The Use of Preconditioned Iterative Linear Solvers in Interior-Point Methods and Related Topics

A Thesis
Presented to
The Academic Faculty

by

Jerome W. O'Neal

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Industrial and Systems Engineering
Georgia Institute of Technology
August 2005

The Use of Preconditioned Iterative Linear Solvers in Interior-Point Methods and Related Topics

Approved by:

Date Approved: June 23, 2005

To my bride, my beloved, and my best friend,

Amy Elizabeth O'Neal

PREFACE

This thesis is a compilation of papers which I have written with my advisor, Professor Renato Monteiro, over the past two years. Other coauthors include Professor Takashi Tsuchiya, Professor Arkadi Nemirovski, and fellow Ph.D. student Zhaosong Lu.

The first portion of Chapter 2, discussing the maximum weight basis preconditioner, is joint work with my advisor and Takashi Tsuchiya published in [44]. The second portion of that chapter, discussing the application of the MWB preconditioner to LP, is given in a technical report published with my advisor in [42]. Chapter 3 is work which has been done with my advisor and Zhaosong Lu. The first portion of Chapter 3 is presented in a technical report in [33] which has been conditionally accepted for publication in SIAM's Journal on Optimization. The second portion of Chapter 3 is given in [34]. Finally, Chapter 4 details work done with my advisor and Arkadi Nemirovski; the results are published in the technical report given in [43].

ACKNOWLEDGEMENTS

It would truly be impossible to thank all of the many people who have so graciously assisted me throughout my years at Georgia Tech. Nevertheless, I would like to highlight those who have helped me in particularly important ways as I pursued my degree.

First and foremost, I would like to thank my advisor, Renato Monteiro. His excitement for the subject of interior-point methods has been truly evident throughout our work together. He has helped me, not only in expanding my knowledge of the subject, but also through his willingness to mentor and teach me along the way.

I would like to thank the members of my committee: William Green, Arkadi Nemirovski, Gary Parker, and Alex Shapiro. I appreciate their insights and assistance during the process of preparing my thesis. I would like to extend particular thanks to Gary Parker for his invaluable support and guidance throughout my years at Georgia Tech, and to Arkadi Nemirovski for the excitement and encouragement he gave me as we jointly pursued the research detailed in Chapter 4.

I would be remiss if I did not thank the other staff and faculty in the School of Industrial and Systems Engineering at Georgia Tech. The professors challenged me through numerous courses, seminars, and discussions, helping me to learn the intricacies of modern optimization. I appreciate their patience and willingness to challenge us throughout our development. The staff in ISyE have always been willing to assist the students with whatever questions or concerns we might have. Particular thanks is due to Pam Morrison, Mark Reese, Eric Mungai, and Mike Barnhill for their help on various types of issues through the years.

Some of the best memories I will have of Georgia Tech involve interactions with fellow Ph.D. students. I especially want to thank my office mate and coauthor Zhaosong Lu; I have truly enjoyed our time together as we worked, and sometimes struggled, with new and challenging research. Thanks also to Marcos Goycoolea and Daniel Espinoza for their friendship and their computer programming skills, and to Jim Luedtke and Sriram Subramanian for their technical expertise and friendship.

Last and most important, I would like to thank my wife Amy and my children, Mary Grace and Andrew. You truly make life worth living and provide a safe haven through the challenges of life. I look forward to the next steps in our life together.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Over the last 25 years, interior-point methods (IPMs) have emerged as a viable class of algorithms for solving various forms of conic optimization problems. Most IPMs use a modified Newton method to determine the search direction at each iteration. The system of equations corresponding to the modified Newton system can often be reduced to the so-called "normal equation," a system of equations whose matrix AD^2A^T is positive definite, yet often ill-conditioned. In this thesis, we first investigate the theoretical properties of the maximum weight basis (MWB) preconditioner, and show that when applied to a matrix of the form AD^2A^T, where $D \in \mathbb{R}^{n \times n}$ is positive definite and diagonal, the MWB preconditioner yields a preconditioned matrix whose condition number is uniformly bounded by a constant depending only on A. Next, we incorporate the results regarding the MWB preconditioner into infeasible, long-step, primal-dual, path-following algorithms for linear programming (LP) and convex quadratic programming (CQP). In both LP and CQP, we show that the number of iterative solver ("inner") iterations of the algorithms can be uniformly bounded by n and a condition number of A, while the algorithmic ("outer") iterations of the IPMs can be polynomially bounded by n and the logarithm of the desired accuracy. We also expand the scope of the LP and CQP algorithms to incorporate a family of preconditioners, of which MWB is a member, to determine an approximate solution to the normal equation.

For the remainder of the thesis, we develop a new preconditioning strategy for solving systems of equations whose associated matrix is positive definite but ill-conditioned. Our so-called "adaptive preconditioning" strategy allows one to change the preconditioner during the course of the conjugate gradient (CG) algorithm by post-multiplying the current preconditioner by a simple matrix, consisting of the identity matrix plus a rank-one update. Our resulting algorithm, the Adaptive Preconditioned CG (APCG) algorithm, is shown to have polynomial convergence properties. Numerical tests are conducted to compare a variant of the APCG algorithm with the CG algorithm on various matrices.

CHAPTER I

INTRODUCTION AND PREVIOUS WORK

1.1 Motivation

The purpose of this thesis is to explore the use of iterative linear solvers within interior-point methods (IPMs). The development of IPMs dates back to the 1960s, where different methods from nonlinear programming were proposed to tackle linear programming (LP) and related problems. The reasoning behind such methods stems from the fact that, at its core, the optimality conditions for LP, known also as the Karush-Kuhn-Tucker (KKT) conditions, can be written in the form $\{z : F(z) = 0, G(z) \geq 0\}$, where $F(\cdot)$ and $G(\cdot)$ are maps and z is a vector. Interior point methods are iterative algorithms generating a sequence of points $\{z^j\}$ lying in the interior of the set $\{z : G(z) \geq 0\}$ (hence the name), and which converge to a point z^* satisfying the KKT conditions.

An early drawback to the use of IPMs for LP was the fact that no theoretical convergence properties, in terms of the data of the problem, were known. (An example of an early IPM is Dikin's method; see [15].) Thus it was impossible to compare the complexity of an IPM with the Simplex Method, a well-known method for LP. The Simplex Method has been shown to be extremely efficient in practice, often requiring $\sim 3m$ iterations to achieve an optimal solution, where m is the number of linear constraints (see e.g. [13]). However, it is not known whether the Simplex Method is a polynomial-time algorithm in theory; indeed, it has been shown to be an exponential-time algorithm for most well-known pivoting rules, as shown via problems such as the Klee-Minty problem [27].

A breakthrough for the problem of complexity came in 1984, in a paper written by Karmarkar [24]. In this paper, he proved that an interior-point method for LP had provable polynomial convergence properties; his algorithm required $\mathcal{O}(n^{3.5} \log \epsilon^{-1})$ arithmetic operations to achieve an ϵ-solution to LP, where n is the length of the decision vector. More recent advances, most notably those by Renegar [50], Gonzaga [19], and Nesterov and Nemirovski

[47], have reduced the arithmetic complexity as low as $\mathcal{O}(n^3 \log \epsilon^{-1})$. (An even more recent advance by Anstreicher reduced the complexity even further, to $\mathcal{O}((n^3/\log n)\log \epsilon^{-1})$ arithmetic operations; see [4]. For a thorough discussion of IPM algorithms and their theoretical convergence properties, see [52].)

In addition to their polynomial convergence properties, another strength of IPMs lies in their ability to solve more complex classes of problems. Specifically, IPMs can be used to solve *conic programming* problems of the form

$$\min\{c^T x : Ax - b \in \mathbf{K}\}, \tag{1}$$

where $\mathbf{K}$ is a closed, convex, pointed cone with non-empty interior (see e.g. [7, 52]). The most well-known classes of conic problems for which IPMs have been developed are

1. LP: $\mathbf{K} = \mathbb{R}^n_+ = \{x \in \mathbb{R}^n : \ x_i \geq 0 \text{ for all } i = 1, ..., n\}$,
2. Second-order cone programming (SOCP): $\mathbf{K} = \mathbf{K_1} \times \cdots \times \mathbf{K_p}$, where each $\mathbf{K_i} = \left\{x \in \mathbb{R}^{n_i} : x_{n_i} \geq \sqrt{x_1^2 + \cdot + x_{n_i-1}^2}\right\}$, and
3. Semidefinite programming (SDP): $\mathbf{K} = \{X \in \mathbb{R}^{n\times n} : X = X^T, y^T X y \geq 0 \text{ for all } y \in \mathbb{R}^n\}$.

It it straightforward to show (see e.g. [7]) that through some simple changes of variables, LP $\subset$ SOCP $\subset$ SDP.

A class of problems which we will consider extensively in this thesis is the class of convex quadratic programming (CQP) problems. CQP is an extension of LP in that a quadratic term is added to the objective function. The CQP problem we will consider in this thesis takes the form

$$\min\left\{\frac{1}{2}x^T Q x + c^T x : \quad Ax = b, \quad x \geq 0\right\}, \tag{2}$$

where the matrix Q is positive semidefinite. It is easy to show the following:

Proposition 1.1.1 *LP $\subset$ CQP $\subset$ SOCP.*

Proof: The first inclusion follows by noting that the standard form for LP is precisely (2) with $Q = 0$. For the second inclusion, we use a method explained in [7]. Specifically, let a factorization $Q = VV^T$ be given. It is clear that (2) is equivalent to the problem

$$\min \left\{ c^T x + \frac{t}{2} : \quad Ax = b, \ x \geq 0, \ \|V^T x\|^2 \leq t \right\}.$$

Since

$$t = \left(\frac{t+1}{2}\right)^2 - \left(\frac{t-1}{2}\right)^2,$$

this is equivalent to

$$\min \left\{ c^T x + \frac{t}{2} : \quad Ax = b, \ x \geq 0, \ \|V^T x\|^2 + \left(\frac{t-1}{2}\right)^2 \leq \left(\frac{t+1}{2}\right)^2 \right\},$$

i.e.

$$\min \left\{ c^T x + \frac{t}{2} : \quad Ax = b, \ x \geq 0, \ \left\| \begin{pmatrix} V^T x \\ \frac{t-1}{2} \end{pmatrix} \right\| \leq \frac{t+1}{2} \right\}.$$

Through a simple substitution of variables $z_1 = V^T x$, $z_2 = (t-1)/2$, and $z_3 = (t+1)/2$, we see that the desired problem is an example of SOCP, as desired. ▪

Because of this result, all SOCP algorithms can solve the equivalent CQP problem. CQP also falls within another class of well-known problems, the so-called *complementary* problems. CQP is a specific case of a mixed linear complementary problem (mLCP) (see [67]). In the next section, we will turn to a class of IPM algorithms which can be used to solve CQPs.

1.2 IPMs for CQP

In this section, we consider a class of IPM algorithms which can be used to solve the CQP problem. We begin by noting that the primal and dual problems of CQP are

$$\min \quad \left\{ \frac{1}{2} x^T Q x + c^T x : \quad Ax = b, \ x \geq 0 \right\}, \tag{3}$$

$$\max \quad \left\{ -\frac{1}{2} \hat{x}^T Q \hat{x} + b^T y : \quad A^T y + s - Q\hat{x} = c, \ s \geq 0 \right\}, \tag{4}$$

where the data are the $n \times n$ positive semidefinite matrix Q, the matrix $A \in \mathbb{R}^{m \times n}$, and the vectors $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. A point $(x, \hat{x}, s, y) \in \mathbb{R}^n_+ \times \mathbb{R}^n \times \mathbb{R}^n_+ \times \mathbb{R}^m$ is optimal for (3) and (4) if and only if it satisfies the following KKT conditions:

$$\begin{aligned} Ax &= b, & (5)\\ A^Ty + s - Q\hat{x} &= c, & (6)\\ Xs &= 0, & (7)\\ Q(x-\hat{x}) &= 0, & (8)\end{aligned}$$

where X is a diagonal matrix having the vector x along its diagonal (i.e. $X = \mathrm{Diag}(x)$). From these conditions, it is clear that if x and $(\hat{x}, s, y)$ are optimal for (3) and (4), respectively, then (x, s, y) is optimal for (4) as well. As a result, most IPMs solve the problem (4) with $\hat{x}$ replaced by x, so that the data of the problem is $w := (x, s, y) \in \mathbb{R}^n_+ \times \mathbb{R}^n_+ \times \mathbb{R}^m$. Given this fact, if x and (x, s, y) are feasible solutions of (3) and (4), respectively, we have the following simple result relating the objective functions of (3) and (4).

Proposition 1.2.1 *Let $w \in \mathbb{R}^n_+ \times \mathbb{R}^n_+ \times \mathbb{R}^m$ be feasible for (3) and (4), respectively. Then, the difference between the objective functions is precisely $n\mu$, where $\mu := \mu(x, s) = x^Ts/n$.*

Proof: By using the linear relations in (3) and (4), it is straightforward to show that the difference between the objective functions is

$$x^TQx + c^Tx - b^Ty \;=\; x^TQx + c^Tx - (Ax)^Ty \;=\; x^T(c - A^Ty + Qx) \;=\; x^Ts,$$

as desired. ∎

The function $\mu(x, s)$ is known as the "duality gap" between the primal and dual objective function values.

IPMs for CQP develop a sequence of iterates $w^k := (x^k, s^k, y^k) \in \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} \times \mathbb{R}^m$, i.e. $(x^k, s^k) > 0$. It is clear that these points lie in the interior of the cone $\mathbf{K} = \{(x, s, y) : (x, s) \geq 0\}$. Given a current iterate $w \equiv w^k$ as above, most of these methods seek to determine a solution to the Newton system of equations

$$\begin{aligned} A\Delta x &= b - Ax \;=:\; -r_p, & (9)\\ A^T\Delta y + \Delta s - Q\Delta x &= c - A^Ty - s + Qx \;=:\; -r_d, & (10)\\ X\Delta s + S\Delta x &= -Xs + \sigma\mu e \;=:\; -r_{xs}, & (11)\end{aligned}$$

Table 1: Iteration Complexity Results for CQPs

Initial point	Neighborhood	Complexity
Feasible	Narrow	$\mathcal{O}(\sqrt{n}\log\epsilon^{-1})$
Feasible	Wide	$\mathcal{O}(n\log\epsilon^{-1})$
Infeasible	Narrow	$\mathcal{O}(n\log\epsilon^{-1})$
Infeasible	Wide	$\mathcal{O}(n^2\log\epsilon^{-1})$

where $S = \mathrm{Diag}(s)$, $\sigma \in [0,1]$ is a user-defined "centering parameter," and μ is the duality gap. One then determines an appropriate step length $\alpha \in (0,1]$ (chosen so that $w + \alpha\Delta w$ lies in the interior of $\mathbf{K}$), then updates $w \leftarrow w + \alpha\Delta w$.

When $\sigma = 0$, the system (9)–(11) reduces to the exact Newton system for determining the direction $\Delta w := (\Delta x, \Delta s, \Delta y)$ at w; in this case, the search direction is known as an "affine scaling" or "predictor" direction. In contrast, when $\sigma = 1$ the search direction allows one to approach the "central path" of the problem, where the central path is the set of points $w(\nu) := (x(\nu), s(\nu), y(\nu)) \in \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} \times \mathbb{R}^m$ which satisfy (5), (6), and $Xs = \nu e$, where e is a vector of all ones and $\nu > 0$; in this case, the search direction is known as a "centering" direction. IPMs require that all iterates w^k lie in some neighborhood of this central path. Two common neighborhoods are the "narrow" and "wide" neighborhoods, which are defined as

- Narrow: $\left\{w \in \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} \times \mathbb{R}^m : \|Xs - \mu e\| \leq \beta\mu\right\}$, where $\beta \in (0,1)$; and
- Wide: $\left\{w \in \mathbb{R}^n_{++} \times \mathbb{R}^n_{++} \times \mathbb{R}^m : Xs \geq (1-\gamma)\mu e\right\}$, where $\gamma \in (0,1)$.

The complexity results for CQP vary, depending on whether the initial point w^0 is feasible for the affine constraints and on the size of neighborhood allowed for the iterates w^k. Table 1 details the iteration complexity results which have been proven for various types of IPM algorithms. (Note that these algorithms must be multiplied by $\mathcal{O}(n^3)$ to obtain arithmetic complexity results.) The justification for this table comes from Chapter 8 of [67] and from [72].

The results in this chapter assumed that we could determine a search direction Δw satisfying (9)–(11). In Subsection 1.2.1, we detail two methods for determining this search direction.

1.2.1 Determining the Search Direction for CQP

In this subsection, we briefly discuss two methods for determining a search direction Δw satisfying (9)–(11). Throughout, we will assume that the current iterate (x, s, y) satisfies $(x, s) > 0$, and that the matrix A has full row rank. We begin by removing the variable Δs from the system of equations. By solving for Δs in (10), then substituting it into (11), we obtain

$$-D^{-2}\Delta x + A^T \Delta y \;=\; X^{-1} r_{xs} - r_d, \tag{12}$$

where $D := (Q + X^{-1}S)^{-1/2}$ is positive definite, hence invertible. Equivalently, we obtain the following system of equations, known as the *augmented system*, in Δx and Δy:

$$\begin{pmatrix} -D^{-2} & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} X^{-1} r_{xs} - r_d \\ -r_p \end{pmatrix}. \tag{13}$$

The matrix in this system is quasidefinite (see [64]), and hence has a unique solution which can be determined through various means, including Cholesky-like factorizations. Once Δx and Δy are determined through this means, Δs can be determined via equation (10).

We can reduce the system of equations (13) even further by eliminating the variable Δx. We do this by multiplying (12) by AD^2 and using equation (9) to obtain the so-called *normal equation*

$$AD^2 A^T \Delta y \;=\; -r_p + AD^2(X^{-1} r_{xs} - r_d). \tag{14}$$

We then determine Δx and Δs according to the formulas

$$\begin{aligned} \Delta x &= D^2(r_d + A^T \Delta y - X^{-1} r_{xs}), & (15) \\ \Delta s &= -r_d - A^T \Delta y + Q\Delta s.. & (16) \end{aligned}$$

It is clear that Δw satisfies (10) in view of (16). Moreover, the definition of D implies that

$$\begin{aligned} X\Delta s + S\Delta x &= X(-r_d - A^T \Delta y + Q\Delta x) + S\Delta x \\ &= -Xr_d - XA^T \Delta y + XD^{-2}\Delta x \;=\; -r_{xs}, \end{aligned}$$

and we observe that

$$A\Delta x \;=\; AD^2A^T\Delta y + AD^2(r_d - X^{-1}r_{xs}) \;=\; -r_p,$$

so equations (9) and (11) are also satisfied. We should note that for the specific case of LP, i.e. $Q = 0$, the normal equation matrix AD^2A^T has the property that $D^2 = XS^{-1}$ is diagonal.

Under the assumptions given at the beginning of this subsection, the matrix AD^2A^T in (14) is positive definite. As a result, numerous algorithms exist for determining a solution to (14). Generally, these algorithms can be classified into one of two forms: *direct* and *iterative* methods. Direct methods (such as Cholesky factorization and Gaussian elimination) create a factorization of the matrix to determine a solution to (14). These methods have been shown to work well on many classes of CQP problems; however, they can be expensive to compute, both in terms of CPU time ($\Theta(m^3)$ operations, where m is the length of the vector Δy), and memory ($\Theta(m^2)$ bytes), since a factorization of AD^2A^T must be computed to determine Δy.

In contrast, iterative methods for solving a system of the form (14) develop a sequence of points $\{u^j\}$ which converge to Δy. These methods can be significantly faster than direct methods on some problems; moreover, they do not require that a factorization of AD^2A^T be obtained, hence they require less memory than direct methods.

Iterative methods possess two significant drawbacks when compared with direct methods. The first is that they only obtain approximate solutions to $Hu = h$. If u is only an approximate solution to (14), it is easy to see that the resulting search direction Δw can only satisfy (9)–(11) only approximately. Next, and potentially even more significant, is that the convergence rate for iterative methods depends on the condition number of the matrix AD^2A^T. For degenerate CQP problems, the condition number of AD^2A^T tends to "blow up" as one approaches an optimal solution (see [32]). Thus, iterative methods can become increasingly slow and unstable when applied to the system (14) in interior-point methods. For a detailed discussion of iterative methods, see [25] and the references contained therein.

One particular iterative method, the so-called Conjugate Gradient (CG) method, is discussed repeatedly in this thesis. In the next section, we present the major theoretical results pertaining to this method.

1.3 The Conjugate Gradient Method

The CG method is an iterative method which determines an approximate solution to the system $Hu = h$, where H is a positive-definite matrix and h is a vector. A good introduction to the CG method can be found in [56]. In this section, we present the CG method and detail the main theoretical results which have been obtained for it.

The CG method is normally implemented with a "preconditioner" matrix Z, chosen so that the condition number of Z^THZ is significantly smaller than the condition number of H. In this case, the algorithm is known as the preconditioned CG (PCG) algorithm. The details of the algorithm are given below.

PCG Algorithm:

```
Start:  Given $H \succ 0$, $h \in \mathbb{R}^m$, an invertible matrix $Z \in \mathbb{R}^{m \times m}$, and $u_0 \in \mathbb{R}^m$.

  1. Set $g_0 = Hu_0 - h$, $d_0 = -ZZ^Tg_0$, and $\gamma_0 = \|Z^Tg_0\|^2$.

  2. For $i = 0, 1, \ldots$ do

       (a) $u_{i+1} = u_i + \alpha_i d_i$, where $\alpha_i = \gamma_i/(d_i^THd_i)$

       (b) $g_{i+1} = g_i + \alpha_i H d_i$

       (c) $\gamma_{i+1} = \|Z^Tg_{i+1}\|^2$

       (d) $d_{i+1} = -ZZ^Tg_{i+1} + \beta_{i+1}d_i$, where $\beta_{i+1} = \gamma_{i+1}/\gamma_i$

     end (for).
```

Before presenting the main theoretical results associated with the PCG algorithm, we define the following notation.

$$\begin{aligned} \widehat{H} &:= Z^T H Z, && (17) \\ \hat{g}_i &:= Z^T g_i, && (18) \\ \widehat{S}_i &:= \operatorname{span}\{\hat{g}_0, \ldots, \widehat{A}^i \hat{g}_0\}, && (19) \\ S_i &:= Z\widehat{S}_i = \{Zv : v \in \widehat{S}_i\}. && (20) \end{aligned}$$

One well-known property of the PCG method is that the sequence of points $\{\hat{u}_i\}$ defined as $\hat{u}_i := Z^{-1}u_i$ is the sequence generated by the standard CG algorithm applied to the system $\widehat{H}\hat{u} = \hat{h}$, where $\widehat{H}$ is defined in (17) and $\hat{h} := Z^T h$. Moreover, the gradient of the energy function $\Phi_{\widehat{H}}(u) := (u - u^*)^T \widehat{H}(u - u^*)$ associated with this system at $\hat{u}_i$ is equal to $\hat{g}_i$ as defined in (18).

The following proposition follows from the above observations and the properties of the standard CG algorithm:

Proposition 1.3.1 *Each step i of the PCG algorithm possesses the following properties:*

(a) $\widehat{S}_i = \operatorname{span}\{\hat{g}_0, \ldots, \hat{g}_i\}$*;*

(b) $\hat{g}_i^T \hat{g}_j = 0$ *for all* $i < j$*;*

(c) $\hat{g}_i^T \widehat{H} \hat{g}_j = 0$ *for all* $i \leq j - 2$*; and*

(d) $u_i = \operatorname{argmin}\{\Phi_H(u) : u \in u_0 + S_{i-1}\}$*.*

Proof: See e.g. pages 295-7 of [62]. ▪

From the above results, it is clear that under exact arithmetic, the PCG algorithm terminates in at most m iterations, where $H \in \mathbb{R}^{m \times m}$. However, the results in Proposition 1.3.1 often fail to hold under finite-precision arithmetic. Indeed, the properties of the PCG algorithm depend heavily on property (b) above. Under finite precision arithmetic, the gradients $\hat{g}_j$ often fail to lose their orthogonality, and may even become linearly dependent. For a detailed discussion of the effects of finite-precision arithmetic on the PCG method, see [20].

Another error bound for the PCG method is based on the application of Chebyshev polynomials (see [56] for an introduction to Chebyshev polynomials). The bound states that, at each iteration of the PCG method, we have

$$\Phi_H(u^j) \ \leq \ 4 \left(\frac{\sqrt{\kappa(Z^T H Z)} - 1}{\sqrt{\kappa(Z^T H z)} + 1} \right)^{2j} \Phi_H(u^0).$$

Since $(t-1)/(t+1) = 1 - 2/(t+1)$ and $1 + \omega \leq e^{\omega}$ for all $\omega \in \mathbb{R}$, it is easy to see that

$$\Phi_H(u^j) \ \leq \ 4 \exp \left\{ \frac{-4j}{\sqrt{Z^T H Z} + 1} \right\} \Phi_H(u^0),$$

and hence an ϵ-solution satisfying $\Phi_H(u^j) \leq \epsilon \Phi_H(u^0)$ can be obtained in at most

$$\mathcal{O}(\sqrt{\kappa(Z^T H Z)} \log \epsilon^{-1}) \tag{21}$$

iterations. It is this bound which we will employ throughout this thesis, in part because bounds similar to this one appear to hold under finite-precision arithmetic (see [20]).

One drawback to the bound (21) is that it is not a polynomial-time bound. Indeed, the condition number of a matrix is well-known to be exponential in the size of the matrix (see [55]). In Chapter 4, we will present a new iterative method which is based on the CG algorithm, but which possesses polynomial convergence properties.

In the next section, we present the main results of our thesis. The results detail the use of iterative methods, including CG, within interior point algorithms for LP and CQP. In addition, the results offer a new approach, based on the CG algorithm, for determining an ϵ-solution to the system $Hu = h$.

1.4 Major Results of the Thesis

In this section we detail the major results of our thesis. The results can be divided into two areas. First, we present new theoretical complexity results for IPMs for LP and CQP, where the search directions are computed approximately by means of appropriately preconditioned iterative linear solvers. The second set of results pertain to a new iterative method for solving a system of equations whose matrix is positive definite.

1.4.1 New IPM Complexity Results for LP and CQP

In Chapters 2 and 3, we present new algorithmic approaches for LP and CQP whose search directions are computed by means of an iterative linear solver. We begin by discussing a well-known preconditioner, the so-called *maximum weight basis* (MWB) preconditioner. The MWB B is determined from a vector $d > 0$ and a corresponding matrix A of full row rank. Given this pair (A, d), the matrix B is formed from columns of A, giving higher priority to columns corresponding to larger elements of d; we then form the MWB preconditioner T according to the formula $T := D_{\mathcal{B}}^{-1} B^{-1}$. In Section 2.2, we show that the preconditioned normal matrix $TAD^2A^TT^T$ has a condition number which is uniformly bounded by m and a condition number of A. It is important to note that our bound given in Section 2.2 does not depend on the values of the diagonal matrix D.

In the remainder of Chapter 2, we use the MWB preconditioner to precondition the normal equation in an inexact, long-step, path-following, primal-dual IPM algorithm for LP. We show that the iterative linear solver, preconditioned with the MWB preconditioner, can obtain a suitable approximate solution to the normal equation within a uniformly bounded number of iterations. Since the iterative solver determines only an approximate solution to the normal equation, we show one method to distribute the error in the search direction so as to ensure polynomial convergence in the number of outer iterations. As we will show, the number of outer iterations required by our algorithm is $\mathcal{O}(n^2 \log \epsilon^{-1})$, the same as for the exact methods discussed in Subsection 1.2.1.

In Chapter 3, we extend the results in Chapter 2 from LP to CQP. The extension from LP to CQP is not immediately apparent at first, since the MWB preconditioner used in LP requires that the matrix D^2 be diagonal, which does not necessarily hold for CQP. In Section 3.2, we remedy this difficulty by presenting a new equation, the so-called *augmented normal equation* (ANE), for determining a portion of the search direction in CQP. The ANE is constructed in such a manner as to be able to use the MWB preconditioner described in Subsection 2.2. Again, we show that the number of iterative solver iterations can be uniformly bounded, while the iterations of the IPM are bounded by $\mathcal{O}(n^2 \log \epsilon^{-1})$ as before. In Section 3.6, we extend the results even further to allow for a class of preconditioners to be used. We detail the results for two of the preconditioners in this class, namely the MWB preconditioner and the partial update method preconditioner. The iterative solver iteration bounds that we obtain for the MWB preconditioner are exactly the same as the bound given in Section 3.3, while the bound for the partial update method is within a logarithmic factor of bounds given in [4] and [47]. Moreover, the method presented in Section 3.8 gives a stopping criterion for the iterative solver which may be checked, allowing us to stop the iterative solver once a suitable approximate solution has been reached. (In contrast, the methods given in [4] and [47] require that a prescribed number of iterative solver iterations must be performed at each step of the IPM to ensure convergence; see Section 3.8 for details.)

1.4.2 The Adaptive PCG (APCG) Method

In Chapter 4, we turn our attention to the solution of a system of equations $Hu = h$ whose matrix H is positive definite but may be extremely ill-conditioned. We present a new approach which allows one to change the preconditioner in the PCG method throughout the course of the algorithm to speed convergence. We introduce this adaptive preconditioning procedure by applying it first to the steepest descent algorithm. Then, the same techniques are extended to the PCG algorithm, and convergence to an ϵ-solution in $\mathcal{O}(\log\det(A) + \sqrt{n}\log\epsilon^{-1})$ iterations is proven, where $\det(A)$ is the determinant of the matrix.

CHAPTER II

THE MAXIMUM WEIGHT BASIS PRECONDITIONER AND ITS USE IN AN INEXACT LP ALGORITHM

2.1 The Maximum Weight Basis: Introduction

Consider the linear programming (LP) problem $\min\{c^T x : Ax = b,\ x \geq 0\}$, where $A \in \mathbb{R}^{m\times n}$ has full row rank. Interior-point methods for solving this problem require that systems of linear equations of the form $AD^2A^T\Delta y = r$, where D is a positive diagonal matrix, be solved at every iteration. It often occurs that the "normal" matrix AD^2A^T, while positive definite, becomes increasingly ill-conditioned as one approaches optimality. In fact, it has been proven (e.g., see Kovacevic and Asic [32]) that for degenerate LP problems, the condition number of the normal matrix goes to infinity. Because of the ill-conditioned nature of AD^2A^T, many methods for solving the system $AD^2A^T\Delta y = r$ become increasingly unstable. The problem becomes even more serious when conjugate gradient methods are used to solve this linear system. Hence, the development of suitable preconditioners which keep the condition number of the coefficient matrix of the scaled system under control is of paramount importance. We should note, however, that in practice, the ill-conditioning of AD^2A^T generally does not cause difficulty when the system $AD^2A^T\Delta y = r$ is solved using a backward-stable direct solver (see e.g. [66, 70] and references contained therein).

In the first section of this chapter, we analyze a known preconditioner for the normal matrix AD^2A^T, the so-called maximum weight basis (MWB) preconditioner, that has been proposed by Vaidya [63] in the context of the minimum cost network flow problem and subsequently by Oliveira and Sorensen [48] for general LP problems. The preconditioning consists of pre- and post-multiplying AD^2A^T by $D_B^{-1}B^{-1}$ and its transpose, respectively, where B is a suitable basis of A and D_B is the corresponding diagonal submatrix of D. Roughly speaking, B is constructed in such a way that columns of A corresponding to larger diagonal elements of D have higher priority to be in B. Our main result is that such preconditioner yields coefficient matrices with uniformly bounded condition number regardless of the value of the diagonal elements of D. In the context of interior-point methods, this means that the condition number of the preconditioned normal matrix has a bound that does not depend on the current iterate, regardless of whether it is well-centered. We also show that when applied to network flow-based LPs, the condition number of the preconditioned normal matrix is bounded by $m(n-m+1)$, where m and n are the number of nodes and arcs of the network.

2.2 The Maximum Weight Basis Preconditioner

In this section, we describe the MWB preconditioner and establish its main properties.

We first describe a procedure, which given an $n \times n$ diagonal matrix $D \in \mathcal{D}_{++}$, finds a suitable basis of a full row rank matrix $A \in \mathbb{R}^{m \times n}$ obtained by giving higher priority to the columns of A with larger corresponding diagonal elements of D. The use of this basis as a way to obtain preconditioners of matrices of the form AD^2A^T was originally proposed by Vaidya [63] in the context of minimum cost network flow problems and was subsequently extended by Oliveira and Sorensen [48] to the context of general LP problems. Note that when this procedure is used in the context of the minimum cost network flow interior point methods, it produces a maximum spanning tree for the network whose arc weights are given by the diagonal elements of D.

Algorithm for determining MWB B: Let a pair $(A, d) \in \mathbb{R}^{m \times n} \times \mathbb{R}^n_{++}$ be given such that $\operatorname{rank}(A) = m$. Then:

```
1. Order the elements of $d$ such that $d_1 \geq ... \geq d_n$; order the columns of $A$ accordingly.
2. Let $\mathcal{B} = \emptyset$ and set $l = 1$.
3. While $|\mathcal{B}| < m$, do
    (a) If $A_l$ is linearly independent of $\{A_j : j \in \mathcal{B}\}$, set $\mathcal{B} \leftarrow \mathcal{B} \cup \{l\}$.
    (b) $l \leftarrow l + 1$.
```

4. Return to the original ordering of A, and determine the set $\mathcal{B}$ according to this ordering.

5. Let $\mathcal{N} = \{1, ..., n\} \setminus \mathcal{B}$, $B = A_{\mathcal{B}}$, and $N = A_{\mathcal{N}}$.

We will refer to a basis B produced by the above scheme as a MWB associated with the pair $(A, d) \in \mathbb{R}^{m \times n} \times \mathbb{R}^n_{++}$. We begin with a technical, but important lemma.

Lemma 2.2.1 *Let $(A, d) \in \mathbb{R}^{m \times n} \times \mathbb{R}^n_{++}$ be given such that rank$(A) = m$. Suppose that B is a MWB associated with the pair (A, d) and define $D \equiv \mathrm{Diag}(d)$ and $D_{\mathcal{B}} \equiv \mathrm{Diag}(d_{\mathcal{B}})$. Then, for every $j = 1, \ldots, n$:*

$$d_j \, \|D_{\mathcal{B}}^{-1} B^{-1} A_j\| \;\leq\; \|B^{-1} A_j\|. \tag{22}$$

As a consequence, we have:

$$\|D_{\mathcal{B}}^{-1} B^{-1} A D\| \;\leq\; \|D_{\mathcal{B}}^{-1} B^{-1} A D\|_F \;\leq\; \|B^{-1} A\|_F. \tag{23}$$

Proof: We first prove (22). For every $j \in \mathcal{B}$, both sides of (22) are the same and hence (22) holds as equality in this case. Assume now that $j \in \mathcal{N}$. We consider the following two distinct cases: i) A_j was not considered to enter the basis B in step 3 of the above scheme, and ii) A_j was a candidate to enter the basis but it failed to make it. Consider first case i). In this case, we have $d_j \leq \min(d_{\mathcal{B}})$ since the d_k's are arranged in nonincreasing order at step 1 of the above scheme. Thus, we have

$$d_j \, \|D_{\mathcal{B}}^{-1} B^{-1} A_j\| \;\leq\; \frac{d_j}{\min(d_{\mathcal{B}})} \|B^{-1} A_j\| \;\leq\; \|B^{-1} A_j\|.$$

Consider now case ii). Suppose that A_j was a candidate to become the r-th column of B. Since it failed to enter B, it must be linearly dependent on the first $r - 1$ columns of B. Hence,

$$B^{-1} A_j \;=\; \begin{pmatrix} u \\ 0 \end{pmatrix},$$

for some $u \in \mathbb{R}^{r-1}$. Hence, using the fact that $d_{B_i} \geq d_j$ for every $i = 1, \ldots, r-1$, we conclude that

$$d_j \, \|D_{\mathcal{B}}^{-1} B^{-1} A_j\| \;\leq\; d_j \left(\sum_{i=1}^{r-1} \frac{u_i^2}{d_{B_i}^2} \right)^{1/2} \;\leq\; \left(\sum_{i=1}^{r-1} u_i^2 \right)^{1/2} \;=\; \|u\| \;=\; \|B^{-1} A_j\|.$$

We next prove (23). The first inequality of (23) is well-known. The second inequality follows from (22), the identity $\|R\|_F^2 = \sum_{j=1}^n \|R_j\|^2$ for every $R \in \mathbb{R}^{m \times n}$, and the fact that the j-th column of $D_{\mathcal{B}}^{-1} B^{-1} AD$ is $d_j D_{\mathcal{B}}^{-1} B^{-1} A_j$. ▪

Given a pair $(A, d) \in \mathbb{R}^{m \times n} \times \mathbb{R}^n_{++}$ such that $\text{rank}(A) = m$, we next consider a preconditioner for a system of equations of the form $AD^2A^Tp = r$, where $D \equiv \text{Diag}(d)$. Pre- and post-multiplying its coefficient matrix by an invertible matrix $T \in \mathbb{R}^{m \times m}$, we obtain the following equivalent system

$$T(AD^2A^T)T^T\tilde{p} \;=\; Tr,$$

where $\tilde{p} = T^{-T}p$. The following results give a suitable choice of T for which the condition number of the coefficient matrix of the above system is uniformly bounded as d varies over $\mathbb{R}^n_{++}$.

Lemma 2.2.2 *Let $(A, d) \in \mathbb{R}^{m \times n} \times \mathbb{R}^n_{++}$ be given such that rank$(A) = m$. Suppose that B is a MWB associated with the pair (A, d) and define $D \equiv \text{Diag}(d)$, $D_{\mathcal{B}} \equiv \text{Diag}(d_{\mathcal{B}})$ and $T \equiv D_{\mathcal{B}}^{-1} B^{-1}$. Then,*

$$\kappa(TAD^2A^TT^T) \;\leq\; \|B^{-1}A\|_F^2.$$

Proof: By Lemma 2.2.1, we have $\|TAD\| \leq \|B^{-1}A\|_F$. Hence,

$$\lambda_{\max}(TAD^2A^TT^T) \;=\; \|TAD\|^2 \;\leq\; \|B^{-1}A\|_F^2.$$

Moreover, since

$$T(ADA^T)T^T \;=\; D_{\mathcal{B}}^{-1} B^{-1}(BD_{\mathcal{B}}^2B^T + ND_{\mathcal{N}}^2N^T)B^{-T}D_{\mathcal{B}}^{-1} \;=\; I + WW^T,$$

where $W \equiv D_{\mathcal{B}}^{-1} B^{-1} ND_{\mathcal{N}}$ and $D_{\mathcal{N}} \equiv \text{Diag}(d_{\mathcal{N}})$, we conclude that

$$\lambda_{\min}(TAD^2A^TT^T) \;\geq\; 1.$$

Hence, the lemma follows. ∎

We will refer to the preconditioner T described in Lemma 2.2.2 as a MWB preconditioner. For the purpose of stating the next result, we now introduce some notation. Let us define

$$\varphi_A \ := \ \max\{\|B^{-1}A\|_F : B \text{ is a basis of } A\}. \tag{24}$$

The constant φ_A is related to the well-known condition number $\bar{\chi}_A$ (see [65]), defined as

$$\bar{\chi}_A \ := \ \sup\{\|A^T(AEA^T)^{-1}AE\| : E \in \text{Diag}(\mathbb{R}^n_{++})\}.$$

Specifically, $\varphi_A \ \leq \ m^{1/2}\bar{\chi}_A$, in view of the facts that $\|C\|_F \ \leq \ m^{1/2}\,\|C\|$ for any matrix $C \in \mathbb{R}^{m\times n}$ and, as shown in [60] and [65],

$$\bar{\chi}_A \ = \ \max\{\|B^{-1}A\| : B \text{ is a basis of } A\}.$$

From the lemma above, we arrive at our main result for the MWB preconditioner.

Theorem 2.2.3 *Let $(A,d) \in \mathbb{R}^{m\times n} \times \mathbb{R}^n_{++}$ be given such that rank$(A) = m$. Suppose that B is a MWB associated with the pair (A,d) and define $D \equiv \text{Diag}(d)$, $D_{\mathcal{B}} \equiv \text{Diag}(d_{\mathcal{B}})$ and $T \equiv D_{\mathcal{B}}^{-1}B^{-1}$. Then,*

$$\kappa(TAD^2A^TT^T) \ \leq \ \varphi_A^2 \ \leq \ m\,\bar{\chi}_A^2.$$

Another important consequence of Lemma 2.2.2 is the following result.

Theorem 2.2.4 *Let $A \in \mathbb{R}^{m\times n}$ denote the node–arc incidence matrix of a connected directed graph with one of its rows deleted. Suppose that B is a MWB associated with the pair (A,d), for some $d \in \mathbb{R}^n_{++}$. Letting $D \equiv \text{Diag}(d)$, $D_{\mathcal{B}} \equiv \text{Diag}(d_{\mathcal{B}})$ and $R \equiv D_{\mathcal{B}}^{-1}B^{-1}$, we have*

$$\kappa(TAD^2A^TT^T) \ \leq \ m(n-m+1).$$

Proof: Using the structure of A, it is easy to see that $\|B^{-1}A\|_F^2 \leq m + (n-m)m = m(n-m+1)$. The result now follows directly from Lemma 2.2.2. ∎

2.3 The Maximum Weight Basis: Concluding Remarks

As mentioned earlier, using a maximum weight basis preconditioner in the context of network flow problems yields a maximum spanning tree preconditioner first proposed by Vaidya [63]. In such a case, Judice et al. [22] have attempted to show that the condition number of the preconditioned matrix is bounded. However, their proof is incomplete, in that it deals only with three out of four possible cases, the neglected case being the most difficult and interesting one. Our proof does not attempt to correct theirs; rather, it is based on an entirely different approach. Moreover, our approach also holds for general LP problems in standard form, and shows that the derived bounds on the condition number of the preconditioned normal matrices $T(AD^2A^T)T^T$ hold for any $D \in \mathcal{D}_{++}$. In the context of interior-point methods, this means that the condition number of the preconditioned normal matrix has a bound that does not depend on the current iterate, regardless of whether it is well centered.

Certain computational issues arise from our analysis. When determining the maximum weight basis B, one must determine whether a set of columns in A is linearly independent. This process tends to be sensitive to roundoff errors. Once the set $\mathcal{B}$ is determined, the matrix T can be computed in a stable fashion, since the condition numbers for all possible bases B are uniformly bounded and multiplication by the diagonal matrix $D_{\mathcal{B}}^{-1}$ can be done in a componentwise manner.

The authors believe that the preconditioner studied in this section will be computationally effective only for some special types of LP problems. For example, the papers [22, 53] developed effective iterative interior-point methods for solving the minimum cost network flow problem based on maximum spanning tree preconditioners. Another class of LP problems for which iterative interior-point methods based on maximum weight basis preconditioners might be useful are those for which bases of A are sparse but the normal matrices AD^2A^T are dense; this situation generally arises in the context of LP problems for which n is much larger than m. Investigation of the many classes of LP problems which would benefit from the application of such methods is certainly an important area for future research.

2.4 Inexact LP using the MWB: Introduction

Consider the standard-form linear programming (LP) problem

$$\min\{c^T x : Ax = b,\, x \geq 0\} \tag{25}$$

which we refer to as the *primal* problem, and its associated *dual* problem

$$\max\{b^T y : A^T y + s = c,\, s \geq 0\}, \tag{26}$$

where the data consists of $(A, b, c) \in \mathbb{R}^{m\times n} \times \mathbb{R}^m \times \mathbb{R}^n$ and the primal-dual variable consists of $(x, s, y) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m$.

This section deals with interior-point (IP) algorithms for solving the pair of LP problems (25) and (26) whose search directions are computed by means of iterative linear solvers. We refer to such algorithms as *iterative IP methods*. An outer iteration of an iterative IP algorithm is similar to that of an exact IP method, except that the Newton search directions are computed approximately by means of an iterative linear solver. In this context, the iterations of the linear solver will be referred to as the *inner iterations* of the iterative IP method. We will consider an iterative version of the long-step primal-dual infeasible IP algorithm considered in [28, 72] and show that its total number of inner iterations is polynomially bounded by m, n and a certain condition number associated with A, while the number of outer iterations is bounded by $\mathcal{O}(n^2 \log \epsilon^{-1})$, where ϵ is a given relative accuracy level.

Given a current iterate (x, s, y) in a generic primal-dual IP algorithm, one usually solves for the Newton search direction $(\Delta x, \Delta s, \Delta y)$ using one of two well-known approaches. In the first approach, one solves an *augmented system* for $(\Delta x, \Delta y)$, which then immediately yields Δs. In the second one, one solves the *normal equation* system

$$AD^2 A^T \Delta y = p, \tag{27}$$

for some appropriate $p \in \mathbb{R}^m$, where $D^2 = S^{-1}X$. Our method is based on the second approach where we solve (27) using an iterative solver.

