*, \lambda^*, s^*) + t(q^{(1)}, q^{(2)}, q^{(3)})$$

is also a solution of the optimality conditions (3.5) for all $t > 0$ sufficiently small. This then implies that $q = (q^{(1)}, q^{(2)}, q^{(3)}) = (0,0,0)$ since (x^*, λ^*, s^*) was assumed to be the only solution of (3.5).

Obviously, the equations $Ax^*(t) = b$ and $A^{\mathrm{T}}\lambda^*(t) + s^*(t) = c$ are satisfied for any $t > 0$ in view of (6.15b) and (6.15a), respectively. Moreover, $x^*(t) \geq 0$ and $s^*(t) \geq 0$ for all $t > 0$ sufficiently small follows from the definitions of the index sets $\mathcal{B}$ and $\mathcal{N}$ together with (6.16). Finally, we also have

$$\begin{aligned}(x^*(t))^{\mathrm{T}}(s^*(t)) &= \\ &\left(x_{\mathcal{B}}^* + tq_{\mathcal{B}}^{(1)}\right)^{\mathrm{T}}\left(s_{\mathcal{B}}^* + tq_{\mathcal{B}}^{(3)}\right) + \left(x_{\mathcal{N}}^* + tq_{\mathcal{N}}^{(1)}\right)^{\mathrm{T}}\left(s_{\mathcal{N}}^* + tq_{\mathcal{N}}^{(3)}\right) = 0\end{aligned} \tag{6.17}$$

since (x^*, λ^*, s^*) satisfies the optimality conditions (3.5), $x_{\mathcal{N}}^* = 0$, $s_{\mathcal{B}}^* = 0$ and because of (6.16). This completes the proof. ■

Theorem 6.6 can also be derived from a similar result by Burke and Xu [8], where the authors consider a smoothing-type method for linear complementarity problems based on the minimum function.

Theorem 6.6 can be interpreted in the following way: Assume that a sequence $\{(x^k, \lambda^k, s^k)\}$ generated by Algorithm 6.1 converges to a solution (x^*, λ^*, s^*) satisfying the strict complementarity condition $x_i^* + s_i^* > 0$ for all $i = 1, \ldots, n$, so that Φ is continuously differentiable around this solution point. Then Theorem 6.6 states that the sequence of Jacobian matrices $\{\Phi'(x^k, \lambda^k, s^k)\}$ converges to a singular matrix whenever (x^*, λ^*, s^*) is not the unique solution of (3.5).

In the remaining part of this section, we want to show that Algorithm 6.1 is locally quadratically convergent if one of the two equivalent conditions from Theorem 6.6 is satisfied and if the smoothing parameter τ_k is updated in an appropriate way. The latter is made more precise in our next two results.

In order to motivate this result, assume that Φ is continuously differentiable at a point (x, λ, s). Then $\Phi'_\tau(x, \lambda, s) \longrightarrow \Phi'(x, \lambda, s)$ for $\tau \longrightarrow 0$. Hence, for any $\delta > 0$, there exists a constant $\bar{\tau} > 0$ such that

$$\|\Phi'_\tau(x, \lambda, s) - \Phi'(x, \lambda, s)\| \leq \delta$$

for all $\tau \in [0, \bar{\tau}]$. However, the existence of such a constant $\bar{\tau}$ does not guarantee that this $\bar{\tau}$ can be computed easily. On the other hand, our local rate of convergence result assumes that such a constant is computable. Therefore, our next two results give explicit values of this $\bar{\tau}$ provided that the difference between the matrices $\Phi'_\tau(x, \lambda, s)$ and $\Phi'(x, \lambda, s)$ is measured in the Frobenius norm.

The first of these two results deals with the case where Φ and Φ_τ are defined by the Fischer-Burmeister function and its smooth counterpart.

Lemma 6.7 *Let Φ and Φ_τ be defined using the Fischer-Burmeister-type functions* (5.2) *and* (5.6)*, respectively. Furthermore, let $(x, \lambda, s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ be any vector with $x_i^2 + s_i^2 > 0$ for all $i = 1, \ldots, n$, and let $\delta > 0$ be arbitrarily given. Then we have*

$$\|\Phi'_\tau(x, \lambda, s) - \Phi'(x, \lambda, s)\|_F \leq \delta$$

for all $\tau \in [0, \bar{\tau}]$, where $\bar{\tau} = \bar{\tau}(x, s, \delta) > 0$ is given by

$$\bar{\tau}(x, s, \delta) := \frac{\delta \min\limits_{i=1,\ldots,n} \left\{x_i^2 + s_i^2\right\}}{2\sqrt{n \max\limits_{i=1,\ldots,n} \left\{x_i^2 + s_i^2\right\}}}.$$

Proof. Since $x_i^2 + s_i^2 > 0$ for all $i = 1, \ldots, n$, the mapping Φ is continuously differentiable at (x, λ, s). Hence its Jacobian $\Phi'(x, \lambda, s)$ exists at (x, λ, s), and an elementary calculation shows that

$$\|\Phi'_\tau(x, \lambda, s) - \Phi'(x, \lambda, s)\|_F^2 =$$
$$\sum_{i=1}^n \left(\frac{\partial \varphi_\tau}{\partial a}(x_i, s_i) - \frac{\partial \varphi}{\partial a}(x_i, s_i)\right)^2 + \sum_{i=1}^n \left(\frac{\partial \varphi_\tau}{\partial b}(x_i, s_i) - \frac{\partial \varphi}{\partial b}(x_i, s_i)\right)^2. \quad (6.18)$$

Throughout this proof, let us use the notation

$$\alpha_{xs} := \min_{i=1,\ldots,n} \{x_i^2 + s_i^2\} > 0.$$

Then we obtain

$$\begin{aligned}
\left|\frac{\partial \varphi_\tau}{\partial a}(x_i, s_i) - \frac{\partial \varphi}{\partial a}(x_i, s_i)\right| &= \left|\frac{x_i}{\sqrt{x_i^2 + s_i^2 + 2\tau^2}} - \frac{x_i}{\sqrt{x_i^2 + s_i^2}}\right| \\
&= |x_i| \left(\frac{1}{\sqrt{x_i^2 + s_i^2}} - \frac{1}{\sqrt{x_i^2 + s_i^2 + 2\tau^2}}\right) \\
&\leq |x_i| \left(\frac{1}{\sqrt{\alpha_{xs}}} - \frac{1}{\sqrt{\alpha_{xs} + 2\tau^2}}\right) \\
&= |x_i| \frac{\sqrt{\alpha_{xs} + 2\tau^2} - \sqrt{\alpha_{xs}}}{\sqrt{\alpha_{xs}}\sqrt{\alpha_{xs} + 2\tau^2}} \\
&\leq |x_i| \frac{\sqrt{2}\tau}{\alpha_{xs}}
\end{aligned}$$

for $i = 1, \dots, n$, where the first inequality follows from the fact that the function

$$f(a) := \frac{1}{\sqrt{a}} - \frac{1}{\sqrt{a + 2\tau^2}}$$

is strictly decreasing for $a > 0$ (since $f'(a) < 0$ for $a > 0$), and the second inequality follows from

$$\sqrt{a+b} \le \sqrt{a} + \sqrt{b}$$

for all $a, b \ge 0$. In a similar way, we get

$$\left| \frac{\partial \varphi_\tau}{\partial b}(x_i, s_i) - \frac{\partial \varphi}{\partial b}(x_i, s_i) \right| \le |s_i| \frac{\sqrt{2}\tau}{\alpha_{xs}}$$

for $i = 1, \dots, n$. Using the definition of $\bar{\tau}$, we therefore obtain for any $\tau \in [0, \bar{\tau}]$ and any $i \in \{1, \dots, n\}$:

$$\begin{aligned}
\left(\frac{\partial \varphi_\tau}{\partial a}(x_i, s_i) - \frac{\partial \varphi}{\partial a}(x_i, s_i) \right)^2 &\le \frac{x_i^2 2\tau^2}{\alpha_{xs}^2} \\
&\le \frac{2x_i^2 \bar{\tau}^2}{\alpha_{xs}^2} \\
&\le \frac{2x_i^2 \delta^2 \alpha_{xs}^2}{4n\alpha_{xs}^2 \max\limits_{i=1,\dots,n} \{x_i^2 + s_i^2\}} \\
&\le \frac{\delta^2}{2n}
\end{aligned}$$

and, similarly,

$$\left(\frac{\partial \varphi_\tau}{\partial b}(x_i, s_i) - \frac{\partial \varphi}{\partial b}(x_i, s_i) \right)^2 \le \frac{\delta^2}{2n}.$$

Using (6.18), this implies

$$\begin{aligned}
&\|\Phi_\tau'(x, \lambda, s) - \Phi'(x, \lambda, s)\|_F \\
&= \sqrt{\sum_{i=1}^n \left(\frac{\partial \varphi_\tau}{\partial a}(x_i, s_i) - \frac{\partial \varphi}{\partial a}(x_i, s_i) \right)^2 + \sum_{i=1}^n \left(\frac{\partial \varphi_\tau}{\partial b}(x_i, s_i) - \frac{\partial \varphi}{\partial b}(x_i, s_i) \right)^2} \\
&\le \sqrt{\frac{n\delta^2}{2n} + \frac{n\delta^2}{2n}} \\
&= \delta.
\end{aligned}$$

This completes the proof. ∎

The next result deals with the case where Φ and Φ_τ are defined by the minimum function and its smooth counterpart.

Lemma 6.8 *Let the operators Φ and Φ_τ be defined using the minimum-type functions* (5.3) *and* (5.7)*, respectively. Furthermore, let $(x, \lambda, s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ be any vector with $x_i \neq s_i$ for all $i = 1, \dots, n$, and let $\delta > 0$ be arbitrarily given. Then we have*

$$\|\Phi'_\tau(x, \lambda, s) - \Phi'(x, \lambda, s)\|_F \leq \delta$$

for all $\tau \in [0, \bar{\tau}]$, where $\bar{\tau} = \bar{\tau}(x, s, \delta) > 0$ is given by

$$\bar{\tau}(x, s, \delta) := \frac{\delta \min\limits_{i=1,\dots,n} \left\{(x_i - s_i)^2\right\}}{2\sqrt{2n \max\limits_{i=1,\dots,n} \{(x_i - s_i)^2\}}}.$$

Proof. The proof is essentially the same as the one given for Lemma 6.7. In fact, if we define

$$\alpha_{xs} := \min_{i=1,\dots,n} \left\{(x_i - s_i)^2\right\} > 0,$$

then the previous proof goes through with only minor modifications. ■

Lemmas 6.7 and 6.8 enable us to state the following local convergence result for Algorithm 6.1 if the smoothing parameter τ_k is being updated sufficiently fast in Step 4 of algorithm 6.1.

The next result is somewhat technical but will be required for the proof of the main convergence theorem of this section. It was originally presented by Kanzow and Pieper [35] for a similar Jacobian smoothing method for non-linear complementarity problems, but can be adapted to our situation without problems.

Lemma 6.9 *Let $\{w^k\} \subseteq \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ be a sequence generated by Algorithm 6.1. Assume that $\{w^k\}$ has an accumulation point w^*, which is a solution of the optimality conditions* (3.5)*. Then the index set K defined in* (6.7) *is infinite and $\{\tau_k\} \longrightarrow 0$.*

Proof. Assume that K is finite. Then it follows from (6.4) and the updating rules for β_k in Step 4 for Algorithm 6.1 that there is a $k_0 \in \mathbb{N}$ such that

$$\beta_k = \beta_{k_0}$$

and

$$\|\Phi(w^{k+1})\| > \max\left\{\eta\beta_k, \frac{\|\Phi(w^{k+1}) - \Phi_\tau(w^{k+1})\|}{\alpha}\right\} \geq \eta\beta_k = \eta\beta_{k_0}$$

for all $k \in \mathbb{N}$ with $k \geq k_0$. However, this contradicts the fact that w^* is a solution of the optimality conditions (3.5), so that we have $\Phi(w^*) = 0$. Hence K is an infinite set. The updating rules for τ_k therefore immediately imply that the whole sequence $\{\tau_k\}$ converges to 0. ■

The following result is also of a technical nature, but will be used to show the local superlinear convergence rate. This result goes back to Chen, Qi and Sun [14].

Lemma 6.10 *If there exists a scalar*

$$\omega \in \left[\frac{1}{2} - \frac{(1-\alpha-2\sigma)^2}{2(2+\alpha)^2}, \frac{1}{2}\right] \tag{6.19}$$

such that

$$\Psi(y) \le \Psi(w^k) - 2\omega\Psi(w^k) \tag{6.20}$$

for some $k \in K$ and $y \in \mathbb{R}^{n+m+n}$, then it holds that

$$\Psi_{\tau_k}(y) \le \Psi_{\tau_k}(w^k) - 2\sigma\Psi(w^k), \tag{6.21}$$

where τ_k is the smoothing parameter in the k-th step.

Proof. By the definitions of τ_0 in Step 0 of Algorithm 6.1 and of the index set K (cf. (6.7)) we have

$$0 < \tau_k \le \frac{\alpha}{2\kappa}\|\Phi(w^k)\|, \quad \text{for } k \in K.$$

Hence from Lemma 5.5, for any $y \in \mathbb{R}^{n+m+n}$ and $k \in K$ we have

$$\|\Phi_{\tau_k}(y)\| \le \|\Phi(y)\| + \frac{\alpha}{2}\|\Phi(w^k)\|.$$

and

$$\|\Phi_{\tau_k}(w^k)\| \ge \|\Phi(w^k)\| - \frac{\alpha}{2}\|\Phi(w^k)\|.$$

Using these two inequalities and (6.20), we obtain

$$\begin{aligned}
\Psi_{\tau_k}(y) - \Psi_{\tau_k}(w^k) &= \frac{1}{2}\|\Phi_{\tau_k}(y)\|^2 - \frac{1}{2}\|\Phi_{\tau_k}(w^k)\|^2 \\
&\le \frac{1}{2}\left(\|\Phi(y)\| + \frac{\alpha}{2}\|\Phi(w^k)\|\right)^2 - \frac{1}{2}\left(1-\frac{\alpha}{2}\right)^2 \|\Phi(w^k)\|^2 \\
&= \Psi(y) + \frac{\alpha}{2}\|\Phi(y)\|\|\Phi(w^k)\| \\
&\qquad + \frac{\alpha^2}{4}\Psi(w^k) - \left(1 - \alpha + \frac{\alpha^2}{4}\right)\Psi(w^k) \\
&= \Psi(y) + \frac{\alpha}{2}\|\Phi(y)\|\|\Phi(w^k)\| - (1-\alpha)\Psi(w^k) \\
&\le \Psi(y) + \alpha\sqrt{1-2\omega}\Psi(w^k) - (1-\alpha)\Psi(w^k) \\
&= \Psi(y) - \Psi(w^k) + \alpha(1+\sqrt{1-2\omega})\Psi(w^k) \\
&\le -(2\omega - \alpha(1+\sqrt{1-2\omega}))\Psi(w^k),
\end{aligned}$$

where the second and the last inequalities follow from (6.20).

Let us denote

$$f(\omega) := \omega - \frac{\alpha}{2}(1+\sqrt{1-2\omega}).$$

To prove (6.21), it suffices to show

$$f(\omega) \geq \sigma \quad \text{for} \quad \omega \in \left[\frac{1}{2} - \frac{(1-\alpha-2\sigma)^2}{2(2+\alpha)^2}, \frac{1}{2}\right] \tag{6.22}$$

Recall that $\alpha \in (0,1)$ and $\sigma \in (0, \frac{1}{2}(1-\alpha))$. With $0 < (1-\alpha-2\sigma)/(2+\alpha) < 1$, it follows that

$$\frac{(1-\alpha-2\sigma)^2}{2(2+\alpha)^2} \leq \frac{(1-\alpha-2\sigma)^2}{(2+\alpha)^2} \leq \frac{1-\alpha-2\sigma}{2+\alpha}. \tag{6.23}$$

Notice that f is monotone increasing in $[0, 1/2]$. Thus it is only necessary to show that (6.22) holds at the interval's lower boundary

$$\bar{\omega} = \frac{1}{2} - \frac{(1-\alpha-2\sigma)^2}{2(2+\alpha)^2}.$$

By the definition of f and (6.23) we have

$$\begin{aligned} f(\bar{\omega}) &= \frac{1}{2} - \frac{(1-\alpha-2\sigma)^2}{2(2+\alpha)^2} - \frac{\alpha}{2}\left(1 + \frac{1-\alpha-2\sigma}{2+\alpha}\right) \\ &\geq \frac{1}{2} - \frac{1-\alpha-2\sigma}{2+\alpha} - \frac{\alpha}{2}\left(1 + \frac{1-\alpha-2\sigma}{2+\alpha}\right) \\ &= \frac{1}{2} - \frac{\alpha}{2} \\ &\geq \sigma. \end{aligned}$$

This completes the proof. ∎

Now we are in a position to state the main convergence result of this section.

Theorem 6.11 *Let $w^* := (x^*, \lambda^*, s^*)$ be the unique solution of the optimality conditions* (3.5)*, and assume that (x^*, λ^*, s^*) is an accumulation point of a sequence $\{(x^k, \lambda^k, s^k)\}$ generated by Algorithm 6.1. Then the entire sequence $\{(x^k, \lambda^k, s^k)\}$ converges to (x^*, λ^*, s^*). Moreover, if τ_k is being updated such that*

$$\tau_{k+1} \in \left(0, \min\left\{\frac{\alpha}{2\kappa}\beta_{k+1}, \frac{\tau_k}{2}, \bar{\tau}_k\right\}\right)$$

for all k sufficiently large, where $\bar{\tau}_k := \bar{\tau}(x^{k+1}, s^{k+1}, \beta_{k+1})$ denotes the constant defined in Lemma 6.7 or Lemma 6.8 (depending on whether we choose the Fischer-Burmeister or the minimum function), then $\{(x^k, \lambda^k, s^k)\}$ converges quadratically to (x^*, λ^*, s^*).

Proof. We begin by showing that the entire sequence $\{w^k\}$ converges to the accumulation point w^*.

Consider a subsequence $\{w^k\}_L$ of $\{w^k\}$ converging to w^*. Then it holds for this subsequence that

$$\begin{aligned}\|w^{k+1} - w^k\| &= \|w^k + t_k\Delta w^k - w^k\| \\ &\leq \|\Delta w^k\| \\ &= \| - \Phi'_{\tau_k}(w^k)^{-1}\Phi(w^k)\| \\ &\leq \|\Phi'_{\tau_k}(w^k)^{-1}\| \cdot \|\Phi(w^k)\| \\ &\leq C \cdot \|\Phi(w^k)\|,\end{aligned}$$

where the last but second line follows from equation (6.2) and last line follows from Lemma 3.14, since $\{\|\Phi'(w^k)^{-1}\|\}_L$ is bounded, since by assumption $\{w^k\}_L$ converges to w^* it follows that $\{\|\Phi(w^k)\|\}_L \longrightarrow 0$ and therefore $\{\|w^{k+1} - w^k\|\}_L \longrightarrow 0$. Taking into account that w^* is the unique solution of the optimality conditions (3.5) it follows from Proposition 3.18 that the entire sequence $\{w^k\}$ converges to w^*.

Next, we prove that sequence $\{w^k\}$ converges globally Q-linear and locally Q-quadratic to the solution vector w^*. To this end, first note that, since the solution (x^*, λ^*, s^*) of the optimality conditions (3.5) satisfies strict complementarity in view of our assumptions as well as the Goldman-Tucker Theorem 3.12, it follows that the two conditions $x_i^2 + s_i^2 > 0$ and $x_i \neq s_i$ for all $i = 1, \dots, n$ used in Lemmas 6.7 and 6.8, respectively, are satisfied in a sufficiently small neighborhood of (x^*, λ^*, s^*). Hence we can apply these two results in our situation.

In view of Theorem 6.6, the equation operator Φ is continuously differentiable at $w^* = (x^*, \lambda^*, s^*)$.

By the proof of Lemma 6.9 we have that the index set K is infinite.

By Lemmas 6.7 and 6.8 we have

$$\|\Phi'_{\tau_k}(w^k) - \Phi'(w^k)\|_F \leq \delta \text{ for all } k \in K,$$

where δ is the constant from Lemma 6.7 and 6.8, depending on the NCP-function which is used. By construction of Algorithm 6.1 we can express this as

$$\|\Phi'_{\tau_k}(w^k) - \Phi'(w^k)\|_F \leq \gamma\beta_k \quad \text{for all } k \in K.$$

Proposition 6.4 yields $\{\beta_k\} \longrightarrow 0$ for $k \longrightarrow \infty$. With Proposition 3.14 there exists a constant $\|\Phi'_\tau(w^k)^{-1}\| \leq M$. With the update rule of Algorithm 6.1 we have

$$\begin{aligned}\|w^k + \Delta w^k - w^*\| &= \|w^k - w^* - \Phi'_{\tau_k}(w^k)^{-1}\Phi(w^k)\| \\ &= \|\Phi'_{\tau_k}(w^k)^{-1}[\Phi'_{\tau_k}(w^k)(w^k - w^*) - \Phi(w^k) + \Phi(w^*)]\| \\ &\leq \|\Phi'_{\tau_k}(w^k)^{-1}\| \cdot \Big(\|[\Phi'_{\tau_k}(w^k) - \Phi'(w^k)](w^k - w^*)\| \\ &\qquad + \|\Phi'(w^k)(w^k - w^*) - \Phi(w^k) + \Phi(w^*)\|\Big) \\ &\leq M(\gamma\beta_k\|w^k - w^*\| \\ &\qquad + \|\Phi'(w^k)(w^k - w^*) - \Phi(w^k) + \Phi(w^*)\|\end{aligned} \tag{6.24}$$

Lemma 3.16 then yields

$$\|\Phi'(w^k)(w^k - w^*) - \Phi(w^k) + \Phi(w^*)\| = o(\|w^k - w^*\|) \quad \text{for } k \longrightarrow \infty \text{ and } k \in K$$

and hence

$$\|w^k + \Delta w^k - w^*\| = o(\|w^k - w^*\|) \quad \text{for } k \longrightarrow \infty \text{ and } k \in K. \tag{6.25}$$

Furthermore, since w^* is a solution of the optimality conditions (3.5), $\{w^k\} \longrightarrow w^*$ and $\Phi'(w^*)$ is non-singular, we also get from Lemma 3.17

$$\|\Phi(w^k + \Delta w^k)\| = o(\|\Phi(w^k)\|) \quad \text{for } k \longrightarrow \infty \text{ and } k \in K. \tag{6.26}$$

Let $\omega = \max\left\{\frac{1}{2} - \frac{(1-\alpha-2\sigma)^2}{2(2+\alpha)^2}, \frac{1-\eta^2}{2}\right\}$. Then (6.26) implies that there exists an index $\bar{k} \in K$ such that

$$\Psi(w^k + \Delta w^k) - \Psi(w^k) \le -2\sigma\Psi(w^k). \tag{6.27}$$

for all $k \in K$ with $k \ge \bar{k}$. By Lemma 6.10 we therefore have for any $k \ge \bar{k}$ and $k \in K$,

$$\Psi_{\tau_k}(w^k + \Delta w^k) - \Psi_{\tau_k}(w^k) \le -2\sigma\Psi(w^k),$$

that is, the full step size $t_k = 1$ will eventually be accepted for all $k \ge \bar{k}$ and $k \in K$. In particular, $w^{\bar{k}+1} = w^{\bar{k}} + \Delta w^{\bar{k}}$ and from (6.27) and the definition of ω we obtain

$$\|\Phi(w^{\bar{k}+1})\| \le \sqrt{1-2\omega}\|\Phi(w^{\bar{k}})\| \le \eta\|\Phi(w^{\bar{k}})\| = \eta\beta_{\bar{k}},$$

which implies that $\bar{k}+1 \in K$. Repeating the above process, we may prove that for all $k \ge \bar{k}$ we have $k \in K$ and $w^{k+1} = w^k + \Delta w^k$. Then by using (6.25), we proved that $\{w^k\}$ converges to w^* superlinearly. Since the Jacobian Φ' is locally Lipschitzian, we obtain from Lemma 3.16 (2)

$$\|\Phi'(w^k)(w^k - w^*) - \Phi(w^k) + \Phi(w^*)\| = O(\|w^k - w^*\|^2).$$

Since Φ is locally Lipschitzian, we further obtain

$$\beta_k = \|\Phi(w^k)\| = \|\Phi(w^k) - \Phi(w^*)\| = O(\|w^k - w^*\|).$$

Hence the Q-quadratic convergence of $\{w^k\}$ to w^* follows from (6.24), using similar arguments as for the proof of the local Q-superlinear convergence. ■

There is a recent result by Tseng [55] which indicates that the search direction used in the Jacobian smoothing method has very good local properties. In fact, Tseng [55] shows that it gives a superlinear rate of convergence even if the solution set of the optimality conditions (3.5) is not a singleton. While this result cannot be applied to the Jacobian smoothing framework from Algorithm 6.1, it will be examined in detail in Chapter 8.

6.3 Implementation Details: The Enhanced Jacobian Smoothing Method

6.3.1 Parameters and Initial Point

When implementing the Jacobian smoothing method from algorithm 6.1, it is quite helpful to observe that the linear systems we have to solve at each iteration have exactly the same structure as those in primal-dual interior-point methods. Hence it is possible to use the linear algebra subroutines from existing interior-point codes. In particular the code LIPSOL by Zhang [57, 58] was used. This is a MATLAB program which, however, calls a FORTRAN sparse Cholesky code by Ng and Peyton [47] in order to solve the linear system of equations at each iteration. Note that it is not necessary to solve the linear system (6.2) directly; instead, one can easily use the special structure of this system in order to see that one has to solve only a positive definite system of dimension m at each iteration. LIPSOL also uses the minimum-degree ordering code by Liu [39]. Dense column handling is performed by using the Sherman-Morrison-Woodbury approach or by applying a preconditioned conjugate gradient method. (The book of George and Liu [26] gives further information on sparse Cholesky algorithms. The paper [27] by the same authors gives an overview of minimum degree ordering and its variants.)

Obviously, is has been necessary to completely rewrite the main program in LIPSOL in order to implement the Jacobian smoothing method. The implementation of Algorithm 6.1 uses the following parameters:

$$\alpha = 0.99995, \;\; \eta = 0.31, \;\; \varrho = 0.9, \;\; \sigma = 10^{-4}.$$

The definitions of Φ and Φ_τ are based on the minimum function (5.3) and its smooth counterpart (5.7), respectively.

6.3.2 Termination criteria

The termination criterion used in the code is

$$\|\Phi(w^k)\| \leq \varepsilon \quad \text{with} \quad \varepsilon = 10^{-3}.$$

This is a much weaker condition than what is typically used in corresponding complementarity software, where the Jacobian smoothing idea originated. However, due to possible singularity problems (cf. the discussion in Section 6.2), it seems that one should not use a too strong termination criterion. Moreover, according to experiments, the approximate solutions found by using the above stopping rule seem to have the same accuracy as those provided by interior-point solvers. Nevertheless, it is important to understand that a suitable stopping criterion for the Jacobian smoothing method is a non-trivial task since the iterates are no longer guaranteed to be feasible.

The algorithm also terminates if no solution has been computed after 300 iterations. As an emergency measure two further termination criteria, based on the step length, have been introduced: The program is stopped immediately if the step length is smaller that 10^{-15} or if the step length has been smaller than 10^{-8} for eight successive iterations.

6.3.3 Initial Point

To lessen the impact of possible infeasible iterates, the starting point (x^0, λ^0, s^0) was chosen in such a way that at least the linear equations

$$Ax = b \quad \text{and} \quad A^{\mathrm{T}}\lambda + s = c$$

are satisfied at $(x, \lambda, s) = (x^0, \lambda^0, s^0)$, it follows that these linear equations are satisfied at all iterates $(x, \lambda, s) = (x^k, \lambda^k, s^k)$. Consequently, the only infeasibility which can occur is in the complementarity conditions

$$x_i \geq 0, \;\; s_i \geq 0, \;\; x_i s_i = 0 \;\text{ for } i = 1, \ldots, n.$$

The precise way the initial point is computed is as follows:

(1) Solve $AA^{\mathrm{T}}y = b$ using a sparse Cholesky code in order to compute $y^0 \in \mathbb{R}^m$.

(2) Set $x^0 := A^{\mathrm{T}}y^0$ (hence we have $Ax^0 = b$).

(3) Define $\lambda^0 := 0$ and $s^0 := c$ (so that we also have $A^{\mathrm{T}}\lambda^0 + s^0 = c$).

This may not be the best choice for a starting point to be used within the Jacobian smoothing method, but it seems to be a reasonable and relatively simple choice. The newer and more advanced predictor-corrector methods described in Chapter 8 feature a different initial point.

6.3.4 Non-Monotone Line Search, Significant Reductions and Watchdog Strategy

The Jacobian smoothing method described in Algorithm 6.1 has been changed in some respects to improve its numerical performance. The modifications are described in the following sections have had a great impact on the overall behavior of the algorithm, as some problems could be solved, which had posed great problems so far.

Non-Monotone Line Search

For the implementation of the Jacobian smoothing method from Algorithm 6.1 the line search in Step 3 has been augmented by a non-monotone step length algorithm, which is activated for the first time after $m_{\mathrm{mono}} = 30$ iterations. This non-monotone step length algorithm computes the maximum value of Ψ_{τ_k} of the m_k preceding iterations, but considers no more than the last $m = 10$ iterations.

These numbers are the result of extensive experimentation with the software and the Netlib test set. Even slight modifications have shown great increases in the number of iterations required to solve many problems, even though some of the test problems could be solved very fast, indeed.

Algorithm 6.1 was modified as follows. The initialization phase was augmented in the following way:

Step 0 (Initialization):

Choose $\varrho, \alpha, \eta \in (0,1), \varepsilon > 0, \sigma \in (0, \frac{1}{2}(1-\alpha)$ and $w^0 := (x^0, \lambda^0, s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$. Set $\beta_0 := \|\Phi(w^0)\|, \tau_0 := \frac{\alpha}{2\kappa}\beta_0$ and set the iteration counter $k := 0$. Also set the line search control counters $m_{\text{mono}} := 30$, $m := 10$, $m_{\text{WD}} := 0$, and $m_{\text{SR}} := 0$.

The last two counters m_{WD} and m_{SR} are required to take the watchdog strategy and the significant reduction test into account. These features are described below.

Additionally, Step 3 was changed to incorporate the non-monotone line search:

Step 3 (Non-Monotone Line Search):

Compute $t_k = \max\left\{\varrho^\ell \;\middle|\; \ell = 0,1,2,\dots\right\}$ such that

$$\Psi_{\tau_k}(w^k + t_k \Delta w^k) \le \max_{0 \le j \le m_k} \Psi_{\tau_k}(w^{k-j}) - 2\sigma t_k \Psi(w^k).$$

If $t_k < 10^{-15}$: Stop. Otherwise, set

$$m_{k+1} := \begin{cases} 0, & \text{if } k \le m_{\text{mono}} \text{ or } m_{\text{WD}} \le 6 \text{ or } m_{SR} \le 6 \\ \min\{m_k + 1, m\}, & \text{otherwise.} \end{cases}$$

Set $m_{\text{WD}} \longleftarrow m_{\text{WD}} + 1$, $m_{\text{SR}} \longleftarrow m_{\text{SR}} + 1$, $w^{k+1} := w^k + t_k \Delta w^k$.

Significant reductions

The monotone line search is also used for 6 iterations after a significant reduction of Ψ_τ occured, in the sense that

$$\Psi_\tau(w^k + t_k \Delta w^k) \le 0.2 \cdot \Psi_\tau(w^k).$$

If such a reduction is encountered, the counter m_{SR} is reset to 0 and the next 6 steps are performed using a monotone line search.

All in all, it appears that the Jacobian smoothing method performs better if the number of iterations in which the non-monotone line search algorithm is employed is rather small. It still has to be noted that a little leeway in the computation of the step length by allowing some non-monotone line searches does improve the results over the use of a strictly monotone step length computation.

Watchdog

The enhanced Jacobian smoothing method also contains a so-called watchdog feature. If, after at least 10 iterations, the difference between the smallest residual of the last $m_{\mathrm{WD}} = 10$ iterations and the current residual is too small in the sense that

$$\| \min_{j=1,\dots,m_{\mathrm{WD}}^{\max}} \Phi_\tau(w^{k-j}) - \Phi_\tau(w^k) \| < 10^{-8},$$

the algorithm returns to the iterate which generated the smallest residual. After an activation of the watchdog strategy, the counter m_{WD} is reset to 0 and a monotone line search is performed for the next 6 iterations. After every activation of the watchdog, the maximum number of iterations to take into account in order to find the best previous iterate is increased by 4, i. e., we set $m_{\mathrm{WD}}^{\max} \longleftarrow m_{\mathrm{WD}}^{\max} + 4$.

Enhanced Jacobian Smoothing Method

For the reader's convenience, the enhanced algorithm is stated completely again.

Algorithm 6.12 (Enhanced Jacobian Smoothing Method)
Step 0 (Initialization):

Choose $\varrho, \alpha, \eta \in (0,1), \varepsilon > 0, \sigma \in (0, \frac{1}{2}(1-\alpha))$ and $w^0 := (x^0, \lambda^0, s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$. Set $\beta_0 := \|\Phi(w^0)\|, \tau_0 := \frac{\alpha}{2\kappa}\beta_0$ and set the iteration counter $k := 0$. Also set the line search control counters $m_{\mathrm{mono}} := 30$, $m := 10$, $m_{\mathrm{WD}} := 0$, $m_{\mathrm{WD}}^{\max} = 10$, and $m_{\mathrm{SR}} := 0$.

Step 1 (Termination Criteria):

Stop if any one of the following criteria is satisfied:

- $\|\Phi(w^k)\| \le \varepsilon$,
- $t_j < 10^{-8}$ for all $j = k, \dots, k-7$,
- $k > 300$.

Step 2 (Search Direction):

Compute a solution $\Delta w^k = (\Delta x^k, \Delta\lambda^k, \Delta s^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ of the linear system

$$\Phi'_{\tau_k}(w^k)\Delta w = -\Phi(w^k).$$

Step 3 (Non-Monotone Line Search):

Line Search:

Compute $t_k = \max\left\{ \varrho^\ell \;\middle|\; \ell = 0,1,2,\dots \right\}$ such that

$$\Psi_{\tau_k}(w^k + t_k\Delta w^k) \le \max_{0\le j\le m_k} \Psi_{\tau_k}(w^{k-j}) - 2\sigma t_k \Psi(w^k).$$

If $t_k < 10^{-15}$: Stop.
Otherwise, set

$$m_{k+1} := \begin{cases} 0, & \text{if } k \le m_{\mathrm{mono}} \text{ or } m_{\mathrm{WD}} \le 6 \\ & \text{or } m_{SR} \le 6 \\ \min\{ m_k + 1, m \}, & \text{otherwise.} \end{cases}$$

Descent Test:

If

$$\Psi_\tau(w^k + t_k \Delta w^k) \le 0.2 \cdot \Psi_\tau(w^k)$$

set $m_{SR} = 0$, otherwise increment the counter $m_{\mathrm{SR}} \longleftarrow m_{\mathrm{SR}} + 1$.

Watchdog:

If $k \ge 10$ and

$$\| \min_{j=1,\dots,m_{\mathrm{WD}}^{\max}} \Phi_\tau(w^{k-j}) - \Phi_\tau(w^k)\| < 10^{-8},$$

set $w^{k+1} := \arg\min_{j=1,\dots,m_{\mathrm{WD}}^{\max}} \|\Phi_\tau(w^{k-j})\|$. Set $m_{\mathrm{WD}}^{\max} \longleftarrow m_{\mathrm{WD}}^{\max} + 4$, reset the counter m_{WD} to 0, and go to Step 4.

Update of Iterate:

Set $w^{k+1} := w^k + t_k \Delta w^k$ and increment the counter m_{WD} by 1.

Step 4 (Update of Smoothing Parameter):

If

$$\|\Phi(w^{k+1})\| \le \max\left\{\eta\beta_k, \frac{\|\Phi(w^{k+1}) - \Phi_{\tau_k}(w^{k+1})\|}{\alpha}\right\}, \tag{6.28}$$

then set

$$\beta_{k+1} := \|\Phi(w^{k+1})\|$$

and choose τ_{k+1} such that

$$\tau_{k+1} \in \left(0, \min\left\{\frac{\alpha}{2\kappa}\beta_{k+1}, \frac{\tau_k}{2}\right\}\right).$$

Otherwise (i. e. if (6.28) is not satisfied) set $\beta_{k+1} := \beta_k$ and $\tau_{k+1} := \tau_k$.

Step 5: Set $k \longleftarrow k + 1$ and go to Step 1.

