Iterative Linear Algebra for Constrained Optimization

Hilary Dollar

Keble College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2005

This thesis is dedicated to my parents Ian and Janice Dollar.

Iterative Linear Algebra for Constrained Optimization

Hilary Dollar, Keble College, University of Oxford

A thesis submitted for the degree of Doctor of Philosophy

Michaelmas 2005

Abstract

Each step of an interior point method for nonlinear optimization requires the solution of a symmetric indefinite linear system known as a KKT system, or more generally, a saddle point problem. As the problem size increases, direct methods become prohibitively expensive to use for solving these problems; this leads to iterative solvers being the only viable alternative.

In this thesis we consider iterative methods for solving saddle point systems and show that a projected preconditioned conjugate gradient method can be applied to these indefinite systems. Such a method requires the use of a specific class of preconditioners, *(extended) constraint preconditioners*, which exactly replicate some parts of the saddle point system that we wish to solve.

The standard method for using constraint preconditioners, at least in the optimization community, has been to choose the constraint preconditioner and then factor it. However, even the most basic choices in constraint preconditioner can be prohibitive to factor. We shall propose an alternative to this method: *implicit factorization constraint preconditioners.* We exhibit how these can be effectively applied whilst only having to factor smaller sub-blocks of the saddle point systems, thus reducing the expected time and memory costs of our iterative method. Numerical experiments are provided which reveal that the benefits of using implicit factorization constraint preconditioners compared to the standard method can be very dramatic.

Acknowledgements

I thank Andy Wathen, my supervisor, for his advice and enthusiasm throughout my DPhil. He has introduced me to the world of linear algebra and provided great inspiration. I thank the Oxford University Computing Laboratory and Marie-Curie Fellowship Association for funding this research. I also thank Wil Schilders for inviting me to the Technische Universiteit Eindhoven, The Netherlands, under the Marie-Curie Fellowship Scheme for six months and for all of the help that he has given me. I thank Nick Gould for his advice during this research and the use of his codes: the discussions that we have had have been invaluable. I also thank Mario Arioli for the use of his code allowing me to interface to `MA57` from within MATLAB®. I thank Oxford University Computing Laboratory, Technische Universiteit Eindhoven, Keble College, Emory University, Stanford University, the IMACS International Symposium on Iterative Methods in Scientific Computing, and the Copper Mountain Conference for their financial support which enabled me to participate in many stimulating conferences. I thank my office mates, house mates, and friends for all their support. Last but not least I thank my family, especially my Mum, Dad, Sarah, Jean, James, and Adrian, for all their love and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The Problem

One of the core components of computational mathematics is the optimization of an objective function involving unknowns that may be constrained in some way. Optimization problems occur in many different areas — nature, business, and engineering are just a small subset of such areas.

The field of continuous optimization has undergone a dramatic change since 1984 with the "interior point revolution". Each iteration of an interior point method involves the solution of a symmetric and indefinite linear system known as the Karush-Kuhn-Tucker (KKT) system (or, more generally, a saddle point system), which we shall assume to be of a sparse nature. The most common approach reduces the indefinite system to a smaller positive definite one called the Schur complement and then uses the Cholesky factorization to solve the system. However, this reduction produces a more ill-conditioned system which is a lot denser than the former [63]. As a result, methods involving the solution of the indefinite system are often preferable. This indefinite system takes the form

$$\underbrace{\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} b \\ d \end{bmatrix}}_{c}, \tag{1.1.1}$$

where $H \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$ are symmetric and $A \in \mathbb{R}^{m\times n}$, $m \leq n$, and A has full row rank.

The interior point methods considered in this thesis are presented in Chapter 2; the spectral properties of the resulting saddle point systems are also analyzed in that chapter. Equation (1.1.1) can be solved either directly or by

the use of an iterative method. We shall consider suitable direct methods in Chapter 3 and then describe possible iterative methods in Chapter 4.

In Section 2.4.2 we show that the coefficient matrix in (1.1.1) is indefinite. This appears to imply that the conjugate gradient method isn't a suitable method for solving such saddle point systems; however, with the use of *constraint preconditioners,* we can effectively apply this method. We shall derive the resulting methods in Chapter 5 and also give theoretical results about the associated convergence properties. For the cases of $C = 0$ and C symmetric and positive definite, preconditioned conjugate gradient-based methods (often called projected preconditioned conjugate gradient methods) are already known, but we reveal that there is a projected preconditioned conjugate gradient method which can be used when C is symmetric and positive semidefinite; when $C = 0$ or C is symmetric and positive definite, this new method encompasses the two previously known methods.

The standard method for using constraint preconditioners, at least in the optimization community, has been to choose the constraint preconditioner and then factor it. There is no reason this should be any more efficient than solving (1.1.1) directly. In fact, in Chapters 6 and 7 we will observe that even the simplest choices can be impractical to use. However, we propose the alternative of *implicit factorization constraint preconditioners.* In Chapter 6 we introduce the idea of implicit factorization preconditioners for the case $C = 0$. In particular, we consider the Schilders factorization. We then extend this to the case $C \neq 0$ in Chapter 7. Numerical results are given in both chapters to indicate how these preconditioners would behave in one iteration of an interior point method.

To be able to use implicit factorization preconditioners we make a fundamental assumption about the structure of the saddle point system (1.1.1): the first m columns of A are linearly independent. Although this is not generally true, we can always carry out a symmetric permutation of the required saddle point problem such that this assumption will hold. We will analyze different choices of permutation in Chapter 8. Numerical examples involving the use of interior point methods to solve test constrained optimization problems will be presented.

We will have so far concentrated on the use of constraint preconditioners within a conjugate gradient method. Several other preconditioners have been suggested for solving saddle point systems that arise through the discretization

of specific classes of partial differential equations; these preconditioners require the use of a more general iterative method, for example, the Bi-conjugate gradient method. We shall compare some of these preconditioners with the implicit factorization constraint preconditioners in Chapter 9.

Finally, we will draw some conclusions and describe several extensions to our ideas, some of which will connect our work to different application areas.

1.2 Definition of Optimization Problems

Optimization is the minimization or maximization of a function subject to a set of constraints on its variables. Throughout this thesis we shall let x be a vector of variables, f be an objective function, and c be a vector of constraints that the variables must satisfy. The function f is a function of x that returns a real value that we wish to minimize or maximize. Since

$$\text{maximum } f(x) = -\text{ minimum } (-f(x))$$

there is no loss in generality in restricting our optimization problem to being that of minimization.

The generalized form of an optimization problem is

$$\begin{array}{lll} \min\limits_{x\in\mathbb{R}^n} & f(x) & \\ \text{subject to} & c_i(x) = 0, & i = 1, 2, \ldots, m; \\ & c_i(x) \geq 0, & i = m+1, \ldots, m', \end{array} \tag{1.2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^{m'}$. We call f the *objective* function, while c_i, $i = 1, \ldots, m$, are the *equality* constraints and c_i, $i = m+1, \ldots, m'$ are the *inequality* constraints. We define the *feasible set* $\mathcal{C}$ to be the set of points x that satisfy the constraints:

$$\mathcal{C} = \{x \mid c_i(x) = 0,\ i = 1, 2, \ldots, m;\ c_i(x) \geq 0,\ i = m+1, \ldots, m'\}, \tag{1.2.2}$$

so that we can write (1.2.1) more compactly as

$$\min_{x\in\mathcal{C}} f(x). \tag{1.2.3}$$

Any point x that lies in $\mathcal{C}$ is said to be *feasible* and any $x \notin \mathcal{C}$ is said to be *infeasible*.

There are a number of important subclasses of optimization problems. If $m' = m = 0$ then we have an *unconstrained minimization* problem — this

is the simplest of all the subclasses. The problems where $0 < m = m'$ are called *equality constrained minimization* problems and if $0 = m < m'$ then we have an *inequality constrained minimization* problem. Finally, if $0 < m < m'$ then the problem is known as a *mixed constraints minimization* problem. For consistency we shall assume that $m \leq n$.

1.3 Important Notation

Suppose that $f(x)$ is at least twice continuously differentiable ($f \in C^2$).

Definition 1.3.1. The *gradient*, $g(x)$, of the objective function f is defined to be

$$g(x) = \nabla_x f(x),$$

where $\nabla_x f(x)$ denotes the vector of first partial derivatives, whose i-th component is $\partial f(x)/\partial x_i$.

Definition 1.3.2. The *Hessian*, $H(x)$, of the objective function f is

$$H(x) = \nabla_{xx} f(x),$$

where the i,j-th component is the second partial derivative $\partial^2 f(x)/\partial x_i \partial x_j$. (Note: $H(x)$ is always symmetric.)

Likewise, the gradient and Hessian of the i-th constraint are respectively defined as $a_i(x) = \nabla_x c_i(x)$ and $H_i(x) = \nabla_{xx} c_i(x)$.

Definition 1.3.3. The *Jacobian* matrix is

$$A(x) = (\nabla_x c(x))^T = \begin{pmatrix} a_1^T(x) \\ \vdots \\ a_{m'}^T(x) \end{pmatrix}.$$

We shall use the usual Euclidean inner product between two t-vectors u and v, i.e. $\langle u, v \rangle = \sum_{i=1}^{t} u_i v_i$.

Definition 1.3.4. If y is a vector (of so-called *Lagrange multipliers*), the *Lagrangian function* $\mathcal{L}(x, y)$ is defined as

$$\mathcal{L}(x, y) = f(x) - \langle y, c(x) \rangle .$$

Its gradient and Hessian with respect to x are

$$\begin{aligned} g(x,y) &= \nabla_x \mathcal{L}(x,y) \equiv g(x) - A^T(x)y, \\ H(x,y) &= \nabla_{xx} \mathcal{L}(x,y) \equiv H(x) - \textstyle\sum_{i=1}^{m} y_i H_i(x). \end{aligned}$$

The i-th unit (canonical) vector is denoted by e_i, whilst e is the vector of ones and I is the identity matrix of appropriate dimension [41]. Given a symmetric matrix M with, respectively, m_+, m_- and m_0 positive, negative and zero eigenvalues, we denote its inertia by $\text{In}(M) = (m_+, m_-, m_0)$.

We will use the following standard order notation throughout; see [19] for more details.

Definition 1.3.5 (Order Notation)**.** Let ϕ be a function of a positive variable h, and let p be fixed.

- If there exists a constant $\kappa_u > 0$ such that $|\phi| \leq \kappa_u h^p$ for all sufficiently small h, then we write $\phi = O(h^p)$. If $p = 0$, we write $\phi = O(1)$ and say that ϕ is finite.
- If there exists a constant $\kappa_l > 0$ such that $|\phi| \geq \kappa_l h^p$ for all sufficiently small h, then we write $\phi = \Omega(h^p)$. If $p = 0$, we write $\phi = \Omega(1)$ and say that ϕ is bounded away from zero.
- If there exist constants $\kappa_l > 0$ and $\kappa_u > 0$ such that $\kappa_l h^p \leq |\phi| \leq \kappa_u h^p$ for all sufficiently small h, then we write $\phi = \Theta(h^p)$. If $p = 0$, we write $\phi = \Theta(1)$ and say that ϕ is both finite and bounded away from zero.

1.4 Optimality Conditions

Let us firstly define what global and local minimizers are with respect to our general optimization problem (1.2.1).

Definition 1.4.1. A point x^* is called a *global minimizer* if $x^* \in \mathcal{C}$ and $f(x^*) \leq f(x)$ for all $x \in \mathcal{C}$.

Definition 1.4.2. A point x^* is called a *local minimizer* if $x^* \in \mathcal{C}$ and there is an open neighbourhood $\mathcal{N}$ of x^* such that $f(x^*) \leq f(x)$ for all $x \in \mathcal{C} \cap \mathcal{N}$. We say that $x^* \in \mathcal{C}$ is *isolated* if there is an open neighbourhood $\mathcal{N}$ of x^* such that $f(x^*) < f(x)$ for all $x \neq x^* \in \mathcal{C} \cap \mathcal{N}$.

It can be extremely difficult to say anything about the solutions of optimization problems because, in general, they may have many local, often non-global, minimizers. We therefore need some optimality conditions; these conditions

provide a means of guaranteeing that a candidate solution is indeed (locally) optimal (the sufficient conditions) and they can also guide us in the design of algorithms for solving such problems.

We will consider the unconstrained, equality constrained, inequality constrained and mixed constraint problems separately.

1.4.1 Optimality Conditions for Unconstrained Optimization

The *necessary conditions* for optimality are derived by assuming that a point x^* is a local minimizer and then proving facts about $g(x^*)$ and $H(x^*)$. We shall not show the proofs to these theorems — proofs can be found in [45] and [62].

Theorem 1.4.3 (First-Order Necessary Conditions)**.** *If x^* is a local minimizer of $f(x)$ and f is continuously differentiable in an open neighbourhood of x^*, then $g(x^*) = 0$.*

Theorem 1.4.4 (Second-Order Necessary Conditions)**.** *If x^* is a local minimizer of $f(x)$ and $\nabla^2 f$ is continuous in an open neighbourhood of x^*, then $g(x^*) = 0$ and $H(x^*)$ is positive semidefinite.*

Suppose that we have found a point that satisfies the above conditions. We can guarantee that it is a minimizer, an isolated one, provided the following second-order sufficient optimality conditions are satisfied.

Theorem 1.4.5 (Second-Order Sufficient Conditions)**.** *Suppose $\nabla^2 f$ is continuous in an open neighbourhood of x^* and that $g(x^*) = 0$ and $H(x^*)$ is positive definite. Then x^* is an isolated local minimizer of $f(x)$.*

1.4.2 Optimality Conditions For Equality Constrained Minimization

When constraints become involved in the problem we must take the role of the feasibility region into account. Given equality constraints, the necessary optimality conditions are as follows.

Theorem 1.4.6 (First-Order Necessary Conditions)**.** *If x^* is a local minimizer of $f(x)$ subject to $c(x) = 0$ and f and c are continuously differentiable in open neighbourhoods of x^*, then, so long as the set of constraint gradients $\{\nabla c_i(x^*),\ i = 1, \ldots, m\}$ is linearly independent, there exists a vector of Lagrange multipliers y^* such that*

$$\begin{aligned} c(x^*) &= 0 \quad (\textit{primal feasibility}), \\ g(x^*) - A^T(x^*)y^* &= 0 \quad (\textit{dual feasibility}). \end{aligned}$$

Theorem 1.4.7 (Second-Order Necessary Conditions)**.** *If x^* is a local minimizer of $f(x)$ subject to $c(x)$ and $\nabla^2 f, \nabla^2 c$ are continuous in open neighbourhoods of x^*, then, so long as the set of constraint gradients $\{\nabla c_i(x^*),\ i = 1, \ldots, m\}$ is linearly independent, there exists a vector of Lagrange multipliers y^* such that $c(x^*) = 0$, $g(x^*) - A^T(x^*)y^* = 0$ and*

$$\langle s, H(x^*, y^*)s\rangle \geq 0 \textit{ for all } s \in \mathcal{N},$$

where

$$\mathcal{N} = \{s \in \mathbb{R}^n | A(x^*)s = 0\}.$$

Strengthening the requirement on $H(x^*, y^*)$ gives us the sufficient conditions:

Theorem 1.4.8 (Second-Order Sufficient Conditions)**.** *Suppose $\nabla^2 f$ and $\nabla^2 c$ are continuous in open neighbourhoods of x^* and that there exists a vector of Lagrange multipliers y^* such that $c(x^*) = 0$, $g(x^*) - A^T(x^*)y^* = 0$ and*

$$\langle s, H(x^*, y^*)s\rangle > 0 \textit{ for all } s \in \mathcal{N},$$

where

$$\mathcal{N} = \{s \in \mathbb{R}^n | A(x^*)s = 0\}.$$

Then x^* is an isolated local minimizer of $f(x)$ subject to $c(x) = 0$.

1.4.3 Optimality Conditions For Inequality Constrained Minimization

Suppose that x^* is a minimizer of the inequality constrained minimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \text{ subject to } c(x) \geq 0.$$

We shall say that a constraint is *active* at x^* if and only if $c_i(x^*) = 0$. The active constraints play an important role in defining the minimizer, but the inactive constraints play no part at all. We'll denote the set of active constraints for a given x^* as $\mathcal{A}(x^*)$ and call it the *active set*.

Definition 1.4.9 (LICQ)**.** Given the point x^* and the active set $\mathcal{A}(x^*)$, we say that the linear independence constraint qualification (LICQ) holds if the set of active constraint gradients $\{\nabla c_i(x^*),\ i = 1, \ldots, m'\}$ is linearly independent.

Theorem 1.4.10 (First-Order Necessary Conditions)**.** *If x^* is a local minimizer of $f(x)$ subject to $c(x) \geq 0$ and f and c are continuously differentiable in open neighbourhoods of x^*, then, so long as the LICQ holds at x^*, there exists a vector of Lagrange multipliers y^* such that*

$$\begin{aligned} c(x^*) &\geq 0 \quad (\textit{primal feasibility}), \\ g(x^*) - A^T(x^*)y^* &= 0 \quad (\textit{dual feasibility}), \\ c_i(x^*)[y^*]_i &= 0 \quad (\textit{complementary slackness}). \end{aligned}$$

The primal/dual feasibility and complementary slackness conditions are known collectively as the Karush-Kuhn-Tucker (KKT) conditions.