Most IP solvers using the normal equation usually solve (27) via a *direct solver* which computes a factorization of AD^2A^T to obtain Δy. The use of an iterative linear solver to solve (27) has two main potential advantages: (i) it can be significantly faster than a direct solver in some LP instances (see e.g. [48, 53]); and (ii) it can take complete advantage of the sparsity of the matrix A. However, iterative solvers possess two potential drawbacks when compared with direct solvers: (a) they solve (27) less accurately than direct methods; and (b) they may have slow convergence if the coefficient matrix AD^2A^T is ill-conditioned.

The effect of item (a) above is that the search direction $(\Delta x, \Delta s, \Delta y)$ can only satisfy the Newton system approximately, regardless of the choice of Δx and Δs. In our approach, we will choose these components so that the equations of the Newton system corresponding to primal and dual feasibility are satisfied exactly, while the equation corresponding to the centrality condition is violated. This way of choosing the search direction is crucial for us to establish that the number of outer iterations of our method is polynomially bounded (see Section 2.5.4).

Item (b) has been a significant problem for those wishing to use iterative methods to solve (27). It is well-known that in degenerate cases, the condition number of AD^2A^T "blows up" as we approach an optimal solution, even if our iterates (x, s, y) lie on the central path (see e.g. [32]). A cure to this problem is to use a preconditioner T so as to make the condition number of $TAD^2A^TT^T$ small. One such preconditioner was introduced by Resende and Veiga [53] in the context of the minimum cost network flow problem, and later generalized by Oliveira and Sorensen [48] for general LP problems. The proof that the above preconditioner makes the condition number of $TAD^2A^TT^T$ uniformly bounded regardless of the values of the diagonal elements of D was proved by Monteiro, O'Neal, and Tsuchiya [44]. In view of this nice property, we will use this preconditioner in our algorithm.

Global convergence analysis of algorithms using inexact search directions has been presented in several papers (see e.g. [17, 28, 31, 39]). Several authors have also used iterative linear solvers to compute an approximate Newton search direction (see e.g. [6, 31, 48, 49, 53]). In particular, Resende and Veiga [53] and Oliveira and Sorensen [48] used an iterative solver in conjunction with the above preconditioner to compute (27) inexactly for network-flow problems and general LP problems, respectively. Their computational results show that iterative IP methods can be extremely useful in practice. To our knowledge, though, no one to date has obtained strong theoretical arithmetic complexities for iterative IP methods.

The remainder of this chapter is organized as follows. Subsection 2.4.1 describes the terminology and notation used throughout the rest of this chapter. Section 2.5 gives the main results we have obtained for LP, and is divided into five parts. Section 2.5.1 describes an exact variant of an infeasible-interior-point algorithm on which the algorithm we study in this chapter is based. Section 2.5.2 discusses the preconditioner T mentioned above and gives some background results. Section 2.5.3 states the convergence results about a generic iterative method for solving (27). Section 2.5.4 gives our algorithm and main results, and Section 2.5.5 discusses the application of our algorithm to network flow problems. In Section 2.6, we prove the results stated in Section 2.5. Finally, some concluding remarks are presented in Section 2.7.

2.4.1 Terminology and Notation

Throughout the remainder of the chapter, upper-case Roman letters denote matrices, lower-case Roman letters denote vectors, and lower-case Greek letters denote scalars. For a matrix A, $A \in \mathbb{R}^{m \times n}$ means that A is an $m \times n$ matrix with real entries; while for a vector x, $x \in \mathbb{R}^n$ means that x is an n-dimensional real vector. More specifically, $x \in \mathbb{R}^n_+$ means that $x \in \mathbb{R}^n$ and $x_i \geq 0$ for all i, while $x \in \mathbb{R}^n_{++}$ means that $x \in \mathbb{R}^n$ and $x_i > 0$ for all i. Next, the vector $|v|$ is the vector whose ith component is $|v_i|$. Also, given a vector v, $\text{Diag}(v)$ is a diagonal matrix whose diagonal elements are the elements of v, i.e. $(\text{Diag}(v))_{ii} = v_i$ for all i.

Five matrices bear special mention: the matrices X, S, D, ΔX, and ΔS all in $\mathbb{R}^{n \times n}$. These matrices are diagonal matrices having the elements of the vectors x, s, d, Δx, and Δs respectively, along their diagonals (i.e. $X = \text{Diag}(x)$, etc.). The symbol 0 will be used to denote a scalar, vector, or matrix of all zeroes; its dimensions should be clear from the context. Also, the vector e is the vector of all 1's, whose dimension is implied by the context.

If a matrix $W \in \mathbb{R}^{m \times m}$ is symmetric ($W = W^T$) and positive definite (has all positive eigenvalues), we write $W \succ 0$. The condition number of W, denoted $\kappa(W)$, is the ratio between its maximum eigenvalue divided by its minimum eigenvalue. We will denote sets by upper-case script Roman letters (e.g. $\mathcal{B}$, $\mathcal{N}$). For a set $\mathcal{B}$, we denote the cardinality of the set by $|\mathcal{B}|$. Given a matrix $A \in \mathbb{R}^{m\times n}$ and a set $\mathcal{B} \subseteq \{1,\dots,n\}$, the matrix $A_{\mathcal{B}}$ is the submatrix consisting of the columns $\{A_i : i \in \mathcal{B}\}$. Similarly, given a vector $v \in \mathbb{R}^n$ and a set $\mathcal{B} \subseteq \{1,\dots,n\}$, the vector $v_{\mathcal{B}}$ is the subvector consisting of the elements $\{v_i : i \in \mathcal{B}\}$.

We will use several different norms throughout the chapter. For a vector $z \in \mathbb{R}^n$, $\|z\| = \sqrt{z^T z}$ is the Euclidian norm, and $\|z\|_\infty = \max_{i=1,\dots,n} |z_i|$ is the "infinity norm". For a matrix $V \in \mathbb{R}^{m\times n}$, $\|V\|$ denotes the operator norm associated with the Euclidian norm: $\|V\| = \max_{z:\|z\|=1} \|Vz\|$. Finally, $\|V\|_F$ denotes the Frobenius norm: $\|V\|_F = (\sum_{i=1}^m \sum_{j=1}^n V_{ij}^2)^{1/2}$.

2.5 Inexact LP using the MWB: Main Results

The main results of the chapter are stated in this section, which is divided into five parts. Section 2.5.1 gives some background and motivation for the method proposed in the chapter. Section 2.5.2 describes the maximum weight basis preconditioner. Section 2.5.3 considers a generic iterative linear solver and derives an upper bound on the number of iterations for it to obtain a reasonably accurate solution of (27). Section 2.5.4 describes how the overall search direction is obtained and states the main algorithm. Finally, Section 2.5.5 describes a tighter complexity for the algorithm when applied to network flow problems.

2.5.1 Preliminaries and Motivation

In this subsection, we discuss a well-known infeasible primal-dual long-step IP algorithm (see for example [28] and [72]) which will serve as the basis for the iterative IP method proposed in this chapter. We also state the complexity results which have been obtained for this algorithm. For the sake of concreteness, we have chosen to work with one specific primal-dual IP method. We note, however, that our analysis applies to other long-step variants as well as to short-step IP methods.

As stated in the introduction, we will be working with the pair of LPs (25) and (26). Letting $\mathcal{S}$ denote the set of primal-dual optimal solutions $(x, s, y) \in \mathbb{R}^{2n} \times \mathbb{R}^m$ of (25) and

(26), it is well-known that $\mathcal{S}$ consists of the triples $(x, s, y) \in \mathbb{R}^{2n} \times \mathbb{R}^m$ satisfying

$$\begin{aligned} Ax &= b, \quad x \geq 0 & (28)\\ A^T y + s &= c, \quad s \geq 0 & (29)\\ Xs &= 0. & (30) \end{aligned}$$

Throughout the chapter, we will make the following assumptions:

Assumption 1 *A has full row rank.*

Assumption 2 *The set $\mathcal{S}$ is nonempty.*

For a point $(x, s, y) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$, let us define

$$\begin{aligned} \mu = \mu(x,s) &:= x^T s/n,\\ r_p = r_p(x) &:= Ax - b,\\ r_d = r_d(s,y) &:= A^T y + s - c,\\ r = r(x,s,y) &:= (r_p, r_d). \end{aligned}$$

Moreover, given $\gamma \in (0,1)$ and an initial point $(x^0, s^0, y^0) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^m$, we define the following neighborhood of the central path:

$$\mathcal{N}(\gamma) := \left\{ (x,s,y) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^m : Xs \geq (1-\gamma)\mu e, \ \frac{\|r\|}{\|r^0\|} \leq \frac{\mu}{\mu_0} \right\}, \tag{31}$$

where $r^0 := r(x^0, s^0, y^0)$ and $\mu_0 := \mu(x^0, s^0)$. Here, we use the convention that $\nu/0$ is equal to 0 if $\nu = 0$ and ∞ if ν is positive.

The infeasible primal-dual algorithm which will serve as the basis for our iterative IP method is as follows:

Algorithm IIP

1. **Start:** Let $\epsilon > 0$, $\gamma \in (0,1)$, $(x^0, s^0, y^0) \in \mathcal{N}(\gamma)$ and $0 < \underline{\sigma} < \overline{\sigma} < 1$ be given. Set $k = 0$.

2. **While** $\mu_k := \mu(x^k, s^k) > \epsilon$ **do**

```
    (a) Let $(x,s,y) := (x^k, s^k, y^k)$ and $w := (x,s,y)$; choose $\sigma \in [\underline{\sigma}, \overline{\sigma}]$

    (b) Let $\Delta w := (\Delta x, \Delta s, \Delta y)$ denote the solution of the linear system
```

$$X\Delta s + S\Delta x = -Xs + \sigma\mu e, \tag{32}$$

$$A\Delta x = -r_p, \tag{33}$$

$$A^T\Delta y + \Delta s = -r_d. \tag{34}$$

```
    (c) Let
```

$$\tilde{\alpha} = \mathrm{argmax}\left\{\alpha \in [0,1] : w + \alpha'\Delta w \in \mathcal{N}(\gamma),\ \forall \alpha' \in [0,\alpha]\right\}.$$

```
    (d) Let $\bar{\alpha} = \mathrm{argmin}\{(x+\alpha\Delta x)^T(s+\alpha\Delta s) : \alpha \in [0,\tilde{\alpha}]\}$.

    (e) Let $(x^{k+1}, s^{k+1}, y^{k+1}) = w + \bar{\alpha}\,\Delta w$, and set $k \leftarrow k+1$.

**end** (while)
```

The main complexity result for Algorithm IIP (see for example [28] and [72]) is as follows:

Theorem 2.5.1 *Assume that the constants γ, $\underline{\sigma}$ and $\overline{\sigma}$ are such that*

$$\max\left\{\gamma^{-1}, (1-\gamma)^{-1}, \underline{\sigma}^{-1}, (1-\overline{\sigma})^{-1}\right\} = \mathcal{O}(1),$$

and that the initial point $(x^0, s^0, y^0) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfies $(x^0, s^0) \geq (\bar{x}, \bar{s})$ for some $(\bar{x}, \bar{s}, \bar{y}) \in \mathcal{S}$. Then, Algorithm IIP finds an iterate $(x^k, s^k, y^k) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfying $\mu_k \leq \epsilon\mu_0$ and $\|r^k\| \leq \epsilon\|r^0\|$ within $\mathcal{O}\left(n^2\log(1/\epsilon)\right)$ iterations.

One way of computing the solution $(\Delta x, \Delta s, \Delta y)$ of (32)-(34) is to first solve for Δy using the following equation, known as the *normal equation*:

$$AD^2A^T\Delta y = -r_p - \sigma\mu AS^{-1}e + Ax - AD^2r_d, \tag{35}$$

where $D^2 = S^{-1}X$, and then compute Δs and Δx using the following formulae:

$$\Delta s = -r_d - A^T\Delta y, \tag{36}$$

$$\Delta x = -x + \sigma\mu S^{-1}e - D^2\Delta s. \tag{37}$$

Theorem 2.5.1 assumes that we can solve (32)-(34), and hence (35), exactly. Normally, the exact solution of (35) is obtained via a Cholesky factorization of AD^2A^T. Instead of this, we would like to use an iterative solver to obtain an approximate solution of (35). However, as mentioned in the introduction, the condition number of AD^2A^T may “blow up” as we approach an optimal solution, making the use of iterative methods for solving (35) undesirable. One cure is to use a preconditioner T such that the condition number $\kappa(TAD^2A^TT^T)$ remains bounded and hopefully small. One such preconditioner will be described in the next subsection, and will play an important role in our main algorithm described in Section 2.5.4.

2.5.2 Preconditioner

In this subsection, we will describe the preconditioner T that we will use to solve (35), and we will state the main results for this preconditioner, as given in [44].

Our proposed approach consists of solving the preconditioned system of linear equations:

$$Wz = q, \tag{38}$$

where

$$\begin{aligned} W &:= TAD^2A^TT^T, && (39) \\ q &:= -Tr_p - \sigma\mu TAS^{-1}e + TAx - TAD^2r_d, && (40) \end{aligned}$$

and T is the preconditioner matrix (which we refer to as the *maximum weight basis* preconditioner) determined by the following algorithm:

Maximum Weight Basis Algorithm

```
Start: Given $A \in \mathbb{R}^{m\times n}$ and $d \in \mathbb{R}^n_{++}$,
  1. Order the elements of $d$ so that $d_1 \geq \ldots \geq d_n$; order the columns of $A$ accordingly.
  2. Let $\mathcal{B} = \emptyset$, $l = 1$.
  3. While $|\mathcal{B}| < m$ do
     (a) If $A_l$ is linearly independent of $\{A_i : i \in \mathcal{B}\}$, set $\mathcal{B} \leftarrow \mathcal{B} \cup \{l\}$.
```

```
        (b) $l \leftarrow l + 1$.
    4. Return to the original ordering of $A$ and $d$; determine the set $\mathcal{B}$ according to this
       ordering and set $\mathcal{N} := \{1, \ldots, n\} \backslash \mathcal{B}$.
    5. Set $B := A_{\mathcal{B}}$, $N := A_{\mathcal{N}}$, $D_{\mathcal{B}} := \mathrm{Diag}(d_{\mathcal{B}})$ and $D_{\mathcal{N}} := \mathrm{Diag}(d_{\mathcal{N}})$.
    6. Let $T := D_{\mathcal{B}}^{-1} B^{-1}$.
end
```

This preconditioner was originally proposed by Resende and Veiga in [53] in the context of network flow problems. In this case, A is a node-arc incidence matrix of a connected directed graph (with one row deleted to ensure that A has full row rank), and the elements of d are weights on the edges of the graph. Using this algorithm, we see that the set $\mathcal{B}$ created by the algorithm above defines a maximum spanning tree on the digraph. Oliveira and Sorensen [48] later proposed the use of this preconditioner for general matrices A.

For the purpose of stating the next result, we now introduce some notation. Let us define

$$\varphi_A := \max\{\|B^{-1}A\|_F : B \text{ is a basis of } A\}. \tag{41}$$

It is easy to show that $\varphi_A \leq \sqrt{m}\,\bar{\chi}_A$, where $\bar{\chi}_A$ is a well-known condition number (see [65]) defined as

$$\bar{\chi}_A := \sup\{\|A^T(ADA^T)^{-1}AD\| : D \in \mathrm{Diag}(\mathbb{R}^n_{++})\}.$$

Indeed, this follows from the fact that $\|C\|_F \leq \sqrt{m}\|C\|$ for any matrix $C \in \mathbb{R}^{m\times n}$ with $m \leq n$ and that an equivalent characterization of $\bar{\chi}_A$ is

$$\bar{\chi}_A := \max\{\|B^{-1}A\| : B \text{ is a basis of } A\}, \tag{42}$$

as shown in [60] and [65].

Recently, Monteiro, O'Neal and Tsuchiya showed the following result in [44].

Proposition 2.5.2 *Let a full row rank matrix $A \in \mathbb{R}^{m\times n}$ and $d \in \mathbb{R}^n_{++}$ be given. Let $T = T(A, d)$ be the preconditioner determined according to the Maximum Weight Basis Algorithm, and define $W := TAD^2A^TT^T$. Then, $\|TAD\| \leq \varphi_A$ and $\kappa(W) \leq \varphi_A^2$.*

Table 2: Values of $c(\kappa)$ and $f(\kappa)$ for Well-Known Iterative Solvers

Solver	$c(\kappa)$	$f(\kappa)$
SD	$\sqrt{\kappa}$	$(\kappa+1)/2$
CG	$2\sqrt{\kappa}$	$(\sqrt{\kappa}+1)/2$

Note that the bound φ_A^2 on $\kappa(W)$ is independent of the diagonal matrix D and depends only on A. In the next subsection, we derive bounds on the number of iterations needed by an iterative solver to solve (38) to a desired accuracy level.

2.5.3 Iteration Complexity for the Iterative Solver

In this subsection, we will develop a bound on the number of iterations that an iterative linear solver needs to perform to obtain a suitable approximate solution to (38). Instead of focusing on one specific solver, we will assume that we have a generic iterative linear solver with a prescribed rate of convergence. More specifically, we will assume that the generic iterative linear solver when applied to (38) generates a sequence of iterates $\{z^j\}$ satisfying the following condition:

$$\|q - Wz^j\| \leq c(\kappa)\left[1 - \frac{1}{f(\kappa)}\right]^j \|q - Wz^0\|, \quad \forall\, j = 0, 1, 2, \ldots, \tag{43}$$

where c and f are positive functions of $\kappa \equiv \kappa(W)$. For our purposes, we will also assume that the initial iterate $z^0 = 0$, so that $q - Wz^0 = q$.

Examples of solvers which satisfy (43) include the steepest descent (SD) and conjugate gradient (CG) methods, with the following values for $c(\kappa)$ and $f(\kappa)$:

The justification for the table above follows from Section 7.6 and Exercise 10 of Section 8.8 of [35].

Before we give the main convergence result, we state the following lemma, which we will prove in Section 3.1:

Lemma 2.5.3 *Assume that $T = T(A, d)$ and the initial point (x^0, s^0, y^0) is such that $s^0 \geq |c - A^T y^0|$ and $(x^0, s^0) \geq (\bar{x}, \bar{s})$ for some $(\bar{x}, \bar{s}, \bar{y}) \in \mathcal{S}$. Suppose also that $(x, s, y) \in \mathcal{N}(\gamma)$ and that $r = \eta r^0$ for some $\eta \in [0, 1]$. Then, the vector q defined in (40) satisfies $\|q\| \leq \Psi\sqrt{\mu}$,* *where*

$$\Psi := \frac{9n\varphi_A}{\sqrt{1-\gamma}} + \sqrt{n}\varphi_A + \sigma\varphi_A\sqrt{\frac{n}{1-\gamma}}. \tag{44}$$

■

The following result gives an upper bound on the number of iterations that the generic iterative linear solver needs to perform to obtain an iterate z^j satisfying $\|q - Wz^j\| \le \rho\sqrt{\mu}$, for some constant $\rho > 0$.

Theorem 2.5.4 *Suppose that the conditions of Lemma 2.5.3 are met and* $(1-\gamma)^{-1} = \mathcal{O}(1)$*. Then, a generic iterative solver with a convergence rate given by (43) generates an iterate* z^j *satisfying* $\|q - Wz^j\| \le \rho\sqrt{\mu}$ *in*

$$\mathcal{O}\left(f(\kappa)\log\left(\frac{c(\kappa)n\varphi_A}{\rho}\right)\right) \tag{45}$$

iterations, where $\kappa := \kappa(W)$*.*

Proof: Let j be any index satisfying

$$j \ge f(\kappa)\log\left(\frac{c(\kappa)\Psi}{\rho}\right). \tag{46}$$

Using the fact that $\log(1+x) \le x$ for all $x > -1$ and the above inequality, we conclude that

$$\begin{aligned}\log\left(\rho\sqrt{\mu}\right) &\ge \log\left(c(\kappa)\Psi\sqrt{\mu}\right) - \frac{j}{f(\kappa)} \\ &\ge \log\left(c(\kappa)\Psi\sqrt{\mu}\right) + j\log\left(1 - \frac{1}{f(\kappa)}\right).\end{aligned}$$

This together with the assumption that $z^0 = 0$, relation (43) and Lemma 2.5.3 imply that

$$\rho\sqrt{\mu} \;\ge\; c(\kappa)\left[1 - \frac{1}{f(\kappa)}\right]^j \Psi\sqrt{\mu} \;\ge\; c(\kappa)\left[1 - \frac{1}{f(\kappa)}\right]^j \|q\| \;\ge\; \|q - Wz^j\|.$$

Since $\Psi = \mathcal{O}(n\varphi_A)$ in view of Lemma 2.5.3, it follows that the right hand side of (46) is majorized by (45), from which the result follows. ■

We will refer to an inner iterate z^j satisfying $\|q - Wz^j\| \le \rho\sqrt{\mu}$ to as ρ-approximate solution of (38). In our interior-point algorithm in the next subsection, we will choose the constant ρ as $\rho = \gamma\sigma/(4\sqrt{n})$. As a consequence of Proposition 2.5.2, we obtain the following corollary.

Corollary 2.5.5 *Suppose that the conditions of Lemma 2.5.3 are met and that* $\max\{\underline{\sigma}^{-1}, \gamma^{-1}, (1-\gamma)^{-1}\} = \mathcal{O}(1)$*. If* $\rho = \gamma\sigma/(4\sqrt{n})$*, then the SD and CG methods generate a* ρ*-approximate solution in* $\mathcal{O}(\varphi_A^2 \log(n\varphi_A))$ *and* $\mathcal{O}(\varphi_A \log(n\varphi_A))$ *iterations, respectively.*

Proof: This result follows immediately from the assumptions, Theorem 2.5.4, Table 2 and Proposition 2.5.2. ∎

Note that the inner-iteration complexity bounds derived in Corollary 2.5.5 are not polynomial in general, since they depend on φ_A. However, these bounds will be polynomial if φ_A is polynomial. We will discuss this impact for general matrices A at the end of Section 2.5.4. In Section 2.5.5, we will consider a specific case when φ_A is polynomial, namely when A is the node-arc incidence matrix of a directed graph.

2.5.4 The Iterative IP Algorithm

In this subsection, we describe our main algorithm. It is essentially the IIP algorithm, except that the search direction $(\Delta x, \Delta s, \Delta y)$ is computed approximately with the use of iterative methods applied to (38).

Notice that under exact computations, the primal and dual residuals $r^k = (r_p^k, r_d^k)$ corresponding to the k-th iterate of Algorithm IIP always lie on the line segment between 0 and r^0 because of (33) and (34). It is well known that this property plays an important role in the convergence analysis of infeasible interior point methods. We will now show that it is still possible to ensure that r^k lies on the segment between 0 and r^0, even when Δy is an approximate solution of (35). Indeed, consider an approximate solution Δy which satisfies the following equation:

$$AD^2A^T\Delta y = -r_p - \sigma\mu AS^{-1}e + Ax - AD^2 r_d + f \tag{47}$$

where f is some error vector. Next, we solve for Δs using (36), so that $(\Delta s, \Delta y)$ satisfies (34).

The usual approach for choosing Δx is to use (37). However, this does not ensure that (33) is satisfied. To ensure that (33) is satisfied, we use the following equation:

$$\Delta x = -x + \sigma\mu S^{-1}e - D^2\Delta s - S^{-1}v \tag{48}$$

where v is a perturbation vector satisfying

$$AS^{-1}v = f. \tag{49}$$

Condition (49) is necessary and sufficient for Δx given by (48) to satisfy (33), since

$$\begin{aligned} A\Delta x &= A\left(-x + \sigma\mu S^{-1}e - D^2\Delta s - S^{-1}v\right) \\ &= -Ax + \sigma\mu AS^{-1}e - AD^2\Delta s - AS^{-1}v \\ &= -Ax + \sigma\mu AS^{-1}e + AD^2 r_d + AD^2A^T\Delta y - AS^{-1}v \\ &= -r_p + f - AS^{-1}v, \end{aligned} \tag{50}$$

due to (48), (36) and (47). Note from (48) that (32) is satisfied exactly if and only if $v = 0$, which in turn satisfies the necessary and sufficient condition $AS^{-1}v = f$ if and only if $f = 0$, i.e. when Δy is an exact solution of (35).

There are numerous choices for v. An obvious choice for v is to choose the least squares solution, i.e., to choose the optimal solution to $\min\{\|v\| : AS^{-1}v = f\}$. However, this is the type of computation we wish to avoid when using an iterative solver. A more effective choice for v is to choose a basis $\tilde{B}$ of A and let

$$v = (v_{\tilde{\mathcal{B}}}, v_{\tilde{\mathcal{N}}}) = (S_{\tilde{\mathcal{B}}}\tilde{B}^{-1}f\,,\,0), \tag{51}$$

where $(\tilde{\mathcal{B}}, \tilde{\mathcal{N}})$ is the index partition corresponding to the basis $\tilde{B}$.

It turns out that an obvious choice for $\tilde{B}$ in our approach is to let $\tilde{B}$ be equal to the maximum weight basis B corresponding to (A, d). More specifically, recall that in our algorithm, we compute Δy approximately using the preconditioned system $Wz = q$. Let $\widetilde{\Delta y}$ denote the final iterate z^j in the approximate solution of $Wz = q$, and let $\tilde{f} = W\widetilde{\Delta y} - q$. Letting $\Delta y = T^T\widetilde{\Delta y}$, it is easy to see that Δy is an approximate solution satisfying (47) with error $f = T^{-1}\tilde{f} = BD_{\mathcal{B}}\tilde{f}$. Using this expression for f and letting $\tilde{B} = B$ in (51), we obtain

$$v = (S_{\tilde{\mathcal{B}}}\tilde{B}^{-1}f\,,\,0) = (S_{\mathcal{B}}B^{-1}BD_{\mathcal{B}}\tilde{f}\,,\,0) = ((X_{\mathcal{B}}S_{\mathcal{B}})^{1/2}\tilde{f}\,,\,0). \tag{52}$$

Note that by (48), we have $X\Delta s + S\Delta x = -XSe + \sigma\mu e - v$, i.e. (32) is satisfied only approximately. To ensure convergence of our method, it turns out that it is important to keep $\|\tilde{f}\|$, and hence $\|v\|$, small. In our algorithm below, we require that

$$\|\tilde{f}\| \leq \frac{\gamma\sigma}{4\sqrt{n}}\sqrt{\mu} \tag{53}$$

so as to ensure that the number of outer iterations of our method is still polynomially bounded.

We now present our main algorithm:

Algorithm IIP-IS:

```
1. Start: Let $\epsilon > 0$, $\gamma \in (0,1)$, $(x^0, s^0, y^0) \in \mathcal{N}(\gamma)$ and $0 < \underline{\sigma} < \overline{\sigma} < 4/5$ be given. Set $k = 0$.

2. While $\mu_k := \mu(x^k, s^k) > \epsilon$ do

   (a) Let $(x, s, y) := (x^k, s^k, y^k)$, and choose $\sigma \in [\underline{\sigma}, \overline{\sigma}]$.
   (b) Set $d = S^{-1/2}X^{1/2}e$, $r_p = Ax - b$, $r_d = A^T y + s - c$, and $r = (r_p, r_d)$.
   (c) Build the preconditioner $T = T(A, d)$ using the Maximum Weight Basis Algorithm.
   (d) Find an approximate solution $\widetilde{\Delta y}$ of (38) such that $\tilde{f} = W\widetilde{\Delta y} - q$ satisfies (53).
   (e) Let $v$ be computed according to (52). Set $\Delta y = T^T\widetilde{\Delta y}$, and compute $\Delta s$ and $\Delta x$ by (36) and (48), respectively.
   (f) Compute $\tilde{\alpha} := \mathrm{argmax}\{\alpha \in [0,1] : w + \alpha'\Delta w \in \mathcal{N}(\gamma),\ \forall \alpha' \in [0, \alpha]\}$, where $w := (x, s, y)$ and $\Delta w := (\Delta x, \Delta s, \Delta y)$.
   (g) Compute $\bar{\alpha} := \mathrm{argmin}\{(x + \alpha\Delta x)^T(s + \alpha\Delta s) : \alpha \in [0, \tilde{\alpha}]\}$.
   (h) Let $(x^{k+1}, s^{k+1}, y^{k+1}) = w + \bar{\alpha}\Delta w$, and set $k \leftarrow k + 1$.

   end (while)
```

Using this algorithm, we obtain nearly the exact same polynomial convergence result as Theorem 2.5.1. The results for Algorithm IIP-IS are summarized in the following theorem, which we will prove in Section 3.2.

Theorem 2.5.6 *Assume that the constants γ, $\underline{\sigma}$ and $\overline{\sigma}$ are such that*

$$\max\left\{\gamma^{-1}\,,\,(1-\gamma)^{-1}\,,\,\underline{\sigma}^{-1}\,,\,\left(1-\frac{5}{4}\overline{\sigma}\right)^{-1}\right\} = \mathcal{O}(1), \tag{54}$$

and that the initial point $(x^0, s^0, y^0) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfies $(x^0, s^0) \geq (\bar{x}, \bar{s})$ for some $(\bar{x}, \bar{s}, \bar{y}) \in \mathcal{S}$. Then, Algorithm IIP-IS generates an iterate $(x^k, s^k, y^k) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfying $\mu_k \leq \epsilon\mu_0$ and $\|r^k\| \leq \epsilon\|r^0\|$ within $\mathcal{O}\left(n^2 \log(1/\epsilon)\right)$ iterations.

Note that while the number of "outer" iterations is polynomial for our algorithm, the overall complexity is not, due to the total number of "inner" iterations. To see this, consider a given iteration of our algorithm with the CG solver. It is easy to see that steps (a), (b), and (e) through (h) can be carried out in $\mathcal{O}(mn)$ flops. Let us now examine the other two steps (c) and (d). The sorting part of the Maximum Weight Basis algorithm can be done in $\mathcal{O}(n \log n)$ with a quick sorting algorithm. Computing T and its corresponding LU factorization can be done in $\mathcal{O}(m^2 n)$ flops. Hence, the entire step (c) takes $\mathcal{O}(n \max\{m^2, \log n\})$ flops. Now, notice that each step of the CG solver requires $\mathcal{O}(mn)$ flops. Since $\mathcal{O}(\varphi_A \log(n\varphi_A))$ iterations of the CG solver are required, we conclude that step (d) takes $\mathcal{O}(mn\varphi_A \log(n\varphi_A))$ flops. Thus, the number of flops per iteration of Algorithm IIP-IS is $\mathcal{O}(n \max\{m\varphi_A \log(n\varphi_A), m^2\})$.

In the next subsection, we will consider a specific case where φ_A is polynomial, namely when A is the node-arc incidence matrix of a directed graph.

2.5.5 Application to Network Flows

Consider a standard network flow problem of the form (25). In this case, A is the node-arc incidence matrix of a simple connected directed graph $\mathcal{G}$, with one row deleted to ensure that A has full row rank. We will show that for this particular problem, Algorithm IIP-IS is indeed a polynomial-time algorithm. The key result comes from the following observation: since A is a node-arc incidence matrix, it is totally unimodular so that every element of $B^{-1}A$ is 1, 0, or -1. Thus, $\varphi_A \leq \sqrt{mn}$, as can be seen from definition (41) of φ_A.

Additional savings in the complexity of our algorithm can be obtained by using the special structure of the matrix A. Indeed, consider a single iteration of Algorithm IIP-IS using the CG solver. Steps (a), (b), and (e) through (h) now take $\mathcal{O}(n)$ flops since A has only $2n$ nonzero entries. Since a maximum weight basis is now a maximum spanning tree on $\mathcal{G}$, it can be found with $\mathcal{O}(n \log n)$ flops using either Prim's or Kruskal's algorithm (see e.g. [14]). The basis matrix B, under a suitable ordering, will be upper triangular, so we do not need to form B^{-1} explicitly; thus step (c) requires $\mathcal{O}(n \log n)$ flops. Next, Corollary 2.5.5 and the fact that $\varphi_A \le \sqrt{mn}$ imply that the CG method takes $\mathcal{O}(\sqrt{mn} \log n)$ iterations to find a suitably accurate solution of (38). As each step of the CG method takes $\mathcal{O}(n)$ flops, step (d) requires $\mathcal{O}(m^{1/2}n^{3/2} \log n)$ flops. Thus, a single outer iteration of Algorithm IIP-IS applied to a minimum-cost network flow problem requires $\mathcal{O}(m^{1/2}n^{3/2} \log n)$ flops.

2.6 Inexact LP using the MWB: Proofs

In this section, we give detailed proofs for the results given in Sections 2.5.3 and 2.5.4. Section 2.6.1 will be devoted to the proofs of the results in Section 2.5.3, while Section 2.6.2 will give the proofs for the results in Section 2.5.4.

2.6.1 Results for the Iterative Solver

Consider a generic iterative solver which solves (38), and suppose that this solver satisfies (43) at each iteration. We will seek to get $\|q - Wz^j\| \le \rho\sqrt{\mu}$ for some term $\rho > 0$. We begin with some technical lemmas:

Lemma 2.6.1 *Let (x^0, s^0, y^0) and (x, s, y) be points such that $r(x,s,y) = \eta r(x^0,s^0,y^0)$ for some $\eta \in \mathbb{R}$, and let $(\bar{x}, \bar{s}, \bar{y})$ be a point such that $r(\bar{x},\bar{s},\bar{y}) = 0$. Then,*

$$\begin{aligned} 0 = \eta^2 x^{0^T} s^0 + (1-\eta)^2 \bar{x}^T \bar{s} + x^T s + \eta(1-\eta)(x^{0^T}\bar{s} + \bar{x}^T s^0) \\ -\eta(x^{0^T} s + x^T s^0) - (1-\eta)(\bar{x}^T s + x^T \bar{s}). \end{aligned} \tag{55}$$

Proof: Using the definition of r, it is easy to see that

$$\begin{aligned} A(x - \eta x^0 - (1-\eta)\bar{x}) &= 0 \\ (s - \eta s^0 - (1-\eta)\bar{s}) + A^T(y - \eta y^0 - (1-\eta)\bar{y}) &= 0 \end{aligned}$$

Multiplying the second relation by $[x - \eta x^0 - (1-\eta)\bar{x}]^T$ on the left and using the first relation, we get

$$[x - \eta x^0 - (1-\eta)\bar{x}]^T [s - \eta s^0 - (1-\eta)\bar{s}] = 0. \tag{56}$$

Expanding this equality, we obtain (55). ▪

Lemma 2.6.2 *Let $(x^0, s^0, y^0) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ be a point such that $(x^0, s^0) \geq (\bar{x}, \bar{s})$ for some $(\bar{x}, \bar{s}, \bar{y}) \in \mathcal{S}$. Then, for any point $(x, s, y) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ such that $r = \eta r^0$ for some $\eta \in [0,1]$ and $\eta \leq x^T s / {x^0}^T s^0$, we have that $\eta({x^0}^T s + {s^0}^T x) \leq 3n\mu$.*

Proof: By assumption, there exists $(\bar{x}, \bar{s}, \bar{y}) \in \mathcal{S}$ such that $\bar{x} \leq x^0$ and $\bar{s} \leq s^0$. Since $r = \eta r^0$ and $(\bar{x}, \bar{s}, \bar{y}) \in \mathcal{S}$, the points (x, s, y), (x^0, s^0, y^0), and $(\bar{x}, \bar{s}, \bar{y})$ satisfy the assumption of the previous lemma. Hence, by equation (55), along with the facts that $\eta \leq x^T s / {x^0}^T s^0$, $\bar{x}^T \bar{s} = 0$, $(x, s) \geq 0$, $(\bar{x}, \bar{s}) \geq 0$, $(x^0, s^0) \geq 0$, $\eta \in [0,1]$, $\bar{x} \leq x^0$, and $\bar{s} \leq s^0$, we conclude that

$$\begin{aligned}
\eta({x^0}^T s + {s^0}^T x) &\leq \eta^2 {x^0}^T s^0 + x^T s + \eta(1-\eta)({x^0}^T \bar{s} + {s^0}^T \bar{x}) \\
&\leq \eta^2 {x^0}^T s^0 + x^T s + 2\eta(1-\eta) {x^0}^T s^0 \\
&\leq 2\eta {x^0}^T s^0 + x^T s \leq 3x^T s.
\end{aligned}$$

▪

Next, we turn to the proof of Lemma 2.5.3:

Proof of Lemma 2.5.3: By (40) and the triangle inequality for norms, we have

$$\|q\| \leq \|Tr_p\| + \|\sigma\mu TAS^{-1}e\| + \|TAx\| + \|TAD^2 r_d\|. \tag{57}$$

We will now bound each of the terms in the right hand side of (57). Let $(\bar{x}, \bar{s}, \bar{y}) \in \mathcal{S}$ and (x^0, s^0, y^0) satisfy the assumptions of Lemma 2.5.3, so that $x^0 \geq \bar{x}$, $s^0 \geq \bar{s}$ and $s^0 \geq |c - A^T y^0|$. Using these inequalities, the assumption that $(x, s, y) \in \mathcal{N}(\gamma)$ and Lemma 2.6.2, we obtain

$$\eta\|S(\bar{x} - x^0)\| \;\leq\; \eta\|Sx^0\| \;\leq\; \eta s^T x^0 \;\leq\; 3n\mu \tag{58}$$

$$\eta\|X(s^0 + A^T y^0 - c)\| \;\leq\; 2\eta\|Xs^0\| \;\leq\; 2\eta x^T s^0 \;\leq\; 6n\mu. \tag{59}$$

Thus, using the relations $r = \eta r^0$, $b = A\bar{x}$, (58) and (59), the fact that $(x, s, y) \in \mathcal{N}(\gamma)$ and Proposition 2.5.2, we obtain

$$\begin{aligned} \|Tr_p\| &= \eta \,\|Tr_p^0\| = \eta \,\|T(b - Ax^0)\| = \eta \,\|TA(\bar{x} - x^0)\| = \eta \,\|(TAD)(XS)^{-1/2}S(\bar{x} - x^0)\| \\ &\leq \eta \,\|TAD\| \,\|(XS)^{-1/2}\| \,\|S(\bar{x} - x^0)\| \leq \varphi_A \frac{1}{\sqrt{(1-\gamma)\mu}} \, 3n\mu = \frac{3n\varphi_A}{\sqrt{1-\gamma}}\sqrt{\mu} \end{aligned}$$

and

$$\begin{aligned} \|TAD^2 r_d\| &\leq \|TAD\| \,\|Dr_d\| = \eta\|TAD\| \,\|Dr_d^0\| = \eta\|TAD\| \,\|D(s^0 + A^T y^0 - c)\| \\ &\leq \eta\|TAD\| \,\|(XS)^{-1/2}\| \,\|X(s^0 + A^T y^0 - c)\| \\ &\leq \varphi_A \frac{1}{\sqrt{(1-\gamma)\mu}} \, 6n\mu = \frac{6n\varphi_A}{\sqrt{1-\gamma}}\sqrt{\mu}. \end{aligned}$$

Similarly, we have

$$\|\sigma\mu TAS^{-1}e\| \leq \sigma\mu\|TAD\| \,\|(XS)^{-1/2}\| \,\|e\| \leq \sigma\mu\varphi_A \frac{1}{\sqrt{(1-\gamma)\mu}}\sqrt{n} = \sigma\varphi_A\sqrt{\frac{n}{1-\gamma}}\sqrt{\mu}$$

and

$$\|TAx\| = \|TAD(XS)^{1/2}e\| \leq \|TAD\| \,\|(XS)^{1/2}e\| \leq \varphi_A\sqrt{n\mu},$$

where in the last inequality we used the fact that $\|(XS)^{1/2}e\| = \sqrt{n\mu}$. The result now follows by combining the four bounds obtained above with (57). ■

2.6.2 Convergence Results for Algorithm IIP-IS

In this subsection, we will provide the proof of Theorem 2.5.6.

For the sake of future reference, we note that $(\Delta x, \Delta s, \Delta y)$ satisfies

$$A\Delta x = -r_p \tag{60}$$

$$A^T\Delta y + \Delta s = -r_d \tag{61}$$

$$X\Delta s + S\Delta x = -Xs + \sigma\mu e - v \tag{62}$$

by equations (49), (50), (36), and (48), respectively. Throughout this section, we use the following notation:

$$\begin{aligned}
(x(\alpha), s(\alpha), y(\alpha)) &:= (x, s, y) + \alpha(\Delta x, \Delta s, \Delta y), \\
\mu(\alpha) &:= x(\alpha)^T s(\alpha)/n, \\
r(\alpha) &:= r(x(\alpha), s(\alpha), y(\alpha)) \ = \ (Ax(\alpha) - b, A^T y(\alpha) + s(\alpha) - c).
\end{aligned}$$

Lemma 2.6.3 *Assume that $(\Delta x, \Delta s, \Delta y)$ satisfies (60)-(62) for some $\sigma \in \mathbb{R}$, $v \in \mathbb{R}^n$ and $(x, s, y) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$. Then, for every $\alpha \in \mathbb{R}$, we have:*

(a) $X(\alpha)s(\alpha) = (1-\alpha)Xs + \alpha\sigma\mu e - \alpha v + \alpha^2 \Delta X \Delta s$;

(b) $\mu(\alpha) = [1 - \alpha(1-\sigma)]\mu - \alpha v^T e/n + \alpha^2 \Delta x^T \Delta s/n$;

(c) $r(\alpha) = (1-\alpha)r$.