6.4 Numerical Results for the Enhanced Jabocian Smoothing Method for the Netlib Test Set

The algorithm was tested on all problems from the Netlib collection. All test runs were made on a Sun Enterprise 450 using UltraSPARC II CPUs. Table 6.1 presents the results of the test runs. Its columns have the following meaning:

Problem:	Name of the test example in the Netlib collection
m:	Number of rows after preprocessing
n:	Number of columns after preprocessing
k:	Number of iterations until termination
WD:	Number of times the watchdog was activated
$\|\Phi(w^f)\|$:	Value of $\|\Phi(w^f)\|$ at the final iterate w^f
Primal Objective:	Value of the primal objective function at the final iterate w^f

Table 6.1: Numerical results of the enhanced Jacobian smoothing method 6.12

Problem	m	n	k	WD	$\|\Phi(w^f)\|$	Primal Objective
25fv47	798	1854	132	2	8.1618e−04	5.5018499123e+03
80bau3b	2235	11516	143	2	4.7837e−04	9.8722419241e+05
adlittle	55	137	18	0	1.1090e−05	2.2549496316e+05
afiro	27	51	8	0	4.8102e−07	−4.6475314311e+02
agg	488	615	56	0	4.0555e−04	−3.5991767287e+07
agg2	516	758	33	0	7.7409e−04	−2.0239252356e+07
agg3	516	758	31	0	2.1183e−07	1.0312115935e+07
bandm	269	436	60	1	7.6182e−04	−1.5862797925e+02
beaconfd	148	270	31	0	1.8352e−06	3.3592485807e+04
blend	74	114	27	0	4.9101e−05	−3.0812150098e+01
bnl1	632	1576	113	2	5.1622e−04	1.9776295614e+03
bnl2	2268	4430	170	3	7.5310e−04	1.8112372371e+03
boeing1	347	722	81	1	1.1542e−04	−3.3521356138e+02
boeing2	140	279	37	0	4.1229e−05	−3.1501872799e+02
bore3d	199	300	43	0	6.4748e−05	1.3730803943e+03
brandy	149	259	47	0	3.0906e−04	1.5185098971e+03
capri	267	476	43	0	1.2333e−04	2.6900129008e+03
cycle	1801	3305	281	4	2.4260e−04	−5.2249279076e+00
czprob	737	3141	60	0	1.8048e−04	2.1851966989e+06
d2q06c	2171	5831	> 300	*maximum number of iterations reached*		
d6cube	404	6184	8	*stagnating step length*		
degen2	444	757	27	0	6.0299e−05	−1.4351780000e+03
degen3	1503	2604	29	0	2.7391e−04	−9.8729399966e+02
dfl001	6071	12230	> 300	*maximum number of iterations reached*		
e226	220	469	69	0	4.7034e−04	−1.8751928767e+01
etamacro	357	692	79	1	5.5534e−04	−7.5571519859e+02
fffff800	501	1005	71	0	7.6383e−04	5.5567956522e+05
finnis	492	1014	52	0	3.0823e−04	1.7279106560e+05
fit1d	24	1049	227	4	1.7997e−06	−9.1463780924e+03
fit1p	627	1677	9	*stagnating step length*		
fit2d	25	10524	203	3	4.2126e−04	−6.8464293294e+04
fit2p	3000	13525	9	*stagnating step length*		
forplan	135	463	44	0	5.0031e−04	−6.6421884104e+02
ganges	1137	1534	54	0	5.3578e−04	−1.0958573613e+05
greenbea	2318	5424	261	4	2.8941e−04	−7.2462472670e+07
greenbeb	2317	5415	260	4	9.6754e−04	−4.3022602105e+06
grow7	140	301	156	2	2.2648e−06	−4.7787811815e+07
grow15	300	645	232	3	1.7870e−04	−1.0687094129e+08
grow22	440	946	250	3	1.5337e−04	−1.6083433648e+08
israel	174	316	162	3	1.8494e−04	−8.9664482186e+05
kb2	43	68	> 300	*maximum number of iterations reached*		
lotfi	151	364	185	1	9.2147e−05	−2.5264706074e+01
maros	835	1921	> 300	*maximum number of iterations reached*		
maros-r7	3136	9408	53	0	5.1758e−04	1.4971851665e+06
modszk1	686	1622	145	2	9.4655e−04	3.2061972915e+02
nesm	654	2922	109	1	6.5163e−05	1.4076036488e+07
perold	625	1530	187	3	1.4893e−04	−9.3804849737e+03
pilot	1441	4657	> 300	*maximum number of iterations reached*		
pilot4	402	1173	> 300	*maximum number of iterations reached*		
pilot87	2030	6460	> 300	*maximum number of iterations reached*		
pilotja	924	2044	281	4	8.1970e−04	−6.1129835116e+03

Table 6.1: Numerical results of the enhanced Jacobian smoothing method 6.12 (continued)

Problem	m	n	k	WD	$\|\Phi(w^f)\|$	Primal Objective
pilotnov	951	2242	155	3	5.0075e−04	−4.4972761882e+03
pilotwe	722	2930	> 300	*maximum number of iterations reached*		
recipe	85	177	13	0	9.8645e−05	−2.6661599983e+02
sc105	105	163	39	0	5.7189e−04	−5.2202061282e+01
sc205	205	317	82	1	3.4097e−04	−5.2202061208e+01
sc50a	49	77	20	0	8.3305e−05	−6.4575077292e+01
sc50b	48	76	27	0	9.4160e−09	−7.0000000000e+01
scagr25	471	671	99	1	2.5476e−04	−1.4753433061e+07
scagr7	129	185	37	0	9.5727e−07	−2.3313898243e+06
scfxm1	322	592	50	0	4.6251e−04	1.8416759030e+04
scfxm2	644	1184	58	0	4.7686e−04	3.6660261565e+04
scfxm3	966	1776	51	0	3.1900e−04	5.4901254549e+04
scorpion	375	453	87	1	7.8545e−04	1.8781248227e+03
scrs8	485	1270	139	3	2.4730e−04	9.0429695385e+02
scsd1	77	760	10	0	3.4190e−04	8.6666666747e+00
scsd6	147	1350	13	0	4.2885e−04	5.0500000078e+01
scsd8	397	2750	20	0	9.4377e−04	9.0500000003e+02
sctap1	300	660	39	0	3.9728e−04	1.4122500002e+03
sctap2	1090	2500	32	0	2.4227e−04	1.7248071429e+03
sctap3	1480	3340	29	0	7.0305e−04	1.4240000000e+03
seba	515	1036	9	*stagnating step length*		
share1b	112	248	162	2	1.2334e−04	−7.6589318579e+04
share2b	96	162	34	0	2.7101e−05	−4.1573224073e+02
shell	496	1487	48	0	9.7677e−07	1.2088253460e+09
ship04l	356	2162	47	0	6.3126e−07	1.7933245380e+06
ship04s	268	1414	43	0	2.6374e−06	1.7987147004e+06
ship08l	688	4339	113	2	4.4860e−05	1.9090552114e+06
ship08s	416	2171	139	3	2.1824e−04	1.9200982105e+06
ship12l	838	5329	54	0	1.3988e−04	1.4701879193e+06
ship12s	466	2293	108	2	1.6138e−04	1.4892361344e+06
sierra	1222	2715	42	0	1.5335e−04	1.5394359964e+07
stair	356	538	92	1	2.0571e−04	−2.5126695120e+02
standata	359	1258	12	0	1.7527e−04	1.2576994995e+03
standgub	361	1366	23	0	3.5092e−06	1.2576995000e+03
standmps	467	1258	36	0	2.4049e−04	1.4060175000e+03
stocfor1	109	157	51	0	8.2467e−10	−4.1131976219e+04
stocfor2	2157	3045	135	2	4.7732e−04	−3.9024408539e+04
stocfor3	16675	23541	> 300	*maximum number of iterations reached*		
truss	1000	8806	34	0	4.4329e−04	4.5881584719e+05
tuff	292	617	40	0	5.8251e−04	2.9549410844e−01
vtpbase	194	325	38	0	1.1010e−05	1.2983146246e+05
wood1p	244	2595	16	0	6.7710e−05	1.4429024111e+00
woodw	1098	8418	43	1	1.5940e−04	1.3044763327e+00

The results in Table 6.1 are not too bad, even though the software fails to solve 13 of the 96 problems in the test suite. Although interior-point software has a better behavior on most of these problems, the reader should take into account that almost all current implementations of interior-point methods for linear programs are predictor-corrector methods, requiring the solution of two linear systems at each iteration, whereas the enhanced Jacobian smoothing method requires only one system to be solved. Moreover, the best choice for the parameters in Algorithm 6.1 (especially the values for α and η) is not clear, whereas interior-point methods are well-understood in the meantime.

Finally, of those problems for which the algorithm terminated with an error, there is a number of problems which can be solved with different parameter settings. Among these are kb2, pilot4, and seba. For some problems the step length t_k became too small ($< 10^{-15}$) or had been very small ($< 10^{-8}$) for 8 successive iterations. Note again that most of these problems can be solved by using different parameter settings.

It is interesting to note that neither the enhanced Jacobian smoothing method, nor the original LIPSOL code were able to solve the problem dfl001, at least not on the machine on which the computations were performed. One code which *could* solve the problem is PCx (requiring 71 iterations and almost three times the CPU-time necessary to solve all remaining problems in the test set). PCx is another interior-point solver based on Mehrotra's predictor-corrector algorithm. Results for the complete Netlib set of test cases are given in Table 8.9 on page 146.

It is notable that some problems, such as cycle, greenbea, greenbeb, fit1d, and pilotja can be solved after a very large number of iterations, with a total of 4 watchdog activations each. This is a rather large number of activations, since most other problems required at most 2 of the partial restarts. These above mentioned problems are apparently very hard to solve for the enhanced Jacobian smoothing method 6.1, not considering the failures, of course. Another triple of test examples which could be solved only slowly are the three grow*-examples. Most other algorithms presented in the following chapters exhibit great problems in solving these three test cases. Other problems, such as afiro, adlittle, scsd1, scsd8, scsd8, and wood1p seem to be rather simple and could be solve in less than 20 iterations.

Another fact to note is that the failures related to a too small stagnating step length occur typically quite early. None of the problems which failed with this error (d6cube, fit1p, fit2p, and seba) did so after more than 9 iterations. This indicates that the step length had been very small, i. e. less than 10^{-8} from the very beginning on, since the algorithm terminates after 8 iterations with $t_k < 10^{-8}$.

Considering the iteration counts in table 6.1, it appears that Algorithm 6.1 performs quite well on most small problems, whereas larger test examples, i. e., those with $m \geq 600$ still pose a problem.

6.5 Comparison with Interior-Point Methods

The Jacobian smoothing method 6.1 is closely related to interior-point methods (primal-dual path-following methods, to be more precise). To see this, recall that interior-point methods typically perturb the complementarity conditions within the optimality conditions (3.5) in order to deal with a system of the form

$$\begin{aligned} A^{\mathrm{T}}\lambda + s &= c, \\ Ax &= b, \\ x_i > 0, \ s_i > 0, \ x_i s_i &= \tau^2, \quad i = 1, \dots, n \end{aligned} \tag{6.29}$$

(here we use τ^2 instead of τ just for technical reasons). Interior-point methods then apply some kind of Newton's method to the *equations* within these perturbed optimality conditions and deal with the non-negativity of the x- and s-variables separately by a suitable line search. By reducing τ in an appropriate way, interior-point methods have a strong theoretical background and an outstanding numerical performance, see, e.g., the book by Wright [56] for further details.

The relation to the Jacobian smoothing method comes from an observation made by Kanzow [32]: The perturbed optimality conditions (5.21) can be rewritten as

$$\begin{aligned} A^{\mathrm{T}}\lambda + s &= c, \\ Ax &= b, \\ \varphi_\tau(x_i, s_i) &= 0, \quad i = 1, \dots, n, \end{aligned}$$

where φ_τ still denotes either one of the Chen-Harker-Kanzow-Smale smoothing function or the smoothed Fischer-Burmeister function defined in (5.7) and (5.6), respectively. Hence the system (6.29) is completely equivalent to the non-linear system of equations

$$\Phi_\tau(x, \lambda, s) = 0 \tag{6.30}$$

which does not contain any non-negativity constraints (at least not explicitly) and which is the basis of the Algorithm 6.1.

Both the theoretical background and the numerical performance of modern interior-point methods are currently stronger than those of the Jacobian smoothing method presented here. However, interior-point methods were not born in just one day, and Algorithm 6.1 represents the first step in the area of Jacobian smoothing methods, at least as far as linear programs are concerned. In fact, Jacobian smoothing methods have some definite advantages if compared with interior-point methods. In particular, the following points are worth being mentioned:

- The Newton-type search direction computed by using the system (6.30) includes explicitly the information that the x- and s-variables should stay non-negative, whereas the Newton-type direction computed by interior-point methods completely disregards this point. Therefore it stands to reason that from a local point of view, the search direction from (6.2) is actually the better one. In fact, in view of the numerical experience, the Jacobian smoothing method seems to converge locally faster than interior-point methods.

- The system (6.30) is an *unconstrained* reformulation of the perturbed optimality conditions (6.29). Hence it is possible to allow negative components in the iterates. In particular, there is no further restriction on the step length, in contrast to interior-point methods.
- Since, as mentioned above, the iterates are not required to belong to the positive orthant, it is relatively easy to combine Jacobian smoothing methods with an active set strategy. This cannot be done in an easy way by interior-point methods since basically any active set method will project at least some components on the boundary of the positive orthant. If this procedure does not give the solution of the linear program, interior-point methods cannot do much with the information provided by such a strategy, whereas Jacobian smoothing methods can start easily from this projected point (even if the projected point is not a solution of the linear program, it might be much closer to a solution and therefore be an attractive point).
- Finally, if one solves a sequence of similar linear programs (like in branch-and-bound techniques for the solution of integer or mixed integer programs), one typically wants to use the solution of the previous problem as a starting point for the next one. This, however, is usually not possible for interior-point methods because the solution of the previous problem does, in general, not belong to the positive orthant of the next problem, whereas the Jacobian smoothing method can easily deal with this situation since we can start at an arbitrary point. In fact, this might even be an advantage if compared to simplex-type schemes.

Since the results presented above seem quite promising for a first implementation of a relatively new approach, further enhancements have been made. Chief among those is the combination of the Jacobian smoothing idea with the approach of regular smoothing methods.

In the following chapter, two regular smoothing methods are described in detail. No numerical results are presented, as these smoothing algorithms only serve as a vehicle for a combination of the Jacobian smoothing idea with the smoothing methods. In Chapter 8 the two smoothing methods from Chapter 7 are merged with the Jacobian smoothing method presented in this chapter to form advanced predictor-corrector algorithms.

7 Smoothing Methods

This chapter covers so-called *smoothing methods*. These methods are closely related to the Jacobian smoothing method discussed in Chapter 6, since they are also based on a perturbed reformulation of the linear program's KKT conditions (3.5) into a system of equations. Newton's method is then applied to the resulting system.

The main difference between Jacobian smoothing and regular smoothing methods is that a different search direction is computed. The Jacobian smoothing method presented in Algorithm 6.1 calculates the search direction Δw as a solution of the following system:

$$\Phi_\tau'(w)\Delta w = -\Phi(w). \tag{7.1}$$

The smoothing methods presented in this chapter use the following equation to compute their search direction:

$$\Phi_\tau'(w)\Delta w = -\Phi_\tau(w).$$

Note that in this case *both* sides are perturbed, as opposed to only the left-hand side as in (7.1).

This chapter discusses the basics of these regular smoothing methods. In Chapter 8 regular smoothing and Jacobian smoothing methods will be combined into a single predictor-corrector-type algorithm. Different approaches are examined.

7.1 A Globally Convergent Smoothing Method with a Smoothing Parameter

In this section we will once again examine the common KKT conditions (3.5) of the primal and dual linear programs (3.1) and (3.2) respectively.

We employ the NPC-functions defined in example 5.2 and the equation operators Φ and Φ_τ from (5.1) and (5.10), respectively, to formulate these conditions as a (non-smooth) system of equations.

7.1.1 Description of Algorithm

We now state a model smoothing algorithm and examine its global convergence properties.

In this section let φ be one of the NCP-functions from example 5.2 and φ_τ the corresponding smoothing function from example 5.3. Let Φ and Φ_τ be the mappings defined in (5.1) and (5.10) respectively, which of course depend on the choice of the NCP-function φ.

Based on the discussion in Section 5.3, following Corollary 5.9, which is applicable for this method as well, the non-linear system

$$\Phi_\tau(w) = 0$$

has a solution $w_\tau := (x_\tau, \lambda_\tau, s_\tau)$ for all $\tau > 0$ under suitable assumptions.

In case of the minimum- or Fischer-Burmeister function this smoothing path coincides with the central path defined in the field of interior-point methods, see Theorem 5.10, even though a different parameterization is used. This is not the case for the two penalized NCP- and smoothing functions.

Smoothing methods try to follow this path numerically. This requires the definition of a suitable neighborhood of the smoothing path. A number of different neighborhoods is common, see, e. g. [4, 5, 8]. Here we will use the neighborhood

$$\mathcal{N}(\beta) := \{w = (x, \lambda, s) \mid \|\Phi_\tau(w)\| \le \beta\tau \text{ for a } \tau > 0\} \tag{7.2}$$

with $\beta > 0$. The notation $\mathcal{N}(\beta)$ is standard both in interior-point and smoothing literature.

If we set

$$\Psi_\tau(w) := \frac{1}{2}\Phi_\tau(w)^{\mathrm{T}}\Phi_\tau(w) = \frac{1}{2}\|\Phi_\tau(w)\|^2, \tag{7.3}$$

again as in (6.1), we are able to formulate a smoothing algorithm. This algorithm is a simplified form of the one presented in [20]. The algorithm used in this paper will be described in greater detail in Section 8.1.

Algorithm 7.1 (Smoothing Method)
Step 0 (Initialization):

Choose $w^0 = (x^0, \lambda^0, s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$, $\tau_0 > 0$, $\beta \ge \|\Phi_{\tau_0}(w^0)\|/\tau_0$, $\varrho, \sigma \in (0, 1)$, $\varepsilon > 0$ and set the iteration counter $k := 0$.

Step 1 (Termination Criterion):

If $\|\Phi(w^k)\| < \varepsilon$: Stop.

Step 2 (Search Direction):

Compute a solution $\Delta w^k = (\Delta x^k, \Delta\lambda^k, \Delta s^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ of the linear system

$$\Phi'_{\tau_k}(w^k)\Delta w = -\Phi_{\tau_k}(w^k). \tag{7.4}$$

Step 3 (Line Search):

Compute a step length $t_k = \max\left\{ \varrho^\ell \;\middle|\; \ell = 0, 1, 2, \ldots \right\}$ such that

$$\Psi_{\tau_k}(w^k + t_k\Delta w^k) \le \Psi_{\tau_k}(w^k) + \sigma t_k \nabla\Psi_{\tau_k}(w^k)^{\mathrm{T}}\Delta w^k \tag{7.5}$$

and set $w^{k+1} := w^k + t_k\Delta w^k$.

Step 4 (Update of Smoothing Parameter):
Compute $\gamma_k = \max\left\{\varrho^\ell \;\middle|\; \ell = 0,1,2,\ldots\right\}$, such that

$$\|\Phi_{(1-\gamma_k)\tau_k}(w^{k+1})\| \leq \beta(1-\gamma_k)\tau_k \tag{7.6}$$

and set $\tau_{k+1} := (1-\gamma_k)\tau_k$.

Step 5: Set $k \longleftarrow k+1$ and go to Step 1.

Algorithm 7.1 is fairly straightforward. After the initialization phase and a test if the solution has been reached, one step of Newton's method is used to solve $\Phi_{\tau_k}(w) = 0$ with a fixed τ_k in Step 2. Proposition 5.6 guarantees that the linear system (7.4) is always uniquely solvable as long as the matrix A has full rank. Afterwards, a new step size $t_k > 0$ is computed by employing the Armijo's rule for Ψ_{τ_k}, which allows us to define new iterates. Finally, in Step 4 the smoothing parameter τ_k is decreased with respect to the requirement that the next iterate should reside in the neighborhood $\mathcal{N}(\beta)$.

For the formal analysis of Algorithm 7.1's convergence properties, we will assume that

$$\|\Phi_{\tau_k}(w^k)\| > 0 \quad \text{for all } k \in \mathbb{N}$$

holds. This results in the fact that no iterate w^k lies directly on the smoothing path. Numerically, this is highly improbable; theoretically this assumption can be made without loss of generality, since Algorithm 7.1 can be adapted so that in the case of $\|\Phi_{\tau_k}(w^k)\| = 0$ the algorithm skips Steps 2 and 3 and proceeds directly to Step 4. This way only the reduction of the smoothing parameter is performed and no actual step is taken. The following results remain true if this modification is made.

7.1.2 Convergence Analysis

Well-Definedness of Algorithm 7.1

We first show that the Algorithm 7.1 is well-defined.

Proposition 7.2 *Under the assumptions made above, the step size t_k computed in Step 3 of Algorithm 7.1 is uniquely defined.*

Proof. For later reference a short discussion on the way the step size t_k is computed in Step 3 is included here. To this end, recall that the gradient of $\Psi_{\tau_k}(w^k)$ is given by $\nabla\Psi_{\tau_k}(w^k) = \Phi'_{\tau_k}(w^k)^{\mathrm{T}}\Phi_{\tau_k}(w^k)$.

With Proposition 5.6 and the full rank assumption for the matrix A, the linear system (7.4) has a unique solution

$$\Delta w^k = -\Phi'_{\tau_k}(w^k)^{-1}\Phi_{\tau_k}(w^k).$$

With $\nabla\Psi_{\tau_k}(w^k) = \Phi'_{\tau_k}(w^k)^{\mathrm{T}}\Phi_{\tau_k}(w^k)$ and $\|\Phi_{\tau_k}(w^k)\| \geq 0$ for all $k \in \mathbb{N}$ it follows that

$$\begin{aligned}\nabla\Psi_{\tau_k}(w^k)^{\mathrm{T}}\Delta w^k &= -\Phi_{\tau_k}(w^k)^{\mathrm{T}}\Phi'_{\tau_k}(w^k)\Phi'_{\tau_k}(w^k)^{-1}\Phi_{\tau_k}(w^k)\\ &= -\|\Phi_{\tau_k}(w^k)\|^2 \qquad (7.7)\\ &< 0.\end{aligned}$$

Therefore, Armijo's rule in Step 3 of Algorithm 7.1 is well-defined. ∎

The next result shows that all iterates w^k generated by Algorithm 7.1 belong to the neighborhood $\mathcal{N}(\beta)$ defined in (7.2) of the smoothing path.

Proposition 7.3 *The iterates w^k generated by Algorithm 7.1 satisfy $\|\Phi_{\tau_k}(w^k)\| \leq \beta\tau_k$ for all $k \in \mathbb{N}$.*

Proof. Taking into account the definition of the merit function $\Psi_\tau(w) := 1/2 \cdot \|\Phi_\tau(w)\|^2$, equation (7.7) allows us to rewrite (7.5) into the following form:

$$\Psi_{\tau_k}(w^k + t_k\Delta w^k) \leq \Psi_{\tau_k}(w^k) - 2t_k\sigma\Psi_{\tau_k}(w^k) = (1 - 2t_k\sigma)\Psi_{\tau_k}(w^k).$$

With $w^{k+1} = w^k + t_k\Delta w^k$ and $\Psi_{\tau_k}(w^k) > 0$ it follows that

$$\|\Phi_{\tau_k}(w^{k+1})\| = \|\Phi_{\tau_k}(w^k + t_k\Delta w^k)\| < \|\Phi_{\tau_k}(w^k)\| \qquad (7.8)$$

for all $k \in \mathbb{N}$.

We now prove by induction on k that $\|\Phi_{\tau_k}(w^k)\| \leq \beta\tau_k$ holds and that the computation of γ_k in Step 4 of Algorithm 7.1 is a finite process.

For $k = 0$ the choice of β in Step 0 guarantees $\|\Phi_{\tau_0}(w^0)\| \leq \beta\tau_0$. From (7.8) it follows that

$$\|\Phi_{\tau_0}(w^1)\| < \beta\tau_0,$$

hence a $\gamma > 0$ exists for which (7.6) holds, since Φ_τ is continuous in τ.

Let $\|\Phi_{\tau_k}(w^k)\| \leq \beta\tau_k$ be true for a $k \geq 0$ and let $\gamma_k > 0$ with (7.6). Similar to the case $k = 0$ it follows from (7.8) that

$$\|\Phi_{\tau_k}(w^{k+1})\| < \beta\tau_k.$$

Therefore, due to continuity of Φ_τ in τ, there exists a positive scalar γ_{k+1} such that (7.6) holds. Furthermore, the definition of τ_{k+1} in Step 4 immediately implies the inequality

$$\|\Phi_{\tau_{k+1}}(w^{k+1})\| < \beta\tau_{k+1}.$$

This completes the proof. ∎

The combination of Propositions 7.2 and 7.3 shows that Algorithm 7.1 is well-defined.

Before starting the convergence analysis, we close this subsection by noting another simple property of Algorithm 7.1: If one chooses a starting point which satisfies the linear equations $A^T\lambda + s = c$ and $Ax = b$, Newton's method then guarantees that the linear equations are also satisfied for all subsequent iterations. This simple induction argument allows us to state the following lemma.

Lemma 7.4 *Assume that the starting point $w^0 = (x^0, \lambda^0, s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ satisfies the linear equations $A^T\lambda^0 + s^0 = c$ and $Ax^0 = b$. Then all iterates $w^k = (x^k, \lambda^k, s^k)$ generated by Algorithm 7.1 satisfy these equations as well, i. e., we have $A^T\lambda^k + s^k = c$ and $Ax^k = b$ for all $k \in \mathbb{N}$.*

Note that it is quite easy to construct points $w^0 = (x^0, \lambda^0, s^0)$ satisfying the assumptions from Lemma 7.4 since our starting points are not assumed to have positive components x^0 and s^0 (in contrast to the initial points of many interior-point methods, for example).

Global convergence

The following result states that the sequence of smoothing parameters $\{\tau_k\}$ actually does converge to 0. For the remainder of this section, let φ denote any one of the four NCP-function defined in Example 5.2 and φ_τ the respective smooth counterpart from Example 5.3.

To prove global convergence of Algorithm 7.1 we first recall from Proposition 7.3 that any sequence $\{w^k\}$ generated by the algorithm has the property

$$\|\Phi_{\tau_k}(w^k)\| \le \beta\tau_k$$

for all $k \in \mathbb{N}$. Therefore, if τ converges to zero we obtain $\|\Phi_{\tau_k}(w^k)\| \longrightarrow 0$, and this, in turn, shows that any accumulation point of the sequence $\{w^k\}$ is a solution of the optimality conditions (3.5), see Theorem 7.7. The basic question which needs to be answered is therefore: Under which assumptions does the sequence $\{\tau_k\}$ converge to zero?

Many papers on smoothing-type methods (mainly in the related context of complementarity problems) deal with the same question and prove $\{\tau_k\} \downarrow 0$ under restrictive assumptions which typically imply that the solution of the underlying problem is unique, see, e. g., [4, 6, 7, 8, 9, 13] (three exceptions to this are [9, 29, 54]).

Since the uniqueness is of the solution is usually viewed as a too strong assumption in the context of linear programs, we want to state a global convergence result which also holds for solution sets that do not necessarily consist of only a single vector. The main step in this direction is the following proposition:

Proposition 7.5 *Assume that the sequence $\{w^k\}$ generated by Algorithm 7.1 has at least one accumulation point. Then the sequence of smoothing parameters $\{\tau_k\}$ converges to* 0.

Proof. By construction, the sequence $\{\tau_k\}$ of smoothing parameters is monotonically decreasing. Since it is also bounded from below by zero, it follows that $\{\tau_k\}$ converges to some number $\bar{\tau} \geq 0$. If $\bar{\tau} = 0$, we are done.

Assume that $\bar{\tau} > 0$. Then Step 4 of Algorithm 7.1 shows that we necessarily have

$$\gamma_k \downarrow 0. \tag{7.9}$$

Now let w^* be an accumulation point of the sequence $\{w^k\}$ whose existence is guaranteed in view of our assumption. Let $\{w^k\}_K$ be a corresponding subsequence converging to w^*. Since the Jacobian $\Phi'_{\bar{\tau}}(w^*)$ is non-singular by Proposition 5.6, it follows from Lemma 3.14 that there is a constant $\hat{c} > 0$ with

$$\|\Phi'_{\tau_k}(w^k)^{-1}\| \leq \hat{c} \tag{7.10}$$

for all $k \in K$ sufficiently large. We now consider the two cases $\liminf_{k\in K} t_k > 0$ and $\liminf_{k\in K} t_k = 0$ separately and derive a contradiction in each of these cases.

Case 1: $\liminf_{k\in K} t_k > 0$.

Then there exists a positive constant $\bar{t} > 0$ such that $t_k \geq \bar{t}$ for all $k \in K$. In view of (7.1.2) and (7.5), this implies

$$\Psi_{\tau_k}(w^{k+1}) \leq (1 - 2t_k\sigma)\Psi_{\tau_k}(w^k) \leq (1 - 2\bar{t}\sigma)\Psi_{\tau_k}(w^k).$$

Proposition 7.3 therefore gives

$$\begin{aligned} \|\Phi_{\tau_k}(w^{k+1})\| &\leq \sqrt{1 - 2\bar{t}\sigma}\,\|\Phi_{\tau_k}(w^k)\| \\ &\leq (1 - \bar{c})\|\Phi_{\tau_k}(w^k)\| \\ &\leq \beta(1 - \bar{c})\tau_k \end{aligned} \tag{7.11}$$

for some constant $\bar{c} \in (0, 1)$. Since one can easily show that the inequality

$$\|\Phi_\tau(w) - \Phi_{\tau'}(w)\| \leq \kappa|\tau - \tau'|$$

holds for all $\tau, \tau' > 0$ and all w, where $\kappa > 0$ denotes the constant from Lemma 5.5 (cf. [33]), we obtain from (7.11)

$$\begin{aligned} \|\Phi_{(1-\gamma)\tau_k}(w^{k+1})\| &\leq \|\Phi_{\tau_k}(w^{k+1})\| + \|\Phi_{(1-\gamma)\tau_k}(w^{k+1}) - \Phi_{\tau_k}(w^{k+1})\| \\ &\leq \beta(1 - \bar{c})\tau_k + \kappa\gamma\tau_k \\ &= (\beta - \bar{c}\beta + \kappa\gamma)\tau_k \end{aligned}$$

for all $\gamma \in (0, 1)$. Since an elementary calculation shows that the inequality

$$(\beta - \bar{c}\beta + \kappa\gamma)\tau_k \leq \beta(1 - \gamma)\tau_k$$

holds for all $\gamma > 0$ satisfying

$$\gamma \leq \bar{c}\frac{\beta}{\kappa + \beta},$$

the computation of γ_k in Step 4 of Algorithm 7.1 yields

$$\gamma_k \geq \varrho \bar{c} \frac{\beta}{\kappa + \beta} =: \bar{\gamma}$$

for all $k \in K$ sufficiently large, i. e., these γ_k are uniformly bounded away from zero by the positive constant $\bar{\gamma}$. This, however, is a contradiction to (7.9).

Case 2: $\liminf_{k \in K} t_k = 0$.
Subsequencing if necessary, we then have $\lim_{k \in K} t_k = 0$. Consequently, the Armijo condition (7.5) is not satisfied for $\alpha_k := t_k/\varrho$ and all $k \in K$ sufficiently large, i. e., we have

$$\Psi_{\tau_k}(w^k + \alpha_k \Delta w^k) > \Psi_{\tau_k}(w^k) + \alpha_k \sigma \nabla \Psi_{\tau_k}(w^k)^{\mathrm{T}} \Delta w^k$$

or, equivalently,

$$\frac{\Psi_{\tau_k}(w^k + \alpha_k \Delta w^k) - \Psi_{\tau_k}(w^k)}{\alpha_k} > \sigma \nabla \Psi_{\tau_k}(w^k)^{\mathrm{T}} \Delta w^k.$$

Using the Mean Value Theorem, we obtain

$$\nabla \Psi_{\tau_k}(\xi^k)^{\mathrm{T}} \Delta w^k > \sigma \nabla \Psi_{\tau_k}(w^k)^{\mathrm{T}} \Delta w^k \tag{7.12}$$

for some vector ξ^k belonging to the line segment joining w^k and $w^k + \alpha_k \Delta w^k$. From (7.10) and (7.4), $\{\tau_k\}_K \longrightarrow \bar{\tau}$, and $\{w^k\}_K \longrightarrow w^*$, it follows that the sequence $\{\Delta w^k\}_K$ converges to a vector Δw^* which solves the linear system

$$\Phi'_{\bar{\tau}}(w^*) \Delta w = -\Phi_{\bar{\tau}}(w^*). \tag{7.13}$$

Taking the limit $k \longrightarrow \infty$ on K and using $\{\alpha_k\}_K \longrightarrow 0$ as well as $\{\tau_k\}_K \longrightarrow \bar{\tau}$, we then obtain from (7.12) and the continuity of $\nabla \Psi$ with respect to w and τ:

$$\nabla \Psi_{\bar{\tau}}(w^*)^{\mathrm{T}} \Delta w^* \geq \sigma \nabla \Psi_{\bar{\tau}}(w^*)^{\mathrm{T}} \Delta w^*.$$

Consequently, we have

$$\nabla \Psi_{\bar{\tau}}(w^*)^{\mathrm{T}} \Delta w^* \geq 0.$$

On the other hand, it follows from $\nabla \Psi_{\tau_k}(w^k)^{\mathrm{T}} \Delta w^k = -\|\Phi_{\tau_k}(w^k)\|^2 \leq 0$ that

$$\nabla \Psi_{\bar{\tau}}(w^*)^{\mathrm{T}} \Delta w^* \leq 0.$$

We therefore get

$$\nabla \Psi_{\bar{\tau}}(w^*)^{\mathrm{T}} \Delta w^* = 0.$$

Since $\nabla\Psi_{\bar\tau}(w^*) = \Phi'_{\bar\tau}(w^*)^{\mathrm{T}}\Phi_{\bar\tau}(w^*)$, we obtain

$$\Phi_{\bar\tau}(w^*) = 0 \tag{7.14}$$

from (7.13). On the other hand, it follows from (7.9) that γ_k/ϱ does not satisfy the test (7.6) for all $k \in K$ sufficiently large, i.e., we have

$$\|\Phi_{(1-\gamma_k/\varrho)\tau_k}(w^{k+1})\| > \beta\left(1 - \frac{\gamma_k}{\varrho}\right)\tau_k.$$

Since $\gamma_k/\varrho \longrightarrow 0, t_k \longrightarrow 0$ and, therefore, $w^{k+1} \longrightarrow w^*$ for $k \in K$, we obtain

$$\|\Phi_{\bar\tau}(w^*)\| \geq \beta\bar\tau > 0,$$

by taking the limit $k \longrightarrow +\infty$ on K. This, however, contradicts (7.14). ∎

Note that the assumption used in Proposition 7.5, namely the existence of at least one accumulation point, is weaker than the condition that the entire sequence $\{w^k\}$ remains bounded. Nevertheless, in Theorem 7.7 below we will give a sufficient condition for the boundedness of the whole sequence $\{w^k\}$.

Before, however, we state the main global convergence result for Algorithm 7.1, which is a direct consequence of Proposition 7.5.

Theorem 7.6 *Every accumulation point of a sequence* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ *generated by Algorithm 7.1 is a solution of the optimality conditions* (3.5).

Proof. In view of Proposition 7.3, we have

$$\|\Phi_{\tau_k}(w^k)\| \leq \beta\tau_k \tag{7.15}$$

for all $k \in \mathbb{N}$. Now let w^* be an accumulation point of $\{w^k\}$, and let $\{w^k\}_K$ be a subsequence converging to w^*. Then we have $\{\tau_k\} \longrightarrow 0$ in view of Proposition 7.5. Therefore, taking the limit $k \longrightarrow \infty$ for $k \in K$ and using the fact that Φ is a continuous function in both w and τ, we obtain from (7.15) that $\|\Phi(w^*)\| = \|\Phi_0(w^*)\| \leq 0$. Hence $\Phi(w^*) = 0$. The definition of Φ therefore implies that w^* is a solution of the optimality conditions (3.5). ∎

The next result gives a sufficient condition for the existence of accumulation points. Note that this result is stated here only for the penalized minimum and the penalized Fischer-Burmeister function, whereas it is not true, in general, for the minimum or Fischer-Burmeister functions themselves.

Theorem 7.7 *Let* φ *denote either the penalized minimum or the penalized Fischer-Burmeister function. Assume there exists a strictly feasible point for the optimality conditions* (3.5) *and that the starting point* $w^0 = (x^0, \lambda^0, s^0)$ *satisfies the linear equations* $Ax^0 = b$ *and* $A^{\mathrm{T}}\lambda^0 + s^0 = c$. *Then the sequence* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ *generated by Algorithm 7.1 remains bounded.*

Proof. Using Lemma 7.4 and our assumption regarding the starting point, we have $A^{\mathrm{T}}\lambda^k + s^k = c$ and $Ax^k = b$ for all $k \in \mathbb{N}$. Since $\|\Phi(w^k)\| \leq \beta\tau_k$ in view of Proposition 7.3 and $\tau_k \leq \tau_0$ for all $k \in \mathbb{N}$, the statement follows immediately from Corollary 5.9. ∎

7.2 A Globally Convergent Smoothing Method with a Smoothing Variable

In this section another smoothing method is presented. This method follows an idea by Jiang [30]. As Algorithm 7.1, it is based on a smooth equation reformulation of the optimality conditions (3.5) themselves but views the smoothing parameter τ not as a parameter but as an additional variable of the system.

It is, however, closely related to both smoothing-type methods and interior-point methods. This is made clear as soon as we develop the algorithm in Section 7.2.1.

The algorithm can be viewed as an improved smoothing-type method due to some better theoretical properties if compared to Algorithm 7.1. The method presented in the following section is based on the corrector step of an algorithm published by Burke and Xu [8]. In Section 8.2 Algorithm 8.11 will be combined with a suitable predictor step to form a predictor-corrector method similar to the one proposed by Burke and Xu.

We will now consider the smoothing parameter τ not as a parameter, but rather as an additional variable of the non-linear system, hence it will be referred to as a *smoothing variable* to articulate this different point of view.

In the following section the algorithm is developed; a detailed statement is given. Both its global and local convergence properties are investigated in Section 7.2.2.

7.2.1 Description of Algorithm

For the remainder of this section let $\varphi\colon \mathbb{R}^2 \longrightarrow \mathbb{R}$ denote the minimum function

$$\varphi(a,b) := 2\min\{a,b\} = a + b - \sqrt{(a-b)^2}.$$

Let $\varphi_\tau\colon \mathbb{R}^2 \longrightarrow \mathbb{R}$ denote the smoothed minimum- or Chen-Harker-Kanzow-Smale smoothing function

$$\varphi_\tau(a,b) := a + b - \sqrt{(a-b)^2 + 4\tau^2},$$

with a smoothing parameter $\tau > 0$.

Using the equation operator Φ as defined in (5.1) and its smoothed counterpart Φ_τ from (5.10), it was noted in Section 5.3 that it is sufficient to solve

$$\Phi(w) = 0$$

in order to solve the linear program's KKT conditions (3.5).