Theorem 1.4.11 (Second-Order Necessary Conditions)**.** *If x^* is a local minimizer of $f(x)$ subject to $c(x) \geq 0$ and $\nabla^2 f$ and $\nabla^2 c$ are continuous in open neighbourhoods of x^*, then, so long as the LICQ holds at x^*, there exists a vector of Lagrange multipliers y^* such that primal/dual feasibility and complementary slackness hold as well as*

$$\langle s, H(x^*, y^*)s \rangle \geq 0 \textit{ for all } s \in \mathcal{N}_+,$$

where

$$\mathcal{N}_+ = \left\{ s \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \langle s, a_i(x^*) \rangle = 0 \textit{ if } c_i(x^*) = 0 \ \&\ [y^*]_i > 0 \textit{ and} \\ \langle s, a_i(x^*) \rangle \geq 0 \textit{ if } c_i(x^*) = 0 \ \&\ [y^*]_i = 0 \end{array} \right\}.$$

The corresponding sufficient condition also holds:

Theorem 1.4.12 (Second-Order Sufficient Conditions)**.** *Suppose $\nabla^2 f$ and $\nabla^2 c$ are continuous in open neighbourhoods of x^* and that there exists a vector of Lagrange multipliers y^* such that $c(x^*) = 0$, $g(x^*) - A^T(x^*)y^* = 0$ and*

$$\langle s, H(x^*, y^*)s \rangle > 0 \textit{ for all } s \in \mathcal{N}_+,$$

where

$$\mathcal{N}_+ = \left\{ s \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \langle s, a_i(x^*)\rangle = 0 \text{ if } c_i(x^*) = 0 \;\&\; [y^*]_i > 0 \text{ and} \\ \langle s, a_i(x^*)\rangle \geq 0 \text{ if } c_i(x^*) = 0 \;\&\; [y^*]_i = 0 \end{array} \right\},$$

then x^* is an isolated local minimizer of $f(x)$ subject to $c(x) \geq 0$.

1.4.4 Optimality Conditions For Mixed Constraints Minimization

Suppose that x^* is a minimizer of the inequality-constrained minimization problem

$$\begin{array}{rl} \min\limits_{x \in \mathbb{R}^n} & f(x) \\ \text{subject to} & c_i(x) = 0, \quad i = 1, 2, \ldots, m; \\ & c_i(x) \geq 0, \quad i = m+1, \ldots, m'. \end{array}$$

The necessary optimality conditions are as follows.

Theorem 1.4.13 (First-Order Necessary Conditions)**.** *If x^* is a local minimizer of $f(x)$ subject to $c_i(x) = 0$, $i = 1, 2, \ldots, m$ and $c_i(x) \geq 0$, $i = m+1, \ldots, m'$, and f and c are continuously differentiable in open neighbourhoods of x^*, then, so long as the LICQ holds at x^*, there exists a vector of Lagrange multipliers y^* such that*

$$\begin{array}{rcll} c_i(x^*) & = & 0 \quad i = 1, \ldots, m & (\textit{primal feasibility} \;\; \text{(a)}), \\ c_i(x^*) & \geq & 0 \quad i = m+1, \ldots, m' & (\textit{primal feasibility} \;\; \text{(b)}), \\ g(x^*) - A^T(x^*)y^* & = & 0 & (\textit{dual feasibility}), \\ c_i(x^*)[y^*]_i & = & 0 & (\textit{complementary slackness}). \end{array}$$

Theorem 1.4.14 (Second-Order Necessary Conditions)**.** *If x^* is a local minimizer of $f(x)$ subject to $c_i(x) = 0$, $i = 1, 2, \ldots, m$ and $c_i(x) \geq 0$, $i = m+1, \ldots, m'$, and $\nabla^2 f$ and $\nabla^2 c$ are continuous in open neighbourhoods of x^*, then, so long as the LICQ holds at x^*, there exists a vector of Lagrange multipliers y^* such that primal/dual feasibility and complementary slackness hold as well as*

$$\langle s, H(x^*, y^*)s\rangle \geq 0 \text{ for all } s \in \mathcal{N}_+,$$

where

$$\mathcal{N}_+ = \left\{ s \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \langle s, a_i(x^*)\rangle = 0 \text{ if } c_i(x^*) = 0 \;\&\; [y^*]_i > 0 \text{ and} \\ \langle s, a_i(x^*)\rangle \geq 0 \text{ if } c_i(x^*) = 0 \;\&\; [y^*]_i = 0 \end{array} \right\}.$$

The corresponding sufficient condition also holds:

Theorem 1.4.15 (Second-Order Sufficient Conditions)**.** *Suppose* $\nabla^2 f$ *and* $\nabla^2 c$ *are continuous in open neighbourhoods of* x^* *and that there exists a vector of Lagrange multipliers* y^* *such that* $c_i(x) = 0$, $i = 1, 2, \ldots, m$ *and* $c_i(x) \geq 0$, $i = m+1, \ldots, m'$, $g(x^*) - A^T(x^*)y^* = 0$ *and*

$$\langle s, H(x^*, y^*)s \rangle > 0 \text{ for all } s \in \mathcal{N}_+,$$

where

$$\mathcal{N}_+ = \left\{ s \in \mathbb{R}^n \;\middle|\; \begin{array}{l} \langle s, a_i(x^*) \rangle = 0 \text{ if } c_i(x^*) = 0 \;\&\; [y^*]_i > 0 \text{ and} \\ \langle s, a_i(x^*) \rangle \geq 0 \text{ if } c_i(x^*) = 0 \;\&\; [y^*]_i = 0 \end{array} \right\},$$

then x^* *is an isolated local minimizer of* $f(x)$ *subject to* $c(x) \geq 0$.

We shall make use of some of these optimality conditions in Chapter 2 when we derive some of the interior point methods.

Chapter 2

Interior Point Methods

From this chapter onwards we shall restrict ourselves to constrained optimization problems. We shall also simplify the equality constraints to be linear constraints which can consequently be written as

$$Ax - d = 0, \tag{2.0.1}$$

where $A \in \mathbb{R}^{m \times n}$ is of full rank, $d \in \mathbb{R}^m$, and $m \leq n$. For the treatment of more general problems refer to [39, 62, 87].

In the 1980s there was a large leap made in the solving of constrained optimization problems. The discovery originally came about by considering linear objective functions, i.e. $f(x) = b^T x$ for some vector b, and formulating them as nonlinear problems. These are then solved with various modifications of nonlinear algorithms such as Newton's method [54, 62]. In these methods all of the iterates are required to satisfy the inequality constraints strictly, so they soon became known as interior point methods. This idea has since been generalized to other forms of objective function.

2.1 Equality Constrained Problems

Suppose we wish to solve an equality constrained minimization problem of the form

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} x^T Q x + b^T x \text{ subject to } Ax - d = 0,$$

where $A \in \mathbb{R}^{m \times n}$ is of full rank, $Q \in \mathbb{R}^{n \times n}$ is symmetric, positive semidefinite, $b \in \mathbb{R}^n$ and $d \in \mathbb{R}^m$.

If x^* is a local minimizer of this problem, then the second-order necessary conditions (Section 1.4.2) imply that

$$\begin{aligned} Ax^* - d &= 0, && (2.1.1)\\ Qx^* + b - A^T y^* &= 0. && (2.1.2) \end{aligned}$$

These equations can be expressed as

$$\begin{bmatrix} Q & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ -y^* \end{bmatrix} = \begin{bmatrix} -b \\ d \end{bmatrix}. \tag{2.1.3}$$

The system (2.1.3) is known as the Karush-Kuhn-Tucker (KKT) system and is an example of a saddle point problem. We need only solve this system to find x^*; no interior point method is required.

2.2 Inequality Constrained Problems

Consider the convex nonlinear optimization problem

$$\min f(x) \;\text{ such that }\; c(x) \geq 0, \tag{2.2.1}$$

where $x \in \mathbb{R}^n$, and $f : \mathbb{R}^n \mapsto \mathbb{R}$ and $c : \mathbb{R}^n \mapsto \mathbb{R}^{m'}$ are convex and twice differentiable. Interior point methods introduce slack variables, s, into (2.2.1) to transform the problem into

$$\min f(x) \;\text{ such that }\; c(x) - s = 0 \;\text{ and }\; s \geq 0. \tag{2.2.2}$$

We then replace the non-negativity constraints of (2.2.2) with a logarithmic barrier term in the objective function, resulting in

$$\min f(x) - \mu \sum_{i=1}^{m'} \ln s_i \;\text{ such that }\; c(x) - s = 0. \tag{2.2.3}$$

The first-order optimality conditions for this problem are

$$\begin{aligned} g(x) - A(x)^T y &= 0, \\ y - \mu S^{-1} e &= 0, \\ c(x) - s &= 0, \\ s \geq 0, \qquad & y \geq 0, \end{aligned} \tag{2.2.4}$$

where $S = \texttt{diag}\{s_1, s_2, \ldots, s_{m'}\}$, y is a vector of Lagrange multipliers, and e is as defined in Section 1.3. Primal-dual methods modify (2.2.4) by multiplying the second equation by S, resulting in the system

$$\begin{aligned} g(x) - A(x)^T y &= 0, \\ SYe - \mu e &= 0, \\ c(x) - s &= 0, \\ s \geq 0, \qquad & y \geq 0, \end{aligned} \tag{2.2.5}$$

where $Y = \texttt{diag}\{y_1, y_2, \ldots, y_{m'}\}$. The following system is solved to find the Newton direction:

$$\begin{bmatrix} H(x,y) & 0 & -A^T(x) \\ 0 & Y & S \\ -A(x) & I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta s \\ \Delta y \end{bmatrix} = \begin{bmatrix} -g(x) + A^T(x)y \\ \mu e - SYe \\ c(x) - s \end{bmatrix}. \tag{2.2.6}$$

By eliminating Δs from (2.2.6) we obtain the saddle point system

$$\underbrace{\begin{bmatrix} H(x,y) & -A(x)^T \\ -A(x) & -SY^{-1} \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -g(x) + A(x)^T y \\ c(x) - \mu Y^{-1} e \end{bmatrix}, \tag{2.2.7}$$

with

$$\Delta s = -s + Y^{-1}(\mu e - S\Delta y).$$

This gives us Algorithm 2.2.1: an interior point method for solving inequality constrained optimization problems of the form given in (2.2.1). The positive barrier parameter, μ, is gradually reduced to guarantee the convergence to the optimal solution of the original problem (2.2.1). In practice, this parameter is generally reduced to be of the order 10^{-8}.

At an optimal point, x^*, $s_i^* y_i^* = 0$ for $i = 1, \ldots, m'$. As we draw near to optimality we find that at the k^{th} iterate of the interior point method when $i \in \mathcal{A}(x)$, where $\mathcal{A}(x^*)$ is the active set defined in Section 1.4.3, $[s^k]_i/[y^k]_i = \mathcal{O}(\mu_k)$; for $i \notin \mathcal{A}(x^*)$, $[s^k]_i/[y^k]_i = \mathcal{O}(\mu_k^{-1})$. The $\mathcal{O}(\mu_k^{-1})$ behaviour of some of the entries in SY^{-1} will result in $\mathcal{H}$ being very ill-conditioned when we draw close to optimality. One way that the optimization community combats this is to split the matrix SY^{-1} such that

$$SY^{-1} = \begin{bmatrix} S_{\mathcal{A}} Y_{\mathcal{A}}^{-1} & 0 \\ 0 & S_{\mathcal{I}} Y_{\mathcal{I}}^{-1} \end{bmatrix},$$

Algorithm 2.2.1 Interior Point Method for Inequality Constrained Problems

Require: x^0 and $(s^0, y^0) > 0$

for $k = 0, 1, 2, \ldots$ **do**

Choose $\sigma^k \in (0,1)$ and set $\mu^k = \sigma^k \left(\frac{\left[s^k\right]^T y^k}{m} \right)$

Solve the following linear system

$$\underbrace{\begin{bmatrix} H(x^k, y^k) & -A(x^k)^T \\ -A(x^k) & -[S^k][Y^k]^{-1} \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} \Delta x^k \\ \Delta y^k \end{bmatrix} = \begin{bmatrix} -g(x^k) + A(x^k)^T y^k \\ c(x^k) - \mu^k [Y^k]^{-1} e \end{bmatrix}$$

Set $\Delta s^k = -s^k + [Y^k]^{-1}(\mu^k e - [S^k]\Delta y^k)$

Choose $\chi^k \in (0,1)$

Choose α^k as the first element in the sequence $\left\{1, [\chi^k], [\chi^k]^2, [\chi^k]^3, \ldots\right\}$ such that $(s^k + \alpha^k \Delta s^k, y^k + \alpha^k \Delta y^k) > 0$

Form the new iterate

$$(s^{k+1}, x^{k+1}, y^{k+1}) = (s^k, x^k, y^k) + \alpha^k (\Delta s^k, \Delta x^k, \Delta y^k)$$

end for

where $S_{\mathcal{A}}[Y_{\mathcal{A}}]_k^{-1} = \mathcal{O}(\mu_k I)$ and $S_{\mathcal{I}}[Y_{\mathcal{I}}]_k^{-1} = \mathcal{O}(\frac{1}{\mu_k} I)$. System (2.2.7) becomes

$$\begin{bmatrix} H(x,y) & -A_{\mathcal{A}}(x)^T & -A_{\mathcal{I}}(x)^T \\ -A_{\mathcal{A}}(x) & -S_{\mathcal{A}} Y_{\mathcal{A}}^{-1} & 0 \\ -A_{\mathcal{I}}(x) & 0 & -S_{\mathcal{I}} Y_{\mathcal{I}}^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y_{\mathcal{A}} \\ \Delta y_{\mathcal{I}} \end{bmatrix} = \begin{bmatrix} -g(x) + A_{\mathcal{A}}(x)^T y_{\mathcal{A}} + A_{\mathcal{I}}(x)^T y_{\mathcal{I}} \\ c_{\mathcal{A}}(x) - \mu Y_{\mathcal{A}}^{-1} e \\ c_{\mathcal{I}}(x) - \mu Y_{\mathcal{I}}^{-1} e \end{bmatrix},$$

where $A(x)^T = \begin{bmatrix} A_{\mathcal{A}}(x)^T & A_{\mathcal{I}}(x)^T \end{bmatrix}$. Eliminating $\Delta y_{\mathcal{I}}$ we obtain

$$\begin{bmatrix} H + A_{\mathcal{I}}^T S_{\mathcal{I}}^{-1} Y_{\mathcal{I}} A_{\mathcal{I}} & -A_{\mathcal{A}}^T \\ -A_{\mathcal{A}} & -S_{\mathcal{A}} Y_{\mathcal{A}}^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y_{\mathcal{A}} \end{bmatrix} = \begin{bmatrix} -g + A_{\mathcal{A}}^T y_{\mathcal{A}} + \mu A_{\mathcal{I}}^T S_{\mathcal{I}}^{-1} e \\ c_{\mathcal{A}} - \mu Y_{\mathcal{A}}^{-1} e \end{bmatrix}.$$

As we approach the optimal solution the entries of $S_{\mathcal{A}}(x) Y_{\mathcal{A}}^{-1}$ become small, as do the entries of $A_{\mathcal{I}}^T S_{\mathcal{I}}^{-1}(x) Y_{\mathcal{I}} A_{\mathcal{I}}$; the system is well behaved in the limit [45].

2.2.1 The Central Path

The above method can also be considered in terms of the central path. Suppose that we consider the inequality problem as defined in (2.2.2):

$$\min f(x) \;\text{ such that }\; c(x) - s = 0 \;\text{ and }\; s \geq 0.$$

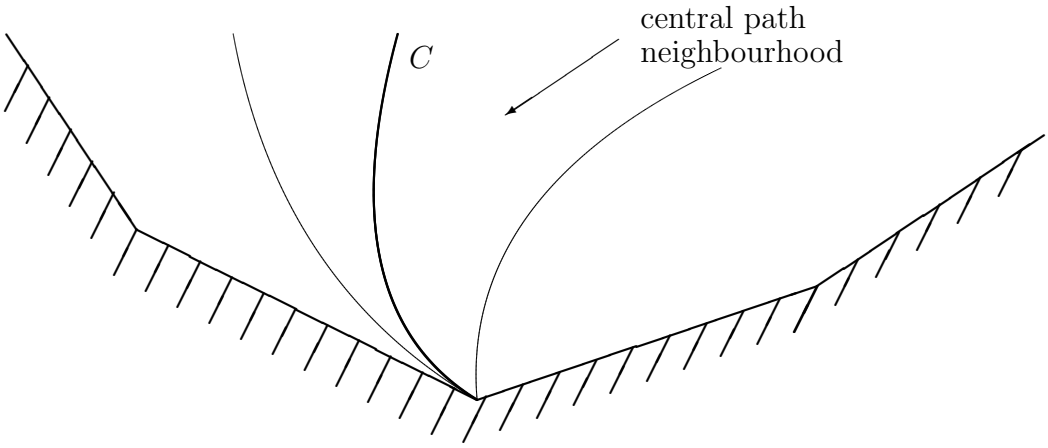

Figure 2.1: Central path, projected into space of primal variables x, showing a typical neighbourhood.

Theorem 1.4.13 gives the following first-order necessary conditions:

$$\begin{aligned} c(x) - s &= 0, \\ g(x) - A^T(x)y &= 0, \\ s_i y_i &= 0, \quad i = 1, \ldots, m', \\ (s, y) &\geq 0. \end{aligned}$$

The central path C is parameterized by a scalar $\tau > 0$ such that each point $(x_\tau, y_\tau, s_\tau) \in C$ solves the system

$$\begin{aligned} c(x) - s &= 0, \\ g(x) - A^T(x)y &= 0, \\ s_i y_i &= \tau, \quad i = 1, \ldots, m', \\ (s, y) &> 0. \end{aligned}$$

We note that these conditions differ from the first-order necessary conditions only in the term τ on the right hand side and the inequality sign becoming a strict inequality. The central path is defined as

$$C = \{(x_\tau, y_\tau, s_\tau)\}.$$

In Figure 2.1 we see a plot of C for a typical problem, projected into the space of primal variables x.