Proof: Using (62), we obtain

$$\begin{aligned}
X(\alpha)s(\alpha) &= (X + \alpha\Delta X)(s + \alpha\Delta s) \\
&= Xs + \alpha(X\Delta s + S\Delta x) + \alpha^2 \Delta X \Delta s \\
&= Xs + \alpha(-Xs + \sigma\mu e - v) + \alpha^2 \Delta X \Delta s \\
&= (1-\alpha)Xs + \alpha\sigma\mu e - \alpha v + \alpha^2 \Delta X \Delta s,
\end{aligned}$$

thereby showing that a) holds. Left multiplying the above equality by e^T and dividing the resulting expression by n, we easily conclude that b) holds. Statement c) can be easily verified by means of (60) and (61). ∎

Lemma 2.6.4 *Assume that $(\Delta x, \Delta s, \Delta y)$ satisfies (60)-(62) for some $\sigma > 0$, $(x, s, y) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ and $v \in \mathbb{R}^n$ satisfying $v^T e/n \leq \sigma\mu/2$. Then, for every scalar α satisfying*

$$0 \leq \alpha \leq \min\left\{1\,,\ \frac{\sigma\mu}{2\,\|\Delta X \Delta s\|_\infty}\right\}, \tag{63}$$

we have

$$\frac{\|r(\alpha)\|}{\|r\|} \leq \frac{\mu(\alpha)}{\mu}. \tag{64}$$

Proof: Using Lemma 2.6.3(b) and the assumption that $v^T e/n \leq \sigma\mu/2$, we conclude for every α satisfying (63) that

$$\begin{aligned}
\mu(\alpha) &= [1-\alpha(1-\sigma)]\mu - \alpha v^T e/n + \alpha^2 \Delta x^T \Delta s/n \\
&\geq [1-\alpha(1-\sigma)]\mu - \frac{1}{2}\,\alpha\sigma\mu + \alpha^2 \Delta x^T \Delta s/n \\
&\geq (1-\alpha)\mu + \frac{1}{2}\,\alpha\sigma\mu - \alpha^2 \|\Delta X \Delta s\|_\infty \\
&\geq (1-\alpha)\mu.
\end{aligned}$$

The result now follows from the last relation and Lemma 2.6.3(c). ∎

Lemma 2.6.5 *Assume that* $(\Delta x, \Delta s, \Delta y)$ *satisfies (60)-(62) for some* $\sigma > 0$, $(x, s, y) \in \mathcal{N}(\gamma)$ *with* $\gamma \in [0, 1]$, *and* $v \in \mathbb{R}^n$ *satisfying* $\|v\|_\infty \leq \gamma\sigma\mu/4$. *Then,* $(x(\alpha), s(\alpha), y(\alpha)) \in \mathcal{N}(\gamma)$ *for every scalar* α *satisfying*

$$0 \leq \alpha \leq \min\left\{1\,,\ \frac{\gamma\sigma\mu}{4\,\|\Delta X \Delta s\|_\infty}\right\}. \tag{65}$$

Proof: Since the assumption that $\gamma \in [0, 1]$ and $\|v\|_\infty \leq \gamma\sigma\mu/4$ imply that $v^T e/n \leq \sigma\mu/2$, it follows from Lemma 2.6.4 that (64) holds for every α satisfying (63), and hence (65). Thus, for every α satisfying (65), we have

$$\frac{\|r(\alpha)\|}{\|r^0\|} = \frac{\|r(\alpha)\|}{\|r\|}\,\frac{\|r\|}{\|r^0\|} \leq \frac{\mu(\alpha)}{\mu}\,\frac{\mu}{\mu_0} = \frac{\mu(\alpha)}{\mu_0}. \tag{66}$$

Now, it is easy to see that for every $u \in \mathbb{R}^n$ and $\tau \in [0, n]$, there holds $\|u - \tau(u^T e/n)e\|_\infty \leq (1+\tau)\|u\|_\infty$. Using this inequality twice, the fact that $(x, s, y) \in \mathcal{N}(\gamma)$ and statements (a) and (b) of Lemma 2.6.3, we conclude for every α satisfying (65) that

$$\begin{aligned}
&X(\alpha)s(\alpha) - (1-\gamma)\,\mu(\alpha)e \\
&\quad = \quad (1-\alpha)\left[Xs - (1-\gamma)\mu e\right] + \alpha\gamma\sigma\mu e - \alpha\left[v - (1-\gamma)\left(\frac{v^T e}{n}\right)e\right] \\
&\qquad\quad + \alpha^2\left[\Delta X\Delta s - (1-\gamma)\left(\frac{\Delta x^T\Delta s}{n}\right)e\right] \\
&\quad \geq \quad \alpha\left[\gamma\sigma\mu - \left\|v - (1-\gamma)\frac{v^T e}{n}\,e\right\|_\infty - \alpha\left\|\Delta x\Delta s - (1-\gamma)\frac{\Delta x^T\Delta s}{n}\,e\right\|_\infty\right]e \\
&\quad \geq \quad \alpha\left(\gamma\sigma\mu - 2\|v\|_\infty - 2\alpha\|\Delta X\Delta s\|_\infty\right)e \;\geq\; \alpha\left(\gamma\sigma\mu - \frac{1}{2}\gamma\sigma\mu - \frac{1}{2}\gamma\sigma\mu\right)e \;\geq\; 0.
\end{aligned}$$

We have thus shown that $(x(\alpha), s(\alpha), y(\alpha)) \in \mathcal{N}(\gamma)$ for every α satisfying (65). ▪

Next, we consider the minimum step length allowed under our algorithm:

Lemma 2.6.6 *In every iteration of Algorithm IIP-IS, the step length $\bar{\alpha}$ satisfies*

$$\bar{\alpha} \geq \min\left\{1, \frac{\min\{\gamma\sigma, 1-\frac{5}{4}\sigma\}\mu}{4\,\|\Delta X\Delta s\|_\infty}\right\} \tag{67}$$

and

$$\mu(\bar{\alpha}) \leq \left[1 - \left(1 - \frac{5}{4}\,\sigma\right)\frac{\bar{\alpha}}{2}\right]\mu. \tag{68}$$

Proof: Using (52) and (53), we conclude that

$$\|v\|_\infty = \|(X_{\mathcal{B}}S_{\mathcal{B}})^{1/2}\tilde{f}\|_\infty \leq \|X_{\mathcal{B}}S_{\mathcal{B}}\|^{1/2}\|\tilde{f}\|_\infty \leq \sqrt{n\mu}\;\frac{\gamma\sigma}{4\sqrt{n}}\sqrt{\mu} = \frac{1}{4}\gamma\sigma\mu. \tag{69}$$

Hence, by Lemma 2.6.5, the quantity $\tilde{\alpha}$ computed in step (g) of Algorithm IIP-IS satisfies

$$\tilde{\alpha} \geq \min\left\{1, \frac{\gamma\sigma\mu}{4\,\|\Delta X\Delta s\|_\infty}\right\}. \tag{70}$$

Moreover, by (69), it follows that the coefficient of α in the expression for $\mu(\alpha)$ in Lemma 2.6.3(b) satisfies

$$-(1-\sigma)\mu - \frac{v^T e}{n} \leq -(1-\sigma)\mu + \|v\|_\infty \leq -(1-\sigma)\mu + \frac{1}{4}\,\gamma\sigma\mu = -\left(1 - \frac{5}{4}\,\sigma\right)\mu < 0, \tag{71}$$

since $\sigma \in (0, \frac{4}{5})$. Hence, if $\Delta x^T\Delta s \leq 0$, it is easy to see that $\bar{\alpha} = \tilde{\alpha}$, and hence that (67) holds in view of (70). Moreover, by Lemma 2.6.3(b) and (71), we have

$$\mu(\bar{\alpha}) \leq [1 - \bar{\alpha}(1-\sigma)]\mu - \bar{\alpha}\frac{v^T e}{n} \leq \left[1 - \left(1 - \frac{5}{4}\,\sigma\right)\bar{\alpha}\right]\mu \leq \left[1 - \left(1 - \frac{5}{4}\,\sigma\right)\frac{\bar{\alpha}}{2}\right]\mu,$$

showing that (68) also holds. We now consider the case where $\Delta x^T \Delta s > 0$. In this case, we have $\bar{\alpha} = \min\{\alpha_{\min}\,,\, \tilde{\alpha}\}$, where $\alpha_{\min}$ is the unconstrained minimum of $\mu(\alpha)$. It is easy to see that

$$\alpha_{\min} \;=\; \frac{n\mu(1-\sigma) + v^T e}{2\Delta x^T \Delta s} \;\geq\; \frac{n\mu(1-\sigma) - \frac{1}{4}\sigma n\mu}{2\Delta x^T \Delta s} \;\geq\; \frac{\mu(1-\frac{5}{4}\sigma)}{2\,\|\Delta X \Delta s\|_\infty}.$$

The last two observations together with (70) imply that (67) holds in this case too. Moreover, since the function $\mu(\alpha)$ is convex, it must lie below the function $\phi(\alpha)$ over the interval $[0, \alpha_{\min}]$, where $\phi(\alpha)$ is the affine function interpolating $\mu(\alpha)$ at $\alpha = 0$ and $\alpha = \alpha_{\min}$. Hence,

$$\mu(\bar{\alpha}) \leq \phi(\bar{\alpha}) = [1 - (1-\sigma)\frac{\bar{\alpha}}{2}]\mu - \bar{\alpha}\frac{v^T e}{2n} \leq \left[1 - \left(1 - \frac{5}{4}\sigma\right)\frac{\bar{\alpha}}{2}\right]\mu, \tag{72}$$

where the second inequality follows from (71). We have thus shown that $\bar{\alpha}$ satisfies (68). ▪

Our next task will be to show that the stepsize $\bar{\alpha}$ remains bounded away from zero. In view of (67), it sufficient to show that the quantity $\|\Delta X \Delta s\|_\infty$ remains bounded. The next lemma addresses this issue.

Lemma 2.6.7 *Let $(x^0, s^0, y^0) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ be such that $(x^0, s^0) \geq (\bar{x}, \bar{s})$ for some $(\bar{x}, \bar{s}, \bar{y}) \in \mathcal{S}$, and let $(x, s, y) \in \mathcal{N}(\gamma)$ be such that $r = \eta r^0$ for some $\eta \in [0, 1]$. Then, the search direction $(\Delta x, \Delta s, \Delta y)$ generated by Algorithm IIP-IS satisfies*

$$\max(\|D^{-1}\Delta x\|, \|D\Delta s\|) \leq \left(1 - 2\sigma + \frac{\sigma^2}{1-\gamma}\right)^{1/2} \sqrt{n\mu} + \frac{6n}{\sqrt{1-\gamma}}\sqrt{\mu} + \frac{\gamma\sigma}{4\sqrt{n}}\sqrt{\mu}.$$

Proof: Relations (60) and (61) and the assumption $r = \eta r^0$ imply that

$$\begin{aligned} A(\Delta x + \eta(x^0 - \bar{x})) &= 0 \\ A^T(\Delta y + \eta(y^0 - \bar{y})) + (\Delta s + \eta(s^0 - \bar{s})) &= 0, \end{aligned}$$

from which it follows that $(\Delta x + \eta(x^0 - \bar{x}))^T(\Delta s + \eta(s^0 - \bar{s})) = 0$. Multiplying (62) on the left by $(XS)^{-1/2}$, we obtain $D^{-1}\Delta x + D\Delta s = H(\sigma) - (XS)^{-1/2}v$, where $H(\sigma) := -(XS)^{1/2}e + \sigma\mu(XS)^{-1/2}e$. Equivalently, we have that

$$\begin{aligned} & D^{-1}(\Delta x + \eta(x^0 - \bar{x})) + D(\Delta s + \eta(s^0 - \bar{x})) \\ &= \; H(\sigma) + \eta(D(s^0 - \bar{s}) + D^{-1}(x^0 - \bar{x})) - (XS)^{-1/2}v. \end{aligned}$$

Using the fact that the two terms on the left hand side of the above identity are orthogonal, along with the fact that $\|(XS)^{-1/2}v\| = \|\tilde{f}\|$ by (52), we obtain

$$
\begin{aligned}
&\max\left(\|D^{-1}(\Delta x + \eta(x^0 - \bar{x}))\|, \|D(\Delta s + \eta(s^0 - \bar{s})\|\right) \\
&\quad \le \|H(\sigma) + \eta(D(s^0 - \bar{s}) + D^{-1}(x^0 - \bar{x})) - (XS)^{-1/2}v\| \\
&\quad \le \|H(\sigma)\| + \eta\left(\|D(s^0 - \bar{s})\| + \|D^{-1}(x^0 - \bar{x})\|\right) + \|\tilde{f}\|.
\end{aligned}
$$

This, together with the triangle inequality and the definition of D, imply that

$$
\begin{aligned}
\max(\|D^{-1}\Delta x\|, \|D\Delta s\|) &\le \|H(\sigma)\| + 2\eta\left(\|D(s^0 - \bar{s})\| + \|D^{-1}(x^0 - \bar{x})\|\right) + \|\tilde{f}\| \\
&\le \|H(\sigma)\| + 2\eta\|(XS)^{-1/2}\|\left(\|X(s^0 - \bar{s})\| + \|S(x^0 - \bar{x})\|\right) + \|\tilde{f}\| \\
&\le \|H(\sigma)\| + \frac{2\eta}{\sqrt{(1-\gamma)\mu}}\left(\|X(s^0 - \bar{s})\| + \|S(x^0 - \bar{x})\|\right) + \|\tilde{f}\| \qquad (73)
\end{aligned}
$$

It is well-known that

$$
\|H(\sigma)\| \le \left(1 - 2\sigma + \frac{\sigma^2}{1-\gamma}\right)^{1/2} \sqrt{n\mu}. \qquad (74)
$$

Moreover, using the fact that $\bar{s} \le s^0$ and $\bar{x} \le x^0$ along with Lemma 2.6.2, we obtain

$$
\eta\|X(s^0 - \bar{s})\| + \|S(x^0 - \bar{x})\| \le \eta({s^0}^T x + {x^0}^T s) \le 3n\mu. \qquad (75)
$$

The result now follows by incorporating inequalities (74), (75) and (53) into (73). ▪

We are now ready to prove Theorem 2.5.6.

Proof of Theorem 2.5.6: Let $(\Delta x^k, \Delta s^k, \Delta y^k)$ denote the search direction, and let $r^k = r(x^k, s^k, y^k)$ and $\mu_k = \mu(x^k, s^k)$, at the k-th iteration of Algorithm IIP-IS. Clearly, $(x^k, s^k, y^k) \in \mathcal{N}(\gamma)$, and using Lemma 2.6.3, it is easy to see that $r^k = \eta r^0$ for some $\eta \in (0,1)$. Hence, using Lemma 2.6.7, assumption (54) and the inequality

$$
\|\Delta X^k \Delta s^k\|_\infty \le \|\Delta X^k \Delta s^k\| \le \|(D^k)^{-1}\Delta x^k\| \, \|D^k \Delta s^k\|,
$$

we easily see that $\|\Delta X^k \Delta s^k\|_\infty = \mathcal{O}(n^2)\mu_k$. Using this conclusion together with assumption (54) and Lemma 2.6.6, we see that, for some universal constant $\beta > 0$, we have

$$
\mu_{k+1} \le \left(1 - \frac{\beta}{n^2}\right)\mu_k, \quad \forall k \ge 0.
$$

The conclusion of the theorem now follows by using the above inequality, the fact that $\|r^k\|/\|r^0\| \leq \mu_k/\mu_0$ for all $k \geq 0$, and some standard arguments (see, for example, Theorem 3.2 of [68]). ▪

2.7 Inexact LP using the MWB: Concluding Remarks

We have shown in this chapter that the outer iteration complexity of Algorithm IIP-IS is the same as that of its direct counterpart, namely Algorithm IIP. We strongly believe that, using approaches similar to ours, it is possible to develop iterative versions of other primal-dual IP methods whose outer iteration complexities match those of their direct counterparts.

As we showed in Section 2.5.4, an inexact Δy leads to an error in the Newton equation (33) corresponding to primal feasibility. We have addressed this problem by introducing the vector v in equation (48) defining Δx. This correction term v can be used, not only in the context of iterative methods, but also in connection with direct methods, as was pointed out in Section 2.5.4. It would be interesting to see how the addition of this correction term v in the context of direct methods could help handle LP problems which are extremely hard to solve due to the ill-conditioning of AD^2A^T. Clearly, a certain overhead exists in computing v, but it might be worthwhile in such a case.

In order to satisfy the polynomial convergence of our methods, we have imposed stringent conditions on $\tilde{f}$. In a practical situation, it may be more appropriate to monitor v directly. Indeed, as long as v satisfies the requirements in Lemmas 2.6.4 and 2.6.5, the outer iteration convergence analysis used in this chapter remains valid. Moreover, weaker requirements than those imposed on v in the two lemmas above might be more advantageous from the practical point of view. This is certainly a topic that deserves further investigation.

CHAPTER III

INEXACT CQP ALGORITHMS USING THE MWB AND OTHER PRECONDITIONERS

3.1 CQP Algorithm Using the MWB: Introduction

In this chapter we develop an interior-point long-step primal-dual infeasible path-following (PDIPF) algorithm for convex quadratic programming (CQP) whose search directions are computed by means of an iterative linear solver. We will refer to this algorithm as an *inexact* algorithm, in the sense that the Newton system which determines the search direction will only be solved approximately at each iteration. The problem we consider is

$$\min_x \left\{ \frac{1}{2} x^T Q x + c^T x : \quad Ax = b, \ x \geq 0 \right\}, \tag{76}$$

where the data are $Q \in \mathbb{R}^{n \times n}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$, and the decision vector is $x \in \mathbb{R}^n$. We also assume that Q is positive semidefinite, and that a factorization $Q = VE^2V^T$ is explicitly given, where $V \in \mathbb{R}^{n \times l}$, and E is a $l \times l$ positive diagonal matrix.

A similar algorithm for solving the special case of linear programming (LP), i.e. problem (76) with $Q = 0$, was developed by Monteiro and O'Neal in [42]. The algorithm studied in [42] is essentially the long-step PDIPF algorithm studied in [28, 72], the only difference being that the search directions are computed by means of an iterative linear solver. We refer to the iterations of the iterative linear solver as the *inner iterations* and to the ones performed by the interior-point method itself as the *outer iterations*. The main step of the algorithm studied in [28, 42, 72] is the computation of the primal-dual search direction $(\Delta x, \Delta s, \Delta y)$, whose Δy component can be found by solving a system of the form $AD^2A^T\Delta y = g$, referred to as the *normal equation*, where $g \in \mathbb{R}^m$ and the positive diagonal matrix D depends on the current primal-dual iterate. In contrast to [28, 72], the algorithm studied in [42] uses an iterative linear solver to obtain an approximate solution to the normal equation. Since the condition number of the normal matrix AD^2A^T may become excessively large on degenerate

LP problems (see e.g. [32]), the maximum weight basis (MWB) preconditioner T introduced in [48, 53, 63] is used to better condition this matrix and an approximate solution of the resulting equivalent system with coefficient matrix $TAD^2A^TT^T$ is then computed. By using a result obtained in [44], which establishes that the condition number of $TAD^2A^TT^T$ is uniformly bounded by a quantity depending on A only, Monteiro and O'Neal [42] show that the number of inner iterations of the algorithm in [42] can be uniformly bounded by a constant depending on n and A.

In the case of CQP, the standard normal equation takes the form

$$A(Q + X^{-1}S)^{-1}A^T\Delta y \;=\; g, \tag{77}$$

for some vector g. When Q is not diagonal, the matrix $(Q + X^{-1}S)^{-1}$ is not diagonal, and hence the coefficient matrix of (77) does not have the form required for the result of [44] to hold. To remedy this difficulty, we develop in this chapter a new linear system, referred to as the *augmented normal equation* (ANE), to determine a portion of the primal-dual search direction. This equation has the form $\tilde{A}\tilde{D}^2\tilde{A}^Tu = w$, where $w \in \mathbb{R}^{m+l}$, $\tilde{D}$ is an $(n+l)\times(n+l)$ positive diagonal matrix and $\tilde{A}$ is a 2×2 block matrix of dimension $(m+l)\times(n+l)$ whose blocks consist of A, V^T, the zero matrix and the identity matrix (see equation (96)). As was done in [42], a MWB preconditioner $\tilde{T}$ for the ANE is computed and an approximate solution of the resulting preconditioned equation with coefficient matrix $\tilde{T}\tilde{A}\tilde{D}^2\tilde{A}^T\tilde{T}^T$ is generated using an iterative linear solver. Using the result of [44], which claims that the condition number of $\tilde{T}\tilde{A}\tilde{D}^2\tilde{A}^T\tilde{T}^T$ is uniformly bounded regardless of $\tilde{D}$, we obtain a uniform bound (depending only on $\tilde{A}$) on the number of inner iterations performed by the iterative linear solver to find a desirable approximate solution to the ANE (see Theorem 3.3.5).

Since the iterative linear solver can only generate an approximate solution to the ANE, it is clear that not all equations of the Newton system, which determines the primal-dual search direction, can be satisfied simultaneously. In the context of LP, Monteiro and O'Neal [42] proposed a recipe to compute an inexact primal-dual search direction so that the equations of the Newton system corresponding to the primal and dual residuals were both satisfied. In the context of CQP, such an approach is no longer possible. Instead, we propose a way to compute an inexact primal-dual search direction so that the equation corresponding to the primal residual is satisfied exactly, while the one corresponding to the dual residual contains a manageable error which allows us to establish a polynomial bound on the number of outer iterations of our method. Interestingly, the presence of this error on the dual residual equation implies that the primal and dual residuals go to zero at different rates. This is a unique feature of the convergence analysis of our algorithm in that it contrasts with the analysis of other interior-point PDIPF algorithms, where the primal and dual residuals are required to go to zero at the same rate.

The use of inexact search directions in interior-point methods has been extensively studied in the context of cone programming problems (see e.g. [4, 5, 17, 31, 39, 47, 73]). Moreover, the use of iterative linear solvers to compute the primal-dual Newton search directions of interior-point path following algorithms has also been extensively investigated in [4, 6, 8, 17, 31, 47, 48, 49, 53]. For feasibility problems of the form $\{x \in \mathcal{H}_1 : \mathcal{A}x = b, x \in \mathcal{C}\}$, where $\mathcal{H}_1$ and $\mathcal{H}_2$ are Hilbert spaces, $\mathcal{C} \subseteq \mathcal{H}_1$ is a closed convex cone satisfying some mild assumptions, and $\mathcal{A} : \mathcal{H}_1 \to \mathcal{H}_2$ is a continuous linear operator, Renegar [51] has proposed an interior-point method where the Newton system that determines the search directions is approximately solved by performing a uniformly bounded number of iterations of the conjugate gradient (CG) method. To our knowledge, no one has used the ANE system in the context of CQP to obtain either an exact or inexact primal-dual search direction.

The first part of this chapter is organized as follows. In Subsection 3.1.1, we give the terminology and notation which will be used throughout the first part of the chapter. Section 3.2 describes the outer iteration framework for our algorithm and the complexity results we have obtained for it, along with presenting the ANE as a means to determine the search direction. In Section 3.3, we discuss the use of iterative linear solvers to obtain a suitable approximate solution to the ANE and the construction of an inexact search direction based on this solution. Section 3.4 gives the proofs of the results presented in Sections 3.2 and 3.3. Finally, we present some concluding remarks in Section 3.5.

3.1.1 Terminology and Notation

Throughout this chapter, upper-case Roman letters denote matrices, lower-case Roman letters denote vectors, and lower-case Greek letters denote scalars. We let $\mathbb{R}^n$, $\mathbb{R}^n_+$ and $\mathbb{R}^n_{++}$ denote the set of n- vectors having real, nonnegative real, and positive real components, respectively. Also, we let $\mathbb{R}^{m\times n}$ denote the set of $m \times n$ matrices with real entries. For a vector $v \in \mathbb{R}^n$, we let $|v|$ denote the vector whose ith component is $|v_i|$, for every $i = 1,\ldots,n$, and we let $\mathrm{Diag}(v)$ denote the diagonal matrix whose i-th diagonal element is v_i, for every $i = 1,\ldots,n$. In addition, given vectors $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$, we denote by (u, v) the vector $(u^T, v^T)^T \in \mathbb{R}^{m+n}$.

Certain matrices bear special notation, namely the matrices X, ΔX, S, D, and $\tilde{D}$. These matrices are the diagonal matrices corresponding to the vectors x, Δx, s, d, and $\tilde{d}$, respectively, as described in the previous paragraph. The symbol 0 will be used to denote a scalar, vector, or matrix of all zeroes; its dimensions should be clear from the context. Also, we denote by e the vector of all 1's, and by I the identity matrix; their dimensions should be clear from the context.

For a symmetric positive definite matrix W, we denote its condition number by $\kappa(W)$, i.e. its maximum eigenvalue divided by its minimum eigenvalue. We will denote sets by upper-case script Roman letters (e.g. $\mathcal{B}$, $\mathcal{N}$). For a finite set $\mathcal{B}$, we denote its cardinality by $|\mathcal{B}|$. Given a matrix $A \in \mathbb{R}^{m\times n}$ and an ordered set $\mathcal{B} \subseteq \{1,\ldots,n\}$, we let $A_{\mathcal{B}}$ denote the submatrix whose columns are $\{A_i : i \in \mathcal{B}\}$ arranged in the same order as $\mathcal{B}$. Similarly, given a vector $v \in \mathbb{R}^n$ and an ordered set $\mathcal{B} \subseteq \{1,\ldots,n\}$, we let $v_{\mathcal{B}}$ denote the subvector consisting of the elements $\{v_i : i \in \mathcal{B}\}$ arranged in the same order as $\mathcal{B}$.

We will use several different norms throughout the chapter. For a vector $z \in \mathbb{R}^n$, $\|z\| = \sqrt{z^T z}$ is the Euclidian norm, $\|z\|_1 = \sum_{i=1}^n |z_i|$ is the "1-norm", and $\|z\|_\infty = \max_{i=1,\ldots,n} |z_i|$ is the "infinity norm". For a matrix $V \in \mathbb{R}^{m\times n}$, $\|V\|$ denotes the operator norm associated with the Euclidian norm: $\|V\| = \max_{z:\|z\|=1} \|Vz\|$. Finally, $\|V\|_F$ denotes the Frobenius norm: $\|V\|_F = (\sum_{i=1}^m \sum_{j=1}^n V_{ij}^2)^{1/2}$.

3.2 CQP Algorithm Using the MWB: Outer Iteration Framework

In this section, we introduce our PDIPF algorithm based on a class of inexact search directions and discuss its iteration complexity. This section is divided into two subsections. In Subsection 3.2.1, we discuss an exact PDIPF algorithm, which will serve as the basis for the inexact PDIPF algorithm given in Subsection 3.2.2, and we give its iteration complexity result. We also present an approach based on the ANE to determine the Newton search direction for the exact algorithm. To motivate the class of inexact search directions used by our inexact PDIPF algorithm, we describe in Subsection 3.2.2 a framework for computing an inexact search direction based on an approximate solution to the ANE. We then introduce the class of inexact search directions, state a PDIPF algorithm based on it, and give its iteration complexity result.

3.2.1 An Exact PDIPF Algorithm and the ANE

Consider the following primal-dual pair of CQP problems:

$$\min_x \quad \left\{ \frac{1}{2} x^T V E^2 V^T x + c^T x : \quad Ax = b, \ x \geq 0 \right\}, \tag{78}$$

$$\max_{(\hat{x},s,y)} \quad \left\{ -\frac{1}{2} \hat{x}^T V E^2 V^T \hat{x} + b^T y : \quad A^T y + s - V E^2 V^T \hat{x} = c, \ s \geq 0 \right\}, \tag{79}$$

where the data are $V \in \mathbb{R}^{n \times l}$, $E \in \mathrm{Diag}(\mathbb{R}^l_{++})$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, and the decision variables are $x \in \mathbb{R}^n$ and $(\hat{x}, s, y) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m$. We observe that the Hessian matrix Q is already given in factored form $Q = V E^2 V^T$.

It is well-known that if x^* is an optimal solution for (78) and $(\hat{x}^*, s^*, y^*)$ is an optimal solution for (79), then (x^*, s^*, y^*) is also an optimal solution for (79). Now, let $\mathcal{S}$ denote the set of all vectors $w := (x, s, y, z) \in \mathbb{R}^{2n+m+l}$ satisfying

$$Ax = b, \quad x \geq 0, \tag{80}$$

$$A^T y + s + Vz = c, \quad s \geq 0, \tag{81}$$

$$Xs = 0, \tag{82}$$

$$EV^T x + E^{-1} z = 0. \tag{83}$$

It is clear that $w \in \mathcal{S}$ if and only if x is optimal for (78), (x, s, y) is optimal for (79), and $z = -E^2V^Tx$. (Throughout this chapter, the symbol w will always denote the quadruple (x, s, y, z), where the vectors lie in the appropriate dimensions; similarly, $\Delta w = (\Delta x, \Delta s, \Delta y, \Delta z)$, $w^k = (x^k, s^k, y^k, z^k)$, $\bar{w} = (\bar{x}, \bar{s}, \bar{y}, \bar{z})$, etc.)

We observe that the presentation of the PDIPF algorithm based on exact Newton search directions in this subsection differs from the classical way of presenting it in that we introduce an additional variable z as above. Clearly, it is easy to see that the variable z is completely redundant and can be eliminated, thereby reducing the method described below to the usual way of presenting it. The main reason for introducing the variable z is due to the development of the ANE presented at the end of this subsection.

We will make the following two assumptions throughout the chapter:

Assumption 3 *A has full row rank.*

Assumption 4 *The set $\mathcal{S}$ is nonempty.*

For a point $w \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$, let us define

$$\mu \;:=\; \mu(w) \;=\; x^Ts/n, \tag{84}$$

$$r_p \;:=\; r_p(w) \;=\; Ax - b, \tag{85}$$

$$r_d \;:=\; r_d(w) \;=\; A^Ty + s + Vz - c, \tag{86}$$

$$r_V \;:=\; r_V(w) \;=\; EV^Tx + E^{-1}z, \tag{87}$$

$$r \;:=\; r(w) \;=\; (r_p(w), r_d(w), r_V(w)). \tag{88}$$

Moreover, given $\gamma \in (0, 1)$ and an initial point $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$, we define the following neighborhood of the central path:

$$\mathcal{N}_{w^0}(\gamma) := \left\{ w \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l} : Xs \geq (1-\gamma)\mu e,\; r = \eta r^0 \text{ for some } 0 \leq \eta \leq \min\left[1, \frac{\mu}{\mu_0}\right] \right\}, \tag{89}$$

where $r := r(w)$, $r^0 := r(w^0)$, $\mu := \mu(w)$, and $\mu_0 := \mu(w^0)$.

We are now ready to state the PDIPF algorithm based on exact Newton search directions.

Exact PDIPF Algorithm

1. **Start:** Let $\epsilon > 0$ and $0 < \underline{\sigma} \le \overline{\sigma} < 1$ be given. Let $\gamma \in (0,1)$ and $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ be such that $w^0 \in \mathcal{N}_{w^0}(\gamma)$. Set $k = 0$.

2. **While** $\mu_k := \mu(w^k) > \epsilon$ **do**

   (a) Let $w := w^k$ and $\mu := \mu_k$; choose $\sigma := \sigma_k \in [\underline{\sigma}, \overline{\sigma}]$.

   (b) Let $\Delta w = (\Delta x, \Delta s, \Delta y, \Delta z)$ denote the solution of the linear system

   $$A\Delta x = -r_p, \tag{90}$$

   $$A^T \Delta y + \Delta s + V\Delta z = -r_d, \tag{91}$$

   $$X\Delta s + S\Delta x = -Xs + \sigma\mu e, \tag{92}$$

   $$EV^T\Delta x + E^{-1}\Delta z = -r_V. \tag{93}$$

   (c) Let $\tilde{\alpha} = \operatorname{argmax}\{\alpha \in [0,1] : w + \alpha'\Delta w \in \mathcal{N}_{w^0}(\gamma),\ \forall \alpha' \in [0,\alpha]\}$.

   (d) Let $\bar{\alpha} = \operatorname{argmin}\left\{(x + \alpha\Delta x)^T(s + \alpha\Delta s) : \alpha \in [0,\tilde{\alpha}]\right\}$.

   (e) Let $w^{k+1} = w + \bar{\alpha}\Delta w$, and set $k \leftarrow k+1$.

   End (while)

A proof of the following result, under slightly different assumptions, can be found in [72].

Theorem 3.2.1 *Assume that the constants γ, $\underline{\sigma}$, and $\overline{\sigma}$ are such that*

$$\max\left\{\gamma^{-1}, (1-\gamma)^{-1}, \underline{\sigma}^{-1}, (1-\overline{\sigma})^{-1}\right\} = \mathcal{O}(1),$$

and that the initial point $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfies $(x^0, s^0) \ge (x^*, s^*)$ for some $w^* \in \mathcal{S}$. Then, the Exact PDIPF Algorithm finds an iterate $w^k \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfying $\mu_k \le \epsilon\mu_0$ and $\|r^k\| \le \epsilon\|r^0\|$ within $\mathcal{O}(n^2\log(1/\epsilon))$ iterations.

A few approaches have been suggested in the literature for computing the Newton search direction (90)-(93). Instead of using one of them, we will discuss below a new approach, referred to in this chapter as the ANE approach, that we believe to be suitable not only for direct solvers but especially for iterative linear solvers as we will see in Section 3.3.

Let us begin by defining the following matrices:

$$\begin{aligned} D &:= X^{1/2}S^{-1/2}, && (94)\\ \tilde{D} &:= \begin{pmatrix} D & 0 \\ 0 & E^{-1} \end{pmatrix} \in \mathbb{R}^{(n+l)\times(n+l)}, && (95)\\ \tilde{A} &:= \begin{pmatrix} A & 0 \\ V^T & I \end{pmatrix} \in \mathbb{R}^{(m+l)\times(n+l)}. && (96) \end{aligned}$$

Suppose that we first solve the following system of equations for $(\Delta y, \Delta z)$:

$$\tilde{A}\tilde{D}^2\tilde{A}^T \begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} = \tilde{A} \begin{pmatrix} x - \sigma\mu S^{-1}e - D^2 r_d \\ 0 \end{pmatrix} + \begin{pmatrix} -r_p \\ -E^{-1}r_V \end{pmatrix} =: h. \tag{97}$$

This system is what we refer to as the ANE. Next, we obtain Δs and Δx according to:

$$\begin{aligned} \Delta s &= -r_d - A^T\Delta y - V\Delta z, && (98)\\ \Delta x &= -D^2\Delta s - x + \sigma\mu S^{-1}e. && (99) \end{aligned}$$

Clearly, the search direction $\Delta w = (\Delta x, \Delta s, \Delta y, \Delta z)$ computed as above satisfies (91) and (92) in view of (98) and (99). Moreover, it also satisfies (90) and (93) due to the fact that by (95), (96), (97), (98) and (99), we have that

$$\begin{aligned} \tilde{A}\begin{pmatrix} \Delta x \\ E^{-2}\Delta z \end{pmatrix} &= \tilde{A}\begin{pmatrix} -D^2\Delta s - x + \sigma\mu S^{-1}e \\ E^{-2}\Delta z \end{pmatrix}\\ &= \tilde{A}\begin{pmatrix} D^2 r_d + D^2A^T\Delta y + D^2V\Delta z - x + \sigma\mu S^{-1}e \\ E^{-2}\Delta z \end{pmatrix}\\ &= \tilde{A}\begin{pmatrix} D^2A^T\Delta y + D^2V\Delta z \\ E^{-2}\Delta z \end{pmatrix} + \tilde{A}\begin{pmatrix} D^2 r_d - x + \sigma\mu S^{-1}e \\ 0 \end{pmatrix}\\ &= \tilde{A}\tilde{D}^2\tilde{A}^T\begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} + \tilde{A}\begin{pmatrix} D^2 r_d - x + \sigma\mu S^{-1}e \\ 0 \end{pmatrix}\\ &= \begin{pmatrix} -r_p \\ -E^{-1}r_V \end{pmatrix}. && (100) \end{aligned}$$

Theorem 3.2.1 assumes that Δw is the exact solution of (97), which is usually obtained by computing the Cholesky factorization of the coefficient matrix of the ANE. In this chapter, we will consider a variant of the Exact PDIPF Algorithm whose search directions are approximate solutions of (97) and ways of determining these inexact search directions by means of a suitable preconditioned iterative linear solver.

3.2.2 An Inexact PDIPF algorithm for CQP

In this subsection, we describe a PDIPF algorithm based on a family of search directions that are approximate solutions to (90)–(93) and discuss its iteration complexity properties.

Clearly, an approximate solution to the ANE can only yield an approximate solution to (90)–(93). In order to motivate the class of inexact search directions used by the PDIPF algorithm presented in this subsection, we present a framework for obtaining approximate solutions to (90)–(93) based on an approximate solution to the ANE.

Suppose that the ANE is solved only inexactly, i.e. that the vector $(\Delta y, \Delta z)$ satisfies

$$\tilde{A}\tilde{D}^2\tilde{A}^T \begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} = h + f \tag{101}$$

for some error vector f. If Δs and Δx were computed by (98) and (99), respectively, then it is clear that the search direction Δw would satisfy (91) and (92). However, (90) and (93) would not be satisfied, since by an argument similar to (100), we would have that

$$\tilde{A}\begin{pmatrix} \Delta x \\ E^{-2}\Delta z \end{pmatrix} = \ldots = \tilde{A}\tilde{D}^2\tilde{A}^T \begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} + \tilde{A}\begin{pmatrix} D^2 r_d - x + \sigma\mu S^{-1}e \\ 0 \end{pmatrix} = \begin{pmatrix} -r_p \\ -E^{-1}r_V \end{pmatrix} + f.$$

Instead, suppose we use (98) to determine Δs as before, but now we determine Δx as

$$\Delta x = -D^2\Delta s - x + \sigma\mu S^{-1}e - S^{-1}p, \tag{102}$$

where the correction vector $p \in \mathbb{R}^n$ will be required to satisfy some conditions which we will now describe.

To motivate the conditions on p, we note that (98), (101), and (102) imply that

$$\begin{aligned}
&\tilde{A}\begin{pmatrix} \Delta x \\ E^{-2}\Delta z \end{pmatrix} + \begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} \\
&= \tilde{A}\begin{pmatrix} -D^2\Delta s - x + \sigma\mu S^{-1}e - S^{-1}p \\ E^{-2}\Delta z \end{pmatrix} + \begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} \\
&= \tilde{A}\begin{pmatrix} D^2 r_d + D^2 A^T\Delta y + D^2 V\Delta z - x + \sigma\mu S^{-1}e - S^{-1}p \\ E^{-2}\Delta z \end{pmatrix} + \begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} \\
&= \tilde{A}\tilde{D}^2\begin{pmatrix} A^T\Delta y + V\Delta z \\ \Delta z \end{pmatrix} + \tilde{A}\begin{pmatrix} D^2 r_d - x + \sigma\mu S^{-1}e \\ 0 \end{pmatrix} - \tilde{A}\begin{pmatrix} S^{-1}p \\ 0 \end{pmatrix} + \begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} \\
&= \tilde{A}\tilde{D}^2\tilde{A}^T\begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} + \tilde{A}\begin{pmatrix} D^2 r_d - x + \sigma\mu S^{-1}e \\ 0 \end{pmatrix} - \tilde{A}\begin{pmatrix} S^{-1}p \\ 0 \end{pmatrix} + \begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} \\
&= f - \tilde{A}\begin{pmatrix} S^{-1}p \\ 0 \end{pmatrix}. \qquad (103)
\end{aligned}$$

Based on the above equation, one is naturally tempted to choose p so that the right hand side of (103) is zero, and consequently (90) and (93) are satisfied exactly. However, the existence of such p cannot be guaranteed and, even if it exists, its magnitude might not be sufficiently small to yield a search direction which is suitable for the development of a polynomially convergent algorithm. Instead, we consider an alternative approach where p is chosen so that the first component of (103) is zero and the second component is small. More specifically, by partitioning $f = (f_1, f_2) \in \mathbb{R}^m \times \mathbb{R}^l$, we choose $p \in \mathbb{R}^n$ such that

$$AS^{-1}p = f_1. \qquad (104)$$

It is clear that p is not uniquely defined. Note that (96) implies that (104) is equivalent to

$$f \;=\; \tilde{A}\begin{pmatrix} S^{-1}p \\ E^{-1}q \end{pmatrix}, \qquad (105)$$

where $q := E(f_2 - V^T S^{-1}p)$. Then, using (96), (103), and (105), we conclude that

$$\begin{aligned}\tilde{A}\begin{pmatrix}\Delta x\\ E^{-2}\Delta z\end{pmatrix}+\begin{pmatrix}r_p\\ E^{-1}r_V\end{pmatrix} &= f-\tilde{A}\begin{pmatrix}S^{-1}p\\ E^{-1}q\end{pmatrix}+\tilde{A}\begin{pmatrix}0\\ E^{-1}q\end{pmatrix}\\ &= \tilde{A}\begin{pmatrix}0\\ E^{-1}q\end{pmatrix}=\begin{pmatrix}0\\ E^{-1}q\end{pmatrix},\end{aligned} \tag{106}$$

from which we see that the first component of (103) is set to 0 and the second component is exactly $E^{-1}q$.

In view of (98), (102), and (106), the above construction yields a search direction Δw satisfying the following modified Newton system of equations:

$$A\Delta x = -r_p, \tag{107}$$

$$A^T\Delta y + \Delta s + V\Delta z = -r_d, \tag{108}$$

$$X\Delta s + S\Delta x = -Xs + \sigma\mu e - p, \tag{109}$$

$$EV^T\Delta x + E^{-1}\Delta z = -r_V + q. \tag{110}$$

As far as the outer iteration complexity analysis of our algorithm is concerned, all we require of our inexact search directions is that they satisfy (107)–(110) and that p and q be relatively small in the following sense:

Definition 1 *Given a point $w \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ and positive scalars τ_p and τ_q, an inexact direction Δw is referred to as a (τ_p, τ_q)-*search direction *if it satisfies (107)–(110) for some p and q satisfying $\|p\|_\infty \le \tau_p\mu$ and $\|q\| \le \tau_q\sqrt{\mu}$, where μ is given by (84).*

We next define a generalized central path neighborhood which is used by our inexact PDIPF algorithm. Given a starting point $w^0 \in \mathbb{R}^{2n}_{++}\times\mathbb{R}^{m+l}$ and parameters $\eta \ge 0$, $\gamma \in [0,1]$, and $\theta > 0$, define the following set:

$$\mathcal{N}_{w^0}(\eta,\gamma,\theta) = \left\{ w \in \mathbb{R}^{2n}_{++}\times\mathbb{R}^{m+l} : \begin{array}{cc} Xs \ge (1-\gamma)\mu e, & (r_p, r_d) = \eta(r_p^0, r_d^0), \\ \|r_V - \eta r_V^0\| \le \theta\sqrt{\mu}, & \eta \le \mu/\mu_0 \end{array} \right\}, \tag{111}$$

where $\mu = \mu(w)$, $\mu_0 = \mu(w^0)$, $r = r(w)$ and $r^0 = r(w^0)$. The generalized central path neighborhood is then given by

$$\mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta) \;=\; \bigcup_{\eta \in [0,1]} \mathcal{N}_{w^0}(\eta, \gamma, \theta). \tag{112}$$

We observe that the neighborhood given by (112) agrees with the neighborhood given by (89) when $\theta = 0$.

We are now ready to state our inexact PDIPF algorithm.

Inexact PDIPF Algorithm:

1. **Start:** Let $\epsilon > 0$ and $0 < \underline{\sigma} \leq \overline{\sigma} < 4/5$ be given. Choose $\gamma \in (0,1)$, $\theta > 0$ and $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ such that $w^0 \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$. Set $k = 0$.

2. **While** $\mu_k := \mu(w^k) > \epsilon$ **do**

   (a) Let $w := w^k$ and $\mu := \mu_k$; choose $\sigma \in [\underline{\sigma}, \overline{\sigma}]$.

   (b) Set

$$\begin{aligned} \tau_p &= \gamma\sigma/4 \quad \text{and} & (113)\\ \tau_q &= \left[\sqrt{1 + (1-0.5\gamma)\,\sigma} - 1\right]\theta. & (114) \end{aligned}$$

   (c) Set $r_p = Ax - b$, $r_d = A^T y + s + Vz - c$, $r_V = EV^T x + E^{-1} z$, and $\eta = \|r_p\|/\|r_p^0\|$.

   (d) Compute a (τ_p, τ_q)-search direction Δw.

   (e) Compute $\tilde{\alpha} := \operatorname{argmax}\{\alpha \in [0,1] : w + \alpha' \Delta w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta),\ \forall \alpha' \in [0, \alpha]\}$.

   (f) Compute $\bar{\alpha} := \operatorname{argmin}\{(x + \alpha \Delta x)^T (s + \alpha \Delta s) : \alpha \in [0, \tilde{\alpha}]\}$.

   (g) Let $w^{k+1} = w + \bar{\alpha}\Delta w$, and set $k \leftarrow k+1$.

   End (while)

The following result gives a bound on the number of iterations needed by the Inexact PDIPF Algorithm to obtain an ϵ-solution to the KKT conditions (80)–(83). Its proof will be given in Subsection 3.4.2.

Theorem 3.2.2 *Assume that the constants γ, $\underline{\sigma}$, $\overline{\sigma}$ and θ are such that*

$$\max\left\{\gamma^{-1}\,,\,(1-\gamma)^{-1}\,,\,\underline{\sigma}^{-1}\,,\,\left(1-\frac{5}{4}\overline{\sigma}\right)^{-1}\right\} = \mathcal{O}(1),\ \ \theta = \mathcal{O}(\sqrt{n}), \tag{115}$$

and that the initial point $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfies $(x^0, s^0) \geq (x^*, s^*)$ for some $w^* \in \mathcal{S}$. Then, the Inexact PDIPF Algorithm generates an iterate $w^k \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfying $\mu_k \leq \epsilon\mu_0$, $\|(r_p^k, r_d^k)\| \leq \epsilon\|(r_p^0, r_d^0)\|$, and $\|r_V^k\| \leq \epsilon\|r_V^0\| + \epsilon^{1/2}\theta\mu_0^{1/2}$ within $\mathcal{O}\left(n^2\log(1/\epsilon)\right)$ iterations.

3.3 CQP Algorithm using the MWB: Determining an Inexact Search Direction Via an Iterative Solver

The results in Subsection 3.2.2 assume we can obtain a (τ_p, τ_q)-search direction Δw, where τ_p and τ_q are given by (113) and (114), respectively. In this section, we will describe a way to obtain a (τ_p, τ_q)-search direction Δw using a uniformly bounded number of iterations of a suitable preconditioned iterative linear solver applied to the ANE. It turns out that the construction of this Δw is based on the recipe given at the beginning of Subsection 3.2.2, together with a specific choice of the perturbation vector p.

This section is divided into two subsections. In Subsection 3.3.1, we introduce the MWB preconditioner which will be used to precondition the ANE. In addition, we also introduce a family of iterative linear solvers used to solve the preconditioned ANE. Subsection 3.3.2 gives a specific approach for constructing a pair (p, q) satisfying (105), and an approximate solution to the ANE so that the recipe described at the beginning of Subsection 3.2.2 yields a (τ_p, τ_q)-search direction Δw. It also provides a uniform bound on the number of iterations that any member of the family of iterative linear solvers needs to perform to obtain such a direction Δw when applied to the preconditioned ANE.

3.3.1 MWB Preconditioner and a Family of Solvers

In this subsection we introduce the MWB preconditioner, and we discuss its use as a preconditioner in solving the ANE via a family of iterative linear solvers. Since the condition number of the ANE matrix $\tilde{A}\tilde{D}^2\tilde{A}^T$ may "blow up" for points w near an optimal solution, the direct application of a generic iterative linear solver for solving the ANE without first preconditioning it is generally not effective. We discuss a natural remedy to this problem which consists of using a preconditioner $\tilde{T}$, namely the MWB preconditioner, such that $\kappa(\tilde{T}\tilde{A}\tilde{D}^2\tilde{A}^T\tilde{T}^T)$ remains uniformly bounded regardless of the iterate w. Finally, we analyze the complexity of the resulting approach to obtain a suitable approximate solution to the ANE.