So far, we have viewed τ as a parameter. In this section, however, it is sometimes useful to treat τ as an independent variable. In order to make this different point of view clear in our notation, let us write

$$\theta(x,s,\tau) := \phi_\tau(x,s). \tag{7.16}$$

The new variable τ will be referred to as the *smoothing variable*. Moreover, we will exploit the mapping $\Theta\colon \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \times [0,\infty) \longrightarrow \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}$ defined by

$$\Theta(w,\tau) := \Theta(x,\lambda,s,\tau) := \begin{bmatrix} A^{\mathrm{T}}\lambda + s - c \\ Ax - b \\ \theta(x,s,\tau) \\ \tau \end{bmatrix}. \tag{7.17}$$

Note that the definition of $\Theta(w,\tau)$ is not equal to $\Phi_\tau(w)$ since one more line has been added. Since the equation $\Theta(w,\tau) = 0$ automatically implies $\tau = 0$, we obtain the equivalence

$$w^* = (x^*,\lambda^*,s^*) \text{ solves (3.5)} \iff (w^*,0) \text{ solves } \Theta(w,\tau) = 0.$$

In this way we therefore get a reformulation of the optimality conditions (3.5) with τ being viewed as an independent variable. This kind of reformulation goes back to Jiang [30].

For later reference, it will be important to exploit the relation between Newton's method applied to the system $\Phi_\tau(w) = 0$ and applied to the system $\Theta(w,\tau) = 0$. First consider the system $\Phi_\tau(w) = 0$, and assume that $w^k = (x^k,\lambda^k,s^k)$ denotes the current iterate and $\tau_k > 0$ the current value of the smoothing parameter. Then we define

$$w^{k+1} = w^k + t_k \Delta w^k$$

for a suitable step size $t_k > 0$, where the correction vector $\Delta w^k = (\Delta x^k, \Delta\lambda^k, \Delta s^k)$ is a solution of the linear system of equations

$$\Phi'_{\tau_k}(w^k)\Delta w^k = -\Phi_{\tau_k}(w^k).$$

Taking into account the definition of Φ_τ, this equation is equivalent to

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & I \\ A & 0 & 0 \\ D^k_{a,\tau} & 0 & D^k_{b,\tau} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -A^{\mathrm{T}}\lambda^k - s^k + c \\ -Ax^k + b \\ -\phi_{\tau_k}(x^k,s^k) \end{bmatrix}, \tag{7.18}$$

where

$$D^k_{a,\tau} := \operatorname{diag}\left(\ldots, \frac{\partial\varphi_{\tau_k}}{\partial a}(x^k_i,s^k_i), \ldots\right) \in \mathbb{R}^{n\times n},$$

and, similarly,

$$D^k_{b,\tau} := \operatorname{diag}\left(\ldots, \frac{\partial\varphi_{\tau_k}}{\partial b}(x^k_i,s^k_i), \ldots\right) \in \mathbb{R}^{n\times n}.$$

On the other hand, if we apply Newton's method to the system $\Theta(w,\tau) = 0$, we have to solve an equation like

$$\Theta'(w^k,\tau_k) \begin{bmatrix} \Delta w \\ \Delta\tau \end{bmatrix} = -\Theta(w^k,\tau_k)$$

at each iteration, where the derivatives are taken with respect to w and τ. Hence the above system becomes

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & I & 0 \\ A & 0 & 0 & 0 \\ D^k_{a,\tau} & 0 & D^k_{b,\tau} & d^k_\tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \\ \Delta\tau \end{bmatrix} = \begin{bmatrix} -A^{\mathrm{T}}\lambda^k - s^k + c \\ -Ax^k + b \\ -\theta(x^k, s^k, \tau_k) \\ -\tau_k \end{bmatrix}, \tag{7.19}$$

where

$$d^k_\tau := \left(\dots, \frac{\partial\theta_i}{\partial\tau}(x^k_i, s^k_i, \tau_k), \dots\right)^{\mathrm{T}} \in \mathbb{R}^n.$$

Motivated by similar considerations in the field of interior-point methods (see, e. g., Wright [56]), we will consider a generalization of the system (7.19) and replace the parameter τ_k on the last row of the right-hand side in (7.19) by $\sigma_k\tau_k$ for some number $\sigma_k \in [0, 1]$, i. e., we solve

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & I & 0 \\ A & 0 & 0 & 0 \\ D^k_{a,\tau} & 0 & D^k_{b,\tau} & d^k_\tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \\ \Delta\tau \end{bmatrix} = \begin{bmatrix} -A^{\mathrm{T}}\lambda^k - s^k + c \\ -Ax^k + b \\ -\theta(x^k, s^k, \tau_k) \\ -\sigma_k\tau_k \end{bmatrix} \tag{7.20}$$

(the choice $\sigma_k = 1$ corresponds to (7.19)). Note, however, that we do not replace τ_k by $\sigma_k\tau_k$ in the definition of the function $\theta(x, s, \tau)$. In order to have a short-hand notation for the linear system (7.20), we introduce the function

$$\Theta_\sigma(w, \tau) := \begin{bmatrix} A^{\mathrm{T}}\lambda + s - c \\ Ax - b \\ \theta(x, s, \tau) \\ \sigma\tau \end{bmatrix}$$

with the subscript σ indicating the dependence of Θ on the parameter σ. Then the linear system (7.20) can be rewritten as

$$\Theta'(w^k, \tau_k) \begin{bmatrix} \Delta w \\ \Delta\tau \end{bmatrix} = -\Theta_\sigma(w^k, \tau_k).$$

Note that this or, equivalently, (7.20) immediately gives

$$\Delta\tau_k = -\sigma_k\tau_k. \tag{7.21}$$

Replacing this expression into the remaining equations of (7.20), we obtain

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & I \\ A & 0 & 0 \\ D^k_{a,\tau} & 0 & D^k_{b,\tau} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -A^{\mathrm{T}}\lambda^k - s^k + c \\ -Ax^k + b \\ -\theta(x^k, s^k, \tau_k) + \sigma_k\tau_k d^k_\tau \end{bmatrix}. \tag{7.22}$$

Obviously, this can be rewritten as

$$\Phi'_{\tau_k}(w^k)\Delta w = -\Phi_{\tau_k}(w^k) + \sigma_k \tau_k \begin{bmatrix} 0 \\ 0 \\ d_\tau^k \end{bmatrix}.$$

This shows that the linear system (7.20) can be viewed as a perturbation of the system (7.18), with the perturbation being active only in the third block row of the right-hand side.

Based on the notation introduced so far, we next give a precise statement of the new smoothing method.

Algorithm 7.8 (Smoothing Method with a Smoothing Variable)
Step 0 (Initialization):
Choose $w^0 := (x^0, \lambda^0, s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ such that $A^{\mathrm{T}}\lambda^0 + s^0 = c$, $Ax^0 = b$, choose $\tau_0 > 0$, select $\beta \geq \|\Phi_{\tau_0}(w^0)\|/\tau_0$, $\varrho \in (0,1)$, $0 < \sigma_{\min} < \sigma_{\max} < 1$, $\varepsilon \geq 0$, and set the iteration counter $k := 0$.

Step 1 (Termination Criterion):
If $\|\Phi(w^k)\| \leq \varepsilon$: Stop.

Step 2 (Search Direction):
Choose $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$.
Compute a solution $(\Delta w^k, \Delta\tau_k) = (\Delta x^k, \Delta\lambda^k, \Delta s^k, \Delta\tau_k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}$ of the linear system

$$\Theta'(w^k, \tau_k) \begin{bmatrix} \Delta w \\ \Delta\tau \end{bmatrix} = -\Theta_{\sigma_k}(w^k, \tau_k). \tag{7.23}$$

Step 3 (Line Search and Update of Smoothing Variable):
Let $t_k = \max\left\{ \varrho^\ell \;\middle|\; \ell = 0, 1, 2, \ldots \right\}$ such that

$$\|\theta(x^k + t_k\Delta x^k, s^k + t_k\Delta s^k, (1 - \sigma_k t_k)\tau_k)\| \leq \beta(1 - \sigma_k t_k)\tau_k. \tag{7.24}$$

Set $w^{k+1} := w^k + t_k \Delta w^k$ and $\tau_{k+1} := (1 - \sigma_k t_k)\tau_k$.

Step 4 (Update):
Set $k \longleftarrow k + 1$, and go to Step 1.

To get a better understanding of the way Algorithm 7.8 works, let us add a couple of comments. In Step 0, we require the starting point $w^0 = (x^0, \lambda^0, s^0)$ to be feasible with respect to the linear equations $A^{\mathrm{T}}\lambda + s = c$ and $Ax = b$. Since the components x^0 and s^0 do not have to be positive (like in interior-point methods), it is relatively easy to find such a starting point. Step 1 contains the termination criterion. Steps 2 and 3 of this algorithm coincide with the corrector step of a smoothing-type predictor-corrector method proposed by Burke and Xu [8]. Note that the linear system (7.23) is precisely the one from (7.20) and includes a perturbation on the right-hand side as well. A procedure to decrease the value of the smoothing parameter is also included in Step 3.

7.2.2 Convergence Analysis

This section investigates the convergence properties of Algorithm 7.8. To this end, we assume throughout this section that the termination parameter ε is equal to zero, and that Algorithm 7.8 generates an infinite number of iterates w^k, i. e., we assume that the algorithm does not stop after a finite number of iterations in a point w^k satisfying the optimality conditions (3.5).

Lemma 7.9 *The following statements hold for any $k \in \mathbb{N}$:*

(1) The linear system (7.23) *has a unique solution.*

(2) The step size t_k in Step 2 is uniquely defined.

(3) $\|\theta(x^k, s^k, \tau_k)\| \leq \beta\tau_k$.

Proof.

(1) The structure of the Jacobian $\Theta'(w^k, \tau_k)$ in (7.19) shows that this matrix is non-singular if and only if $\Phi'_{\tau_k}(w^k)$ is non-singular. This has already been shown in Proposition 5.6.

(2), (3) We prove the last two parts of this lemma at the same time by induction.

First, define the function $\psi(x, s, \tau) := \|\theta(x, s, \tau)\|$. Then

$$\psi'(x, s, \tau) = \frac{\theta(x, s, \tau)^{\mathrm{T}}\theta'(x, s, \tau)}{\|\theta(x, s, \tau)\|}.$$

For the directional derivative it holds together with (7.23) that

$$\begin{aligned}
\psi'((x, s, \tau); (\Delta x, \Delta s, \Delta\tau)) &= \psi'(x, s, \tau)^{\mathrm{T}} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta\tau \end{bmatrix} \\
&= \frac{\theta(x, s, \tau)^{\mathrm{T}}}{\|\theta(x, s, \tau)\|}\theta'(x, s, \tau) \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta\tau \end{bmatrix} \\
&= -\frac{\theta(x, s, \tau)^{\mathrm{T}}}{\|\theta(x, s, \tau)\|}\theta(x, s, \tau) \\
&= -\frac{\|\theta(x, s, \tau)\|^2}{\|\theta(x, s, \tau)\|} \\
&= -\|\theta(x, s, \tau)\|.
\end{aligned}$$

Now we proceed with the actual induction argument.

The inequality $\|\theta(x^0, s^0, \tau_0)\| \leq \beta\tau_0$ holds by the choice of β in Step 0 of Algorithm 7.8.

Assume that (7.24) does not hold, i. e.

$$\psi(x^0 + \varrho^\ell \Delta x^0, s^0 + \varrho^\ell \Delta s^0, (1 - \sigma_0 \varrho^\ell)\tau_0) > \beta(1 - \sigma_0 \varrho^\ell)\tau_0 \quad \text{for all } \ell = 0, 1, \ldots.$$

Using the definition of ψ and $\|\theta(x^0, s^0, \tau_0)\| \le \beta\tau_0$ one gets:

$$\begin{aligned}\psi(x^0 + \varrho^\ell \Delta x^0, s^0 + \varrho^\ell \Delta s^0, (1 - \sigma_0 \varrho^\ell)\tau_0) &> (1 - \sigma_0 \varrho^\ell)\|\theta(w^0, \tau_0)\| \\ &= (1 - \sigma_0 \varrho^\ell)\psi(x^0, s^0, \tau_0)\end{aligned}$$

It then follows that

$$\frac{\psi(x^0 + \varrho^\ell \Delta x^0, s^0 + \varrho^\ell \Delta s^0, (1 - \sigma_0 \varrho^\ell)\tau_0) - \psi(x^0 k, s^0, \tau_0)}{\varrho^\ell} > -\sigma_0 \psi(x^0, s^0, \tau_0)$$

Taking the limit $\ell \longrightarrow \infty$ gives

$$\psi'((x^0, s^0, \tau_0); (\Delta x^0, \Delta s^0, \Delta\tau_0)) \ge -\sigma_0 \|\theta(x^0, s^0, \tau_0)\|,$$

which is obviously equivalent to

$$\|\theta(x^0, s^0, \tau_0)\| \le \sigma_0 \|\theta(x^0, s^0, \tau_0)\|.$$

With $0 < \sigma_{\min} < \sigma_0 < \sigma_{\max} < 1$ from Step 0 of Algorithm 7.8, this contradicts the assumption, therefore there exists a finite step length t_0. This also implies that both

$$w^1 = w^0 + t_0 \Delta w^0 \quad \text{and} \quad \tau_1 = (1 - \sigma_0 t_0)\tau_0$$

exist and that $\|\theta(x^1, s^1, \tau_1)\| \le \beta\tau_1$ by construction.

Assume that (7.24) holds for the k-th iterate, i. e., there exists t_k such that the updating rules in Step 3 yield the existence of $w^{k+1} = w^* + t_k \Delta w^k$ and

$$\|\theta(x^{k+1}, s^{k+1}, \tau_{k+1})\| \le \beta\tau_{k+1}. \tag{7.25}$$

Now consider the $(k + 1)$-st step.

Using the same argumentation as above and keeping in mind (7.25), one gets that there exists a step length t_{k+1} such that

$$\begin{aligned}\|\theta(x^{k+1} + t_{k+1}\Delta x^{k+1}, s^{k+1} + t_{k+1}\Delta s^{k+1}, (1 - \sigma_{k+1} t_{k+1})\tau_{k+1})\| \\ \le \beta(1 - \sigma_{k+1} t_{k+1})\tau_{k+1}.\end{aligned}$$

Therefore the updating rules in Step 3 give

$$w^{k+2} = w^{k+1} + t_{k+1}\Delta w^{k+1} \quad \text{and} \quad \tau_{k+2} = \beta(1 - \sigma_{k+1} t_{k+1})\tau_{k+1}.$$

Since $\sigma_{k+1} \in (0, 1)$ this also implies

$$\|\theta(x^{k+2}, s^{k+2}, \tau_{k+2})\| \le \beta\tau_{k+2}.$$

This completes the proof. ■

Note that this result yields that Algorithm 7.8 is well-defined.

We next state some simple properties of Algorithm 7.8 to which we will refer in our subsequent analysis.

Lemma 7.10 *The two sequences* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ *and* $\{\tau_k\}$ *generated by Algorithm 7.8 have the following properties:*

(1) $A^{\mathrm{T}}\lambda^k + s^k = c$ *and* $Ax^k = b$ *for all* $k \in \mathbb{N}$.

(2) $\tau_k = \tau_{k-1}(1 - \sigma_{k-1}t_{k-1}) \cdot \ldots \cdot \tau_0(1 - \sigma_0 t_0)$ *for all* $k \in \mathbb{N}$.

Proof.

(1) For $k = 0$, this follows from the choice of the starting point in Step 0. Newton's method then guarantees that the linear equations $A^{\mathrm{T}}\lambda + s = c$ and $Ax = b$ are also satisfied for all $k \geq 1$.

(2) The updating rules in Step 3 of Algorithm 7.8 give $\tau_{k+1} = (1 - \sigma_k t_k)\tau_k$. This gives the desired formula.

The next result is quite simple and will be used in order to show that the sequence $\{w^k\}$ generated by Algorithm 7.8 will be bounded under certain conditions.

Lemma 7.11 *The two sequences* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ *and* $\{\tau_k\}$ *generated by Algorithm 7.8 satisfy the inequality*

$$\|\min\{x^k, s^k\}\|_\infty \leq \hat{\kappa}\tau_k$$

for all $k \in \mathbb{N}$ *with* $\hat{\kappa} := (2 + \beta)/2$.

Proof. Let θ_i denote the i-th component function of θ, i. e.,

$$\theta_i(a, b, \tau) := a + b - \sqrt{(a - b)^2 + 4\tau^2}.$$

Then it follows from Lemma 5.4 that the inequality

$$|\theta_i(a, b, 0) - \theta_i(a, b, \tau)| \leq 2\tau$$

holds for all $a, b \in \mathbb{R}$ and all $\tau > 0$. Using Lemma 7.9 (3), it then follows that

$$\begin{aligned} 2\left|\min\left\{x_i^k, s_i^k\right\}\right| &= |\theta_i(x_i^k, s_i^k, 0)| \\ &\leq |\theta_i(x_i^k, s_i^k, \tau_k)| + |\theta_i(x_i^k, s_i^k, 0) - \theta_i(x_i^k, s_i^k, \tau_k)| \\ &\leq \|\theta(x^k, s^k, \tau_k)\| + |\theta_i(x_i^k, s_i^k, 0) - \theta_i(x_i^k, s_i^k, \tau_k)| \\ &\leq (\beta + 2)\tau_k \end{aligned}$$

for all $k \in \mathbb{N}$ and all $i = 1, \ldots, n$. This implies

$$\|\min\{x^k, s^k\}\|_\infty \leq \hat{\kappa}\tau_k$$

for all $k \in \mathbb{N}$, where $\hat{\kappa}$ is the constant specified in the statement of the lemma. ■

Next we show that the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ generated by Algorithm 7.8 remains bounded provided that there is a strictly feasible point for the optimality conditions 3.5 (i. e., a vector $\hat{w} = (\hat{x}, \hat{\lambda}, \hat{s})$ satisfying $A^{\mathrm{T}}\hat{\lambda} + \hat{s} = c, A\hat{x} = b$ and $\hat{x} > 0, \hat{s} > 0$) and that the initial smoothing parameter $\tau_0 > 0$ is sufficiently small. This boundedness result is similar to one given by Chen and Ye [15] in the context of box constrained variational inequality problems.

Proposition 7.12 *Assume that there is a strictly feasible point* $(\hat{x}, \hat{\lambda}, \hat{s})$ *for the optimality conditions* (3.5)*, and suppose that the initial smoothing parameter* $\tau_0 > 0$ *satisfies*

$$\tau_0 < \frac{1}{\hat{\kappa}} \min_{i=1,\dots,n} \{\hat{x}_i, \hat{s}_i\},$$

where $\hat{\kappa} := (2 + \beta)/2$ *denotes the constant from Lemma 7.11. Then the sequence* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ *generated by Algorithm 7.8 is bounded.*

Proof. Consider the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ generated by Algorithm 7.8. Assume that this sequence is unbounded. Since $\{\tau_k\}$ is monotonically decreasing, it follows from Lemma 7.11 that

$$|\min\{x_i^k, s_i^k\}| \le \|\min\{x^k, s^k\}\|_\infty \le \hat{\kappa}\tau_k \le \hat{\kappa}\tau_0 \tag{7.26}$$

for all $k \in \mathbb{N}$ and all $i = 1, \dots, n$. This obviously implies that there is no index $i \in \{1, \dots, n\}$ such that $x_i^k \longrightarrow -\infty$ or $s_i^k \longrightarrow -\infty$ on a subsequence. Therefore, all components of the two sequences $\{x^k\}$ and $\{s^k\}$ are bounded from below.

On the other hand, the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ is unbounded by assumption. This implies that there is at least one component $i \in \{1, \dots, n\}$ such that $x_i^k \longrightarrow +\infty$ or $s_i^k \longrightarrow +\infty$ on a subsequence since otherwise the two sequences $\{x^k\}$ and $\{s^k\}$ would be bounded which, in turn, would imply the boundedness of the sequence $\{\lambda^k\}$ as well, because we have $A^{\mathrm{T}}\lambda^k + s^k = c$ for all $k \in \mathbb{N}$ (cf. Lemma 7.10 (1)) and because A is assumed to have full rank.

Now let $\hat{w} = (\hat{x}, \hat{\lambda}, \hat{s}) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ be the strictly feasible point from our assumption. Then, in particular, we have

$$A^{\mathrm{T}}\hat{\lambda} + \hat{s} = c \quad \text{and} \quad A\hat{x} = b.$$

Since we also have

$$A^{\mathrm{T}}\lambda^k + s^k = c \quad \text{and} \quad Ax^k = b$$

for all $k \in \mathbb{N}$ by Lemma 7.10 (1), we get

$$A^{\mathrm{T}}(\hat{\lambda} - \lambda^k) + (\hat{s} - s^k) = 0 \quad \text{and} \quad A(\hat{x} - x^k) = 0 \tag{7.27}$$

by subtracting these equations. Premultiplying the first equation in (7.27) with $(\hat{x} - x^k)^{\mathrm{T}}$ and taking into account the second equation in (7.27) gives

$$\sum_{i=1}^{n} (\hat{x}_i - x_i^k)(\hat{s}_i - s_i^k) = (\hat{x} - x^k)^{\mathrm{T}}(\hat{s} - s^k) = 0. \tag{7.28}$$

We now assume without loss of generality that there is at least one component i such that $\{x_i^k\}$ is unbounded, i.e., $\{x_i^k\}_K \longrightarrow +\infty$ for a suitable subset $K \subseteq \mathbb{N}$ (the argument would be similar if there would exist at least one component i with $\{s_i^k\}$ being unbounded). Let us define the following index sets:

$$\begin{aligned} I_x &:= \Big\{ i \in \{1, \dots, n\} \;\Big|\; \{x_i^k\}_K \text{ is unbounded} \Big\}, \\ I_s &:= \Big\{ i \in \{1, \dots, n\} \;\Big|\; \{s_i^k\}_K \text{ is unbounded} \Big\}, \\ I_b &:= \Big\{ i \in \{1, \dots, n\} \;\Big|\; \{x_i^k\}_K \text{ and } \{s_i^k\}_K \text{ are bounded} \Big\}. \end{aligned}$$

Note that I_x is non-empty, whereas I_s (and I_b) might be empty. Using the definitions of these three index sets and subsequencing if necessary, we obtain from the inequality (7.26) that

$$\{x_i^k\}_K \longrightarrow +\infty \quad \text{and} \quad s_i^k \le \hat{\kappa}\tau_0 \quad \forall k \in K, \ \forall i \in I_x \tag{7.29}$$

and

$$\{s_i^k\}_K \longrightarrow +\infty \quad \text{and} \quad x_i^k \le \hat{\kappa}\tau_0 \quad \forall k \in K, \ \forall i \in I_s, \tag{7.30}$$

whereas there is a constant $c \in \mathbb{R}$ such that

$$\sum_{i \in I_b} (x_i^k - \hat{x}_i)(s_i^k - \hat{s}_i) \le c$$

for all $k \in K$. Using (7.28) then gives

$$\begin{aligned} c &\ge \sum_{i \in I_b} (x_i^k - \hat{x}_i)(s_i^k - \hat{s}_i) \\ &= \sum_{i \in I_x} (x_i^k - \hat{x}_i)(\hat{s}_i - s_i^k) + \sum_{i \in I_s} (x_i^k - \hat{x}_i)(\hat{s}_i - s_i^k) \end{aligned}$$

for all $k \in K$. However, the right-hand side is unbounded on a subsequence due to (7.29) and (7.30) since $\hat{s}_i - s_i^k \ge \hat{s}_i - \hat{\kappa}\tau_0 > 0$ for $i \in I_x$ and $\hat{x}_i - x_i^k \ge \hat{x}_i - \hat{\kappa}\tau_0 > 0$ for $i \in I_s$ in view of our choice of $\tau_0 > 0$. This contradiction completes the proof. ■

Looking ahead at Algorithm 8.11, it is quite interesting to note that Proposition 7.12 guarantees the boundedness of the iterates w^k provided that the initial smoothing parameter τ_0 is sufficiently small. Burke and Xu [8], on the other hand, can prove the boundedness of their iterates under the assumption that τ_0 is sufficiently large. In fact, Burke and Xu [8] can provide a lower bound for their choice of τ_0 which is known in advance, whereas our upper bound from Proposition 7.12 is, in general, not known. However, the lower bound from [8] could be very large, and this, in turn, could have a bad influence on the numerical behavior of the smoothing-type method. — In any case, it should be noted that some interior-point methods generate bounded iterates under the sole assumption that the primal and dual linear programs (3.1) and (3.2), respectively, are feasible (rather than strictly feasible).

Next we give a global convergence result for Algorithm 7.8.

Theorem 7.13 *Assume that the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ generated by Algorithm 7.8 has at least one accumulation point. Then $\{\tau_k\}$ converges to zero.*

Proof. Since the sequence $\{\tau_k\}$ is monotonically decreasing and bounded from below by zero, it converges to a number $\tau_* \geq 0$. If $\tau_* = 0$, the assertion is proven.

So assume that $\tau_* > 0$. For $k \in \mathbb{N}$ sufficiently large Lemma 7.10 (2) and $\sigma_k \geq \sigma_{\min}$ yield

$$\tau_k = \tau_0 \prod_{j=0}^{k-1} (1 - \sigma_j t_j) \leq \tau_0 \prod_{j=0}^{k-1} (1 - \sigma_{\min} t_j). \tag{7.31}$$

Since $\tau_k \longrightarrow \tau_* > 0$ by assumption, it follows from (7.31) that $\lim_{k \to \infty} t_k = 0$. Therefore, the step size $\alpha_k := t_k/\varrho$ does not satisfy the line search criterion (7.24) for all $k \in \mathbb{N}$ sufficiently large. Hence we have

$$\|\theta(x^k + \alpha_k \Delta x^k, s^k + \alpha_k \Delta s^k, (1 - \sigma_k \alpha_k)\tau_k)\| > \beta(1 - \sigma_k \alpha_k)\tau_k \tag{7.32}$$

for all $k \in \mathbb{N}$.

Now let $w^* = (x^*, \lambda^*, s^*)$ be an accumulation point of the sequence $\{w^k\}$, and let $\{w^k\}_K$ be a subsequence converging to w^*. Since $\sigma_k \in [\sigma_{\min}, \sigma_{\max}]$ for all $k \in \mathbb{N}$, we can assume without loss of generality that the subsequence $\{\sigma_k\}_K$ converges to some number $\sigma_* \in [\sigma_{\min}, \sigma_{\max}]$. Furthermore, since $\tau_* > 0$, it follows from Lemma 7.9 (1) that the corresponding subsequence $\{(\Delta w^k, \Delta \tau_k)\}_K$ converges to a vector $(\Delta w^*, \Delta \tau_*) = (\Delta x^*, \Delta \lambda^*, \Delta s^*, \Delta \tau_*)$, where $(\Delta w^*, \Delta \tau_*)$ is the unique solution of the linear equation

$$\Theta'(w^*, \tau_*) \begin{bmatrix} \Delta w \\ \Delta \tau \end{bmatrix} = -\Theta_{\sigma_*}(w^*, \tau_*), \tag{7.33}$$

cf. (7.23). Using $\{\alpha_k\}_K \longrightarrow 0$ and taking the limit $k \longrightarrow \infty$ on the subset K, we then obtain from (7.32) that

$$\|\theta(x^*, s^*, \tau_*)\| \geq \beta \tau_* > 0. \tag{7.34}$$

On the other hand, we get from (7.32), Lemma 7.10 (3), and $\sigma_k \leq \sigma_{\max}$ that

$$\begin{aligned} \|\theta(x^k + \alpha_k \Delta x^k, s^k + \alpha_k \Delta s^k, (1 - \sigma_k \alpha_k)\tau_k\| &> (1 - \sigma_k \alpha_k)\beta \tau_k \\ &\geq (1 - \sigma_k \alpha_k)\|\theta(x^k, s^k, \tau_k)\| \\ &\geq (1 - \sigma_{\max} \alpha_k)\|\theta(x^k, s^k, \tau_k)\| \end{aligned}$$

for all $k \in \mathbb{N}$ sufficiently large. Using $\Delta\tau_k = -\sigma_k\tau_k$ (cf. (7.21)), this implies

$$\frac{\|\theta(x^k + \alpha_k\Delta x^k, s^k + \alpha_k\Delta s^k, \tau_k + \alpha_k\Delta\tau_k)\| - \|\theta(x^k, s^k, \tau_k)\|}{\alpha_k} \\ \geq -\sigma_{\max}\|\theta(x^k, s^k, \tau_k)\|.$$

Due to (7.34) $\|\theta(\cdot,\cdot,\cdot)\|$ is a continuously differentiable function at (x^*, s^*, τ_*). Taking the limit $k \longrightarrow \infty$ for $k \in K$ then gives

$$\frac{\theta(x^*, s^*, \tau_*)^{\mathrm{T}}}{\|\theta(x^*, s^*, \tau_*)\|}\theta'(x^*, s^*, \tau_*)\begin{bmatrix}\Delta x^* \\ \Delta s^* \\ \Delta\tau_*\end{bmatrix} \geq -\sigma_{\max}\|\theta(x^*, s^*, \tau_*)\|,$$

where $(\Delta x^*, \Delta\lambda^*, \Delta s^*, \Delta\tau_*)$ denotes the solution of the linear system (7.33). Using (7.33) then gives

$$-\|\theta(x^*, s^*, \tau_*)\| \geq -\sigma_{\max}\|\theta(x^*, s^*, \tau_*)\|.$$

Since $\sigma_{\max} \in (0,1)$, this implies $\|\theta(x^*, s^*, \tau_*)\| = 0$, which contradicts (7.34). ■

Note that the assumed existence of an accumulation point in Theorem 7.13 is automatically satisfied under the conditions of Proposition 7.12. — An immediate consequence of Theorem 7.13 is the following result.

Corollary 7.14 *Every accumulation point of a sequence* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ *generated by Algorithm 7.8 is a solution of the optimality conditions* (3.5).

Proof. Let $w^* = (x^*, \lambda^*, s^*)$ be an accumulation point of the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$, and let $\{w^k\}_K$ denote a subsequence converging to w^*. Then we have $\tau_k \longrightarrow 0$ in view of Theorem 7.13. Hence Lemma 7.9 (3) implies

$$\|\theta(x^*, s^*, 0)\| = \lim_{k\in K}\|\theta(x^k, s^k, \tau_k)\| \leq \beta\lim_{k\in K}\tau_k = 0,$$

i. e., we have $x^* \geq 0$, $s^* \geq 0$ and $x_i^* s_i^* = 0$ for $i = 1, \dots, n$ due to the definition of θ. Since Lemma 7.10 (1) also shows that we have $A^{\mathrm{T}}\lambda^* + s^* = c$ and $Ax^* = b$, we see that $w^* = (x^*, \lambda^*, s^*)$ is indeed a solution of the optimality conditions (3.5). ■

This completes the global convergence analysis of Algorithm 7.8. The above results will be used extensively in Sections 8.2 and 8.3, since the predictor-corrector methods 8.11 and 8.18 are based on Algorithm 7.8 discussed in this section.

8 Predictor-Corrector Methods

This chapter will provide details on how the Jacobian smoothing idea presented in Chapter 6 can be merged with regular smoothing methods as described in Chapter 7. The resulting three predictor-corrector algorithms have shown quite interesting numerical performance. Detailed results for the complete Netlib test set are presented for each one of the algorithms.

8.1 Global and Quadratic Convergence of a Smoothing Predictor-Corrector Method

The algorithm which is presented and analyzed in the following section is directly related to Algorithm 7.1. For the predictor-corrector method presented here, Steps 2 through 4 are cast into the new corrector step, while the predictor step follows the Jacobian smoothing idea introduced in Chapter 6.

Basically, the smoothing-type steps are used to monitor global convergence, while the Jacobian smoothing steps guarantee fast local convergence. Notable about this approach is that the local convergence theory supplied here (which is partially based on a paper by Tseng [55]) does not imply that the linear program's solution set is a singleton.

8.1.1 Description of Algorithm

Let φ be any of the NCP-functions introduced in example 5.2. The basic idea of our method is to solve the optimality conditions (3.5) by solving the equivalent nonlinear system of equations $\Phi(w) = 0$, where Φ is defined via (5.1). However, since Φ is non-smooth and may also have singular Jacobian matrices, we replace the system $\Phi(w) = 0$ by the smooth equation $\Phi_\tau(w) = 0$ to which we apply Newton's method. Note that we consider the perturbation τ as a parameter again, not as an independent variable. Of course, we have to take care of the way we update the smoothing parameter τ. Using the notation introduced in (6.1)

$$\Psi_\tau(w) := \frac{1}{2}\Phi_\tau(w)^T\Phi_\tau(w) = \frac{1}{2}\|\Phi_\tau(w)\|^2,$$

we first give a formal statement of the smoothing-type method. The method can be viewed as a simplification of a recently proposed continuation method by Chen and Chen [9] (in [9], the authors use some additional perturbations of the mapping Φ in order to improve their global convergence properties).

Algorithm 8.1 (Predictor-Corrector Smoothing Method)

Step 0 (Initialization):

Choose $w^0 := (x^0, \lambda^0, s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$, $\tau_0 > 0, \beta \geq \|\Phi_{\tau_0}(w^0)\|/\tau_0$, $\varrho, \sigma \in (0,1), \varepsilon \geq 0$, and set the iteration counter $k := 0$.

Step 1 (Termination Criterion):

If $\|\Phi(w^k)\| \leq \varepsilon$: Stop.

Step 2 (Jacobian Smoothing Predictor Step):

Compute a solution $\Delta w^k = (\Delta x^k, \Delta\lambda^k, \Delta s^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ of the linear system

$$\Phi'_{\tau_k}(w^k)\Delta w = -\Phi(w^k). \tag{8.1}$$

If $\|\Phi(w^k + \Delta w^k)\| \leq \varepsilon$: Stop. Otherwise, if

$$\|\Phi_{\tau_k}(w^k + \Delta w^k)\| > \beta\tau_k,$$

then set

$$\hat{w}^k := w^k, \;\; \hat{\tau}_k := \tau_k,$$

else compute $t_k = \varrho^{\ell_k}$, where ℓ_k is the non-negative integer such that

$$\begin{aligned} \|\Phi_{\varrho^j\tau_k}(w^k + \Delta w^k)\| &\leq \beta\varrho^j\tau_k \quad \forall j = 0,1,2,\dots,\ell_k \text{ and} \\ \|\Phi_{\varrho^{\ell_k+1}\tau_k}(w^k + \Delta w^k)\| &> \beta\varrho^{\ell_k+1}\tau_k. \end{aligned}$$

Set $\hat{\tau}_k := t_k\tau_k$ and

$$\hat{w}^k := \begin{cases} w^k & \text{if } \ell_k = 0, \\ w^k + \Delta w^k & \text{otherwise.} \end{cases}$$

Step 3 (Smoothing Corrector Step):

Compute a solution $\Delta\hat{w}^k = (\Delta\hat{x}^k, \Delta\hat{\lambda}^k, \Delta\hat{s}^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ of the linear system

$$\Phi'_{\hat{\tau}_k}(\hat{w}^k)\Delta\hat{w} = -\Phi_{\hat{\tau}_k}(\hat{w}^k). \tag{8.2}$$

Let $\hat{t}_k = \max\left\{\varrho^\ell \;\middle|\; \ell = 0,1,2,\dots\right\}$ such that

$$\Psi_{\hat{\tau}_k}(\hat{w}^k + \hat{t}_k\Delta\hat{w}^k) \leq \Psi_{\hat{\tau}_k}(\hat{w}^k) + \hat{t}_k\sigma\nabla\Psi_{\hat{\tau}_k}(\hat{w}^k)^{\mathrm{T}}\Delta\hat{w}^k, \tag{8.3}$$

and set

$$w^{k+1} := \hat{w}^k + \hat{t}_k\Delta\hat{w}^k.$$

Compute $\gamma_k = \max\left\{\varrho^\ell \;\middle|\; \ell = 0,1,2,\dots\right\}$ such that

$$\|\Phi_{(1-\gamma_k)\hat{\tau}_k}(w^{k+1})\| \leq \beta(1-\gamma_k)\hat{\tau}_k, \tag{8.4}$$

and set $\tau_{k+1} := (1-\gamma_k)\hat{\tau}_k$.

Step 4 (Update):

Set $k \longleftarrow k + 1$, and go to Step 1.

Algorithm 8.1 solves two linear systems of equations at each iteration, one in the predictor step and one in the corrector step. However, the coefficient matrices of these two systems are identical in case the predictor step is not successful; in this case only one matrix factorization per iteration needs to be calculated.

The global and local convergence properties of Algorithm 8.1 will be discussed in Section 8.1.2. Nevertheless, it needs to be emphasized at this point that the global properties mainly follow from the corrector step, whereas the local theory is entirely based on the predictor step. To this end, it is also interesting to note a major difference between the two linear systems (8.1) and (8.2): While the linear system (8.2) arising in the corrector step is just Newton's method applied to the non-linear equation $\Phi_{\hat{\tau}_k}(w) = 0$, the right-hand side of the linear system (8.1) in the predictor step is unperturbed, i. e., it does not depend on the smoothing parameter. Hence the linear system (8.1) cannot be interpreted as a pure Newton equation. The resulting direction is called a Jacobian smoothing step, see Chapter 6 and, e. g., [14, 35, 23]. Such a Jacobian smoothing step may be viewed as the counterpart of the affine scaling step from the interior-point literature (see, e. g., [56]) for smoothing-type methods.

Next we give some further explanations about the way Algorithm 8.1 works. The basic philosophy behind Algorithm 8.1 is quite simple: It tries to generate a sequence $\{w^k\}$ of iterates and a sequence $\{\tau_k\}$ of smoothing parameters satisfying the relation

$$\|\Phi_{\tau_k}(w^k)\| \leq \beta\tau_k \tag{8.5}$$

for all $k \in \mathbb{N}$. In this way, it is guaranteed that each w^k belongs to a certain neighborhood of the smoothing path. Moreover, we control the size of the neighborhood by the smoothing parameter τ_k. A neighborhood of the form (8.5) for the smoothing path was first used by Chen and Xiu [13] (for the minimum function), whereas other authors sometimes prefer to use slightly different neighborhoods for the smoothing path, see, e. g., [4, 5, 8]. We have already used this neighborhood for the smoothing method 7.1 from Section 7.1, (cf. (7.2)).