Instead of taking Newton steps directly onto C, interior point methods take Newton steps towards points on C for which $\tau > 0$. To achieve this, a centering parameter $\sigma \in (0, 1)$ and a duality measure $\widetilde{\mu}$ defined by

$$\widetilde{\mu} = \frac{1}{m'} \sum_{i=1}^{m'} y_i s_i = \frac{y^T s}{m'}$$

are introduced. We then solve the system

$$\begin{aligned} c(x) - s &= 0, \\ g(x) - A^T(x)y &= 0, \\ SYe &= \sigma\widetilde{\mu}e, \\ (s, y) &> 0. \end{aligned}$$

Note that by setting $\mu = \sigma\widetilde{\mu}$ we exactly obtain (2.2.5). Thus, Algorithm 2.2.1 is equivalently given by this central path idea.

2.3 Mixed Constraints Optimization Problems

Let us consider how to solve constrained optimization problems of the following form:

$$\begin{array}{ll} \min\limits_{x} & f(x) \\ \text{subject to} & Ax - d = 0 \text{ and } x \geq 0, \end{array} \tag{2.3.1}$$

where $f(x)$ has continuous first and second derivatives, $x \in \mathbb{R}^n$, $d \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$ has full row rank m. As in the previous section, interior point methods usually replace the inequality constraints with logarithmic barriers to obtain

$$\begin{array}{ll} \min\limits_{x} & f(x) - \mu \sum_{j=1}^{n} \ln x_j \\ \text{subject to} & Ax - d = 0, \end{array} \tag{2.3.2}$$

where μ is a barrier parameter [34, 39, 62, 87]. Any finite solution of (2.3.2) is a stationary point of the Lagrangian function

$$\mathcal{L}(x, y, \mu) = f(x) - y^T(Ax - d) - \mu \sum_{j=1}^{n} \ln x_j.$$

For the conditions of the stationary point we write

$$\nabla_x \mathcal{L}(x, y, \mu) = \nabla_x f(x) - A^T y - \mu X^{-1} e = 0, \tag{2.3.3}$$

$$\nabla_y \mathcal{L}(x, y, \mu) = Ax - d = 0, \tag{2.3.4}$$

where $X^{-1} = \texttt{diag}\{x_1^{-1}, x_2^{-1}, \ldots, x_n^{-1}\}$.

Let $s = \mu X^{-1} e$, i.e. $XSe = \mu e$, where $S = \texttt{diag}\{s_1, s_2, \ldots, s_n\}$. The first-order optimality conditions for the barrier problem are therefore given by

$$\begin{aligned} Ax &= d, && (2.3.5)\\ \nabla_x f(x) - A^T y &= s, && (2.3.6)\\ XSe &= \mu e. && (2.3.7) \end{aligned}$$

The interior point algorithm applies the Newton method to solve this system of nonlinear equations. We obtain the Newton direction by solving the system of linear equations:

$$\begin{bmatrix} A & 0 & 0 \\ \nabla_{xx} f(x) & A^T & -I \\ -S & 0 & -X \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} \xi_p \\ \xi_d \\ \xi_\mu \end{bmatrix}, \qquad (2.3.8)$$

where

$$\begin{aligned} \xi_p &= d - Ax, \\ \xi_d &= s - \nabla_x f + A^T y, \\ \xi_\mu &= XSe - \mu e. \end{aligned}$$

This gives us Algorithm 2.3.1 which is an interior point method for solving mixed constraints optimization problems of the form given in (2.3.1).

By eliminating Δs from (2.3.8) and rearranging we obtain the symmetric indefinite system of linear equations

$$\begin{bmatrix} \nabla_{xx} f(x) + SX^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \end{bmatrix} = \begin{bmatrix} \xi_d - X^{-1}\xi_\mu \\ \xi_p \end{bmatrix}. \qquad (2.3.9)$$

The system (2.3.9) is a KKT/saddle point problem. Once this system has been solved a positive step length α is chosen such that $(x+\alpha\Delta x, y+\alpha\Delta y) \geq 0$, and the variables are updated according to $x \leftarrow x + \alpha\Delta x$, $y \leftarrow y + \alpha\Delta y$ and $\mu \leftarrow \tau\mu$, where $\tau \in (0, 1)$. (Sometimes a different α may be used for x and y.)

2.3.1 The Quadratic Programming Problem

If the function $f(x)$ is a quadratic function, then the associated optimization problem is known as a *quadratic programming problem*. Suppose f is expressed as

$$f(x) = \frac{1}{2} x^T Q x + b^T x,$$

Algorithm 2.3.1 Interior Point Method for Mixed Constraint Problems

Require: y^0 and $(x^0, s^0) > 0$
for $k = 0, 1, 2, \ldots$ **do**
Choose $\sigma^k \in (0, 1)$ and set $\mu^k = \sigma^k \left(\frac{[x^k]^T [s^k]}{n} \right)$
Solve the following linear system

$$\begin{bmatrix} A & 0 & 0 \\ \nabla_{xx} f(x^k) & A^T & -I \\ -[S^k] & 0 & -[X^k] \end{bmatrix} \begin{bmatrix} \Delta x^k \\ -\Delta y^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} d - Ax^k \\ s^k - \nabla_x f(x^k) + A^T y^k \\ -\mu e + [X^k][S^k]e \end{bmatrix}$$

Choose $\chi^k \in (0, 1)$
Choose α^k as the first element in the sequence $\{1, [\chi^k], [\chi^k]^2, [\chi^k]^3, \ldots\}$ such that $(s^k + \alpha^k \Delta s^k, x^k + \alpha^k \Delta x^k) > 0$
Form the new iterate

$$(s^{k+1}, x^{k+1}, y^{k+1}) = (s^k, x^k, y^k) + \alpha^k (\Delta s^k, \Delta x^k, \Delta y^k)$$

end for

where $Q \in \mathbb{R}^{n \times n}$ is a positive semi-definite matrix and $b \in \mathbb{R}^n$. The corresponding KKT/saddle point system is

$$\begin{bmatrix} Q + SX^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \end{bmatrix} = \begin{bmatrix} \xi_d - X^{-1} \xi_\mu \\ \xi_p \end{bmatrix}, \tag{2.3.10}$$

where

$$\begin{aligned} \xi_p &= d - Ax, \\ \xi_d &= s - b - Qx - A^T y, \\ \xi_\mu &= XSe - \mu e. \end{aligned}$$

2.3.2 Mehrotra's Predictor-Corrector Method

The interior point method introduced in Section 2.3 is no longer the most commonly used version in practice. Soon after its appearance, Mehrotra's predictor-corrector variant [57] became a very popular method. The main difference between the predictor-corrector approach and the previous algorithm is the choice of search direction. In the predictor-corrector method the new search directions are selected by the solution of two linear systems. Firstly we

solve

$$\begin{bmatrix} A & 0 & 0 \\ \nabla_{xx} f(x^k) & A^T & -I \\ -[S^k] & 0 & -[X^k] \end{bmatrix} \begin{bmatrix} \Delta\widetilde{x}^k \\ -\Delta\widetilde{y}^k \\ \Delta\widetilde{s}^k \end{bmatrix} = \begin{bmatrix} d - Ax^k \\ s^k - \nabla_x f(x^k) + A^T y^k \\ [X^k][S^k]e \end{bmatrix},$$

and $\Delta\widetilde{s}^k$, $\Delta\widetilde{x}^k$, $\Delta\widetilde{y}^k$ are called *affine directions.* The search directions are then given by

$$\begin{bmatrix} A & 0 & 0 \\ \nabla_{xx} f(x^k) & A^T & -I \\ -[S^k] & 0 & -[X^k] \end{bmatrix} \begin{bmatrix} \Delta x^k \\ -\Delta y^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} d - Ax^k \\ s^k - \nabla_x f(x^k) + A^T y^k \\ -\mu e + [X^k][S^k]e + [\Delta\widetilde{X}^k][\Delta\widetilde{S}^k]e \end{bmatrix}.$$

As in the original variant, the variables $\Delta\widetilde{s}^k$ and Δs^k are usually eliminated from these linear systems and saddle point problems are obtained.

This variant significantly reduces the number of iterations of the interior point method. Although this reduction is at the cost of us needing to solve two linear systems instead of one, we note that the matrix is the same in both the systems. It has been found in practice that the reduction in iterations easily compensates the requirement of solving these two systems, particularly because we need only factorize the coefficient matrix once.

2.4 General Structure of the Linear Systems

In the previous sections of this chapter we have shown the need to be to able to solve an important class of indefinite linear systems. These systems are of a saddle point structure

$$\underbrace{\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}, \tag{2.4.1}$$

where $H \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$ are symmetric and $A \in \mathbb{R}^{m\times n}$. We shall assume that $m \leq n$ and A is of full rank. Another reasonable assumption for us to make is that H and C are positive semi-definite. More shall be said about this assumption in Chapter 5.

2.4.1 Invertibility Conditions

If H is invertible, the saddle point matrix can be factored into block triangular factors:

$$\mathcal{H} = \begin{bmatrix} H & A^T \\ A & -C \end{bmatrix} = \begin{bmatrix} I & 0 \\ AH^{-1} & I \end{bmatrix} \begin{bmatrix} H & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & H^{-1}A^T \\ 0 & I \end{bmatrix}, \tag{2.4.2}$$

where $S = -(C + AH^{-1}A^T)$ is the Schur complement of H in $\mathcal{H}$. We are able to derive a number of important properties about the saddle point matrix from this factorization, as well as it forming the basis of many popular solution algorithms.

Assuming H is nonsingular, it is clear from (2.4.2) that $\mathcal{H}$ is nonsingular if and only if S is also nonsingular. Unfortunately this is a very general statement and it may be difficult to say much about the invertibility of the Schur complement S.

We shall firstly consider the case $C = 0$. When H is symmetric positive semidefinite, we have the following result, [8].

Theorem 2.4.1. *Assume that H is symmetric positive semidefinite, A has full rank, and $C = 0$. Then a necessary and sufficient condition for the saddle point matrix $\mathcal{H}$ to be nonsingular is that* $\ker(A) \cap \ker(H) = \{0\}$.

Proof. Let $u = \begin{bmatrix} x \\ y \end{bmatrix}$ be such that $\mathcal{H}u = 0$. Hence $Hx + A^Ty = 0$ and $Ax = 0$. It follows that $x^THx = -x^TA^Ty = -(Ax)^Ty = 0$. Since H is symmetric positive semidefinite, $x^THx = 0$ implies that $Hx = 0$, and therefore $x \in \ker(A) \cap \ker(H)$, thus $x = 0$. Also, $y = 0$ since $A^Ty = 0$ and A^T has full column rank. Therefore $u = 0$, and $\mathcal{H}$ is nonsingular. This proves the sufficiency of the condition.

Assume now that $\ker(A) \cap \ker(H) \neq \{0\}$. Taking $x \in \ker(A) \cap \ker(H)$, $x \neq 0$ and letting $u = \begin{bmatrix} x \\ 0 \end{bmatrix}$ we have $\mathcal{H}u = 0$, implying that $\mathcal{H}$ is singular. Hence, the condition is necessary. □

This is easily extended to the case of C symmetric positive semidefinite, [8].

Theorem 2.4.2. *Assume that H is symmetric positive semidefinite, A has full rank, and C is symmetric positive semidefinite (possibly zero). Then a necessary and sufficient condition for the saddle point matrix $\mathcal{H}$ to be nonsingular is that* $\ker(A) \cap \ker(H) = \{0\}$.

2.4.2 Spectral properties of saddle point matrices

As will be seen in later chapters, the spectral properties of saddle point matrices are relevant when solving the equations by direct and iterative methods. We shall assume that H is symmetric positive definite, A has full rank, and C is symmetric positive semidefinite (possibly zero). Then from (2.4.2) we obtain

$$\begin{bmatrix} I & 0 \\ -AH^{-1} & I \end{bmatrix} \begin{bmatrix} H & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} I & -H^{-1}A^T \\ 0 & I \end{bmatrix} = \begin{bmatrix} H & 0 \\ 0 & S \end{bmatrix}, \tag{2.4.3}$$

where $S = -(C + AH^{-1}A^T)$ is symmetric negative semidefinite. It therefore follows from the Sylvester Law of Inertia [41, p. 403] that $\mathcal{H}$ is indefinite, with n positive and m negative eigenvalues.

The following result from [71] establishes eigenvalue bounds for the case $C = 0$.

Theorem 2.4.3. *Assume H is symmetric positive definite, A has full rank, and $C = 0$. Let λ_1 and λ_n denote the largest and smallest eigenvalues of H respectively, and let σ_1 and σ_m denote the largest and smallest singular values of A respectively. Let $\lambda(\mathcal{H})$ denote the spectrum of $\mathcal{H}$. Then*

$$\lambda(\mathcal{H}) \subset I^- \cup I^+,$$

where

$$I^- = \left[\frac{1}{2}\left(\lambda_n - \sqrt{\lambda_n^2 + 4\sigma_1^2}\right), \frac{1}{2}\left(\lambda_1 - \sqrt{\lambda_1^2 + 4\sigma_m^2}\right)\right]$$

and

$$I^+ = \left[\lambda_n, \frac{1}{2}\left(\lambda_1 + \sqrt{\lambda_1^2 + 4\sigma_1^2}\right)\right].$$

This theorem can be extended to the case $C \neq 0$ [77]:

Theorem 2.4.4. *Assume H is symmetric positive definite and A has full rank. Let λ_1 and λ_n denote the largest and smallest eigenvalues of H respectively, and let σ_1 and σ_m denote the largest and smallest singular values of A respectively. Let $\lambda(\mathcal{H})$ denote the spectrum of $\mathcal{H}$. Then*

$$\lambda(\mathcal{H}) \subset I^- \cup I^+,$$

where

$$I^- = \left[\frac{1}{2}\left(\lambda_n - \|C\| - \sqrt{(\lambda_n + \|C\|)^2 + 4\sigma_1^2}\right), \frac{1}{2}\left(\lambda_1 - \sqrt{\lambda_1^2 + 4\sigma_m^2}\right)\right]$$

and

$$I^+ = \left[\lambda_n, \frac{1}{2}\left(\lambda_1 + \sqrt{\lambda_1^2 + 4\sigma_1^2}\right)\right].$$

2.4.2.1 Quadratic programming problem

As we saw in Section 2.3.1, in each iteration of the interior point method we are required to solve a system involving a saddle point matrix of the form

$$\begin{bmatrix} Q + SX^{-1} & A^T \\ A & 0 \end{bmatrix}.$$

At the optimal point $x_i^* s_i^* = 0$ for $i = 1, \ldots, n$. Let us define two index sets $\mathcal{A}$ and $\mathcal{I}$ as follows.

$$\mathcal{A} = \left\{ j \in \{1, 2, \ldots, n\} | s_j^* \neq 0 \right\}, \tag{2.4.4}$$

$$\mathcal{I} = \left\{ j \in \{1, 2, \ldots, n\} | x_j^* \neq 0 \right\}. \tag{2.4.5}$$

These two sets form a partition of the index set $\{1, 2, \ldots, n\}$. It is easy enough to prove that the two sets are disjoint: if there were an index j that belonged to both sets, then we would have $x_j^* s_j^* > 0$, contradicting the complementarity condition $x_i^* s_i^* = 0$ for $i = 1, \ldots, n$. Hence $\mathcal{A} \cap \mathcal{I} = \emptyset$.

The result $\mathcal{A} \cup \mathcal{I} = \{1, 2, \ldots, n\}$ is known as the Goldman-Tucker theorem [87]. Note, as for inequality constrained problems, the set $\mathcal{A}$ is often known as the active set (see Section 1.4.3).

For the interior point methods given in Section 2.3 we have $s_i^k / x_i^k = \mathcal{O}(\mu_k^{-1})$ for $i \in \mathcal{A}$, and $s_i^k / x_i^k = \mathcal{O}(\mu_k)$ for $i \in \mathcal{I}$ as we draw close to the optimal point. Therefore, for small values of μ, we are able to approximate $\mathcal{H}$ by

$$\widetilde{\mathcal{H}} = \begin{bmatrix} Q + \frac{1}{\mu}(\mathtt{diag}(s^*))^2 & A^T \\ A & 0 \end{bmatrix}.$$

We can conclude that the eigenvalues $\{\lambda_i\}$ of $Q + SX^{-1}$ can be approximated by

$$\begin{aligned} |\lambda_i| &\approx \frac{\varrho_i}{\mu}, \quad i = 1, \ldots, \hat{n}, \\ |\lambda_i| &\approx \varrho_i, \quad i = \hat{n} + 1, \ldots, n, \end{aligned}$$

where $\{\varrho_i\}$ are positive constants.

Using Theorem 2.4.3, an interval that the spectrum of $\mathcal{H}$, $\lambda(\mathcal{H})$, lies in can be approximated for small μ by

$$\lambda(\mathcal{H}) \subset I^- \cup I^+ \approx \widetilde{I}^- \cup \widetilde{I}^+,$$

where

$$\begin{aligned}\widetilde{I}^{-} &= \left[\frac{\varrho_n}{2\mu}\left(1-\sqrt{1+\left(\frac{2\mu\sigma_1}{\varrho_n}\right)^2}\right), \frac{\varrho_1}{2\mu}\left(1-\sqrt{1+\left(\frac{2\mu\sigma_m}{\varrho_1}\right)^2}\right)\right] \\ &\approx \left[-\alpha, -\frac{2\mu\sigma_m^2}{\varrho_1}\right],\end{aligned}$$

and

$$\begin{aligned}\widetilde{I}^{+} &= \left[\lambda_n, \frac{1}{2}\left(\lambda_1+\sqrt{\lambda_1^2+4\sigma_1^2}\right)\right] \\ &\approx \left[\varrho_n, \frac{\varrho_1}{\mu}\right],\end{aligned}$$

where α is a positive constant. As μ grows smaller, both the intervals $\widetilde{I}^{-}$ and $\widetilde{I}^{+}$ grow in length but one end of each of the intervals remains fixed. It is therefore theoretically possible that the eigenvalues of $\mathcal{H}$ remain the same as μ approaches zero, however, we can use Gerschgorin's Theorem to prove that $\hat{n}$ of the eigenvalues grow proportionally to $\frac{1}{\mu}$.