We start by describing the MWB preconditioner. Its construction essentially consists of building a basis B of $\tilde{A}$ which gives higher priority to the columns of $\tilde{A}$ corresponding to larger diagonal elements of $\tilde{D}$. More specifically, the MWB preconditioner is determined by the following algorithm:

Maximum Weight Basis Algorithm

```
Start: Given $\tilde{d} \in \mathbb{R}^{(n+l)}_{++}$, and $\tilde{A} \in \mathbb{R}^{(m+l)\times(n+l)}$ such that $\text{rank}(\tilde{A}) = m + l$,
    1. Order the elements of $\tilde{d}$ so that $\tilde{d}_1 \geq \ldots \geq \tilde{d}_{n+l}$; order the columns of $\tilde{A}$ accordingly.
    2. Let $\mathcal{B} = \emptyset$, $j = 1$.
    3. While $|\mathcal{B}| < m + l$ do
        (a) If $\tilde{A}_j$ is linearly independent of $\{\tilde{A}_i : i \in \mathcal{B}\}$, set $\mathcal{B} \leftarrow \mathcal{B} \cup \{j\}$.
        (b) $j \leftarrow j + 1$.
    4. Return to the original ordering of $\tilde{A}$ and $\tilde{d}$; determine the set $\mathcal{B}$ according to this
       ordering and set $\mathcal{N} := \{1, \ldots, n + l\} \backslash \mathcal{B}$.
    5. Set $B := \tilde{A}_{\mathcal{B}}$ and $\tilde{D}_{\mathcal{B}} := \text{Diag}(\tilde{d}_{\mathcal{B}})$.
    6. Let $\tilde{T} = \tilde{T}(\tilde{A}, \tilde{d}) := \tilde{D}_{\mathcal{B}}^{-1} B^{-1}$.
end
```

Note that the above algorithm can be applied to the matrix $\tilde{A}$ defined in (96) since this matrix has full row rank due to Assumption 1. The MWB preconditioner was originally proposed by Vaidya [63] and Resende and Veiga [53] in the context of the minimum cost network flow problem. In this case, $\tilde{A} = A$ is the node-arc incidence matrix of a connected digraph (with one row deleted to ensure that $\tilde{A}$ has full row rank), the entries of $\tilde{d}$ are weights on the edges of the graph, and the set $\mathcal{B}$ generated by the above algorithm defines a maximum spanning tree on the digraph. Oliveira and Sorensen [48] later proposed the use of this preconditioner for general matrices $\tilde{A}$. Boman et. al. [9] have proposed variants of the MWB preconditioner for diagonally dominant matrices, using the fact that they can be represented as $D_1 + AD_2A^T$, where D_1 and D_2 are nonnegative diagonal and positive diagonal matrices, respectively, and A is a node-arc incidence matrix.

For the purpose of stating the next result, we now introduce some notation. Let us define

$$\varphi_{\tilde{A}} \ := \ \max\{\|B^{-1}\tilde{A}\|_F : B \text{ is a basis of } \tilde{A}\}. \tag{116}$$

The constant $\varphi_{\tilde{A}}$ is related to the well-known condition number $\bar{\chi}_{\tilde{A}}$ (see [65]), defined as

$$\bar{\chi}_{\tilde{A}} \ := \ \sup\{\|\tilde{A}^T(\tilde{A}\tilde{E}\tilde{A}^T)^{-1}\tilde{A}\tilde{E}\| : \tilde{E} \in \mathrm{Diag}(\mathbb{R}^{(n+l)}_{++})\}.$$

Specifically, $\varphi_{\tilde{A}} \le (n+l)^{1/2}\bar{\chi}_{\tilde{A}}$, in view of the facts that $\|C\|_F \le (n+l)^{1/2}\,\|C\|$ for any matrix $C \in \mathbb{R}^{(m+l)\times(n+l)}$ and, as shown in [60] and [65],

$$\bar{\chi}_{\tilde{A}} \ = \ \max\{\|B^{-1}\tilde{A}\| : B \text{ is a basis of } \tilde{A}\}.$$

The following result, which establishes the theoretical properties of the MWB preconditioner, follows as a consequence of Lemmas 2.1 and 2.2 of [44].

Proposition 3.3.1 *Let $\tilde{T} = \tilde{T}(\tilde{A}, \tilde{d})$ be the preconditioner determined according to the Maximum Weight Basis Algorithm, and define $W := \tilde{T}\tilde{A}\tilde{D}^2\tilde{A}^T\tilde{T}^T$. Then, $\|\tilde{T}\tilde{A}\tilde{D}\| \le \varphi_{\tilde{A}}$ and $\kappa(W) \le \varphi^2_{\tilde{A}}$.*

Note that the bound $\varphi^2_{\tilde{A}}$ on $\kappa(W)$ is independent of the diagonal matrix $\tilde{D}$ and depends only on $\tilde{A}$. This will allow us to obtain a uniform bound on the number of iterations needed by any member of the family of iterative linear solvers described below to obtain a suitable approximate solution of (97). This topic is the subject of the remainder of this subsection.

Instead of dealing directly with (97), we consider the application of an iterative linear solver to the preconditioned ANE:

$$Wu = v, \tag{117}$$

Table 3: Values of $c(\kappa)$ and $\psi(\kappa)$ for Well-Known Iterative Solvers

Solver	$c(\kappa)$	$\psi(\kappa)$
SD	$\sqrt{\kappa}$	$(\kappa+1)/2$
CG	$2\sqrt{\kappa}$	$(\sqrt{\kappa}+1)/2$

where

$$W := \tilde{T}\tilde{A}\tilde{D}^2\tilde{A}^T\tilde{T}^T, \qquad v := \tilde{T}h. \tag{118}$$

For the purpose of our analysis below, the only thing we will assume regarding the iterative linear solver when applied to (117) is that it generates a sequence of iterates $\{u^j\}$ such that

$$\|v - Wu^j\| \;\le\; c(\kappa)\left[1 - \frac{1}{\psi(\kappa)}\right]^j \|v - Wu^0\|, \quad \forall\, j = 0, 1, 2, \ldots, \tag{119}$$

where c and ψ are positive, nondecreasing functions of $\kappa \equiv \kappa(W)$.

Examples of solvers which satisfy (119) include the steepest descent (SD) and CG methods, with the following values for $c(\kappa)$ and $\psi(\kappa)$:

The justification for the table above follows from Section 7.6 and Exercise 10 of Section 8.8 of [35].

The following result gives an upper bound on the number of iterations required by any iterative linear solver satisfying (119) needs to perform to obtain a *ξ-approximate solution of (117)*, i.e. an iterate u^j such that $\|v - Wu^j\| \le \xi\sqrt{\mu}$ for some constant $\xi > 0$:

Proposition 3.3.2 *Let u^0 be an arbitrary starting point. Then, a generic iterative linear solver with a convergence rate given by (119) generates an iterate u^j satisfying $\|v - Wu^j\| \le \xi\sqrt{\mu}$ in*

$$\mathcal{O}\left(\psi(\kappa)\log\left(\frac{c(\kappa)\|v - Wu^0\|}{\xi\sqrt{\mu}}\right)\right) \tag{120}$$

iterations, where $\kappa \equiv \kappa(W)$.

Proof: Let j be any iteration such that $\|v - Wu^j\| > \xi\sqrt{\mu}$. We use relation (119) and the fact that $1 + \omega \le e^\omega$ for all $\omega \in \mathbb{R}$ to observe that

$$\xi\sqrt{\mu} \;<\; \|v - Wu^j\| \;\le\; c(\kappa)\left[1 - \frac{1}{\psi(\kappa)}\right]^j \|v - Wu^0\| \;\le\; c(\kappa)\exp\left\{\frac{-j}{\psi(\kappa)}\right\}\|v - Wu^0\|.$$

Rearranging the first and last terms of the inequality, it follows that

$$j \; < \; \psi(\kappa) \log \left(\frac{c(\kappa)\|v - Wu^0\|}{\xi\sqrt{\mu}} \right),$$

and the result is proven. ∎

From Proposition 3.3.2, it is clear that different choices of u^0 and ξ lead to different bounds on the number of iterations performed by the iterative linear solver. In Subsection 3.3.2, we will describe a suitable way of selecting u^0 and ξ so that (i) the bound (120) is independent of the iterate w and (ii) the approximate solution $\tilde{T}^T u^j$ of the ANE, together with a suitable pair (p, q), yields a (τ_p, τ_q)-search direction Δw through the recipe described in Subsection 3.2.2.

3.3.2 Computation of the Inexact Search Direction Δw

In this subsection, we use the results of Subsections 3.2.2 and 3.3.1 to build a (τ_p, τ_q)-search direction Δw, where τ_p and τ_q are given by (113) and (114), respectively. In addition, we describe a way of choosing u^0 and ξ which ensures that the number of iterations of an iterative linear solver satisfying (119) applied to the preconditioned ANE is uniformly bounded by a constant depending on n and $\varphi_{\tilde{A}}$.

Suppose that we solve (117) inexactly according to Subsection 3.3.1. Then our final solution u^j satisfies $Wu^j - v = \tilde{f}$ for some vector $\tilde{f}$. Letting

$$\begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} = \tilde{T}^T u^j, \tag{121}$$

we easily see from (118) that (101) is satisfied with $f := \tilde{T}^{-1}\tilde{f}$. We can then apply the recipe of Subsection 3.2.2 to this approximate solution, using the pair (p, q) which we will now describe.

First, note that (105) with f as defined above is equivalent to the system

$$\tilde{f} \; = \; \tilde{T}\tilde{A} \begin{pmatrix} S^{-1}p \\ E^{-1}q \end{pmatrix} \; = \; \tilde{T}\tilde{A}\tilde{D} \begin{pmatrix} (XS)^{-1/2} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} p \\ q \end{pmatrix}. \tag{122}$$

Now, let $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_{m+l})$ be the ordered set of basic indices computed by the MWB Algorithm applied to the pair $(\tilde{A}, \tilde{d})$ and note that, by step 6 of this algorithm, the $\mathcal{B}_i$-th column of $\tilde{T}\tilde{A}\tilde{D}$ is the ith unit vector for every $i = 1, \ldots, m+l$. Then, the vector $t \in \mathbb{R}^{n+l}$ defined as $t_{\mathcal{B}_i} = \tilde{f}_i$ for $i = 1, \ldots, m+l$ and $t_j = 0$ for every $j \notin \{\mathcal{B}_1, \ldots, \mathcal{B}_{m+l}\}$ clearly satisfies

$$\tilde{f} \ = \ \tilde{T}\tilde{A}\tilde{D}\, t. \tag{123}$$

We then obtain a pair $(p, q) \in \mathbb{R}^n \times \mathbb{R}^l$ satisfying (105) by defining

$$\begin{pmatrix} p \\ q \end{pmatrix} \ := \ \begin{pmatrix} (XS)^{1/2} & 0 \\ 0 & I \end{pmatrix} t. \tag{124}$$

It is clear from (124) and the fact that $\|t\| = \|\tilde{f}\|$ that

$$\|p\| \ \le \ \|XS\|^{1/2}\|\tilde{f}\|, \qquad \|q\| \ \le \ \|\tilde{f}\|. \tag{125}$$

As an immediate consequence of this relation, we obtain the following result.

Lemma 3.3.3 *Suppose that $w \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ and positive scalars τ_p and τ_q are given. Assume that u^j is a ξ-approximate solution of (117), or equivalently $\tilde{f} \le \xi\sqrt{\mu}$, where $\xi := \min\{n^{-1/2}\tau_p, \tau_q\}$. Let Δw be determined according to the recipe given in Subsection 3.2.2 using the approximate solution (121) and the pair (p, q) given by (124). Then Δw is a (τ_p, τ_q)-search direction.*

Proof: It is clear from the previous discussion that Δw and the pair (p, q) satisfy (107)–(110). Next, relation (125) and the facts that $\xi \le n^{-1/2}\tau_p$ and $\|XS\|^{1/2} \le \sqrt{n\mu}$ imply that

$$\|p\|_\infty \ \le \ \|p\| \ \le \ \|XS\|^{1/2}\|\tilde{f}\| \ \le \ \sqrt{n\mu}\,\xi\sqrt{\mu} \ \le \ \tau_p\mu.$$

Similarly, (125) and the fact that $\xi \le \tau_q$ imply that $\|q\| \le \tau_q\sqrt{\mu}$. Thus, Δw is a (τ_p, τ_q)-search direction as desired. ∎

Lemma (3.3.3) implies that, to construct a (τ_p, τ_q)-search direction Δw as in step 2(d) of the Inexact PDIPF Algorithm, it suffices to find a ξ-approximate solution to (117), where

$$\xi \ := \ \min\left\{\frac{\gamma\sigma}{4\sqrt{n}}\,, \ \left[\sqrt{1 + \left(1 - \frac{\gamma}{2}\right)\sigma} - 1\right]\theta\right\}. \tag{126}$$

We next describe a suitable way of selecting u^0 so that the number of iterations required by an iterative linear solver satisfying (119) to find a ξ-approximate solution of (117) can be uniformly bounded by a universal constant depending only on the quantities n and $\varphi_{\tilde{A}}$. First, compute a point $\bar{w} = (\bar{x}, \bar{s}, \bar{y}, \bar{z})$ such that

$$\tilde{A}\begin{pmatrix} \bar{x} \\ E^{-2}\bar{z} \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \qquad A^T\bar{y} + \bar{s} + V\bar{z} = c. \tag{127}$$

Note that vectors $\bar{x}$ and $\bar{z}$ satisfying the first equation in (127) can be easily computed once a basis of $\tilde{A}$ is available (e.g., the one computed by the Maximum Weight Basis Algorithm in the first outer iteration of the Inexact PDIPF Algorithm). Once $\bar{y}$ is arbitrarily chosen, a vector $\bar{s}$ satisfying the second equation of (127) is immediately available. We then define

$$u^0 \; = \; -\eta \tilde{T}^{-T} \begin{pmatrix} y^0 - \bar{y} \\ z^0 - \bar{z} \end{pmatrix}. \tag{128}$$

The following lemma gives a bound on the size of the initial residual $\|Wu^0 - v\|$. Its proof will be given in Subsection 3.4.1.

Lemma 3.3.4 *Assume that $\tilde{T} = \tilde{T}(\tilde{A}, \tilde{d})$ is given and that $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ and $\bar{w}$ are such that $(x^0, s^0) \geq |(\bar{x}, \bar{s})|$ and $(x^0, s^0) \geq (x^*, s^*)$ for some $w^* \in \mathcal{S}$. Further, assume that $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ for some $\gamma \in [0,1]$ and $\theta > 0$, and that W, v and u^0 are given by (118) and (128), respectively. Then, the initial residual in (119) satisfies $\|v - Wu^0\| \leq \Psi\sqrt{\mu}$, where*

$$\Psi \; := \; \left[\frac{7n + \theta^2/2}{\sqrt{1-\gamma}} + \theta\right]\varphi_{\tilde{A}}. \tag{129}$$

As an immediate consequence of Proposition 3.3.2 and Lemmas 3.3.3 and 3.3.4, we can bound the number of inner iterations required by an iterative linear solver satisfying (119) to yield a (τ_p, τ_q)-search direction Δw.

Theorem 3.3.5 *Assume that ξ is defined in (126), where σ, γ, θ are such that*

$$\max\{\sigma^{-1}, \gamma^{-1}, (1-\gamma)^{-1}, \theta, \theta^{-1}\}$$

is bounded by a polynomial of n. Assume also that $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ and $\bar{w}$ are such that $(x^0, s^0) \geq |(\bar{x}, \bar{s})|$ and $(x^0, s^0) \geq (x^*, s^*)$ for some $w^* \in \mathcal{S}$. Then, a generic iterative linear solver with a convergence rate given by (119) generates a ξ-approximate solution, which leads to a (τ_p, τ_q)-search direction Δw, in

$$\mathcal{O}\left(\psi(\varphi^2_{\tilde{A}}) \log\left(c(\varphi^2_{\tilde{A}}) n \varphi_{\tilde{A}}\right)\right) \tag{130}$$

iterations. As a consequence, the SD and CG methods generate this approximate solution u^j in $\mathcal{O}(\varphi^2_{\tilde{A}} \log(n\varphi_{\tilde{A}}))$ and $\mathcal{O}(\varphi_{\tilde{A}} \log(n\varphi_{\tilde{A}}))$ iterations, respectively.

Proof: The proof of the first part of Theorem 3.3.5 immediately follows from Propositions 3.3.1 and 3.3.2 and Lemmas 3.3.3 and 3.3.4. The proof of the second part of Theorem 3.3.5 follows immediately from Table 3 and Proposition 3.3.1. ∎

Using the results of Sections 3.2 and 3.3, we see that the number of "inner" iterations of an iterative linear solver satisfying (119) is uniformly bounded by a constant depending on n and $\varphi_{\tilde{A}}$, while the number of "outer" iterations in the Inexact PDIPF Algorithm is polynomially bounded by a constant depending on n and $\log \epsilon^{-1}$.

3.4 CQP Algorithm using the MWB: Technical Results

This section is devoted to the proofs of Lemma 3.3.4 and Theorem 3.2.2. Subsection 3.4.1 presents the proof of Lemma 3.3.4, and Subsection 3.4.2 presents the proof of Theorem 3.2.2.

3.4.1 Proof of Lemma 3.3.4

In this subsection, we will provide the proof of Lemma 3.3.4. We begin by establishing three technical lemmas.

Lemma 3.4.1 *Suppose that $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$, $w \in \mathcal{N}_{w^0}(\eta, \gamma, \theta)$ for some $\eta \in [0, 1]$, $\gamma \in [0, 1]$ and $\theta > 0$, and $w^* \in \mathcal{S}$. Then*

$$(x - \eta x^0 - (1-\eta)x^*)^T (s - \eta s^0 - (1-\eta)s^*) \quad \geq \quad -\frac{\theta^2}{4}\mu. \tag{131}$$

Proof: Let us define $\tilde{w} := w - \eta w^0 - (1-\eta)w^*$. Using the definitions of $\mathcal{N}_{w^0}(\eta,\gamma,\theta)$, r, and $\mathcal{S}$, we have that

$$\begin{aligned} A\tilde{x} &= 0 \\ A^T\tilde{y} + \tilde{s} + V\tilde{z} &= 0 \\ V^T\tilde{x} + E^{-2}\tilde{z} &= E^{-1}(r_V - \eta r_V^0). \end{aligned}$$

Multiplying the second relation by $\tilde{x}^T$ on the left, and using the first and third relations along with the fact that $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$, we see that

$$\begin{aligned} \tilde{x}^T\tilde{s} &= -\tilde{x}^T V\tilde{z} \;=\; [E^{-2}\tilde{z} - E^{-1}(r_V - \eta r_V^0)]^T\tilde{z} \;=\; \|E^{-1}\tilde{z}\|^2 - (E^{-1}\tilde{z})^T(r_V - \eta r_V^0) \\ &\geq \|E^{-1}\tilde{z}\|^2 - \|E^{-1}\tilde{z}\|\|r_V - \eta r_V^0\| \;=\; \left(\|E^{-1}\tilde{z}\| - \frac{\|r_V - \eta r_V^0\|}{2}\right)^2 - \frac{\|r_V - \eta r_V^0\|^2}{4} \\ &\geq -\frac{\|r_V - \eta r_V^0\|^2}{4} \;\geq\; -\frac{\theta^2}{4}\mu. \end{aligned}$$

■

Lemma 3.4.2 *Suppose that $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ such that $(x^0, s^0) \geq (x^*, s^*)$ for some $w^* \in \mathcal{S}$. Then, for any $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$ with $\eta \in [0,1]$, $\gamma \in [0,1]$ and $\theta > 0$, we have*

$$\eta(x^Ts^0 + s^Tx^0) \leq \left(3n + \frac{\theta^2}{4}\right)\mu. \tag{132}$$

Proof: Using the fact $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$ and (131), we obtain

$$\begin{aligned} & x^Ts - \eta(x^Ts^0 + s^Tx^0) + \eta^2 x^{0^T}s^0 - (1-\eta)(x^Ts^* + s^Tx^*) \\ & +\eta(1-\eta)(x^{*^T}s^0 + s^{*^T}x^0) + (1-\eta)^2 x^{*^T}s^* \;\;\geq\;\; -\frac{\theta^2}{4}\mu. \end{aligned}$$

Rearranging the terms in this equation, and using the facts that $\eta \leq x^Ts/x^{0^T}s^0$, $x^{*^T}s^* = 0$,

$(x,s) \geq 0$, $(x^*,s^*) \geq 0$, $(x^0,s^0) > 0$, $\eta \in [0,1]$, $x^* \leq x^0$, and $s^* \leq s^0$, we conclude that

$$\begin{aligned}
\eta(x^Ts^0 + s^Tx^0) &\leq \eta^2 {x^0}^T s^0 + x^Ts + \eta(1-\eta)({x^*}^T s^0 + {s^*}^T x^0) + \frac{\theta^2}{4}\mu \\
&\leq \eta^2 {x^0}^T s^0 + x^Ts + 2\eta(1-\eta){x^0}^T s^0 + \frac{\theta^2}{4}\mu \\
&\leq 2\eta {x^0}^T s^0 + x^Ts + \frac{\theta^2}{4}\mu \\
&\leq 3x^Ts + \frac{\theta^2}{4}\mu \;=\; \left(3n + \frac{\theta^2}{4}\right)\mu.
\end{aligned}$$

■

Lemma 3.4.3 *Suppose $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$, $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$ for some $\eta \in [0,1]$, $\gamma \in [0,1]$ and $\theta > 0$, and $\bar{w}$ satisfies (127). Let W, v and u^0 be given by (118) and (128), respectively. Then,*

$$Wu^0 - v \;=\; \tilde{T}\tilde{A}\begin{pmatrix} -x + \sigma\mu S^{-1}e + \eta(x^0 - \bar{x}) + \eta D^2(s^0 - \bar{s}) \\ E^{-1}(r_V - \eta r_V^0) \end{pmatrix}. \tag{133}$$

Proof: Using the fact that $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$ along with (96), (111) and (127), we easily obtain that

$$\begin{aligned}
\begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} &= \begin{pmatrix} \eta r_p^0 \\ \eta E^{-1}r_V^0 + E^{-1}(r_V - \eta r_V^0) \end{pmatrix} \\
&= \eta\tilde{A}\begin{pmatrix} x^0 - \bar{x} \\ E^{-2}(z^0 - \bar{z}) \end{pmatrix} + \tilde{A}\begin{pmatrix} 0 \\ E^{-1}(r_V - \eta r_V^0) \end{pmatrix}, &(134) \\
s^0 - \bar{s} &= -A^T(y^0 - \bar{y}) - V(z^0 - \bar{z}) + r_d^0. &(135)
\end{aligned}$$

Using relations (95), (96), (118), (111), (128), (134) and (135), we obtain

$$\begin{aligned}
Wu^0 - v &= \tilde{T}\tilde{A}\tilde{D}^2\tilde{A}^T\tilde{T}^T u^0 - \tilde{T}\tilde{A}\begin{pmatrix} x - \sigma\mu S^{-1}e - D^2 r_d \\ 0 \end{pmatrix} + \tilde{T}\begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} \\
&= -\eta\tilde{T}\tilde{A}\tilde{D}^2\tilde{A}^T\begin{pmatrix} y^0 - \bar{y} \\ z^0 - \bar{z} \end{pmatrix} - \tilde{T}\tilde{A}\begin{pmatrix} x - \sigma\mu S^{-1}e - \eta D^2 r_d^0 \\ 0 \end{pmatrix} + \tilde{T}\begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} \\
&= -\eta\tilde{T}\tilde{A}\begin{pmatrix} D^2A^T(y^0 - \bar{y}) + D^2V(z^0 - \bar{z}) - D^2 r_d^0 \\ E^{-2}(z^0 - \bar{z}) \end{pmatrix} \\
&\quad - \tilde{T}\tilde{A}\begin{pmatrix} x - \sigma\mu S^{-1}e \\ 0 \end{pmatrix} + \tilde{T}\begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix}, \\
&= -\eta\tilde{T}\tilde{A}\begin{pmatrix} -D^2(s^0 - \bar{s}) \\ E^{-2}(z^0 - \bar{z}) \end{pmatrix} - \tilde{T}\tilde{A}\begin{pmatrix} x - \sigma\mu S^{-1}e \\ 0 \end{pmatrix} \\
&\quad + \eta\tilde{T}\tilde{A}\begin{pmatrix} x^0 - \bar{x} \\ E^{-2}(z^0 - \bar{z}) \end{pmatrix} + \tilde{T}\tilde{A}\begin{pmatrix} 0 \\ E^{-1}(r_V - \eta r_V^0) \end{pmatrix},
\end{aligned}$$

which yields equation (133), as desired. ▪

We now turn to the proof of Lemma 3.3.4.

Proof of Lemma 3.3.4: Since $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$, we have that $x_i s_i \geq (1-\gamma)\mu$ for all i, which implies

$$\|(XS)^{-1/2}\| \;\leq\; \frac{1}{\sqrt{(1-\gamma)\mu}}. \tag{136}$$

Note that $\|Xs - \sigma\mu e\|$, when viewed as a function of $\sigma \in [0,1]$, is convex. Hence, it is maximized at one of its endpoints, which, together with the facts $\|Xs - \mu e\| < \|Xs\|$ and $\sigma \in [\underline{\sigma}, \overline{\sigma}] \subset [0,1]$, immediately implies that

$$\|Xs - \sigma\mu e\| \;\leq\; \|Xs\| \;\leq\; \|Xs\|_1 \;=\; x^T s \;=\; n\mu. \tag{137}$$

Using the fact that $(x^0, s^0) \geq |(\bar{x}, \bar{s})|$ together with Lemma 3.4.2, we obtain that

$$\begin{aligned}
\eta\|S(x^0 - \bar{x}) + X(s^0 - \bar{s})\| &\leq \eta\big\{\|S(x^0 - \bar{x})\| + \|X(s^0 - \bar{s})\|\big\} \;\leq\; 2\eta\big\{\|Sx^0\| + \|Xs^0\|\big\} \\
&\leq 2\eta(x^T s^0 + x^T s^0) \;\leq\; \left(6n + \frac{\theta^2}{2}\right)\mu. \qquad (138)
\end{aligned}$$

Since $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$, there exists $\eta \in [0,1]$ such that $w \in \mathcal{N}_{w^0}(\eta, \gamma, \theta)$. It is clear that the requirements of Lemma 3.4.3 are met, so equation (133) holds. By (94), (95) and (133), we see that

$$\begin{aligned}
\|v - Wu^0\| &= \left\| \tilde{T}\tilde{A}\tilde{D} \begin{pmatrix} (XS)^{-1/2}\{Xs - \sigma\mu e - \eta[S(x^0 - \bar{x}) + X(s^0 - \bar{s})]\} \\ r_V - \eta r_V^0 \end{pmatrix} \right\| \\
&\le \|\tilde{T}\tilde{A}\tilde{D}\| \Big\{ \|(XS)^{-1/2}\| \Big[\|Xs - \sigma\mu e\| + \eta\|X(s^0 - \bar{s}) + S(x^0 - \bar{x})\| \Big] \\
&\quad + \|r_V - \eta r_V^0\| \Big\}, \\
&\le \varphi_{\tilde{A}} \left\{ \frac{1}{\sqrt{(1-\gamma)\mu}} \left[n\mu + \left(6n + \frac{\theta^2}{2}\right)\mu \right] + \theta\sqrt{\mu} \right\} \;=\; \Psi\sqrt{\mu},
\end{aligned}$$

where the last inequality follows from Proposition 3.3.1, relations (136), (137), (138), and the assumption that $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$. ▪

3.4.2 "Outer" Iteration Results – Proof of Theorem 3.2.2

In this subsection, we will present the proof of Theorem 3.2.2. Specifically, we will show that the Inexact PDIPF Algorithm obtains an ϵ-approximate solution to (80)–(83) in $\mathcal{O}(n^2 \log(1/\epsilon))$ outer iterations.

Throughout this section, we use the following notation:

$$w(\alpha) := w + \alpha\Delta w, \quad \mu(\alpha) := \mu(w(\alpha)), \quad r(\alpha) := r(w(\alpha)).$$

Lemma 3.4.4 *Assume that Δw satisfies (107)-(110) for some $\sigma \in \mathbb{R}$, $w \in \mathbb{R}^{2n+m+l}$ and $(p,q) \in \mathbb{R}^n \times \mathbb{R}^l$. Then, for every $\alpha \in \mathbb{R}$, we have:*

(a) $X(\alpha)s(\alpha) = (1-\alpha)Xs + \alpha\sigma\mu e - \alpha p + \alpha^2 \Delta X \Delta s$;

(b) $\mu(\alpha) = [1 - \alpha(1-\sigma)]\mu - \alpha p^T e/n + \alpha^2 \Delta x^T \Delta s/n$;

(c) $(r_p(\alpha), r_d(\alpha)) = (1-\alpha)(r_p, r_d)$;

(d) $r_V(\alpha) = (1-\alpha)r_V + \alpha q$.

Proof: Using (109), we obtain

$$\begin{aligned} X(\alpha)s(\alpha) &= (X + \alpha\Delta X)(s + \alpha\Delta s) \\ &= Xs + \alpha(X\Delta s + S\Delta x) + \alpha^2 \Delta X \Delta s \\ &= Xs + \alpha(-Xs + \sigma\mu e - p) + \alpha^2 \Delta X \Delta s \\ &= (1-\alpha)Xs + \alpha\sigma\mu e - \alpha p + \alpha^2 \Delta X \Delta s, \end{aligned}$$

thereby showing that (a) holds. Left multiplying the above equality by e^T and dividing the resulting expression by n, we easily conclude that (b) holds. Statement (c) can be easily verified by means of (107) and (108), while statement (d) follows from (110). ▪

Lemma 3.4.5 *Assume that Δw satisfies (107)-(110) for some $\sigma \in \mathbb{R}$, $w \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ and $(p, q) \in \mathbb{R}^n \times \mathbb{R}^l$ such that $\|p\|_\infty \leq \gamma\sigma\mu/4$. Then, for every scalar α satisfying*

$$0 \leq \alpha \leq \min\left\{1\,,\ \frac{\sigma\mu}{4\,\|\Delta X \Delta s\|_\infty}\right\}, \tag{139}$$

we have

$$\frac{\mu(\alpha)}{\mu} \ \geq\ 1 - \alpha. \tag{140}$$

Proof: Since $\|p\|_\infty \leq \gamma\sigma\mu/4$, we easily see that

$$|p^T e/n| \ \leq\ \|p\|_\infty \ \leq\ \sigma\mu/4. \tag{141}$$

Using this result and Lemma 3.4.4(b), we conclude for every α satisfying (139) that

$$\begin{aligned} \mu(\alpha) &= [1 - \alpha(1-\sigma)]\mu - \alpha p^T e/n + \alpha^2 \Delta x^T \Delta s/n \\ &\geq [1 - \alpha(1-\sigma)]\mu - \frac{1}{4}\,\alpha\sigma\mu + \alpha^2 \Delta x^T \Delta s/n \\ &\geq (1-\alpha)\mu + \frac{1}{4}\,\alpha\sigma\mu - \alpha^2 \|\Delta X \Delta s\|_\infty \\ &\geq (1-\alpha)\mu. \end{aligned}$$

▪

Lemma 3.4.6 *Assume that Δw is a (τ_p, τ_q)-search direction, where τ_p and τ_q are given by (113) and (114), respectively. Assume also that $\sigma > 0$ and that $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ with $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$, $\gamma \in [0,1]$ and $\theta \geq 0$. Then, $w(\alpha) \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ for every scalar α satisfying*

$$0 \leq \alpha \leq \min\left\{1\,,\ \frac{\gamma\sigma\mu}{4\,\|\Delta X \Delta s\|_\infty}\right\}. \tag{142}$$

Proof: Since $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$, there exists $\eta \in [0,1]$ such that $w \in \mathcal{N}_{w^0}(\eta, \gamma, \theta)$. We will show that $w(\alpha) \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta) \subseteq \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ for every α satisfying (142).

First, we note that $(r_p(\alpha), r_d(\alpha)) = (1-\alpha)\eta(r_p^0, r_d^0)$ by Lemma 3.4.4(c) and the definition of $\mathcal{N}_{w^0}(\eta, \gamma, \theta)$. Next, it follows from Lemma 3.4.5 that (140) holds for every α satisfying (139), and hence (142) due to $\gamma \in [0,1]$. Thus, for every α satisfying (142), we have

$$(1-\alpha)\eta \leq \frac{\mu(\alpha)}{\mu}\eta \ \leq\ \frac{\mu(\alpha)}{\mu}\,\frac{\mu}{\mu_0} \ =\ \frac{\mu(\alpha)}{\mu_0}. \tag{143}$$

Now, it is easy to see that for every $u \in \mathbb{R}^n$ and $\tau \in [0,n]$, there holds $\|u - \tau(u^T e/n)e\|_\infty \leq (1+\tau)\|u\|_\infty$. Using this inequality twice, the fact that $w \in \mathcal{N}_{w^0}(\eta, \gamma, \theta)$, relation (113) and statements (a) and (b) of Lemma 3.4.4, we conclude for every α satisfying (142) that

$$\begin{aligned}
&X(\alpha)s(\alpha) - (1-\gamma)\,\mu(\alpha)e \\
&\quad= (1-\alpha)\left[Xs - (1-\gamma)\mu e\right] + \alpha\gamma\sigma\mu e - \alpha\left[p - (1-\gamma)\left(\frac{p^T e}{n}\right)e\right] \\
&\qquad + \alpha^2\left[\Delta X \Delta s - (1-\gamma)\left(\frac{\Delta x^T \Delta s}{n}\right)e\right] \\
&\quad\geq \alpha\left[\gamma\sigma\mu - \left\|p - (1-\gamma)\frac{p^T e}{n}\,e\right\|_\infty - \alpha\left\|\Delta X\Delta s - (1-\gamma)\frac{\Delta x^T \Delta s}{n}\,e\right\|_\infty\right]e \\
&\quad\geq \alpha\left(\gamma\sigma\mu - 2\|p\|_\infty - 2\alpha\|\Delta X \Delta s\|_\infty\right)e \ \geq\ \alpha\left(\gamma\sigma\mu - \frac{1}{2}\gamma\sigma\mu - \frac{1}{2}\gamma\sigma\mu\right)e \ =\ 0.
\end{aligned}$$

Next, by Lemma 3.4.4(d), we have that

$$r_V(\alpha) = (1-\alpha)r_V + \alpha q = \ (1-\alpha)\eta r_V^0 + \hat{a},$$

where $\hat{a} = (1-\alpha)(r_V - \eta r_V^0) + \alpha q$. To complete the proof, it suffices to show that $\|\hat{a}\| \leq \theta\sqrt{\mu(\alpha)}$ for every α satisfying (142). By using equation (114) and Lemma 3.4.4(b) along with the facts that $\|r_V - \eta r_V^0\| \leq \theta\sqrt{\mu}$ and $\alpha \in [0,1]$, we have

$$\begin{aligned}
\|\hat{a}\|^2 - \theta^2\mu(\alpha) &= (1-\alpha)^2\|r_V - \eta r_V^0\|^2 + 2\alpha(1-\alpha)[r_V - \eta r_V^0]^T q + \alpha^2\|q\|^2 - \theta^2\mu(\alpha) \\
&\leq (1-\alpha)^2\theta^2\mu + 2\alpha(1-\alpha)\theta\sqrt{\mu}\|q\| + \alpha^2\|q\|^2 \\
&\quad - \theta^2\left\{[1-\alpha(1-\sigma)]\mu - \alpha\frac{p^Te}{n} + \alpha^2\frac{\Delta x^T\Delta s}{n}\right\} \\
&\leq \alpha^2\|q\|^2 + 2\alpha\theta\sqrt{\mu}\|q\| - \alpha\theta^2\sigma\mu + \alpha\theta^2\frac{p^Te}{n} - \alpha^2\theta^2\frac{\Delta x^T\Delta s}{n} \\
&\leq \alpha\left[\|q\|^2 + 2\theta\sqrt{\mu}\|q\| - \left(1-\frac{\gamma}{4}\right)\theta^2\sigma\mu + \theta^2\alpha\|\Delta X\Delta s\|_\infty\right] \\
&\leq \alpha\left[\|q\|^2 + 2\theta\sqrt{\mu}\|q\| - \left(1-\frac{\gamma}{2}\right)\theta^2\sigma\mu\right] \;\leq\; 0,
\end{aligned}$$

where the last inequality follows from the quadratic formula and the fact that $\|q\| \leq \tau_q$, where τ_q is given by (114). ▪

Next, we derive a lower bound on the stepsize of the Inexact PDIPF Algorithm.

Lemma 3.4.7 *In every iteration of the Inexact PDIPF Algorithm, the step length $\bar{\alpha}$ satisfies*

$$\bar{\alpha} \;\geq\; \min\left\{1, \frac{\min\{\gamma\sigma, 1-\frac{5}{4}\sigma\}\mu}{4\,\|\Delta X\Delta s\|_\infty}\right\} \tag{144}$$

and

$$\mu(\bar{\alpha}) \;\leq\; \left[1 - \left(1-\frac{5}{4}\,\sigma\right)\frac{\bar{\alpha}}{2}\right]\mu. \tag{145}$$

Proof: We know that Δw is a (τ_p, τ_q)-search direction in every iteration of the Inexact PDIPF Algorithm, where τ_p and τ_q are given by (113) and (114). Hence, by Lemma 3.4.6, the quantity $\tilde{\alpha}$ computed in step (g) of the Inexact PDIPF Algorithm satisfies

$$\tilde{\alpha} \;\geq\; \min\left\{1, \frac{\gamma\sigma\mu}{4\,\|\Delta X\Delta s\|_\infty}\right\}. \tag{146}$$

Moreover, by (141), it follows that the coefficient of α in the expression for $\mu(\alpha)$ in Lemma 3.4.4(b) satisfies

$$-(1-\sigma)\mu - \frac{p^Te}{n} \;\leq\; -(1-\sigma)\mu + \|p\|_\infty \;\leq\; -(1-\sigma)\mu + \frac{1}{4}\gamma\sigma\mu \;=\; -\left(1-\frac{5}{4}\,\sigma\right)\mu \;<\; 0, \tag{147}$$

since $\sigma \in (0, 4/5)$. Hence, if $\Delta x^T \Delta s \le 0$, it is easy to see that $\bar{\alpha} = \tilde{\alpha}$, and hence that (144) holds in view of (146). Moreover, by Lemma 3.4.4(b) and (147), we have

$$\mu(\bar{\alpha}) \;\le\; [1-\bar{\alpha}(1-\sigma)]\mu - \bar{\alpha}\frac{p^T e}{n} \;\le\; \left[1-\left(1-\frac{5}{4}\sigma\right)\bar{\alpha}\right]\mu \;\le\; \left[1-\left(1-\frac{5}{4}\sigma\right)\frac{\bar{\alpha}}{2}\right]\mu,$$

showing that (145) also holds. We now consider the case where $\Delta x^T \Delta s > 0$. In this case, we have $\bar{\alpha} = \min\{\alpha_{\min}, \tilde{\alpha}\}$, where $\alpha_{\min}$ is the unconstrained minimum of $\mu(\alpha)$. It is easy to see that

$$\alpha_{\min} \;=\; \frac{n\mu(1-\sigma)+p^T e}{2\Delta x^T \Delta s} \;\ge\; \frac{n[\mu(1-\sigma)-\frac{1}{4}\sigma\mu]}{2\Delta x^T \Delta s} \;\ge\; \frac{\mu(1-\frac{5}{4}\sigma)}{2\,\|\Delta X \Delta s\|_\infty}.$$

The last two observations together with (146) imply that (144) holds in this case too. Moreover, since the function $\mu(\alpha)$ is convex, it must lie below the function $\phi(\alpha)$ over the interval $[0, \alpha_{\min}]$, where $\phi(\alpha)$ is the affine function interpolating $\mu(\alpha)$ at $\alpha = 0$ and $\alpha = \alpha_{\min}$. Hence,

$$\mu(\bar{\alpha}) \;\le\; \phi(\bar{\alpha}) \;=\; [1-(1-\sigma)\frac{\bar{\alpha}}{2}]\mu - \bar{\alpha}\frac{p^T e}{2n} \;\le\; \left[1-\left(1-\frac{5}{4}\sigma\right)\frac{\bar{\alpha}}{2}\right]\mu, \qquad (148)$$

where the second inequality follows from (147). We have thus shown that $\bar{\alpha}$ satisfies (145).

■

Our next task will be to show that the stepsize $\bar{\alpha}$ remains bounded away from zero. In view of (144), it suffices to show that the quantity $\|\Delta X \Delta s\|_\infty$ can be suitably bounded. The next lemma addresses this issue.

Lemma 3.4.8 *Let $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ be such that $(x^0, s^0) \ge (x^*, s^*)$ for some $w^* \in \mathcal{S}$, and let $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ for some $\gamma \ge 0$ and $\theta \ge 0$. Then, the inexact search direction Δw used in the Inexact PDIPF Algorithm satisfies*

$$\max(\|D^{-1}\Delta x\|, \|D\Delta s\|) \;\le\; \left(1-2\sigma+\frac{\sigma^2}{1-\gamma}\right)^{1/2}\sqrt{n\mu} + \frac{1}{\sqrt{1-\gamma}}\left(\frac{\gamma\sigma}{4}\sqrt{n}+6n+\frac{\theta^2}{2}\right)\sqrt{\mu}+\theta\sqrt{\mu}. \qquad (149)$$

Proof: Since $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$, there exists $\eta \in [0,1]$ such that $w \in \mathcal{N}_{w^0}(\eta, \gamma, \theta)$. Let $\widetilde{\Delta w} := \Delta w + \eta(w^0 - w^*)$. Using relations (107), (108), (110), and the fact that $w \in$

$\mathcal{N}_{w^0}(\eta,\gamma,\theta)$, we easily see that

$$A\widetilde{\Delta x} = 0 \tag{150}$$

$$A^T\widetilde{\Delta y} + \widetilde{\Delta s} + V\widetilde{\Delta z} = 0, \tag{151}$$

$$V^T\widetilde{\Delta x} + E^{-2}\widetilde{\Delta z} = E^{-1}(q - r_V + \eta r_V^0). \tag{152}$$

Pre-multiplying (151) by $\widetilde{\Delta x}^T$ and using (150) and (152), we obtain

$$\begin{aligned} \widetilde{\Delta x}^T\widetilde{\Delta s} &= -\widetilde{\Delta x}^T V\widetilde{\Delta z} \;=\; [E^{-2}\widetilde{\Delta z} - E^{-1}(q - r_V + \eta r_V^0)]^T\widetilde{\Delta z} \\ &= \|E^{-1}\widetilde{\Delta z}\|^2 - (q - r_V + \eta r_V^0)^T(E^{-1}\widetilde{\Delta z}) \\ &\geq \|E^{-1}\widetilde{\Delta z}\|^2 - \|q - r_V + \eta r_V^0\|\,\|E^{-1}\widetilde{\Delta z}\| \;\geq\; -\frac{\|q - r_V + \eta r_V^0\|^2}{4}. \end{aligned} \tag{153}$$

Next, we multiply equation (109) by $(XS)^{-1/2}$ to obtain $D^{-1}\Delta x + D\Delta s = H(\sigma) - (XS)^{-1/2}p$, where $H(\sigma) := -(XS)^{1/2}e + \sigma\mu(XS)^{-1/2}e$. Equivalently, we have that

$$D^{-1}\widetilde{\Delta x} + D\widetilde{\Delta s} = H(\sigma) - (XS)^{-1/2}p + \eta\left[D(s^0 - s^*) + D^{-1}(x^0 - x^*)\right] =: g.$$

Taking the squared norm of both sides of the above equation and using (153), we obtain

$$\begin{aligned} \|D^{-1}\widetilde{\Delta x}\|^2 + \|D\widetilde{\Delta s}\|^2 &= \|g\|^2 - 2\widetilde{\Delta x}^T\widetilde{\Delta s} \;\leq\; \|g\|^2 + \frac{\|q - r_V + \eta r_V^0\|^2}{2} \\ &\leq \left(\|g\| + \frac{\|q\| + \|r_V - \eta r_V^0\|}{\sqrt{2}}\right)^2 \;\leq\; \left(\|g\| + \theta\sqrt{\mu}\right)^2, \end{aligned}$$

since $\|q\| + \|r_V - \eta r_V^0\| \leq \left[\sqrt{2} - 1\right]\theta\sqrt{\mu} + \theta\sqrt{\mu} = \sqrt{2}\theta\sqrt{\mu}$ by (111), (114), and the fact that $1 + (1 - \gamma/2)\sigma \leq 2$. Thus, we have

$$\begin{aligned} \max(\|D^{-1}\widetilde{\Delta x}\|\,,\,\|D\widetilde{\Delta s}\|) &\leq \|g\| + \theta\sqrt{\mu} \\ &\leq \|H(\sigma)\| + \|(XS)^{-1/2}\|\,\|p\| + \eta\left[\|D(s^0 - s^*)\| + \|D^{-1}(x^0 - x^*)\|\right] + \theta\sqrt{\mu}. \end{aligned}$$

This, together with the triangle inequality, the definitions of D and $\widetilde{\Delta w}$, and the fact that $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$, imply that

$$\begin{aligned} &\max(\|D^{-1}\Delta x\|, \|D\Delta s\|) \\ &\leq \|H(\sigma)\| + \|(XS)^{-1/2}\|\,\|p\| + 2\eta\left[\|D(s^0 - s^*)\| + \|D^{-1}(x^0 - x^*)\|\right] + \theta\sqrt{\mu} \\ &\leq \|H(\sigma)\| + \|(XS)^{-1/2}\|\,\|p\| + 2\eta\|(XS)^{-1/2}\|\left[\|X(s^0 - s^*)\| + \|S(x^0 - x^*)\|\right] + \theta\sqrt{\mu} \\ &\leq \|H(\sigma)\| + \frac{1}{\sqrt{(1-\gamma)\mu}}\left[\|p\| + 2\eta\left(\|X(s^0 - s^*)\| + \|S(x^0 - x^*)\|\right)\right] + \theta\sqrt{\mu}. \end{aligned} \tag{154}$$

It is well-known (see e.g. [30]) that

$$\|H(\sigma)\| \leq \left(1 - 2\sigma + \tfrac{\sigma^2}{1-\gamma}\right)^{1/2} \sqrt{n\mu}. \tag{155}$$

Moreover, using the fact that $s^* \leq s^0$ and $x^* \leq x^0$ along with Lemma 3.4.2, we obtain

$$\eta \left(\|X(s^0 - s^*)\| + \|S(x^0 - x^*)\|\right) \;\leq\; \eta(s^T x^0 + x^T s^0) \;\leq\; \left(3n + \frac{\theta^2}{4}\right)\mu. \tag{156}$$

The result now follows by noting that $\|p\| \leq \sqrt{n}\|p\|_\infty$, and by incorporating inequalities (155), (156) and (113) into (154). ∎

We are now ready to prove Theorem 3.2.2.