Having this philosophy in mind, the predictor step in Algorithm 8.1 does the following: After solving the Jacobian smoothing equation (8.1), it checks whether the full step $w^k + \Delta w^k$ still belongs to the neighborhood (8.5) in the sense that

$$\|\Phi_{\tau_k}(w^k + \Delta w^k)\| \leq \beta\tau_k. \tag{8.6}$$

If this is true, we set $\hat{w}^k := w^k + \Delta w^k$ and reduce the smoothing parameter τ_k as much as possible with the restriction that a condition like (8.5) holds at the intermediate iterate $\hat{w}^k$ and for the new smoothing parameter $\hat{\tau}_k$. In Proposition 8.2 below we will see that the calculation of $\hat{\tau}_k$ is a finite procedure. Otherwise, if (8.6) does not hold, we basically skip the predictor step and set $\hat{w}^k := w^k$ and $\hat{\tau}_k := \tau_k$. However, there is one exception which will become important in our global convergence analysis: Even if (8.6) is satisfied, we do not set $\hat{w}^k = w^k + \Delta w^k$ if $\ell_k = 0$

(so that $\hat{\tau}_k = \tau_k$). By (not) performing the update in this way, we make sure that always both the intermediate iterate $\hat{w}^k$ and the smoothing parameter $\hat{\tau}_k$ are modified, i. e., the predictor step is only accepted if the smoothing parameter could be decreased sufficiently.

The corrector step then computes a new Newton direction $\Delta\hat{w}^k$ in (8.2), followed by a line search in order to reduce the value of $\Psi_\tau(\cdot)$, so that $w^{k+1} = \hat{w}^k + \hat{t}_k\Delta\hat{w}^k$ becomes the new iterate, where of course $\hat{t}_k > 0$ denotes the step length in the corrector step. In addition, we try to reduce the smoothing parameter in the corrector step in such a way that a condition like (8.5) still holds at iteration $k+1$.

In the remaining part of this section, we state some elementary properties of Algorithm 8.1 and show, in particular, that the method is well-defined. To this end, we first recall that the two linear systems (8.1) and (8.2) have a unique solution due to Proposition 5.6. We next show that the computation of $t_k > 0$ in the predictor step is a finite procedure.

Proposition 8.2 *The step size $t_k > 0$ in Step 2 of Algorithm 8.1 is uniquely defined.*

The proof is quite standard, see, e.g., [8], and is included here only for the sake of completeness.

Proof. When computing the step size $t_k > 0$ in Step 2 of Algorithm 8.1, one is in the situation that $\|\Phi(w^k + \Delta w^k)\| > 0$ and $\|\Phi_{\tau_k}(w^k + \Delta w^k)\| \le \beta\tau_k$. Therefore, since Φ_τ is continuous in both w and τ, there is a unique exponent ℓ_k such that $t_k = \varrho^{\ell_k}$ has the properties given in Step 2. ∎

The computation of the step size $\hat{t}_k$ in Step 3 of Algorithm 8.1 corresponds to a standard globalization of Newton's method for non-linear systems of equations by Armijo's rule and is known to be well-defined. Hence we can state the following result. For a proof the reader is referred to Proposition 7.2, as the arguments are identical.

Proposition 8.3 *The step size $\hat{t}_k > 0$ in Step 3 of Algorithm 8.1 is uniquely defined.*

8.1.2 Convergence Analysis

In this section the convergence properties of Algorithm 8.1 are investigated.

The algorithm's global convergence properties only depend on the corrector step. This step, however, has already been analyzed as part of Algorithm 7.8. Therefore we can apply the global convergence theory from Section 7.2.

Now we concentrate on the local behavior. To prove fast local convergence, the following assumptions are made:

Assumption 8.4 *Assume the following:*

(1) φ denotes the minimum function.

(2) The starting point $w^0 = (x^0, \lambda^0, s^0)$ satisfies $A^T\lambda^0 + s^0 = c$ and $Ax^0 = b$.

(3) *The parameter β satisfies $\beta \geq \|\Phi_{\tau_0}(w^0)\|/\tau_0$ and $\beta > \kappa$, where $\kappa > 0$ denotes the (known) constant from Lemma 5.5.*

(4) *The sequence $\{w^k\}$ converges to a vector $w^* = (x^*, \lambda^*, s^*)$ satisfying the strict complementarity condition $x_i^* + s_i^* > 0$ for all $i = 1, \dots, n$.*

Note that this also implies $\lim_{k \longrightarrow \infty} \tau_k = 0$.

Assumption 8.4 (1) is a restriction compared to the global convergence theory from the previous section. However, the subsequent analysis could not be extended to any of the NCP-functions stated in Example 5.2 other than the minimum function. Assumption 8.4 (2) requires the starting point $w^0 = (x^0, \lambda^0, s^0)$ to satisfy the linear equations $A^{\mathrm{T}}\lambda^0 + s^0 = c$ and $Ax^0 = b$. Recall that this is a rather weak assumption since we do not assume, in addition, that x^0 and s^0 have positive components. Condition 8.4 (3) states that the parameter β is sufficiently large. The first condition in Assumption 8.4 (3) is identical to the initialization of the globally convergent Algorithm 8.1; the second condition is only used in order to establish local fast convergence, as will become clear from the analysis in this section. Finally, it is currently not clear whether Assumption 8.4 (4) holds automatically and is therefore superfluous. Note, however, that it is certainly satisfied if the optimality conditions (3.5) have a unique solution since then this solution satisfies the strict complementarity condition by the Goldman-Tucker Theorem 3.12.

Also note, however, that our subsequent analysis would remain true if Assumption 8.4 (4) would be replaced by the requirement that

$$\liminf_{k \longrightarrow \infty} (x_i^k + s_i^k) > 0,$$

so that every accumulation point satisfies the strict complementarity assumption, see also Tseng [55]. Since, in any case, part of the analysis is somewhat technical, it is preferable to use the slightly stronger condition 8.4 (4) throughout this section.

Before we state the next lemma, recall the definition of the solution set of a linear program's optimality conditions (3.5) as introduced in (3.6):

$$S := \left\{ (x, \lambda, s) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \;\middle|\; (x, \lambda, s) \text{ solves the optimality conditions (3.5)} \right\}.$$

Furthermore, recall the definitions of S_P and S_D, the solution sets of the primal and dual linear programs, (3.1) and (3.2), respectively:

$$\begin{aligned} S_P &:= \{x \mid x \text{ solves the primal problem (3.1)}\}, \\ S_D &:= \{(\lambda, s) \mid (\lambda, s) \text{ solves the dual problem (3.2)}\}. \end{aligned}$$

The following result states that the function $\|\Phi(\cdot)\|$ provides a local error bound for the distance of a vector w to the solution set S (cf. Definition 3.11) of the optimality conditions (3.5).

Lemma 8.5 *There exists a constant $\gamma > 0$ with*

$$\operatorname{dist}_S(w^k) \leq \gamma \|\Phi(w^k)\|$$

for all $k \in \mathbb{N}$ sufficiently large.

Proof. We first note that Φ^{-1} is a polyhedral multifunction (cf. Definition 3.19). Therefore, it follows from Proposition 3.21 that Φ^{-1} is locally upper Lipschitzian at the origin, i.e., there exist scalars $\varepsilon > 0$ and $\gamma > 0$ such that

$$\Phi^{-1}(z) \subseteq \Phi^{-1}(0) + \gamma \|z\| \mathcal{B}_1(0)$$

for all z with $\|z\| < \varepsilon$, where $\mathcal{B}_1(0)$ denotes the unit ball in $\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$. Using $z = \Phi(w^k)$ and taking into account that $w^k \longrightarrow w^*$ for some vector $w^* \in S$ by Assumption 8.4 (4), it follows that

$$w^k \in \Phi^{-1}(0) + \gamma \|\Phi(w^k)\| \mathcal{B}_1(0)$$

for all $k \in \mathbb{N}$ sufficiently large. Since $\Phi^{-1}(0)$ is equal to the solution set S of the optimality conditions (3.5), this implies

$$\operatorname{dist}_S(w^k) \leq \gamma \|\Phi(w^k)\|$$

for all $k \in \mathbb{N}$ large enough. ■

Next we show that the distance of the current iterate w^k to the solution set S is bounded from above by a constant multiplied with the current smoothing parameter τ_k.

Lemma 8.6 *We have*

$$\operatorname{dist}_S(w^k) = O(\tau_k)$$

for all $k \in \mathbb{N}$ large enough.

Proof. Since the primal-dual solution set S is closed and non-empty, there exists, for each $k \in \mathbb{N}$, a vector $w^{*,k} \in S$ such that

$$\|w^k - w^{*,k}\| = \operatorname{dist}_S(w^k).$$

In view of Lemma 8.5, there is a constant $\gamma > 0$ with

$$\|w^k - w^{*,k}\| \leq \gamma \|\Phi(w^k)\|$$

for all $k \in \mathbb{N}$ sufficiently large. Using Proposition 7.3 and Lemma 5.5, this gives

$$\begin{aligned}
\|w^k - w^{*,k}\| &\leq \gamma \|\Phi(w^k)\| \\
&\leq \gamma \left(\|\Phi_{\tau_k}(w^k)\| + \|\Phi(w^k) - \Phi_{\tau_k}(w^k)\| \right) \\
&\leq \gamma \left(\beta \tau_k + \kappa \tau_k \right) \\
&= \gamma (\beta + \kappa) \tau_k,
\end{aligned}$$

where $\kappa > 0$ denotes the constant from Lemma 5.5. This implies our statement. ■

Next we estimate the growth behavior of the vector Δw^k, where Δw^k denotes the solution of the linear system (8.1) in the predictor step. Typically, one obtains simple estimates if the inverse of the Jacobian matrices $\Phi_{\tau_k}(w^k)^{-1}$ would remain bounded, and, indeed, this is the standard way in which most papers on smoothing-type methods prove local fast convergence results for their methods. However, according to Theorem 6.6 (see also [8, 23]), the above Jacobian matrices converge to a singular matrix unless the optimality conditions (3.5) have a unique solution. Since we want to avoid the uniqueness assumption here, we cannot use standard perturbation results. Instead, techniques used in the recent paper by Tseng [55] are applied and show by a somehow technical analysis that the term $\|\Delta w^k\|$ can be estimated by a constant multiplied by the current smoothing parameter. This result will play a crucial role in order to establish local quadratic convergence of our method at the end of this section.

Lemma 8.7 *We have*

$$\|\Delta w^k\| = O(\tau_k)$$

for all $k \in \mathbb{N}$ sufficiently large.

Proof. Taking into account the structure of the Jacobian $\Phi'_{\tau_k}(w^k)$ (cf. (5.12)) and the definition of Φ, we get from the linear system (8.1) in the predictor step that

$$A^{\mathrm{T}}\Delta\lambda^k + \Delta s^k = 0, \tag{8.7a}$$

$$A\Delta x^k = 0, \tag{8.7b}$$

$$D^k_{a,\tau}\Delta x^k + D^k_{b,\tau}\Delta s^k = -\phi(x^k, s^k), \tag{8.7c}$$

where, similar to (5.13),

$$\begin{aligned} D^k_{a,\tau} &:= \operatorname{diag}\left(\dots, \frac{\partial\varphi_{\tau_k}}{\partial a}(x^k_i, s^k_i), \dots\right) \in \mathbb{R}^{n\times n}, \\ D^k_{b,\tau} &:= \operatorname{diag}\left(\dots, \frac{\partial\varphi_{\tau_k}}{\partial b}(x^k_i, s^k_i), \dots\right) \in \mathbb{R}^{n\times n}, \\ \phi(x^k, s^k) &:= \left(\dots, \varphi(x^k_i, s^k_i), \dots\right)^{\mathrm{T}} \in \mathbb{R}^n, \end{aligned}$$

and the right-hand sides of (8.7a) and (8.7b) are zero because of Assumption 8.4 (2) and Lemma 7.4. For each $k \in \mathbb{N}$, let $w^{*,k}$ be a solution of the optimality conditions (3.5) such that

$$\operatorname{dist}_S(w^k) = \|w^k - w^{*,k}\|. \tag{8.8}$$

Since $w^{*,k} = (x^{*,k}, \lambda^{*,k}, s^{*,k})$ satisfies (3.5), we have

$$A^{\mathrm{T}}\lambda^{*,k} + s^{*,k} - c = 0, \tag{8.9a}$$

$$Ax^{*,k} - b = 0, \tag{8.9b}$$

$$\phi(x^{*,k}, s^{*,k}) = 0. \tag{8.9c}$$

Furthermore, Lemma 7.4 implies that $w^k = (x^k, \lambda^k, s^k)$ satisfies the linear equations

$$A^{\mathrm{T}}\lambda^k + s^k - c = 0, \tag{8.10a}$$

$$Ax^k - b = 0. \tag{8.10b}$$

Adding (8.7a), (8.9a), and (8.10a) on the one hand, and adding (8.7b), (8.9b), and (8.10b) on the other hand, we obtain

$$A^{\mathrm{T}}(\lambda^k + \Delta\lambda^k - \lambda^{*,k}) + (s^k + \Delta s^k - s^{*,k}) = 0, \tag{8.11a}$$

$$A(x^k + \Delta x^k - x^{*,k}) = 0. \tag{8.11b}$$

Premultiplying (8.11a) with $(x^k + \Delta x^k - x^{*,k})^{\mathrm{T}}$ and taking into account (8.11b) gives

$$(x^k + \Delta x^k - x^{*,k})^{\mathrm{T}}(s^k + \Delta s^k - s^{*,k}) = 0. \tag{8.12}$$

Using (8.7c), we further have

$$D^k_{a,\tau}(x^k + \Delta x^k - x^{*,k}) + D^k_{b,\tau}(s^k + \Delta s^k - s^{*,k}) = r^k \tag{8.13}$$

with

$$r^k := D^k_{a,\tau}(x^k - x^{*,k}) + D^k_{b,\tau}(s^k - s^{*,k}) - \phi(x^k, s^k). \tag{8.14}$$

Having derived these preliminary formulas, we now show in three steps that

$$\begin{aligned}\|\Delta x^k\| &= O(\tau_k),\\ \|\Delta s^k\| &= O(\tau_k), \text{ and}\\ \|\Delta\lambda^k\| &= O(\tau_k).\end{aligned}$$

Step 1: In this step, we show that $\|\Delta x^k\| = O(\tau_k)$. Premultiplying (8.13) with $(x^k + \Delta x^k - x^{*,k})^{\mathrm{T}}(D^k_{b,\tau})^{-1}$, using (8.12), and taking into account that the matrix product $D^k_{a,\tau}D^k_{b,\tau}$ is positive definite, we obtain

$$\begin{aligned}&\lambda_{\min}(D^k_{a,\tau}D^k_{b,\tau})\|(D^k_{b,\tau})^{-1}(x^k + \Delta x^k - x^{*,k})\|^2\\ &\le (x^k + \Delta x^k - x^{*,k})^{\mathrm{T}}(D^k_{b,\tau})^{-1}D^k_{b,\tau}D^k_{a,\tau}(D^k_{b,\tau})^{-1}(x^k + \Delta x^k - x^{*,k})\\ &= (x^k + \Delta x^k - x^{*,k})^{\mathrm{T}}D^k_{a,\tau}(D^k_{b,\tau})^{-1}(x^k + \Delta x^k - x^{*,k})\\ &= (x^k + \Delta x^k - x^{*,k})^{\mathrm{T}}(D^k_{b,\tau})^{-1}r^k\\ &\le \|(D^k_{b,\tau})^{-1}(x^k + \Delta x^k - x^{*,k})\|\,\|r^k\|\end{aligned}$$

and therefore

$$\lambda_{\min}(D^k_{a,\tau}D^k_{b,\tau})\|(D^k_{b,\tau})^{-1}(x^k + \Delta x^k - x^{*,k})\| \le \|r^k\|. \tag{8.15}$$

Using the definition of r^k in (8.14) as well as the fact that

$$\bar{x}_{\mathcal{N}} = 0 \text{ for all } \bar{x} \in S_P$$

and

$$\bar{s}_{\mathcal{B}} = 0 \text{ for all } (\bar{\lambda}, \bar{s}) \in S_D$$

by Lemma 3.13 (1) together with the simple observation that

$$D^k_{a,\tau} + D^k_{b,\tau} = 2I$$

for all $k \in \mathbb{N}$ (recall that φ denotes the minimum function by Assumption 8.4 (1)), we obtain from Lemma 3.13 (2) and Assumption 8.4 (4) that

$$\begin{aligned} r^k_{\mathcal{B}} &= (2I - D^k_{b,\tau})_{\mathcal{B}\mathcal{B}}(x^k_{\mathcal{B}} - x^{*,k}_{\mathcal{B}}) + (D^k_{b,\tau})_{\mathcal{B}\mathcal{B}}(s^k_{\mathcal{B}} - s^{*,k}_{\mathcal{B}}) - 2\min\{x^k_{\mathcal{B}}, s^k_{\mathcal{B}}\} \\ &= (2I - D^k_{b,\tau})_{\mathcal{B}\mathcal{B}}(x^k_{\mathcal{B}} - x^{*,k}_{\mathcal{B}}) + (D^k_{b,\tau})_{\mathcal{B}\mathcal{B}} s^k_{\mathcal{B}} - 2s^k_{\mathcal{B}} \\ &= (2I - D^k_{b,\tau})_{\mathcal{B}\mathcal{B}}\left(x^k_{\mathcal{B}} - x^{*,k}_{\mathcal{B}} - s^k_{\mathcal{B}} + s^{*,k}_{\mathcal{B}}\right) \\ &= (D^k_{a,\tau})_{\mathcal{B}\mathcal{B}}\left(x^k_{\mathcal{B}} - x^{*,k}_{\mathcal{B}} - s^k_{\mathcal{B}} + s^{*,k}_{\mathcal{B}}\right) \end{aligned}$$

and

$$\begin{aligned} r^k_{\mathcal{N}} &= (2I - D^k_{b,\tau})_{\mathcal{N}\mathcal{N}}(x^k_{\mathcal{N}} - x^{*,k}_{\mathcal{N}}) + (D^k_{b,\tau})_{\mathcal{N}\mathcal{N}}(s^k_{\mathcal{N}} - s^{*,k}_{\mathcal{N}}) - 2\min\{x^k_{\mathcal{N}}, s^k_{\mathcal{N}}\} \\ &= (2I - D^k_{b,\tau})_{\mathcal{N}\mathcal{N}} x^k_{\mathcal{N}} + (D^k_{b,\tau})_{\mathcal{N}\mathcal{N}}(s^k_{\mathcal{N}} - s^{*,k}_{\mathcal{N}}) - 2x^k_{\mathcal{N}} \\ &= (D^k_{b,\tau})_{\mathcal{N}\mathcal{N}}\left(s^k_{\mathcal{N}} - s^{*,k}_{\mathcal{N}} - x^k_{\mathcal{N}} + x^{*,k}_{\mathcal{N}}\right). \end{aligned}$$

Therefore, we have

$$\|r^k\| \le \max\left\{\|(D^k_{a,\tau})_{\mathcal{B}\mathcal{B}}\|, \|(D^k_{b,\tau})_{\mathcal{N}\mathcal{N}}\|\right\}\left(\|x^k - x^{*,k}\| + \|s^k - s^{*,k}\|\right). \tag{8.16}$$

Hence (8.15) implies

$$\begin{aligned} &\lambda_{\min}(D^k_{a,\tau}D^k_{b,\tau}) \cdot \|(D^k_{b,\tau})^{-1}(x^k + \Delta x^k - x^{*,k})\| \\ &\quad \le \max\left\{\|(D^k_{a,\tau})_{\mathcal{B}\mathcal{B}}\|, \|(D^k_{b,\tau})_{\mathcal{N}\mathcal{N}}\|\right\} \cdot \left(\|x^k - x^{*,k}\| + \|s^k - s^{*,k}\|\right). \end{aligned} \tag{8.17}$$

Since one can verify that the sequence

$$\left\{\frac{\max\left\{\|(D^k_{a,\tau})_{\mathcal{B}\mathcal{B}}\|, \|(D^k_{b,\tau})_{\mathcal{N}\mathcal{N}}\|\right\}}{\lambda_{\min}(D^k_{a,\tau}D^k_{b,\tau})}\right\} \tag{8.18}$$

remains bounded for $k \longrightarrow \infty$ by Assumption 8.4 (4), it follows from (8.17) and the observation that

$$\|D^k_{b,\tau}\| \le 2$$

for all $k \in \mathbb{N}$ that

$$\begin{aligned}
\frac{1}{2}\left(\|\Delta x^k\| - \|x^k - x^{*,k}\|\right) &\leq \frac{\left|\|\Delta x^k\| - \|x^k - x^{*,k}\|\right|}{2} \\
&\leq \frac{\left|\|\Delta x^k\| - \|x^k - x^{*,k}\|\right|}{\|D_{b,\tau}^k\|} \\
&\leq \frac{\|x^k + \Delta x^k - x^{*,k}\|}{\|D_{b,\tau}^k\|} \\
&\leq \|(D_{b,\tau}^k)^{-1}(x^k + \Delta x^k - x^{*,k})\| \\
&\leq \alpha_1\left(\|x^k - x^{*,k}\| + \|s^k - s^{*,k}\|\right)
\end{aligned}$$

for some constant $\alpha_1 > 0$. As a result, we obtain from (8.8) and Lemma 8.6 that

$$\|\Delta x^k\| \leq \alpha_2\|w^k - w^{*,k}\| = \alpha_2 \operatorname{dist}_S(w^k) = O(\tau_k),$$

where $\alpha_2 > 0$ denotes a suitable constant.

Step 2: Here we show that $\|\Delta s^k\| = O(\tau_k)$. To this end, we premultiply (8.13) with $(D_{b,\tau}^k)^{-1}$ to obtain

$$\begin{aligned}
&\|s^k + \Delta s^k - s^{*,k}\| \\
&\leq \|(D_{b,\tau}^k)^{-1} r^k\| + \|(D_{a,\tau}^k)(D_{b,\tau}^k)^{-1}\left(x^k + \Delta x^k - x^{*,k}\right)\| \\
&\leq \|(D_{a,\tau}^k)^{-1}(D_{b,\tau}^k)^{-1}\|\,\|D_{a,\tau}^k\|\,\|r^k\| + \|D_{a,\tau}^k\|\,\|(D_{b,\tau}^k)^{-1}\left(x^k + \Delta x^k - x^{*,k}\right)\| \\
&\leq 2\lambda_{\max}\left((D_{a,\tau}^k)^{-1}(D_{b,\tau}^k)^{-1}\right)\|r^k\| + 2\|(D_{b,\tau}^k)^{-1}\left(x^k + \Delta x^k - x^{*,k}\right)\| \\
&= 2\frac{1}{\lambda_{\min}(D_{a,\tau}^k D_{b,\tau}^k)}\|r^k\| + 2\|(D_{b,\tau}^k)^{-1}\left(x^k + \Delta x^k - x^{*,k}\right)\|
\end{aligned}$$

since

$$\|D_{a,\tau}^k\| \leq 2$$

for all $k \in \mathbb{N}$. Therefore, using (8.16), (8.17) and the boundedness of the sequence in (8.18), we see that

$$\|s^k + \Delta s^k - s^{*,k}\| \leq \alpha_3\left(\|x^k - x^{*,k}\| + \|s^k - s^{*,k}\|\right)$$

for all $k \in \mathbb{N}$ sufficiently large and a constant $\alpha_3 > 0$. Using Lemma 8.6 and (8.8), this implies

$$\begin{aligned}
\|\Delta s^k\| &\leq \|s^k + \Delta s^k - s^{*,k}\| + \|s^k - s^{*,k}\| \\
&\leq \alpha_4\|w^k - w^{*,k}\| \\
&= \alpha_4 \operatorname{dist}_S(w^k) \\
&= O(\tau_k)
\end{aligned} \tag{8.19}$$

for all $k \in \mathbb{N}$ sufficiently large and a suitable constant $\alpha_4 > 0$.

Step 3: Using (8.19), (8.7a), the full rank of the matrix A immediately implies that

$$\|\Delta\lambda^k\| = O(\tau_k),$$

which proves Step 3.

Putting together Steps 1 through 3, we finally obtain

$$\|\Delta w^k\| = O(\tau_k)$$

for all $k \in \mathbb{N}$ sufficiently large. ■

The next result shows that, eventually, the full predictor direction Δw^k computed in (8.1) satisfies the neighborhood condition

$$\|\Phi_{\tau_k}(w^k + \Delta w^k)\| \leq \beta\tau_k,$$

so that Step 2 of Algorithm 8.1 is eventually successful.

Lemma 8.8 *We have*

$$\|\Phi_{\tau_k}(w^k + \Delta w^k)\| \leq \beta\tau_k$$

for all $k \in \mathbb{N}$ sufficiently large.

Proof. Let us use the abbreviations

$$a_{\tau,i}^k := \frac{\partial\varphi_{\tau_k}}{\partial a}(x_i^k, s_i^k) \quad \text{for } i = 1, \ldots, n$$

and

$$b_{\tau,i}^k := \frac{\partial\varphi_{\tau_k}}{\partial b}(x_i^k, s_i^k) \quad \text{for } i = 1, \ldots, n$$

for the diagonal elements of the matrices $D_{a,\tau}^k$ and $D_{b,\tau}^k$, respectively. Since

$$a_{\tau,i}^k + b_{\tau,i}^k = 2 \quad \text{for all } i = 1, \ldots, n \text{ and for all } k \in \mathbb{N},$$

we obtain from the linear system (8.1) and Assumption 8.4 (1) that

$$(2 - b_{\tau,i}^k)\Delta x_i^k + b_{\tau,i}^k\Delta s_i^k = -2\min\{x_i^k, s_i^k\} \quad \text{for } i = 1, \ldots, n, \tag{8.20}$$

cf. (5.12). Since the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ converges to a strictly complementary solution $w^* = (x^*, \lambda^*, s^*)$ of the optimality conditions (3.5) by Assumption 8.4 (4), Lemma 3.13 (2) implies that there is a constant $\delta > 0$ such that

$$x_i^k \geq \delta \text{ for all } i \in \mathcal{B} \quad \text{and} \quad s_i^k \geq \delta \text{ for all } i \in \mathcal{N} \tag{8.21}$$

and

$$|x_i^k| < \frac{\delta}{2} \text{ for all } i \in \mathcal{N} \quad \text{and} \quad |s_i^k| < \frac{\delta}{2} \text{ for all } i \in \mathcal{B} \tag{8.22}$$

and for all $k \in \mathbb{N}$ large enough.

We now consider an arbitrary index $i \in \mathcal{B}$. Using (8.21) and (8.22), we get

$$x_i^k - s_i^k \geq \frac{\delta}{2}$$

for all $k \in \mathbb{N}$ sufficiently large. Furthermore, (8.20) implies

$$(2 - b_{\tau,i}^k)(\Delta x_i^k - \Delta s_i^k) = -2(s_i^k + \Delta s_i^k). \tag{8.23}$$

The definition of $b_{\tau,i}^k$ gives

$$\begin{aligned}
2 - b_{\tau,i}^k &= 2 - \frac{\partial \varphi_{\tau_k}}{\partial b}(x_i^k, s_i^k) \\
&= 2 - \left(1 + \frac{x_i^k - s_i^k}{\sqrt{(x_i^k - s_i^k)^2 + 4\tau_k^2}}\right) \\
&= 1 - \frac{x_i^k - s_i^k}{\sqrt{(x_i^k - s_i^k)^2 + 4\tau_k^2}} \\
&= \frac{\sqrt{(x_i^k - s_i^k)^2 + 4\tau_k^2} - (x_i^k - s_i^k)}{\sqrt{(x_i^k - s_i^k)^2 + 4\tau_k^2}} \\
&\leq \frac{2\tau_k}{x_i^k - s_i^k} \\
&\leq 4\frac{\tau_k}{\delta}.
\end{aligned}$$

Consequently, we obtain from (8.23) and Lemma 8.7 that

$$\begin{aligned}
2|s_i^k + \Delta s_i^k| &= (2 - b_{\tau,i}^k)|\Delta x_i^k - \Delta s_i^k| \\
&\leq \frac{4\tau_k \left(\|\Delta x^k\| + \|\Delta s^k\|\right)}{\delta} \\
&= O(\tau_k^2).
\end{aligned}$$

Since

$$x_i^k + \Delta x_i^k \geq \frac{\delta}{2}$$

in view of (8.21) and Lemma 8.7, it follows that

$$|\varphi(x_i^k + \Delta x_i^k, s_i^k + \Delta s_i^k)| = 2\left|\min\{x_i^k + \Delta x_i^k, s_i^k + \Delta s_i^k\}\right| = 2|s_i^k + \Delta s_i^k| = O(\tau_k^2)$$

for all $i \in \mathcal{B}$.

In a similar way, we can also show that

$$|\varphi(x_i^k + \Delta x_i^k, s_i^k + \Delta s_i^k)| = 2|\min\{x_i^k + \Delta x_i^k, s_i^k + \Delta s_i^k\}| = 2|x_i^k + \Delta x_i^k| = O(\tau_k^2)$$

for all $i \in \mathcal{N}$. Consequently, we have

$$\|\Phi(w^k + \Delta w^k)\| = O(\tau_k^2)$$

because of Assumption 8.4 (2) and Lemma 7.4. Using Lemma 5.5 as well as the fact that $\beta > \kappa$ by Assumption 8.4 (3), we finally obtain

$$\begin{aligned}\|\Phi_{\tau_k}(w^k + \Delta w^k)\| &\le \|\Phi(w^k + \Delta w^k)\| + \|\Phi_{\tau_k}(w^k + \Delta w^k) - \Phi(w^k + \Delta w^k)\| \\ &\le O(\tau_k^2) + \kappa\tau_k \\ &\le \beta\tau_k\end{aligned}$$

for all $k \in \mathbb{N}$ sufficiently large since $\tau_k \downarrow 0$ by Assumption 8.4 (4). ■

Next we show that the update of the smoothing parameter $\hat{\tau}_k = t_k\tau_k$ in the predictor step of Algorithm 8.1 is such that the factor t_k is in the order of the smoothing parameter τ_k itself.

Lemma 8.9 *We have*

$$t_k = O(\tau_k)$$

for all $k \in \mathbb{N}$ sufficiently large.

Proof. As shown in the proof of Lemma 8.8, we have

$$\|\Phi(w^k + \Delta w^k)\| = O(\tau_k^2).$$

Hence there is a constant $\alpha_5 > 0$ such that

$$\|\Phi(w^k + \Delta w^k)\| \le \alpha_5\tau_k^2$$

for all $k \in \mathbb{N}$ large enough. This implies

$$\begin{aligned}\|\Phi_{t\tau_k}(w^k + \Delta w^k)\| &\le \|\Phi(w^k + \Delta w^k)\| + \|\Phi_{t\tau_k}(w^k + \Delta w^k) - \Phi(w^k + \Delta w^k)\| \\ &\le \alpha_5\tau_k^2 + \kappa t\tau_k\end{aligned}$$

for all $t > 0$, cf. Lemma 5.5. Now, an easy calculation shows that the inequality

$$\alpha_5\tau_k^2 + \kappa t\tau_k \le \beta t\tau_k$$

holds for all $t \ge \alpha_5\tau_k/(\beta - \kappa)$ (recall that $\beta > \kappa$ by Assumption 8.4 (3)). Therefore, the definition of t_k in Step 2 of Algorithm 8.1 implies that

$$\varrho^{\ell_k+1} = \varrho \cdot \varrho^{\ell_k} = \varrho t_k < \frac{\alpha_5}{\beta - \kappa}\tau_k.$$

Hence we have

$$t_k \leq \frac{\alpha_5}{\varrho(\beta - \kappa)} \tau_k = O(\tau_k)$$

for all $k \in \mathbb{N}$ large enough. ■

We are now in the position to state our main local convergence result for Algorithm 8.1. It says that we eventually have local quadratic convergence of the smoothing parameter τ_k to zero.

Theorem 8.10 *We have*

$$\tau_{k+1} = O(\tau_k^2)$$

for all $k \in \mathbb{N}$ large enough, i. e., the sequence τ_k converges to zero at least Q-quadratically.

Proof. In view of Lemma 8.9 and the updating rules for the smoothing parameter in Algorithm 8.1, we have

$$\tau_{k+1} \leq \hat{\tau}_k = t_k \tau_k = O(\tau_k^2)$$

for all $k \in \mathbb{N}$ sufficiently large. ■

It appears very likely that Theorem 8.10 is also true for some of the other NCP-functions mentioned in example 5.2, or at least for the Fischer-Burmeister function, but currently there is no formal proof for this.

8.1.3 Numerical Results

As the predictor-corrector method 6.1, Algorithm 8.1 was implemented in MATLAB by modifying the LIPSOL code by Zhang [57, 58].

Numerous approaches to choose a starting point $w^0 = (x^0, \lambda^0, s^0)$ have been tested. There are several possibilities for choosing w^0.

(1) One could take the starting point provided by LIPSOL. However, this starting point is optimized for interior-point methods and there is no reason why it should be useful for the smoothing-type methods as well. In particular, $w^0 = (x^0, \lambda^0, s^0)$ does not, in general, satisfy the linear equations $A^T\lambda^0 + s^0 = c$ and $Ax^0 = b$, although this might be an important property for Algorithm 8.1, see, in particular, Proposition 5.7 and Theorem 7.7.

(2) For the implementation of the Jacobian smoothing method introduced in Chapter 6 (see also [23]) a starting point $w^0 = (x^0, \lambda^0, s^0)$ is used, whose components are computed as follows:

   a) Compute $y^0 \in \mathbb{R}^m$ by solving $AA^T y = b$ using a sparse Cholesky code.
   b) Set $x^0 := A^T y^0$.

c) Define $\lambda^0 := 0$ and $s^0 := c$.

Obviously, the resulting vector $w^0 = (x^0, \lambda^0, s^0)$ satisfies the linear equations $A^T\lambda^0 + s^0 = c$ and $Ax^0 = b$. However, the two vectors x^0 and s^0 typically have many negative components.

(3) The initial point $w^0 = (x^0, \lambda^0, s^0)$ actually used in the implementation of the predictor-corrector method 8.1 is calculated as follows:

a) Compute $y^0 \in \mathbb{R}^m$ by solving $AA^Ty = b$ using a sparse Cholesky code.

b) Set $x^0 := A^Ty^0$.

c) Solve $AA^T\lambda = Ac$ using a sparse Cholesky code in order to compute $\lambda^0 \in \mathbb{R}^m$.

d) Set $s^0 := c - A^T\lambda^0$.

One arrives at this starting point by solving the two simple programs

$$\min \frac{1}{2}\|x\|^2 \quad \text{subject to} \quad Ax = b$$

for x^0 and

$$\min \frac{1}{2}\|s\|^2 \quad \text{subject to} \quad A^T\lambda + s = c$$

for λ^0 and s^0. Again, the resulting vector $w^0 = (x^0, \lambda^0, s^0)$ satisfies the linear equations $A^T\lambda^0 + s^0 = c$ and $Ax^0 = b$. The difference between this starting vector and the one described in the previous procedure (2) lies in the computation of the vector λ^0. The idea behind the strategy here is to have possibly many components of s^0 equal to zero (or at least small) by solving the equation $A^T\lambda = c$ in the least squares sense. The advantage of having zero (or small) components in s^0 is that the initial residual $\|\Phi(w^0)\|$ is smaller, so that this starting point w^0 is hopefully closer to the solution set of (3.5).

However, the choice of the starting point is crucial and the procedure given in (3) is not always better than the one stated in (2). Furthermore, since interior-point and smoothing-type methods use different starting points, it is difficult to compare both methods with each other.

Another crucial part of the implementation is the stopping criterion. Interior-point methods typically terminate the iteration if the (weighted) duality gap

$$\mu_k := \frac{(x^k)^T s^k}{n} \tag{8.24}$$

(or a certain residual vector whose size is controlled by the duality gap) is sufficiently small in the sense that

$$\mu_k < 10^{-8}.$$

This criterion cannot be used by our method since the x- and s-components of the iterates $w^k = (x^k, \lambda^k, s^k)$ are no longer guaranteed to stay positive. Hence it is necessary to use a different stopping criterion. The one implemented is

$$\tau_k < 10^{-4}, \tag{8.25}$$

where, of course, τ_k denotes the smoothing parameter. The motivation for this stopping rule is as follows: Using the number μ_k from (8.24), interior-point methods basically follow the central path conditions

$$\begin{aligned} A^{\mathrm{T}}\lambda + s &= c, \\ Ax &= b, \\ x_i > 0,\ \ s_i > 0,\ \ x_i s_i &= \mu_k \quad \forall i = 1, \dots, n \end{aligned}$$

parameterized by μ_k. Comparing with (5.21), it therefore follows that μ_k plays (more or less) the role of τ_k^2, hence the criterion $\mu_k < 10^{-8}$ corresponds to $\tau_k^2 < 10^{-8}$ or, equivalently, to $\tau_k < 10^{-4}$. Moreover, in view of Proposition 7.3 and Lemma 5.5, the violation $\|\Phi(w^k)\|$ is also bounded from above by a constant times τ_k.

After some preliminary experiments with Algorithm 8.1, is was observed that the predictor step is accepted only very few times. Moreover, the situation did not improve by allowing some kind of restricted line search in the predictor step; by this it is meant that if the full step Δw^k did not satisfy the neighborhood criterion (8.6), Δw^k was replaced by, say, $1/2\Delta w^k$ or even $1/4\Delta w^k$ and accepted this step in case (8.6) is satisfied with this shorter direction.