Theorem 2.4.5 (Gerschgorin's Theorem)**.** *Every eigenvalue of a matrix A lies in at least one of the circular discs with centers a_{ii} and radii $\sum_{j\neq i}|a_{ij}|$. If k of the circular discs form a connected domain which is isolated from the other discs, then there are precisely k eigenvalues within this connected domain.*

Suppose we let D_i denote the Gerschgorin disc associated with row i of our matrix $\widetilde{\mathcal{H}}$, then for $i \in \mathcal{A}$

$$D_i = \{z \in \mathbb{C} : \left| z - q_{ii} - \frac{1}{\mu}[s_i^*]^2 \right| \leq \sum_{j=1, j\neq i}^{n} |q_{ij}| + \sum_{j=1}^{m} |a_{ji}|\}, \tag{2.4.6}$$

for $i \in \mathcal{I}$

$$D_i = \{z \in \mathbb{C} : \left| z - q_{ii} - \mu[x_i^*]^{-2} \right| \leq \sum_{j=1, j\neq i}^{n} |q_{ij}| + \sum_{j=1}^{m} |a_{ji}|\}, \tag{2.4.7}$$

and for $i = n+1, \ldots, n+m$ we have

$$D_i = \{z \in \mathbb{C} : |z| \leq \sum_{j=1}^{n} a_{i-n,j}\}, \tag{2.4.8}$$

where $Q = \{q_{ij}\}$ and $A = \{a_{ij}\}$. For small μ we can therefore conclude that the eigenvalues $\{\lambda_i\}$ of $\mathcal{H}$ can be approximated by

$$\begin{aligned}|\lambda_i| &\approx \frac{\rho_i}{\mu}, \quad i \in \mathcal{A} \\ |\lambda_i| &\approx \rho_i, \quad i \in \mathcal{I} \cup \{n+1, \ldots, n+m\},\end{aligned}$$

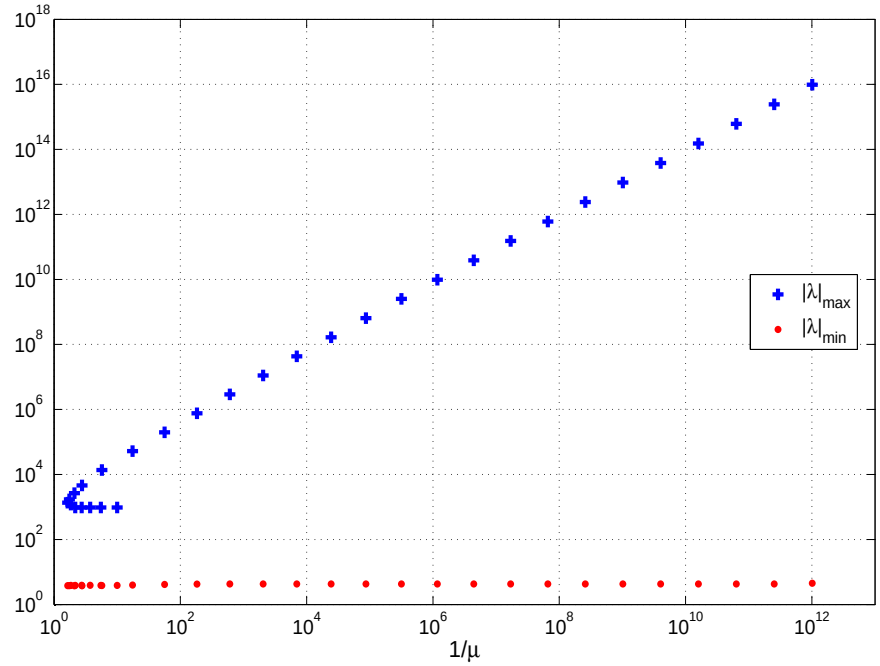

Figure 2.2: Maximum and minimum values of $|\lambda|$ associated with the KKT matrix for problem CVXQP2_S (m=25, n=100) plotted against the inverse of the barrier parameter, μ, as Mehrotra's Predictor-Corrector Method for finding x^* progresses.

where $\{\rho_i\}$ are positive constants. Therefore, as we draw near to optimality the saddle point systems will become very ill-conditioned.

We shall consider a small subset of problems from the `CUTEr` collection of quadratic programming problems [47] and use Mehrotra's Predictor-Corrector Method to find the optimal value x^*. When solving the saddle point system we shall use the backslash operator in MATLAB®7.0 for the few examples in this section. We shall store the matrices in the sparse format of MATLAB®, see [36].

Figure 2.2 shows how the maximum and minimum absolute eigenvalues vary with the barrier parameter, μ, as we run our algorithm for finding x^* for the problem CVXQP2_S. We observe that initially the value of μ increases because the starting choices of $s_0 = e$, $x_0 = e$ and $y_0 = e$ are far from s^*, x^*, and y^*. Once the values of s_k, x_k, and y_k have been modified enough by the algorithm, the value of μ starts to decrease and we start to converge towards the optimal solution. We find that, as expected, the maximum eigenvalue has the expected $\Theta(\frac{1}{\mu})$ behaviour as we approach optimality. The eigenvalue of

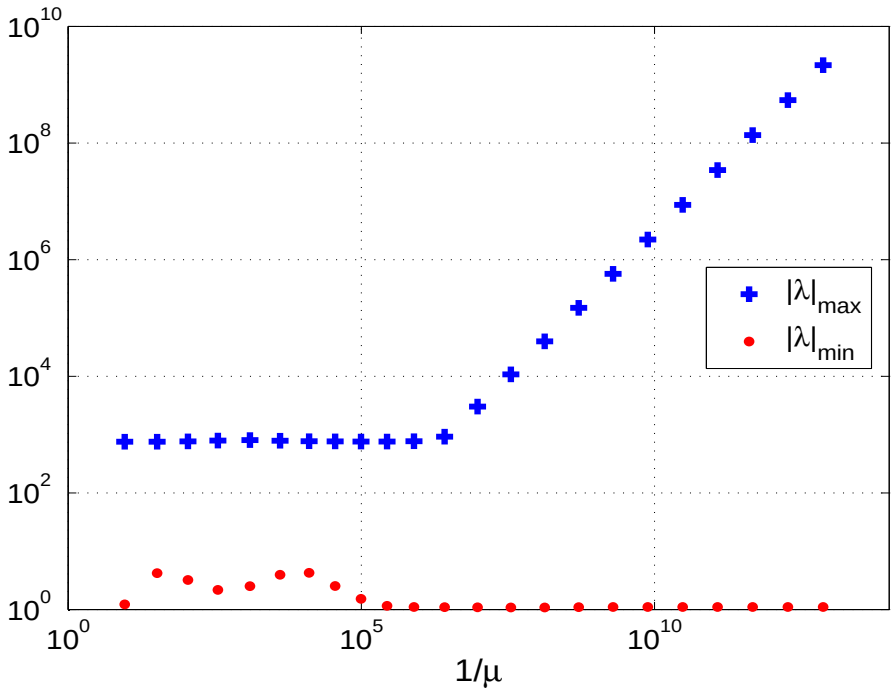

Figure 2.3: Maximum and minimum values of $|\lambda|$ associated with the KKT matrix for problem DUAL1 (m=1, n=85) plotted against the inverse of the barrier parameter, μ, as Mehrotra's Predictor-Corrector Method for finding x^* progresses.

minimum absolute value also has the predicted $\Theta(1)$ behaviour. This behaviour starts to occur almost immediately once the value of μ starts to decrease in this problem. The problem CVXQP2_S has $m = 25$, $n = 100$ and a KKT matrix with 820 non-zero entries. This corresponds to just 5.2% of the entries in the KKT matrix being filled with non-zero values — a density of 0.052.

Figure 2.3 shows how the maximum and minimum absolute eigenvalues vary with the barrier parameter, μ, as we run our algorithm for finding x^* for the problem DUAL1. This problem has $n = 85$ and just one equality constraint ($m = 1$). The corresponding KKT matrix has 7201 non-zero entries giving a density of 0.97. For $\mu < 10^{-7}$ we obtain the expected $\Theta(\frac{1}{\mu})$ behaviour of the maximum eigenvalue and $\Theta(1)$ behaviour of the eigenvalue of minimum absolute value.

2.5 Other applications requiring the solution of saddle point problems

Saddle point problems also arise in a variety of other applications. We shall indicate a couple of these applications here, but they will not be covered in the remainder of the work. The first example comes from an application in incompressible fluid dynamics:

Example 2.5.1 (Stokes)**.** Mixed finite element (and other) discretization of the Stokes equations

$$\begin{aligned} -\nabla^2 \vec{u} + \nabla p &= \vec{f} \quad \text{in } \Omega \\ \nabla \cdot \vec{u} &= 0 \quad \text{in } \Omega, \end{aligned}$$

for the fluid velocity $\vec{u}$ and pressure p in the domain $\Omega \subset \mathbb{R}^2$ or $\mathbb{R}^3$ yields linear systems in the saddle-point form (2.4.1) (for derivation and the following properties of this example see [28]). The symmetric block H arises from the diffusion terms $-\nabla^2 \vec{u}$ and A^T represents the discrete gradient operator whilst A represents its adjoint, the (negative) divergence. When (inf-sup) stable mixed finite element spaces are employed, $C = 0$, however for equal order and other spaces which are not inherently stable, stabilized formulations yield symmetric and positive semi-definite matrices C which typically have a large-dimensional kernel – for example, for the famous $\boldsymbol{Q}_1$–$\boldsymbol{P}_0$ element which has piecewise bilinear velocities and piecewise constant pressures in 2-dimensions, C typically has a kernel of dimension $m/4$.

The following example comes from image reconstruction and nonlinear image restoration applications [30]:

Example 2.5.2 (Weighted Toeplitz least squares)**.** Consider the weighted Toeplitz least squares problem

$$\min_y \|By - c\|_2^2, \tag{2.5.1}$$

where the rectangular coefficient matrix B and the right-hand side c are of the form

$$B = \begin{bmatrix} DA^T \\ \mu I \end{bmatrix} \quad \text{and} \quad c = \begin{bmatrix} Db \\ 0 \end{bmatrix}. \tag{2.5.2}$$

Here, A is a Toeplitz matrix, D is a non-constant diagonal matrix with real positive entries, b a given right-hand side and $\mu > 0$ a regularization parameter. If A is m by n ($m \leq n$), then it is straightforward to see that the problem (2.5.1) with B and c as in (2.5.2) is equivalent to the saddle point problem

$$\begin{bmatrix} D^{-2} & A^T \\ A & -\mu^2 I \end{bmatrix},$$

where the auxiliary variable $x = D(b - A^T y)$ represents a weighted residual.

A comprehensive list of other applications can be found in [8].

Chapter 3

Direct Solution Algorithms for Saddle Point Problems

Suppose we use one of the interior point methods described in the previous chapter to solve a constrained optimization problem. The most expensive part of the algorithm is finding the solution of the KKT system

$$\underbrace{\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} b \\ d \end{bmatrix}}_{c}, \tag{3.0.1}$$

where $H \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$ are symmetric and $A \in \mathbb{R}^{m\times n}$, for each value of μ, the barrier parameter. In this chapter we shall consider how we might solve these systems by the use of direct methods. In the following chapter we will look at a different class of methods that could be used: iterative methods.

3.1 Gaussian Elimination

One of the most widely known direct methods is that of Gaussian Elimination. This method transforms a linear system into an upper triangular one; in order to do this it applies simple linear transformations on the left. Suppose that we wish to solve a system

$$\mathcal{H}u = c,$$

where $\mathcal{H} \in \mathbb{C}^{N\times N}$ is a square matrix. The main idea is to transform $\mathcal{H}$ into an $N \times N$ upper triangular matrix U by introducing zeros below the diagonal, first in column 1, then column 2, and so on. To do this we subtract multiples of each row with subsequent rows. We can think of this as being equivalent to

premultiplying $\mathcal{H}$ by a sequence of lower triangular matrices L_k:

$$\underbrace{L_{m-1} \dots L_2 L_1}_{L^{-1}} \mathcal{H} = U.$$

We can then obtain an LU factorization of $\mathcal{H}$,

$$\mathcal{H} = LU,$$

where U is upper triangular and L is unit lower triangular. Backward and forward substitutions are then used to solve the linear system.

In practical Gaussian elimination methods the matrices L_k are not formed and multiplied together. Instead, the multipliers ℓ_{jk} are computed and stored directly into L [81]. To increase the stability of the Gaussian elimination method a pivoting procedure is often added into the algorithm: this may be in the form of *complete pivoting* [81] or *threshold pivoting* [23]. We can view this as there being a permutation matrix P such that $P\mathcal{H} = LU$. A column permutation may also be applied to increase the stability: this can be viewed as there also being a permutation matrix Q such that $P\mathcal{H}Q = LU$.

The computation cost of this method can be expressed as

$$\text{work} = \frac{2}{3}N^3 + \mathcal{O}\left(N^2\right) \text{ flops},$$

where each addition, subtraction, multiplication, division or square root counts as a *flop* (floating point operation) [81].

3.2 Cholesky-Type Factorization

It is a basic belief that structure should be exploited when solving a problem. In the previous section we introduced Gaussian elimination as a possible method for solving systems of linear equations; this is a very general method which can be used to solve nonsingular systems of equations. A variant of the LU factorization is the LDM$^{\mathrm{T}}$ factorization where $\mathcal{H}$ is factored as LDM^T with D diagonal, and L, M are unit lower triangular matrices. We observe that the LDM$^{\mathrm{T}}$ factorization can be found by using Gaussian elimination to compute $\mathcal{H} = LU$ and then determining D and M from the equation $U = DM^T$. An alternative algorithm for computing L, D, and M can be found in [41, Section 4.1.1]. If $\mathcal{H}$ is symmetric, then we have the following theorem:

Theorem 3.2.1. *If $\mathcal{H} = LDM^T$ is the LDM^T factorization of a nonsingular symmetric matrix $\mathcal{H}$, then $L = M$.*

Proof. The matrix $M^{-1}\mathcal{H}M^{-T} = M^{-1}LD$ is both symmetric and lower triangular and therefore diagonal. Since D is nonsingular, this implies that $M^{-1}L$ is also diagonal. But $M^{-1}L$ is unit lower triangular and so $M^{-1}L = I$. □

This result halves the amount of work required to carry out an LDM$^{\mathrm{T}}$ factorization when it is applied to a symmetric matrix. If $\mathcal{H}$ is symmetric and not positive definite, then pivoting may be necessary (see Section 3.2.2).

3.2.1 Symmetric Positive Definite Systems

If $\mathcal{H}$ is positive definite, then the factorization $\mathcal{H} = LDL^T$ exists and D has positive diagonal entries [41]. If $\mathcal{H}$ is also symmetric then we can factor it such that $\mathcal{H} = LD^{\frac{1}{2}}\left(LD^{\frac{1}{2}}\right)^T$, where $D = \mathrm{diag}\,(d_1, d_2, \ldots, d_m)$ is given by the factorization $\mathcal{H} = LDL^T$ and $D^{\frac{1}{2}} = \mathrm{diag}\left(\sqrt{d_1}, \sqrt{d_2}, \ldots, \sqrt{d_m}\right)$. Suppose we set $\widehat{L} = LD^{\frac{1}{2}}$, then the factorization $\mathcal{H} = \widehat{L}\widehat{L}^T$ is known as the *Cholesky factorization.*

More efficient algorithms for computing the Cholesky factorization are available than that suggested by the construction of $\widehat{L}$ above, see [41].

3.2.2 Symmetric Indefinite Systems

If a matrix, $\mathcal{H}$, is symmetric and indefinite, then, although it may have an LDL$^{\mathrm{T}}$ factorization, the entries in the factors can have arbitrary magnitude. A well known example for this can be found in [41]:

$$\begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{\epsilon} & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 0 \\ 0 & -\frac{1}{\epsilon} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{\epsilon} \\ 0 & 1 \end{bmatrix}.$$

The difference in magnitudes of the entries in the factors can cause stability problems. When carrying out Gaussian elimination a pivoting strategy is applied to try to improve the stability of the method. However, many of the pivoting strategies suggested would destroy the symmetry of the problem if we were to use them, and, hence, we would lose the "Cholesky speed" of our proposed algorithm; symmetric pivoting must be used, i.e., $\mathcal{H} \longleftarrow P\mathcal{H}P^T$, to maintain the symmetry. Unfortunately this does not always stabilize the

$\mathrm{LDL^T}$ computation because small entries on the diagonal will not be removed and so large numbers will be found in the factorization.

Fortunately this sort of factorization can be modified to give a stable factorization for symmetric indefinite systems. This factorization takes the form

$$\mathcal{H} = P^T L D L^T P,$$

where P is a permutation matrix, L is unit lower triangular, and D is a block diagonal matrix with blocks of dimension 1 and 2. This factorization was developed by Bunch and Parlett [13] after initial ideas suggested by W. Kahan (1965) (in correspondence with R. de Meersman and L. Schotsman) using the work of Lagrange (1759). The algorithm of Bunch and Parlett is stable with a cost comparible to that of a Cholesky factorization for symmetric positive definite matrices. They use a pivoting strategy which is like that of complete pivoting [11, 12]. Alternative pivoting strategies have subsequently been developed which generally require fewer comparisons. The Bunch-Kaufman pivoting strategy is now widely accepted to be the algorithm of choice with many implementations available (for example, `MA27` and `MA57` from the `HSL` library [24, 22] are based on such algorithms but additionally take sparsity into account).