Proof of Theorem 3.2.2: Let Δw^k denote the search direction, and let $r^k = r(w^k)$ and $\mu_k = \mu(w^k)$, at the k-th iteration of the Inexact PDIPF Algorithm. Clearly, $w^k \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$. Hence, using Lemma 3.4.8, assumption (115) and the inequality

$$\|\Delta X^k \Delta s^k\|_\infty \;\leq\; \|\Delta X^k \Delta s^k\| \;\leq\; \|(D^k)^{-1}\Delta x^k\|\,\|D^k \Delta s^k\|,$$

we easily see that $\|\Delta X^k \Delta s^k\|_\infty = \mathcal{O}(n^2)\mu_k$. Using this conclusion together with assumption (115) and Lemma 3.4.7, we see that, for some universal constant $\beta > 0$, we have

$$\mu_{k+1} \;\leq\; \left(1 - \frac{\beta}{n^2}\right)\mu_k, \quad \forall k \geq 0.$$

Using this observation and some standard arguments (see, for example, Theorem 3.2 of [68]), we easily see that the Inexact PDIPF Algorithm generates an iterate $w^k \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ satisfying $\mu_k/\mu_0 \leq \epsilon$ within $\mathcal{O}\left(n^2 \log(1/\epsilon)\right)$ iterations. The theorem now follows from this conclusion and the definition of $\mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$. ∎

3.5 CQP Algorithm using the MWB: Concluding Remarks

We have shown that the long-step PDIPF algorithm for LP based on an iterative linear solver presented in [42] can be extended to the context of CQP. This was not immediately obvious at first since the standard normal equation for CQP does not fit into the mold required for the results of [44] to hold. By considering the ANE, we were able to use the results about the MWB preconditioner developed in [44] in the context of CQP. Another difficulty we encountered was the proper choice of the starting iterate u^0 for the iterative linear solver. By choosing $u^0 = 0$ as in the LP case, we obtain $\|v - Wu^0\| = \|v\|$, which can only be shown to be $\mathcal{O}(\max\{\mu, \sqrt{\mu}\})$. In this case, for every $\mu > 1$, Proposition 3.3.2 would guarantee that the number of inner iterations of the iterative linear solver is

$$\mathcal{O}\left(\psi(\varphi_{\tilde{A}}^2)\max\left\{\log\left(c(\varphi_{\tilde{A}}^2)n\varphi_{\tilde{A}}\right)\ , \log\mu\ \right\}\right),$$

a bound which depends on the logarithm of the current duality gap. On the other hand, Theorem 3.3.5 shows that choosing u^0 as in (128) results in a bound that does not depend on the current duality gap.

We observe that under exact arithmetic, the CG algorithm applied to $Wu = v$ generates an exact solution in at most $m + l$ iterations (since $W \in \mathbb{R}^{(m+l)\times(m+l)}$). It is clear, then, that the bound (130) is generally worse than the well-known finite termination bound for CG. However, our results in Section 3.3 were given for a family of iterative linear solvers, only one member of which is CG. Also, under finite precision arithmetic, the CG algorithm loses its finite termination property, and its convergence rate behavior in this case is still an active topic of research (see e.g. [20]). Certainly, the impact of finite precision arithmetic on our results is an interesting open issue.

Clearly, the MWB preconditioner is not suitable for dense CQP problems since, in this case, the cost to construct the MWB is comparable to the cost to form and factorize $\tilde{A}\tilde{D}^2\tilde{A}^T$, and each inner iteration would require $\Theta((m + l)^2)$ arithmetic operations, the same cost as a forward and back substitution. There are, however, some classes of CQP problems for which the method proposed in this chapter might be useful. One class of problems for which PDIPF methods based on MWB preconditioners might be useful are those for which bases of $\tilde{A}$ are sparse but the ANE coefficient matrices $\tilde{A}\tilde{D}^2\tilde{A}^T$ are dense; this situation generally occurs in sparse CQP problems for which n is much larger than $m + l$. Other classes of problems for which our method might be useful are network flow problems. The paper [53] developed interior-point methods for solving the minimum cost network flow problem based on iterative linear solvers with maximum spanning tree preconditioners. Related to this work, we believe that the following two issues could be investigated: (i) will the incorporation of the correction term p defined in (104) in the algorithm implemented in [53] improve the convergence of the method? (ii) whether our algorithm might be efficient for network flow problems where the costs associated with the arcs are quadratic functions of the arc flows? Identification of other classes of CQP problems which could be efficiently solved by the method proposed in this chapter is another topic for future research.

Regarding the second question above, it is easy to see (after a suitable permutation of the variables) that $V^T = \begin{pmatrix} I & 0 \end{pmatrix}$ and E^2 is a positive diagonal matrix whose diagonal elements are the positive quadratic coefficients. In this case, it can be shown that $\tilde{A}$ is totally unimodular, hence $\varphi_{\tilde{A}}^2 \leq (m+l)(n-m+1)$ by Cramer's Rule (see [44]).

The usual way of defining the dual residual is as the quantity

$$R_d := A^T y + s - VE^2V^T x - c,$$

which, in view of (86) and (87), can be written in terms of the residuals r_d and r_V as

$$R_d = r_d - VEr_V. \tag{157}$$

Note that, along the iterates generated by the Inexact PDIPF Algorithm, we have $r_d = \mathcal{O}(\mu)$ and $r_V = \mathcal{O}(\sqrt{\mu})$, implying that $R_d = \mathcal{O}(\sqrt{\mu})$. Hence, while the usual primal residual converges to 0 according to $\mathcal{O}(\mu)$, the usual dual residual does so according to $\mathcal{O}(\sqrt{\mu})$. This is a unique feature of the convergence analysis of our algorithm in that it contrasts with the analysis of other interior-point PDIPF algorithms, where the primal and dual residuals are required to go to zero at the same rate. The convergence analysis under these circumstances is possible due to the specific form of the $\mathcal{O}(\sqrt{\mu})$-term present in (157), i.e. one that lies in the range space of VE.

CQP problems where V is explicitly available arise frequently in the literature. One important example arises in portfolio optimization (see [12]), where the rank of V is often small. In such problems, l represents the number of observation periods used to estimate the data for the problem. We believe that the Inexact PDIPF Algorithm could be of particular use for this type of application.

3.6 CQP Algorithm for a Class of Preconditioners: Introduction

In this paper we develop an interior-point long-step primal-dual infeasible path-following (PDIPF) algorithm for solving convex quadratic programming (CQP) based on inexact search directions. The CQP problem we consider has the form

$$\min_x \left\{ \frac{1}{2} x^T \mathbf{Q} x + c^T x : \quad Ax = b, \ x \geq 0 \right\}, \tag{158}$$

where the data are $\mathbf{Q} \in \mathbb{R}^{n \times n}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$, and the decision vector is $x \in \mathbb{R}^n$. We assume that $\mathbf{Q}$ is given in the form $\mathbf{Q} = VE^2V^T + Q$, where $V \in \mathbb{R}^{n \times l}$, E is a $l \times l$ positive diagonal matrix, and Q is a $n \times n$ positive semidefinite matrix.

In [33], the authors also developed an inexact PDIPF algorithm for solving (158). This inexact PDIPF algorithm is essentially the infeasible long-step primal-dual path-following algorithm in [28, 72], the only difference being that the search directions are computed by means of an iterative linear solver. We refer to the iterations of the iterative linear solver as the *inner iterations*, and the iterations performed by the actual interior-point method as the *outer iterations*. The main step in the inexact PDIPF algorithm in [33] is the computation of a primal-dual search direction $(\Delta x, \Delta s, \Delta y, \Delta z)$, whose subvector $(\Delta y, \Delta z)$ can be found by solving the so-called *augmented normal equation*, or ANE. This ANE is of the form $\tilde{A}\tilde{D}^2\tilde{A}^T(\Delta y, \Delta z) = g$, where $\tilde{D}$ is a positive diagonal matrix and $\tilde{A}$ is a 2×2 block matrix whose blocks consist of A, V^T, the zero matrix and the identity matrix. In contrast to direct methods, the inexact PDIPF algorithm in [33] assumes that an approximate solution to the ANE is obtained via an iterative linear solver. Since the condition number of the ANE matrix may become excessively large on degenerate QP problems (see e.g. [32]), the maximum weight basis (MWB) preconditioner T introduced in [48, 53, 63] is used to better precondition the matrix. A suitable approximate solution can then be determined within a uniformly bounded number of iterations of an iterative linear solver. Since the ANE is solved only approximately, it cannot yield a search direction which satisfies all equations of the primal-dual Newton system. Thus, we developed a recipe in [33] for determining an inexact search direction, based on an approximate solution to the ANE and the MWB preconditioner, which accomplishes the following two goals: (i) problem (158) can be solved within a polynomial number of iterations, and (ii) the required approximate solution to the ANE can be found within a uniformly bounded number of inner iterations.

This paper extends the authors' previous work [33] in the following two ways. The first extension which we present in this paper is to introduce a new linear system, which we refer to as the *hybrid augmented normal equation* (HANE), as a means to determine the search directions for the PDIPF algorithm studied in this paper. The development of the HANE stems from the desire to take into account the structure of $\mathbf{Q}$, given by $\mathbf{Q} = VE^2V^T + Q$, in the computation of the search direction. To motivate the approach based on the HANE, we will assume in this paragraph that Q is nonnegative diagonal. Consider the two extreme cases where $V = 0$ or $Q = 0$. In the first case, since $\mathbf{Q} = Q$ is diagonal, the search directions can be effectively computed via the standard normal equation, since the latter has a structure similar to that of a linear programming problem. In the second case, the approach based on the ANE developed in [33] provides a viable alternative for computing the search direction. The approach based on the HANE combines the ideas involved in these two extreme cases in order to handle the mixed structure of $\mathbf{Q}$ as stated above. The second extension, which is the major contribution of this paper, is to show that a large class of preconditioners can be used in place of the MWB preconditioner in the recipe for determining inexact search directions proposed in [33]. In this regard, this extension will be done in the more general context of the HANE equation, rather than in the context of the ANE used by [33]. We will also discuss the situation where the preconditioned conjugate gradient method is used in conjunction with the partial update preconditioner proposed by Karmarkar in [24] (see also [19, 29, 40]) and derive the corresponding inner iteration complexity bound.

We observe that the use of iterative linear solvers to compute the primal-dual Newton search directions of interior-point path following algorithms has been extensively studied in [4, 6, 8, 17, 31, 47, 48, 49, 53]. The use of inexact search directions in interior-point methods has been investigated in the context of conic programming problems (see e.g. [4, 5, 17, 31, 39, 47, 73, 61]). For feasibility problems of the form $\{x \in \mathcal{H}_1 : \mathcal{A}x = b, x \in \mathcal{C}\}$, where $\mathcal{H}_1$ and $\mathcal{H}_2$ are Hilbert spaces, $\mathcal{C} \subseteq \mathcal{H}_1$ is a closed convex cone satisfying some mild assumptions, and $\mathcal{A} : \mathcal{H}_1 \to \mathcal{H}_2$ is a continuous linear operator, Renegar [51] has proposed an interior-point method where the Newton system that determines the search directions is approximately solved by performing a uniformly bounded number of iterations of the conjugate gradient (CG) method.

Our paper is divided into five sections. In Subsection 3.6.1, we present some terminology and notation which will be used throughout this paper. In Section 3.7, we present an inexact PDIPF algorithm based on a class of inexact search directions, and we also partially describe a recipe based on the HANE for determining an inexact search direction suitable for this algorithm. In Section 3.8, we introduce the class of preconditioners used in a crucial step of the above recipe for constructing a vector of a required size, thereby providing the final details of the recipe that were left undetermined in Section 3.7. Section 3.9 gives proofs of some of the results presented in Section 3.8. Finally, some concluding remarks are given in Section 3.10.

3.6.1 Terminology and Notation

Throughout this paper, upper-case Roman letters denote matrices, lower-case Roman letters denote vectors, and lower-case Greek letters denote scalars. We let $\mathbb{R}^n$, $\mathbb{R}^n_+$ and $\mathbb{R}^n_{++}$ denote the set of n- vectors having real, nonnegative real, and positive real components, respectively. Also, we let $\mathbb{R}^{m\times n}$ denote the set of $m \times n$ matrices with real entries, and let $\mathcal{S}^n_+$ denote the set of $n \times n$ positive semidefinite real symmetric matrices. For a vector $v \in \mathbb{R}^n$, we let $|v|$ denote the vector whose ith component is $|v_i|$, for every $i = 1, \dots, n$, and we let $\mathrm{Diag}(v)$ denote the diagonal matrix whose ith diagonal element is v_i, for every $i = 1, \dots, n$. In addition, given vectors $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$, we denote by (u, v) the vector $(u^T, v^T)^T \in \mathbb{R}^{m\times n}$.

If a matrix $Z \in \mathbb{R}^{m\times m}$ has all positive eigenvalues, we denote by $\kappa(Z)$ its spectral condition number, i.e. its maximum eigenvalue divided by its minimum eigenvalue. Also, if a matrix $W \in \mathbb{R}^{m\times m}$ is symmetric ($W = W^T$) and positive definite (resp., positive semidefinite), we write $W \succ 0$ (resp., $W \succeq 0$). The range space of W, denoted $\mathcal{R}(W)$, is the set $\{Wv : v \in \mathbb{R}^m\}$. Certain matrices bear special mention, namely the matrices X and S. These matrices are the diagonal matrices corresponding to the vectors x and s, respectively, as described in the previous paragraph. The symbol 0 will be used to denote a scalar, vector, or matrix of all zeroes; its dimensions should be clear from the context. Also, we denote by e the vector of all 1's, and by I the identity matrix; their dimensions should be clear from the context.

We will use several different norms throughout the paper. For a vector $z \in \mathbb{R}^n$, $\|z\| = \sqrt{z^T z}$ is the Euclidean norm and $\|z\|_\infty = \max_{i=1,\dots,n} |z_i|$ is the "infinity norm". Also, given a matrix $C \succ 0$, we define the norm $\|z\|_C = \sqrt{z^T C z}$. Finally, given a matrix $V \in \mathbb{R}^{m\times n}$, $\|V\|$ denotes the operator norm associated with the Euclidean norm: $\|V\| = \max_{z:\|z\|=1} \|Vz\|$.

3.7 CQP Algorithm for a Class of Preconditioners: Outer Iteration Framework

In this section, we introduce an inexact PDIPF algorithm based on a class of inexact search directions and discuss its iteration complexity. The algorithm is essentially equivalent to the one presented in [33]. This section is divided into subsections. In Subsection 3.7.1, we introduce the class of inexact search directions, state the inexact PDIPF algorithm based on it, and give its iteration complexity result. In Subsection 3.7.2, we will discuss how the HANE naturally appears as a way of computing the exact search direction. We will also describe how an approximate solution to the HANE can be used to compute an inexact search direction which is suitable for the inexact PDIPF algorithm.

3.7.1 An Inexact PDIPF Algorithm for CQP

Consider the following primal-dual pair of QP problems:

$$\min_x \quad \left\{ \frac{1}{2} x^T \mathbf{Q} x + c^T x : \quad Ax = b,\ x \geq 0 \right\}, \tag{159}$$

$$\max_{(\hat{x},s,y)} \quad \left\{ -\frac{1}{2} \hat{x}^T \mathbf{Q} \hat{x} + b^T y : \quad A^T y + s - \mathbf{Q}\hat{x} = c,\ s \geq 0 \right\}, \tag{160}$$

where the data are $\mathbf{Q} \in \mathcal{S}^n_+$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$, and the decision variables are $x \in \mathbb{R}^n$ and $(\hat{x}, s, y) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R}^m$. We will assume that $\mathbf{Q}$ is given in the form

$\mathbf{Q} = VE^2V^T + Q$ for some $V \in \mathbb{R}^{n\times l}$, $E \in \text{Diag}(\mathbb{R}^l_{++})$ and $Q \in \mathcal{S}^n_+$. In addition, we will make the following two assumptions throughout the paper:

Assumption 5 *rank*$(A) = m < n$.

Assumption 6 *The set of optimal solutions of (159) and (160) is nonempty.*

It is well-known that if x^* is an optimal solution for (159) and $(\hat{x}^*, s^*, y^*)$ is an optimal solution for (160), then (x^*, s^*, y^*) is also an optimal solution for (160). Now, let $\mathcal{S}$ denote the set of all vectors $w := (x, s, y, z) \in \mathbb{R}^{2n+m+l}$ satisfying

$$Ax = b, \quad x \geq 0, \tag{161}$$

$$A^Ty + s + Vz - Qx = c, \quad s \geq 0, \tag{162}$$

$$Xs = 0, \tag{163}$$

$$EV^Tx + E^{-1}z = 0. \tag{164}$$

It is clear that $w \in \mathcal{S}$ if and only if x is optimal for (159), (x, s, y) is optimal for (160), and $z = -E^2V^Tx$. (Throughout this paper, the symbol w will always denote the quadruple (x, s, y, z), where the vectors lie in the appropriate dimensions; similarly, $\Delta w = (\Delta x, \Delta s, \Delta y, \Delta z)$, $w^k = (x^k, s^k, y^k, z^k)$, etc.)

For a point $w \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$, let us define

$$\mu := \mu(w) = x^Ts/n, \tag{165}$$

$$r_p := r_p(w) = Ax - b, \tag{166}$$

$$r_d := r_d(w) = A^Ty + s + Vz - Qx - c, \tag{167}$$

$$r_V := r_V(w) = EV^Tx + E^{-1}z, \tag{168}$$

$$r := r(w) = (r_p(w), r_d(w), r_V(w)). \tag{169}$$

Given a point $u \in \mathcal{R}(Q)$, it is easy to show that the function t^TQt is constant over the manifold $\{t \in \mathbb{R}^n : Qt = u\}$. Hence, the function $|||\cdot|||_Q : \mathcal{R}(Q) \mapsto \mathbb{R}_+$ given by

$$|||u|||_Q = \sqrt{t^TQt} \text{ for any } t \in \mathbb{R}^n \text{ such that } Qt = u \tag{170}$$

is well-defined. The following proposition shows that this function is a norm on $\mathcal{R}(Q)$.

Proposition 3.7.1 *Let $|||\cdot|||_Q$ be as defined in (170), and let $u \in \mathcal{R}(Q)$. Then, the following statements hold:*

1. *Given a factorization $Q = \widetilde{V}\widetilde{V}^T$, where $\widetilde{V}$ has full column rank, we have that $|||u|||_Q = \|\mathbf{v}\|$, where $\mathbf{v}$ is the unique vector satisfying $\widetilde{V}\mathbf{v} = u$;*
2. *$|||\cdot|||_Q$ defines a norm on $\mathcal{R}(Q)$; and*
3. *$\|u\| \le \|Q\|^{1/2}\,|||u|||_Q$.*

Proof: Let $u \in \mathcal{R}(Q)$ be given, and let t be a vector such that $Qt = u$. If we define $\mathbf{v} := \widetilde{V}^T t$, then the assumption that $Q = \widetilde{V}\widetilde{V}^T$ along with (170) implies that

$$|||u|||_Q \;=\; \sqrt{t^T Q t} \;=\; \|\widetilde{V}^T t\| = \|\mathbf{v}\|. \tag{171}$$

Since $u = Qt = \widetilde{V}\widetilde{V}^T t = \widetilde{V}\mathbf{v}$ and $\widetilde{V}$ has full column rank, it is clear that $\mathbf{v}$ is uniquely determined by the formula $\mathbf{v} = [\widetilde{V}^T\widetilde{V}]^{-1}\widetilde{V}^T u$, and statement 1 is proven.

Using the above formula for $\mathbf{v}$ and statement 1, it is easy to see that $|||\cdot|||_Q$ is a seminorm on $\mathcal{R}(Q)$. It is indeed a norm, since, in view of (171), $|||u|||_Q = 0$ implies that $\mathbf{v} = 0$, and hence that $u = \widetilde{V}\mathbf{v} = 0$.

To prove the third statement, let t be a vector such that $Qt = u$. Then (170) implies that

$$\|u\| \;=\; \|Qt\| \;\le\; \|Q^{1/2}\|\,\|Q^{1/2}t\| \;=\; \|Q\|^{1/2}\sqrt{t^T Q t} \;=\; \|Q\|^{1/2}\,|||u|||_Q.$$

■

Next, given a point $w \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ and scalars $\sigma \in [0,1]$, $\tau_p > 0$, and $\tau_q > 0$, we will say that a search direction Δw is a (τ_p, τ_q)-*search direction* at w (with centrality parameter σ) if Δw satisfies

$$A\Delta x \;=\; -r_p, \tag{172}$$

$$A^T\Delta y + \Delta s + V\Delta z - Q\Delta x \;=\; -r_d - g, \tag{173}$$

$$X\Delta s + S\Delta x \;=\; -Xs + \sigma\mu e - p, \tag{174}$$

$$EV^T\Delta x + E^{-1}\Delta z \;=\; -r_V + q \tag{175}$$

for some $(g, p, q) \in \mathcal{R}(Q) \times \mathbb{R}^n \times \mathbb{R}^l$ such that

$$\|p\|_\infty \;\le\; \tau_p \mu, \qquad |||g|||_Q^2 + \|q\|^2 \;\le\; \tau_q^2 \mu, \tag{176}$$

where μ is given by (165). Note that while p and q can vary over the whole Euclidean spaces $\mathbb{R}^n$ and $\mathbb{R}^l$, respectively, the error g is required to be in $\mathcal{R}(Q)$.

We will now point out the relationship between the definition above and the definition of a (τ_p, τ_q)-search direction given in paper [33]. It is clear that system (32)-(35) in [33] for determining an inexact search direction can be viewed as a special case of system (172)-(175) by setting $Q = 0$, which also implies that $g = 0$ due to the fact that $g \in \mathcal{R}(Q)$. However, it is also possible to transform system (172)-(175) into a system of the form specified by equations (32)-(35) of [33] (see the proof of Theorem 3.7.2 in Subsection 3.9.1). Hence, these two systems for defining inexact search directions are essentially equivalent. We consider system (172)-(175) in this paper because it naturally lends itself to the development of the HANE as a means to determine the search direction Δw (see Subsection 3.7.2).

Next, given $\eta \in [0, 1]$, $\gamma \in (0, 1)$, $\theta > 0$, and an initial point $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$, we define the following set:

$$\mathcal{N}_{w^0}(\eta, \gamma, \theta) := \left\{ w \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l} : \begin{array}{c} Xs \ge (1-\gamma)\mu e,\ r_p = \eta r_p^0,\ \eta \le \mu/\mu_0, \\ r_d - \eta r_d^0 \in \mathcal{R}(Q), \\ |||r_d - \eta r_d^0|||_Q^2 + \|r_V - \eta r_V^0\|^2 \le \theta^2 \mu. \end{array} \right\}, \tag{177}$$

where $\mu = \mu(w)$, $\mu_0 = \mu(w^0)$, $r = r(w)$ and $r^0 = r(w^0)$. The central path neighborhood used by the inexact PDIPF algorithm described below is given by

$$\mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta) \;=\; \bigcup_{\eta \in [0,1]} \mathcal{N}_{w^0}(\eta, \gamma, \theta). \tag{178}$$

We are now ready to state the inexact PDIPF algorithm.

Inexact PDIPF Algorithm:

1. **Start:** Let $\epsilon > 0$ and $0 < \underline{\sigma} \le \overline{\sigma} < 4/5$ be given. Choose $\gamma \in (0, 1)$, $\theta > 0$ and $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ such that $w^0 \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$. Set $k = 0$.

```
2. **While** $\mu_k := \mu(w^k) > \epsilon$ **do**
    (a) Let $w := w^k$ and $\mu := \mu_k$; choose $\sigma := \sigma_k \in [\underline{\sigma}, \overline{\sigma}]$.
    (b) Set
            $\tau_p = \gamma\sigma/4$ and                                   (179)
            $\tau_q = \left[\sqrt{1+(1-0.5\gamma)\,\sigma} - 1\right]\theta$.       (180)
    (c) Compute a $(\tau_p, \tau_q)$-search direction $\Delta w := \Delta w^k$.
    (d) Compute $\tilde{\alpha} := \mathrm{argmax}\{\alpha \in [0,1] : w + \alpha'\Delta w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta),\ \forall \alpha' \in [0,\alpha]\}$.
    (e) Compute $\bar{\alpha} := \mathrm{argmin}\{(x+\alpha\Delta x)^T(s+\alpha\Delta s) : \alpha \in [0,\tilde{\alpha}]\}$.
    (f) Let $w^{k+1} = w + \bar{\alpha}\Delta w$, and set $k \leftarrow k+1$.
   **End** (while)
```

The following result gives a bound on the number of iterations needed by the inexact PDIPF algorithm to obtain an ϵ-solution to the KKT conditions (161)–(164). Its proof will be given in Subsection 3.9.1.

Theorem 3.7.2 *Assume that the constants γ, $\underline{\sigma}$, $\overline{\sigma}$ and θ are such that*

$$\max\left\{\gamma^{-1}\,,\ (1-\gamma)^{-1}\,,\ \underline{\sigma}^{-1}\,,\ \left(1-\frac{5}{4}\overline{\sigma}\right)^{-1}\right\} = \mathcal{O}(1),\ \ \theta = \mathcal{O}(\sqrt{n}), \tag{181}$$

and that the initial point $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfies $(x^0, s^0) \geq (x^*, s^*)$ for some $w^* \in \mathcal{S}$. Then, the inexact PDIPF algorithm generates an iterate $w^k \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ satisfying $\mu_k \leq \epsilon\mu_0$, $\|r_p^k\| \leq \epsilon\|r_p^0\|$, $\|r_d^k\| \leq \epsilon\|r_d^0\| + \epsilon^{1/2}\theta\|Q\|^{1/2}\mu_0^{1/2}$ and $\|r_V^k\| \leq \epsilon\|r_V^0\| + \epsilon^{1/2}\theta\mu_0^{1/2}$ within $\mathcal{O}\left(n^2 \log \epsilon^{-1}\right)$ iterations.

It is possible to show that if w^0 is a strictly feasible point, i.e. $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ and $r^0 = 0$, then the iteration complexity of the above algorithm is bounded by $\mathcal{O}(n \log \epsilon^{-1})$ iterations. It is also possible to develop a primal-dual short-step path-following algorithm based on the inexact search directions introduced above, which would have iteration complexity bounds $\mathcal{O}(n \log \epsilon^{-1})$ and $\mathcal{O}(\sqrt{n} \log \epsilon^{-1})$ for infeasible and feasible starting points, respectively. One interesting characteristic of the feasible algorithms discussed in this paragraph is that, although the algorithms start with a primal- and dual-feasible point w^0, the algorithms only maintain primal feasibility throughout, while the dual residuals satisfy $\|r_d\| = \mathcal{O}(\sqrt{\mu})$. For the sake of brevity, we will only deal with the long-step PDIPF algorithm stated above.

3.7.2 Framework for Computing an Inexact Search Direction

In this subsection we will provide a framework for computing inexact search directions and give sufficient conditions for them to be (τ_p, τ_q)-search directions.

We begin by defining the following matrices:

$$\begin{aligned} D &:= (Q + X^{-1}S)^{-1/2}, && (182)\\ \widehat{D} &:= \begin{pmatrix} D & 0 \\ 0 & E^{-1} \end{pmatrix} \in \mathbb{R}^{(n+l)\times(n+l)}, && (183)\\ \widehat{A} &:= \begin{pmatrix} A & 0 \\ V^T & I \end{pmatrix} \in \mathbb{R}^{(m+l)\times(n+l)}, && (184)\\ H &:= \widehat{A}\widehat{D}^2\widehat{A}^T, && (185) \end{aligned}$$

and the vector

$$h := \widehat{A} \begin{pmatrix} D^2(s - \sigma\mu X^{-1}e - r_d) \\ 0 \end{pmatrix} - \begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix}. \qquad (186)$$

One approach to compute an exact search direction, i.e. a direction Δw satisfying (172)–(175) with $(g, p, q) = 0$, is as follows. First, we solve the following system of equations for $(\Delta y, \Delta z)$:

$$H \begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} = h.$$

This system is what we refer to as the HANE. (We observe that if $V = 0$, i.e. $\mathbf{Q} = Q$, then this system reduces to the standard normal equation for QP, while if $Q = 0$, i.e. $\mathbf{Q} = VE^2V^T$, it reduces to the ANE in [33].) Once $(\Delta y, \Delta z)$ is determined, we obtain Δx and Δs according to formulas (188) and (189) below with $g = p = 0$.

Suppose now that the HANE is solved only inexactly, i.e. that the vector $(\Delta y, \Delta z)$ satisfies

$$H \begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} = h + f \tag{187}$$

for some error vector f. We then compute Δx and Δs according to the following formulas:

$$\begin{aligned} \Delta x &= D^2 \left(r_d + A^T \Delta y + V \Delta z - s + \sigma\mu X^{-1} e + g - X^{-1} p \right), &(188)\\ \Delta s &= -r_d - A^T \Delta y + Q \Delta x - V \Delta z - g, &(189) \end{aligned}$$

where the pair of correction vectors $(g, p) \in \mathcal{R}(Q) \times \mathbb{R}^n$ will be required to satisfy some conditions which we describe below. Clearly, the search direction $\Delta w = (\Delta x, \Delta s, \Delta y, \Delta z)$ computed as above satisfies (173) in view of (189). Moreover, (174) is satisfied, since equations (182), (188), and (189) imply that

$$\begin{aligned} X\Delta s + S\Delta x &= -Xr_d - XA^T \Delta y - XV\Delta z - Xg + (XQ + S)\Delta x \\ &= -Xr_d - XA^T \Delta y - XV\Delta z - Xg + XD^{-2}\Delta x \\ &= -Xs + \sigma\mu e - p. \end{aligned}$$

To motivate the conditions we will impose on the pair $(g, p) \in \mathcal{R}(Q) \times \mathbb{R}^n$, we note that equations (183)–(189) imply that

$$\begin{aligned} &\widehat{A} \begin{pmatrix} \Delta x \\ E^{-2}\Delta z \end{pmatrix} + \begin{pmatrix} r_p \\ E^{-1} r_V \end{pmatrix} \\ &= \widehat{A} \begin{pmatrix} D^2 \left((A^T \Delta y + V\Delta z) + (-s + \sigma\mu X^{-1} e + r_d) + (g - X^{-1} p) \right) \\ E^{-2}\Delta z \end{pmatrix} + \begin{pmatrix} r_p \\ E^{-1} r_V \end{pmatrix} \\ &= \widehat{A}\widehat{D}^2 \begin{pmatrix} A^T \Delta y + V\Delta z \\ \Delta z \end{pmatrix} - h - \widehat{A} \begin{pmatrix} D^2 (X^{-1} p - g) \\ 0 \end{pmatrix} \\ &= H \begin{pmatrix} \Delta y \\ \Delta z \end{pmatrix} - h - \widehat{A} \begin{pmatrix} D^2 (X^{-1} p - g) \\ 0 \end{pmatrix} = f - \widehat{A} \begin{pmatrix} D^2 (X^{-1} p - g) \\ 0 \end{pmatrix}. \end{aligned} \tag{190}$$

Our strategy will be to choose the pair $(g, p) \in \mathcal{R}(Q) \times \mathbb{R}^n$ so that the first component of (190) is zero, and hence that (172) is satisfied. Specifically, let us partition $f = (f_1, f_2) \in$

$\mathbb{R}^m \times \mathbb{R}^l$. We will choose $(g,p) \in \mathcal{R}(Q) \times \mathbb{R}^n$ such that

$$AD^2(X^{-1}p - g) \;=\; f_1. \tag{191}$$

Observe that g and p are not uniquely defined. Letting

$$q \;=\; E\left(f_2 - V^T D^2(X^{-1}p - g)\right)$$

and using (184), we easily see that (191) is equivalent to

$$f \;=\; \widehat{A}\begin{pmatrix} D^2(X^{-1}p - g) \\ E^{-1}q \end{pmatrix}. \tag{192}$$

Then, using (184), (190), and (192), we conclude that

$$\begin{aligned} \widehat{A}\begin{pmatrix} \Delta x \\ E^{-2}\Delta z \end{pmatrix} + \begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} \;&=\; f - \widehat{A}\begin{pmatrix} D^2(X^{-1}p - g) \\ E^{-1}q \end{pmatrix} + \widehat{A}\begin{pmatrix} 0 \\ E^{-1}q \end{pmatrix} \\ &=\; \widehat{A}\begin{pmatrix} 0 \\ E^{-1}q \end{pmatrix} \;=\; \begin{pmatrix} 0 \\ E^{-1}q \end{pmatrix}, \end{aligned} \tag{193}$$

from which we see that the first component of (190) is set to 0 and the second component is exactly $E^{-1}q$. We have thus shown that the above construction yields a search direction Δw satisfying equations (172)–(175).

Before ending this subsection, we provide a framework for computing a triple $(g,p,q) \in \mathcal{R}(Q) \times \mathbb{R}^n \times \mathbb{R}^l$ satisfying (192). First, choose a vector $v := (v_1, v_2) \in \mathbb{R}^n \times \mathbb{R}^l$ satisfying

$$\widehat{A}v = f. \tag{194}$$

Next, we choose the triple $(g,p,q) \in \mathcal{R}(Q) \times \mathbb{R}^n \times \mathbb{R}^l$ according to

$$g := -Qv_1, \quad p := Sv_1, \quad q := Ev_2. \tag{195}$$

Then (182), (194), and (195) imply that

$$\widehat{A}\begin{pmatrix} D^2(X^{-1}p - g) \\ E^{-1}q \end{pmatrix} \;=\; \widehat{A}\begin{pmatrix} D^2(X^{-1}S + Q)v_1 \\ v_2 \end{pmatrix} \;=\; \widehat{A}v \;=\; f,$$

i.e. $(g,p,q) \in \mathcal{R}(Q) \times \mathbb{R}^n \times \mathbb{R}^l$ satisfies (192). Note that in view of Assumption 5 and (184), system (194) has multiple solutions. Strategies for choosing a specific vector v satisfying (194) will be discussed in Subsection 3.8.1.

The following result relates the size of $\widehat{D}^{-1}v$ with the magnitude of the triple $(g,p,q) \in \mathcal{R}(Q) \times \mathbb{R}^n \times \mathbb{R}^l$, and gives a sufficient condition for the search direction described above to be a (τ_p, τ_q)-search direction.

Proposition 3.7.3 *Let $w \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ be given, and consider the vector $v \in \mathbb{R}^{n+l}$ and the triple $(g,p,q) \in \mathcal{R}(Q) \times \mathbb{R}^n \times \mathbb{R}^l$ as defined in (194) and (195). Then, we have*

$$\|p\| \ \leq \ \sqrt{n\mu}\, \|\widehat{D}^{-1}v\|, \qquad |||g|||_Q^2 + \|q\|^2 \ \leq \ \|\widehat{D}^{-1}v\|^2. \tag{196}$$

As a consequence, if $\|\widehat{D}^{-1}v\| \leq \xi\sqrt{\mu}$, where ξ is defined as

$$\xi \ := \ \min\{n^{-1/2}\tau_p, \tau_q\}, \tag{197}$$

then the corresponding inexact search direction Δw as described above is a (τ_p, τ_q)-search direction.

Proof: Using (182) and the fact that $(x,s) > 0$, we conclude that $Q \preceq Q + X^{-1}S = D^{-2}$. Next, the second identity in (195) along with (170) implies that $|||g|||_Q^2 = v_1^T Q v_1$. Using these facts along with (183) and (195), we obtain

$$|||g|||_Q^2 + \|q\|^2 \ = \ v_1^T Q v_1 + \|Ev_2\|^2 \ \leq \ v_1^T D^{-2} v_1 + \|Ev_2\|^2 \ = \ \|D^{-1}v_1\|^2 + \|Ev_2\|^2 \ = \ \|\widehat{D}^{-1}v\|^2.$$

Similarly, we have $X^{-1}S \preceq D^{-2}$, which clearly implies that $D^2 \preceq XS^{-1}$. This result, along with the fact that $x_i s_i \leq n\mu$ for all i, implies that $SD^2S \preceq XS \preceq n\mu I$, and hence that $\|SD\| = \|SD^2S\|^{1/2} \leq \sqrt{n\mu}$. We use this result along with (183) and the first relation in (195) to obtain

$$\|p\| \ = \ \|Sv_1\| \ \leq \ \|SD\|\,\|D^{-1}v_1\| \ \leq \ \sqrt{n\mu}\,\|D^{-1}v_1\| \ \leq \ \sqrt{n\mu}\,\|\widehat{D}^{-1}v\|.$$

Thus (196) is proven. The second part of the proposition follows from the fact that (196), (197), and the assumption that $\|\widehat{D}^{-1}v\| \leq \xi\sqrt{\mu}$ imply that (176) holds. ∎

3.8 CQP Algorithm for a Class of Preconditioners: Inner Iteration Complexity

In this section, we complete the description of the recipe given in Subsection 3.7.2 to determine a (τ_p, τ_q)-search direction Δw. The section is divided into two subsections. In Subsection 3.8.1, we derive a uniform upper bound on the number of iterations a generic iterative linear solver requires to obtain a sufficiently accurate solution $(\Delta y, \Delta z)$ to the HANE, which will then yield a (τ_p, τ_q)-search direction Δw, as required in step 2(d) of the inexact PDIPF algorithm. One of the key ideas in this paper, which is described in Subsection 3.7.1, is the use of a suitable approximation F of $\widehat{D}^2$ to define the vector v as a linear function of u. In Subsection 3.7.2, we present two examples of matrices F which are suitable approximations of $\widehat{D}^2$. We also obtain specific expressions for the iteration complexity developed in Subsection 3.7.1 when the iterative solver used to obtain an approximate solution to the HANE is the preconditioned conjugate gradient (PCG) method with preconditioner given by $\widehat{A}F\widehat{A}^T$.

3.8.1 Inner Iteration Complexity Analysis

In this subsection, we will complete the description of the recipe given in Subsection 3.7.2 to determine a (τ_p, τ_q)-search direction Δw. For simplicity of notation, in this section we will denote the variable of unknowns in the HANE by u, so that the HANE takes the form $Hu = h$, where H and h are given by (185) and (186), respectively. Recall that the only thing that was unspecified in the recipe of Subsection 3.7.2 was the choice of a vector v satisfying (194). Recall also from Lemma 3.7.3 that by choosing v such that $\|\widehat{D}^{-1}v\| \le \xi\sqrt{\mu}$, where ξ is given by (197), the corresponding inexact search direction Δw is guaranteed to be a (τ_p, τ_q)-search direction, simply by choosing (g, p, q) according to (195). One of the key ideas in this paper, which is described in this subsection, is the use of a generic preconditioner for H to define the vector v as a linear function of u. This subsection also discusses the iteration complexity of a generic iterative solver to obtain an iterate u so that the corresponding $v = v(u)$ satisfies the condition $\|\widehat{D}^{-1}v\| \le \xi\sqrt{\mu}$. We also discuss an appropriate choice of the starting point u^0 and conditions on the generic preconditioner for H which guarantee that the inner iteration complexity bound is uniformly bounded throughout the iterations of the inexact PDIPF algorithm.

We will first discuss the criterion we use to measure the complexity of an iterative solver to obtain an approximate solution to a system of the form $Hu = h$. A common way of measuring the closeness of u to $u^* := H^{-1}h$ is by the distance $\|u - u^*\|_H = \|f(u)\|_{H^{-1}}$, where $f(u) := Hu - h$. Many algorithms for solving the system $Hu = h$ produce a sequence of iterates which decrease this distance at every step (see [20, 25, 35]). Other equivalent distances could be used in our discussion below, but we will only consider the one above without any loss of generality. We will say that the complexity of an iterative solver (with respect to the above distance) is bounded above by a nondecreasing function $\Upsilon : [1, \infty) \mapsto \mathbb{Z}_+$ if, for any $\delta \geq 1$, $\Upsilon(\delta)$ denotes an upper bound on the number of iterations required by the iterative solver, started at any u^0, to obtain an iterate u such that $\|f(u)\|_{H^{-1}} \leq \delta^{-1}\|f(u^0)\|_{H^{-1}}$.

Next, we will discuss a way of choosing a vector v satisfying (194) and the condition

$$\|\widehat{D}^{-1}v\| \;\leq\; K\|f(u)\|_{H^{-1}} \tag{198}$$

for some suitable constant $K \geq 1$. For fixed $f(u)$, consider the ideal case for which we set $v = v_{LS}$, where $v_{LS} = \operatorname{argmin}\{\|\widehat{D}^{-1}v\| : \widehat{A}v = f(u)\}$. It is straightforward to show that

$$v_{LS} \;=\; \widehat{D}^2\widehat{A}^T H^{-1} f(u) \;=\; \widehat{D}^2\widehat{A}^T(\widehat{A}\widehat{D}^2\widehat{A}^T)^{-1} f(u), \tag{199}$$

where H is given by (185). Thus we have that

$$\|\widehat{D}^{-1}v_{LS}\| \;=\; \sqrt{f(u)^T(\widehat{A}\widehat{D}^2\widehat{A}^T)^{-1}f(u)} \;=\; \|f(u)\|_{H^{-1}}, \tag{200}$$

and hence that (198) is satisfied with $K = 1$. Unfortunately, the computation of v_{LS} requires the computation of $H^{-1}f(u)$, or equivalently the solution of a system of linear equations with the same coefficient matrix as the HANE we are trying to solve. To remedy this problem, we will approximate $\widehat{D}^2$ by a matrix $F \succeq 0$ such that $G := \widehat{A}F\widehat{A}^T \succ 0$ and $G^{-1}f(u)$ is much cheaper to compute than $H^{-1}f(u)$. We then replace $\widehat{D}^2$ in (199) by F to obtain a vector v according to

$$v \;:=\; v(F, u) \;=\; F\widehat{A}^T G^{-1} f(u) \;=\; F\widehat{A}^T(\widehat{A}F\widehat{A}^T)^{-1} f(u). \tag{201}$$

It is clear that v defined in this manner satisfies (194). By imposing some conditions on the approximation F according to the definition below, v will also satisfy (198) for some constant $K \geq 1$. We will require that F approximate $\widehat{D}^2$ in the following sense.

Definition 2 *Let constants $0 < \lambda_L \leq \lambda_U$ be given. We will say that a matrix F is a (λ_L, λ_U)-approximation of $\widehat{D}^2$ if $0 \preceq F \preceq \lambda_U \widehat{D}^2$ and $\widehat{A}F\widehat{A}^T \succeq \lambda_L \widehat{A}\widehat{D}^2\widehat{A}^T$.*

Using the above definition, we can now state the following result.