It turned out that this strategy works more successful if the right-hand side in (8.1) is replaced by $\Phi_{\tau_k}(w^k)$, so that the predictor step becomes a real smoothing step. Since this does not necessarily guarantee local fast convergence under Assumption 8.4, it was finally decided to implement a three-step modification of Algorithm 8.1. This modification adds another step between the predictor step (Step 2) and the corrector step (Step 3), let us call it Step 2a. Step 2a is exactly the same as Step 2 only that the right-hand side in (8.1) is replaced by $\Phi_{\tau_k}(w^k)$. In the following, we call Step 2 the *Jacobian smoothing predictor step*, and Step 2a the *smoothing predictor step*, while Step 3 is still be called the *corrector step.* Thus, Step 2a has the following form:

Step 2a (Smoothing Predictor Step):

Compute a solution $\Delta w^k = (\Delta x^k, \Delta\lambda^k, \Delta s^k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ of the linear system

$$\Phi'_{\tau_k}(w^k)\Delta w = -\Phi_{\tau_k}(w^k).$$

If $\|\Phi(w^k + \Delta w^k)\| \leq \varepsilon$: Stop. Otherwise, if

$$\|\Phi_{\tau_k}(w^k + \Delta w^k)\| > \beta\tau_k,$$

then set

$$\hat{w}^k := w^k, \quad \hat{\tau}_k := \tau_k,$$

else compute $t_k = \varrho^{\ell_k}$, where ℓ_k is the non-negative integer such that

$$\begin{aligned}&\|\Phi_{\varrho^j \tau_k}(w^k + \Delta w^k)\| \le \beta \varrho^j \tau_k \quad \forall j = 0, 1, 2, \dots, \ell_k \text{ and}\\ &\|\Phi_{\varrho^{\ell_k+1}\tau_k}(w^k + \Delta w^k)\| > \beta \varrho^{\ell_k+1}\tau_k,\end{aligned}$$

and set $\hat{\tau}_k := t_k \tau_k$ and

$$\hat{w}^k := \begin{cases} w^k & \text{if } \ell_k = 0,\\ w^k + \Delta w^k & \text{otherwise.}\end{cases}$$

Obviously, this modified method has the same local and global convergence properties as Algorithm 8.1, but it seems to work better from a practical point of view.

On the other hand, the amount of work per iteration is higher for this three-step method than for Algorithm 8.1. In theory, it could happen that up to three linear systems with different coefficient matrices need to be solved. Algorithm 8.1 on the other hand required the factorization of at most two matrices per iteration. However, if the Jacobian smoothing predictor step and/or the smoothing predictor step cannot be accepted, then all three, or at least two of the coefficient matrices are identical. In this case, we still solve three linear systems, but the amount of work is significantly smaller since we factorize only one or two matrices per iteration.

In practice, averaging over the test suite used for the numerical experiments, it turns out that we have to factorize less than two matrices per iteration. It should be mentioned, however, that this is still more than what is done by LIPSOL and many other implementations of interior-point methods which typically solve two linear systems per iteration having the same coefficient matrix so that only one factorization is needed.

Taking into account the modifications mentioned above, Algorithm 8.1 was implemented using the penalized minimum function and the parameters

$$\tau_0 := \min\left\{\|\Phi(w^0)\|, 10^5\right\}, \quad \beta := \frac{\|\Phi_{\tau_0}(w^0)\|}{\tau_0}, \quad \varrho = 0.9, \quad \sigma = 10^{-4}.$$

In addition to the termination criterion $\tau_k < 10^{-4}$ (cf. (8.25)), the algorithm also terminates after 150 iterations. The step length-based emergency stopping criteria described in Section 6.3.2 are implemented as well.

Experiments showed that for this algorithm the penalized minimum function does indeed give the best results, as far as the total number of iterations is concerned.

The program was run on a Sun Enterprise 450 on UltraSPARC II processors. The results for the complete Netlib test suite are given in Table 8.1, whose columns have the following meanings:

Problem:	Name of the test problem in the Netlib collection
m:	Number of rows (after preprocessing)
n:	Number of columns (after preprocessing)
k.:	Number of iterations until termination
J:	Number of accepted Jacobian smoothing predictor steps
S:	Number of accepted smoothing predictor steps
τ_f:	Value of τ_k in final iterate
$\|\Phi(w^f)\|_\infty$:	Value of $\|\Phi(w^k)\|_\infty$ in final iterate
Primal Objective:	Value of primal objective function in final iterate.

Table 8.1: Numerical results of the three-step predictor-corrector method

Problem	m	n	k	J	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective
25fv47	798	1854	20	4	15	4.23e−05	2.972e−03	5.50184572e+03
80bau3b	2235	11516	31	7	22	9.72e−05	1.084e−02	9.87224192e+05
adlittle	55	137	10	4	6	4.92e−05	6.716e−04	2.25494960e+05
afiro	27	51	6	6	4	2.11e−07	2.682e−06	−4.64739283e+02
agg	488	615	24	7	14	1.59e−06	4.800e−05	−3.59917679e+07
agg2	516	758	21	8	17	6.06e−08	2.698e−06	−2.02392524e+07
agg3	516	758	23	5	15	2.46e−05	3.885e−04	1.03121159e+07
bandm	269	436	13	6	9	1.76e−05	3.163e−04	−1.58628020e+02
beaconfd	148	270	17	4	12	1.95e−05	3.304e−04	3.35924858e+04
blend	74	114	8	3	6	5.00e−06	8.538e−05	−3.08121507e+01
bnl1	632	1576	21	5	16	9.08e−05	2.192e−03	1.97762726e+03
bnl2	2268	4430	30	5	17	7.34e−05	6.468e−03	1.81123654e+03
boeing1	347	722	21	4	17	2.98e−05	5.541e−04	−3.35214102e+02
boeing2	140	279	20	5	14	8.03e−05	2.689e−03	−3.15018724e+02
bore3d	199	300	26	5	12	1.27e−06	2.882e−05	1.37308039e+03
brandy	149	259	15	8	10	1.22e−05	3.153e−04	1.51850990e+03
capri	267	476	13	6	12	6.20e−05	2.011e−03	2.69001283e+03
cycle	1801	3305	124	2	9	9.55e−05	2.425e−03	−5.06987267e+00
czprob	737	3141	22	4	12	3.46e−06	2.035e−04	2.18519670e+06
d2q06c	2171	5831	65	2	29	9.46e−05	8.490e−03	1.22784238e+05
d6cube	404	6184	8	5	6	6.49e−05	9.216e−03	3.15491659e+02
degen2	444	757	6	5	4	1.16e−05	2.441e−04	−1.43517800e+03
degen3	1503	2604	11	6	5	3.60e−05	1.071e−03	−9.87294069e+02
dfl001	6071	12230	> 150			*maximum number of iterations exceeded*		
e226	220	469	11	7	9	6.92e−05	2.279e−03	−1.87519260e+01
etamacro	357	692	16	4	15	4.80e−05	7.180e−04	−7.55715241e+02
fffff800	501	1005	33	6	23	3.75e−05	9.561e−04	5.55679578e+05
finnis	492	1014	21	3	20	5.97e−05	1.380e−03	1.72791093e+05
fit1d	24	1049	27	6	12	1.12e−08	6.994e−07	−9.14637809e+03
fit1p	627	1677	11	1	2	*stagnating corrector step length*		
fit2d	25	10524	13	6	6	8.97e−06	1.596e−03	−6.84642932e+04
fit2p	3000	13525	11	1	2	*stagnating corrector step length*		
forplan	135	463	34	3	17	1.27e−05	4.200e−04	−6.64218236+02
ganges	1137	1534	18	7	16	2.46e−05	1.117e−03	−1.09585736e+05
gfrd−pnc	600	1144	30	5	11	3.04e−05	7.234e−04	6.90223600e+06
greenbea	2318	5424	56	6	14	5.93e−06	7.958e−04	−7.24625478e+07
greenbeb	2317	5415	41	4	21	*stagnating corrector step length*		
israel	174	316	29	5	15	1.46e−06	4.085e−05	−8.96644822e+05
kb2	43	68	19	7	4	7.83e−05	3.024e−04	−1.74990013e+03

Table 8.1: Numerical results of the three-step predictor-corrector method (continued)

Problem	m	n	k	J	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective
lotfi	151	364	38	3	18	5.72e−05	7.785e−04	−2.52644858e+01
maros	835	1921	60	4	25	8.95e−05	5.867e−03	−5.80630232e+04
maros-r7	3136	9408	23	5	7	4.10e−05	3.919e−03	1.49718517e+06
modszk1	686	1622	28	8	16	3.69e−06	2.555e−04	3.20619728e+02
nesm	654	2922	32	3	25	8.09e−05	4.882e−03	1.40760365e+07
perold	625	1530	58	3	24	5.40e−05	2.456e−03	−9.38065157e+03
pilot	1441	4657	36	2	25	7.56e−05	5.118e−03	−5.57483879e+02
pilotja	924	2044	97	1	30	7.02e−05	1.665e−03	−6.11207802e+03
pilotwe	722	2930	34	1	28	6.92e−05	2.407e−03	−2.72010735e+06
pilot4	402	1173	41	3	22	4.80e−05	1.364e−03	−2.58113492e+03
pilot87	2030	6460	122	3	23	3.44e−05	1.703e−03	3.017486460e+02
pilotnov	951	2242	47	4	16	2.80e−05	2.462e−03	−4.49727619e+03
recipe	85	177	7	4	5	6.98e−06	9.192e−05	−2.66616001e+02
sc105	105	163	9	4	7	2.88e−05	9.198e−05	−5.22020560e+01
sc205	205	317	30	4	5	6.25e−06	1.052e−04	−5.22020613e+01
sc50a	49	77	11	4	5	2.84e−05	2.130e−04	−6.45750516e+01
sc50b	48	76	8	4	3	5.55e−05	2.475e−05	−6.99999917e+01
scagr25	471	671	27	5	10	2.92e−10	9.765e−09	−1.47534331e+07
scagr7	129	185	15	6	8	5.89e−07	1.114e−05	−2.33138982e+06
scfxm1	322	592	15	5	12	9.16e−05	3.799e−03	1.84167590e+04
scfxm2	644	1184	19	5	11	2.18e−06	1.403e−04	3.66602616e+04
scfxm3	966	1776	19	7	14	6.83e−07	2.837e−05	5.49012545e+04
scorpion	375	453	15	3	10	2.93e−06	7.267e−05	1.87812482e+03
scrs8	485	1270	13	4	11	8.57e−05	3.904e−03	9.04296952e+02
scsd1	77	760	6	4	5	5.73e−11	8.188e−10	8.66666667e+00
scsd6	147	1350	7	5	4	1.77e−05	2.211e−04	5.05000001e+01
scsd8	397	2750	4	4	4	5.69e−05	3.501e−03	9.04986876e+02
sctap1	300	660	13	5	5	4.68e−05	2.275e−03	1.41225000e+03
sctap2	1090	2500	11	6	7	4.91e−07	1.671e−05	1.72480714e+03
sctap3	1480	3340	9	5	6	3.06e−06	3.146e−04	1.42400000e+03
seba	515	1036	10	10	0	*stagnating corrector step length*		
share1b	112	248	33	6	22	9.45e−05	1.294e−03	−7.65893186e+04
share2b	96	162	10	2	8	7.63e−07	1.048e−05	−4.15732241e+02
shell	496	1487	19	7	15	3.92e−09	9.638e−08	1.20882535e+09
ship04l	356	2162	17	5	7	1.406e−06	1.136e−04	1.79332454e+06
ship04s	268	1414	14	6	5	1.86e−08	9.237e−07	1.79871470e+06
ship08l	688	4339	19	8	6	9.18e−05	1.182e−02	1.90905521e+06
ship08s	416	2171	18	7	5	6.31e−06	4.677e−04	1.92009821e+06
ship12l	838	5329	18	5	10	1.41e−05	1.268e−03	1.47018792e+06
ship12s	466	2293	14	5	7	2.36e−06	1.659e−04	1.48923613e+06
sierra	1222	2715	18	9	9	4.49e−07	2.065e−05	1.53943623e+07
stair	356	538	12	6	10	1.96e−05	4.306e−04	−2.51268061e+02
standata	359	1258	13	6	8	1.91e−06	8.002e−05	1.25769934e+03
standgub	361	1366	7	5	5	2.29e−08	5.693e−07	1.25769941e+03
standmps	467	1258	13	5	7	1.86e−08	6.371e−07	1.40601750e+03
stocfor1	109	157	18	2	6	6.22e−06	8.962e−05	−4.11319765e+04
stocfor2	2157	3045	32	5	10	4.53e−05	3.001e−03	−3.90244086e+04
stocfor3	16675	23541	65	4	20	7.34e−05	8.415e−03	−3.99767845e+04
stocfor3old	16675	23541	45	2	19	9.74e−05	1.237e−02	−3.99767840e+04
truss	1000	8806	15	6	6	4.15e−07	2.641e−05	4.58815845e+05
tuff	292	617	22	5	17	9.98e−05	3.865e−03	2.92147322e−01

Table 8.1: Numerical results of the three-step predictor-corrector method (continued)

Problem	m	n	k	J	P	τ_f	$\\|\Phi(w^f)\\|_\infty$	Primal Objective
vtpbase	194	325	13	7	10	2.32e−07	8.298e−06	1.29831463e+05
wood1p	244	2595	10	5	7	4.36e−05	3.447e−03	1.44290241e+00
woodw	1098	8418	15	2	11	9.49e−05	1.276e−02	1.30445782e+00

The examples `grow7`, `grow15`, and `grow22` have not been included in Table 8.1, as they exhibit a quite interesting behavior which requires an in-depth look. Table 8.2 first gives those results which have not been included in table 8.1.

Table 8.2: Numerical results of the `grow*` examples (author's initial point)

problem	m	n	k	J	S	τ_f	$\\|\Phi(w^f)\\|_\infty$	Primal Objective
grow7	140	301	130	3	5	9.9175e−07	1.9719e−05	−4.7787811822e+07
grow15	300	645	201	3	6	4.1590e−07	8.0793e−06	−1.0687094132e+08
grow22	440	946	295	3	5	1.0204e−09	1.4146e−08	−1.6083433648e+08

It is notable that the number of iterations required to obtain a solution is quite large with up to almost 300 iteration for the problem `grow22`. Interestingly, the results improve dramatically if the initial point is chosen differently. Table 8.3 gives the results for the `grow*` test examples, this time using the default starting point provided by LIPSOL.

Table 8.3: Numerical results of the `grow*` examples (LIPSOL's initial point)

problem	m	n	k	J	S	τ_f	$\\|\Phi(w^f)\\|_\infty$	Primal Objective
grow7	140	301	33	4	11	3.8723e−10	5.1334e−09	−4.7787811815e+07
grow15	300	645	29	5	12	6.6616e−10	1.1677e−08	−1.0687094129e+08
grow22	440	946	24	4	12	5.0103e−05	9.5985e−04	−1.6083433738e+08

Even though the iteration count is enormous when Algorithm 8.1 is applied with the default initial point, the local behavior, should be looked at more closely. Consider, for example, the last five iterations for the problem `grow22`. The values in table 8.4 demonstrate the local quadratic rate of convergence quite nicely.

Table 8.4: Local behavior of `grow22` (author's initial point)

k	τ_k	$\\|\Phi(w^k)\\|$	Primal Objective
291	1.0544e+00	1.0384e+01	−1.6083406637e+08
292	1.0544e+00	1.0453e+01	−1.6083419330e+08
293	5.6036e−01	2.6892e+00	−1.6083431197e+08
294	1.3814e−03	3.6227e−05	−1.6083433648e+08
295	9.1839e−10	5.4320e−08	−1.6083433648e+08

Looking at Table 8.1, it is also notable that the failure to solve the problem `dfl001` persists. The test cases `greenbea` and `greenbeb` as well as `stocfor3` could be solved by Algorithm 8.1 without problems. Problems `fit1p` and `fit2p` still produce failures due to a too small step length in the corrector step. In contrast to the Jacobian smoothing method 6.1 Algorithm 8.1 fails to solve the examples `grow15` and `grow22` within the given maximal number of iterations. Other problems, such as `cycle`, `pilot.ja`, or `pilot87` require a large number of iterations to obtain a solution. Still, it has to be noted that the total number of errors dropped to 7 for this method. Opposed to the Jacobian smoothing method 6.1, Algorithm 8.1 performs quite well, even on larger problems, such as `maros-r7` or `modzki1`. This can be viewed as another major improvement of the predictor-corrector approach compared to the pure Jacobian smoothing method, especially as most problems could be solved in less than 40–50 iterations.

8.2 Global and Quadratic Convergence of a Predictor-Corrector Method with a Smoothing Variable

In this section another smoothing-type predictor-corrector method based on the method introduced in Section 7.2 is presented. This method follows an idea developed by Jiang [30]. The algorithm presented in this section is quite similar to the one proposed by Burke and Xu [8] (see also [5]). As the method introduced in Section 8.1, the algorithm is a predictor-corrector method with the corrector step being responsible for the global convergence and the predictor step guaranteeing local fast convergence under suitable assumptions. In fact, the corrector step of Algorithm 8.11 is identical to the one used by Burke and Xu [8], but we will prove a different global convergence result, using less stringent assumptions. The predictor step is essentially the same as in Algorithm 8.1 (and was introduced by Chen, Qi and Sun [14]) and can be shown to be locally quadratically convergent under weaker assumptions than those used by Burke and Xu [8]. Due to its stronger convergence properties it improves Algorithm 8.1. The numerical results stated in Table 8.5 are somewhat better than the ones for Algorithm 8.1.

8.2.1 Description of Algorithm

For this algorithm we use the same approach as for Algorithm 7.8, i. e., we consider the smoothing parameter τ as an independent variable.

For the remainder of this section, let φ always denote the minimum function (5.3) from example 5.2 and φ_τ its smoothed counterpart, the Chen-Harker-Kanzow-Smale smoothing function (5.7) from example 5.3.

We will also employ the mappings θ (cf. (7.16)) and Θ (cf. (7.17)) which were already introduced in Section 7.2.1.

The only difference between Algorithm 8.11 and Algorithm 7.8 is the addition of a predictor step to the following algorithm. This predictor step will allow us to prove that the resulting algorithm converges locally Q-quadratically.

Algorithm 8.11
Step 0 (Initialization):

Choose $w^0 := (x^0, \lambda^0, s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ such that $A^{\mathrm{T}}\lambda^0 + s^0 = c, Ax^0 = b$, choose $\tau_0 > 0$, select $\beta \geq \|\Phi_{\tau_0}(w^0)\|/\tau_0, \varrho \in (0,1), 0 < \hat{\sigma}_{\min} < \hat{\sigma}_{\max} < 1, \varepsilon \geq 0$, and set the iteration counter $k := 0$.

Step 1 (Termination Criterion):

If $\|\Phi(w^k)\| \leq \varepsilon$: Stop.

Step 2 (Predictor Step):

Compute a solution $(\Delta w^k, \Delta\tau_k) = (\Delta x^k, \Delta\lambda^k, \Delta s^k, \Delta\tau_k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}$ of the linear system

$$\Theta'(w^k, \tau_k) \begin{bmatrix} \Delta w \\ \Delta\tau \end{bmatrix} = -\Theta(w^k, 0). \tag{8.26}$$

If $\|\Phi(w^k + \Delta w^k)\| \le \varepsilon$: Stop. Otherwise, if

$$\|\theta(x^k + \Delta x^k, s^k + \Delta s^k, \tau_k)\| > \beta\tau_k,$$

then set

$$\hat{w}^k := w^k, \;\; \hat{\tau}_k := \tau_k, \;\; \eta_k := 1,$$

else compute $\eta_k = \varrho^{\ell_k}$, where ℓ_k is the non-negative integer such that

$$\|\theta(x^k + \Delta x^k, s^k + \Delta s^k, \varrho^j\tau_k)\| \le \beta\varrho^j\tau_k \quad \forall j = 0,1,2,\dots,\ell_k \text{ and}$$
$$\|\theta(x^k + \Delta x^k, s^k + \Delta s^k, \varrho^{\ell_k+1}\tau_k)\| > \beta\varrho^{\ell_k+1}\tau_k,$$

and set $\hat{\tau}_k := \eta_k\tau_k$ and

$$\hat{w}^k := \begin{cases} w^k & \text{if } \ell_k = 0, \\ w^k + \Delta w^k & \text{otherwise.} \end{cases}$$

Step 3 (Corrector Step):
Choose $\hat{\sigma}_k \in [\hat{\sigma}_{\min}, \hat{\sigma}_{\max}]$.
Compute a solution $(\Delta\hat{w}^k, \Delta\hat{\tau}_k) = (\Delta\hat{x}^k, \Delta\hat{\lambda}^k, \Delta\hat{s}^k, \Delta\hat{\tau}_k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}$ of the linear system

$$\Theta'(\hat{w}^k, \hat{\tau}_k) \begin{bmatrix} \Delta\hat{w} \\ \Delta\hat{\tau} \end{bmatrix} = -\Theta_{\hat{\sigma}_k}(\hat{w}^k, \hat{\tau}_k). \tag{8.27}$$

Let $\hat{t}_k = \max\left\{ \varrho^\ell \;\middle|\; \ell = 0,1,2,\dots \right\}$ such that

$$\|\theta(\hat{x}^k + \hat{t}_k\Delta\hat{x}^k, \hat{s}^k + \hat{t}_k\Delta\hat{s}^k, (1-\hat{\sigma}_k\hat{t}_k)\hat{\tau}_k)\| \le \beta(1-\hat{\sigma}_k\hat{t}_k)\hat{\tau}_k. \tag{8.28}$$

Set $w^{k+1} := \hat{w}^k + \hat{t}_k\Delta\hat{w}^k$ and $\tau_{k+1} := (1-\hat{\sigma}_k\hat{t}_k)\hat{\tau}_k$.

Step 4 (Update):
Set $k \longleftarrow k+1$, and go to Step 1.

To get a better understanding of the way Algorithm 8.11 works, let us add a couple of comments. In Step 0, we require the starting point $w^0 = (x^0, \lambda^0, s^0)$ to be feasible with respect to the linear equations $A^T\lambda + s = c$ and $Ax = b$. Since the components x^0 and s^0 do not have to be positive (in contrast to interior-point methods), it is relatively easy to find such a starting point.

In the predictor step, we first compute a search direction by solving the linear system (8.26). The interesting part about this linear system is the fact that the right-hand side of (8.26) is unperturbed with respect to τ, whereas we use the standard Jacobian of the perturbed function $\Theta(w,\tau)$ on the left. This may be viewed as the counterpart of the affine scaling step typically used as a predictor step in primal-dual interior-point methods, see Wright [56]. Like in the interior-point setting, this predictor step will eventually guarantee local fast convergence (under suitable assumptions).

After having computed the search direction in (8.26), we try to reduce the smoothing parameter τ_k as much as possible with the only restriction that the full step stays within a certain neighborhood of the central path (cf. Lemma 7.10 (3), which still holds for Algorithm 8.11).

While our predictor step is different from the one used by Burke and Xu [8] in their smoothing-type method, the corrector step coincides with the one from [8], as already mentioned in the comments following Algorithm 7.8. Note that the linear system (8.27) is precisely the one from (7.20) and includes a perturbation on the right-hand side as well. The predictor step also contains a procedure to reduce the smoothing parameter. This procedure will guarantee global convergence in the sense that τ_k decreases to zero under mild conditions.

8.2.2 Convergence Analysis

This section investigates the global and local convergence properties of Algorithm 8.11. To this end, we assume throughout this section that the termination parameter ε is equal to zero, and that Algorithm 8.11 generates an infinite number of iterates w^k, i. e., we assume that the algorithm does not stop after a finite number of iterations in a point w^k satisfying the optimality conditions (3.5).

Since Algorithm 8.11's corrector step coincides with Step 3 of Algorithm 7.8, many of the following results can be taken directly from Section 7.2.2.

We now note that Algorithm 8.11 is well defined.

Lemma 8.12 *The following statements hold for any $k \in \mathbb{N}$:*

(1) The linear systems (8.26) *and* (8.27) *have a unique solution.*

(2) There exists a unique η_k satisfying the conditions in Step 2.

(3) The step size $\hat{t}_k$ in Step 3 is uniquely defined.

(4) $\|\theta(x^k, s^k, \tau_k)\| \le \beta\tau_k$ for all $k \in \mathbb{N}$.

Proof.

(1) The structure of the Jacobian $\Theta'(w^k, \tau_k)$ (cf. (7.19)) shows that this matrix is non-singular if and only if $\Phi'_{\tau_k}(w^k)$ is non-singular. This has already been shown in Proposition 5.6.

(2) This statement follows with the same arguments as Proposition 8.2.

(3), (4) These results can be proven analogously to parts (2) and (3) of Lemma 7.9 ∎

Note that as a consequence, Algorithm 8.11 is well-defined.

The following properties of Algorithm 8.11 will be used a number of times later on. The parts (1) and (4) are identical to the corresponding properties listed in Lemma 7.9 and Lemma 7.10. Part (2) needed to be adapted to accommodate the predictor step's step length variable η_k.

Lemma 8.13 *The sequences* $\{(x^k, \lambda^k, s^k)\}$ *and* $\{\tau_k\}$ *generated by Algorithm 8.11 have the following properties:*

(1) $A^T\lambda^k + s^k = c$ *and* $Ax^k = b$ *for all* $k \in \mathbb{N}$.

(2) $\tau_k = \tau_{k-1}(1 - \hat{\sigma}_{k-1}\hat{t}_{k-1})\eta_{k-1} \cdot \ldots \cdot \tau_0(1 - \hat{\sigma}_0\hat{t}_0)\eta_0$ *for all* $k \in \mathbb{N}$.

Proof.

(1) The assertion follows with the same arguments as used in the proof of Lemma 7.10 (1).

(2) Step 2 of Algorithm 8.11 implies that $\hat{\tau}_k = \eta_k\tau_k$. The updating rules in Step 3 therefore give

$$\tau_{k+1} = (1 - \hat{\sigma}_k\hat{t}_k)\hat{\tau}_k = (1 - \hat{\sigma}_k\hat{t}_k)\eta_k\tau_k$$

for all $k \in \mathbb{N}$. This gives the desired formula. ■

We can now state a global convergence result for Algorithm 8.11. Note that this result is different from the one provided by Burke and Xu [8]. (They use an assumption which is even stronger than the one we use for our local convergence result in Theorem 8.16; on the other hand, the main emphasis in [8] was to prove a global linear rate of convergence result.)

Theorem 8.14 *Assume that the sequence* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ *generated by Algorithm 8.11 has at least one accumulation point. Then* $\{\tau_k\}$ *converges to zero.*

Proof. Since the sequence $\{\tau_k\}$ is monotonically decreasing and bounded from below by zero, it converges to a number $\tau_* \geq 0$. If $\tau_* = 0$, we are done.

So assume that $\tau_* > 0$. Then the updating rules in Step 2 of Algorithm 8.11 immediately give

$$\hat{w}^k = w^k, \quad \hat{\tau}_k = \tau_k, \quad \text{and} \quad \eta_k = 1 \tag{8.29}$$

for all $k \in \mathbb{N}$ sufficiently large. Taking this into account, the remainder of the proof is identical to that of Theorem 7.13. The results from Section 7.2.2 employed in this proof still hold for Algorithm 8.11. ■

We now give a local convergence result. To this end, we first note that the search direction we obtain in our predictor step is identical to the one obtained in the predictor step of Algorithm 8.1 (see also [20]).

Lemma 8.15 *The vector* $(\Delta w^k, \Delta\tau_k)$ *is a solution of the linear system* (8.26) *if and only if* Δw^k *solves the system*

$$\Phi'_{\tau_k}(w^k)\Delta w = -\Phi_0(w^k),$$

and $\Delta\tau_k = 0$.

Proof. Since the smoothing parameter on the right-hand side of the linear system (8.26) is equal to zero, the assertion follows immediately from the discussion following (7.19). ∎

The previous result implies that we can apply the local rate of convergence analysis from Section 8.1.2 (see also [20]). Hence we obtain the following result, which is equivalent to Theorem 8.10 (see also Tseng [55]).

Theorem 8.16 *Assume that the sequence $\{w^k\}$ generated by Algorithm 8.11 converges to a strictly complementary solution of the optimality conditions* (3.5). *Suppose further that the parameter β from Step 0 of Algorithm 8.11 is chosen sufficiently large such that $\beta > 2\sqrt{n}$ (cf. Lemma 5.5). Then the predictor step is eventually accepted, and we have*

$$\tau_{k+1} = O(\tau_k^2)$$

for all $k \in \mathbb{N}$ sufficiently large, i.e., the smoothing parameter converges locally Q-quadratically to zero.

Note that a typical interior-point method can guarantee the convergence of the corresponding iteration sequence to a strictly complementary solution, so from this point of view, the assumptions we use in Theorem 8.16 are stronger. However, this is basically the only difference, in particular, we stress that the assumptions used in Theorem 8.16 do not necessarily imply that the solution set of the optimality conditions 3.5 reduces to a singleton.

We close this section by noting that all results (with the possible exception of Theorem 8.16) would still be true if φ would denote the Fischer-Burmeister function from (5.2) together with its smooth counterpart from (5.6); this can be seen by an easy inspection of the above proofs. On the other hand, it is currently an open question whether Theorem 8.16 also holds for the Fischer-Burmeister function or not.

8.2.3 Numerical Results

Algorithm 8.11 has been implemented in MATLAB, utilizing once again Zhang's interior-point code LIPSOL [57, 58].

The starting point for the smoothing-type adaption of Zhang's code is computed as described in number (3) on page 105 in Section 8.1.3 for Algorithm 8.1.

Furthermore, the initial smoothing parameter τ_0 is taken such that

$$\tau_0 \geq \sqrt{x_i^0 s_i^0} \quad \text{for all } i \in \{1, \dots, n\} \text{ with } x_i^0 > 0 \text{ and } s_i^0 > 0.$$

This choice guarantees that we have $\theta(x^0, s^0, \tau_0) \leq 0$ (for both the minimum and the Fischer-Burmeister function).

The iteration is terminated if one of the following conditions holds:

(1) $\tau_k < 10^{-4}$ or

(2) $\|\Phi(w^k)\|_\infty < 10^{-4}$ or

(3) $\|\Phi(w^k)\|_\infty < 10^{-3}$ and $\|\Phi(w^k)\|_\infty / \|\Phi(w^0)\|_\infty < 10^{-6}$.

Note that we do use the equation operator Φ from (5.1) rather than the operator Θ from (7.17), this allows for a better comparison of the results.

Criterion (1) was used for the implementation of Algorithm 8.1 and is motivated by the fact that the square of τ does, more or less, play the role of the duality gap in interior-point methods (cf. (4.2)) for which 10^{-8} is a typical value for the stopping parameter. Criterion (2) is an absolute error measuring the total residual $\|\Phi(w^k)\|_\infty$, whereas (3) is a mixture between a weakened form of this absolute error and a relative error comparing the k-th residual $\|\Phi(w^k)\|_\infty$ with the initial residual $\|\Phi(w^0)\|_\infty$. Additionally the program was configured to terminate after 150 iterations in case no solution was reached. As usual, the known step length related emergency stops were added.

Another enhancement which has been made to Algorithm 8.11 is the implementation of a criterion to adaptively modify the neighborhood $\mathcal{N}(\beta_k)$ by increasing β_k. If the Jacobian smoothing predictor step has not been accepted and $\beta_k \le \gamma_\kappa \kappa$, we multiply β_k by a constant $\gamma_\beta > 1$. For the implementation described here, a choice of $\gamma_\beta = 2$ in combination with $\gamma_\kappa = 1$ produced good results.

The remaining parameters from Step 0 of Algorithm 8.11 were chosen as follows:

$$\varrho = 0.9, \quad \beta := \frac{\|\Phi_{\tau_0}(w^0)\|}{\tau_0}$$

and φ being the Fischer-Burmeister function (according to our experience, the Fischer-Burmeister function gives better results than the minimum function, at least within the framework of Algorithm 8.11).

Finally, the parameter $\hat{\sigma}_k$ from Step 3 of Algorithm 8.11 was always taken to be 0.5.

All test runs were done on a SUN Enterprise 450. Table 8.5 contains the corresponding results, with the columns of table 8.5 having the following meanings:

Problem:	Name of the test problem in the Netlib collection,
m:	Number of equality constraints (after preprocessing),
n:	Number of variables (after preprocessing),
k:	Number of iterations until termination,
P:	Number of accepted predictor steps,
τ_f:	value of τ_k at the final iterate,
$\|\Phi(w^f)\|_\infty$:	value of $\|\Phi(w^k)\|_\infty$ at the final iterate,
Primal Objective:	Value of the primal objective function at final iterate.

Table 8.5: Numerical results of Algorithm 8.11

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective
25fv47	798	1854	34	17	6.0250e−04	2.9537e−04	5.5018459053e+03
80bau3b	2235	11516	29	23	1.0987e−03	7.1465e−04	9.8722419211e+05

Table 8.5: Numerical results of Algorithm 8.11 (continued)

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective
adlittle	55	137	15	15	2.1737e−02	8.7395e−05	2.2549496391e+05
afiro	27	51	10	10	4.7999e−02	3.2452e−05	−4.6474687177e+02
agg	488	615	23	20	1.3406e−02	6.9052e−04	−3.5991767286e+07
agg2	516	758	25	18	7.6969e−03	4.7607e−04	−2.0239252355e+07
agg3	516	758	30	14	7.0617e−03	1.1522e−04	1.0312115936e+07
bandm	269	436	20	19	4.9684e−04	9.3110e−05	−1.5862801756e+02
beaconfd	148	270	18	15	2.7426e−03	5.8563e−04	3.3592485986e+04
blend	74	114	13	12	2.7468e−03	2.9106e−06	−3.0812134385e+01
bnl1	632	1576	26	16	3.5355e−04	7.0895e−05	1.9776295617e+03
bnl2	2268	4430	26	14	9.5617e−04	4.8772e−04	1.8112367543e+03
boeing1	347	722	26	16	2.2843e−03	4.9528e−04	−3.3521310546e+02
boeing2	140	279	16	15	5.4054e−03	9.8945e−04	−3.1500732408e+02
bore3d	199	300	28	22	1.3979e−03	4.1970e−05	1.3730804026e+03
brandy	149	259	19	15	1.1040e−03	7.8205e−05	1.5185099104e+03
capri	267	476	20	19	9.5525e−03	6.4732e−04	2.6900133856e+03
cycle	1801	3305	39	19	9.4566e−05	1.0106e−02	−5.2249915841e+00
czprob	737	3141	22	19	1.1163e−02	2.8069e−04	2.1851966995e+06
d2q06c	2171	5831	57	19	8.3779e−05	4.0045e−05	1.2278421095e+05
d6cube	404	6184	25	21	1.7077e−03	2.6014e−05	3.1549167161e+02
degen2	444	757	23	23	2.3842e−03	9.9759e−05	−1.4351779632e+03
degen3	1503	2604	16	16	7.9692e−04	5.7716e−05	−9.8729398786e+02
dfl001	6071	12230	> 150		*maximum number of iterations exceeded*		
e226	220	469	27	25	2.4792e−04	5.3902e−05	−1.8751928739e+01
etamacro	357	692	26	13	1.5436e−04	7.3792e−05	−7.5571522983e+02
fffff800	501	1005	36	14	6.2879e−03	8.5460e−04	5.5567957590e+05
finnis	492	1014	31	20	1.3882e−03	2.9195e−04	1.7279127031e+05
fit1d	24	1049	20	18	5.7480e−04	4.0534e−05	−9.1463780917e+03
fit1p	627	1677	19	19	1.7472e−03	7.3692e−06	9.1463780936e+03
fit2d	25	10524	22	20	6.0248e−04	8.2675e−05	−6.8464293289e+04
fit2p	3000	13525	20	20	1.2942e−03	3.0570e−04	6.8464293283e+04
forplan	135	463	28	17	4.7384e−03	9.3267e−04	−6.6421820761e+02
ganges	1137	1534	25	20	3.0644e−03	6.4436e−04	−1.0958573612e+05
gfrd-pnc	600	1144	23	16	1.2681e−02	2.4942e−04	6.9022360024e+06
greenbea	2318	5424	25	20	5.7593e−03	8.5643e−04	−7.2462520306e+07
greenbeb	2317	5415	35	15	2.2551e−03	4.7930e−04	−4.3022602607e+06
grow7	140	301	34	19	3.2322e−02	2.2135e−05	−4.7787811813e+07
grow15	300	645	37	18	2.4283e−02	3.7293e−06	−1.0687094129e+08
grow22	440	946	37	15	1.0882e−01	7.5577e−06	−1.6083433646e+08
israel	174	316	27	17	3.9926e−03	2.8436e−04	−8.9664482178e+05
kb2	43	68	32	10	3.3160e−03	9.8866e−05	−1.7499000911e+03
lotfi	151	364	35	16	3.7591e−03	8.0923e−04	−2.5263066012e+01
maros	835	1921	37	12	3.1239e−03	7.1513e−04	−5.8063742927e+04
maros-r7	3136	9408	22	22	4.7684e−03	9.7763e−04	1.4971851671e+06
modszk1	686	1622	26	17	1.1499e−02	8.3608e−04	3.2061981508e+02
nesm	654	2922	52	14	2.9135e−04	2.4794e−04	1.4076036489e+07
perold	625	1530	33	14	3.3115e−03	7.3875e−04	−9.3805322461e+03
pilot	1441	4657	81	14	1.1707e−04	1.4059e−04	−5.5748445014e+02
pilotja	924	2044	76	14	9.6009e−05	4.3618e−03	−6.1130052461e+03
pilotwe	722	2930	61	13	5.8582e−04	3.7472e−04	−2.7201075333e+06
pilot4	402	1173	132	13	9.5377e−05	6.9395e−03	−2.5810606602e+03
pilot87	2030	6460	63	14	8.3070e−05	8.8733e−03	3.0173031374e+02

Table 8.5: Numerical results of Algorithm 8.11 (continued)

Problem	m	n	k	P	τ_f	$\lVert\Phi(w^f)\rVert_\infty$	Primal Objective
pilotnov	951	2242	27	22	2.5409e−03	3.0714e−04	−4.4972761773e+03
recipe	85	177	14	14	1.2207e−03	3.5042e−05	−2.6661598322e+02
sc105	105	163	19	13	1.4451e−03	7.7940e−05	−5.2202033312e+01
sc205	205	317	22	19	7.5991e−04	1.3473e−04	−5.2202035425e+01
sc50a	49	77	15	10	3.6860e−03	4.9576e−05	−6.4575009902e+01
sc50b	48	76	14	11	6.6556e−03	7.6186e−06	−6.9999776566e+01
scagr25	471	671	19	17	2.3406e−02	2.9238e−04	−1.4753433056e+07
scagr7	129	185	19	18	3.4159e−03	3.3656e−04	−2.3313898243e+06
scfxm1	322	592	20	19	5.9750e−03	6.4703e−04	1.8416759818e+04
scfxm2	644	1184	26	18	4.3503e−03	8.7736e−04	3.6660262213e+04
scfxm3	966	1776	26	21	4.9124e−03	9.6075e−04	5.4901255716e+04
scorpion	375	453	21	20	2.7373e−04	1.8825e−05	1.8781248227e+03
scrs8	485	1270	21	19	6.6791e−04	4.5185e−05	9.0429695560e+02
scsd1	77	760	22	22	4.7684e−03	9.5696e−06	8.6666991041e+00
scsd6	147	1350	15	15	4.0199e−04	1.1125e−06	5.0500000067e+01
scsd8	397	2750	13	13	1.1068e−02	4.6656e−05	9.0500023711e+02
sctap1	300	660	24	23	4.0019e−03	5.1964e−05	1.4122500207e+03
sctap2	1090	2500	18	16	1.5629e−03	1.4780e−05	1.7248071430e+03
sctap3	1480	3340	18	17	2.1396e−03	9.2047e−05	1.4240000008e+03
seba	515	1036	23	15	3.1158e−03	9.3931e−05	1.5711600096e+04
share1b	112	248	43	14	3.0326e−03	9.3991e−04	−7.6589318369e+04
share2b	96	162	16	16	3.0518e−04	4.4814e−07	−4.1573224024e+02
shell	496	1487	22	14	1.7418e−01	1.6486e−05	1.2088253461e+09
ship04l	356	2162	20	20	1.6465e−02	8.0608e−04	1.7933245380e+06
ship04s	268	1414	20	20	1.2018e−02	6.5490e−05	1.7987147004e+06
ship08l	688	4339	21	20	1.0490e−02	3.7678e−04	1.9090552114e+06
ship08s	416	2171	20	20	1.8916e−02	1.4499e−04	1.9200982105e+06
ship12l	838	5329	21	20	8.4547e−03	5.6258e−04	1.4701879193e+06
ship12s	466	2293	20	19	8.9471e−03	6.2617e−04	1.4892361344e+06
sierra	1222	2715	22	19	1.9638e−02	9.5043e−04	1.5394362263e+07
stair	356	538	19	18	2.3050e−03	1.6815e−04	−2.5126689656e+02
standata	359	1258	13	12	7.2079e−02	9.5241e−06	1.2577586668e+03
standgub	361	1366	13	12	7.2079e−02	9.5241e−06	1.2577586668e+03
standmps	467	1258	18	14	9.0258e−03	1.1631e−05	1.4060176463e+03
stocfor1	109	157	16	11	3.3401e−02	1.2536e−04	−4.1131976111e+04
stocfor2	2157	3045	29	16	8.3230e−04	2.4732e−05	−3.9024408532e+04
stocfor3	16675	23541	63	17	1.8715e−04	2.8605e−04	−3.9976784284e+04
stocfor3old	16675	23541	70	13	8.7370e−05	6.0456e−04	−3.9976783942e+04
truss	1000	8806	19	18	6.3715e−03	5.6972e−05	4.5881584778e+05
tuff	292	617	32	16	3.2629e−04	1.5624e−04	2.9216987102e−01
vtpbase	194	325	19	19	3.8147e−02	3.2374e−05	1.2983146617e+05
wood1p	244	2595	13	13	3.3617e−04	6.1025e−05	1.4429024524e+00
woodw	1098	8418	34	22	2.2390e−04	9.7122e−05	1.3044869516e+00

The overall results are quite good and seem to be better than the corresponding results from the three-step method given in table 8.1. This method only fails on problem `dfl001`, which, as stated before, cannot be solved by LIPSOL either. Most test problems can be solved in less than 20–30 iterations. Even formerly hard-to-solve problems, such as `cylce` or `d2q06c` can now be solved much quicker in 39 and 57 iterations, as opposed to 124 and 65 iterations, respectively. Unfortunately, to solve the `pilot*`-test cases Algorithm 8.11 still requires more iterations than for most other problems. One still has to take into account that this algorithm, opposed to Algorithm 8.1 is *not* a three-step method, but a regular two-step predictor-corrector algorithm. This means that, Algorithm 8.11 only solves at most two linear systems with different coefficient matrices per iteration, as opposed to a maximum of three matrix factorizations required by Algorithm 8.1.