3.3 Schur Complement

Let us assume that H and $\mathcal{H}$ are nonsingular, then by (2.4.2) the matrix $S = -\left(C + AH^{-1}A^T\right)$ is also nonsingular. Expanding the saddle point system gives

$$Hx + A^T y = b, \tag{3.3.1}$$

$$Ax - Cy = d. \tag{3.3.2}$$

Premultiplying both sides of (3.3.1) by AH^{-1} we obtain

$$Ax + AH^{-1}A^T y = AH^{-1}b.$$

Using (3.3.2) and rearranging gives

$$\left(C + AH^{-1}A^T\right) y = AH^{-1}b - d. \tag{3.3.3}$$

This is a reduced system involving the (negative) Schur complement $-S = C + AH^{-1}A^T$. Once we know y^* we can use (3.3.1) to obtain x^* :

$$Hx^* = b - A^T y^*. \tag{3.3.4}$$

This method is really just block Gaussian elimination applied to (3.0.1). Observe that using block LU factorization we get the system

$$\begin{bmatrix} I & 0 \\ -AH^{-1} & I \end{bmatrix} \begin{bmatrix} H & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} I & 0 \\ -AH^{-1} & I \end{bmatrix} \begin{bmatrix} b \\ d \end{bmatrix},$$

i.e.,

$$\begin{bmatrix} H & A^T \\ 0 & S \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d - AH^{-1}b \end{bmatrix}.$$

We can solve this system by the use of block back substitution. This leads to the two reduced systems (3.3.3) and (3.3.4). If H and $-S$ are symmetric positive definite then we can use methods such as Cholesky factorization or preconditioned conjugate gradients to solve these systems (see Sections 3.2 and 4.2.2 respectively).

In structural mechanics this method is known as the *displacement method*. In electrical engineering it is called the *nodal analysis method*, whilst in optimization it is the *range-space method*. In all these applications H is symmetric positive (semi)definite and $C = 0$.

If m is sufficiently small and if the linear systems involving the coefficient matrix H can be solved efficiently, then this method will be attractive. Its main disadvantages are that H must be nonsingular and S may be full and too expensive to compute or factor (even though the saddle point system was originally assumed to be sparse).

In cases where H is positive semidefinite and singular, the Schur complement reduction method may still be applied by firstly making use of the augmented Lagrangian technique (Section 5.2) to produce an equivalent saddle point system in which the (1,1) block is nonsingular.

3.4 Nullspace Methods

In this section we shall assume that $C = 0$, $\ker A \cap \ker H = \{0\}$, and H is symmetric and positive definite in the nullspace of A. The nullspace method assumes that we have

- A matrix $Z \in \mathbb{R}^{n\times(n-m)}$ such that $AZ = 0$; that is range$(Z) = \ker A$,
- A matrix $Y \in \mathbb{R}^{n\times m}$ such that $\begin{bmatrix} Y & Z \end{bmatrix}$ spans $\mathbb{R}^n$; that is range$(Y) =$ range(A^T).

Any solution $\widehat{x}$ of the linear equations $Ax = d$ can be written as

$$\widehat{x} = Yx_{_Y} + Zx_{_Z}. \tag{3.4.1}$$

The saddle point system (3.0.1) can therefore be expressed as

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \underbrace{\begin{bmatrix} Y & Z & 0 \\ 0 & 0 & I \end{bmatrix}}_{\mathcal{Y}} \begin{bmatrix} x_{_Y} \\ x_{_Z} \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}.$$

Premultiplying this expression by $\mathcal{Y}^T$ gives

$$\begin{bmatrix} Y^THY & Y^THZ & Y^TA^T \\ Z^THY & Z^THZ & 0 \\ AY & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{_Y} \\ x_{_Z} \\ y \end{bmatrix} = \begin{bmatrix} Y^Tb \\ Z^Tb \\ d \end{bmatrix}. \tag{3.4.2}$$

In practice Y is often set to be equal to A^T. In this case we observe that an $m \times m$ system determines $x^*_{_Y}$:

$$AA^Tx^*_{_Y} = d. \tag{3.4.3}$$

Since A is of full rank, AA^T is symmetric, positive definite; we could therefore solve this system using the Cholesky factorization method on AA^T if the dimension of the system is small enough [41, p. 143]. From (3.4.2), having found $x^*_{_Y}$ we can find $x^*_{_Z}$ by solving

$$H_{_{ZZ}}x^*_{_Z} = -b_{_Z}, \tag{3.4.4}$$

where

$$H_{_{ZZ}} = Z^THZ, \qquad b_{_Z} = Z^T(HA^Tx^*_{_Y} - b).$$

The matrix H is symmetric and positive definite in the nullspace of A, hence Z^THZ is symmetric and positive definite. A Cholesky factorization could be used to solve this system or the conjugate gradient method applied to compute an approximate solution to the system (3.4.4). We will look at the application of the conjugate gradient method in Chapter 5.

Substituting x_Z^* into (3.4.1) will give us a (possibly approximate) solution for x^*. Using (3.4.2) with (3.4.1) we obtain a system that can be solved to give a (approximate) solution for y^*:

$$AA^T y^* = A(b - Hx^*). \tag{3.4.5}$$

If we used a Cholesky factorization to find x_Y^*, then this same factorization could be employed to solve (3.4.5).

Nullspace methods are quite popular in optimization, where they are usually referred to as *reduced (or projected) Hessian methods* [39, 62]. The nullspace method is also used extensively in structural mechanics where it is known as the *force method*, since x, the vector of internal forces, is calculated first. In fluid mechanics this method is called the *dual variable method*, and in electrical engineering it goes under the name of *loop analysis* [8].

Nullspace methods are particularly attractive when $n - m$ is small. If Z is sparse then it may be possible to factor $Z^T HZ$ explicitly, otherwise iterative methods might be used. This method is less attractive if $n - m$ is large, and it cannot be directly applied to the case $C \neq 0$ (see Section 5.3). However, its main difficulty lies in the necessity of needing a nullspace basis Z. Different nullspace bases can be used implying that there is a whole family of nullspace methods.

One possible candidate for Z is known as the *fundamental basis.* Let $\widetilde{P}$ denote a permutation matrix such that $A\widetilde{P} = \begin{bmatrix} A_1 & A_2 \end{bmatrix}$, where A_1 is $m \times m$ and nonsingular. It is straightforward to verify that the matrix

$$Z = \widetilde{P} \begin{bmatrix} -A_1^{-1} A_2 \\ I \end{bmatrix} \tag{3.4.6}$$

is a nullspace basis for A.

In principle there are many possible choices for $\widetilde{P}$ such that the resulting A_1 is nonsingular. One possibility for selecting $\widetilde{P}$ is to carry out an LU factorization with pivoting of A^T, i.e., use Gaussian elimination to find $\widetilde{P}$, L and U such that $A^T = \widetilde{P}LU$, L is unit lower triangular, and U is upper triangular. We shall consider other possible methods for choosing $\widetilde{P}$ in Chapter 8.

Backward stability of the nullspace method for two different choices of Z is shown in [1, 2].

3.5 Solving ill-conditioned systems

The effects of changes in the c and $\mathcal{H}$ on the exact solution of $\mathcal{H}u = c$ are well known [41, 52]. Let $\widetilde{u}$ denote the exact solution of $\mathcal{H}\widetilde{u} = c + \Delta c$, and let $\Delta u = \widetilde{u} - u$. Then

$$\|\Delta u\| \leq \left\|\mathcal{H}^{-1}\right\| \|\Delta c\| \quad \text{and} \quad \frac{\|\Delta u\|}{\|u\|} \leq \kappa(\mathcal{H}) \frac{\|\Delta c\|}{\|c\|}, \tag{3.5.1}$$

where $\kappa(\mathcal{H}) = \|\mathcal{H}\| \|\mathcal{H}^{-1}\|$ is called the condition number of $\mathcal{H}$. Equality can be achieved in both of these inequalities.

When the matrix is perturbed by $\Delta\mathcal{H}$, the exact solution $\widetilde{u}$ of the perturbed system satisfies

$$(\mathcal{H} + \Delta\mathcal{H})\,\widetilde{u} = \mathcal{H}u = c, \quad \text{or} \quad \widetilde{u} - u = -\left(\mathcal{H} + \Delta\mathcal{H}\right)^{-1} \Delta\mathcal{H}u. \tag{3.5.2}$$

Ignoring second order terms, an approximation to (3.5.2) is satisfied by $\Delta u \approx \widetilde{u} - u$:

$$\mathcal{H}\Delta u = -\Delta\mathcal{H}u, \quad \text{or} \quad \Delta u = -\mathcal{H}^{-1}\Delta\mathcal{H}u, \tag{3.5.3}$$

giving the bounds

$$\|\Delta u\| \leq \left\|\mathcal{H}^{-1}\right\| \|\Delta\mathcal{H}\| \|u\| \quad \text{and} \quad \frac{\|\Delta u\|}{\|u\|} \leq \kappa(\mathcal{H}) \frac{\|\Delta\mathcal{H}\|}{\|\mathcal{H}\|}. \tag{3.5.4}$$

Equality can hold in these relations for any vector c [52]. Since these bounds in (3.5.1) and (3.5.4) can be achieved, when $\mathcal{H}$ is ill-conditioned we might expect substantial relative inaccuracy in the computed solution.

Most of the widely used numerical methods provide a computed solution of a linear system which is typically the exact solution of a nearby problem (see, for example, [41, 52]). In particular, when solving the symmetric system $\mathcal{H}u = c$ in finite precision with any backward-stable method, the computed solution $\widetilde{u}$ is the exact solution of a nearby system involving a perturbed symmetric matrix $\widetilde{\mathcal{H}}$:

$$\widetilde{\mathcal{H}}\widetilde{u} = c,$$

where $\widetilde{\mathcal{H}} = \mathcal{H} + \Delta\mathcal{H}$ and $\Delta\mathcal{H} = (\Delta\mathcal{H})^T$. The most common backward-stable methods performed on a machine with unit roundoff $\mathbf{u}$ will produce a perturbation $\Delta\mathcal{H}$ which satisfies

$$\|\Delta\mathcal{H}\| \leq \mathbf{u}\gamma_N \|\mathcal{H}\|,$$

where γ_N is a function involving a low-order polynomial in N and characteristics of $\mathcal{H}$ (such as the growth factor). This function γ_N is known under various conditions for different classes of problems [52].

We have seen in Chapter 2 how $\mathcal{H}$ can become ill-conditioned as we draw near to the optimal point when solving a quadratic programming problem with an interior point method. However, the structure of $\mathcal{H}$ means that the singular values split into two well behaved groups. Wright [86] shows that this, along with the way errors form in the right hand side c, results in $\widetilde{u}$ having absolute error comparable to machine precision. Thus, the ill-conditioning of $\mathcal{H}$ does not noticeably impair the accuracy of the computed search direction.

The size and structure of some of the constrained optimization problems have started to outgrow the software packages available for solving saddle point problems through direct methods [9]. In the following chapter we shall consider how such systems may be solved via iterative methods.

Chapter 4

Iterative Solution Algorithms for Saddle Point Problems

Direct methods for sparse linear systems are often unsuitable for solving large saddle point problems of the form

$$\underbrace{\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}, \tag{4.0.1}$$

where $H \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$ are symmetric and $A \in \mathbb{R}^{m\times n}$ with $m \leq n$. As we assumed in Chapter 1, the coefficient matrix is sparse, so the linear system (4.0.1) may be solved efficiently with iterative solvers. We shall firstly consider stationary schemes and then go on to look at Krylov subspace methods.

4.1 Stationary Iterations

Stationary iterations are now more commonly used as preconditioners for Krylov subspace methods, but they have also been popular for many years as "standalone" solvers. Another common use for these methods is as smoothers for multigrid methods, see [8, Section 11].

4.1.1 Arrow-Hurwicz and Uzawa methods

Arrow, Hurwicz and Uzawa developed one of the first iterative schemes for the solution of a general type of saddle point system [3]. The Arrow-Hurwicz and Uzawa methods use simultaneous iterations for both x and y, and can be expressed in terms of splittings of the coefficient matrix $\mathcal{H}$.

Uzawa's method is particularly well known in fluid dynamics, especially for solving the (steady) Stokes problem [72]. For simplicity we assume that H is invertible and $C = 0$, but generalization is straightforward. Uzawa's method is given in Algorithm 4.1.1: the parameter $\omega > 0$ is a relaxation parameter.

Algorithm 4.1.1 Uzawa's method.

Choose x_0 and y_0.
for $k = 0, 1, \ldots$ **do**
  $x_{k+1} = H^{-1}(b - A^T y_k)$,
  $y_{k+1} = y_k + \omega(Ax_{k+1} - d)$.
end for

The iteration in Algorithm 4.1.1 can be written in terms of a matrix splitting $\mathcal{H} = \mathcal{P} - \mathcal{Q}$ as the fixed-point iteration

$$u_{k+1} = \mathcal{P}^{-1}Qu_k + \mathcal{P}^{-1}c,$$

where

$$\mathcal{P} = \begin{bmatrix} H & 0 \\ A & -\frac{1}{\omega}I \end{bmatrix}, \quad \mathcal{Q} = \begin{bmatrix} 0 & -A^T \\ 0 & -\frac{1}{\omega}I \end{bmatrix}, \quad c = \begin{bmatrix} b \\ d \end{bmatrix}, \quad \text{and} \quad u_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}. \tag{4.1.1}$$

If, instead, we eliminate x_k from the iteration in Algorithm 4.1.1, then we obtain

$$y_{k+1} = y_k + \omega \left(AH^{-1}(b - A^T y_k) - d \right). \tag{4.1.2}$$

This is nothing but a Richardson iteration for solving the linear system

$$AH^{-1}A^T y = AH^{-1}b - d.$$

If H is symmetric and positive definite, then the Schur complement $AH^{-1}A^T$ is also positive definite. If $\lambda_{\max}$ is the largest eigenvalue of $AH^{-1}A^T$, then it is known that the Richardson's iteration (4.1.2) converges for all ω such that

$$0 < \omega < \frac{2}{\lambda_{\max}},$$

see [72].

Solves with the system H may be too expensive, so the Arrow-Hurwicz method can be used instead. This method may be regarded as an inexpensive alternative to Uzawa's method.

In Uzawa's method, Algorithm 4.1.1, finding x_{k+1} by solving a system involving H is equivalent to finding the minimum of the quadratic function

$$\min_{x\in\mathbb{R}^n} f_k(x) = \frac{1}{2}\langle x, Hx\rangle + \langle x, b - Ay_k\rangle . \tag{4.1.3}$$

We can therefore derive a less expensive method by taking one step in the direction of the (negative) gradient of $f_k(x)$ with a fixed step length α. The resulting Arrow-Hurwicz method is given in Algorithm 4.1.2.

Algorithm 4.1.2 Arrow-Hurwicz method.

Choose x_0 and y_0.
for $k = 0, 1, \ldots$ **do**
  $x_{k+1} = x_k + \alpha(b - Hx_k - A^T y_k)$,
  $y_{k+1} = y_k + \omega(Ax_{k+1} - d)$.
end for

As we did for the Uzawa method, we can recast the Arrow-Hurwicz iteration into a fixed-point iteration induced by the splitting

$$\mathcal{P} = \begin{bmatrix} \frac{1}{\alpha}I & 0 \\ A & -\frac{1}{\omega}I \end{bmatrix}, \quad \mathcal{Q} = \begin{bmatrix} \frac{1}{\alpha}I - H & -A^T \\ 0 & -\frac{1}{\omega}I \end{bmatrix}, \quad c = \begin{bmatrix} b \\ d \end{bmatrix}, \quad \text{and} \quad u_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}. \tag{4.1.4}$$

Convergence of this method is usually rather slow, so various improvements have been suggested, including the idea of a preconditioned variant. For more details see [8].

4.2 Krylov Subspace Methods

In this section we will consider Krylov subspace methods for solving (preconditioned) saddle point problems. Rather than discussing all existing methods and implementations, we will describe the main properties of the most commonly used methods.

4.2.1 General Krylov Subspace Theory

Suppose we wish to solve a system of the form

$$\mathcal{H}u = c. \tag{4.2.1}$$

Let u_0 be an initial guess for the solution u and define the initial residual to be $r_0 = c - \mathcal{H}u_0$. Krylov subspace methods are iterative methods whose kth iterate u_k satisfies

$$u_k \in u_0 + \mathcal{K}_k(\mathcal{H}, r_0), \quad k = 1, 2, \ldots, \tag{4.2.2}$$

where

$$\mathcal{K}_k(\mathcal{H}, r_0) \equiv \mathtt{span}\{r_0, \mathcal{H}r_0, \ldots, \mathcal{H}^{k-1}r_0\} \tag{4.2.3}$$

denotes the kth Krylov subspace generated by $\mathcal{H}$ and r_0. Krylov subspace methods are typically used to solve large sparse systems. They involve finding an "optimal" solution in a given space, augmenting the space, and repeating the procedure. The basis for $\mathcal{K}_k(\mathcal{H}, r_0)$ given in (4.2.3) is not used: an orthogonal basis for the Krylov subspace is usually created instead. More complete descriptions of Krylov subspace methods can be found in [49, 72, 84].

4.2.2 Preconditioned Conjugate Gradient Method

The conjugate gradient (CG) method is one of the best known iterative techniques for solving *sparse symmetric positive definite* linear systems. The method converges to the solution via the minimization of the $\mathcal{H}$-norm of the error as the Krylov subspace is increased at each step [51]. If $\mathcal{H} \in \mathbb{R}^{N\times N}$, then the method will take at most N steps to calculate the exact solution but rounding errors may prevent this. The systems being solved also frequently had N being prohibitively large. However, in practice, convergence to acceptable accuracy often occurs after only a few steps [69]. The CG method is given in Algorithm 4.2.1.