Lemma 3.8.1 *Suppose that a matrix F is a (λ_L, λ_U)-approximation of $\widehat{D}^2$. Then the vector v given by (201) satisfies (198) with $K = \sqrt{\lambda_U/\lambda_L}$.*

Proof: Let $G = \widehat{A}F\widehat{A}^T$, and recall the definition of H in (185). Using the assumption that F is a (λ_L, λ_U)-approximation of $\widehat{D}^2$ and Definition 2, we have that $G^{-1} \preceq \lambda_L^{-1} H^{-1}$ and $\widehat{D}^{-1}F\widehat{D}^{-1} \preceq \lambda_U I$. Using these inequalities along with (201), we conclude that

$$\begin{aligned}
\|\widehat{D}^{-1}v\| &\leq \|\widehat{D}^{-1}F^{1/2}\| \, \|F^{1/2}\widehat{A}^T G^{-1} f(u)\| \;=\; \|\widehat{D}^{-1}F^{1/2}\| \sqrt{f(u)^T G^{-1}(\widehat{A}F\widehat{A}^T)G^{-1}f(u)} \\
&= \|\widehat{D}^{-1}F\widehat{D}^{-1}\|^{1/2}\sqrt{f(u)^T G^{-1} f(u)} \;\leq\; \sqrt{\lambda_U/\lambda_L}\sqrt{f(u)^T H^{-1} f(u)} \\
&= \sqrt{\lambda_U/\lambda_L}\|f(u)\|_{H^{-1}}.
\end{aligned}$$

■

Note that if u is a point such that $\|f(u)\|_{H^{-1}} \leq \delta^{-1}\|f(u^0)\|_{H^{-1}}$, and if v is formed according to (201), where F is a (λ_L, λ_U)-approximation of $\widehat{D}^2$, we have

$$\frac{\|\widehat{D}^{-1}v\|}{\|f(u^0)\|_{H^{-1}}} \;\leq\; \sqrt{\lambda_U/\lambda_L}\frac{\|f(u)\|_{H^{-1}}}{\|f(u^0)\|_{H^{-1}}} \;\leq\; \delta^{-1}\sqrt{\lambda_U/\lambda_L} \tag{202}$$

in view of Lemma 3.8.1. The issues to be considered now are (i) the choice of the starting point u^0 and (ii) the choice of δ. Regarding (i), we will show that a starting point u^0 can always be chosen so that

$$\|f(u^0)\|_{H^{-1}} \;\leq\; \Psi\sqrt{\mu} \tag{203}$$

for some universal constant Ψ. Assuming this fact, the constant δ in issue (ii) can be chosen as

$$\delta \;=\; (\Psi/\xi)\sqrt{\lambda_U/\lambda_L}, \tag{204}$$

where ξ is given by (197). Indeed, by (202)–(204), it follows that the resulting vector v satisfies $\|\widehat{D}^{-1}v\| \le \xi\sqrt{\mu}$, as desired.

We will now concentrate our attention on the construction of a starting point u^0 satisfying (203). First, compute a point $w' = (x', y', s', z')$ satisfying the following system of linear equations:

$$\widehat{A}\begin{pmatrix} x' \\ E^{-2}z' \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}, \qquad A^T y' + s' + Vz' - Q_2 x' = c. \tag{205}$$

We then define

$$u^0 \;=\; -\eta \begin{pmatrix} y^0 - y' \\ z^0 - z' \end{pmatrix}, \tag{206}$$

where $\eta = \|r_p\|/\|r_p^0\|$. Notice that all of the starting points generated by the above formula are multiples of the same vector, which can be computed once at the beginning of the inexact PDIPF algorithm. Moreover, if the starting point w^0 of the algorithm is feasible to (159) and (160), then we may choose $w' = w^0$, and hence $u^0 = 0$. The following lemma gives a bound on $\|f(u^0)\|_{H^{-1}}$.

Lemma 3.8.2 *Assume that w^0 and w' are such that $(x^0, s^0) \ge |(x', s')|$ and $(x^0, s^0) \ge (x^*, s^*)$ for some $w^* \in \mathcal{S}$. Further, assume that $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ for some $\gamma \in [0,1]$ and $\theta > 0$, and that H, h and u^0 are given by (185), (187) and (206), respectively. Then, $f(u^0)$ satisfies (203), where μ is given by (165) and Ψ is defined as*

$$\Psi \;:=\; \frac{6}{\sqrt{1-\gamma}}\, n + \left(1 - 2\sigma + \frac{\sigma^2}{1-\gamma}\right)^{1/2} \sqrt{n} + \frac{\theta^2}{2\sqrt{1-\gamma}} + \theta. \tag{207}$$

The proof of this lemma will be given in Subsection 3.9.2. Our next lemma provides insight into the size of the ratio Ψ/ξ in (204).

Lemma 3.8.3 *Suppose that* $\max\{\sigma, \sigma^{-1}, \gamma^{-1}, (1-\gamma)^{-1}, \theta^{-1}\} = \mathcal{O}(1)$ *and $\theta = \mathcal{O}(\sqrt{n})$ in the inexact PDIPF algorithm, and that τ_p, τ_q, ξ and Ψ are as defined in (179), (180), (197) and (207), respectively. Then, we have that $\Psi/\xi = \mathcal{O}(n^{3/2})$.*

Proof: Under the assumptions above, it is easy to see that $\Psi = \mathcal{O}(n)$ and $\xi^{-1} = \mathcal{O}(\sqrt{n})$, and the result follows immediately. ▪

We summarize the results of this subsection in the following theorem.

Theorem 3.8.4 *Suppose that the conditions of Lemmas 3.8.1–3.8.3 are met. Then, an iterative solver with complexity bounded by $\Upsilon(\cdot)$ generates an iterate u such that $v = v(F,u)$ satisfies $\|\widehat{D}^{-1}v\| \leq \xi\sqrt{\mu}$ in at most*

$$\Upsilon\left(\mathcal{O}\left(n^{3/2}\sqrt{\lambda_U/\lambda_L}\right)\right)$$

iterations.

It is important to observe that, although the requirements given in this subsection are sufficient to ensure that the resulting Δw is a (τ_p, τ_q)-search direction, they are not necessary. Indeed, it is only necessary to check the sizes of $\|p\|_\infty$ and $|||g|||_Q^2 + \|q\|^2$ to ensure that the resulting Δw is a (τ_p, τ_q)-search direction. Once a candidate vector v is generated, then $(g, p, q) \in \mathcal{R}(Q) \times \mathbb{R}^n \times \mathbb{R}^l$ (and their corresponding magnitudes) can be easily computed according to (195).

3.8.2 Specific Applications

In this subsection, we present two examples of matrices F which are (λ_L, λ_U)-approximations of $\widehat{D}^2$, and an estimation of their corresponding constants λ_L and λ_U. As a consequence, we will obtain specific expressions for the iteration complexity developed in Theorem 3.8.4 when the iterative solver used to obtain an approximate solution to the HANE is the preconditioned conjugate gradient (PCG) method with preconditioner given by $\widehat{A}F\widehat{A}^T$.

The first example of a matrix F we will consider in this subsection is the maximum weight basis (MWB) preconditioner originally proposed by Vaidya [63] (see also [53]). For the purposes of this example only, we will assume that Q is diagonal, which clearly implies that $\widehat{D}$ is also diagonal. The MWB is a basis B of $\widehat{A}$ formed by giving higher priority to columns of $\widehat{A}$ corresponding to larger diagonal elements of $\widehat{D}$. The MWB preconditioner is then given by $\widehat{T}^{-1}\widehat{T}^{-T}$, where $\widehat{T} = \widehat{D}_{\mathcal{B}}^{-1}B^{-1}$ and $\widehat{D}_{\mathcal{B}}$ is the diagonal submatrix of $\widehat{D}$ corresponding to the columns of B. (See [44] for a complete description of the MWB). Note that this preconditioner can be written as

$$G \;=\; B\widehat{D}_{\mathcal{B}}^2 B^T \;=\; \widehat{A}\begin{pmatrix} \widehat{D}_{\mathcal{B}}^2 & 0 \\ 0 & 0 \end{pmatrix}\widehat{A}^T \;=\; \widehat{A}F\widehat{A}^T,$$

where

$$F = \begin{pmatrix} \widehat{D}_{\mathcal{B}}^2 & 0 \\ 0 & 0 \end{pmatrix}.$$

It is clear from this definition that $0 \preceq F \preceq \widehat{D}^2$. Next, Lemma 2.1 in [44] implies that $\|\widehat{T}\widehat{A}\widehat{D}\| \le \varphi_{\widehat{A}}$, where $\varphi_{\widehat{A}}$ is defined as

$$\varphi_{\widehat{A}} \;:=\; \max\{\|B^{-1}\widehat{A}\|_F : B \text{ is a basis for } \widehat{A}\}.$$

It follows that $\widehat{T}H\widehat{T}^T = \widehat{T}(\widehat{A}\widehat{D}^2\widehat{A}^T)\widehat{T}^T \preceq \varphi_{\widehat{A}}^2 I$, which implies that $G \succeq \varphi_{\widehat{A}}^{-2}H$. In view of definition 2, we have thus shown that F is a $(\varphi_{\widehat{A}}^{-2}, 1)$-approximation of $\widehat{D}^2$.

Another way of obtaining an approximation of $\widehat{D}^2$ is by using the partial updating strategy which was first proposed by Karmarkar [24] (see also Gonzaga [19]) in the context of primal-only interior-point methods, and extended by Monteiro and Adler [40] and Kojima et. al. [29] to the context of primal-dual path-following methods. At each iteration of a path-following algorithm, the strategy consists of generating a diagonal matrix $\bar{D}$ satisfying

$$\rho^{-1}\frac{s_i}{x_i} \;\le\; \bar{D}_{ii} \;\le\; \rho\frac{s_i}{x_i}, \quad \text{for all } i = 1, ..., n \tag{208}$$

for some constant $\rho \ge 1$, and using

$$F \;:=\; \begin{pmatrix} (Q+\bar{D})^{-1} & 0 \\ 0 & E^{-2} \end{pmatrix} \tag{209}$$

as the approximation for $\widehat{D}^2$. The current approximation $\bar{D}$ is obtained by updating the approximation used at the previous iterate in the following manner. If the ith diagonal element of $\bar{D}$ used at the previous iterate violates (208), then it is changed to s_i/x_i; otherwise it is left unchanged. Using (182), (183), (208), and (209), we easily see that $\rho^{-1}\widehat{D}^2 \preceq F \preceq \rho\widehat{D}^2$, which implies that $G = \widehat{A}F\widehat{A}^T \succeq \rho^{-1}H$. Hence F is a (ρ^{-1}, ρ)-approximation of $\widehat{D}^2$.

In the remainder of this subsection, we will obtain specific expressions for the iteration complexity developed in Theorem 3.8.4 when the iterative solver used to obtain an approximate solution to the HANE is the PCG method with preconditioner $\widehat{A}F\widehat{A}^T$, where F is obtained via the MWB and partial update methods, respectively. It should be noted that under exact arithmetic, the PCG algorithm is in fact a finite termination algorithm, achieving an exact solution to the HANE in at most $m+l$ iterations, since $H \in \mathcal{S}^{m+l}_{++}$ (see for example [25, 35]). For our purposes, we will view the PCG method as an iterative method, which is known to satisfy the following convergence property: if $G \in \mathcal{S}^{m+l}_{++}$ is used as a preconditioner for the HANE, then the method obtains an iterate u such that $\|f(u)\|_{H^{-1}} \le \delta^{-1}\|f(u^0)\|_{H^{-1}}$ in at most

$$\Upsilon(\delta) \;=\; \mathcal{O}\left\{\sqrt{\kappa(G^{-1}H)}\log\delta\right\} \tag{210}$$

iterations, where we recall that $\kappa(\cdot)$ represents the spectral condition number of $(\cdot)$. The following lemma gives a bound on the spectral condition number of $G^{-1}H$ when $G = \widehat{A}F\widehat{A}^T$ and F is a (λ_L, λ_U)-approximation of $\widehat{D}^2$.

Lemma 3.8.5 *Suppose that F is a (λ_L, λ_U)-approximation of $\widehat{D}^2$, and define $G := \widehat{A}F\widehat{A}^T$. Then, $\kappa(G^{-1}H) \le \lambda_U/\lambda_L$.*

Proof: Let L be an invertible matrix such that $LL^T = G^{-1}$. We observe that $G^{-1}H$ and L^THL are similar, and hence $\kappa(L^THL) = \kappa(G^{-1}H)$. Since F is a (λ_L, λ_U)-approximation of $\widehat{D}^2$, we have that $F \preceq \lambda_U\widehat{D}^2$ and $G \succeq \lambda_L H$. These relations, along with (185) and the definition of G, imply that $\lambda_L H \preceq G \preceq \lambda_U H$. This observation together with the fact that $G = L^{-T}L^{-1}$ then implies that $\lambda_U^{-1}I \preceq L^THL \preceq \lambda_L^{-1}I$, and hence that $\kappa(G^{-1}H) = \kappa(L^THL) \le \lambda_U/\lambda_L$. ∎

Using Lemma 3.8.5 along with (210), we see that Theorem 3.8.4 yields the iteration complexity bound

$$\mathcal{O}\left\{\sqrt{\lambda_U/\lambda_L}\,\log(n\,\lambda_U/\lambda_L)\right\} \tag{211}$$

for the PCG method with preconditioner $G = \widehat{A}F\widehat{A}^T$, where F is a (λ_L, λ_U)-approximation of $\widehat{D}^2$. For the MWB and partial update preconditioners, this bound becomes $\mathcal{O}(\varphi_{\widehat{A}}\log(n\varphi_{\widehat{A}}))$ and $\mathcal{O}(\rho\log(n\rho))$ iterations, since the respective matrices F are $(\varphi_{\widehat{A}}^{-2}, 1)$- and (ρ^{-1}, ρ)-approximations of $\widehat{D}^2$, respectively. We observe that the resulting bound for the MWB preconditioner is precisely the same as the one obtained in [33].

In the remaining part of this subsection, we will make a few observations about the inner iteration complexity bound (211). As mentioned in Subsection 3.7.1, it is possible to develop a short-step method based on the inexact search directions introduced in Subsection 3.7.1. When this method is started from a feasible point, then it can be shown that the inner iteration complexity bound is the same as (211), but with the factor n removed from the logarithm. Recall that the term $\log n$ in (211) follows from the fact that the ratio Ψ/ξ in Lemma 3.8.3 is $\mathcal{O}(n^{3/2})$, which in turn follows from the fact that Ψ in Lemma 3.8.2 and ξ^{-1} in (197) satisfy $\Psi = \mathcal{O}(n)$ and $\xi^{-1} = \mathcal{O}(\sqrt{n})$. In the context of a short-step feasible method, it is possible to show that for an appropriate choice of σ, γ, and θ, $\Psi = \mathcal{O}(1)$ and $\xi^{-1} = \mathcal{O}(1)$. The latter follows from the fact that the bound derived in (196) for $\|p\|$ can be reduced by a factor of $\mathcal{O}(\sqrt{n})$, and hence that ξ can be chosen as $\Theta(\min\{\tau_p, \tau_q\})$.

In view of the discussion in the previous paragraph, the short-step variant of the inexact PDIPF algorithm, started from a feasible point, has inner iteration complexity bound $\mathcal{O}(\rho \log \rho)$ if the partial update preconditioner is used to solve the HANE. It is interesting to compare this bound with the inner iteration complexity bound of the inexact path-following method presented by Anstreicher in [4]. His paper presents a short-step, dual-only, path-following method with feasible starting point, where the normal equation is solved by the PCG method using the partial update preconditioner. It shows that the outer and inner complexity bounds are $\mathcal{O}(\sqrt{n} \log \epsilon^{-1})$ and $\mathcal{O}(\rho)$ iterations, respectively. In order to minimize the overall arithmetic complexity of his method, including the work of updating the preconditioner through a series of rank-one updates, Anstreicher shows that the best choice for ρ is $\rho = \mathcal{O}(m^{\beta})$ for some $\beta \in (0, 1/2)$, which yields the optimal arithmetic complexity of $\mathcal{O}((n^3/\log n) \log \epsilon^{-1})$.

Note that the inner iteration complexity bound in [4] is a factor of $\log \rho = \mathcal{O}(\log(\lambda_U/\lambda_L))$ better than the same bound in our method. The main reason for this difference is that, while Anstreicher's method generates an iterate u satisfying

$$\frac{\|f(u)\|_{H^{-1}}}{\|f(u^0)\|_{H^{-1}}} \;\leq\; \delta^{-1}, \tag{212}$$

where $\delta = \mathcal{O}(1)$, our method generates an iterate u such that $\|\widehat{D}^{-1}v(F,u)\|/(\xi\sqrt{\mu}) \leq 1$. Noting that Lemmas 3.8.1 and 3.8.2 imply that

$$\frac{\|\widehat{D}^{-1}v(F,u)\|}{\xi\sqrt{\mu}} \;=\; \frac{\Psi}{\xi}\cdot\frac{\|\widehat{D}^{-1}v(F,u)\|}{\Psi\sqrt{\mu}} \;\leq\; \frac{K\Psi}{\xi}\frac{\|f(u)\|_{H^{-1}}}{\|f(u^0)\|_{H^{-1}}},$$

where $K = \sqrt{\lambda_U/\lambda_L}$, our requirement on the iterate u can be accomplished by enforcing (212) with $\delta = K\Psi/\xi$. Since, for a short-step method with a feasible starting point, we have that this choice of δ satisfies $\delta = \mathcal{O}(\rho)$, it follows that our inner iteration complexity has an additional $\log\delta = \mathcal{O}(\log\rho)$ factor compared to the complexity of [4]. Note that if the ideal choice of $v = v_{LS}$ given by (199) is made, then $K = 1$ in view of (200) and $\delta = \mathcal{O}(1)$. Then we would have an inner iteration complexity bound of $\mathcal{O}(\rho)$, the same as in [4]. Hence, the dual-only method in [4] can be thought of as being comparable, in terms of the number of inner iterations, to the inexact PDIPF algorithm proposed in this paper, with this ideal (but expensive) choice of inexact search direction. Note that, since the left hand side of (212) cannot be computed, and hence cannot be used to check for early termination of the PCG method, exactly $\Upsilon(\delta)$ iterations of the PCG method must be performed at each outer iteration of Anstreicher's algorithm, where $\Upsilon(\delta)$ is given by (210). In this respect, our approach is preferable to the one in [4], since it has a measurable termination criterion, namely $\|\widehat{D}^{-1}v(F,u)\|/(\xi\sqrt{\mu}) \leq 1$. It is possible to incorporate a measurable stopping criterion into Anstreicher's approach, but in that case, the resulting inner iteration complexity bound would increase to $\mathcal{O}(\rho\log\rho)$, the same bound as in our method.

3.9 CQP Algorithm for a Class of Preconditioners: Technical Results

In this subsection, we present the proof of Theorem 3.7.2 and Lemma 3.8.2. Subsection 3.9.1 presents the proof of Theorem 3.7.2, while Subsection 3.9.2 gives the proof of Lemma 3.8.2.

3.9.1 Proof of Theorem 3.7.2

In this subsection, we prove Theorem 3.7.2 by showing that the inexact PDIPF algorithm of Subsection 3.7.1 is completely equivalent to the algorithm presented in [33], and hence has similar convergence properties as the latter one.

Proof of Theorem 3.7.2: Let $\widetilde{V}$ be a matrix of full column rank such that $Q = \widetilde{V}\widetilde{V}^T$. It is clear that we may write $\mathbf{Q} = \mathbf{V}\mathbf{E}^2\mathbf{V}^T$, where

$$\mathbf{V} := \begin{pmatrix} V & \widetilde{V} \end{pmatrix}, \qquad \mathbf{E} := \begin{pmatrix} E & 0 \\ 0 & I \end{pmatrix}.$$

Note that $\mathbf{Q}$ has the form required for the inexact PDIPF algorithm in [33]. Recall that the algorithm in [33] generates a sequence of iterates $\mathbf{w}^k = (x^k, s^k, y^k, (z^k, \tilde{z}^k))$ to approximate a solution of the equivalent reformulation of the optimality conditions (161)–(164):

$$\begin{aligned} Ax &= b, \quad x \geq 0, \\ A^T y + s + Vz + \widetilde{V}\tilde{z} &= c, \quad s \geq 0, \\ Xs &= 0, \\ EV^T x + E^{-1} z &= 0, \\ \widetilde{V}^T x + \tilde{z} &= 0. \end{aligned}$$

More specifically, the algorithm in [33] generates a sequence of points $\mathbf{w}^k$ which lie in the neighborhood $\mathbf{N}_{\mathbf{w}^0}(\gamma, \theta) := \cup_{\eta \in [0,1]} \mathbf{N}_{\mathbf{w}^0}(\eta, \gamma, \theta)$, where

$$\mathbf{N}_{\mathbf{w}^0}(\eta, \gamma, \theta) := \left\{ \mathbf{w} \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l+\tilde{l}} : \begin{array}{c} Xs \geq (1-\gamma)\mu e, \ (r_p, \tilde{r}_d) = \eta(r_p^0, \tilde{r}_d^0), \ \eta \leq \mu/\mu_0, \\ \|r_V - \eta r_V^0\|^2 + \|r_{\widetilde{V}} - \eta r_{\widetilde{V}}^0\|^2 \leq \theta^2 \mu \end{array} \right\},$$

and the residuals $\tilde{r}_d$ and $r_{\widetilde{V}}$ are defined as

$$\begin{aligned} \tilde{r}_d &:= A^T y + s + Vz + \widetilde{V}\tilde{z} - c, \\ r_{\widetilde{V}} &:= \widetilde{V}^T x + \tilde{z}. \end{aligned}$$

Given a point $\mathbf{w} \in \mathbf{N}_{\mathbf{w}^0}(\gamma, \theta)$, the inexact algorithm in [33] generates a (τ_p, τ_q)-search direction $\mathbf{\Delta w} = (\Delta x, \Delta s, \Delta y, (\Delta z, \widetilde{\Delta z}))$, which in that context means a search direction satisfying

$$\begin{aligned} A\Delta x &= -r_p, \\ A^T\Delta y + \Delta s + V\Delta z + \widetilde{V}\widetilde{\Delta z} &= -\tilde{r}_d, \\ X\Delta s + S\Delta x &= -Xs + \sigma\mu e - p, \\ EV^T\Delta x + E^{-1}\Delta z &= -r_V + q, \\ \widetilde{V}^T\Delta x + \widetilde{\Delta z} &= -r_{\widetilde{V}} + \tilde{q}, \end{aligned}$$

for some vectors p, q, and $\tilde{q}$ satisfying $\|p\|_\infty \leq \tau_p\mu$ and $\|(q,\tilde{q})\| \leq \tau_q\sqrt{\mu}$, where τ_p and τ_q are defined in (179) and (180), respectively. The inexact PDIPF algorithm in [33] determines a stepsize α in the exact same manner as steps (d) and (e) of the inexact algorithm in Subsection 3.7.1, but with w, Δw and $\mathcal{N}_{w^0}((1-\alpha)\eta,\gamma,\theta)$ replaced by $\mathbf{w}$, $\mathbf{\Delta w}$ and $\mathbf{N}_{\mathbf{w}^0}(\gamma,\theta)$, respectively, and determines the next iterate $\mathbf{w}^+$ according to $\mathbf{w}^+ = \mathbf{w} + \alpha\mathbf{\Delta w}$.

It is straightforward to show that the inexact PDIPF algorithm in Subsection 3.7.1, started at w^0 is completely equivalent to the inexact PDIPF algorithm in [33], started at $\mathbf{w}^0 = (x^0, s^0, y^0, (z^0, \tilde{z}^0))$, where $\tilde{z}^0 = -\widetilde{V}^T x^0$, due to the following claims:

1. A vector $w = (x,s,y,z) \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$ if and only if there exists a vector $\tilde{z}$ such that $\mathbf{w} = (x,s,y,(z,\tilde{z})) \in \mathbf{N}_{\mathbf{w}^0}(\eta,\gamma,\theta)$, in which case $\tilde{z}$ is unique; and

2. If w and $\mathbf{w}$ are related as in statement 1 above, a search direction $\Delta w = (\Delta x, \Delta s, \Delta y, \Delta z)$ is a (τ_p,τ_q)-search direction at w if and only if there exists a vector $\widetilde{\Delta z}$ such that the search direction $\mathbf{\Delta w} = (\Delta x, \Delta s, \Delta y, (\Delta z, \widetilde{\Delta z}))$ is a (τ_p,τ_q)-search direction at $\mathbf{w}$ (in the sense of [33]), in which case $\widetilde{\Delta z}$ is unique.

The proofs of claims 1 and 2 are based on the following observations, which are valid under the assumption that $\tilde{z}^0 = -\widetilde{V}^T x^0$:

- If $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$, let t be the unique vector such that $\widetilde{V}t = r_d - \eta r_d^0$, and define $\tilde{z} = -\widetilde{V}^T x - t$. Then $\mathbf{w} \in \mathbf{N}_{\mathbf{w}^0}(\eta,\gamma,\theta)$.

- If $\mathbf{w} \in \mathbf{N}_{\mathbf{w}^0}(\eta,\gamma,\theta)$, then we have that $\tilde{r}_d = \eta\tilde{r}_d^0 = \eta r_d^0 = r_d + \widetilde{V}r_{\widetilde{V}}$. Thus $r_d - \eta r_d^0 \in \mathcal{R}(Q)$, and statement 1 of Proposition 3.7.1 and the fact that $r_{\widetilde{V}}^0 = 0$ imply that $|||r_d - \eta r_d^0|||_Q = \|r_{\widetilde{V}}\| = \|r_{\widetilde{V}} - \eta r_{\widetilde{V}}^0\|$. It follows that $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$.

- Let Δw be a (τ_p, τ_q)-search direction with error terms $(g,p,q) \in \mathcal{R}(Q) \times \mathbb{R}^n \times \mathbb{R}^l$, let $\tilde{q}$ be the unique vector such that $\widetilde{V}\tilde{q} = g$, and let $\widetilde{\Delta z}$ be given by $\widetilde{\Delta z} = -\widetilde{V}^T \Delta x - r_{\widetilde{V}} + \tilde{q}$. Then $\mathbf{\Delta w}$ is a (τ_p, τ_q)-search direction at $\mathbf{w}$ with error terms $(p, (q, \tilde{q}))$.

- Let $\mathbf{\Delta w}$ be a (τ_p, τ_q)-search direction at $\mathbf{w}$ with error terms $(p, (q, \tilde{q}))$, and let $g = \widetilde{V}\tilde{q}$. It follows that Δw is a (τ_p, τ_q)-search direction with error terms $(g,p,q) \in \mathcal{R}(Q) \times \mathbb{R}^n \times \mathbb{R}^l$.

We leave a detailed proof of claims 1 and 2 to the reader.

Given $\epsilon > 0$, Theorem 2.2 of [33] claims that the inexact algorithm in [33] finds a point $\mathbf{w}^k \in \mathbf{N}_{\mathbf{w}^0}(\gamma, \theta)$ satisfying $\mu_k \leq \epsilon\mu_0$ in at most $\mathcal{O}(n^2 \log \epsilon^{-1})$ iterations. Translated to the inexact PDIPF algorithm in Subsection 3.7.1, this means that a point $w^k \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ satisfying $\mu_k \leq \epsilon\mu_0$ can be found in at most $\mathcal{O}(n^2 \log \epsilon^{-1})$ iterations. The remaining conditions on w^k in our theorem follow from the definition of $\mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ in (178), the fact that $\mu_k \leq \epsilon\mu_0$, and statement 3 of Proposition 3.7.1. ▪

3.9.2 Proof of Lemma 3.8.2

In this subsection, we present the proof of Lemma 3.8.2. We first present some technical lemmas.

Lemma 3.9.1 *Suppose that $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$ such that $(x^0, s^0) \geq (x^*, s^*)$ for some $w^* \in \mathcal{S}$. Then, for any $w \in \mathcal{N}_{w^0}(\eta, \gamma, \theta)$ with $\eta \in [0,1]$, $\gamma \in [0,1]$ and $\theta > 0$, we have*

$$\eta(x^T s^0 + s^T x^0) \leq \left(3n + \frac{\theta^2}{4}\right)\mu.$$

▪

Proof: Recall from Subsection 3.9.1 that any point $w \in \mathcal{N}_{w^0}(\eta, \gamma, \theta)$ can be mapped into a point $\mathbf{w} \in \mathbf{N}_{\mathbf{w}^0}(\eta, \gamma, \theta)$, such that the x and s components of w and $\mathbf{w}$ are precisely the same. The result now follows by applying Lemma 4.1 of [33] to $\mathbf{w}$. ▪

Lemma 3.9.2 *Let H be defined as in (185), and suppose that $(x,s,y,z) \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$. Then, for any $w \in \mathbb{R}^{n+l}$ we have that $\|\widehat{A}\widehat{D}w\|_{H^{-1}} \leq \|w\|$.*

Proof: Observe that $\widehat{D}\widehat{A}^T H^{-1} \widehat{A}\widehat{D}$ is a projection matrix, which implies that $\widehat{D}\widehat{A}^T H^{-1} \widehat{A}\widehat{D} \preceq$

I. Thus, for any $w \in \mathbb{R}^{n+l}$ we have that

$$\|\widehat{A}\widehat{D}w\|_{H^{-1}} \;=\; \sqrt{w^T(\widehat{D}\widehat{A}^T H^{-1}\widehat{A}\widehat{D})w} \;\le\; \sqrt{w^T w} \;=\; \|w\|.$$

■

For the purpose of the next proof, let us define

$$J(\sigma) \;:=\; -(XS)^{1/2}e + \sigma\mu(XS)^{-1/2}e. \tag{213}$$

Lemma 3.9.3 *Suppose $w^0 \in \mathbb{R}^{2n}_{++} \times \mathbb{R}^{m+l}$, $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$ for some $\eta \in [0,1]$, $\gamma \in [0,1]$ and $\theta > 0$, and w' satisfies (205). Let H, h and u^0 be given by (185), (187) and (206), respectively. Then,*

$$Hu^0 - h \;=\; \widehat{A}\widehat{D}\begin{pmatrix} DX^{-1/2}S^{1/2}J(\sigma) + \eta DX^{-1}\left[X(s^0-s') + S(x^0-x')\right] + D(r_d - \eta r_d^0) \\ r_V - \eta r_V^0 \end{pmatrix}. \tag{214}$$

Proof: Using the fact that $w \in \mathcal{N}_{w^0}(\eta,\gamma,\theta)$ along with (177), (184) and (205), we easily obtain that

$$\begin{aligned} \begin{pmatrix} r_p \\ E^{-1}r_V \end{pmatrix} &= \begin{pmatrix} \eta r_p^0 \\ \eta E^{-1}r_V^0 + E^{-1}(r_V - \eta r_V^0) \end{pmatrix} \\ &= \eta\widehat{A}\begin{pmatrix} x^0 - x' \\ E^{-2}(z^0 - z') \end{pmatrix} + \widehat{A}\begin{pmatrix} 0 \\ E^{-1}(r_V - \eta r_V^0) \end{pmatrix} && (215) \\ s^0 - s' &= -A^T(y^0 - y') + Q(x^0 - x') - V(z^0 - z') + r_d^0. && (216) \end{aligned}$$

From (213), we easily see that

$$-s + \sigma\mu X^{-1}e \;=\; X^{-1/2}S^{1/2}J(\sigma). \tag{217}$$

Equation (182) implies that

$$I - D^2Q \;=\; D^2(D^{-2} - Q) \;=\; D^2X^{-1}S. \tag{218}$$

Using relations (177), (183), (184), (185), (186), (206), (215) and (216), we obtain

$$
\begin{aligned}
Hu^0 - h &= \widehat{A}\widehat{D}^2\widehat{A}^T u^0 - \widehat{A}\begin{pmatrix} D^2(s - \sigma\mu X^{-1}e - r_d) \\ 0 \end{pmatrix} + \begin{pmatrix} r_p \\ E^{-1} r_V \end{pmatrix} \\
&= -\eta\widehat{A}\widehat{D}^2\widehat{A}^T \begin{pmatrix} y^0 - y' \\ z^0 - z' \end{pmatrix} - \widehat{A}\begin{pmatrix} D^2(s - \sigma\mu X^{-1}e - \eta r_d^0 - (r_d - \eta r_d^0)) \\ 0 \end{pmatrix} \\
&\quad + \begin{pmatrix} r_p \\ E^{-1} r_V \end{pmatrix} \\
&= -\eta\widehat{A}\begin{pmatrix} D^2\left(A^T(y^0 - y') - Q(x^0 - x') + V(z^0 - z') - r_d^0\right) \\ E^{-2}(z^0 - z') \end{pmatrix} \\
&\quad - \widehat{A}\begin{pmatrix} D^2\left(\eta Q(x^0 - x') - (r_d - \eta r_d^0)\right) \\ 0 \end{pmatrix} - \widehat{A}\begin{pmatrix} D^2(s - \sigma\mu X^{-1}e) \\ 0 \end{pmatrix} \\
&\quad + \begin{pmatrix} r_p \\ E^{-1} r_V \end{pmatrix} \\
&= -\eta\widehat{A}\begin{pmatrix} -D^2(s^0 - s') \\ E^{-2}(z^0 - z') \end{pmatrix} - \widehat{A}\begin{pmatrix} D^2\left(\eta Q(x^0 - x') - (r_d - \eta r_d^0)\right) \\ 0 \end{pmatrix} \\
&\quad - \widehat{A}\begin{pmatrix} D^2(s - \sigma\mu X^{-1}e) \\ 0 \end{pmatrix} + \eta\widehat{A}\begin{pmatrix} x^0 - x' \\ E^{-2}(z^0 - z') \end{pmatrix} + \widehat{A}\begin{pmatrix} 0 \\ E^{-1}(r_V - \eta r_V^0) \end{pmatrix} \\
&= \widehat{A}\begin{pmatrix} -D^2(s - \sigma\mu X^{-1}e) + \eta D^2(s^0 - s') + \eta(I - D^2Q)(x^0 - x') + D^2(r_d - \eta r_d^0) \\ E^{-1}(r_V - \eta r_V^0) \end{pmatrix}
\end{aligned}
$$

which together with (183), (217), and (218) yields (214), as desired. ▪

We now turn to the proof of Lemma 3.8.2.

Proof of Lemma 3.8.2: The fact that $w \in \mathcal{N}_{w^0}((1-\alpha)\eta, \gamma, \theta)$ implies that $w \in$

$\mathcal{N}_{w^0}(\eta,\gamma,\theta)$ for some $\eta \in [0,1]$. By Lemmas 3.9.2 and 3.9.3, we have that

$$\begin{aligned}
&\|Hu^0 - h\|_{H^{-1}} \\
&\quad = \left\| \widehat{A}\widehat{D} \begin{pmatrix} DX^{-1/2}S^{1/2}J(\sigma) + \eta DX^{-1}\left[X(s^0-s') + S(x^0-x')\right] + D(r_d - \eta r_d^0) \\ r_V - \eta r_V^0 \end{pmatrix} \right\|_{H^{-1}} \\
&\quad \le \left\| \begin{pmatrix} DX^{-1/2}S^{1/2}J(\sigma) + \eta DX^{-1}\left[X(s^0-s') + S(x^0-x')\right] + D(r_d - \eta r_d^0) \\ r_V - \eta r_V^0 \end{pmatrix} \right\| \\
&\quad \le \|DX^{-1/2}S^{1/2}\|\,\|J(\sigma)\| \;+\; \eta\|DX^{-1}\|\,\|S(x^0-x') + X(s^0-s')\| \\
&\qquad + \left\| \begin{pmatrix} D(r_d - \eta r_d^0) \\ r_V - \eta r_V^0 \end{pmatrix} \right\|. \qquad (219)
\end{aligned}$$

We will examine each norm in (219) in turn. First, since $w \in \mathcal{N}_{w^0}((1-\alpha)\eta,\gamma,\theta)$, we have that $x_i s_i \ge (1-\gamma)\mu$ for all i. It follows from a well-known result (see e.g. [30]) that

$$\|J(\sigma)\| \;\le\; \left(1 - 2\sigma + \frac{\sigma^2}{1-\gamma}\right)^{1/2} \sqrt{n\mu}. \qquad (220)$$

Moreover, using (182) and the facts that $Q \succeq 0$ and $x_i s_i \ge (1-\gamma)\mu$ for all i, we obtain that

$$\begin{aligned}
\|DX^{-1}\| &= \|X^{-1}D^2X^{-1}\|^{1/2} = \|X^{-1}(Q + X^{-1}S)^{-1}X^{-1}\|^{1/2} \\
&\le \|(XS)^{-1}\|^{1/2} \le \frac{1}{\sqrt{(1-\gamma)\mu}}. \qquad (221)
\end{aligned}$$

Similarly, we have

$$\max\{\|DX^{-1/2}S^{1/2}\|, \|DQ^{1/2}\|\} \le 1. \qquad (222)$$

Using the fact that $(x^0,s^0) \ge |(x',s')|$ and $(x^0,s^0) \ge (x^*,s^*)$ together with Lemma 3.9.1, we obtain that

$$\begin{aligned}
\eta\|S(x^0-x') + X(s^0-s')\| &\le \eta\left(\|S(x^0-x')\| + \|X(s^0-s')\|\right) \;\le\; 2\eta\left(\|Sx^0\| + \|Xs^0\|\right) \\
&\le 2\eta(x^Ts^0 + x^Ts^0) \;\le\; \left(6n + \frac{\theta^2}{2}\right)\mu. \qquad (223)
\end{aligned}$$

The fact that $\|DQ^{1/2}\| \le 1$ implies that $Q^{1/2}D^2Q^{1/2} \preceq I$, which in turn implies that $QD^2Q \preceq Q$. Next, the fact that $w \in \mathcal{N}_{w^0}((1-\alpha)\eta,\gamma,\theta)$ implies that $r_d - \eta r_d^0 = Qt$ for some vector t. We use these facts along with (170) to observe that

$$\begin{aligned}
\left\| \begin{pmatrix} D(r_d - \eta r_d^0) \\ r_V - \eta r_V^0 \end{pmatrix} \right\| &= \left[t^T (QD^2Q)t + \|r_V - \eta r_V^0\|^2 \right]^{1/2} \\
&\leq \left[t^T Qt + \|r_V - \eta r_V^0\|^2 \right]^{1/2} \\
&= \left[|||r_d - \eta r_d^0|||_Q^2 + \|r_V - \eta r_V^0\|^2 \right]^{1/2} \leq \theta\sqrt{\mu}. \qquad (224)
\end{aligned}$$

The result now follows by combining bounds (220)–(224) into (219). ▪

3.10 CQP Algorithm for a Class of Preconditioners: Concluding Remarks

In this paper, we presented two important extensions to the results of [33]. First, we extended the available choices of preconditioners in the recipe for constructing inexact search directions to a whole class of preconditioners which includes the MWB preconditioner used in [33] as a special case. These preconditioners are indexed by a positive semidefinite matrix F, and convergence using these preconditioners depends on how well F approximates $\widehat{D}^2$. Second, we presented the HANE as a new method to determine an approximate search direction in the inexact PDIPF algorithm.

In the specific case of LP, the results presented in this paper can be simplified considerably. First, the HANE reduces to the standard normal equation, since $\mathbf{Q} = 0$. It follows that the residual r_V disappears, as does (175) and the constant g in (173). These facts imply that the constant τ_q is unnecessary, and hence we may set $\xi := \gamma\sigma/(4\sqrt{n})$ in (197). It also follows that the last inequality in the definition of $\mathcal{N}_{w^0}(\eta, \gamma, \theta)$ in (177) disappears, and hence we may choose $\theta = 0$. It follows that the constants Ψ and $\widehat{\Psi}$ in Lemma 3.8.2 can be tightened by removing terms containing θ. Finally, it is clear that one may use the starting point $u^0 = 0$ for the iterative solver. This follows from the fact that only the first component of u^0 in (206) is involved in LP, and that s' may be chosen so that $y' = y^0$.

One added benefit of the MWB preconditioner $\widehat{T}$ discussed in Subsection 3.8.2 is the fact that $\widehat{T}H\widehat{T}^T \succeq I$, as was shown in [44]. Thus the Adaptive PCG (APCG) method in [43] may be used as the iterative solver to determine an approximate solution to the preconditioned HANE. The APCG method, applied to the preconditioned HANE with initial preconditioner $\widehat{T}$, determines a solution u such that $\|f\|_{H^{-1}} \leq \delta^{-1}\|f^0\|_{H^{-1}}$ in at most

$$\mathcal{O}(\log\det(\widehat{T}H\widehat{T}^T) + (m+l)^{1/2}\log\delta)$$

iterations (see [43]). Since

$$\log\det(\widehat{T}H\widehat{T}^T) \;\leq\; (m+l)\log\lambda_{\max}(\widehat{T}H\widehat{T}^T) \;\leq\; 2(m+l)\log\varphi_{\widehat{A}},$$

it follows that a suitable approximate solution to the HANE can be found in at most

$$\mathcal{O}((m+l)\log\varphi_{\widehat{A}} + (m+l)^{1/2}\log(n\varphi_{\widehat{A}})) \tag{225}$$

iterations of the APCG method. One unique feature of the APCG method is that the preconditioner $\widehat{T}$ is updated to better condition the preconditioned matrix. The bound (225) assumes that we form v according to (201) using the preconditioner $G = \widehat{T}^{-1}\widehat{T}^{-T}$ employed at the beginning of the APCG method. It would be interesting to investigate whether v could be formed using the preconditioner after it is updated during the APCG method. One question which would need to be addressed is whether the updated preconditioner fits into the form $G = \widehat{A}F\widehat{A}^T$ required for the results in Section 3.8 to hold. Exploring adaptive preconditioning strategies, such as the one employed by the APCG method, for generating inexact search directions in the context of the inexact PDIPF algorithm is certainly an interesting area for future research.

CHAPTER IV

THE ADAPTIVE PRECONDITIONED CONJUGATE GRADIENT ALGORITHM

4.1 Introduction

In this chapter, we will present a new procedure for determining the solution x_* to the system $Ax = b$, where A is a real, positive definite $n \times n$ matrix and b is a real n-dimensional vector. Solution methods for this classic problem are varied and wide-ranging; some of these methods date back centuries (e.g. Gaussian elimination), while others are more recent. However, all methods for solving the above system of equations fall into one of two primary categories: direct methods and iterative methods. Direct methods first build a factorization of A, then perform a series of substitutions to determine x_*. In contrast, iterative methods create a sequence of points $\{x_j\}$ which converge to x_*.

Iterative methods possess several advantages when compared with their direct counterparts, including (1) the development of intermediate, "approximate" solutions, (2) faster performance on sparse, well-conditioned systems, and (3) lower memory storage requirements. On the other hand, iterative methods have a convergence rate which depends on the condition number of the matrix A. This, combined with the cumulative effects of roundoff errors in finite-precision arithmetic, may make iterative methods ineffective when employed on extremely ill-conditioned systems.

The most well-known of the iterative methods is the conjugate gradient (CG) method. This method is known to have excellent theoretical properties, to include n-step finite termination. However, under finite arithmetic these properties are lost, and the CG method behaves similarly to other iterative methods, with a convergence rate proportional to the square root of the condition number of A.

In this chapter, we will adapt the CG method so that our convergence rate depends, not on the square root of the condition number of A, but on the logarithm of the determinant of A. We do this by using an adaptive preconditioning strategy, one which first determines the quality of our preconditioner matrix at the current iterate. If the quality of the preconditioner is good, then we use it to perform a standard CG iteration. If the preconditioner is of poor quality at the current iterate, the preconditioner will be updated by multiplying it by a rank-one update matrix. This update matrix, which incorporates ideas from the Ellipsoid Method (see e.g. [26, 46]) and is reminiscent of the update matrices used in space dilation methods (see e.g. [57, 58]), reduces the determinant of the preconditioned matrix, while keeping the minimum eigenvalue bounded away from zero. We note that our algorithm places one key restriction on A, namely that the minimum eigenvalue of A be at least one.

The early development of the CG method dates to the 1950's, particularly to the seminal work of Hestenes and Stiefel [21]. CG methods based on preconditioners, known as preconditioned CG (PCG) methods, we first proposed in the early 1960's. Since then, a wide array of preconditioners have been suggested, many for specific classes of problems. For descriptions of preconditioners currently employed with the PCG algorithm, see [20, 54]. The early history of the PCG method is detailed in the survey work by Golub and O'Leary [18]. The PCG method has been widely used in optimization; see for example [42, 53]. Finally, for those unfamiliar with the PCG method, a good introduction is presented in [56].

Our chapter is organized as follows. Subsection 4.1.1 presents terminology and notation which are used throughout the chapter. Section 4.4 introduces the adaptive preconditioning strategy in the context of the steepest descent method in order to avoid obfuscating its main ideas due to the challenges inherent in the analysis of the PCG method. Section 4.3 is devoted to two sets of results pertaining to the PCG method. In Subsection 4.3.1, some classical theoretical results are reviewed, and in Subsection 4.3.2, some new convergence rate results are obtained in the case where the preconditioner matrix is of good quality over the first j iterates. In Section 4.4, the adaptive preconditioning strategy is extended to the context of the PCG method and as a result, an adaptive PCG method is developed and corresponding convergence results are derived. Some numerical results regarding the APCG method are given in Section 4.5, and concluding remarks are presented in Section 4.6.