When comparing iteration numbers, the results presented here are close to the performance of interior-point methods (for a set of results which the solver PCx by Czyzyk, Mehrotra, Wagner, and Wright [16, 17] produced, see table 8.9 on page 146). The comparison of these results, however, has to be done carefully; one has to take into account that Algorithm 8.11 may have to solve up to two linear system with different coefficient matrices per iteration, whereas interior-point methods, which are mostly based on Mehrotra's Algorithm 4.6, only perform one factorization per iteration. Still, one has to remember that, in case the predictor step is not accepted, Algorithm 8.11 will have to factorize only one matrix, which reduces the overall amount of work necessary.

8.3 A More Flexible Update of the Smoothing Variable

In this section an improved version of Algorithm 8.11 is discussed. The new method features a more flexible update rule for the smoothing variable τ in addition to a different generalization of the smoothing operator Θ. It also features no Jacobian smoothing component anymore. The contents of this section are mainly based on the article [21].

8.3.1 Development of Algorithm

As in the previous section, we will only consider the minimum function and its smooth counterpart, the Chen-Harker-Kanzow-Smale smoothing function as the basis for the smoothing method.

We will use the same technique to reformulate the KKT system into a non-linear system of equations as described in Section 7.2.1. This way we are left with a non-linear system $\Theta(x, \lambda, s, \tau) = 0$ which we have to solve.

Consider a function $\psi\colon [0, \infty) \longrightarrow \mathbb{R}$ having the following properties:

(P. 1) ψ is continuously differentiable with $\psi(0) = 0$.

(P. 2) $\psi'(\tau) > 0$ for all $\tau \in [0, \infty)$.

(P. 3) $\psi(\tau) \leq \psi'(\tau) \cdot \tau$ for all $\tau \in [0, \infty)$.

(P. 4) For each $\tau_0 > 0$, there is a constant $\gamma > 0$ (possibly depending on τ_0) such that $\psi(\tau) \geq \gamma \cdot \psi'(\tau) \cdot \tau$ for all $\tau \in [0, \tau_0]$.

Lemma 8.17 *The following functions have properties (P. 1) through (P. 4):*

(1) $\psi(\tau) := \tau$,

(2) $\psi(\tau) := (1 + \tau)^2 - 1$,

(3) $\psi(\tau) := \exp(\tau) - 1$,

Proof.

(1) $\psi(\tau) := \tau$:

(P. 1): This is obvious.

(P. 2): This follows immediately from $\psi'(\tau) \equiv 1 > 0$ for all $\tau \in [0, \infty)$.

(P. 3): $\psi(\tau) \leq \psi'(\tau)\tau \Leftrightarrow \tau \leq 1 \cdot \tau$. Therefore the function $\psi(\tau) := \tau$ obviously has the Property (P. 3).

(P. 4): $\psi(\tau) \geq \gamma \cdot \psi'(\tau) \cdot \tau \Leftrightarrow \tau \geq \gamma \cdot \tau$ immediately yields $\gamma = 1$ for all τ. Note that the value of γ is independent of τ_0 in this case.

(2) $\psi(\tau) := (1 + \tau)^2 - 1$:

(P. 1): This is obvious.

(P. 2): $\psi'(\tau) = 2 + 2\tau \geq 2 > 0$ for all $\tau \in [0, \infty)$.

(P. 3): For all $\tau \in [0, \infty)$ it holds that

$$\psi(\tau) = (1+\tau)^2 - 1 = 2\tau + \tau^2 = (2+\tau)\tau \leq (2+2\tau)\tau = \psi'(\tau)\tau.$$

Therefore this function obviously has the Property (P. 3).

(P. 4): For all $\tau \in [0, \infty)$ it holds

$$\begin{aligned}\psi(\tau) &= 2\tau + \tau^2 = (2+\tau)\tau = \tau + (1+\tau)\tau = \tau + \frac{1}{2}(2+2\tau)\tau \\ &\geq \frac{1}{2}(2+2\tau)\tau = \frac{1}{2}\psi'(\tau)\tau.\end{aligned}$$

Hence with $\gamma := 1/2$ this function has the Property (P. 4) where γ is independent of τ_0.

(3) $\psi(\tau) := \exp(\tau) - 1$:

(P. 1): This is obvious since $\exp(0) = 1$.

(P. 2): Since $\psi'(\tau) = \exp(\tau)$ it is obvious that $\psi'(\tau) \geq 1 > 0$ for all $\tau \in [0, \infty)$.

(P. 3): It holds for all $\tau \in [0, \infty)$:

$$\begin{array}{rrcl} & \psi(\tau) & \leq & \psi'(\tau)\cdot\tau \\ \Leftrightarrow & \exp(\tau) - 1 & \leq & \exp(\tau)\cdot\tau \\ \Leftrightarrow & \exp(\tau) & \leq & \exp(\tau)\cdot\tau + 1 \\ \Leftrightarrow & \exp(\tau) & \leq & \exp(\tau)\cdot\tau + \exp(\tau)\exp(-\tau) \\ \Leftrightarrow & \exp(\tau) & \leq & \exp(\tau)(\tau + \exp(-\tau)) \\ \Leftrightarrow & 1 & \leq & \tau + \exp(-\tau) \\ \Leftrightarrow & 1 - \tau & \leq & \exp(-\tau). \end{array}$$

Therefore $\psi(\tau) := \exp(\tau) - 1$ has Property (P. 3).

(P. 4): It holds for all $\tau \in [0, \tau_0]$ and $\tau_0 > 0$:

$$\begin{array}{rrcl} & \psi(\tau) & \geq & \gamma\psi'(\tau)\tau \\ \Leftrightarrow & \exp(\tau) - 1 & \geq & \gamma\exp(\tau)\tau \\ \Leftrightarrow & 1 - \exp(-\tau) & \geq & \gamma\tau \\ \Leftrightarrow & \exp(-\tau) - 1 & \leq & -\gamma\tau \\ \Leftrightarrow & \exp(-\tau) & \leq & 1 - \gamma\tau. \end{array}$$

The monotonously decreasing function $1 - \gamma\tau$ is equal to $\exp(-\tau)$ for $\tau = 0$ and for some $\tau_0 > 0$. Considering this last equality

$$\exp(-\tau_0) = 1 - \gamma\tau_0$$

now gives us the value of $\gamma(\tau_0)$ for which the inequality from (P. 4) is satisfied. Hence $\gamma := \gamma(\tau_0) := [1 - \exp(-\tau_0)]/\tau_0$. This shows that $\psi(\tau) := \exp(\tau) - 1$ has Property (P. 4). ■

We will use the following generalization of the equation operator Θ:

$$\Theta_{\sigma,\psi}(x,\lambda,s,\tau) := \begin{bmatrix} A^{\mathrm{T}}\lambda + s - c \\ Ax - b \\ \theta(x,s,\tau) \\ \sigma\psi(\tau) \end{bmatrix}, \tag{8.30}$$

and its Jacobian

$$\Theta'_{\sigma,\psi}(x,\lambda,s,\tau) = \begin{bmatrix} 0 & A^{\mathrm{T}} & I & 0 \\ A & 0 & 0 & 0 \\ D^k_{a,\tau} & 0 & D^k_{b,\tau} & d^k_\tau \\ 0 & 0 & 0 & \sigma\psi'(\tau) \end{bmatrix}. \tag{8.31}$$

Here $\sigma \in (0,1]$ denotes a suitable centering parameter, and, as usual,

$$D^k_{a,\tau} := \operatorname{diag}\left(\dots, \frac{\partial\varphi_{\tau_k}}{\partial a}(x^k_i, s^k_i), \dots\right) \in \mathbb{R}^{n\times n},$$

$$D^k_{b,\tau} := \operatorname{diag}\left(\dots, \frac{\partial\varphi_{\tau_k}}{\partial b}(x^k_i, s^k_i), \dots\right) \in \mathbb{R}^{n\times n},$$

$$d^k_\tau := \left(\dots, \frac{\partial\theta_i}{\partial\tau}(x^k_i, s^k_i, \tau_k), \dots\right)^{\mathrm{T}} \in \mathbb{R}^n,$$

where θ_i is the i-th component function of θ.

Note that the choice $\psi(\tau) = \tau$ corresponds to the one used in [5, 8], whereas here we aim to generalize the approach from [5, 8] in order to allow a more flexible procedure to decrease τ. Since the precise reduction of τ has a significant influence on the overall performance of the smoothing-type method presented in the previous section, such a generalization is very important from a computational point of view.

Before we give a precise statement of our algorithm, let us add some further comments on the properties of the function ψ: (P. 1) is obviously needed since we want to apply a Newton-type method to the system of equations $\Theta_{\sigma,\psi}(x,\lambda,s,\tau) = 0$, hence ψ has to be sufficiently smooth. The second property (P. 2) implies that ψ is strictly monotonically increasing. Together with $\psi(0) = 0$ from Property (P. 1), this means that the non-linear system of equations

$$\Theta_{\sigma,\psi}(x,\lambda,s,\tau) = 0$$

is equivalent to the optimality conditions (3.5) themselves (and not to the central path conditions (4.2)), since the last row immediately gives $\tau = 0$. The third property (P. 3) will be used in order to show that the algorithm to be presented below is well-defined, cf. the proof of Lemma 8.19 (3). Furthermore, Properties (P. 3) and (P. 4) together will guarantee that the sequence $\{\tau_k\}$ is monotonically decreasing and converges to zero, see the proof of Theorem 8.23.

We now return to the description of the algorithm. The method to be presented below is a predictor-corrector algorithm with the predictor step being responsible for the local fast rate of convergence, and with the corrector step guaranteeing global convergence. More precisely, the predictor step consists of one Newton iteration applied to the system $\Theta(x,\lambda,s,\tau) = 0$, followed by a suitable update of τ which tries to reduce τ as much as possible. The corrector step then applies one Newton iteration to the system $\Theta_{1,\psi}(x,\lambda,s,\tau) = 0$, but with the usual right-hand side $\Theta_{1,\psi}(x,\lambda,s,\tau)$ being replaced by $\Theta_{\sigma,\psi}(x,\lambda,s,\tau)$ for some centering parameter $\sigma \in (0,1)$. This Newton step is followed by an Armijo-type line search.

The computation of all iterates is carried out in such a way that they belong to the neighborhood

$$\mathcal{N}(\beta) := \{(x,\lambda,s,\tau) \mid A^{\mathrm{T}}\lambda + s = c,\ \ Ax = b,\ \ \|\theta(x,s,\tau)\| \le \beta\tau\}$$

of the central path, where $\beta > 0$ denotes a suitable constant. In addition, we will see later that all iterates automatically satisfy the inequality $\theta(x,s,\tau) \le 0$, which will be important in order to establish a result regarding the boundedness of the iterates, cf. Lemma 8.21 and Proposition 8.22 below.

The precise statement of our algorithm is as follows (recall that Θ and $\Theta_{\sigma,\psi}$ denote the mappings from (7.17) and (8.30), respectively).

Algorithm 8.18 (Generalized Predictor-Corrector Smoothing Method)
Step 0 (Initialization):

Choose $w^0 := (x^0,\lambda^0,s^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ and $\tau_0 > 0$ such that $A^{\mathrm{T}}\lambda^0 + s^0 = c, Ax^0 = b$, and $\theta(x^0,s^0,\tau_0) \le 0$, select $\beta \ge \|\theta(x^0,s^0,\tau_0)\|/\tau_0, \varrho \in (0,1), 0 < \hat{\sigma}_{\min} < \hat{\sigma}_{\max} < 1, \varepsilon \ge 0$, and set the iteration counter $k := 0$.

Step 1 (Termination Criterion):

If $\|\theta(x^k,s^k,0)\| \le \varepsilon$: Stop.

Step 2 (Predictor Step):

Compute a solution $(\Delta w^k,\Delta\tau_k) = (\Delta x^k,\Delta\lambda^k,\Delta s^k,\Delta\tau_k) \in \mathbb{R}^n\times\mathbb{R}^m\times\mathbb{R}^n\times\mathbb{R}$ of the linear system

$$\Theta'(w^k,\tau_k)\begin{bmatrix}\Delta w\\ \Delta\tau\end{bmatrix} = -\Theta(w^k,\tau_k). \tag{8.32}$$

If $\|\theta(x^k+\Delta x^k,s^k+\Delta s^k,0)\| \le \varepsilon$: Stop. Otherwise, if

$$\|\theta(x^k+\Delta x^k,s^k+\Delta s^k,\tau_k)\| > \beta\tau_k,$$

then set

$$\hat{w}^k := w^k, \quad \hat{\tau}_k := \tau_k, \quad \eta_k := 1,$$

else compute $\eta_k = \varrho^{\ell_k}$, where ℓ_k is the non-negative integer such that

$$\begin{aligned}\|\theta(x^k+\Delta x^k,s^k+\Delta s^k,\varrho^j\tau_k)\| &\le \beta\varrho^j\tau_k \quad \forall j = 0,1,2,\dots,\ell_k \text{ and}\\ \|\theta(x^k+\Delta x^k,s^k+\Delta s^k,\varrho^{\ell_k+1}\tau_k)\| &> \beta\varrho^{\ell_k+1}\tau_k,\end{aligned} \tag{8.33}$$

and set

$$\hat{\tau}_k := \eta_k \tau_k,$$
$$\hat{w}^k := \begin{cases} w^k & \text{if } \ell_k = 0, \\ w^k + \Delta w^k & \text{otherwise.} \end{cases} \tag{8.34}$$

Step 3 (Corrector Step):

Choose $\hat{\sigma}_k \in [\hat{\sigma}_{\min}, \hat{\sigma}_{\max}]$.

Compute a solution $(\Delta\hat{w}^k, \Delta\hat{\tau}_k) = (\Delta\hat{x}^k, \Delta\hat{\lambda}^k, \Delta\hat{s}^k, \Delta\hat{\tau}_k) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n \times \mathbb{R}$ of the linear system

$$\Theta'_{1,\psi}(\hat{w}^k, \hat{\tau}_k) \begin{bmatrix} \Delta\hat{w} \\ \Delta\hat{\tau} \end{bmatrix} = -\Theta_{\hat{\sigma}_k,\psi}(\hat{w}^k, \hat{\tau}_k). \tag{8.35}$$

Let $\hat{t}_k = \max\left\{ \varrho^\ell \;\middle|\; \ell = 0, 1, 2, \dots \right\}$ such that

$$\|\theta(\hat{x}^k + \hat{t}_k\Delta\hat{x}^k, \hat{s}^k + \hat{t}_k\Delta\hat{s}^k, \hat{\tau}_k + \hat{t}_k\Delta\hat{\tau}_k)\| \le \beta(\hat{\tau}_k + \hat{t}_k\Delta\hat{\tau}_k) \tag{8.36}$$

and set

$$w^{k+1} := \hat{w}^k + \hat{t}_k\Delta\hat{w}^k \quad \text{and} \quad \tau_{k+1} := \hat{\tau}_k + \hat{t}_k\Delta\hat{\tau}_k. \tag{8.37}$$

Step 4 (Update):

Set $k \longleftarrow k + 1$ and go to Step 1.

Algorithm 8.18 is closely related to Algorithm 8.11. In fact, when choosing $\psi(\tau) = \tau$ the only difference which remains is that a different right-hand side is used in the predictor step, namely $\Theta(w^k, \tau_k)$ instead of $\Theta(w^k, 0)$. The latter choice seems to give slightly better local properties, however, the current version allows to prove better global convergence results. Note that by doing so, we abandon the Jacobian smoothing idea, as in Algorithm 8.18 both sides of all linear system are perturbed.

From now on, it is always assumed that the termination parameter ε in Algorithm 8.18 is equal to zero and that Algorithm 8.18 generates an infinite sequence $\{(x^k, \lambda^k, s^k, \tau_k)\}$, i. e., we assume that the stopping criteria in Steps 1 and 2 are never satisfied. This is not restrictive at all, since otherwise w^k or $w^k + \Delta w^k$ would be a solution of the optimality conditions (3.5).

We first note that Algorithm 8.18 is well-defined.

Lemma 8.19 *The following statements hold for any $k \in \mathbb{N}$:*

(1) The linear systems (8.32) *and* (8.35) *have a unique solution.*

(2) There exists a unique η_k satisfying the conditions in Step 2.

(3) The step size $\hat{t}_k$ in Step 3 is uniquely defined.

Consequently, Algorithm 8.18 is well-defined.

Proof.

(1) Taking into account the structure of the Jacobians $\Theta'(w,\tau)$ and $\Theta'_{\sigma,\psi}(w,\tau)$ and using the fact that $\psi'(\tau) > 0$ by Property (P. 2), part (1) is an immediate consequence of Proposition 5.6.

(2) The statement follows with the same arguments as Proposition 8.12 (2) and Proposition 8.2.

(3) In order to verify the third statement, assume there is an iteration index k such that

$$\|\theta(\hat{x}^k + \varrho^\ell \Delta\hat{x}^k, \hat{s}^k + \varrho^\ell \Delta\hat{s}^k, \hat{\tau}_k + \varrho^\ell \Delta\hat{\tau}_k)\| > \beta(\hat{\tau}_k + \varrho^\ell \Delta\hat{\tau}_k)$$

for all $\ell \in \mathbb{N}$. Since $\|\theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k)\| \le \beta\hat{\tau}_k$, we obtain from Property (P. 3) that

$$\begin{aligned}\beta(\hat{\tau}_k + \varrho^\ell \Delta\hat{\tau}_k) &= \beta(\hat{\tau}_k - \varrho^\ell \hat{\sigma}_k \psi(\hat{\tau}_k)/\psi'(\hat{\tau}_k)) \\ &\ge \beta(\hat{\tau}_k - \varrho^\ell \hat{\sigma}_k \hat{\tau}_k) \\ &\ge (1 - \hat{\sigma}_{\max}\varrho^\ell)\beta\hat{\tau}_k \\ &\ge (1 - \hat{\sigma}_{\max}\varrho^\ell)\|\theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k)\|.\end{aligned}$$

Taking this inequality into account, the proof can now be completed by using a standard argument for the Armijo line search rule in the same manner as in the proof of Lemma 7.9 (2).

We next state some simple properties of Algorithm 8.18 to which we will refer a number of times in our subsequent analysis.

Lemma 8.20 *The two sequences $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ and $\{\tau_k\}$ generated by Algorithm 8.18 have the following properties:*

(1) $A^{\mathrm{T}}\lambda^k + s^k = c$ and $Ax^k = b$ for all $k \in \mathbb{N}$.

(2) $\tau_k \le \tau_0(1 - \gamma\hat{\sigma}_0\hat{t}_0)\eta_0 \cdot \ldots \cdot (1 - \gamma\hat{\sigma}_{k-1}\hat{t}_{k-1})\eta_{k-1}$ for all $k \in \mathbb{N}$, where $\gamma > 0$ denotes the constant from property (P. 4).

(3) $\|\theta(x^k, s^k, \tau_k)\| \le \beta\tau_k$ for all $k \in \mathbb{N}$.

Proof. Part (1) holds for $k = 0$ by our choice of the starting point (x^0, λ^0, s^0). Hence it holds for all $k \in \mathbb{N}$ since Newton's method solves linear systems exactly.

In order to verify statement (2), we first note that we get

$$\Delta\hat{\tau}_k = -\hat{\sigma}_k \psi(\hat{\tau}_k)/\psi'(\hat{\tau}_k) \tag{8.38}$$

from the fourth block row of the linear equation (8.35). Since τ_k, $\hat{\tau}_k \in [0, \tau_0]$ for all $k \in \mathbb{N}$, it therefore follows from Property (P. 4) and the updating rules in Steps 2 and 3 of Algorithm 8.18 that

$$\begin{aligned}
\tau_{k+1} &= \hat{\tau}_k + \hat{t}_k \Delta \hat{\tau}_k \\
&= \hat{\tau}_k - \hat{t}_k \hat{\sigma}_k \psi(\hat{\tau}_k) / \psi'(\hat{\tau}_k) \\
&\leq \hat{\tau}_k - \gamma \hat{t}_k \hat{\sigma}_k \hat{\tau}_k \\
&= (1 - \gamma \hat{t}_k \hat{\sigma}_k) \eta_k \tau_k.
\end{aligned}$$

Using a simple induction argument, we see that (2) holds.

Finally, statement (3) is a direct consequence of the updating rules in Algorithm 8.18. ■

8.3.2 Convergence Analysis

In this section, we analyze the global and local convergence properties of Algorithm 8.18. Even though the analysis for the local rate of convergence is essentially the same as in [8] (recall that our predictor step is identical to the one from [8]), the relevant results are included here for the sake of completeness. In particular, we will show that all iterates (x^k, λ^k, s^k) remain bounded under a strict feasibility assumption. This was noted by Burke and Xu [8] for a particular member of our class of methods (namely for the choice $\psi(\tau) := \tau$), but is not true for many other smoothing-type methods like those from [11, 12, 13, 14, 20, 22, 55, 54].

The central observation which allows us to prove the boundedness of the iterates (x^k, λ^k, s^k) is that they automatically satisfy the inequality

$$\theta(x^k, s^k, \tau_k) \leq 0$$

for all $k \in \mathbb{N}$ provided this inequality holds for $k = 0$. This is precisely the statement of our first result.

Lemma 8.21 *The sequences* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$, $\{\hat{w}^k\} = \{(\hat{x}^k, \hat{\lambda}^k, \hat{s}^k)\}$ *and* $\{\tau_k\}$, $\{\hat{\tau}_k\}$ *generated by Algorithm 8.18 have the following properties:*

(1) $\theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k) \leq 0$ *for all* $k \in \mathbb{N}$.

(2) $\theta(x^k, s^k, \tau_k) \leq 0$ *for all* $k \in \mathbb{N}$.

Proof. We first derive some useful inequalities, and then verify the two statements simultaneously by induction on k.

We begin with some preliminary discussions regarding statement (1). To this end, let $k \in \mathbb{N}$ be fixed for the moment, and assume that we take $\hat{w}^k = w^k + \Delta w^k$ in Step 2 of Algorithm 8.18. Since each component of the function θ is concave, we then obtain

$$\begin{aligned}
&\theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k) \\
&= \theta(x^k + \Delta x^k, s^k + \Delta s^k, \eta_k \tau_k) \\
&= \theta(x^k + \Delta x^k, s^k + \Delta s^k, \tau_k + (\eta_k - 1)\tau_k) \\
&\le \theta(x^k, s^k, \tau_k) + \theta'(x^k, s^k, \tau_k) \begin{bmatrix} \Delta x^k \\ \Delta s^k \\ (\eta_k - 1)\tau_k \end{bmatrix} \\
&= \theta(x^k, s^k, \tau_k) + \theta'(x^k, s^k, \tau_k) \begin{bmatrix} \Delta x^k \\ \Delta s^k \\ \Delta \tau_k \end{bmatrix} + \theta'(x^k, s^k, \tau_k) \begin{bmatrix} 0 \\ 0 \\ (\eta_k - 1)\tau_k - \Delta\tau_k \end{bmatrix}.
\end{aligned} \tag{8.39}$$

From the third block row of (8.32), we have

$$\theta'(x^k, s^k, \tau_k) \begin{bmatrix} \Delta x^k \\ \Delta s^k \\ \Delta \tau_k \end{bmatrix} = -\theta(x^k, s^k, \tau_k).$$

Hence we get from (8.39):

$$\theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k) \le \theta'(x^k, s^k, \tau_k) \begin{bmatrix} 0 \\ 0 \\ (\eta_k - 1)\tau_k - \Delta\tau_k \end{bmatrix}. \tag{8.40}$$

We claim that the right-hand side of (8.40) is non-positive. To prove this statement, we first note that

$$\theta'(x^k, s^k, \tau_k) \begin{bmatrix} 0 \\ 0 \\ (\eta_k - 1)\tau_k - \Delta\tau_k \end{bmatrix} = ((\eta_k - 1)\tau_k - \Delta\tau_k) d_\tau^k$$

with

$$d_\tau^k := \left(\dots, \frac{\partial \theta_i}{\partial \tau}(x_i^k, s_i^k, \tau_k), \dots, \right)^{\mathrm{T}} = \left(\dots, \frac{-4\tau_k}{\sqrt{(x_i^k - s_i^k)^2 + 4\tau_k^2}}, \dots \right)^{\mathrm{T}} \le 0,$$

where θ_i denotes the i-th component function of θ. Hence it remains to show that

$$(\eta_k - 1)\tau_k - \Delta\tau_k \ge 0.$$

However, this is obvious since the last row of the linear system (8.32) immediately implies $\Delta\tau_k = -\tau_k$.

We next derive some useful inequalities regarding statement (2). To this end, we still assume that $k \in \mathbb{N}$ is fixed. Using once again the fact that θ is a concave function in each component, we obtain from (8.35)

$$\begin{aligned} \theta(x^{k+1}, s^{k+1}, \tau_{k+1}) &= \theta(\hat{x}^k + \hat{t}_k \Delta\hat{x}^k, \hat{s}^k + \hat{t}_k \Delta\hat{s}^k, \hat{\tau}_k + \hat{t}_k \Delta\hat{\tau}_k) \\ &\le \theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k) + \hat{t}_k \theta'(\hat{x}^k, \hat{s}^k, \hat{\tau}_k) \begin{bmatrix} \Delta\hat{x}^k \\ \Delta\hat{s}^k \\ \Delta\hat{\tau}_k \end{bmatrix} \\ &= \theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k) - \hat{t}_k \theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k) \\ &= (1 - \hat{t}_k)\theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k), \end{aligned} \tag{8.41}$$

and this completes our preliminary discussions.

We now verify statements (1) and (2) by induction on k. For $k = 0$, we have $\theta(x^0, s^0, \tau_0) \le 0$ by our choice of the starting point $w^0 = (x^0, \lambda^0, s^0)$ and the initial smoothing parameter $\tau_0 > 0$ in Step 0 of Algorithm 8.18. Therefore, if we set $\hat{w}^0 = w^0$ in Step 2 of Algorithm 8.18, we also have $\hat{\tau}_0 = \tau_0$ and therefore $\theta(\hat{x}^0, \hat{s}^0, \hat{\tau}_0) \le 0$. On the other hand, if we set $\hat{w}^0 = w^0 + \Delta w^0$ in Step 2, the argument used in the beginning of this proof shows that the inequality $\theta(\hat{x}^0, \hat{s}^0, \hat{\tau}_0) \le 0$ also holds in this case.

Suppose that we have $\theta(x^k, s^k, \tau_k) \le 0$ and $\theta(\hat{x}^k, \hat{s}^k, \hat{\tau}_k) \le 0$ for some $k \in \mathbb{N}$. Then (8.41) immediately implies that we have $\theta(x^{k+1}, s^{k+1}, \tau_{k+1}) \le 0$. Consequently, if we have $\hat{w}^{k+1} = w^{k+1}$ in Step 2 of Algorithm 8.18, we obviously have $\theta(\hat{x}^{k+1}, \hat{s}^{k+1}, \hat{\tau}_{k+1}) \le 0$. Otherwise, i. e., if we set $\hat{w}^{k+1} = w^{k+1} + \Delta w^{k+1}$ in Step 2, the argument used in the beginning part of this proof shows that the same inequality holds. This completes the formal proof by induction. ■

Proposition 8.22 *Assume that there is a strictly feasible point for the optimality conditions* (3.5). *Then the sequence* $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ *generated by Algorithm 8.18 is bounded.*

Proof. The statement is essentially due to Burke and Xu [8], therefore the proof is included here only for the sake of completeness.

Assume that the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ generated by Algorithm 8.18 is unbounded. Since $\{\tau_k\}$ is monotonically decreasing by Lemma 8.20 (2), it follows from Lemma 8.20 (3) that

$$\|\theta(x^k, s^k, \tau_k)\| \le \beta\tau_k \le \beta\tau_0 \tag{8.42}$$

for all $k \in \mathbb{N}$. The definition of the (smoothed) minimum function therefore implies that there is no index $i \in \{1, \dots, n\}$ such that $x_i^k \longrightarrow -\infty$ or $s_i^k \longrightarrow -\infty$ on a subsequence, since otherwise we would have $\theta_i(x_i^k, s_i^k, \tau_k) \longrightarrow -\infty$ which, in turn, would imply $\|\theta(x^k, s^k, \tau_k)\| \longrightarrow +\infty$ on a subsequence in contrast to (8.42). Therefore, all components of the two sequences $\{x^k\}$ and $\{s^k\}$ are bounded from below, i. e.,

$$x_i^k \ge \gamma \quad \text{and} \quad s_i^k \ge \gamma \quad \text{for all } i = 1, \dots, n \text{ and for all } k \in \mathbb{N}, \tag{8.43}$$

where $\gamma \in \mathbb{R}$ denotes a suitable (possibly negative) constant.

On the other hand, the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ is unbounded by assumption. This implies that there is at least one component $i \in \{1, \dots, n\}$ such that $x_i^k \longrightarrow +\infty$ or $s_i^k \longrightarrow +\infty$ on a subsequence since otherwise the two sequences $\{x^k\}$ and $\{s^k\}$ would be bounded which, in turn, would imply the boundedness of the sequence $\{\lambda^k\}$ as well because we have $A^{\mathrm{T}}\lambda^k + s^k = c$ for all $k \in \mathbb{N}$ (cf. Lemma 8.20 (1)) and because A is assumed to have full rank.

Now let $\hat{w} = (\hat{x}, \hat{\lambda}, \hat{s}) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^n$ be a strictly feasible point for (3.5) whose existence is guaranteed by our assumption. Then, in particular, we have

$$A^{\mathrm{T}}\hat{\lambda} + \hat{s} = c \quad \text{and} \quad A\hat{x} = b.$$

Since we also have

$$A^{\mathrm{T}}\lambda^k + s^k = c \quad \text{and} \quad Ax^k = b$$

for all $k \in \mathbb{N}$ by Lemma 8.20 (1), we get

$$A^{\mathrm{T}}(\hat{\lambda} - \lambda^k) + (\hat{s} - s^k) = 0 \quad \text{and} \quad A(\hat{x} - x^k) = 0 \tag{8.44}$$

by subtracting these equations. Premultiplying the first equation in (8.44) with $(\hat{x} - x^k)^{\mathrm{T}}$ and taking into account the second equation in (8.44) gives

$$(\hat{x} - x^k)^{\mathrm{T}}(\hat{s} - s^k) = 0.$$

Reordering this equation, we obtain

$$\hat{s}^{\mathrm{T}}x^k + \hat{x}^{\mathrm{T}}s^k = (x^k)^{\mathrm{T}}s^k + \hat{x}^{\mathrm{T}}\hat{s} \tag{8.45}$$

for all $k \in \mathbb{N}$. Using (8.43) as well as $\hat{x} > 0$ and $\hat{s} > 0$ in view of the strict feasibility of the vector $\hat{w} = (\hat{x}, \hat{\lambda}, \hat{s})$, it follows from (8.45) and the fact that $x_i^k \longrightarrow +\infty$ or $s_i^k \longrightarrow +\infty$ on a subsequence for at least one index $i \in \{1, \dots, n\}$ that

$$(x^k)^{\mathrm{T}}s^k \longrightarrow +\infty.$$

Hence there exists a component $j \in \{1, \dots, n\}$ (independent of k) such that

$$x_j^k s_j^k \longrightarrow +\infty \tag{8.46}$$

on a suitable subsequence.

Now, using Lemma 8.21 (2), we have

$$\theta(x^k, s^k, \tau_k) \leq 0$$

for all $k \in \mathbb{N}$. Taking into account the definition of θ and looking at the j-th component, this implies

$$x_j^k + s_j^k \leq \sqrt{(x_j^k - s_j^k)^2 + 4\tau_k^2} \tag{8.47}$$

for all $k \in \mathbb{N}$. Using (8.46) and (8.43), we see that we necessarily have $x_j^k > 0$ and $s_j^k > 0$ for all those k belonging to the subsequence for which (8.46) holds. Therefore, taking the square in (8.47), we obtain

$$4x_j^k s_j^k \leq 4\tau_k^2$$

after some simplifications. However, since the right-hand side of this expression is bounded by $4\tau_0^2$, this gives a contradiction to (8.46). ∎

We next prove a global convergence result for Algorithm 8.18. Note that this result is different from the one provided by Burke and Xu [8] and is more in the spirit of those from [54, 20, 22]. (Burke and Xu [8] use a stronger assumption in order to prove a global linear rate of convergence for the sequence $\{\tau_k\}$.)

Theorem 8.23 *Assume that the sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ generated by Algorithm 8.18 has at least one accumulation point. Then $\{\tau_k\}$ converges to zero.*

Proof. Since the sequence $\{\tau_k\}$ is monotonically decreasing (by Lemma 8.20 (2)) and bounded from below by zero, it converges to a number $\tau_* \geq 0$. If $\tau_* = 0$, we are done.

So assume that $\tau_* > 0$. Then the updating rules in Step 2 of Algorithm 8.18 immediately give

$$\hat{w}^k = w^k, \quad \hat{\tau}_k = \tau_k, \quad \text{and} \quad \eta_k = 1 \tag{8.48}$$

for all $k \in \mathbb{N}$ sufficiently large. Subsequencing if necessary, we assume without loss of generality that (8.48) holds for all $k \in \mathbb{N}$. Then Lemma 8.20 (2) and $\hat{\sigma}_k \geq \hat{\sigma}_{\min}$ yield

$$\tau_k \leq \tau_0 \prod_{j=0}^{k-1} (1 - \gamma\hat{\sigma}_j \hat{t}_j) \leq \tau_0 \prod_{j=0}^{k-1} (1 - \gamma\hat{\sigma}_{\min}\hat{t}_j). \tag{8.49}$$

The remainder of this proof is analogous to the proof of Theorem 7.13 ∎

Due to Proposition 8.22, the assumed existence of an accumulation point in Theorem 8.23 is automatically satisfied if there is a strictly feasible point for the optimality conditions (3.5). An immediate consequence of Theorem 8.23 is the following result, which can be proven in the same way as Corollary 7.14.