Algorithm 4.2.1 Conjugate Gradient Method.

Choose u_0.
Set $r_0 = c - \mathcal{H}u_0$ and $p_0 = r_0$.
for $k = 0, 1, \ldots$ **do**
  $\alpha_k = \langle r_k, r_k\rangle \,/\, \langle \mathcal{H}p_k, p_k\rangle$,
  $u_{k+1} = u_k + \alpha_k p_k$,
  $r_{k+1} = r_k - \alpha_k \mathcal{H}p_k$,
  $\beta_k = \langle r_{k+1}, r_{k+1}\rangle \,/\, \langle r_k, r_k\rangle$,
  $p_{k+1} = r_{k+1} + \beta_k p_k$.
end for

This method uses a 3-term recurrence relation, so as we increase the subspace from which we seek a solution, we need only recall the approximations

from the two most recent subspaces to produce the u_k that minimizes $\|e_k\|_{\mathcal{H}}$, where $e_k = u - u_k$ is the error at the kth step. Hence, the memory requirements will be small and so is the computation for each iteration. The error can be bounded from above by the following (classical) convergence theorem [41, 72, 81, 84]:

Theorem 4.2.1. *After k steps of the conjugate gradient method, the iteration error $e_k = u - u_k$ satisfies the bound*

$$\|e_k\|_{\mathcal{H}} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \|e_0\|_{\mathcal{H}}\,, \tag{4.2.4}$$

where $\kappa = \lambda_{max}(\mathcal{H})/\lambda_{min}(\mathcal{H})$.

The bound (4.2.4) intuitively leads to the notion that if a matrix has small condition number κ, then the convergence of CG will be rapid. However, the converse isn't true. Using this observation, we consider the idea of preconditioning. The basic idea is to construct a matrix K that approximates the coefficient matrix $\mathcal{H}$ but such that little work is required for carrying out $K^{-1}v$ for some given vector v. We can then consider solving

$$K^{-1}\mathcal{H}u = K^{-1}c \tag{4.2.5}$$

instead of $\mathcal{H}u = c$. If K is a good approximation to $\mathcal{H}$, then we might expect the convergence of the CG method to be more rapid for the preconditioned system (4.2.5) than the original problem.

The preconditioned coefficient matrix must be symmetric for us to apply the CG method, so suppose that we choose a symmetric positive definite matrix K, let $K = MM^T$, and consider the system

$$M^{-1}\mathcal{H}M^{-T}v = K^{-1}c, \quad v = M^T u. \tag{4.2.6}$$

The coefficient matrix $M^{-1}\mathcal{H}M^{-T}$ is symmetric positive definite, so the CG method can be applied. We note that at the kth iteration, the preconditioned method and the original CG method will (in exact arithmetic) produce the same u_k because we have

$$\begin{aligned}
\|v - v_k\|^2_{M^{-1}\mathcal{H}M^{-T}} &= \left\|M^T(u-u_k)\right\|^2_{M^{-1}\mathcal{H}M^{-T}} \\
&= (u-u_k)^T M(M^{-1}\mathcal{H}M^{-T})M^T(u-u_k) \\
&= \|u-u_k\|^2_{\mathcal{H}}\,.
\end{aligned}$$

Theorem 4.2.1 shows that the convergence of the preconditioned conjugate gradient (PCG) iteration depends on the eigenvalues of $M^{-1}\mathcal{H}M^{-T}$, which are identical to the eigenvalues of $K^{-1}\mathcal{H}$ because of the similarity transformation $M^{-T}(M^{-1}\mathcal{H}M^{-T})M^{T} = K^{-1}\mathcal{H}$. The preconditioned conjugate gradient (PCG) method is given in Algorithm 4.2.2.

Algorithm 4.2.2 Preconditioned Conjugate Gradient Method.

Choose u_0.
Set $r_0 = c - \mathcal{H}u_0$.
Solve $Kz_0 = r_0$.
$p_0 = z_0$.
for $k = 0, 1, \ldots$ **do**
  $\alpha_k = \langle z_k, r_k \rangle \,/\, \langle \mathcal{H}p_k, p_k \rangle$,
  $u_{k+1} = u_k + \alpha_k p_k$,
  $r_{k+1} = r_k - \alpha_k \mathcal{H}p_k$,
  Solve $Kz_{k+1} = r_{k+1}$,
  $\beta_k = \langle z_{k+1}, r_{k+1} \rangle \,/\, \langle z_k, r_k \rangle$,
  $p_{k+1} = z_{k+1} + \beta_k p_k$.
end for

The requirement that $\mathcal{H}$ and K are symmetric positive definite is needed to prevent the possible breakdown in the calculation of α_k and β_k in Algorithms 4.2.1 and 4.2.2. If our matrices do not fulfil this requirement, then we will normally need to use another method. If the matrix is indefinite but symmetric, then we could use the MINRES or SYMMLQ algorithms [64]. Instead of minimizing the $\mathcal{H}$-norm of the error, the solution using MINRES is found via minimization of the 2-norm of the residual in Krylov subspaces of increasing dimension. SYMMLQ minimizes the 2-norm of the error over a different Krylov space, and is based on the LQ factorization of the tridiagonal matrices formed in the Lanczos method.

4.2.3 Generalized Minimum Residual Method

If our coefficient matrix, $\mathcal{H} \in \mathbb{R}^{N\times N}$, is unsymmetric we can still seek to find an approximation in a particular subspace which minimizes the 2-norm of the residual. The Generalized Minimum Residual (GMRES) Method [73] is one such procedure that is also robust. The algorithm generates an orthogonal basis for the Krylov subspace via the Arnoldi method, Algorithm 4.2.3. Algorithm 4.2.4 then gives the GMRES method which makes use of the Arnoldi process.

Algorithm 4.2.3 Arnoldi Method.

Given q_1 such that $\|q_1\| = 1$.
for $j = 1, 2, \ldots$ **do**
  $\widetilde{q}_{k+1} = \mathcal{H} q_k$,
  for $i = 1, 2, \ldots, j$ **do**
    $h_{ij} = \langle \widetilde{q}_{j+1}, q_i \rangle$,
    $\widetilde{q}_{j+1} = \widetilde{q}_{j+1} - h_{ij} q_i$,
  end for
  $h_{j+1,j} = \|\widetilde{q}_{j+1}\|$,
  $q_{j+1} = \widetilde{q}_{j+1} / h_{j+1,j}$.
end for

Algorithm 4.2.4 GMRES Method.

Choose u_0.
Set $r_0 = c - \mathcal{H} u_0$.
Set $q_1 = r_0 / \|r_0\|$.
for $k = 1, 2, \ldots$ **do**
  Compute q_{k+1} and $h_{i,k}$, $i = 1, 2, \ldots, k+1$ using Arnoldi,
  Solve the least squares problem $\min_y \|\beta e_1 - H_{k+1,k} y\|$ to find y_k, where $\beta = \|r_0\|$,
  Set $u_k = u_0 + Q_k y_k$, where $Q_k \in \mathbb{R}^{N \times k}$ has as columns the orthogonal basis vectors q_i.
end for

The convergence of GMRES is not as clear as it is in the case of symmetric problems. See [72, Section 6.11.4] for a concise description of the convergence properties for this method. In a similar manner to the conjugate gradient method, preconditioners are often used in conjunction with GMRES to improve the rate of convergence. Left preconditioning, right preconditioning, or a mixture of both may be employed. With left preconditioning we solve the problem

$$K^{-1}\mathcal{H}u = K^{-1}c$$

instead of the original problem, where K is the chosen preconditioner. For right preconditioning the problem

$$\mathcal{H}K^{-1}v = c$$

is solved and then we solve $Ku = v$ to find u. If K_1 and K_2 are the left and right preconditioners, respectively, then the mixed preconditioning means that the problem

$$K_1^{-1}\mathcal{H}K_2^{-1}v = K_1^{-1}c$$

is solved and then $K_2u = v$ is solved to give u. It is important to note that left preconditioning will change the norm in which we are minimizing the residual when we apply GMRES or MINRES:

$$\begin{aligned} \left\|K^{-1}\mathcal{H}u - K^{-1}c\right\|_2^2 &= \left\|K^{-1}(\mathcal{H}u - c)\right\|_2^2 \\ &= (\mathcal{H}u - c)^T K^{-T}K^{-1}(\mathcal{H}u - c) \\ &= \|\mathcal{H}u - c\|^2_{(KK^T)^{-1}} . \end{aligned}$$

4.2.4 Other Krylov Subspace Methods

Although GMRES is the "ideal" choice of iterative solver for large unsymmetric problems, in terms of it producing the "optimal" solution in a Krylov subspace at each iteration, it is often not used in practice. The GMRES method does not have a short recurrence that can be exploited as in CG and MINRES, so another vector must be stored at each iteration of the GMRES method and an increasing amount of work has to be performed. This can result in the method becoming prohibitively expensive to use if an acceptably accurate solution isn't found rapidly. We shall outline some methods which have been developed to try to overcome this problem.

Restarted GMRES

As we just noted, the GMRES method can have excessive computational and storage needs. One way to try to curb this is the restarted GMRES method, GMRES(m), which restarts the algorithm every m iterations with u_m being used as the new initial guess. Although this can be successful, the problem of finding a good m can be difficult because the convergence behaviour is not well known. It is also not always the case that GMRES(m) performs better as m increases. Indeed, it was shown by Embree [29] that there are some problems which can be solved by GMRES(1) for which GMRES(2) stagnates!

Bi-Conjugate Gradient Method

Symmetric problems do not suffer from this ever increasing workload and storage need because we can use a 3-term recurrence relation to form the orthogonal basis. Hence, one idea is to reinterpret the unsymmetric problem as symmetric problem. Suppose that instead of considering the unsymmetric problem

$$\mathcal{H}u = c,$$

we instead solve

$$\widetilde{\mathcal{H}} \begin{bmatrix} \widetilde{u} \\ u \end{bmatrix} = \begin{bmatrix} 0 & \mathcal{H} \\ \mathcal{H}^T & 0 \end{bmatrix} \begin{bmatrix} \widetilde{u} \\ u \end{bmatrix} = \begin{bmatrix} \widetilde{c} \\ c \end{bmatrix}$$

with preconditioner

$$\widetilde{K} = \begin{bmatrix} 0 & K \\ K^T & 0 \end{bmatrix},$$

where K is a preconditioner for $\mathcal{H}$, via the conjugate gradient method, see Section 4.2.2. This leads to the Bi-CG method for $\mathcal{H}$.

An extension to Bi-CG is the Bi-CGSTAB method [83]: this method is really a combination of Bi-CG and GMRES(1). Further variations have been developed from this idea to produce Bi-CGSTAB(l), $l > 1$, which in essence is a combination of Bi-CG and GMRES(l).

We have only touched on the subject of Krylov subspace methods in this section. Table 4.1 summarizes the methods that we have introduced.

Method	Required $\mathcal{H}$	Type	Recurrence	Required K
CG	symm. def.	optimal	3-term	symm. def.
MINRES/SYMMLQ	symm. indef.	optimal	3-term	symm. def.
GMRES	general	optimal	full	general
Bi-CGSTAB	general	nonoptimal	3-term	general

Table 4.1: Summary of Krylov subspace methods discussed in Section 4.2

Chapter 5

The Preconditioned Conjugate Gradient Method Applied to Saddle Point Problems

In Section 2.4.2 we saw that the linear systems (saddle point problems) we wish to solve are indefinite. It, therefore, cannot be assumed that the preconditioned conjugate gradient method can be immediately applied to solve the saddle point problems. If we are able to use a conjugate gradient style method to solve these systems, then we can use the known convergence theorems of PCG to help us choose effective preconditioners. In the first two sections we shall consider saddle point systems with $C = 0$, but this will then be extended to the case $C \neq 0$.

5.1 Projected Conjugate Gradient Method for the Case $C = 0$

Let us assume that $C = 0$; as is the case in the mixed constraints optimization problems of Section 2.3. The resulting systems take the form

$$\underbrace{\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}, \tag{5.1.1}$$

where $H \in \mathbb{R}^{n\times n}$ is symmetric and $A \in \mathbb{R}^{m\times n}$ $(m \leq n)$ has full rank. We shall assume that H is positive definite in the nullspace of A.

5.1.1 CG method for the reduced system

We wish to solve the system (5.1.1) to find $[x^{*T}y^{*T}]^T$. We start in a similar manner to Section 3.4 and go on to derive the projected conjugate gradient method as found in [44]. Let Z be an $n\times(n-m)$ matrix spanning the nullspace of A; then $AZ = 0$. The columns of A^T together with the columns of Z span $\mathbb{R}^n$ and any solution x^* of linear equations $Ax = d$ can be written as

$$x^* = A^T x^*_{_A} + Z x^*_{_Z}. \tag{5.1.2}$$

Substituting this into (5.1.1) gives

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} A^T x^*_{_A} + Z x^*_{_Z} \\ y^* \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}. \tag{5.1.3}$$

Let us split the matrix $\mathcal{H}$ into a block 3×3 structure where each corner block is of dimension m by m. We can also expand out the vector $[x^{*T}y^{*T}]^T$ into a matrix-vector product, $\mathcal{Y}\rho^*$. Let $Z = [\ Z_1^T \quad Z_2^T\ ]^T$. Expression (5.1.3) then becomes

$$\begin{bmatrix} H_{1,1} & H_{1,2} & A_1^T \\ H_{2,1} & H_{2,2} & A_2^T \\ A_1 & A_2 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} A_1^T & Z_1 & 0 \\ A_2^T & Z_2 & 0 \\ 0 & 0 & I \end{bmatrix}}_{\mathcal{Y}} \underbrace{\begin{bmatrix} x^*_{_A} \\ x^*_{_Z} \\ y^* \end{bmatrix}}_{\rho^*} = \begin{bmatrix} b_1 \\ b_2 \\ d \end{bmatrix}. \tag{5.1.4}$$

To maintain symmetry of our system we premultiply (5.1.4) by $\mathcal{Y}^T$. Multiplying out the matrix expression $\mathcal{Y}^T\mathcal{H}\mathcal{Y}$ and simplifying, we obtain the linear system

$$\begin{bmatrix} AHA^T & AHZ & AA^T \\ Z^THA^T & Z^THZ & 0 \\ AA^T & 0 & 0 \end{bmatrix} \begin{bmatrix} x^*_{_A} \\ x^*_{_Z} \\ y^* \end{bmatrix} = \begin{bmatrix} Ab \\ Z^Tb \\ d \end{bmatrix}. \tag{5.1.5}$$

We observe that an $m \times m$ system determines $x^*_{_A}$:

$$AA^T x^*_{_A} = d. \tag{5.1.6}$$

Since A is of full rank, AA^T is symmetric and positive definite. We could solve this system using the Cholesky factorization of AA^T if the dimension of the system is small enough, [41, p. 143]. From (5.1.5), having found $x^*_{_A}$ we can find $x^*_{_Z}$ by solving

$$H_{_{ZZ}} x^*_{_Z} = -b_{_Z}, \tag{5.1.7}$$

where

$$H_{ZZ} = Z^T H Z, \qquad b_Z = Z^T(HA^T x_A^* - b).$$

The matrix H is symmetric and positive definite in the nullspace of A, hence $Z^T HZ$ is symmetric and positive definite. Anticipating our technique, we can apply the CG method to compute an approximate solution to the system (5.1.7). Substituting this into (5.1.2) will give us an approximate solution for x^*. Using (5.1.5) with (5.1.2) we obtain a system that can be solved to give an approximate solution for y^*:

$$AA^T y^* = A(b - Hx^*). \tag{5.1.8}$$

If we used a Cholesky factorization to find x_A^*, then this same factorization could be employed to solve (5.1.8).

Let us consider the practical application of the CG method to the system (5.1.7). As we noted in Section 4.2, the use of preconditioning can improve the rate of convergence of the CG iteration. Let us assume that a preconditioner W_{ZZ} is given, where W_{ZZ} is a symmetric, positive definite matrix of dimension $n-m$. Let us consider the class of preconditioners of the form $W_{ZZ} = Z^T GZ$, where G is a symmetric matrix such that $Z^T GZ$ is positive definite. The PCG method applied to the $(n-m)$-dimensional reduced system $H_{ZZ}x_Z^* = -b_Z$, is as given in Algorithm 5.1.1 [41, p. 532]: the arbitrary choice of left preconditioning is used.

Algorithm 5.1.1 Projected PCG for reduced systems

Choose an initial point x_Z^*.
Compute $r_Z = Z^T HZx_Z^* + b_Z$, $g_Z = (Z^T GZ)^{-1} r_Z$ and $p_Z = -g_Z$.
repeat
  $\alpha = {r_Z}^T g_Z / {p_Z}^T Z^T HZp_Z$,
  $x_Z \leftarrow x_Z + \alpha p_Z$,
  $r_Z^+ = r_Z + \alpha Z^T HZp_Z$,
  $g_Z^+ = (Z^T GZ)^{-1} r_Z^+$,
  $\beta = (r_Z^+)^T g_Z^+ / {r_Z}^T g_Z$,
  $p_Z \leftarrow -g_Z^+ + \beta p_Z$,
  $g_Z \leftarrow g_Z^+$,
  $r_Z \leftarrow r_Z^+$,
until a termination step is satisfied.

Gould, Hribar and Nocedal [44] suggest terminating this iteration when ${r_Z}^T (Z^T GZ)^{-1} r_Z$ is sufficiently small. In the next section we modify this algorithm to avoid operating with the nullspace basis Z.