4.1.1 Terminology and Notation

Throughout this chapter, uppercase Roman letters denote matrices, lowercase Roman letters denote vectors, and lowercase Greek letters denote scalars. The set $\mathbb{R}^{n\times n}$ denotes the set of all $n\times n$ matrices with real components; likewise, the set $\mathbb{R}^n$ denotes the set of n-dimensional vectors with real components. Linear operators (except matrices) will be denoted with script uppercase letters.

Given a linear operator $\mathcal{F} : E \mapsto F$ between two finite dimensional inner product spaces $(E, \langle\cdot,\cdot\rangle_E)$ and $(F, \langle\cdot,\cdot\rangle_F)$, its adjoint is the unique operator $\mathcal{F}^* : F \mapsto E$ satisfying $\langle \mathcal{F}(u), v\rangle_F = \langle u, \mathcal{F}^*(v)\rangle_E$ for all $u \in E$ and $v \in F$. A linear operator $\mathcal{G} : E \mapsto E$ is called self-adjoint if $\mathcal{G} = \mathcal{G}^*$. Moreover, $\mathcal{G}$ is said to be positive semidefinite (resp. positive definite) if $\langle \mathcal{G}(u), u\rangle_E \geq 0$ for all $u \in E$ (resp., $\langle \mathcal{G}(u), u\rangle_E > 0$ for all $0 \neq u \in E$).

Given a matrix $A \in \mathbb{R}^{n\times n}$, we denote its eigenvalues by $\lambda_i(A)$, $i = 1, \ldots, n$; its maximum and minimum eigenvalues are denoted by $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$, respectively. If a symmetric matrix A is positive semidefinite (resp. positive definite), we write $A \succeq 0$ (resp., $A \succ 0$); also, we write $A \succeq B$ to mean $A - B \succeq 0$. Given $A \succ 0$, the condition number of A, denoted $\kappa(A)$, is equal to $\lambda_{\max}(A)/\lambda_{\min}(A)$. The size of the matrix A, denoted size(A), is the number of bits required to store the matrix A. The identity matrix will be denoted by I; its dimensions should be clear from the context. The notation $\|x\|$ denotes Euclidean norm for vectors, i.e. $\|x\| = \sqrt{x^T x}$. The function $\log \alpha$ denotes the natural logarithm of α. Finally, the set $B(0,1)$ denotes the Euclidean ball centered at the origin, i.e. $B(0,1) := \{z : \|z\| \leq 1\}$.

4.2 The Adaptive Steepest Descent Algorithm

In this section, we introduce the concept of adaptive preconditioning in the context of the preconditioned steepest descent (PSD) algorithm to develop an adaptive PSD algorithm. The section is divided into two subsections. In Subsection 4.2.1, we motivate the concept of adaptive preconditioning by discussing the PSD algorithm and showing how a generic update matrix F, applied to the original preconditioner matrix C, can improve the convergence of the algorithm. In Subsection 4.2.2, we present the update matrix F and prove that it has the properties required for the Adaptive PSD Algorithm in Subsection 4.2.1.

4.2.1 Motivation from the Steepest Descent Method

In this subsection, we discuss the motivation behind adaptive preconditioning procedures.

Let $x_* = A^{-1}b$ denote the unique solution to $Ax = b$. The following "energy" function will play an important role in the analysis of the algorithms discussed in this chapter:

$$\Phi_A(x) \ := \ \frac{1}{2}(x - x_*)^T A(x - x_*). \tag{226}$$

Notice that the gradient of this function is

$$\nabla \Phi_A(x) \ = \ Ax - b \ =: \ g(x).$$

The methods described in this chapter reduce the energy function (226) at each step. We note, however, that this function is never evaluated in the course of our algorithms. Rather, it just serves as an analytic tool in the complexity analysis of the algorithms discussed in this chapter.

All methods in this chapter are gradient methods. Recall that a gradient method (for minimizing $\Phi_A(\cdot)$) is a method which generates a sequence of iterates $\{x_k\}$ according to $x_{k+1} = x_k + \alpha_k d_k$, where d_k is a search direction satisfying $d_k^T g(x_k) < 0$ and $\alpha_k > 0$ is a stepsize chosen so that $\Phi_A(x_{k+1}) < \Phi_A(x_k)$ (see page 25 in Bertsekas [**?**]). In this chapter, we will only discuss gradient methods in which the sequence of stepsizes $\{\alpha_k\}$ is chosen using the minimization rule, i.e. $\alpha_k = \text{argmin}\{\Phi_A(x_k + \alpha d_k) : \alpha \geq 0\}$.

The following definition provides a means for determining "good" search directions in gradient methods.

Definition 3 Given a constant $\zeta > 0$ and a point $x \in \mathbb{R}^n$ such that $g := g(x) \neq 0$, we say that a search direction $0 \neq d \in \mathbb{R}^n$ is *ζ-scaled at x* if

$$\sqrt{(g^T A^{-1} g)(d^T A d)} \leq -\sqrt{\zeta}(g^T d). \tag{227}$$

It turns out that if d is ζ-scaled at x, then minimizing Φ_A along the ray $\{x+\alpha d : \alpha \geq 0\}$ yields a significant reduction in $\Phi_A(\cdot)$. Indeed, Definition 1 ensures that d is a descent direction at x, since the left hand side of (227) is strictly positive. Moreover, it can be shown that

$$\alpha_{\text{new}} \;:=\; \operatorname{argmin}\{\Phi_A(x+\alpha d) : \alpha \geq 0\} \;=\; -\frac{d^T g}{d^T A d}$$

and

$$\frac{\Phi_A(x) - \Phi_A(x_{\text{new}})}{\Phi_A(x)} = \frac{(g^T d)^2}{(g^T A^{-1} g)(d^T A d)},$$

where $x_{\text{new}} = x + \alpha_{\text{new}} d$ (see Chapter 7.6 in Luenberger [35]). Hence, if d is ζ-scaled at x, Definition 1 implies that the next iterate x_{new} satisfies

$$\Phi_A(x_{\text{new}}) \leq \left(1 - \frac{1}{\zeta}\right) \Phi_A(x). \tag{228}$$

Thus, if all search directions of a gradient method are ζ-scaled at their respective iterates, the method will obtain an iterate x_k such that $\Phi_A(x_k) \leq \epsilon \Phi_A(x_0)$ in at most $\mathcal{O}(\zeta \log \epsilon^{-1})$ iterations.

We now give a sufficient condition for a search direction d of the form $d = -CC^T g(x)$, where $C \in \mathbb{R}^{n\times n}$ is an invertible matrix, to be well-scaled at x. We first give a definition.

Definition 4 For a given constant $\nu > 0$, a matrix $C \in \mathbb{R}^{n\times n}$ is called a *ν-preconditioner at x* if C is invertible and $g^T C(C^T A C)C^T g \leq \nu \|C^T g\|^2$, where $g = g(x)$.

Proposition 4.2.1 *If $C^T A C \succeq \xi I$ for some $\xi > 0$, and if C is a ν-preconditioner at a point x such that $g = g(x) \neq 0$, then $d = -CC^T g$ is (ν/ξ)-scaled at x.*

Proof: Note first that $d \neq 0$ since $g \neq 0$ and C is invertible. The assumption $C^T A C \succeq \xi I$ implies that $(C^T A C)^{-1} \preceq \xi^{-1} I$, and hence

$$g^T A^{-1} g \;=\; (C^T g)^T (C^T A C)^{-1} (C^T g) \;\leq\; \xi^{-1}(g^T C C^T g) \;=\; -\xi^{-1} g^T d.$$

The assumptions that C is a ν-preconditioner at x and $d = -CC^T g$ clearly imply that $d^T A d \leq -\nu(g^T d)$. The conclusion that $d = -CC^T g$ is (ν/ξ)-scaled at x follows by noting Definition 1 and combining the above two inequalities. ∎

We will now outline a basic step of our adaptive preconditioned steepest descent (APSD) algorithm under the assumption that $A \succeq I$. At every iteration of this method, we generate preconditioners C satisfying Definition 4 and $C^TAC \succeq I$. For reasons that will become apparent later, we assume from now on that $\nu > n$. Suppose that x is the current iterate and C is an invertible matrix such that $C^TAC \succeq I$. (If x is the first iterate, we can set $C = I$ since we are assuming that $A \succeq I$; otherwise, we can let C be the preconditioner used to compute the search direction at the previous iterate.) If C is a ν-preconditioner at x, we can use it to generate a search direction at x according to Proposition 4.2.1. If C is not a ν-preconditioner at x, we can obtain a ν-preconditioner at x by successively post-multiplying the most recent C by an invertible matrix F satisfying the following three properties:

P1. $F = \sigma_1 I + \sigma_2 pp^T$ for some vector $p \in \mathbb{R}^n$ and constants σ_1 and σ_2,

P2. $F(C^TAC)F \;=\; (CF)^TA(CF) \;\succeq\; I$, and

P3. $\det F \leq \eta(\nu) := \sqrt{n/\nu}\exp\{(1-n/\nu)/2\} < 1$.

We will describe how to construct a matrix satisfying properties P1–P3 in Subsection 4.2.2. The process of replacing C by CF will be referred to as an *update* of C. For now, we will make a few observations regarding these updates. Property P2 ensures that the updated matrix CF still satisfies the requirement that $(CF)^TA(CF) \succeq I$. Moreover, properties P2 and P3 together ensure that after a finite number of updates of the form $C \leftarrow CF$, a ν-preconditioner C at x will be obtained. Indeed, since

$$\det F(C^TAC)F \;=\; (\det F)^2 \det C^TAC \;\leq\; \eta(\nu)^2 \det C^TAC, \tag{229}$$

we see that $\det C^TAC$ decreases by a factor of $\eta(\nu)^2$ each time an update $C \leftarrow CF$ is performed. Since by property P2, $\det C^TAC \geq 1$ for every preconditioner C generated, it is clear that only a finite number of updates can be performed.

Finally, property P1 ensures that the process of updating C to CF is a simple process requiring $\mathcal{O}(n^2)$ flops if C is kept in explicit form. On the other hand, if C is kept in factored form (i.e., $C = F_1 \cdots F_l$, where l is the total number of updates performed throughout the method), then each matrix F_j requires $\mathcal{O}(n)$ units of storage, and multiplying F_j by a vector requires $\mathcal{O}(n)$ flops.

We are now ready to state the main algorithm of this section using the above ideas.

Algorithm APSD

```
Start:  Given $A \succeq I$, $x_0 \in \mathbb{R}^n$, $b \in \mathbb{R}^n$, and constants $\nu > n$ and $\epsilon > 0$.

  1. Set $i = 0$, $g_0 = Ax_0 - b$, and $C = I$.

  2. While $\Phi_A(x_i) > \epsilon\Phi_A(x_0)$ do

      (a) $d = -CC^T g_i$

      (b) $\alpha = -g_i^T d/(d^T Ad)$

      (c) If $\alpha < \nu^{-1}$ then

            • Create an update matrix $F$ satisfying properties P1–P3 above

            • Set $C = CF$ and go to step 2(a)

          end (if)

      (d) $x_{i+1} = x_i + \alpha d$

      (e) $g_{i+1} = g_i + \alpha Ad$

      (f) Set $i = i + 1$

     end (while).
```

Let us make a few observations about the above algorithm. First, if $\nu \geq \lambda_{\max}(A)$, then it is clear that no updates will be performed and that the algorithm reduces to the standard SD algorithm. Hence, the novel and interesting case to consider is when ν is chosen so that $\nu < \lambda_{\max}(A)$. Second, notice that the test $\alpha < \nu^{-1}$ performed in step 2(c) of Algorithm APSD is equivalent to testing whether C is a ν-preconditioner at x_j. This follows by inserting the definition of d given in 2(a) into the formula for α in 2(b). Finally, observe that whenever the test in step 2(c) is satisfied, an update is made to the matrix C.

The following theorem details the main convergence results for Algorithm APSD.

Theorem 4.2.2 *Assume that $A \succeq I$ and that $\nu < \lambda_{\max}(A)$ in Algorithm APSD. Then, the following statements hold:*

(a) The number of updates is bounded by $N_\psi := (\log\det A)/(\psi^{-1} - 1 + \log\psi)$, where $\psi := \nu/n$.

(b) The number of iterates x_i generated by the algorithm cannot exceed $\lceil \nu \log \epsilon^{-1} \rceil$.

(c) The algorithm has an arithmetic complexity of $\mathcal{O}(n^2(N_\psi + \nu\log\epsilon^{-1}))$ flops if C is kept in explicit form, and $\mathcal{O}(\max\{nN_\psi, n^2\}(N_\psi + \nu\log\epsilon^{-1}))$ flops if C is kept in factored form.

Proof: For the proof of (a), let C_k denote the preconditioner matrix in Algorithm APSD after k updates have been performed. In view of (229), we have that $\det(C_k^T A C_k) \le \eta(\nu)^{2k} \det A$. Also, property P2 implies that $\det(C_k^T A C_k) \ge \det(I) = 1$. Combining these two inequalities yields $1 \le \eta(\nu)^{2k}\det A$. By taking logarithms on both sides of this equation, we get that $k \le [\log\det A]/[2\log(1/\eta(\nu))]$. Statement (a) follows by substituting the definition of $\eta(\nu)$ given in P3 into this inequality.

For (b), notice that we require $\alpha \ge \nu^{-1}$ at iterate x_j before we generate a new iterate x_{j+1}. Equivalently, we ensure that the matrix C is a ν-preconditioner at iterate x_j before generating x_{j+1}. The assumption that $A \succeq I$ and property P2 ensure that $C^T A C \succeq I$; hence Proposition 4.2.1 implies that the search direction d used to generate x_{j+1} is ν-scaled at x_j. As a result, $\Phi_A(x_{j+1}) \le (1 - \nu^{-1})\Phi_A(x_j)$ by (228), and the result follows by using standard arguments.

For the proof of (c), we begin by claiming that the process used to create an update matrix F requires $\mathcal{O}(n^2)$ flops if C is kept in explicit form, and $\mathcal{O}(\max\{nN_\psi, n^2\})$ flops if C is kept in factored form. (The proof of this fact follows immediately from Theorem 4.2.6 and equation (235) in Subsection 4.2.2.) Based on this result, it is clear that a single iteration or update of Algorithm APSD requires $\mathcal{O}(n^2)$ flops if C is kept explicitly and $\mathcal{O}(\max\{nN_\psi, n^2\})$ flops if C is kept in factored form. The result follows from this observation and statements (a) and (b). ∎

It is interesting to examine how N_ψ varies for $\psi \in (1, \lambda_{\max}(A)/n)$. Note that N_ψ is a strictly decreasing function of ψ, since the function $\psi^{-1} - 1 + \log \psi$ is strictly increasing for all $\psi > 1$. Moreover, it is easy to see that $N_\psi \to \infty$ as $\psi \downarrow 1$. Next, if we denote the eigenvalues of A by $\lambda_i(A)$, we see that

$$\log \det A \;=\; \log\left(\prod_{i=1}^{n} \lambda_i(A)\right) \;=\; \sum_{i=1}^{n} \log \lambda_i(A) \;\leq\; n \log \lambda_{\max}(A).$$

Hence, if $\psi \uparrow \lambda_{\max}(A)/n$, then we have

$$N_\psi \;\leq\; \frac{n \log \lambda_{\max}(A)}{\log \psi - 1} \;=\; \mathcal{O}\left(\frac{n \log \lambda_{\max}(A)}{\log \lambda_{\max}(A) - \log n - 1}\right) \;=\; \mathcal{O}(n).$$

Hence, the value of N_ψ decreases from infinity to $\mathcal{O}(n)$ as ψ increases from one to $\lambda_{\max}(A)/n$.

While the SD algorithm is not necessarily polynomial in n and the size of A, the following lemma shows that Algorithm APSD is, under some reasonable assumptions on $\psi \in (1, \lambda_{\max}(A)/n)$.

Lemma 4.2.3 *Let $\psi := \nu/n$, and assume that $\max\{\psi, (\psi-1)^{-1}\} = \mathcal{O}(p(n))$ for some polynomial $p(\cdot)$. Assume also that A is a rational matrix such that $A \succeq I$. Then, the arithmetic complexity of Algorithm APSD is polynomial in n and the sizes of A and ϵ^{-1}.*

Proof: Let us first get an upper bound on $h(\psi) := [\psi^{-1} - 1 + \log \psi]^{-1}$. If $\psi > 3/2$, then it is clear that $h(\psi) = \mathcal{O}(1)$, so assume that $\psi \leq 3/2$. Using the assumption that $(\psi - 1)^{-1} = \mathcal{O}(p(n))$ and the fact that $\log \psi \geq (\psi - 1) - (\psi - 1)^2/2$ for all $\psi \geq 1$, we have that

$$\begin{aligned}
[\psi^{-1} - 1 + \log \psi]^{-1} \;&\leq\; \left[\frac{1-\psi}{\psi} + (\psi - 1) - \frac{(\psi-1)^2}{2}\right]^{-1} \;=\; \left[(\psi-1)^2\left(\frac{1}{\psi} - \frac{1}{2}\right)\right]^{-1} \\
&\leq\; \left[(\psi-1)^2\left(\frac{2}{3} - \frac{1}{2}\right)\right]^{-1} \;=\; 6(\psi-1)^{-2} \;=\; \mathcal{O}(p^2(n)).
\end{aligned}$$

Hence, we see that $N_\psi = \mathcal{O}(p^2(n) \log \det A)$. Next, Theorem 3.2 of [55] shows that size$(\det A) \leq 2\,$size(A). Using this result along with the fact that $\log \det A = \mathcal{O}(\text{size}(\det A))$ and the bound on N_ψ, we have that $N_\psi = \mathcal{O}(p^2(n)\text{size}(A))$. It is clear that the assumption $\psi = \mathcal{O}(p(n))$ implies that $\nu = \psi n = \mathcal{O}(np(n))$. The result follows from these facts and Theorem 4.2.2(c). ■

4.2.2 The Ellipsoid Preconditioner

In this subsection, we will show how we can construct a matrix F which satisfies properties P1–P3.

Our first lemma gives necessary and sufficient conditions for F to satisfy P2.

Lemma 4.2.4 *Let $\widehat{A} \succ 0$ and $F \in \mathbb{R}^{n\times n}$ be given. Then, $F\widehat{A}F \succeq I$ if and only if $E(\widehat{A}) \subseteq \mathcal{E}(F)$, where*

$$\begin{aligned} \mathcal{E}(F) &:= \{Fu : u \in B(0,1)\}, \\ E(\widehat{A}) &:= \{z : z^T\widehat{A}z \le 1\}. \end{aligned}$$

Proof: Let U denote the boundary of $B(0,1)$, i.e. $U := \{u : u^Tu = 1\}$. We have

$$\begin{aligned} E(\widehat{A}) \subseteq \mathcal{E}(F) &\Leftrightarrow \quad Fu \notin \operatorname{int} E(\widehat{A}), \quad \forall\, u \in U \\ &\Leftrightarrow \quad (Fu)^T\widehat{A}(Fu) = u^T(F\widehat{A}F)u \ge 1, \quad \forall\, u \in U \\ &\Leftrightarrow \quad F\widehat{A}F \succeq I. \end{aligned}$$

■

Our update matrix F possesses the following special property: its ellipsoid $\mathcal{E}(F)$ is the minimum volume ellipsoid containing a certain "stripe" Π intersected with the unit ball. The next lemma provides the details surrounding the construction of F.

Lemma 4.2.5 *Let a unit vector $p \in \mathbb{R}^n$ and a constant $\tau < 1$ be given, and consider the stripe $\Pi := \Pi(p,\tau) = \{z : |z^Tp| \le \tau\}$. Then, the smallest volume ellipsoid containing $\Pi \cap B(0,1)$ is $\mathcal{E}(F)$, where*

$$F \;=\; F(p,\tau) \;:=\; \mu(I - pp^T) + \theta pp^T, \tag{230}$$

with

$$\theta \;=\; \theta(\tau) \;:=\; \min\{\tau\sqrt{n}, 1\}, \ \textit{and} \tag{231}$$

$$\mu \;=\; \mu(\tau) \;:=\; \sqrt{\frac{n-\theta^2}{n-1}}. \tag{232}$$

Moreover, if $\tau < n^{-1/2}$, we have

$$\det F \;\le\; \tau\sqrt{n}\;\exp\{(1-\tau^2 n)/2\} \;<\; 1. \tag{233}$$

Proof: For the proof of the first part of the lemma, see Theorem 2(ii) in [59]. We will only prove equation (233). Since p is a unit vector, it is clear that p is an eigenvector of F with eigenvalue θ. In addition, any vector perpendicular to p is also an eigenvector of F with eigenvalue μ. Since $\det F$ is equal to the product of its eigenvalues, it follows that $\det F = \theta\mu^{n-1}$. Also, the assumption that $\tau < n^{-1/2}$ and (231) imply that $\theta = \tau\sqrt{n} < 1$. Using the above observations together with the fact that $1+\omega \le \exp(\omega)$ for all $\omega \in \mathbb{R}$, we obtain

$$\begin{aligned}\det F \;&=\; \theta\mu^{n-1} \;=\; \theta\left[\frac{n-\theta^2}{n-1}\right]^{\frac{n-1}{2}} \;=\; \theta\left[1+\frac{1-\theta^2}{n-1}\right]^{\frac{n-1}{2}} \;\le\; \theta\left[\exp\left\{\frac{1-\theta^2}{n-1}\right\}\right]^{\frac{n-1}{2}}\\ &=\; \theta\exp\{(1-\theta^2)/2\} \;=\; \tau\sqrt{n}\;\exp\{(1-\tau^2 n)/2\} < 1,\end{aligned}$$

where the last inequality is due to the facts that $f(s) = s\exp[(1-s^2)/2]$ is a strictly increasing function over the interval $[0,1]$ and $f(1) = 1$. ▪

We will now show how to construct a matrix F satisfying P1–P3 under the assumptions that $\widehat{A} = C^T AC \succeq I$, $\nu > n$, and C is not a ν-preconditioner at x. First, we observe that since $\widehat{A} \succeq I$, we have $E(\widehat{A}) \subseteq B(0,1)$. Suppose now that a unit vector p and a scalar τ are chosen so that $E(\widehat{A}) \subseteq \Pi$, where $\Pi = \Pi(p,\tau)$ is the stripe defined in Lemma 4.2.5. Then, the matrix $F = F(p,\tau)$ given by (230) clearly satisfies property P1 and, by Lemma 4.2.5, we have

$$E(\widehat{A}) \;\subseteq\; \Pi \cap B(0,1) \;\subseteq\; \mathcal{E}(F).$$

This together with Lemma 4.2.4 implies that F satisfies property P2.

We will now show how to construct a unit vector p and a scalar τ so as to ensure that $E(\widehat{A}) \subset \Pi(p,\tau)$ and that property P3 also holds under the assumptions above. Indeed, since C is *not* a ν-preconditioner at x, it follows from Definition 2 that $w := C^T g(x)$ satisfies

$$w^T \widehat{A} w > \nu \|w\|^2. \tag{234}$$

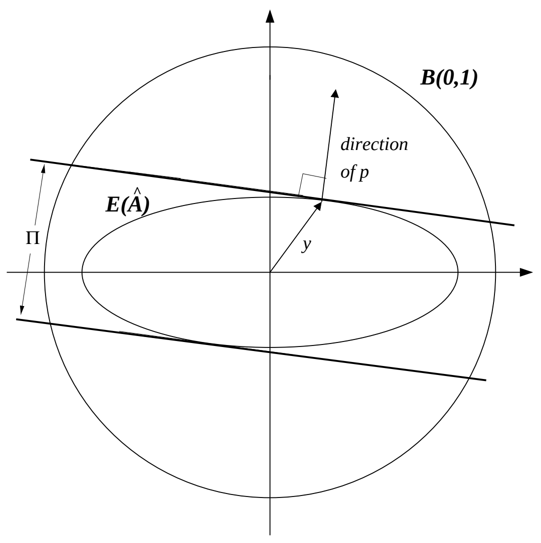

Figure 1: Vector p normal to boundary of $E(\widehat{A})$; stripe Π.

Now, letting $y := w/\sqrt{w^T\widehat{A}w}$, we have that $y^T\widehat{A}y = 1$, that is, y lies on the boundary of $E(\widehat{A})$. As Figure 1 illustrates, the boundary of our stripe Π will consist of the hyperplanes tangent to the boundary of $E(\widehat{A})$ at y and $-y$. It is easy to show that $\Pi = \{z : |z^Tp| \le \tau\}$, where

$$p \;:=\; \frac{\widehat{A}w}{\|\widehat{A}w\|} \;=\; \frac{\widehat{A}y}{\|\widehat{A}y\|} \qquad \text{and} \qquad \tau \;:=\; \frac{\sqrt{w^T\widehat{A}w}}{\|\widehat{A}w\|}. \tag{235}$$

We note that the formula for p follows from the fact that the vector $\widehat{A}y$ is normal to the boundary of $E(\widehat{A})$ at the point y, since the gradient of the function $z^T\widehat{A}z$ at y is $2\widehat{A}y$. This construction clearly implies that $E(\widehat{A}) \subseteq \Pi$.

It remains for us to show that the matrix F satisfies property P3. Indeed, by (233) and the fact that $f(s) = s\exp\{(1-s^2)/2\}$ is strictly increasing on the interval $[0,1]$, we conclude that F satisfies property P3 whenever the condition $\tau\sqrt{n} \le \sqrt{n/\nu} < 1$ holds. The latter inequality holds due to the assumption that $\nu > n$, while the first inequality is due to the fact that relations (234) and (235) and the Cauchy-Schwartz inequality imply that

$$\tau \;=\; \frac{\sqrt{w^T\widehat{A}w}}{\|\widehat{A}w\|} \;=\; \frac{w^T\widehat{A}w}{\|\widehat{A}w\|\sqrt{w^T\widehat{A}w}} \;\le\; \frac{\|w\|}{\sqrt{w^T\widehat{A}w}} \;<\; \nu^{-1/2}.$$

Notice that the construction above does not use the facts that $\widehat{A} = C^TAC$ and $w = C^Tg(x)$, but only the facts that $\widehat{A} \succeq I$ and (234) hold. Hence in the discussion above we have established the following more general result.

Theorem 4.2.6 *Let $\widehat{A} \succeq I$ be given, and let $\nu > n$ be a given constant. Suppose that a vector $w \in \mathbb{R}^n$ satisfies (234), and let p, τ, and $F = F(p, \tau)$ be determined by equations (235) and (230), respectively. Then, the matrix F satisfies properties P1–P3, i.e.*

(P1) $F = \mu I + (\theta - \mu)pp^T$*, where μ and θ are given by (231) and (232), respectively;*

(P2) $F\widehat{A}F \ \succeq \ I$*; and*

(P3) $\det F \ \leq \ \sqrt{n/\nu}\exp\{(1 - n/\nu)/2\} \ < \ 1.$

Before we end this section, we will briefly motivate the remaining part of this chapter. Note that in Algorithm APSD, we either perform a standard SD iteration or an update of the preconditioner matrix C at each step. These two sets of computations require roughly the same number of arithmetic operations in view of Theorem 4.2.2(c), and hence may be considered equivalent from a complexity standpoint. For the purpose of the discussion in this paragraph, we will refer to both sets of computations as *iterations* of the whole algorithm. Recall that the standard SD algorithm has an iteration-complexity of $\mathcal{O}(\kappa(A) \log \epsilon^{-1})$. In view of our assumption that $\lambda_{\min}(A) \geq 1$, this implies that the SD algorithm has an iteration-complexity of $\mathcal{O}(\lambda_{\max}(A) \log \epsilon^{-1})$, and that all search directions in the SD algorithm are $\lambda_{\max}(A)$-scaled. By contrast, Algorithm APSD forces its search directions to be ν-scaled at every iteration; as a result, the algorithm achieves an improved iteration-complexity of $\mathcal{O}(N_\psi + \nu \log \epsilon^{-1})$. On the other hand, the conjugate gradient (CG) algorithm is known to possess an iteration-complexity of $\mathcal{O}(\sqrt{\kappa(A)} \log \epsilon^{-1}) \leq \mathcal{O}(\sqrt{\lambda_{\max}(A)} \log \epsilon^{-1})$. Hence, it is natural to conjecture whether we can reduce its iteration-complexity to $\mathcal{O}(N_\psi + \sqrt{\nu} \log \epsilon^{-1})$ by means of an adaptive preconditioning scheme. We will show that this is indeed possible. The development of an adaptive PCG (APCG) algorithm and the proof of its convergence properties is the subject of the remainder of this chapter.

4.3 The Conjugate Gradient Method Revisited

In this section, we examine the conjugate gradient algorithm in detail. The section is divided into two subsections: Subsection 4.3.1 is devoted to a review of classical results, while Subsection 4.3.2 presents new convergence rate results obtained under the assumption that the preconditioner matrix is a good preconditioner at iterates $x_0, \ldots, x_j$.

4.3.1 Review of the Classical Conjugate Gradient Algorithm

In this subsection, we review the classical preconditioned conjugate gradient (PCG) algorithm and some of its well-known theoretical properties.

The PCG algorithm is an iterative algorithm which generates a sequence $\{x_i\}$ of approximate solutions to the system $Ax = b$, where $A \succ 0$. The PCG algorithm, which is stated next, uses an invertible matrix $Z \in \mathbb{R}^{n \times n}$ as a preconditioner.

PCG Algorithm:

```
Start:  Given $A \succ 0$, $b \in \mathbb{R}^n$, an invertible matrix $Z \in \mathbb{R}^{n\times n}$, and $x_0 \in \mathbb{R}^n$.

  1. Set $g_0 = Ax_0 - b$, $d_0 = -ZZ^T g_0$, and $\gamma_0 = \|Z^T g_0\|^2$.

  2. For $i = 0, 1, \ldots$ do

       (a) $x_{i+1} = x_i + \alpha_i d_i$, where $\alpha_i = \gamma_i/(d_i^T A d_i)$
       (b) $g_{i+1} = g_i + \alpha_i A d_i$
       (c) $\gamma_{i+1} = \|Z^T g_{i+1}\|^2$
       (d) $d_{i+1} = -ZZ^T g_{i+1} + \beta_{i+1} d_i$, where $\beta_{i+1} = \gamma_{i+1}/\gamma_i$

     end (for).
```

To present the main theoretical results associated with the PCG algorithm, we introduce the following notation.

$$\widehat{A} \;:=\; Z^T A Z, \tag{236}$$

$$\hat{g}_i \;:=\; Z^T g_i, \tag{237}$$

$$\widehat{S}_i \;:=\; \text{span}\{\hat{g}_0, \ldots, \widehat{A}^i \hat{g}_0\}, \tag{238}$$

$$S_i \;:=\; Z\widehat{S}_i = \{Zv : v \in \widehat{S}_i\}. \tag{239}$$

A well-known interpretation of the PCG method is that the sequence of points $\{\hat{x}_i\}$ defined as $\hat{x}_i := Z^{-1} x_i$ is the one that is generated by the standard CG algorithm applied to the system $\widehat{A}\hat{x} = \hat{b}$, where $\widehat{A}$ is defined in (236) and $\hat{b} := Z^T b$. Moreover, the gradient of the energy function $\Phi_{\widehat{A}}(\cdot)$ associated with this system at $\hat{x}_i$ is equal to $\hat{g}_i$ as defined in (237).

The following proposition follows from the above observations and the properties of the standard CG algorithm:

Proposition 4.3.1 *Each step i of the PCG algorithm possesses the following properties:*

(a) $\widehat{S}_i = \operatorname{span}\{\hat{g}_0, \ldots, \hat{g}_i\}$*;*

(b) $\hat{g}_i^T \hat{g}_j = 0$ *for all* $i < j$*;*

(c) $\hat{g}_i^T \widehat{A} \hat{g}_j = 0$ *for all* $i \leq j - 2$*; and*

(d) $x_i = \operatorname{argmin}\{\Phi_A(x) : x \in x_0 + S_{i-1}\}$.

Proof: See e.g. pages 295-7 of [62]. ∎

We note that these properties may fail to hold under finite arithmetic. Indeed, the PCG method relies heavily on the fact that the search directions $\hat{d}_i$ in the transformed space are conjugate to one another (i.e. $\hat{d}_i^T \widehat{A} \hat{d}_j = 0$ for $i \neq j$). However, under finite-precision arithmetic, it is well-known that the search directions will often lose conjugacy and may become linearly dependent (see e.g. [20]). As a result, the PCG algorithm tends to perform poorly on extremely ill-conditioned systems.

On the other hand, when $\widehat{A}$ is well-conditioned, the PCG algorithm performs reasonably well. As we will see in the next subsection, a weaker condition for the PCG algorithm to perform well at iterates $x_0, \ldots, x_j$ is for Z to be a good preconditioner at these iterates.

4.3.2 Revisiting the Performance of the PCG Algorithm

In this subsection, we examine the rate of convergence of the iterates $x_0, \ldots, x_j$ of the PCG algorithm under the assumption that Z is a good preconditioner at those iterates.

First, assume that Z is a ν-preconditioner at x_i and that $Z^T A Z \succeq \xi I$ for some positive constants ν and ξ. Let $\tilde{x}_{i+1}$ be the point obtained by taking a step of the PSD algorithm at x_i using Z as preconditioner. Using the fact that, by Lemma 4.2.1, $d = -ZZ^T g_i$ is

ν/ξ-scaled at x_i along with (228) and statements (a) and (d) of Proposition 4.3.1, we have that

$$\Phi_A(x_{i+1}) \;\le\; \Phi_A(\tilde{x}_{i+1}) \;\le\; \left(1 - \frac{1}{\chi}\right) \Phi_A(x_i), \tag{240}$$

where $\chi := \nu/\xi$. Hence, if Z is a ν-preconditioner at $x_0, \ldots, x_{j-1}$, we obtain

$$\Phi_A(x_j) \;\le\; \left(1 - \frac{1}{\chi}\right)^j \Phi_A(x_0).$$

It turns out that we can derive a convergence rate stronger than the one obtained above, as the following theorem states.

Theorem 4.3.2 *Assume in Algorithm PCG that Z is a ν-preconditioner at x_i for all $i = 0, \ldots, j$, and that $\widehat{A} \succeq \xi I$. Further, let us define $\chi := \nu/\xi$. Then,*

$$\Phi_A(x_j) \;\le\; 4\chi \left(\frac{\sqrt{3\chi} - 1}{\sqrt{3\chi} + 1}\right)^{2j} \Phi_A(x_0).$$

The proof of this theorem will be given at the end of this subsection after we present some technical results.

We begin with some important definitions. First, consider the linear operator $\mathcal{B}_j : \widehat{S}_j \mapsto \widehat{S}_j$ defined as

$$\mathcal{B}_j(u) \;:=\; P_{\widehat{S}_j}(\widehat{A}u), \tag{241}$$

where $P_{\widehat{S}_j}$ denotes the orthogonal projection operator from $\mathbb{R}^n$ onto $\widehat{S}_j$. Since for all $u, v \in \widehat{S}_j$,

$$u^T \mathcal{B}_j(v) \;=\; u^T P_{\widehat{S}_j}(\widehat{A}v) \;=\; (P_{\widehat{S}_j} u)^T \widehat{A} v \;=\; u^T \widehat{A} v, \tag{242}$$

and $\widehat{A} \succ 0$, it follows that $\mathcal{B}_j$ is self-adjoint and positive definite, and hence invertible. Next, define the function $\Psi_j : x_0 + S_{j-1} \mapsto \mathbb{R}$ as

$$\Psi_j(x) \;:=\; \frac{1}{2}[Z^T g(x)]^T \mathcal{B}_j^{-1} [Z^T g(x)]. \tag{243}$$

Before continuing, we need to show that $\Psi_j(\cdot)$ is well-defined on the affine space $x_0 + S_{j-1}$, i.e. that $Z^T g(x) \in \widehat{S}_j$ for all $x \in x_0 + S_{j-1}$. In fact, this assertion follows from the inclusion $\hat{g}_0 + \widehat{A}\widehat{S}_{j-1} \subseteq \widehat{S}_j$, which holds in view of (238), and the following technical result.

Lemma 4.3.3 *Define*

$$\mathcal{P}_j \;:=\; \{P_j : P_j \textit{ is a polynomial of degree at most } j \textit{ such that } P_j(0) = 1\}. \qquad (244)$$

Then, the following statements are equivalent:

i) $x \in x_0 + S_{j-1}$*;*

ii) $Z^T g(x) \in \hat{g}_0 + \widehat{A}\widehat{S}_{j-1}$*;*

iii) $Z^T g(x) \in P_j(\widehat{A})\,\hat{g}_0$ *for some* $P_j \in \mathcal{P}_j$*.*

Proof: The equivalence between ii) and iii) is obvious in view of (238) and the definition of $\mathcal{P}_j$. Now, (236), (237), and (239) imply that

$$\begin{aligned} x \in x_0 + S_{j-1} \quad &\Leftrightarrow \quad x - x_0 \in Z\widehat{S}_{j-1} \Leftrightarrow Z^T A(x - x_0) \in \widehat{A}\widehat{S}_{j-1} \\ &\Leftrightarrow \quad Z^T(g(x) - g_0) \in \widehat{A}\widehat{S}_{j-1} \;\Leftrightarrow\; Z^T g(x) \in \hat{g}_0 + \widehat{A}\widehat{S}_{j-1}, \end{aligned}$$

i.e. i) and ii) are equivalent. ▪

The relevance of the function Ψ_j is revealed by the following lemma, which relates the functions $\Phi_A(\cdot)$ and $\Psi_j(\cdot)$ on the space $x_0 + S_{j-1}$.

Lemma 4.3.4 *Let* $x \in x_0 + S_{j-1}$*, and let* x_{j+1} *be the* $j+1$*-st iterate of the PCG method. Then*

$$\Phi_A(x) - \Phi_A(x_{j+1}) \;=\; \Psi_j(x). \qquad (245)$$

Proof: Let $u \in \widehat{S}_j$ be given, and define $v := \mathcal{B}_j^{-1}(u) \in \widehat{S}_j$. By the definition of $\mathcal{B}_j$, $u = \mathcal{B}_j(v) = \widehat{A}v + p$ for some unique vector $p = p(u) \in \widehat{S}_j^{\perp}$. Thus,

$$\widehat{A}^{-1}u \;=\; v + \widehat{A}^{-1}p. \qquad (246)$$

Multiplying (246) by p^T and using the facts that $v \in \widehat{S}_j$ and $p \in \widehat{S}_j^{\perp}$, we obtain

$$u^T\widehat{A}^{-1}p \;=\; v^Tp + p^T\widehat{A}^{-1}p \;=\; p^T\widehat{A}^{-1}p. \qquad (247)$$

On the other hand, multiplying (246) by u^T and using (247), we see that

$$u^T\widehat{A}^{-1}u \;=\; u^Tv + u^T\widehat{A}^{-1}p \;=\; u^T\mathcal{B}_j^{-1}(u) + u^T\widehat{A}^{-1}p \;=\; u^T\mathcal{B}_j^{-1}(u) + p^T\widehat{A}^{-1}p. \tag{248}$$

Now, let $x \in x_0 + S_{j-1}$ be given. We will show that (248) with $u = Z^Tg(x)$ implies (245). Indeed, first note that $Z^Tg(x_{j+1}) \in \widehat{S}_j^{\perp}$ in view of (237) and statements (a) and (b) of Proposition 4.3.1. Moreover, since $x, x_{j+1} \in x_0 + S_j$, it follows from Lemma 4.3.3 that $Z^Tg(x) - Z^Tg(x_{j+1}) \in \widehat{A}\widehat{S}_j$. These two observations together imply that if $u = Z^Tg(x)$ then $p(u) = Z^Tg(x_{j+1})$. The result now follows from equality (248) with $u = Z^Tg(x)$ and $p = Z^Tg(x_{j+1})$, the definition of Ψ_j and the fact that $\Phi_A(x) = \frac{1}{2}g(x)^TA^{-1}g(x) = \frac{1}{2}(Z^Tg(x))^T\widehat{A}^{-1}(Z^Tg(x))$ for every $x \in \mathbb{R}^n$. ■

Lemma 4.3.5 *Assume in Algorithm PCG that Z is a ν-preconditioner at x_j, and that $Z^TAZ \succeq \xi I$. Then,*

$$\frac{\Phi_A(x_j)}{\Phi_A(x_0)} \le \chi\frac{\Psi_j(x_j)}{\Psi_j(x_0)},$$

where $\chi := \nu/\xi$.

Proof: Since Z is a ν-preconditioner at x_j, equation (240) holds for $i = j$. By rearranging the terms in (240) and invoking Lemma 4.3.4 with $x = x_j$, we see that

$$\Phi_A(x_j) \;\le\; \chi[\Phi_A(x_j) - \Phi_A(x_{j+1})] \;=\; \chi\Psi_j(x_j). \tag{249}$$

Moreover, Lemma 4.3.4 along with the facts that $x_0 \in x_0 + S_{j-1}$ and $\Phi_A(x_{j+1}) \ge 0$ imply that $\Phi_A(x_0) \ge \Psi_j(x_0)$. Combining this inequality with (249) yields the desired result. ■

Observe that Theorem 4.3.2 gives an upper bound on the ratio $\Phi_A(x_j)/\Phi_A(x_0)$. In view of Lemma 4.3.5, such an upper bound can be obtained by simply developing an upper bound for the ratio $\Psi_j(x_j)/\Psi_j(x_0)$, which will be accomplished in Lemma 4.3.7 below. First, we establish some bounds on the eigenvalues of $\mathcal{B}_j$ in the following result.

Lemma 4.3.6 *Assume in Algorithm PCG that Z is a ν-preconditioner at every x_i for $i = 0, \ldots, j$, and that $Z^TAZ \succeq \xi I$. Then, all of the eigenvalues of $\mathcal{B}_j$ lie in the interval*

$[\xi, 3\nu]$.

Proof: Since $\mathcal{B}_j$ is self-adjoint, its eigenvalues are all real-valued. To prove the conclusion of the lemma, it suffices to prove that $\xi u^T u \le u^T \mathcal{B}_j(u) \le 3\nu(u^T u)$ for all $u \in \widehat{S}_j$. However, in view of (242), we have that $u^T \mathcal{B}_j u = u^T \widehat{A} u$ for all $u \in \widehat{S}_j$. Thus, it suffices to prove that

$$\xi u^T u \ \le \ u^T \widehat{A} u \ \le 3\nu(u^T u) \ \text{ for all } u \in \widehat{S}_j. \tag{250}$$

To that end, let $u \in \widehat{S}_j$ be given. Since $\widehat{A} \succeq \xi I$, we have that $\xi u^T u \le u^T \widehat{A} u$, proving the first inequality in (250). Next, by Proposition 4.3.1(a), there exist $\alpha_0, \ldots, \alpha_j \in \mathbb{R}$ such that $u = \sum_{i=0}^{j} \alpha_i \hat{g}_i$. Using Proposition 4.3.1(b), we see that

$$u^T u \ = \ \left(\sum_{i=0}^{j} \alpha_i \hat{g}_i \right)^T \left(\sum_{i=0}^{j} \alpha_i \hat{g}_i \right) \ = \ \sum_{i=0}^{j} \alpha_i^2 \|\hat{g}_i\|^2. \tag{251}$$

The fact that Z is a ν-preconditioner at x_i implies that $\hat{g}_i^T \widehat{A} \hat{g}_i \le \nu \|\hat{g}_i\|^2$ for $i = 0, \ldots, j$. Using this fact, along with Proposition 4.3.1(c), the Cauchy-Schwartz inequality, and equation (251), we have that

$$\begin{aligned}
u^T \widehat{A} u \ &= \ \left(\sum_{i=0}^{j} \alpha_i \hat{g}_i \right)^T \widehat{A} \left(\sum_{i=0}^{j} \alpha_i \hat{g}_i \right) \ = \ \sum_{i=0}^{j} \alpha_i^2 \ \hat{g}_i^T \widehat{A} \hat{g}_i + 2 \sum_{0 \le i < l \le j} \alpha_i \alpha_l \ \hat{g}_i^T \widehat{A} \hat{g}_l \\
&= \ \nu \sum_{i=0}^{j} \alpha_i^2 \hat{g}_i^T \hat{g}_i + 2 \sum_{i=0}^{j-1} \alpha_i \alpha_{i+1} \hat{g}_i^T \widehat{A} \hat{g}_{i+1}, \\
&\le \ \nu \sum_{i=0}^{j} \alpha_i^2 \|\hat{g}_i\|^2 + 2 \sum_{i=0}^{j-1} |\alpha_i| |\alpha_{i+1}| \sqrt{\hat{g}_i^T \widehat{A} \hat{g}_i} \sqrt{\hat{g}_{i+1}^T \widehat{A} \hat{g}_{i+1}}, \\
&\le \ \nu \sum_{i=0}^{j} \alpha_i^2 \|\hat{g}_i\|^2 + 2 \sum_{i=0}^{j-1} |\alpha_i| \ |\alpha_{i+1}| \ (\sqrt{\nu} \|\hat{g}_i\|)(\sqrt{\nu} \|\hat{g}_{i+1}\|), \\
&\le \ \nu \sum_{i=0}^{j} \alpha_i^2 \|\hat{g}_i\|^2 + \sum_{i=0}^{j-1} \left(\nu \alpha_i^2 \|\hat{g}_i\|^2 + \nu \alpha_{i+1}^2 \|\hat{g}_{i+1}\|^2 \right), \\
&\le \ 3\nu \sum_{i=0}^{j} \alpha_i^2 \|\hat{g}_i\|^2 \ = \ 3\nu(u^T u),
\end{aligned}$$

where in the second to last inequality we use the fact that $2\beta\gamma \le \beta^2 + \gamma^2$ for all β and γ. Thus, the second inequality in (250) holds. ∎

The following lemma provides a bound on the ratio $\Psi_j(x_j)/\Psi_j(x_0)$.