Corollary 8.24 *Every accumulation point of a sequence $\{w^k\} = \{(x^k, \lambda^k, s^k)\}$ generated by Algorithm 8.18 is a solution of the optimality conditions* (3.5).

The following result will give us some insight on a property of the Chen-Harker-Kanzow-Smale smoothing function which will be used in the proof of the main convergence result. This result was originally proven by Qi and Sun [50].

Lemma 8.25 *Let $\theta(a,b,\tau)\colon \mathbb{R}\times\mathbb{R}\times[0,\infty) \longrightarrow \mathbb{R}$ denote the Chen-Harker-Kanzow-Smale smoothing function. Then we have for any $(a,b,\tau)\in\mathbb{R}\times\mathbb{R}\times[0,\infty)$:*

$$\|\theta''(a,b,\tau)\| \le \frac{4}{\sqrt{(a-b)^2+4\tau^2}} \le \frac{2}{\tau}$$

Proof. Simple calculations show that

$$\theta'(a,b,\tau) = \begin{bmatrix} 1-\dfrac{a-b}{\sqrt{(a-b)^2+4\tau^2}} \\ 1+\dfrac{a-b}{\sqrt{(a-b)^2+4\tau^2}} \\ \dfrac{-4\tau}{\sqrt{(a-b)^2+4\tau^2}} \end{bmatrix}$$

and

$$\theta''(a,b,\tau) = \frac{4}{\left(\sqrt{(a-b)^2+4\tau^2}\right)^3}\begin{bmatrix} -\tau^2 & \tau^2 & (a-b)\tau \\ \tau^2 & -\tau^2 & -(a-b)\tau \\ (a-b)\tau & -(a-b)\tau & -(a-b)^2 \end{bmatrix}$$

Therefore,

$$\begin{aligned}
\|\theta''(a,b,\tau)\| &\le \|\theta''(a,b,\tau)\|_F \\
&\le \frac{4}{\left(\sqrt{(a-b)^2+4\tau^2}\right)^3}\sqrt{(a-b)^4+4(a-b)^2\tau^2+4\tau^4} \\
&= \frac{4}{\left(\sqrt{(a-b)^2+4\tau^2}\right)^3}\left((a-b)^2+2\tau^2\right) \\
&\le \frac{4}{\sqrt{(a-b)^2+4\tau^2}} \\
&\le \frac{2}{\tau}
\end{aligned}$$

■

We finally state our local rate of convergence result. Since our predictor step coincides with the one by Burke and Xu [8], the proof of this result is essentially the same as in [8], and is presented here for the sake of completeness.

The central observation in order to prove Theorem 8.26 is that the sequence of Jacobian matrices $\Theta'(w^k,\tau_k)$ converges to a non-singular matrix under the assumption of Theorem 8.26. In fact, as noted in Theorem 6.6 (and [8] and [23], for that matter), the convergence of this sequence to a non-singular Jacobian matrix is equivalent to the unique solvability of the optimality conditions (3.5).

Theorem 8.26 *Let the parameter β satisfy the inequality $\beta > 2\sqrt{n}$, assume that the optimality conditions* (3.5) *have a unique solution $w^* = (x^*,\lambda^*,s^*)$, and suppose that the sequence $\{w^k\} = \{(x^k,\lambda^k,s^k)\}$ generated by Algorithm 8.18 converges to w^*. Then $\{\tau_k\}$ converges locally quadratically to zero.*

Proof. Assume that Algorithm 8.18 does not terminate in a solution of the optimality conditions (3.5).

First consider the system (8.32). Applying the same technique as used to rewrite system (7.22), and taking into account that $\Delta\tau_k = -\tau_k$, this equation is reduced to

$$\Phi'_{\tau_k}(w^k)\Delta w = -\Phi_{\tau_k}(w^k) + \tau_k \begin{bmatrix} 0 \\ 0 \\ d_\tau^k \end{bmatrix},$$

Together with the choice of the starting point in Step 0 this yields

$$\begin{aligned} A^{\mathrm{T}}\Delta\lambda + \Delta s &= 0 \\ A\Delta x &= 0 \\ D^k_{a,\tau_k}\Delta x + D^k_{b,\tau_k}\Delta s &= -\theta(x^k, s^k, \tau_k) + \tau_k d^k_\tau. \end{aligned}$$

Using this reformulation, it is possible to attain a bound for $\|(\Delta x, \Delta s)\|$. From

$$\begin{bmatrix} 0 & A^{\mathrm{T}} & -I \\ A & 0 & 0 \\ D^k_{a,\tau} & 0 & D^k_{b,\tau} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\theta(x_k, s_k, \tau_k) + \tau_k d^k_\tau \end{bmatrix}$$

and

$$|(d^k_\tau)_i| = \left|\frac{\partial\theta_i}{\partial\tau}(x^k_i, s^k_i, \tau_k)\right| = \left|-\frac{4\tau_k}{\sqrt{(a-b)^2 + 4\tau^2}}\right| = \left|-\frac{4\tau}{\sqrt{4\tau^2}}\right| \le 2$$

as well as Lemma 8.13 (4) it follows that

$$\begin{aligned} \left\|\begin{bmatrix} \Delta x \\ \Delta s \end{bmatrix}\right\| \le \left\|\begin{bmatrix} \Delta x \\ \Delta\lambda \\ \Delta s \end{bmatrix}\right\| &\le \left\|\begin{bmatrix} 0 & A^{\mathrm{T}} & -I \\ A & 0 & 0 \\ D^k_{a,\tau} & 0 & D^k_{b,\tau} \end{bmatrix}\right\|^{-1} \cdot \left\|\begin{bmatrix} 0 \\ 0 \\ -\theta(x_k, s_k, \tau_k) + \tau_k d^k_\tau \end{bmatrix}\right\| \\ &\le C\left(\|\theta(x_k, s_k, \tau_k)\| + \tau_k\|d^k_\tau\|\right) \\ &\le C(\beta\tau_k + \tau_k 2\sqrt{n}) \\ &= C(\beta + 2\sqrt{n})\tau_k. \end{aligned} \tag{8.50}$$

Here C is the constant from Lemma 3.14.

Note that the unique solution satisfies the strict complementarity conditions due to the Goldman-Tucker Theorem 3.12. Therefore Lemma 8.25 indicates that there exists constants $\varepsilon > 0$ and $L > 0$ such that

$$\|\theta''(x, s, \tau)\| \le L, \quad \text{whenever} \quad \|(x, s, \tau) - (x^*, s^*, 0)\| \le \varepsilon. \tag{8.51}$$

Hence, for all k sufficiently large and $\eta \in (0,1]$, we have for each $i \in \{1,\ldots,n\}$ that

$$\begin{aligned}
&|\theta_i(x_i^k + \Delta x_i^k, s_i^k + \Delta s_i^k, \eta\tau_k)| \\
&\le \left| \theta_i(x_i^k, s_i^k, \tau_k) + \theta_i'(x_i^k, s_i^k, \tau_k)^{\mathrm{T}} \begin{bmatrix} \Delta x_i^k \\ \Delta s_i^k \\ (\eta - 1)\tau_k \end{bmatrix} \right| \\
&\quad + \frac{1}{2} \left| \begin{bmatrix} \Delta x_i^k \\ \Delta s_i^k \\ (\eta - 1)\tau_k \end{bmatrix}^{\mathrm{T}} \theta_i''(x_i^k + \vartheta_i \Delta x_i^k, s_i^k + \vartheta_i \Delta s_i^k, (1 + \vartheta_i(\eta - 1))\tau_k) \begin{bmatrix} \Delta x_i^k \\ \Delta s_i^k \\ (\eta - 1)\tau_k \end{bmatrix} \right| \\
&= \left| \theta_i(x_i^k, s_i^k, \tau_k) + \theta_i'(x_i^k, s_i^k, \tau_k)^{\mathrm{T}} \begin{bmatrix} \Delta x_i^k \\ \Delta s_i^k \\ -\tau_k \end{bmatrix} + \eta\tau_k \frac{\partial \theta_i}{\partial \tau}(x^k, s^k, \tau_k) \right| \\
&\quad + \frac{1}{2} \left| \begin{bmatrix} \Delta x_i^k \\ \Delta s_i^k \\ (\eta - 1)\tau_k \end{bmatrix}^{\mathrm{T}} \theta_i''(x_i^k + \vartheta_i \Delta x_i^k, s_i^k + \vartheta_i \Delta s_i^k, (1 + \vartheta_i(\eta - 1))\tau_k) \begin{bmatrix} \Delta x_i^k \\ \Delta s_i^k \\ (\eta - 1)\tau_k \end{bmatrix} \right| \\
&\le \eta\tau_k \left| \frac{\partial \theta_i}{\partial \tau}(x^k, s^k, \tau_k) \right| + \frac{L}{2} \left\| \begin{bmatrix} \Delta x_i^k \\ \Delta s_i^k \\ (\eta - 1)\tau_k \end{bmatrix} \right\|^2 \\
&\le 2\eta\tau_k + \frac{L}{2} \left(\left\| \begin{bmatrix} \Delta x_i^k \\ \Delta s_i^k \end{bmatrix} \right\|^2 + (1 - \eta)^2 \tau_k^2 \right),
\end{aligned}$$

where the last inequality follows from the fact that $|\partial\theta_i(a,b,\tau)/\partial\tau| \le 2$ for all a, b, and τ. With (8.50) and $\eta \in [0,1)$ it is now easy to see that

$$\begin{aligned}
\|\theta(x^k + \Delta x, s^k + \Delta s, \eta\tau_k)\| &\le 2\sqrt{n}\eta\tau_k + \frac{L}{2}\left(C^2(\beta + 2\sqrt{n})^2 + \sqrt{n}(1 - \eta)^2\right)\tau_k^2 \\
&\le 2\sqrt{n}\eta\tau_k + \frac{L}{2}\left(C^2(\beta + 2\sqrt{n})^2 + \sqrt{n}\right)\tau_k^2
\end{aligned} \tag{8.52}$$

Hence the inequality (8.33) holds with $\ell_k = 0$ for all k sufficiently large. Taking into account that $\beta > 2\sqrt{n}$, it is easy to verify that

$$2\sqrt{n}\eta\tau_k + \frac{L}{2}\left(C^2(\beta + 2\sqrt{n})^2 + \sqrt{n}\right)\tau_k^2 \le \eta\beta\tau_k$$

whenever

$$\eta \ge \frac{L}{2(\beta - 2\sqrt{n})}\left(C^2(\beta + 2\sqrt{n})^2 + \sqrt{n}\right)\tau_k,$$

and so

$$\eta_k \le \frac{L}{2\varrho(\beta - 2\sqrt{n})}\left(C^2(\beta + 2\sqrt{n})^2 + \sqrt{n}\right)\tau_k,$$

for all k sufficiently large. Therefore by (8.34) and (8.37) it follows that

$$\tau_{k+1} = O(\tau_k^2),$$

i. e., the sequence $\{\tau_k\}$ converges locally quadratically to zero. ■

8.3.3 Numerical Results

Implementation

Algorithm 8.18 was implemented in C. In order to simplify the work, the code PCx by Czyzyk, Mehrotra, Wagner, and Wright [16, 17] was used and modified appropriately. PCx is a predictor-corrector interior-point solver for linear programs, written in C and calling a FORTRAN subroutine in order to solve certain linear systems using the sparse Cholesky method by Ng and Peyton [47] which is also employed by Zhang's LIPSOL code. PCx, as well as LIPSOL, uses the minimum degree ordering code by Liu [39]. Dense column handling is performed by using the Sherman-Morrison-Woodbury approach or by applying a preconditioned conjugate gradient method.

Since the linear systems occurring in Algorithm 8.18 have essentially the same structure as those arising in interior-point methods, it has been possible to use the numerical linear algebra part from PCx for the implementation of Algorithm 8.18. The C-code's main program has been rewritten from scratch.

The initial point $w^0 = (x^0, \lambda^0, s^0)$ is the same as the one used for the numerical experiments in Section 8.2.3. The termination criteria described in Section 8.2.3 are used again.

Jacobian Smoothing Predictor Step

Another fact to note is that Algorithm 8.18's implementation includes a Jacobian smoothing-component, i. e., the linear system (8.32) solved in the predictor step (Step 2) has been changed to:

$$\Theta'(w^k, \tau_k) \begin{bmatrix} \Delta w \\ \Delta \tau \end{bmatrix} = -\Theta(w^k, 0).$$

Numerical experiments have shown implementation to be superior to the one described in algorithm 8.18 when comparing the iteration counts produced by both implementations.

Presolving

PCx's preprocessor was also applied to the problems. Note that this preprocessor, while being more advanced than LIPSOL's presolver, is still far away from perfect. Based on techniques described by Andersen and Andersen [1], it only tries to resolve some basic inconsistencies in the problem formulation. These include the following checks:

Zero rows and columns in A:

When the column $A_{\cdot i}$ is zero, the value of x_i has no effect on the product Ax. We can deduce the value of x_i by considering its bounds and the i-th element c_i of the cost vector. If $c_i < 0$, then the product $c_i x_i$ is minimized by setting x_i to its upper bound; if there is no upper bound, the primal objective must be unbounded below. Similarly, when $c_i > 0$, then x_i is forced to its lower bound. When $c_i = 0$, then x_i is irrelevant to the problem.

Duplicate rows and columns in A:

When two columns of A are simply scalar multiples of one another, we can replace the two components of x with a single primal variable and reduce n by 1. Similarly, when A contains duplicated rows, we can combine two components of λ and reduce m by one.

Row singleton:

When the i-th row of A contains only a single non-zero element A_{ij}, we can immediately set $x_j = b_i / A_{ij}$ and eliminate row i and column j from the problem, reducing both m and n by 1.

Forced rows:

Sometimes the combination of a single constraint and bounds forces a collection of variables to take on certain fixed values. Therefore all of these variables can be eliminated from the program, further reducing its size.

For example, if one if the constraints is

$$10x_3 - 4x_{10} + x_{12} = -4,$$

and the bounds on the variables are

$$x_3 \in [0, \infty), \ x_{10} \in [0, 1], \text{ and, } x_{12} \in [0, \infty),$$

the values of these three variables must be $x_3 = 0$, $x_{10} = 1$, and $x_{12} = 0$. Therefore all three variables and the corresponding rows of A can be removed from the linear program.

PCx's presolver makes several sweeps through the rows and columns of A. When an opportunity to reduce the problem is detected, the appropriate rows and columns are flagged as "deleted" and the information is pushed onto a stack. Another sweep through A can uncover new opportunities for reductions, so repeated passes through the data are performed until no more reductions are found.

Once the reduced problem is solved a postprocessing phase is activated to undo the changes made by the presolver by popping the stack. This way the solution can be expressed in terms of the original model.

Since LIPSOL's preprocessor only catches some of these potential problems, one should expect an improvement concerning the efficiency and robustness of the actual LP solver by the use of PCx's preprocessor. As redundant information is removed, the resulting problems usually contain fewer variables, which also lessens the time the actual solver has spend on the problems. At times, some components of the solution can be determined without recourse to a sophisticated algorithm.

Presolving is less expensive in terms of CPU-time than a single primal-dual iteration, so it is almost always worth doing. For more information see Andersen and Andersen [1], Andersen, et. al [2] and references therein.

Adaptive Choice of the Centering Parameter

The centering parameter $\hat{\sigma}_k$ is chosen as follows:

We let $\hat{\sigma}_{\min} = 0.4, \hat{\sigma}_{\max} = 0.6, \gamma = 0.1$, start with $\hat{\sigma}_0 = 0.5$ and set

$$\hat{\sigma}_{k+1} := \min\{\hat{\sigma}_{\max}, \hat{\sigma}_k + \gamma\}$$

if the predictor step was successful (i. e., if we were allowed to take $\hat{w}^{k+1} = w^k + \Delta w^k$), and

$$\hat{\sigma}_{k+1} := \max\{\hat{\sigma}_{\min}, \hat{\sigma}_k - \gamma\}$$

otherwise. This strategy guarantees that all centering parameters belong to the interval $[\hat{\sigma}_{\min}, \hat{\sigma}_{\max}]$. According to experimental numerical results, a larger value of $\hat{\sigma}_k$ usually gives faster convergence, but the entire behavior of the algorithm becomes less stable. A smaller value of the centering parameter gives a more stable behavior, while the overall number of iterations increases. The dynamic choice of $\hat{\sigma}_k$ given above tries to combine these observations in a suitable way.

The remaining parameters from Step 0 of Algorithm 8.18 are chosen as follows:

$$\varrho = 0.79 \quad \text{and} \quad \beta := \frac{\|\theta(x^0, s^0, \tau_0)\|}{\tau_0}.$$

First we consider the function $\psi(\tau) := \tau$ (this, more or less, corresponds to the method from Section 8.1.3)

Numerical Results of Algorithm 8.18 for Different ψ-Functions

For easier comparison to results produced by PCx a column indicating the CPU-time required to solve each problem is included. All programs were compiled using the Sun FORTE FORTRAN 77 compiler version 6u2 for the sparse Cholesky solver and the GNU C Compiler version 2.95.2 for the C-part of program. Other combinations of GNU and Sun compilers have proven to produce less efficient binary code and therefore require more CPU-time to solve a problem. This is most likely due to the fact that neither the GNU f77 compiler front end nor Sun's C compiler are as highly optimized as the Sun FORTRAN and GNU C compilers, respectively. The code was compiled with the highest possible level of optimization enabled

(-O6 for the C- and -O for the FORTRAN compiler). All test runs were done on a Sun Enterprise 450.

Table 8.6 contains the results for the minimum function and $\psi(\tau) := \tau$, with the columns of table 8.6 having the following meanings:

Problem:	Name of the test problem in the Netlib collection,
m:	Number of equality constraints (after preprocessing),
n:	Number of variables (after preprocessing),
k:	Number of iterations until termination,
P:	Number of accepted predictor steps,
τ_f:	Value of τ_k at the final iterate,
$\|\Phi(w^f)\|_\infty$:	Value of $\|\Phi(w^k)\|_\infty$ at the final iterate,
Primal Objective:	Value of the primal objective function at final iterate.
CPU:	CPU-time in seconds required to solve the problem. This does not include the time required to perform disc i/o to read the MPS-file containing the problem data and to produce possible output files, nor is the time required to run the presolver included. Both of these times are identical for all PCx-based codes and are therefore irrelevant for the comparison.

Table 8.6: Numerical results of Algorithm 8.18 with $\psi(\tau) := \tau$ using the minimum function

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective	CPU
25fv47	788	1843	29	13	1.2e−04	1.509e−04	5.5018459e+03	2.50
80bau3b	2140	11066	61	17	1.9e−04	3.787e−04	9.8722419e+05	20.03
adlittle	55	137	15	13	1.4e−02	1.388e−05	2.2549496e+05	0.06
afiro	27	51	13	10	2.5e−02	3.678e−05	−4.6475311e+02	0.03
agg	390	477	26	17	1.6e−02	1.411e−04	−3.5991767e+07	0.68
agg2	514	750	31	19	2.5e−03	2.644e−04	−2.0239252e+07	1.47
agg3	514	750	27	17	1.4e−02	2.059e−04	1.0312116e+07	1.30
bandm	240	395	16	13	1.1e−04	3.972e−05	−1.5862802e+02	0.23
beaconfd	86	171	22	19	5.4e−04	7.035e−04	3.3592486e+04	0.18
blend	71	111	10	10	8.2e−04	4.086e−05	−3.0812150e+01	0.04
bnl1	610	1491	37	18	2.9e−04	3.167e−04	1.9776296e+03	1.67
bnl2	1964	4008	30	12	2.6e−04	5.068e−04	1.8112366e+03	6.47
boeing1	331	697	22	15	1.5e−04	2.144e−04	−3.3521358e+02	0.58
boeing2	126	265	23	10	2.4e−03	9.683e−06	−3.1501873e+02	0.23
bore3d	81	138	15	12	1.5e−03	3.814e−06	1.3730804e+03	0.08
brandy	133	238	18	15	3.4e−04	3.623e−04	1.5185099e+03	0.22
capri	241	436	16	14	2.4e−03	8.084e−04	2.6900118e+03	0.24
cycle	1420	2773	31	13	8.0e−05	1.759e−04	−5.2263987e+00	4.14
czprob	671	2779	16	14	2.5e−03	1.753e−05	2.1851967e+06	0.81
d2q06c	2132	5728	55	11	4.1e−04	6.343e−04	1.2278421e+05	21.69
d6cube	403	5443	15	11	1.3e−04	2.326e−05	3.1549157e+02	3.99
degen2	444	757	11	11	8.3e−04	1.071e−05	−1.4351780e+03	0.56
degen3	1503	2604	12	11	1.6e−04	2.585e−05	−9.8729405e+02	6.81
dfl001	6071	12230	> 350		*maximum number of iterations reached*			
e226	198	429	23	14	8.5e−05	5.179e−05	−2.5864931e+01	0.38
etamacro	334	669	19	14	1.5e−04	3.581e−05	−7.5571523e+02	0.48
fffff800	322	826	31	17	4.9e−04	8.090e−04	5.5567957e+05	1.19
finnis	438	935	21	19	3.1e−04	8.108e−04	1.7279107e+05	0.51

Table 8.6: Numerical results of Algorithm 8.18 with $\psi(\tau) := \tau$ using the minimum function (continued)

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective	CPU
fit1d	24	1049	15	14	1.1e−03	3.174e−04	−9.1463796e+03	0.92
fit1p	627	1677	17	14	8.9e−05	1.169e−02	9.1464198e+03	3.46
fit2d	25	10524	51	21	1.5e−04	2.851e−05	−6.8464293e+04	40.42
fit2p	3000	13525	24	20	1.8e−04	3.812e−04	6.8464293e+04	43.30
forplan	121	447	29	17	4.9e−04	8.401e−04	−6.6421901e+02	0.64
ganges	1113	1510	22	20	3.9e−04	4.636e−04	−1.0958574e+05	1.47
gfrd-pnc	590	1134	20	16	1.2e−02	3.308e−05	6.9022360e+06	0.52
greenbea	1933	4153	26	7	*step length too small*			
greenbeb	1932	4154	30	16	3.5e−04	6.945e−04	−4.3022607e+06	4.98
grow15	300	645	244	8	9.4e−05	2.143e−02	−9.8628817e+07	12.45
grow22	440	946	> 350		*maximum number of iterations reached*			
grow7	140	301	198	10	9.9e−05	1.705e−02	−4.3214951e+07	4.77
israel	174	316	20	15	2.8e−03	5.505e−04	−8.9664482e+05	0.70
kb2	43	68	15	11	9.0e−04	1.308e−06	−1.7499001e+03	0.05
lotfi	133	346	25	13	3.1e−04	4.226e−04	−2.5264717e+01	0.20
maros	655	1437	39	17	9.9e−04	7.280e−04	−5.8063744e+04	1.96
maros-r7	2152	7440	27	15	4.8e−04	8.965e−04	1.4971852e+06	63.21
modszk1	665	1599	22	18	2.9e−03	1.682e−04	3.2061973e+02	0.87
nesm	654	2922	51	11	8.3e−05	6.817e−03	1.4076036e+07	5.73
perold	593	1374	29	16	2.2e−04	7.921e−04	−9.3807553e+03	1.72
pilot	1368	4543	72	12	7.0e−05	2.652e−04	−5.5728443e+02	52.49
pilot.ja	810	1804	63	14	2.7e−04	9.259e−04	−6.1131365e+03	8.66
pilot.we	701	2814	42	16	2.7e−04	8.339e−04	−2.7201075e+06	3.36
pilot4	396	1022	29	14	9.9e−05	3.745e−04	−2.5811393e+03	1.63
pilot87	1971	6373	70	11	9.9e−05	8.339e−03	3.0171649e+02	140.35
pilotnov	848	2117	20	17	6.5e−05	1.779e−04	−4.4972762e+03	2.72
recipe	64	123	12	11	4.1e−04	1.426e−06	−2.6661600e+02	0.05
sc105	104	162	19	14	6.2e−04	4.420e−05	−5.2202061e+01	0.08
sc205	203	315	27	16	2.1e−04	1.486e−04	−5.2202062e+01	0.22
sc50a	49	77	15	12	1.8e−03	1.583e−05	−6.4575077e+01	0.04
sc50b	48	76	15	11	4.2e−03	3.091e−05	−6.9999998e+01	0.04
scagr25	469	669	32	14	6.6e−03	1.663e−05	−1.4753433e+07	0.59
scagr7	127	183	19	14	7.0e−04	5.924e−04	−2.3313898e+06	0.09
scfxm1	305	568	17	15	1.4e−03	4.693e−04	1.8416758e+04	0.30
scfxm2	610	1136	25	19	7.5e−04	3.825e−04	3.6660261e+04	0.91
scfxm3	915	1704	20	15	9.6e−04	7.701e−04	5.4901254e+04	1.08
scorpion	340	412	18	15	7.4e−05	5.759e−05	1.8781248e+03	0.21
scrs8	421	1199	19	15	3.6e−03	7.262e−04	9.0429695e+02	0.50
scsd1	77	760	13	12	3.1e−03	9.581e−06	8.6666670e+00	0.20
scsd6	147	1350	12	9	1.2e−04	8.437e−06	5.0499980e+01	0.30
scsd8	397	2750	10	9	4.8e−03	1.945e−05	9.0500000e+02	0.53
sctap1	284	644	13	11	3.4e−03	7.324e−06	1.4122500e+03	0.20
sctap2	1033	2443	20	14	2.3e−03	1.371e−05	1.7248071e+03	1.22
sctap3	1408	3268	36	14	2.5e−03	5.208e−05	1.4240000e+03	3.18
seba	448	901	20	13	1.9e−03	4.716e−05	1.5711600e+04	4.98
share1b	112	248	38	18	4.5e−04	5.924e−04	−7.6589319e+04	0.30
share2b	96	162	17	14	3.5e−04	3.101e−06	−4.1573224e+02	0.10
shell	487	1451	22	17	1.3e−01	6.100e−04	1.2088254e+09	0.69
ship04l	292	1905	21	16	5.1e−04	6.280e−04	1.7933245e+06	0.79
ship04s	216	1281	15	15	3.1e−03	4.180e−04	1.7987147e+06	0.34

Table 8.6: Numerical results of Algorithm 8.18 with $\psi(\tau) := \tau$ using the minimum function (continued)

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective	CPU
ship08l	470	3121	27	17	1.3e−02	5.954e−04	1.9090552e+06	1.80
ship08s	276	1604	17	15	1.1e−02	1.032e−04	1.9200982e+06	0.51
ship12l	610	4171	40	14	1.9e−03	3.211e−04	1.4701879e+06	3.60
ship12s	340	1943	26	16	1.7e−03	2.844e−04	1.4892361e+06	1.03
sierra	1212	2705	28	16	6.1e−03	1.170e−04	1.5394362e+07	2.66
stair	356	532	22	15	6.4e−05	1.273e−04	−2.5126695e+02	0.79
standata	314	796	11	9	6.0e−02	2.032e−04	1.2577010e+03	0.19
standgub	314	796	11	9	6.0e−02	2.032e−04	1.2577010e+03	0.19
standmps	422	1192	17	13	5.5e−03	2.199e−05	1.4060175e+03	0.46
stocfor1	102	150	13	13	2.1e−02	3.290e−04	−4.1131983e+04	0.06
stocfor2	1980	2868	18	13	3.3e−03	1.029e−05	−3.9024400e+04	1.44
stocfor3	15362	22228	99	19	1.3e−04	2.031e−04	−3.9976784e+04	77.12
stocfor3old	15362	22228	99	19	1.3e−04	2.031e−04	−3.9976784e+04	78.69
truss	1000	8806	28	13	4.0e−03	7.474e−05	4.5881585e+05	6.48
tuff	257	567	21	14	6.3e−05	1.327e−04	2.9214753e−01	0.51
vtp.base	72	111	11	11	1.3e−01	1.952e−04	1.2983146e+05	0.05
wood1p	171	1718	20	12	6.5e−05	1.258e−04	1.4428956e+00	4.67
woodw	708	5364	44	10	5.6e−05	2.768e−03	1.3044762e+00	6.54

Note that the run on the entire Netlib problem suite as given above required a total of 3572 iterations, for which 711.24 seconds of CPU-time were necessary.

These results do not seem to be an improvement over the results computed by Algorithm 8.11 (see Table 8.5 on page 117). This is the case since Algorithm 8.18 fails to solve the test cases greenbea and grow22, while Algorithm 8.11 is able to solve all of the grow*-examples as well as the test case greenbea rather quickly. The performance of Algorithm 8.18 on the problems stocfor3 and stocfor3old is also a reason for concern.

In most cases the iteration numbers are higher than those presented in table 8.5. Algorithm 8.18 performs better on the pilot* examples, but it is not able to solve the problems grow7 and grow15 in a reasonable number if iterations.

Even though it is not quite clear if the theory can be adapted to the use of the smoothed Fischer-Burmeister function instead of the Chen-Harker-Kanzow-Smale smoothing function, some numerical experiments have shown quite interesting results, which are even better than those for the minimum function. These results are presented in the following two tables.

The results for the Fischer-Burmeister function and $\psi(\tau) := \tau$ are presented in Table 8.7.

Table 8.7: Numerical results of Algorithm 8.18 with $\psi(\tau) := \tau$ using the Fischer-Burmeister function

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective	CPU
25fv47	788	1843	27	11	1.0e−03	1.758e−04	5.5018459e+03	2.06
80bau3b	2140	11066	29	15	6.0e−04	2.527e−04	9.8722419e+05	6.09
adlittle	55	137	14	12	3.3e−02	2.239e−04	2.2549496e+05	0.05
afiro	27	51	12	12	1.9e−02	5.456e−06	−4.6475316e+02	0.02

Table 8.7: Numerical results of Algorithm 8.18 with $\psi(\tau) := \tau$ using the Fischer-Burmeister function (continued)

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective	CPU
agg	390	477	22	17	3.8e−02	6.257e−04	−3.5991767e+07	0.54
agg2	514	750	22	16	2.1e−02	4.992e−04	−2.0239252e+07	1.00
agg3	514	750	21	16	3.1e−02	5.673e−04	1.0312116e+07	0.96
bandm	240	395	15	12	2.6e−04	9.065e−05	−1.5862803e+02	0.19
beaconfd	86	171	21	18	2.5e−03	5.156e−04	3.3592486e+04	0.16
blend	71	111	10	9	8.1e−04	8.746e−05	−3.0812183e+01	0.04
bnl1	610	1491	30	16	8.2e−04	1.737e−04	1.9776296e+03	1.11
bnl2	1964	4008	25	10	9.3e−04	4.893e−04	1.8112350e+03	5.02
boeing1	331	697	18	13	1.7e−03	3.652e−04	−3.3521357e+02	0.43
boeing2	126	265	18	13	2.7e−03	4.166e−06	−3.1501873e+02	0.14
bore3d	81	138	14	11	5.9e−03	3.980e−05	1.3730804e+03	0.07
brandy	133	238	16	14	2.1e−03	3.469e−04	1.5185099e+03	0.20
capri	241	436	15	14	4.0e−03	9.161e−04	2.6900100e+03	0.19
cycle	1420	2773	30	14	8.2e−05	8.349e−05	−5.2263996e+00	3.60
czprob	671	2779	17	13	5.7e−03	5.207e−05	2.1851967e+06	0.67
d2q06c	2132	5728	48	18	1.8e−04	5.591e−04	1.2278421e+05	18.46
d6cube	403	5443	13	10	5.3e−04	3.157e−05	3.1549167e+02	3.20
degen2	444	757	10	10	1.9e−03	2.901e−05	−1.4351780e+03	0.47
degen3	1503	2604	10	10	9.2e−04	7.742e−05	−9.8729400e+02	5.87
dfl001	6071	12230	> 350		*maximum number of iterations reached*			
e226	198	429	14	13	2.4e−04	1.889e−05	−2.5864929e+01	0.21
etamacro	334	669	20	13	8.8e−05	1.625e−03	−7.5571523e+02	0.42
fffff800	322	826	28	17	1.2e−03	5.876e−04	5.5567956e+05	0.96
finnis	438	935	20	17	2.0e−03	7.843e−04	1.7279107e+05	0.41
fit1d	24	1049	14	14	2.7e−03	8.491e−05	−9.1463781e+03	0.77
fit1p	627	1677	17	14	7.0e−05	3.469e−02	9.1464871e+03	3.18
fit2d	25	10524	17	17	1.4e−03	7.494e−04	−6.8464293e+04	8.94
fit2p	3000	13525	19	19	1.5e−03	9.397e−05	6.8464293e+04	26.93
forplan	121	447	26	17	2.2e−03	4.722e−04	−6.6421896e+02	0.53
ganges	1113	1510	20	19	2.4e−03	1.218e−04	−1.0958574e+05	1.21
gfrd-pnc	590	1134	17	15	3.2e−02	4.308e−04	6.9022360e+06	0.33
greenbea	1933	4153	248	3	*failure*			
greenbeb	1932	4154	43	13	1.7e−03	9.559e−04	−4.3022603e+06	6.15
grow7	140	301	185	15	8.5e−05	2.149e−02	−4.3371411e+07	13.71
grow15	300	645	325	7	1.0e−04	1.959e−02	−9.8472971e+07	3.80
grow22	440	946	> 350		*maximum number of iterations reached*			
israel	174	316	17	15	1.0e−02	4.732e−04	−8.9664482e+05	0.59
kb2	43	68	15	10	1.6e−03	1.653e−06	−1.7499001e+03	0.05
lotfi	133	346	23	12	3.2e−03	7.087e−04	−2.5264704e+01	0.16
maros	655	1437	22	14	2.4e−03	1.738e−04	−5.8063744e+04	0.94
maros-r7	2152	7440	39	14	4.0e−03	8.053e−04	1.4971852e+06	83.03
modszk1	665	1599	21	17	7.2e−03	3.330e−04	3.2061973e+02	0.68
nesm	654	2922	46	9	4.7e−04	4.718e−04	1.4076037e+07	4.19
perold	593	1374	26	12	2.1e−03	6.564e−04	−9.3807553e+03	1.34
pilot	1368	4543	71	9	9.0e−05	1.600e−02	−5.5727421e+02	48.00
pilot.ja	810	1804	30	14	7.1e−04	9.749e−04	−6.1131365e+03	3.93
pilot.we	701	2814	36	10	2.6e−03	9.981e−04	−2.7201075e+06	2.19
pilot4	396	1022	26	12	1.7e−03	6.888e−04	−2.5811392e+03	1.31
pilot87	1971	6373	67	11	9.1e−05	6.095e−03	3.0171547e+20	132.35
pilotnov	848	2117	15	15	2.3e−03	4.059e−04	−4.4972762e+03	1.99

Table 8.7: Numerical results of Algorithm 8.18 with $\psi(\tau) := \tau$ using the Fischer-Burmeister function (continued)

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective	CPU
recipe	64	123	11	10	1.2e−03	4.205e−05	−2.6661600e+02	0.04
sc105	104	162	18	13	1.2e−03	2.793e−05	−5.2202062e+01	0.07
sc205	203	315	24	14	6.8e−04	1.030e−04	−5.2202062e+01	0.16
sc50a	49	77	14	11	4.7e−03	8.546e−05	−6.4575080e+01	0.04
sc50b	48	76	15	10	7.1e−03	7.714e−06	−7.0000005e+01	0.03
scagr25	469	669	31	13	4.7e−03	1.049e−04	−1.4753433e+07	0.46
scagr7	127	183	15	14	3.6e−03	4.563e−04	−2.3313898e+06	0.07
scfxm1	305	568	15	13	8.4e−03	6.230e−04	1.8416759e+04	0.24
scfxm2	610	1136	18	15	2.7e−03	1.834e−04	3.6660262e+04	0.56
scfxm3	915	1704	20	15	5.4e−03	9.098e−04	5.4901255e+04	0.94
scorpion	340	412	19	14	2.4e−04	1.815e−05	1.8781248e+03	0.19
scrs8	421	1199	17	14	1.9e−03	2.169e−04	9.0429322e+02	0.34
scsd1	77	760	12	12	4.2e−03	7.203e−06	8.6666636e+00	0.14
scsd6	147	1350	11	8	3.8e−04	1.937e−05	5.0500000e+01	0.24
scsd8	397	2750	9	9	8.7e−03	3.131e−05	9.0499999e+02	0.37
sctap1	284	644	12	12	9.4e−03	8.910e−06	1.4122500e+03	0.15
sctap2	1033	2443	11	11	7.2e−03	8.233e−05	1.7248071e+03	0.52
sctap3	1408	3268	12	12	3.8e−03	1.051e−05	1.4240000e+03	0.76
seba	448	901	19	12	2.5e−03	1.550e−06	1.5711600e+04	4.46
share1b	112	248	29	14	2.2e−03	3.762e−04	−7.6589319e+04	0.19
share2b	96	162	15	10	2.5e−03	8.099e−05	−4.1573224e+02	0.07
shell	487	1451	19	16	3.4e−01	4.313e−04	1.2088254e+09	0.45
ship04l	292	1905	22	16	5.3e−03	7.616e−04	1.7933245e+06	0.67
ship04s	216	1281	16	13	6.9e−03	1.561e−04	1.7987147e+06	0.28
ship08l	470	3121	25	15	2.1e−03	7.592e−04	1.9090552e+06	1.14
ship08s	276	1604	15	13	3.0e−02	7.416e−04	1.9200982e+06	0.34
ship12l	610	4171	21	13	7.0e−03	2.670e−04	1.4701879e+06	1.28
ship12s	340	1943	18	14	5.3e−03	7.548e−05	1.4892361e+06	0.48
sierra	1212	2705	20	16	7.7e−03	2.548e−05	1.5394362e+07	1.35
stair	356	532	18	14	6.9e−04	2.105e−04	−2.5126695e+02	0.61
standata	314	796	11	10	3.9e−02	1.699e−05	1.2576993e+03	0.14
standgub	314	796	11	10	3.9e−02	1.699e−05	1.2576993e+03	0.15
standmps	422	1192	14	12	9.6e−03	4.418e−05	1.4060175e+03	0.30
stocfor1	102	150	13	13	2.1e−02	2.014e−05	−4.1131983e+04	0.05
stocfor2	1980	2868	14	13	2.0e−03	1.481e−06	−3.9024400e+04	0.93
stocfor3	15362	22228	23	19	2.8e−04	5.514e−05	−3.9976784e+04	13.12
stocfor3old	15362	22228	23	19	2.8e−04	5.514e−05	−3.9976784e+04	13.24
truss	1000	8806	16	13	2.5e−03	3.621e−04	4.5881579e+05	2.77
tuff	257	567	20	15	1.9e−04	4.977e−05	2.9214785e−01	0.45
vtp.base	72	111	10	10	2.9e−01	3.126e−04	1.2983146e+05	0.04
wood1p	171	1718	22	11	1.5e−04	9.009e−05	1.4428646e+00	5.01
woodw	708	5364	36	10	1.0e−04	5.104e−03	1.3044083e+00	4.29

Note that this run on the Netlib problem suite as given above required a total of 1841 iterations for which 432.11 seconds of CPU-time were necessary. A total of 3021 linear systems with different coefficient matrices were solved. These totals do not include the problems `brandy`, `dfl001`, `greenbea`, `greenbeb` and the three `grow*`-problems. This has been done for easier comparison with the results of PCx, which can be found in Table 8.9.