5.1.2 CG method for the full system

Explicit use of Algorithm 5.1.1 would require knowledge of Z and the solution of systems involving W_{ZZ}. The algorithm may, however, be rewritten to make explicit use of a preconditioner which has no need for Z at all. In the following algorithm, the n-vectors x, r, g, p satisfy $x = Zx_{_Z} + A^T x^*_{_A}$, $Z^T r = r_{_Z}$, $g = Zg_{_Z}$, and $p = Zg_{_Z}$. We also define the scaled projection matrix

$$P = Z(Z^T G Z)^{-1} Z^T. \tag{5.1.9}$$

We will later see that P is independent of the choice of nullspace basis Z.

Algorithm 5.1.2 Projected PCG in expanded form

Choose an initial point x satisfying $Ax = d$.
Compute $r = Hx - b$, $g = Pr$, and $p = -g$.
repeat
  $\alpha = r^T g / p^T H p$,
  $x \leftarrow x + \alpha p$,
  $r^+ = r + \alpha H p$,
  $g^+ = P r^+$,
  $\beta = (r^+)^T g^+ / r^T g$,
  $p \leftarrow -g^+ + \beta p$,
  $g \leftarrow g^+$,
  $r \leftarrow r^+$,
until a convergence test is satisfied.

Note that the definition of g^+ via the projection step $g^+ = Pr^+$. Following the terminology of [44], the vector g^+ will be called the *preconditioned residual*. It is defined to be in the nullspace of A. We now wish to be able to apply the projection operator $Z(Z^T G Z)^{-1} Z^T$ without a representation of the nullspace basis Z. From [39, section 5.4.1] we find that if G is nonsingular, then P can be expressed as

$$P = G^{-1}(I - A^T (A G^{-1} A^T)^{-1} A G^{-1}), \tag{5.1.10}$$

and we can find g^+ by solving the system

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} g^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} r^+ \\ 0 \end{bmatrix} \tag{5.1.11}$$

whenever $z^T G z \neq 0$ for all nonzero z for which $Az = 0$. The idea in Algorithm 5.1.3 below is to replace $g^+ = Pr^+$ with the solution of (5.1.11) to define the same g^+.

Discrepancy in the magnitudes of g^+ and r^+ can cause numerical difficulties for which Gould, Hribar and Nocedal [44] suggest using a residual update strategy that redefines r^+ so that its norm is closer to that of g^+. This dramatically reduces the roundoff errors in the projection operation in practice.

Algorithm 5.1.3 Projected PCG with residual update (PPCG)

Choose an initial point x satisfying $Ax = d$
Compute $r = Hx - b$
Solve $\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} g \\ v \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}$
Set $p = -g$, $y = v$ and $r \leftarrow r - A^T y$
repeat
  $\alpha = r^T g / p^T H p$
  $x \leftarrow x + \alpha p$
  $r^+ = r + \alpha H p$
  Solve $\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} g^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} r^+ \\ 0 \end{bmatrix}$
  $\beta = (r^+)^T g^+ / r^T g$
  $p \leftarrow -g^+ + \beta p$
  $g \leftarrow g^+$
  $r \leftarrow r^+ - A^T v^+$
until a convergence test is satisfied

The key point is that Algorithm 5.1.3 does not require the computation of any nullspace basis but is a CG procedure for a symmetric and positive definite system which yields the solution of the indefinite system (5.1.1).

5.2 Constraint Preconditioners

In Algorithm 5.1.3 a preconditioner of the form

$$K = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix},$$

where $G \in \mathbb{R}^{n \times n}$, is required. Such preconditioners are known as *constraint preconditioners* [8, 55]. The term constraint preconditioner was introduced in [55] because the (1,2) and (2,1) matrix blocks of the preconditioner are exactly the same as those in $\mathcal{H}$, Equation (5.1.1), where these blocks represent constraints.

For K to be a meaningful preconditioner for Algorithm 5.1.3, it is vital that its inertia satisfies

$$\text{In}(K) = (n, m, 0), \tag{5.2.1}$$

see [42, Theorem 2.1].

The preconditioned system $K^{-1}\mathcal{H}$ has very specific eigenvalues as given in the following theorem. The proof can be found in [55].

Theorem 5.2.1. *Let $\mathcal{H} \in \mathbb{R}^{(n+m)\times(n+m)}$ be a symmetric and indefinite matrix of the form*

$$\mathcal{H} = \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix},$$

where $H \in \mathbb{R}^{n\times n}$ is symmetric and $A \in \mathbb{R}^{m\times n}$ is of full rank. Assume Z is an $n \times (n-m)$ basis for the nullspace of A. Preconditioning $\mathcal{H}$ by a matrix of the form

$$K = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix},$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric, and $A \in \mathbb{R}^{m\times n}$ is as above, implies that the matrix $K^{-1}\mathcal{H}$ has

1. *an eigenvalue at 1 with multiplicity $2m$;*
2. *$n-m$ eigenvalues λ which are defined by the generalized eigenvalue problem*

$$Z^T H Z x_z = \lambda Z^T G Z x_z. \tag{5.2.2}$$

This accounts for all of the eigenvalues.

Assume, in addition, that $Z^T G Z$ is positive definite. Then $K^{-1}\mathcal{H}$ has the following $m + i + j$ linearly independent eigenvectors:

1. *m eigenvectors of the form $[0^T, y^T]^T$ corresponding to the eigenvalue 1 of $K^{-1}\mathcal{H}$;*
2. *i $(0 \le i \le n)$ eigenvectors of the form $[w^T, y^T]^T$ corresponding to the eigenvalue 1 of $K^{-1}\mathcal{H}$, were the components w arise from the generalized eigenvalue problem $Hw = Gw$;*
3. *j $(0 \le j \le n - m)$ eigenvectors of the form $[x_z^T, 0^T, y^T]^T$ corresponding to the eigenvalues of $K^{-1}\mathcal{H}$ not equal to 1, where the components x_z arise from the generalized eigenvalue problem $Z^T H Z x_z = \lambda Z^T G Z x_z$ with $\lambda \neq 1$.*

From the above theorem we observe that there are at most $n - m + 1$ distinct eigenvalues but we cannot guarantee the eigenvectors of the preconditioned system, $K^{-1}\mathcal{H}$, to form a set of $n+m$ linearly independent eigenvectors. Indeed, it is often the case that $G^{-1}H$ has no unit eigenvalues, so i in the above theorem is zero.

Example 5.2.2 (Minimum bound)**.** Consider the matrices

$$\mathcal{H} = \begin{bmatrix} 1 & 2 & 0 \\ 2 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 3 & 0 \\ 3 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix},$$

so that $m = 1$ and $n = 2$. The preconditioned matrix $K^{-1}\mathcal{H}$ has an eigenvalue at 1 with multiplicity 3, but only one eigenvector arising from case (1) of Theorem 5.2.1. This eigenvector may be taken to be $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

This implies that classical results for the dimension of the Krylov subspace cannot be applied. However, it is possible to show that the nonnormality does not hurt us considerably [55]:

Theorem 5.2.3. *Let $\mathcal{H}, K \in \mathbb{R}^{(n+m)\times(n+m)}$ and their sub-blocks be as defined in Theorem 5.2.1 (using the same notation and assumptions). If Z^TGZ is positive definite, where Z is an $n \times (n-m)$ basis for the nullspace of A, then the dimension of the Krylov subspace $\mathcal{K}(K^{-1}\mathcal{H}, b)$ is at most $n - m + 2$.*

This decrease in the upper bound of the Krylov subspace from $n + m$ to $n-m+2$ can make the difference between an iterative method being practical or impractical to apply. The eigenvalues of (5.2.2) are real since (5.2.1) implies that Z^TGZ is positive definite [14, 42].

Although we are not expecting or requiring that G (or H) be positive definite, it is well known that this is often not a significant handicap.

Theorem 5.2.4 ([4, 20] for example)**.** *The inertial requirement (5.2.1) holds for a given G if and only if there exists a positive semi-definite matrix $\bar{\Delta}$ such that $G + A^T\Delta A$ is positive definite for all Δ for which $\Delta - \bar{\Delta}$ is positive semi-definite.*

Since any preconditioning system

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} r \\ s \end{bmatrix} \tag{5.2.3}$$

may equivalently be written as

$$\begin{bmatrix} G + A^T \Delta A & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} r \\ s \end{bmatrix} \tag{5.2.4}$$

where $w = v - \Delta A u$, there is little to be lost (save sparsity in G) in using (5.2.4), with its positive-definite leading block, rather than (5.2.3). This observation allowed Golub, Greif and Varah [40, 50] to suggest[1] a variety of methods for solving (5.1.1) in the case that H is positive semi-definite, although the scope of their suggestions does not appear fundamentally to be limited to this case. Lukšan and Vlček [56] make related suggestions for more general G. Replacing the original system by the equivalent system (5.2.4) is known as an *augmented Lagrangian* technique.

Note, however, that although Theorem 5.2.4 implies the existence of a suitable Δ, it alas does not provide a suitable value. In [50], the authors propose heuristics to use as few nonzero components of Δ as possible (on sparsity grounds) when G is positive semidefinite, but it is unclear how this extends for general G. Golub, Greif and Varah's methods aim particularly to produce well-conditioned $G + A^T \Delta A$. Notice, though, that perturbations of this form do not change the eigenvalue distribution given by Theorem 5.2.1, since if $H(\Delta_H) = H + A^T \Delta_H A$ and $G(\Delta_G) = G + A^T \Delta_G A$, for (possibly different) Δ_H and Δ_G,

$$Z^T H(\Delta_H) Z = Z^T H Z v = \lambda Z^T G Z v = \lambda Z^T G(\Delta_G) Z v,$$

and thus the generalized eigenvalue problem (5.2.2), and hence eigenvalues of $K(\Delta_G)^{-1}\mathcal{H}(\Delta_H)$, are unaltered.

5.2.1 Improved eigenvalue bounds for the reduced-space basis

Constraint preconditioners are often used with little consideration of the spectral properties of the resulting preconditioned system. In this section we consider several natural choices for constraint preconditioners and analyze the

[1] They actually propose the alternative

$$\begin{bmatrix} G + A^T \Delta A & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} r + A^T \Delta s \\ s \end{bmatrix}$$

although this is not significant.

spectral properties; the aim is to use the results to guide us in our choice of preconditioner. This analysis is new and has recently been published in the work of Dollar, Gould and Wathen [21].

As in Section 3.4 when we considered the fundamental null space basis, we shall suppose that we may partition the columns of A so that

$$A = [A_1 \;\; A_2],$$

and that its leading m by m sub-matrix A_1 and its transpose are easily invertible. We reiterate that since there is considerable flexibility in choosing the "basis" A_1 from the rectangular matrix A by suitable column interchanges, the assumption that A_1 and its transpose are easily invertible is often easily, and sometimes trivially, satisfied. Note that the problem of determining the "sparsest" A_1 is NP hard [16, 17], while numerical considerations must be given to ensure that A_1 is not badly conditioned if at all possible [38]. We shall consider in detail how to choose A_1 in Chapter 8. More generally, we do not necessarily assume that A_1 is sparse or has a sparse factorization, merely that there are effective ways to solve systems involving A_1 and A_1^T. For example, for many problems involving constraints arising from the discretization of partial differential equations, there are highly effective *iterative* methods for such systems [10].

Given our assumption about A_1, we shall be particularly concerned with the *fundamental* basis matrix

$$Z = \begin{bmatrix} R \\ I \end{bmatrix}, \text{ where } R = -A_1^{-1}A_2. \tag{5.2.5}$$

Such basis matrices play vital roles in Simplex (pivoting)-type methods for linear programming [7, 32], and more generally in active set methods for nonlinear optimization [38, 59, 60].

Let us partition G and H such that

$$G = \begin{bmatrix} G_{11} & G_{21}^T \\ G_{21} & G_{22} \end{bmatrix} \quad \text{and} \quad H = \begin{bmatrix} H_{11} & H_{21}^T \\ H_{21} & H_{22} \end{bmatrix}, \tag{5.2.6}$$

where G_{11} and H_{11} are (respectively) the leading m by m sub-matrices of G and H. Equations (5.2.5) and (5.2.6) give

$$\begin{aligned} Z^T G Z &= G_{22} + R^T G_{21}^T + G_{21} R + R^T G_{11} R \\ \text{and} \quad Z^T H Z &= H_{22} + R^T H_{21}^T + H_{21} R + R^T H_{11} R. \end{aligned}$$

To improve the eigenvalue distribution resulting from preconditioning $\mathcal{H}$ by K we consider what happens if we pick G to reproduce certain portions of $\mathcal{H}$.

Let us consider the case where

$$G_{22} = H_{22}, \quad \text{but} \quad G_{11} = 0 \quad \text{and} \quad G_{21} = 0. \tag{5.2.7}$$

Theorem 5.2.5. *Suppose that G and H are as in (5.2.6) and that (5.2.7) holds. Suppose furthermore that H_{22} is positive definite, and let*

$$\rho = \min[\ rank(A_2), rank(H_{21})\ ] + \min[\ rank(A_2), rank(H_{21}) + \min[rank(A_2), rank(H_{11})]\ ].$$

Then $K^{-1}\mathcal{H}$ has at most

$$rank(R^T H_{21}^T + H_{21}R + R^T H_{11}R) + 1 \leq \min(\rho, n-m) + 1 \leq \min(2m, n-m) + 1$$

distinct eigenvalues.

Proof. Now,

$$\begin{aligned} & \operatorname{rank}(Z^T H Z - Z^T G Z) \\ = \ & \operatorname{rank}(R^T H_{21}^T + H_{21}R + R^T H_{11}R) \\ \leq \ & \min\left[\operatorname{rank}(H_{21}R) + \operatorname{rank}(R^T\left(H_{21}^T + H_{11}R\right)), n-m\right], \end{aligned}$$

and

$$\begin{aligned} & \operatorname{rank}(R^T\left(\operatorname{rank}(H_{21}R) + H_{21}^T + H_{11}R\right)) \\ \leq \ & \min\left[\operatorname{rank}(R), \operatorname{rank}(H_{21})\right] + \min\left[\operatorname{rank}(R), \operatorname{rank}(H_{21}^T + H_{11}R)\right] \\ \leq \ & \min\left[\operatorname{rank}(A_2), \operatorname{rank}(H_{21})\right] + \min\left[\operatorname{rank}(A_2), \operatorname{rank}(H_{21}^T) + \min\left[\operatorname{rank}(R), \operatorname{rank}(H_{11})\right]\right] \\ = \ & \rho. \end{aligned}$$

Since $Z^T G Z$ is, by assumption, positive definite, we may write $Z^T G Z = W^T W$ for some nonsingular W. Thus

$$W^{-1} Z^T H Z W^{-T} = I + W^{-1}(R^T H_{21}^T + H_{21}R + R^T H_{11}R)W^{-T}$$

differs from the identity matrix by a matrix of rank at most $\min(\rho, n-m)$, and hence the generalized eigenvalue problem (5.2.2) has at most $\min(\rho, n-m)$ non-unit eigenvalues. □

As we have seen in Theorem 5.2.4, the restriction that H_{22} be positive definite is not as severe as it first might seem, particularly if we entertain the possibility of using positive definite $H_{22} + A_2^T \Delta A_2$ instead.

The eigenvalue situation may be improved if we consider the case

$$G_{11} = H_{11} \quad \text{and} \quad G_{22} = H_{22}, \quad \text{but} \quad G_{21} = 0. \tag{5.2.8}$$

Theorem 5.2.6. *Suppose that G and H are as in (5.2.6) and that (5.2.8) holds. Suppose furthermore that $H_{22} + R^T H_{11}^T R$ is positive definite, and that*

$$\nu = 2 \min [\ \mathit{rank}(A_2), \mathit{rank}(H_{21})\ ].$$

Then $K^{-1}\mathcal{H}$ has at most

$$\mathit{rank}(R^T H_{21}^T + H_{21} R) + 1 \leq \nu + 1 \leq \min(2m, n-m) + 1$$

distinct eigenvalues.

Proof. The result follows as in the proof of Theorem 5.2.5 since now $Z^T H Z - Z^T G Z = R^T H_{21}^T + H_{21} R$ is of rank at most ν. □

Similarly when

$$G_{21} = H_{21} \quad \text{and} \quad G_{22} = H_{22}, \quad \text{but} \quad G_{11} = 0. \tag{5.2.9}$$

Theorem 5.2.7. *Suppose that G and H are as in (5.2.6) and that (5.2.9) holds. Suppose furthermore that $H_{22} + R^T H_{21}^T + H_{21} R$ is positive definite, and that*

$$\mu = \min [\ \mathit{rank}(A_2), \mathit{rank}(H_{11})\ ].$$

Then $K^{-1}\mathcal{H}$ has at most

$$\mathit{rank}(R^T H_{11} R) + 1 \leq \mu + 1 \leq \min(m, n-m) + 1$$

distinct eigenvalues.

Proof. The result follows as before since now $Z^T H Z - Z^T G Z = R^T H_{11} R$ is of rank at most μ. □

In Tables 5.1 and 5.2 we illustrate these results by considering a subset of linear and quadratic programming examples from the Netlib [35] and CUTEr [47] test sets. Tables containing results for the complete set of test problems from Netlib and CUTEr can be found in Appendix A.

All inequality constraints have been converted to equations by adding slack variables, and a suitable "barrier" penalty term (in this case, 1.0) is added to the diagonal of H for each bounded or slack variable to simulate systems that might arise during an iteration of an interior point method for such problems, see Chapter 2.

Given A, a suitable basis matrix A_1 can be found by finding a sparse LU factorization of A^T using the HSL [53] packages `MA48` and `MA51` [25]. An attempt to correctly identify rank is controlled by tight threshold column pivoting, in which any pivot may not be smaller than a factor $\tau = 2$ of the largest entry in its (uneliminated) column [38]. The rank is estimated as the number of pivots, $\rho(A)$, completed before the remaining uneliminated submatrix is judged to be numerically zero, and the indices of the $\rho(A)$ pivotal rows and columns of A define A_1—if $\rho(A) < m$, the remaining rows of A are judged to be dependent, and are discarded[2]. Although such a strategy may not be as robust as, say, a singular value decomposition or a QR factorization with pivoting, both our and others' experience [38] indicate it to be remarkably reliable and successful in practice. Further discussion on the computation of a stable fundamental basis can be found in Chapter 8.