Lemma 4.3.7 *Assume in Algorithm PCG that $\widehat{A} \succeq \xi I$, and that Z is a ν-preconditioner at every x_i for $i = 0, \dots, j$. Then,*

$$\frac{\Psi_j(x_j)}{\Psi_j(x_0)} \;\le\; 4\left(\frac{\sqrt{3\chi}-1}{\sqrt{3\chi}+1}\right)^{2j}.$$

Proof: Let $\lambda_0, \dots, \lambda_j$ denote the eigenvalues of $\mathcal{B}_j$. We begin by observing that since $\mathcal{B}_j$ is self-adjoint, it has an orthonormal basis of eigenvectors $v_0, \dots, v_j$ associated with its eigenvalues $\lambda_0, \dots, \lambda_j$, which all lie in the interval $[\xi, 3\nu]$ by Lemma 4.3.6. Using the fact that $Z^T g_0 = \hat{g}_0$ clearly belongs to $\widehat{S}_j$ in view of (237) and (238), we conclude that there exist $\alpha_0, \dots, \alpha_j \in \mathbb{R}$ such that $Z^T g_0 = \sum_{i=0}^{j} \alpha_i v_i$. By equation (243), we have that

$$\Psi_j(x_0) \;=\; \frac{1}{2}\left(\sum_{i=0}^{j} \alpha_i v_i\right)^T \mathcal{B}_j^{-1}\left(\sum_{i=0}^{j} \alpha_i v_i\right) \;=\; \frac{1}{2}\left(\sum_{i=0}^{j} \alpha_i v_i\right)^T \left(\sum_{i=0}^{j} \frac{\alpha_i}{\lambda_i} v_i\right) \;=\; \frac{1}{2}\sum_{i=0}^{j} \frac{\alpha_i^2}{\lambda_i}.$$

Next, let $x \in x_0 + S_{j-1}$ be given. In view of Lemma 4.3.3, there exists $P_j \in \mathcal{P}_j$ such that $Z^T g(x) = P_j(\widehat{A})\hat{g}_0$. Using (238) and the fact that $\widehat{A}u = \mathcal{B}_j(u)$ for all $u \in \widehat{S}_j$, we easily see that

$$Z^T g(x) \;=\; P_j(\widehat{A})\hat{g}_0 \;=\; P_j(\mathcal{B}_j)(\hat{g}_0) \;=\; P_j(\mathcal{B}_j)\left(\sum_{i=0}^{j} \alpha_i v_i\right) \;=\; \sum_{i=0}^{j} \alpha_i P_j(\lambda_i) v_i.$$

Thus, in view of (243), we have

$$\begin{aligned}
\Psi_j(x) \;&=\; \frac{1}{2}\left(\sum_{i=0}^{j} \alpha_i [P_j(\lambda_i)] v_i\right)^T \mathcal{B}_j^{-1}\left(\sum_{i=0}^{j} \alpha_i [P_j(\lambda_i)](v_i)\right) \\
&=\; \frac{1}{2}\left(\sum_{i=0}^{j} \alpha_i [P_j(\lambda_i)] v_i\right)^T \left(\sum_{i=0}^{j} \alpha_i [P_j(\lambda_i)] \mathcal{B}_j^{-1}(v_i)\right) \\
&=\; \frac{1}{2}\left(\sum_{i=0}^{j} \alpha_i [P_j(\lambda_i)] v_i\right)^T \left(\sum_{i=0}^{j} \frac{\alpha_i}{\lambda_i} [P_j(\lambda_i)] v_i\right) \;=\; \frac{1}{2}\sum_{i=0}^{j} \frac{\alpha_i^2}{\lambda_i}[P_j(\lambda_i)]^2 \\
&\le\; \left\{\max_{i=0,\dots,j} [P_j(\lambda_i)]^2\right\} \Psi_j(x_0) \;\le\; \max_{\lambda \in [\xi, 3\nu]} [P_j(\lambda)]^2 \Psi_j(x_0),
\end{aligned}$$

where the last inequality is due to the fact that $\lambda_i \in [\xi, 3\nu]$ for all $i = 0, \dots, j$. The last relation together with Lemma 4.3.3 then imply

$$\min_{x \in x_0 + S_{j-1}} \Psi_j(x) \;\le\; \left[\min_{P_j \in \mathcal{P}_j}\left\{\max_{\lambda \in [\xi, 3\nu]} [P_j(\lambda)]^2\right\}\right] \Psi_j(x_0) \;\le\; 4\left(\frac{\sqrt{3\chi}-1}{\sqrt{3\chi}+1}\right)^{2j} \Psi_j(x_0),$$

where the last inequality is well-known (see for example pages 55-56 of [56]). The result now follows by noting that Proposition 4.3.1(d) and the fact that, by Lemma 4.3.4, $\Psi_j(\cdot)$ and $\Phi_A(\cdot)$ differ only by a constant on $x_0 + S_{j-1}$, imply that $\Psi_j(x_j) = \min_{x \in x_0 + S_{j-1}} \Psi_j(x)$.

∎

We now note that Theorem 4.3.2 follows as an immediate consequence of Lemma 4.3.5 and Lemma 4.3.7.

4.4 An Adaptive PCG Algorithm

In this section, we will develop an algorithm which incorporates the adaptive preconditioning scheme from Section into the PCG method. We start by discussing a step of our adaptive PCG (APCG) algorithm. A *step* falls into one of the following three categories: (i) a standard PCG iteration, (ii) an update of the preconditioner matrix Z followed by a one-step backtrack in the PCG algorithm if possible, or (iii) an update of the preconditioner matrix Z followed by a restart of the PCG algorithm. To describe a general step of the APCG algorithm, suppose that we have already generated the jth iteration of the PCG algorithm using Z as a preconditioner. Assume that Z satisfies $Z^TAZ \succeq \xi I$ for some $\xi \in (0,1]$, and that Z is a ν-preconditioner at the PCG iterates $x_0, \ldots, x_{j-1}$ for some constant $\nu > n$. (In the first step of the APCG algorithm, we assume that $A \succeq I$; hence we may choose $Z = I$ and $\xi = 1$.) We will split our discussion into two cases, depending on whether Z is a ν-preconditioner at x_j.

If Z is a ν-preconditioner at x_j, then we simply perform a PCG iteration with Z as the preconditioner, which corresponds to a step of type (i). Assume from now on that Z is not a ν-preconditioner at x_j. In this case, the step of the APCG algorithm consists of updating Z and ξ, then either backtracking one PCG iteration if possible (i.e., a type (ii) step) or restarting the PCG algorithm with ξ reset to one (i.e., a type (iii) step). More specifically, let $C := \xi^{-1/2}Z$, and note that $\widehat{A} := C^TAC \succeq I$. The facts that Z is not a ν-preconditioner at x_j and that $\xi \leq 1$ imply that

$$g_j^T C(C^TAC)C^T g_j \;=\; \frac{1}{\xi^2} g_j^T Z(Z^TAZ)Z^T g_j \;>\; \frac{1}{\xi^2}\nu \|Z^T g_j\|^2 \;=\; \frac{\nu}{\xi}\|C^T g_j\|^2 \;\geq\; \nu \|C^T g_j\|^2.$$

Hence $w := C^T g_j$ satisfies equation (234); as a result, we may use Theorem 4.2.6 to generate an update matrix F satisfying properties P1–P3. Next, let $H := F/\mu$, where μ is given by (232). It is important to observe that H takes the following form:

$$H = I + \zeta p p^T,$$

where p is parallel to $\widehat{A}\hat{g}_j$ and ζ is a constant. By Proposition 4.3.1(c), we have that $p \perp \hat{g}_i$ for all $i \leq j-2$, and as a result, $Hw = w$ for all $w \in \widehat{S}_{j-2}$. Using this fact, it is easy to see that

1. The PCG algorithms corresponding to Z and ZH are completely identical up to step $j-2$ and generate the same iterate x_{j-1}, and
2. ZH is a ν-preconditioner at $x_0, \ldots, x_{j-2}$.

Moreover, the preconditioner ZH satisfies

$$(ZH)^T A (ZH) \;=\; \mu^{-2}(ZF)^T A (ZF) \;=\; \xi \mu^{-2} C^T A C \;\succeq\; \xi \mu^{-2} I.$$

Hence, by performing the updates $Z \leftarrow ZH$ and $\xi \leftarrow \xi\mu^{-2}$, we have that $Z^T A Z \succeq \xi I$. Also, by condition 2 above, if we replace j by $\max\{j-1, 0\}$, we have the conditions assumed at the beginning of this step. The process of replacing Z by ZH, ξ by $\xi\mu^{-2}$, and j by $\max\{j-1,0\}$ is a type (ii) step of the APCG algorithm. Note that a backtrack is possible if and only if $j > 0$.

The above description of a step would be complete were it not for the fact that ξ might get too small, which would adversely affect the rate of convergence of the PCG algorithm in view of Theorem 4.3.2. To prevent this from occurring, we perform a type (iii) step whenever ξ becomes too small. More specifically, fix a constant $\delta \in (0,1)$. Perform the updates $Z \leftarrow ZH$ and $\xi \leftarrow \xi\mu^{-2}$ as before (in the case where Z is not a ν-preconditioner at the current iterate x_j). Next, check to see whether $\xi > \delta$. If it is, we complete a type (ii) step by replacing $j \leftarrow \max\{j-1, 0\}$ as in the previous paragraph. Otherwise, if $\xi \leq \delta$, we complete a type (iii) step by restarting the PCG algorithm (with $j = 0$) using the last PCG iterate as the starting point with the preconditioner Z updated to $\xi^{-1/2}Z$ and ξ updated to

1 in this exact order. Note that these last two updates preserve the fact that $Z^TAZ \succeq \xi I$ and also prevents ξ from becoming too small by resetting it to one. Note also that if j is set to zero in a type (ii) step, the PCG algorithm clearly restarts, but ξ is not reset to one as is done in a type (iii) step.

We are now ready to state our main algorithm.

Algorithm APCG:

```
Start: Given $A \succeq I$, $b \in \mathbb{R}^n$, $x_0 \in \mathbb{R}^n$, and constants $\nu > n$, $\delta \in (0,1)$, and $\epsilon > 0$.

1. Set $\Phi_0 = \Phi_A(x_0)$ and $Z = I$.
2. Set $i = 0$, $\xi = 1$, $g_0 = Ax_0 - b$, $d_{-1} = 0$, $\beta_0 = 0$, and $\gamma_0 = \|Z^T g_0\|^2$.
3. While $\Phi_A(x_i) > \epsilon\Phi_0$ do
    (a) While $g_i^T Z(Z^TAZ)Z^T g_i \; > \; \nu\gamma_i$ do
        i. Build a matrix $F$ per Theorem 4.2.6 with $w := \xi^{-1/2}Z^Tg_i$ and $\widehat{A} := \xi^{-1}Z^TAZ$
        ii. Set $Z = ZF/\mu$ and $\xi = \xi\mu^{-2}$, where $\mu$ is given by (232)
        iii. If $\xi \le \delta$ then go to Step 2 with $Z := \xi^{-1/2}Z$ and $x_0 := x_i$
        iv. Set $i = \max\{i-1, 0\}$
        end (while)
    (b) $d_i = -ZZ^Tg_i + \beta_i d_{i-1}$, where $\beta_i = \gamma_i/\gamma_{i-1}$
    (c) $x_{i+1} = x_i + \alpha_i d_i$, where $\alpha_i = \gamma_i/(d_i^TAd_i)$
    (d) $g_{i+1} = g_i + \alpha_i Ad_i$
    (e) $\gamma_{i+1} = \|Z^Tg_{i+1}\|^2$
    (f) Set $i = i+1$
    end (while).
```

Note that in the above algorithm, x_i may denote several ith iterates of the PCG method, since the latter may be restarted several times during the course of Algorithm APCG. We now present the main convergence result we have obtained for Algorithm APCG, which shows that an ϵ-solution to $Ax = b$ can be obtained in $\mathcal{O}(N_\psi + \sqrt{n}\log\epsilon^{-1})$ steps.

Theorem 4.4.1 *Assume that the starting conditions of Algorithm APCG are met, and that* $\nu = \mathcal{O}(n)$ *and* $\max\{(1-\delta)^{-1}, \delta^{-1}\} = \mathcal{O}(1)$*. Then, Algorithm APCG generates a point* x_i *satisfying* $\Phi_A(x_i) \leq \epsilon\Phi_0$ *in*

$$\mathcal{O}\left(N_\psi + \sqrt{n}\log\epsilon^{-1}\right)$$

steps, where N_ψ *is defined in Theorem 4.2.2(a).*

Proof: For the purposes of this proof, we say that a *cycle* begins whenever step 2 of Algorithm APCG occurs, or equivalently, whenever ξ is reset to one, and that a cycle ends whenever a new one begins. Consider any iterate x_i in Algorithm APCG such that $\Phi_A(x_i) > \epsilon\Phi_0$, and let l denote the cycle number in which this iterate occurs. Also, for any cycle $r < l$, let i_r and y_r denote the last PCG iterate number and last PCG iterate, respectively, of that cycle. Finally, we define

$$t \ := i + \sum_{r=1}^{l-1} i_r.$$

(The index t denotes the "current" PCG iterate number, if the iterate count is not reset to 0 when a restart occurs.)

Recall that at the beginning of this section, we divided the steps in Algorithm APCG into types (i)–(iii). Our objective is to bound the number of these steps required to get to iterate x_i. Let us first consider the number of type (ii) and (iii) steps. If we define $C := \xi^{-1/2}Z$, it is clear that $C = F_1 \cdots F_k$, where the matrices F_j, $j = 1, \ldots, k$, are the ones obtained via Theorem 4.2.6. Thus, an argument similar to the one given in Theorem 4.2.2(a) can be used to show that the number of updates is bounded by N_ψ. Since each type (ii) and (iii) step requires that an update be performed, the number of type (ii) and (iii) steps is bounded by N_ψ.

Let us now consider the number of type (i) steps required to get to iterate x_i. We observe that when a type (ii) step occurs, one PCG iteration may be lost; thus, the total number of type (i) steps cannot exceed t plus the number of type (ii) steps. Hence, we have the following bound on the total number of steps:

$$\text{Total steps} \ \leq \ t + (\text{number of type (ii) steps}) + N_\psi \ = \ t + \mathcal{O}(N_\psi). \qquad (252)$$

It remains for us to determine a valid bound on t. To that end, let us examine the performance of the PCG iterates within a given cycle. In particular, consider the final step of the first cycle; it is clear that the current PCG iterate for this step is $y_1 = x_{i_1}$. At the beginning of this step, we have a preconditioner Z which is a ν-preconditioner at $x_0, \dots, x_{(i_1-1)}$ and which satisfies $Z^T AZ \succeq \xi I$. Hence, we apply Theorem 4.3.2 with $j = i_1 - 1$ and use the fact that $\Phi_A(x_{i_1}) \leq \Phi_A(x_{(i_1-1)})$ by Proposition 4.3.1(d) to obtain

$$\Phi_A(y_1) \;=\; \Phi_A(x_{i_1}) \;\leq\; \Phi_A(x_{(i_1-1)}) \;\leq\; 4\chi \left(1 - \frac{2}{\sqrt{3\chi}+1}\right)^{2(i_1-1)} \Phi_0.$$

Now y_1 also serves as the starting iterate for the second cycle; as a result, we have that

$$\Phi_A(y_2) \;\leq\; 4\chi \left(1 - \frac{2}{\sqrt{3\chi}+1}\right)^{2(i_2-1)} \Phi_A(y_1) \;\leq\; (4\chi)^2 \left(1 - \frac{2}{\sqrt{3\chi}+1}\right)^{2(i_1+i_2-2)} \Phi_0.$$

By induction, it follows that

$$\Phi_A(x_i) \;\leq\; (4\chi)^l \left(1 - \frac{2}{\sqrt{3\chi}+1}\right)^{2t-2l} \Phi_0.$$

We use this result along with the fact that $\Phi_A(x_i) > \epsilon\Phi_0$ to observe that

$$\epsilon \;<\; (4\chi)^l \left(1 - \frac{2}{\sqrt{3\chi}+1}\right)^{2t-2l} \;\leq\; (4\chi)^l \exp\left\{\frac{-4t+4l}{\sqrt{3\chi}+1}\right\}.$$

By taking logarithms on both sides of this inequality, we obtain

$$t \;<\; \frac{1}{4}\left[\left(\sqrt{3\chi}+1\right)\left(l\log(4\chi) + \log\epsilon^{-1}\right)\right] + l \;=\; \mathcal{O}\left(l\sqrt{\chi}\log\chi + \sqrt{\chi}\log\epsilon^{-1}\right). \tag{253}$$

We will now show that $l = \mathcal{O}(N_\psi/n)$. Observe that since $(1-\delta)^{-1} = \mathcal{O}(1)$, we have that $\log\delta^{-1} \geq \omega$ for some constant $\omega > 0$. Suppose that since the beginning of a cycle, we have performed k updates on the preconditioner Z, and assume that $k < \omega(n-1)$. It follows that $k < (n-1)\log\delta^{-1}$, and by rearranging terms, we have that $\delta < \exp\{-k/(n-1)\}$. However, notice that in step 3(a)ii of Algorithm APCG, we have by (232) that

$$\mu^2 \;=\; \frac{n-\theta^2}{n-1} \;\leq\; \frac{n}{n-1} \;=\; 1 + \frac{1}{n-1} \;\leq\; \exp\left\{\frac{1}{(n-1)}\right\},$$

i.e., $\mu^{-2} \geq \exp\{-1/(n-1)\}$. Hence, our current $\xi \geq \exp\{-k/(n-1)\}$, which implies that $\xi > \delta$. Thus, we can still perform another update within the same cycle. This shows that the number of updates in a complete cycle is $\geq \omega(n-1)$, which implies that $l \leq N_\psi/[\omega(n-1)] + 1 = \mathcal{O}(N_\psi/n)$.

To obtain a bound on χ, we observe that at each type (i) step, $\xi > \delta$. In view of the assumptions that $\delta^{-1} = \mathcal{O}(1)$ and $\nu = \mathcal{O}(n)$, it follows that $\chi = \nu\xi^{-1} < \nu\delta^{-1} = \mathcal{O}(n)$. We incorporate the bounds on l and χ into (253) to conclude that

$$t \;=\; \mathcal{O}\left(\frac{N_\psi \log n}{\sqrt{n}} + \sqrt{n}\log\epsilon^{-1}\right) \;=\; \mathcal{O}(N_\psi + \sqrt{n}\log\epsilon^{-1}). \tag{254}$$

The result follows by incorporating this bound into (252). ▪

It is important to observe that by using analysis similar to Lemma 4.2.3, it can be shown that Algorithm APCG is also a polynomial-time algorithm under the same assumptions as those given in that lemma.

We conclude the section by discussing some computational aspects of Algorithm APCG. It is important to note that if step 3(a) occurs repeatedly, we may find ourselves regressing through the PCG iterates. As a result, we need to either keep all of the PCG data in memory or determine a way to recreate the data as needed. When lack of memory is an issue, the latter option is the only viable alternative. In such a case, it is easy to see that all of the iterates and search directions generated by the PCG algorithm can be recreated by only storing the constants β_i in memory and by simply reversing the PCG algorithm.

4.5 Numerical Results

In this section, we provide numerical results for testing we have done using the APSD/APCG algorithms. All tests were run in Linux on a Pentium 4M 1.9 GHz processor with 1.0 GB RAM, using MATLAB Student Version 6, Release 12.

We conducted a series of preliminary tests on various algorithms, including APSD, APCG, APCG with limited preconditioners, and combinations of the above. The best algorithm from a practical standpoint, and the one whose results are presented below, is `apsd_to_pcg.m`. This algorithm begins by setting the preconditioner $C = I$, then performs the APSD algorithm from Subsection 4.2.1 until either the desired accuracy is reached, or until $\texttt{SD} \geq \texttt{P} + 20$, where `SD` is the number of steepest descent steps and `P` is the number of preconditioners created. If the desired accuracy has not yet been reached, the algorithm then transitions to the standard PCG method, using the preconditioner created in the APSD algorithm. It should be noted that the APSD algorithm in `apsd_to_pcg.m` varies from the one given in Subsection 4.2.1 to allow for matrices of the form $A \succeq \delta I$. Specifically, the following changes are made to step 2(c) to the APSD algorithm:

- Change the test from $\alpha < \nu^{-1}$ to $\alpha < (\nu\delta)^{-1}$, and
- Change the definition of τ in (235) to $\tau := \sqrt{\delta \cdot w^T \widehat{A} w}/\|\widehat{A}w\|$.

The tests below compare the convergence of Algorithm `apsd_to_pcg.m` with the standard MATLAB PCG algorithm without preconditioners. For each test, we also present the accuracy of the solution x obtained via a direct method, namely Cholesky factorization, for comparison with our method. It should be noted that on better-conditioned matrices in the tests below, Cholesky gives a much more accurate solution than our method. This follows from the fact that our method terminates once the desired accuracy of 10^{-6} is reached. We should also point out that the PCG portion of Algorithm `apsd_to_pcg.m` may terminate prior to obtaining a solution of accuracy 10^{-6}, for various reasons (for example, the PCG algorithm will terminate if a constant fails to satisfy required bounds, or if the algorithm stalls).

The matrices A and vectors x^0 and b were determined according to the following algorithms:

- Matrix A: Given user-defined dimension $\mathtt{n}$ and constant $\mathtt{const}$, for $\mathtt{i} = \mathtt{1} : \mathtt{n}$, set $\mathtt{d(i)} = \mathtt{const}^{\mathtt{i}} \times \mathtt{rand(1)}$ and $[\mathtt{Q},\mathtt{R}] = \mathtt{qr(rand(n))}$. Next, define $\mathtt{A} = \mathtt{Q} * \mathtt{diag(d)} * \mathtt{Q}'$; then set $\mathtt{A} = \mathtt{A} + \mathtt{A}'$ (to ensure symmetry); and finally set $\mathtt{A} = \mathtt{A}/\mathtt{trace(A)}$.
- Vector x^0: set `x0 = zeros(n,1)`.
- Vector b: set $\mathtt{b} = \mathtt{rand(n,1)}$.

Throughout the tests below (except as noted), the user-defined constants were set to the values shown in Table 4. Table 5 presents notation used in Tables 6–8, and Tables 6, 7, and 8 present the numerical results for test matrices 1–5, 6–10, and 11–15, respectively. In

Table 4: User-Defined Values for Algorithm `apsd_to_pcg.m` with $A \in \mathbb{R}^{n\times n}$

Constant	Value
ν	$2n$
δ	2^{-40}
max_precond	$1.5n$
ϵ	$1e-6$

Table 5: Notation for Tables 6–8

Notation	Description
$\kappa(A)$	Condition number of A
CH acc.	Accuracy of Cholesky Factorization Method
ATP acc.	Accuracy of Algorithm `apsd_to_pcg.m` (ATP)
#SD	Number of steepest descent steps taken by ATP
#PCG	Number of PCG steps taken by ATP
#P	Number of updates made to the preconditioner C in ATP
CG acc.	Accuracy obtained by best CG iterate after 10000 iterations

many tests, the value for CG accuracy is `1e0`; this indicates that the best iteration found by CG was the starting vector x^0. Some additional comments regarding Table 8 are required here:

* Test #11: The CG method converged within the desired tolerance of 10^{-6} in 2899 iterations; in all other tests, the CG method failed to converge in 10000 iterations.

** Test #15: Algorithm `apsd_to_pcg.m` failed to converge using $\delta = 2^{-40}$; the results shown are for $\delta = 2^{-50}$.

The following observations can be deduced from the results in Tables 6–8. First, it is

Table 6: Results of tests: $n = 100$, `const` $= 1.3$

Test	$\kappa(A)$	CH acc.	ATP acc.	#SD	#PCG	#P	CG acc.
1	`1.624e+12`	`2.187e-6`	`5.266e-6`	100	83	80	`1e0`
2	`6.724e+11`	`1.301e-5`	`2.218e-5`	101	72	81	`1e0`
3	`2.287e+12`	`1.039e-5`	`1.379e-5`	100	113	80	`6.238e-1`
4	`1.169e+12`	`3.213e-6`	`1.227e-5`	102	93	82	`6.994e-1`
5	`1.624e+12`	`2.922e-6`	`3.364e-6`	101	78	81	`6.561e-1`

Table 7: Results of tests: $n = 200$, `const` $= 1.1$

Test	$\kappa(A)$	CH acc.	ATP acc.	#SD	#PCG	#P	CG acc.
6	`4.761e+9`	`2.330e-9`	`9.328e-7`	83	0	200	`2.393e-2`
7	`1.017e+9`	`6.637e-9`	`6.407e-7`	83	0	200	`1.514e-2`
8	`7.166e+9`	`1.600e-8`	`1.722e-7`	98	0	200	`6.718e-1`
9	`2.813e+9`	`1.139e-8`	`8.435e-7`	87	0	199	`1.262e-1`
10	`1.163e+9`	`6.385e-9`	`8.445e-7`	88	0	201	`1.231e-2`

Table 8: Results of tests: $n = 500$, `const` varies

Test	`const`	$\kappa(A)$	CH acc.	ATP acc.	#SD	#PCG	#P	CG acc.
11*	1.02	`6.407e+6`	`6.415e-12`	`9.415e-7`	68	0	505	`7.618e-7`
12	1.03	`1.113e+8`	`3.350e-10`	`9.753e-7`	98	0	504	`2.374e-2`
13	1.04	`3.626e+10`	`3.966e-8`	`3.323e-7`	470	3	450	`1e0`
14	1.05	`1.133e+12`	`2.930e-6`	`7.863e-6`	386	502	366	`1e0`
15**	1.06	`2.185e+14`	`2.922e-4`	`3.141e-4`	449	196	429	`1e0`

clear that the APSD algorithm creates an excellent preconditioner $C = F_1 \cdots F_k$, normally after about $k = \mathcal{O}(n)$ updates. (For example, in tests 14 and 15, the final preconditioned matrices $C^T AC$ have condition number `8.804e+3` and `1.803e+3`, respectively.) Once the preconditioner is created, the PCG algorithm tends to converge quickly. Second, when the matrix is better conditioned, as in tests 6–12 in Tables 7–8, the APSD algorithm will often find a solution within the desired tolerance, without ever passing to the PCG Algorithm. Finally, the choice of δ seems to be quite important, as note (**) for test 15 indicates. Indeed, the APSD algorithm as implemented assumes that $A \succeq \delta I$, hence $\lambda_{\min}(A) \geq \delta$. If δ is chosen much larger than $\lambda_{\min}(A)$, then the updates created by the algorithm are still valid; however, the APSD algorithm may not create a sufficient number of updates, and the PCG algorithm may fail to converge as a result. We can almost always guarantee a reasonable level of convergence, simply by choosing δ sufficiently small.

We should mention one unfortunate aspect of the algorithm: it does appear that the algorithm must create almost all of the required preconditioners before quick convergence in PCG can be ensured. This fact, along with the observation that the vectors p in (235) are likely to be dense, implies that the method as currently implemented will not work as effectively on sparse matrices as it will on dense matrices. Investigating possible extensions of our algorithm to sparse matrices is certainly an important area for future research.

4.6 Concluding Remarks

It is well-known that under exact arithmetic, the standard CG algorithm terminates in at most n iterations. However, in finite-precision arithmetic, the standard CG algorithm loses this property and instead possesses an iteration-complexity bound of $\mathcal{O}(\sqrt{\kappa(A)}\log \epsilon^{-1})$. Our algorithm also loses its theoretical properties under finite arithmetic; indeed, Lemma 4.3.6 relies heavily on statements (b) and (c) of Proposition 4.3.1, which only hold under exact arithmetic. Nevertheless, one may hope to gain significant reductions in the number of CG iterations using our algorithm in finite-precision arithmetic, since the update matrices F_j have the effect of making the preconditioned matrix Z^TAZ better conditioned as the algorithm progresses.

One important assumption in our algorithm is the requirement that $A \succeq I$. It is possible to ensure that this assumption holds for matrices for which $A \succ 0$, simply by premultiplying A by some large positive constant $\omega \geq (\lambda_{\min}(A))^{-1}$. In a future paper, we wish to relax the requirement that $A \succeq I$, as well as provide further computational results measuring the performance of our approach.

CHAPTER V

CONCLUSION

The results in this thesis have been primarily in two areas. In IPMs, we were able to prove theoretical complexity results for inexact PDIPF algorithms for LP and CQP. It is clear that the inner iteration results obtained for those algorithms depend on two key facts: (1) the initial energy and desired final energy of the residual for the normal equation, and (2) the properties of the preconditioner used in the preconditioned iterative solver and in the computation of the correction term v. For the outer iteration results, we were able to show that by distributing the error from the normal equation in a suitable manner, we were still able to obtain polynomial convergence in a manner similar to the exact algorithms given in [68] and [72].

With regard to the APCG method, we were able to take an algorithm with iteration complexity $\mathcal{O}(\sqrt{\kappa(A)}\log\epsilon^{-1})$ and reduce its complexity to $\mathcal{O}(\log\det A + \sqrt{n}\log\epsilon^{-1})$ iterations through the use of adaptive preconditioning. In practice, the algorithm appears to produce $k \leq n$ updates to the preconditioners, where k denotes the number of "large" eigenvalues of A; once these updates are built, the PCG algorithm with this preconditioner performs extremely well.

Numerous extensions to this research are possible. With regard to IPMs, the most likely extension of our results lies in the area of semidefinite programming (SDP). In SDP, numerous problems arise for which it is much easier to multiply by the normal equation operator $\mathcal{A}\mathcal{E}^{-1}\mathcal{F}\mathcal{A}^*$ than it is to form and factorize it as a matrix. In this case, iterative methods such as in Chapter 4 might be extremely useful in obtaining an approximate solution to the normal equation. Distributing the error from the normal equation would then ensure polynomial convergence in the outer iterations, as we have proven for LP and CQP.

One of the current drawbacks of the APCG method lies in the fact that numerous updates must be created before substantial improvement is seen in the number of CG iterations. This can create substantial memory issues on large, sparse problems. Two possible solutions are (1) developing a new method for finding the vectors w which strengthens the updated preconditioner and (2) making the vectors p sparse, thus making it easier to store and multiply by the vectors. These possibilities are certainly important areas for potential research.

REFERENCES

[1] Altman, A., "Higher order primal-dual interior point method for separable convex quadratic optimization," *Control and Cybernetics*, vol. 25, pp. 761–772, 1996.

[2] Altman, A. and Gondzio, J., "Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization," *Optimization Methods and Software*, vol. 11–12, pp. 275–302, 1999.

[3] Anderson, E., "Personal conversation," August 20, 2003.

[4] Anstreicher, K., "Linear programming in $\mathcal{O}(n^3/(\ln n)\mathrm{L})$ operations," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 803–812, 1999.

[5] Baryamureeba, V. and Steihaug, T., "On the convergence of an inexact primal-dual interior point method for linear programming," *Journal of Optim. Theory Appl.* To appear.

[6] Baryamureeba, V., Steihaug, T., and Zhang, Y., "Properties of a class of preconditioners for weighted least squares problems," Tech. Rep. 16, Department of Computational and Applied Mathematics, Rice University, 1999.

[7] Ben-Tal, A. and Nemirovski, A., *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications.* SIAM, 2001.

[8] Bergamaschi, L., Gondzio, J., and Zilli, G., "Preconditioning indefinite systems in interior point methods for optimization," *Comput. Optim. Appl.*, vol. 28, no. 2, pp. 149–171, 2004.

[9] Boman, E., Chen, D., Hendrickson, B., and Toledo, S., "Maximum-weight-basis preconditioners," *Numer. Linear Algebra Appl.*, vol. 11, pp. 695–721, 2004.

[10] Carpenter, T., Lustig, I., Mulvey, J., and Shanno, D., "Higher order predictor-corrector interior-point methods with application to quadratic programming," *SIAM Journal on Optimization*, vol. 3, pp. 696–725, 1993.

[11] Carpenter, T., Lustig, I., Mulvey, J., and Shanno, D., "Separable quadratic programming via a primal-dual interior point method and its use in a sequential procedure," *ORSA Journal on Computing*, vol. 5, no. 2, pp. 182–191, 1993.

[12] Carpenter, T. and Vanderbei, R., "Symmetric indefinite systems for interior-point methods," *Mathematical Programming*, vol. 58, pp. 1–32, 1993.

[13] Chvatal, V., *Linear Programming.* W.H. Freeman, 1983.

[14] Cook, W., Cunningham, W., Pulleybank, W., and Schrijver, A., *Combinatorial Optimization.* Wiley, 1997.

[15] Dikin, I., "Iterative solution of problems of linear and quadratic programming (in Russian)," *Doklady Akademia Nauk SSSR*, vol. 8, pp. 674–675, 1967.

[16] Durazzi, C., Ruggiero, V., and Zanghirati, G., "Parallel interior-point method for linear and quadratic programs with special structure," *Journal of Optimization Theory and Applications*, vol. 110, no. 2, pp. 289–313, 2001.

[17] Freund, R., Jarre, F., and Mizuno, S., "Convergence of a class of inexact interior-point algorithms for linear programs," *Mathematics of Operations Research*, vol. 24, no. 1, pp. 50–71, 1999.

[18] Golub, G. and O'Leary, D., "Some history of the conjugate gradient and Lanczos algorithms: 1948-1976," *SIAM Review*, vol. 31, pp. 50–102, 1989.

[19] Gonzaga, C., "An algorithm for solving linear programming problems in $\mathcal{O}(n^3\mathrm{L})$ operations," *in Progress in Mathematical Programming: Interior-Point and Related Methods, ch. 1*, pp. 1–28, 1989.

[20] Greenbaum, A., *Iterative Methods for Solving Linear Systems.* SIAM, 1997.

[21] Hestenes, M. and Stiefel, E., "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–436, 1952.

[22] Judice, J., Patricio, J., Portugal, L., Resende, M., and Veiga, G., "A study of preconditioners for network interior point methods," *Computational Optimization and its Applications*, vol. 24, pp. 5–35, 2003.

[23] Kapoor, S. and Vaidya, P., "Fast algorithms for convex quadratic programming in multicommodity flows," *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pp. 147–159, 1986. Berkeley, CA.

[24] Karmarkar, N., "A new polynomial-time algorithm for linear programming," *Combinatorica*, vol. 4, pp. 373–395, 1984.

[25] Kelley, C., *Iterative Methods for Linear and Nonlinear Equations.* SIAM, 1995.

[26] Khachiyan, L., "A polynomial algorithm in linear programming," *Doklady Akademia Nauk SSSR*, vol. 244, pp. 1093–1096, 1979.

[27] Klee, V. and Minty, G., "How good is the simplex algorithm?," *Inequalities, III (Proc. Third Sympos., UCLA, Calif., 1969; dedicated to the memory of Theodore S. Motzkin)*, pp. 159–175, 1972.

[28] Kojima, M., Megiddo, N., and Mizuno, S., "A primal-dual infeasible-interior-point algorithm for linear programming," *Mathematical Programming*, vol. 61, no. 3, pp. 263–280, 1993.

[29] Kojima, M., Mizuno, S., and Yoshise, A., "A polyonimal-time algorithm for a class of linear complementarity problems," *Mathematical Programming*, vol. 44, no. 1, pp. 1–26, 1989.

[30] Kojima, M., Mizuno, S., and Yoshise, A., "A primal-dual interior point algorithm for linear programming," *in Progress in Mathematical Programming: Interior-Point and Related Methods, ch. 2*, pp. 29–47, 1989.

[31] Korzak, J., "Convergence analysis of inexact infeasible-interior-point algorithms for solving linear programming problems," *SIAM Journal on Optimization*, vol. 11, no. 1, pp. 133–148, 2000.

[32] Kovacevic-Vujcic, V. and Asic, M., "Stabilization of interior-point methods for linear programming," *Computational Optimization and Applications*, vol. 14, pp. 331–346, 1999.

[33] Lu, Z., Monteiro, R., and O'Neal, J., "An iterative solver-based infeasible primal-dual path-following algorithm for convex QP," tech. rep., Georgia Institute of Technology, 2004. Conditionally accepted in *SIAM Journal on Optimization.*

[34] Lu, Z., Monteiro, R., and O'Neal, J., "An iterative solver-based primal-dual path-following algorithm for convex QP based on a class of preconditioners," tech. rep., Georgia Institute of Technology, 2005.

[35] Luenberger, D., *Linear and Nonlinear Programming.* Addison-Wesley, 1984.

[36] Lustig, I., "Feasibility issues for primal-dual interior-point method in linear programming," *Mathematical Programming*, vol. 49, pp. 145–162, 1991.

[37] Megiddo, N., "Pathways to the optimal set in linear programming," *in Progress in Mathematical Programming: Interior-Point and Related Methods, ch. 8*, pp. 131–158, 1989.

[38] Mehrotra, S., "On the implementation of a primal-dual interior point method," *SIAM Journal on Optimization*, vol. 2, pp. 575–601, 1992.

[39] Mizuno, S. and Jarre, F., "Global and polynomial-time convergence of an infeasible-interior-point algorithm using inexact computation," *Mathematical Programming*, vol. 84, pp. 357–373, 1999.

[40] Monteiro, R. and Adler, I., "Interior path-following primal-dual algorithms, part I: Linear programming," *Mathematical Programming*, vol. 44, pp. 27–41, 1989.

[41] Monteiro, R. and Adler, I., "Interior path-following primal-dual algorithms, part ii: Convex quadratic programming," *Mathematical Programming*, vol. 44, pp. 43–66, 1989.

[42] Monteiro, R. and O'Neal, J., "Convergence analysis of a long-step primal-dual infeasible interior-point LP algorithm based on iterative linear solvers," tech. rep., Georgia Institute of Technology, 2003.

[43] Monteiro, R., O'Neal, J., and Nemirovski, A., "A new conjugate gradient algorithm incorporating adaptive ellipsoid preconditioning," tech. rep., Georgia Institute of Technology, 2004.

[44] Monteiro, R., O'Neal, J., and Tsuchiya, T., "Uniform boundedness of a preconditioned normal matrix used in interior point methods," *SIAM Journal on Optimization*, vol. 15, no. 1, pp. 96–100, 2004.

[45] Monteiro, R. and Pang, J., "On two interior-point mappings for nonlinear semidefinite complementarity problems," *Mathematics of Operations Research*, vol. 23, pp. 39–60, 1998.

[46] Nemirovski, A. and Yudin, D., "Optimization methods adapting to the 'significant' dimension of the problem," *Automatica i Telemekhanika*, vol. 38, pp. 75–87, 1977.

[47] Nesterov, Y. and Nemirovskii, A., *Interior-Point Polynomial Algorithms in Convex Programming.* SIAM, 1995.

[48] Oliveira, A. and Sorensen, D., "Computational experience with a preconditioner for interior point methods for linear programming," Tech. Rep. 28, Department of Computational and Applied Mathematics, Rice University, 1997.

[49] Portugal, L., Resende, M., Veiga, G., and Judice, J., "A truncated primal-infeasible dual-feasible network interior point method," *Networks*, vol. 35, pp. 91–108, 2000.

[50] Renegar, J., "A polynomial-time algorithm, based on Newton's method, for linear programming," *Mathematical Programming*, vol. 40, no. 1, pp. 59–93, 1988.

[51] Renegar, J., "Condition numbers, the barrier method, and the conjugate-gradient method," *SIAM Journal on Optimization*, vol. 6, pp. 879–912, 1996.

[52] Renegar, J., *A Mathematical View of Interior-Point Methods in Convex Optimization.* SIAM, 2001.

[53] Resende, M. and Veiga, G., "An implementation of the dual affine scaling algorithm for minimum cost flow on bipartite uncapacitated networks," *SIAM Journal on Optimization*, vol. 3, pp. 516–537, 1993.

[54] Saad, Y., *Iterative Methods for Sparse Linear Systems.* SIAM, 2003.

[55] Schrijver, A., *Theory of Linear and Integer Programming.* Wiley, 1986.

[56] Shewchuk, J., "An introduction to the conjugate gradient method without the agonizing pain," tech. rep., Carnegie Mellon University, 1994.

[57] Shor, N., *Minimization Methods for Non-Differentiable Functions.* Springer-Verlag, 1985.

[58] Shor, N., *Nondifferentiable Optimization and Polynomial Problems.* Kluwer, 1998.

[59] Todd, M., "On minimum volume ellipsoids containing part of a given ellipsoid," *Mathematics of Operations Research*, vol. 7, pp. 253–261, 1982.

[60] Todd, M., Tunçel, L., and Ye, Y., "Probabilistic analysis of two complexity measures for linear programming problems," *Mathematical Programming*, vol. 90, pp. 59–69, 2001.

[61] Toh, K., Tütüncü, R., and Todd, M., "Inexact primal-dual path-following algorithms for a special class of convex quadratic SDP and related problems," tech. rep., Cornell University, 2005.

[62] Trefethen, L. and Bau, D., *Numerical Linear Algebra.* SIAM, 1997.

[63] Vaidya, P., "Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners," tech. rep. A talk based on the manuscript was presented at the IMA Workshop on Graph Theory and Sparse Matrix Computation, October 1991, Minneapolis.

[64] Vanderbei, R., "Symmetric quasidefinite matrices," *SIAM Journal on Optimization*, vol. 5, pp. 100–113, 1995.

[65] Vavasis, S. and Ye, Y., "A primal-dual interior point method whose running time depends only on the constraint matrix," *Mathematical Programming A*, vol. 74, pp. 79–120, 1996.

[66] Wright, M., "Ill-conditioning and computational error in interior methods for nonlinear programming," *SIAM Journal on Optimization*, vol. 9, pp. 84–111, 1998.

[67] Wright, S., "Stability of linear equations solvers in interior-point methods," *SIAM Journal on Matrix Analysis and Applications*, vol. 16, no. 4, pp. 1287–1307, 1995.

[68] Wright, S., *Primal-Dual Interior-Point Methods.* SIAM, 1997.

[69] Wright, S., "Modified Cholesky factorizations in interior-point algorithms for linear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 1159–1191, 1999.

[70] Wright, S., "Effects of finite-precision arithmetic on interior-point methods for nonlinear programming," *SIAM Journal on Optimization*, vol. 12, pp. 36–78, 2001.

[71] Ye, Y. and Tse, E., "An extension of Karmarkar's projective algorithm for convex quadratic programming," *Mathematical Programming*, vol. 44, pp. 157–179, 1989.

[72] Zhang, Y., "On the convergence of a class of infeasible interior-point methods for the horizontal linear complementarity problem," *SIAM Journal on Optimization*, vol. 4, no. 1, pp. 208–227, 1994.

[73] Zhou, G. and Toh, K.-C., "Polynomiality of an inexact infeasible interior point algorithm for semidefinite programming," *Mathematical Programming*, vol. 99, pp. 261–282, 2004.

VITA

Jerome Wade O'Neal was born in Silver Spring, MD on December 23, 1970. Raised in Asheville, NC, he graduated from high school in 1989, then attended the United States Military Academy at West Point from 1989–1993. In 1993, he received a B.S. in Mathematics and Foreign Languages from West Point, and was commissioned as a Second Lieutenant in the U.S. Army Corps of Engineers. He served for almost eight years in the U.S. Army, culminating in his role as Commander of A Company, 588th Engineer Battalion, 4th Infantry Division (Mechanized) in 2000. He left the Army as a Captain in 2001 and entered the Ph.D. program in the School of Industrial and Systems Engineering at the Georgia Institute of Technology. He obtained a M.S. in Operations Research in 2004 and a Ph.D. in Industrial and Systems Engineering in August 2005. That same year, he began work at Delta Technology in their Research, Modelling, and Design group.