Table 8.7 presents results which are much better than the ones obtained by using the minimum function (cf. Table 8.6). Especially the previously extremely high iteration counts for problems stocfor3 and stocfor3old could be lowered drastically from 99 to 23 in both cases. Unfortunately the failure to solve problem greenbea still exists. After 348 iterations the code stops since the smoothing variable attains a value smaller than 10^{-4}, but the iterate was still far away from the solution, as the value of the primal objective function documents: Instead of $-7.246240 \cdot 10^7$, which is the objective function's value in the exact solution (as documented in the Netlib files), Algorithm 8.18 computed $c^T x^f = 1.2707 \cdot 10^3$ when it terminated. Another error which occurs in both algorithms is the failure to solve problem grow22. However, one has to note that the final iterate generated by Algorithm 8.18 when using the minimum function is apparently closer to the solution than the final iterate generated when using the Fischer-Burmeister function, at least, when the objective function's values are compared. it is still noteworthy, that the Fischer-Burmeister function variant of Algorithm 8.18 comes close to the solution in a significantly shorter time than the minimum function variant.

When comparing these results to those of Algorithm 8.11 it is notable that Algorithm 8.18 fails on three problems in the test set (dfl001, greenbea, grow22), whereas Algorithm 8.11 only fails on dfl001. Comparing the iteration counts for most problems, it becomes obvious that an improvement has been made, especially in the case of the hard problems, such as the pilot*-examples.

In fact, for almost all examples the number of iterations could be reduced considerably regarding both Algorithm 8.11 and Algorithm 8.18 (with the minimum function). This becomes clear when looking at the total number of iterations needed to solve all problems in the Netlib test suite. When using the minimum function, Algorithm 8.18 requires 3572 iterations as opposed to 2760 iterations when using the Fischer-Burmeister function.

We finally state some results for the function $\psi(\tau) := (1+\tau)^2 - 1$.

Table 8.8: Numerical results of Algorithm 8.18 with $\psi(\tau) := (1+\tau)^2 - 1$ using the Fischer-Burmeister function

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective
25fv47	788	1843	32	10	6.9e−04	1.516e−04	5.50184589e+03
80bau3b	2140	11066	49	7	6.5e−04	7.665e−04	9.87224192e+05
adlittle	55	137	16	10	2.7e−02	7.941e−05	2.25494963e+05
afiro	27	51	12	10	1.9e−02	2.034e−05	−4.64753174e+02
agg	390	477	24	14	1.6e−02	1.007e−05	−3.59917673e+07
agg2	514	750	28	13	1.1e−02	7.963e−04	−2.02392524e+07
agg3	514	750	25	14	1.8e−02	1.105e−05	1.03121159e+07
bandm	240	395	16	12	2.8e−04	8.822e−06	−1.58628018e+02
beaconfd	86	171	38	8	4.3e−03	7.533e−04	3.35924858e+04
blend	71	111	10	9	8.1e−04	8.648e−05	−3.08121831e+01
bnl1	610	1491	34	15	4.2e−04	2.046e−04	1.97762954e+03
bnl2	1964	4008	33	7	8.9e−04	8.182e−04	1.81342102e+03
boeing1	331	697	26	9	1.1e−03	2.607e−04	−3.35213566e+02
boeing2	126	265	19	7	6.8e−03	3.505e−05	−3.15018741e+02

Table 8.8: Numerical results of Algorithm 8.18 with $\psi(\tau) := (1+\tau)^2 - 1$ using the Fischer-Burmeister function (continued)

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective
bore3d	81	138	14	10	5.5e−03	3.087e−05	1.37308039e+03
brandy	133	238	21	12	2.1e−03	3.965e−04	1.51850990e+03
capri	241	436	20	9	1.2e−02	3.888e−04	2.69001290e+03
cycle	1420	2773	33	10	3.1e−04	8.515e−05	−5.22639040e+00
czprob	671	2779	26	7	4.1e−03	3.548e−06	2.18519670e+06
d2q06c	2132	5728	68	13	7.2e−05	1.179e−03	1.22784214e+05
d6cube	403	5443	16	11	2.6e−04	1.569e−05	3.15491667e+02
degen2	444	757	11	10	1.2e−03	5.838e−07	−1.43517800e+03
degen3	1503	2604	10	9	1.2e−03	8.302e−05	−9.87294002e+02
dfl001	6071	12230	> 350		*maximum number of iterations reached*		
e226	198	429	13	11	3.7e−04	3.843e−05	−2.58649291e+01
etamacro	334	669	23	11	1.0e−04	1.473e−05	−7.55715232e+02
fffff800	322	826	33	12	1.1e−03	9.914e−04	5.55679564e+05
finnis	438	935	32	7	8.5e−04	8.702e−04	1.72827128e+05
fit1d	24	1049	21	14	8.3e−04	2.259e−04	−9.14637809e+03
fit1p	627	1677	38	13	6.6e−05	1.269e−01	9.14645845e+03
fit2d	25	10524	29	14	8.9e−05	2.406e−03	−6.84642934e+04
fit2p	3000	13525	> 350		*maximum number of iterations reached*		
forplan	121	447	29	12	3.1e−03	5.568e−04	−6.64218850e+02
ganges	1113	1510	26	14	2.7e−03	1.539e−04	−1.09585736e+05
gfrd-pnc	590	1134	23	9	2.9e−02	4.750e−04	6.90223600e+06
greenbea	1933	4153	296	3	*step length too small*		
greenbeb	1932	4154	56	7	1.5e−03	7.772e−04	−4.30225229e+06
grow15	300	645	> 350		*maximum number of iterations reached*		
grow22	440	946	> 350		*maximum number of iterations reached*		
grow7	140	301	228	5	7.1e−05	1.694e−02	−4.32769031e+07
israel	174	316	28	9	6.2e−03	3.699e−05	−8.96644822e+05
kb2	43	68	14	9	2.8e−03	4.988e−05	−1.74990013e+03
lotfi	133	346	25	9	2.7e−03	5.972e−04	−2.52647150e+01
maros	655	1437	23	10	2.8e−03	3.551e−04	−5.80637437e+04
maros-r7	2152	7440	25	12	3.1e−03	6.525e−04	1.49718517e+06
modszk1	665	1599	28	12	4.3e−03	8.353e−04	3.20598092e+02
nesm	654	2922	43	3	1.6e−03	9.851e−04	1.40760365e+07
perold	593	1374	33	8	1.5e−03	4.714e−04	−9.38075528e+03
pilot	1368	4543	66	7	1.7e−04	5.401e−04	−5.57303588e+02
pilot.ja	810	1804	50	6	2.2e−04	4.265e−04	−6.11313633e+03
pilot.we	701	2814	45	6	1.9e−03	8.489e−04	−2.72010753e+06
pilot4	396	1022	33	9	1.7e−03	7.245e−04	−2.58113842e+03
pilot87	1971	6373	107	3	9.2e−05	1.010e−02	3.02716952e+02
pilotnov	848	2117	25	6	1.1e−03	3.756e−04	−4.49727619e+03
recipe	64	123	11	10	1.1e−03	2.346e−05	−2.66616000e+02
sc105	104	162	16	13	1.3e−03	5.456e−05	−5.22020619e+01
sc205	203	315	18	14	4.4e−04	3.869e−05	−5.22020615e+01
sc50a	49	77	15	12	1.9e−03	3.248e−06	−6.45750773e+01
sc50b	48	76	12	11	4.5e−03	3.960e−06	−7.00000019e+01
scagr25	469	669	20	10	1.3e−02	6.860e−06	−1.47534331e+07
scagr7	127	183	19	13	1.7e−03	1.379e−04	−2.33138982e+06
scfxm1	305	568	26	12	3.9e−03	1.724e−04	1.84167590e+04
scfxm2	610	1136	28	9	4.4e−03	2.755e−04	3.66602615e+04
scfxm3	915	1704	27	12	3.2e−03	2.184e−04	5.49012545e+04

Table 8.8: Numerical results of Algorithm 8.18 with $\psi(\tau) := (1+\tau)^2 - 1$ using the Fischer-Burmeister function (continued)

Problem	m	n	k	P	τ_f	$\|\Phi(w^f)\|_\infty$	Primal Objective
scorpion	340	412	20	12	2.3e−04	2.095e−05	1.87812482e+03
scrs8	421	1199	26	10	3.0e−03	1.583e−04	9.04296954e+02
scsd1	77	760	13	11	4.9e−03	1.700e−05	8.66665929e+00
scsd6	147	1350	11	8	3.8e−04	2.034e−05	5.05000001e+01
scsd8	397	2750	9	8	1.3e−02	8.734e−05	9.04999977e+02
sctap1	284	644	14	10	6.9e−03	2.492e−06	1.41225000e+03
sctap2	1033	2443	14	11	4.2e−03	8.108e−06	1.72480714e+03
sctap3	1408	3268	14	11	4.1e−03	1.126e−05	1.42400000e+03
seba	448	901	16	11	2.3e−03	1.263e−06	1.57116000e+04
share1b	112	248	27	13	2.2e−03	3.420e−04	−7.65893186e+04
share2b	96	162	14	10	1.2e−03	1.112e−05	−4.15732241e+02
shell	487	1451	29	11	9.4e−02	9.766e−06	1.20882535e+09
ship04l	292	1905	20	9	1.2e−02	7.977e−04	1.79332454e+06
ship04s	216	1281	18	9	2.5e−02	7.025e−04	1.79871470e+06
ship08l	470	3121	20	10	8.1e−03	2.125e−04	1.90905521e+06
ship08s	276	1604	18	9	3.1e−02	9.049e−04	1.92009821e+06
ship12l	610	4171	22	9	1.1e−02	7.134e−04	1.47018792e+06
ship12s	340	1943	20	10	3.5e−03	3.788e−04	1.48923611e+06
sierra	1212	2705	27	10	1.0e−02	5.200e−05	1.53943623e+07
stair	356	532	20	12	5.9e−04	1.683e−04	−2.51266951e+02
standata	314	796	15	7	5.7e−02	2.779e−05	1.25769906e+03
standgub	314	796	15	7	5.7e−02	2.779e−05	1.25769906e+03
standmps	422	1192	20	8	9.1e−03	4.456e−05	1.40601750e+03
stocfor1	102	150	19	7	2.9e−02	4.431e−05	−4.11319832e+04
stocfor2	1980	2868	20	9	2.3e−03	1.886e−06	−3.90243999e+04
stocfor3	15362	22228	40	8	4.6e−04	1.099e−04	−3.99767824e+04
stocfor3old	15362	22228	40	8	4.6e−04	1.099e−04	−3.99767824e+04
truss	1000	8806	23	15	3.7e−03	7.499e−06	4.58815847e+05
tuff	257	567	26	9	4.1e−04	1.046e−04	2.92149276e−01
vtp.base	72	111	13	10	2.0e−01	9.302e−04	1.29831462e+05
wood1p	171	1718	30	8	3.6e−04	8.402e−05	1.44290243e+00
woodw	708	5364	40	7	5.1e−05	1.826e−04	1.30447607e+00

Looking at table 8.8, one notices that the results are not quite as good as the ones presented in table 8.7, but still better than the ones generated by the some other algorithms presented in this thesis. It is also notable that new failures are present for problems `grow15` and `fit2p`. The number of iteration required to obtain a solution for the two `stocfor3*` problems has increased from 23 to 40.

Considering the above tables shows that the best numerical performance has been achieved by using Algorithm 8.18 together with $\psi(\tau) := \tau$.

It is quite notable that all of the algorithms presented here have problems to solve the test cases `grow7`, `grow15` and `grow22`. Other problems which are apparently hard to solve for smoothing-type methods include `pilot87` and `pilot.we`. It is not clear, why this is the case, as the Netlib files include no information regarding the structure and the background of the test problems.

Numerical Results of PCx

Now compare the above results with those from a run of PCx [16, 17] on the entire Netlib test set. Version 1.2beta of PCx was used to make these computations. Table 8.9 contains the corresponding results, with the columns of table 8.9 having the following meanings:

Problem:	Name of the test problem in the Netlib collection,
m:	Number of equality constraints (after preprocessing),
n:	Number of variables (after preprocessing),
k:	Number of iterations until termination,
Primal Objective:	Value of the primal objective function at final iterate.
Status	Status with which PCx terminated: optimal (opt), infeasible or unknown. PCx terminates with status *unknown* in cases where the code is unable to resolve the question of feasibility, i. e., either if it exhibits slow convergence or if the improvement in the duality measure μ is greater than the improvement in primal and dual infeasibility. See [16] for further details.
CPU	CPU-time in seconds required to solve the problem. This does not include time spend on disk i/o or the time required for the presolver to run as these are identical for all PCx-based codes used here.

Table 8.9: Numerical results of PCx

Problem	m	n	k	Primal Objective	Status	CPU
25fv47	788	1843	22	5.50184598e+03	optimal	1.09
80bau3b	2140	11066	37	9.87224443e+05	optimal	3.08
adlittle	55	137	12	2.25494963e+05	optimal	0.02
afiro	27	51	8	−4.64753143e+02	optimal	0.01
agg	390	477	17	−3.59917673e+07	optimal	0.26
agg2	514	750	20	−2.02392524e+07	optimal	0.56
agg3	514	750	19	1.03121159e+07	optimal	0.54
bandm	240	395	17	−1.58628018e+02	optimal	0.11
beaconfd	86	171	10	3.35924858e+04	optimal	0.05
blend	71	111	10	−3.08121498e+01	optimal	0.02
bnl1	610	1491	39	1.97762957e+03	optimal	0.68
bnl2	1964	4008	31	1.81123661e+03	optimal	4.13
boeing1	331	697	20	−3.35213567e+02	optimal	0.22
boeing2	126	265	14	−3.15018728e+02	optimal	0.06
bore3d	81	138	16	1.37308039e+03	optimal	0.04
brandy	133	238	16	1.51850990e+03	*unkown*	
capri	241	436	19	2.69001291e+03	optimal	0.13
cycle	1420	2773	21	−5.22639302e+00	optimal	1.64
czprob	671	2779	26	2.18519682e+06	optimal	0.40
d2q06c	2132	5728	24	1.22784233e+05	optimal	6.16
d6cube	403	5443	16	3.15491668e+02	optimal	2.25
degen2	444	757	11	−1.43517800e+03	optimal	0.32
degen3	1503	2604	14	−9.87293999e+02	optimal	5.08
dfl001	5984	12143	71	1.12663961e+07	optimal	802.29
e226	198	429	18	−2.58649291e+01	optimal	0.14

Table 8.9: Numerical results of PCx (continued)

Problem	m	n	k	Primal Objective	Status	CPU
etamacro	334	669	25	−7.55715223e+02	optimal	0.28
fffff800	322	826	25	5.55679599e+05	optimal	0.50
finnis	438	935	25	1.72791066e+05	optimal	0.24
fit1d	24	1049	17	−9.14637659e+03	optimal	0.47
fit1p	627	1677	17	9.14637812e+03	optimal	0.63
fit2d	25	10524	23	−6.84642917e+04	optimal	6.24
fit2p	3000	13525	19	6.84644144e+04	optimal	4.20
forplan	121	447	20	−6.64218961e+02	optimal	0.24
ganges	1113	1510	17	−1.09585736e+05	optimal	0.54
gfrd-pnc	590	1134	18	6.90223600e+06	optimal	0.15
greenbea	1933	4153	9	2.19982120e+03	*infeasible*	
greenbeb	1932	4154	37	−4.30226031e+06	*unknown*	
grow7	140	301	17	−4.77878118e+07	optimal	0.15
grow15	300	645	21	−1.06870941e+08	optimal	0.38
grow22	440	946	22	−1.60834336e+08	optimal	0.60
israel	174	316	19	−8.96644817e+05	optimal	0.39
kb2	43	68	13	−1.74990013e+03	optimal	0.02
lotfi	133	346	15	−2.52647061e+01	optimal	0.05
maros	655	1437	20	−5.80637437e+04	optimal	0.42
maros-r7	2152	7440	14	1.49718517e+06	optimal	22.62
modszk1	665	1599	22	3.20619895e+02	optimal	0.33
nesm	654	2922	27	1.40760378e+07	optimal	1.06
perold	593	1374	31	−9.38075527e+03	optimal	0.94
pilot	1368	4543	31	−5.57489695e+02	optimal	16.73
pilot.ja	810	1804	29	−6.11313646e+03	optimal	2.44
pilot.we	701	2814	46	−2.72010748e+06	optimal	1.17
pilot4	396	1022	46	−2.58113926e+03	optimal	1.33
pilot87	1971	6373	30	3.01710519e+02	optimal	48.89
pilotnov	848	2117	16	−4.49727619e+03	optimal	1.24
recipe	64	123	9	−2.66616000e+02	optimal	0.02
sc105	104	162	10	−5.22020612e+01	optimal	0.02
sc205	203	315	11	−5.22020612e+01	optimal	0.04
sc50a	49	77	8	−6.45750771e+01	optimal	0.01
sc50b	48	76	6	−7.00000000e+01	optimal	0.01
scagr25	469	669	18	−1.47534331e+07	optimal	0.13
scagr7	127	183	14	−2.33138982e+06	optimal	0.03
scfxm1	305	568	17	1.84167590e+04	optimal	0.14
scfxm2	610	1136	20	3.66602616e+04	optimal	0.31
scfxm3	915	1704	20	5.49012545e+04	optimal	0.46
scorpion	340	412	12	1.87812482e+03	optimal	0.07
scrs8	421	1199	22	9.04296961e+02	optimal	0.20
scsd1	77	760	9	8.66666667e+00	optimal	0.05
scsd6	147	1350	12	5.05000002e+01	optimal	0.11
scsd8	397	2750	11	9.05000075e+02	optimal	0.21
sctap1	284	644	16	1.41225000e+03	optimal	0.10
sctap2	1033	2443	14	1.72480714e+03	optimal	0.32
sctap3	1408	3268	15	1.42400000e+03	optimal	0.46
seba	448	901	12	1.57116000e+04	optimal	1.85
share1b	112	248	19	−7.65893186e+04	optimal	0.07
share2b	96	162	17	−4.15732241e+02	optimal	0.05
shell	487	1451	21	1.20882535e+09	optimal	0.20

Table 8.9: Numerical results of PCx (continued)

Problem	m	n	k	Primal Objective	Status	CPU
ship04l	292	1905	13	1.79332454e+06	optimal	0.15
ship04s	216	1281	13	1.79871471e+06	optimal	0.10
ship08l	470	3121	16	1.90905521e+06	optimal	0.29
ship08s	276	1604	12	1.92009821e+06	optimal	0.12
ship12l	610	4171	16	1.47018797e+06	optimal	0.37
ship12s	340	1943	13	1.48923613e+06	optimal	0.15
sierra	1212	2705	21	1.53943622e+07	optimal	0.57
stair	356	532	13	−2.51266951e+02	optimal	0.28
standata	314	796	13	1.25769951e+03	optimal	0.08
standgub	314	796	13	1.25769951e+03	optimal	0.08
standmps	422	1192	26	1.40601750e+03	optimal	0.23
stocfor1	102	150	12	−4.11319832e+04	optimal	0.03
stocfor2	1980	2868	20	−3.90243966e+04	optimal	0.65
stocfor3	15362	22228	31	−3.99767824e+04	optimal	8.55
stocfor3old	15362	22228	31	−3.99767824e+04	optimal	8.47
truss	1000	8806	20	4.58815847e+05	optimal	1.57
tuff	257	567	18	2.92147823e−01	optimal	0.24
vtp.base	72	111	11	1.29831463e+05	optimal	0.02
wood1p	171	1718	20	1.44290241e+00	optimal	2.62
woodw	708	5364	31	1.30447633e+00	optimal	1.69

PCx required 1682 iterations for this run. For better comparison with the results of Algorithm 8.18 this count does not include the problems brandy, dfl001, greenbea, greenbeb as well as the three grow*-problems, since these problems could not be solved either by PCx or by Algorithm 8.18). The PCx-software failed on the problems brandy, greenbea, and greenbeb. It took 802.29 seconds of CPU-time to solve the problem dfl001; all remaining test cases could be solved in another 169.32 seconds.

One of the most interesting aspect of the comparison of the results from Tables 8.7 and 8.9 is the following: PCx required approximately 91% of the number of iterations to solve the problems from the Netlib test suite, if compared to Algorithm 8.18 (1682 vs. 1841 iterations). Compared to PCx, Algorithm 8.18 required 2.55 times the CPU-time (432.11 s vs. 169.23 s). This does not include the problems either one of the algorithms failed to solve (brandy, dfl001, greenbea, greenbeb, as well as the three grow*-problems.

When comparing the CPU-times one has to keep in mind that the implementation of Algorithm 8.18 had to perform almost twice as many matrix factorizations as PCx (3021 vs. 1682 factorizations).

With these results one can be hopeful that further research in this area will yield an algorithm which, if implemented as efficiently as state-of-the-art interior-point codes, is at least comparable, if not superior to interior-point solvers in terms of CPU-time. This is especially the case since the performance of Algorithm 8.18 appears to be better on larger problems than on the smaller ones.

9 Concluding Remarks

This thesis presents a new approach to the numerical solution of linear optimization problems, ranging from a simple Jacobian smoothing method to complex multi-step predictor-corrector algorithms.

Even though the Jacobian smoothing approach exhibits good theoretical convergence properties, it did not provide a satisfactory performance in terms of the number of iterations required to find an approximate solution of the linear programming problem. This could be improved by combining the Jacobian smoothing method with a regular smoothing method to achieve better global convergence behavior. The numerical results produced by the two methods presented here are quite good. When comparing iteration counts, those of the three step method from Section 8.1.3 are quite promising. Unfortunately each iteration is rather expensive in terms of CPU-time, as up to three linear systems with different coefficient matrices have to be solved. An implementation of Algorithm 8.1 based on the PCx framework could provide even better results, since it benefits from PCx's preprocessor. In comparison to LIPSOL' presolver this preprocessor is by far superior, even though there are better ones (commercially) available.

The current state-of-the-art in the field of smoothing-type methods for linear programs is presented in Algorithm 8.18, which produces results almost as good as a modern-day interior-point solver. Note, however, that the implementation differs in one major aspect from the theoretical model of the algorithm: As opposed to the presentation of Algorithm 8.18, the actual implementation contains a Jacobian smoothing predictor step rather than a regular smoothing predictor step.

Even though the best results presented here (Table 8.7) are not backed by a convergence theory, as they were achieved using the Fischer-Burmeister NCP-function, for which no convergence result could be established, they are remarkable. One needs to keep in mind that most implementations of interior-point methods are based on Mehrotra's predictor-corrector algorithm, which also relies heavily heuristics, especially as far as the generation of an initial point, the computation of the primal and dual step lengths and the calculation of the centering parameter are concerned.

Research in the area of smoothing-type algorithms is still ongoing. The author feels that this research will eventually result in methods with a performance equal to, if not superior to, interior-point algorithms. Interior-point methods have been developed and improved over a timespan of more than a decade, whereas the field of smoothing-type methods is by far younger. Therefore one should expect further advances in this area, so that it is quite possible that these algorithms could replace interior-point methods as the leaders in the field of linear programming.

Bibliography

[1] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Mathematical Programming*, 71:2221–245, 1995.

[2] E. D. Andersen, J. Gondzio, C. Mészáros, and X. Xu. Implementation of interior-point methods for large scale linear programming. Technical report, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, Geneva, Switzerland, January 1996. To appear in Interior Point Methods in Mathematical Programming, Kluwer Academic Publishers.

[3] K.-H. Borgwardt. *The Simplex Method. A Probabilistig Analysis.* Springer Verlag, Berlin, 1987.

[4] J. V. Burke and S. Xu. The global linear convergence of a non-interior path-following algorithm for linear complementarity problems. *Mathematics of Operations Research*, 23:719–734, 1998.

[5] J. V. Burke and S. Xu. A non-interior predictor-corrector path-following method for LCP. In Masao Fukushima and S. Qi, editors, *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, pages 45–63. Kluwer Academic Press, 1998.

[6] J. V. Burke and S. Xu. Preliminary numerical experience with non-interior path following methods for LCP. Talk presented at the *International Conference on Non-Linear Programming and Variational Inequalities*, Hongkong, December 1998.

[7] J. V. Burke and S. Xu. A polynomial time interior-point path-following algorithm for LCP based on Chen-Harker-Kanzow-Smale smoothing techniques. *Mathematical Programming*, 86:91–103, 1999.

[8] J. V. Burke and S. Xu. A non-interior predictor-corrector path following algorithm for the monotone linear complementarity problem. *Mathematical Programming*, 87:113–130, 2000.

[9] B. Chen and X. Chen. A global and local superlinear continuation-smoothing method for P_0 and R_0 NCP or monotone NCP. *SIAM Journal on Optimization*, 9:624–645, 1999.

[10] B. Chen, X. Chen, and C. Kanzow. A penalized Fischer-Burmeister NCP-function. *Mathematical Programming*, 88:211–216, 2000.

[11] B. Chen and P. T. Harker. A non-interior-point continuation method for linear complementarity problems. *SIAM Journal on Matrix Analysis and Applications*, 14:1168–1190, 1993.

[12] B. Chen and O. L. Mangasarian. A class of smoothing functions for non-linear and mixed complementarity problems. *Computational Optimization and Applications*, 5:97–138, 1996.

[13] B. Chen and N. Xiu. A global linear and local quadratic non-interior continuation method for non-linear complementarity problems based on Chen-Mangasarian smoothing functions. *SIAM Journal on Optimization*, 9:605–623, 1999.

[14] X. Chen, L. Qi, and D. Sun. Global and superlinear convergence of the smoothing Newton method and its applications to general box constrained variational inequalities. *Mathematica of Computation*, 67:519–540, 1998.

[15] X. Chen and Y. Ye. On homotopy-smoothing methods for box-constrained variational inequalities. *SIAM Journal on Control and Optimization*, 67:589–616, 1999.

[16] J. Czyzyk, S. Mehrotra, M. Wagner, and S. J. Wright. *PCx Users Guide (Version 1.1)*, November 1997.

[17] J. Czyzyk, S. Mehrotra, and S. J. Wright. PCx: An interior-point code for linear programming. *Optimization Methods and Software*, 11, 12:397–430, 1999.

[18] G. B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, N.J., 1963.

[19] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Non-Linear Equations*. Prentice Hall, Englewood Cliffs, NJ, 1983. Reprinted by SIAM, Philadelphia, PA, 1996.

[20] S. Engelke and C. Kanzow. Predictor-corrector smoothing methods for the solution of linear programs. *Hamburger Beiträge zur Angewandten Mathematik*, March 2000. Preprint 153.

[21] S. Engelke and C. Kanzow. Predictor-corrector smoothing methods for linear programs with a more flexible update of the smoothing parameter. *Hamburger Beiträge zur Angewandten Mathematik*, January 2001. Preprint 162.

[22] S. Engelke and C. Kanzow. Improved smoothing-type methods for the solution of linear programs. *Numerische Mathematik*, to appear.

[23] S. Engelke and C. Kanzow. On the solution of linear programs by Jacobian smoothing methods. *Annals of Operations Research*, Optimization and Numerical Algebra, to appear.

[24] A. Fischer. A special Newton-type optimization method. *Optimization*, 24:269–284, 1992.

[25] C. Geiger and C. Kanzow. *Numerische Verfahren zur Lösung unrestringierter Optimierungsaufgaben*. Springer Verlag, Berlin, Heidelberg, New York, 1999.

[26] A. George and J. W-H. Liu. *Computer Solution of Large Sparse Positive Definite Systems*. Prentice-Hall, Eaglewood Cliffs, New Jersey, 1981.

[27] A. George and J. W-H. Liu. The evolution of the minimum degree ordering. *SIAM Review*, 31:1–19, 1989.

[28] A. J. Goldman and A. W. Tucker. Theory of linear programming. In *Linear Equalities and Related Systems*, pages 53–97. Princeton University Press, Princeton, N. J., 1956.

[29] K. Hotta and A. Yoshise. Global convergence of a class of non-interior-point algorithms using Chen-Harker-Kanzow-Smale functions for non-linear complementarity problems. *Mathematical Programming*, 86:105–133, 1999.

[30] H. Jiang. Smoothed Fischer-Burmeister equation methods for the complementarity problem. Technical report, Department of Mathematics, University of Melbourne, Melbourne, Australia, June 1997.

[31] C. Kanzow. *C-Funktionen und ihre Anwendungen auf restringierte Optimierungsaufgaben und Komplementaritätsprobleme*. PhD thesis, Universität Hamburg, Hamburg, April 1995.

[32] C. Kanzow. Some non-interior continuation methods for linear complementarity problems. *SIAM Journal on Matrix Analysis and Applications*, 17:851–868, 1996.

[33] C. Kanzow. A new approach to continuation methods for complementarity problems with uniform P-functions. *Operations Research Letters*, 20:85–92, 1997.

[34] C. Kanzow. Interior-point methods for linear programs. Lecture notes, University of Hamburg, 1999.

[35] C. Kanzow and H. Pieper. Jacobian smoothing methods for non-linear complementarity problems. *SIAM Journal on Optimization*, 9:342–373, 1999.

[36] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[37] L. G. Khachiyan. A polynomial algorithm in linear programming. *Soviet Mathematics Doklady*, 20:191–194, 1979.

[38] V. Klee and G. J. Minty. How good is the Simplex algorithm? In O. Shisha, editor, *Inequalities*, pages 159–175. Academic Press, New Yory, N.Y., 1972.

[39] J. W.-H. Liu. Modification of the minimum degree algorithm by multiple elimination. *ACM Transactions on Mathematical Software*, 11:141–153, 1985.

[40] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a primal-dual interior-point method for linear programming. *Linear Algebra and its Applications*, 152:191–222, 1991.

[41] I. J. Lustig, R. E. Marsten, and D. F. Shanno. On implementing Mehrotra's predictor-corrector interior-point menthod for linear programming. *SIAM Journal on Optimization*, 2(3):435–449, August 1992.

[42] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior Point and Related Methods*, pages 131–158. Springer-Verlag, 1989.

[43] S. Mehrotra. On the implementation of a primal-dual interior-point method. *SIAM Journal on Optimization*, 2:575–601, November 1992.

[44] H. D. Mittelmann. Benchmark of some PD interior point solvers (and Soplex). *ftp://plato.la.asu.edu/pub/lpbench.txt*, May 18, 2000.

[45] R. D. C. Monteiro, I. Adler, and M. G. C. Resende. A polynomial-time primal-dual affine scaling algorithm for linear and convex quadratic programming and its power series extension. *Mathematics of Operations Research*, 15:191–214, 1990.

[46] J. J. Moré and D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing*, 4:553–572, 1983.

[47] E. Ng and B. W. Peyton. Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM Journal On Scientific Computing*, 14:1034–1056, 1993.

[48] J.-S. Pang. Newtons's method for B-differentiable equations. *Mathematics of Operations Research*, 15:311–341, 1990.

[49] J.-S. Pang. A b-differentiable equation-based, globally and locally quadratically convergent algorigthm for non-linear programs, complementarity and variational inequality problems. *Mathematical Programming*, 51:101–131, 1991.

[50] L. Qi and D. Sun. Globally linearly, and globally and locally superlinearly convergent versions of the Hotta-Yoshise non-interior-point algorithm for nonlinear complementarity problems. Technical report, School of Mathematics, The University of New South Wales, Sidney 2052, Australia, May 1997.

[51] S. M. Robinson. Some continuity properties of polyhedral multifunctions. *Mathematical Programming Study*, 14:206–214, 1981.

[52] W. Rudin. *Priniciples of Mathematical Analysis.* McGraw-Hill, Tokyo, Auckland, 3rd edition, 1976.

[53] S. Smale. Algorithms for solving equations. In *Proceedings of the International Congress of Mathematicians*, pages 172–195, Providence, 1987. AMS.

[54] P. Tseng. Analysis of a non-interior continuation method based on Chen-Mangasarian smoothing functions for complementarity problems. In Masao Fukushima and S. Qi, editors, *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, pages 381–404. Kluwer Academic Press, 1998.

[55] P. Tseng. Error bounds and superlinear convergence analysis of some Newton-type methods in optimization. In G. Di PIllo and F. Giannessi, editors, *Non-Linear Optimization and Related Topics*. Kluwer Academic Publishers, to appear.

[56] S. J. Wright. *Primal-Dual Interior-Point Methods.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[57] Y. Zhang. Solving large-scale linear programs by interior-point methods under the MATLAB environment. *Optimization Methods and Software*, 10:1–31, 1998.

[58] Y. Zhang. User's guide to LIPSOL: Linear programming interior-point solver. *Optimization Methods and Software*, 10 and 12:385–396, 1999.

Abstract/Zusammenfassung

Abstract

This thesis presents a new approach to the numerical solution of linear optimization problems, ranging from a simple Jacobian smoothing method to complex multi-step predictor-corrector algorithms.

All presented algorithms employ a reformulation of the linear program's optimality conditions based on one of several NCP-functions. The resulting non-linear system of equations is then solved by a Newton-type method.

In order to solve the system of equations common smoothing methods perturb both sides of the Newton equation, whereas Jacobian smoothing methods only apply the smoothing technique to the left-hand side of the system. This last approach exhibits nice theoretical properties, chief among those is the locally quadratic rate of convergence.

The proposed globally convergent smoothing methods are used as a basis upon which more advanced predictor-corrector algorithms are build which combine both smoothing and Jacobian smoothing ideas. The resulting methods feature a Jacobian smoothing predictor step and a regular smoothing corrector step. In this context the smoothing step supplies the global convergence properties, while the addition of the Jacobian smoothing step yields locally quadratic convergent methods.

The performance of this type of algorithm is close to that of modern implementations of interior-point solvers.

Zusammenfassung

Diese Arbeit stellt einen neuen Zugang zur Lösung linearer Optimierungsaufgaben auf der Basis von Glättungs- und Jacobi-Glättungsverfahren vor. Es werden sowohl einfache Einschritt-Verfahren, als auch komplexere Prädiktor-Korrektor-Algorithmen dargestellt.

Alle vorgestellten Verfahren nutzen eine Umformulierung der Optimalitätsbedingungen eines linearen Programms in ein nichtlineares Gleichungssystem unter Zuhilfenahme nicht differenzierbarer NCP-Funktionen zur Lösung des Optimierungsproblems.

Um zur Lösung des aus dieser Umformulierung resultierenden nichtlinearen Gleichungssystems das Netwon-Verfahren anwenden zu können, werden von herkömmliche Glättungsverfahren beide Seiten des im Verfahren auftretenden linearen Gleichungsystems gestört. Im Gegensatz hierzu wird beim Jacobi-Glättungsansatz nur die linke Seite des zu lösenden linearen Gleichungssystems (die Jacobi-Matrix) geglättet. Dieser Ansatz besticht durch seine guten theoretischen Eigenschaften. Insbesondere ist das betrachtete Verfahren lokal quadratisch Konvergent. Die vorgestellten global konvergenten Glättungsverfahren dienen als Basis für eine Reihe von Prädiktor-Korrektor-Algorithmen, in denen der Glättungs- und der Jacobi-Glattungsansatz kombiniert werden.

Die Kombination von einfachen Glättungs- und Jacobi-Glättungsalgorithmen zu komplexeren Prädiktor-Korrektor-Verfahren wird derart durchgeführt, dass zunächst als Prädiktor-Schritt ein Jacobi-Glättungschritt durchgeführt wird, um dann als Korrektor-Schritt einen reinen Glättungsschritt anzuschließen. Hierbei zeichnet sich der Glättungsschritt für die globalen Konvergenzeigenschaften des Verfahrens verantwortlich, der Jacobi-Glättungsschritt hingegen für die lokal quadratisch Konvergenz des Algorithmus' verantwortlich ist.

Die mit dieser Verfahrensklasse erzeugten numerischen Resultate kommen denen moderner Implementationen Innerer-Punkte-Methoden sehr nahe.

Curriculum Vitae/Lebenslauf

Persönliche Daten

Name:	Stephan Engelke
Geburtstag:	28. September 1972
Geburtsort:	Hamburg

Schule, Studium

1979–1983:	Besuch der Grundschule *Am Weiher* in Hamburg.
1983–1989:	Besuch des Gymnasiums *Kaiser-Friedrich-Ufer* in Hamburg.
1989–1990:	Besuch der *Holtville High School* in Holtville, CA, USA. Abschluss mit dem Diploma of Graduation.
1990–1992:	Besuch des Gymnasiums *Kaiser-Friedrich-Ufer* in Hamburg. Abschluss mit der Allgemeinen Hochschulreife.
1992–1998:	Studium der Wirtschaftsmathematik an der Universität Hamburg. Abschluss mit dem Diplom.
1999–2001:	Promotion im Fach Mathematik an der Universität Hamburg.

Beruflicher Werdegang

1999–2001:	Wissenschaftlicher Mitarbeiter am Fachbereich Mathematik der Universität Hamburg.