Having found A_1, the factors are discarded, and a fresh LU decomposition of A_1, with a looser threshold column pivoting factor $\tau = 100$, is computed in order to try to encourage sparse factors. All other estimates of rank in Tables 5.1 and 5.2 are obtained in the same way. The columns headed "iteration bounds" illustrate Theorems 5.2.1 ("any G"), 5.2.5 ("exact H_{22}") and 5.2.7 ("exact H_{22} & H_{21}"). Note that in the linear programming case, $H_{21} \equiv 0$, so we have omitted the "exact H_{22}" statistics from Tables 5.1, since these would be identical to those reported as "exact H_{22} & H_{21}".

We observe that in some cases there are useful gains to be made from trying to reproduce H_{22} and, less often, H_{21}. Moreover, the upper bounds on rank obtained in Theorems 5.2.5 and 5.2.7 can be significantly larger than even the estimates $\rho + 1$ and $\mu + 1$ of the number of distinct eigenvalues. However

[2]Note that if this happens, the right-hand inequalities in Theorems 5.2.5–5.2.7 will depend on $n - \text{rank}(A)$ not $n - m$.

the trend is far from uniform, and in some cases there is little or no apparent advantage to be gained from reproducing portions of H. We will carry out numerical tests with the projected preconditioned conjugate gradient method in the following chapter.

Table 5.1: NETLIB LP problems

							iteration bound		
			rank				any G	exact H_{22} & H_{21}	
name	n	m	A	A_2	H_{11}	H_{12}		$\mu+1$	upper
80BAU3B	12061	2262	2262	2231	2262	0	9800	2232	2263
BLEND	114	74	74	37	74	0	41	38	41
D6CUBE	6184	415	404	403	404	0	5781	404	416
FIT2P	13525	3000	3000	3000	3000	0	10526	3001	3001
GROW7	301	140	140	140	140	0	162	141	141
MAROS-R7	9408	3136	3136	3136	3136	0	6273	3137	3137
MODEL	1557	38	38	11	38	0	1520	12	39
PILOT4	1123	410	410	367	333	0	714	334	411
QAP15	22275	6330	6285	5632	6285	0	15991	5633	6331
SCSD1	760	77	77	77	77	0	684	78	78
SIPOW2	2002	2000	2000	2	1999	0	3	3	3
WOODW	8418	1098	1098	1098	1098	0	7321	1099	1099

Table 5.2: CUTEr QP problems

							iteration bound				
			rank				any G	exact H_{22}		exact H_{22} & H_{21}	
name	n	m	A	A_2	H_{11}	H_{12}		$\rho+1$	upper	$\mu+1$	upper
AUG2DCQP	20200	10000	10000	10000	10000	0	10201	10001	10201	10001	10001
BLOCKQP1	10011	5001	5001	5001	5001	5000	5011	5011	5011	5002	5002
CONT-300	90597	90298	90298	299	90298	0	300	300	300	300	300
CVXQP1	10000	5000	5000	2000	5000	2000	5001	4001	5001	2001	5001
KSIP	1021	1001	1001	20	1001	0	21	21	21	21	21
PRIMAL1	410	85	85	85	85	0	326	86	171	86	86
STCQP2	8193	4095	4095	0	4095	1191	4099	1	4099	1	4096
UBH1	9009	6000	6000	3003	6	0	3010	7	3010	7	3010

5.3 Projected Conjugate Gradient Method for the Positive Semidefinite C

We would like to extend the ideas of Sections 5.1 and 5.2 to the more general form of saddle point problem. Given a symmetric n by n matrix H, a second symmetric m by m matrix C and a full rank m by n ($m \leq n$) matrix A, we

are interested in solving structured linear systems of equations

$$\underbrace{\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}, \tag{5.3.1}$$

by iterative methods. There is little loss of generality in assuming the right-hand side of (5.3.1) has the form given rather than with the more general form

$$\underbrace{\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}. \tag{5.3.2}$$

For, so long as we have some mechanism for finding an initial (x_0, y_0) for which $Ax_0 - Cy_0 = d$, linearity of (5.3.1) implies that $(\bar{x}, \bar{y}) = (x_0 - x, y_0 - y)$ solves (5.3.2) when $b = Hx_0 + A^T y_0 - r$.

In Section 2.2 we saw how such systems arise when solving inequality constrained optimization problems. While it would be perfectly possible to apply a general-purpose preconditioned iterative method like GMRES or QMR to (5.3.1), in a similar manner to Section 5.1 when $C = 0$, it is often possible to use the more effective preconditioned conjugate gradient (PCG) method instead. Such a method is already known for the case of C being symmetric and positive definite [43], but we reveal that there is a projected conjugate gradient method which can be used when C is symmetric and positive semi-definite; when $C = 0$ or C is symmetric and positive definite, this new method encompasses the two previously known methods.

Suppose that C is of rank l, and that we find a decomposition

$$C = EDE^T, \tag{5.3.3}$$

where E is m by l and D is l by l and invertible—either a spectral decomposition or an LDL^T factorization with pivoting are suitable, but the exact form is not relevant. In this case, on defining the additional variable $z = -DE^T y$, we may rewrite (5.3.1) as

$$\begin{bmatrix} H & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{bmatrix} \begin{bmatrix} x \\ z \\ y \end{bmatrix} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}. \tag{5.3.4}$$

By noting the trailing zero block in the coefficient matrix of (5.3.4) we observe that the required (x, z) components of the solution lie in the nullspace of $[A \;\; E]$.

Let the columns of the matrix

$$Z = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}$$

form a basis for this nullspace. Then

$$\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} w \tag{5.3.5}$$

for some w, and (5.3.4) implies

$$H_Z w = Z_1^T r, \tag{5.3.6}$$

where

$$H_Z = Z_1^T H Z_1 + Z_2^T D^{-1} Z_2. \tag{5.3.7}$$

Since we would like to apply the PCG method to solve (5.3.6), our fundamental assumption is then that H_Z is positive definite. Fortunately this assumption is often easy to verify. For we have

Theorem 5.3.1. *Suppose that the coefficient matrix $\mathcal{H}$ of (5.3.1) is non-singular and has m_{H-} negative eigenvalues and that C has c_- negative ones. Then H_Z is positive definite if and only if*

$$m_{H-} + c_- = m. \tag{5.3.8}$$

Proof. It is well known [42, Thm. 2.1] that under the assumption that H_Z is positive definite the coefficient matrix E_H of (5.3.4) has inertia $(n + l, m, 0)$. We can use the Schur complement, Section 2.4.2, to factor E_H :

$$E_H = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ AH^{-1} & ED & I \end{bmatrix} \begin{bmatrix} H & 0 & 0 \\ 0 & D^{-1} & 0 \\ 0 & 0 & -\left(AH^{-1}A^T + C\right) \end{bmatrix} \begin{bmatrix} I & 0 & H^{-1}A^T \\ 0 & I & DE^T \\ 0 & 0 & I \end{bmatrix}.$$

The result then follows directly from Sylvester's law of inertia, since then $\mathrm{In}(E_H) = \mathrm{In}(D^{-1}) + \mathrm{In}(\mathcal{H})$ and D^{-1} has as many negative eigenvalues as C has negative eigenvalues by construction (5.3.3). □

Under the assumption that H_Z is positive definite, we may apply the PCG method to (5.3.6) to find w, and hence recover (x, z) from (5.3.5). Notice that

such an approach does not determine y, and additional calculations may need to be performed to recover it if it is required.

More importantly, it has been shown in Section 5.1 and [15, 18, 44, 67] that rather than computing the iterates explicitly within the nullspace via (5.3.5), it is possible to perform the iteration in the original (x, z) space so long as the preconditioner is chosen carefully. Specifically, let G be any symmetric matrix for which the matrix

$$G_Z = Z_1^T G Z_1 + Z_2^T D^{-1} Z_2 \tag{5.3.9}$$

is positive definite, which we can check using Theorem 5.3.1. Then the appropriate projected preconditioned conjugate gradient (PPCG) algorithm is as given in Algorithm 5.3.1 below - in this case y is not required.

Algorithm 5.3.1 Projected Preconditioned Conjugate Gradients (variant 1)

Given $x = 0$, $z = 0$ and $s = 0$

Solve $\begin{bmatrix} G & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{bmatrix} \begin{bmatrix} g \\ h \\ v \end{bmatrix} = \begin{bmatrix} r \\ s \\ 0 \end{bmatrix}$

Set $(p, q) = -(g, h)$ and $\sigma = g^T r + h^T s$

repeat

Form Hp and $D^{-1}q$

$\alpha = \sigma/(p^T Hp + q^T D^{-1} q)$

$x \leftarrow x + \alpha p$

$z \leftarrow z + \alpha q$

$r \leftarrow r + \alpha Hp$

$s \leftarrow s + \alpha D^{-1} q$

Solve $\begin{bmatrix} G & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{bmatrix} \begin{bmatrix} g \\ h \\ v \end{bmatrix} = \begin{bmatrix} r \\ s \\ 0 \end{bmatrix}$

$\sigma_{\text{new}} = g^T r + h^T s$

$\beta = \sigma_{\text{new}}/\sigma$

$\sigma \leftarrow \sigma_{\text{new}}$

$p \leftarrow -g + \beta p$

$q \leftarrow -h + \beta q$

until a termination step is satisfied

The scalar σ in Algorithm 5.3.1 gives an appropriate optimality measure [44], and a realistic termination rule is to stop when σ is small relative to its original value.

While this method is acceptable when a decomposition (5.3.3) of C is known, it is preferable to be able to work directly with C. To this end, suppose that at each iteration

$$s = -E^T a, \quad q = -DE^T d \text{ and } h = -DE^T t$$

for unknown vectors a, d and t—this is clearly the case at the start of the algorithm. Then, letting $w = Ca$, it is straightforward to show that $t = v + a$, and that we can replace our previous algorithm with Algorithm 5.3.2.

Algorithm 5.3.2 Projected Preconditioned Conjugate Gradients (variant 2)

Given $x = 0$, and $a = w = 0$
Solve $\begin{bmatrix} G & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} g \\ v \end{bmatrix} = \begin{bmatrix} r \\ w \end{bmatrix}$
Set $p = -g$, $d = -v$ and $\sigma = g^T r$.
repeat
  Form Hp and Cd
  $\alpha = \sigma/(p^T Hp + d^T Cd)$
  $x \leftarrow x + \alpha p$
  $a \leftarrow a + \alpha d$
  $r \leftarrow r + \alpha Hp$
  $w \leftarrow w + \alpha Cd$
  Solve $\begin{bmatrix} G & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} g \\ v \end{bmatrix} = \begin{bmatrix} r \\ w \end{bmatrix}$
  $t = a + v$
  $\sigma_{\text{new}} = g^T r + t^T w$
  $\beta = \sigma_{\text{new}}/\sigma$
  $\sigma \leftarrow \sigma_{\text{new}}$
  $p \leftarrow -r + \beta p$
  $d \leftarrow -t + \beta d$
until a termination step is satisfied

Notice now that z no longer appears, and that the preconditioning is carried out using the matrix

$$K = \begin{bmatrix} G & A^T \\ A & -C \end{bmatrix}. \tag{5.3.10}$$

Also note that although this variant involves two more vectors than its predecessor, t is simply used as temporary storage and may be omitted if necessary, while w may also be replaced by Ca if storage is tight.

When $C = 0$, this is essentially the algorithm given by [44], but for this case the updates for a, d and w are unnecessary and may be discarded. At the

other extreme, when C is nonsingular the algorithm is precisely that proposed by [43, Alg. 2.3], and is equivalent to applying PCG to the system

$$(H + A^T C^{-1} A)x = r$$

using a preconditioner of the form $G + A^T C^{-1} A$.

Which of the two variants is preferable depends on whether we have a decomposition (5.3.3) and whether l is small relative to m: the vectors h and s in the first variant are of length l, while the corresponding a and d in the second are of length m. Notice also that although the preconditioning steps in the first variant require that we solve

$$\begin{bmatrix} G & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{bmatrix} \begin{bmatrix} g \\ h \\ v \end{bmatrix} = \begin{bmatrix} r \\ s \\ 0 \end{bmatrix}, \tag{5.3.11}$$

this is entirely equivalent to solving

$$\begin{bmatrix} G & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} g \\ v \end{bmatrix} = \begin{bmatrix} r \\ w \end{bmatrix}, \tag{5.3.12}$$

where $w = -EDs$, and recovering

$$h = D(s - E^T v).$$

5.4 Extending Constraint Preconditioners for Positive Semidefinite C

As we have already noted, the term constraint preconditioner was introduced in [55] because the (1,2) and (2,1) matrix blocks of the preconditioner are exact representations of those in $\mathcal{H}$, where these blocks represent constraints. However, we observe that the (2,2) matrix block is also an exact representation when $C = 0$: this motivates us to generalize the term constraint preconditioner to take the form

$$K = \begin{bmatrix} G & A^T \\ A & -C \end{bmatrix}, \tag{5.4.1}$$

where $G \in \mathbb{R}^{n \times n}$ approximates, but is not the same as H. We note that this is the exact form of the preconditioner used in the previous section.

We recall that it is the distribution of the generalized eigenvalues λ for which

$$H_Z\bar{v} = \lambda G_Z\bar{v} \tag{5.4.2}$$

that determines the convergence of the preceding PPCG algorithms, and thus we will be particularly interested in preconditioners which cluster these eigenvalues. We will firstly look at the spectral properties of $K^{-1}\mathcal{H}$ and afterwards focus on the distribution of the eigenvalues of (5.4.2).

5.4.1 Spectral Properties of $K^{-1}\mathcal{H}$

For symmetric (and in general normal) matrix systems, the convergence of an applicable iterative method is determined by the distribution of the eigenvalues of the coefficient matrix. It is often desirable for the number of distinct eigenvalues to be small so that convergence (termination) is guaranteed to occur quickly. For nonnormal systems the convergence is not so readily described, see [49, page 6], [61] and [82].

Theorem 5.4.1. *Let $\mathcal{H} \in \mathbb{R}^{(n+m)\times(n+m)}$ be a symmetric and indefinite matrix of the form*

$$\mathcal{H} = \begin{bmatrix} H & A^T \\ A & -C \end{bmatrix},$$

where $H \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$ are symmetric and $A \in \mathbb{R}^{m\times n}$ is of full rank. Assume that C has rank $l > 0$. Let $Z \in \mathbb{R}^{n\times(n-m)}$ be a basis for the nullspace of A. Preconditioning $\mathcal{H}$ by a matrix of the form

$$K = \begin{bmatrix} G & A^T \\ A & -C \end{bmatrix},$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric, and $A \in \mathbb{R}^{m\times n}$, $C \in \mathbb{R}^{m\times m}$ are as above, implies that the matrix $K^{-1}\mathcal{H}$ has at most $i + j + 1$ distinct eigenvalues which satisfy

- *at least m eigenvalues at 1,*
- *i $(0 \le i \le n - l)$ non-unit eigenvalues that satisfy*

$$\lambda(K^{-1}\mathcal{H}) = \frac{x_Z^T Z^T H Z x_Z}{x_Z^T Z^T G Z x_Z}$$

for some $x_Z \neq 0$,

- *j $(0 \leq j \leq l)$ non-unit eigenvalues that satisfy*

$$\lambda(K^{-1}\mathcal{H}) = \frac{x^T H x + y^T C y}{x^T G x + y^T C y},$$

for some $y \neq 0$, and $Ax = Cy$.

If C is non-singular, then the j $(0 \leq j \leq m)$ non-unit eigenvalues also satisfy

$$\lambda(K^{-1}\mathcal{H}) = \frac{x^T(H + A^T C^{-1} A)x}{x^T(G + A^T C^{-1} A)x}$$

for some $x \neq 0$.

Proof. The eigenvalues of the preconditioned coefficient matrix $K^{-1}\mathcal{H}$ may be derived by considering the generalized eigenvalue problem

$$\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} G & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{5.4.3}$$

Expanding this out we obtain

$$Hx + A^T y = \lambda G x + \lambda A^T y, \tag{5.4.4}$$

and

$$Ax - Cy = \lambda A x - \lambda C y. \tag{5.4.5}$$

Suppose that $\lambda = 1$. Then (5.4.5) will trivially hold, and (5.4.4) implies that

$$Hx = Gx.$$

Hence, for any $y \in \mathbb{R}^m$ and $x = 0$, Equations (5.4.4) and (5.4.5) will hold when $\lambda = 1$. Therefore, there are m linearly independent eigenvectors of the form $[0^T, y^T]^T$ associated with the eigenvalue at 1. This implies that there are at least m eigenvalues at 1.

If $\lambda \neq 1$ and $Cy = 0$, then (5.4.5) implies that $Ax = 0$. Hence, $x \in Null(A)$, so we can write $x = Zx_Z$ for some $x_Z \in \mathbb{R}^{n-m}$. Premultiplying (5.4.4) by x and substituting in $x = Zx_Z$ gives

$$\lambda(K^{-1}\mathcal{H}) = \frac{x_Z^T Z^T H Z x_Z}{x_Z^T Z^T G Z x_Z}. \tag{5.4.6}$$

Therefore, if $Cy = 0$, then there are at most $(n - m) + (m - l)$ linearly independent eigenvectors of the form $[x_Z^T Z^T, y^T]^T$ associated with the non-unit eigenvalues.

If $\lambda \neq 1$ and $Cy \neq 0$, then (5.4.5) implies that $Ax \neq 0$. We can write $x = A^T x_A + Z x_Z$, where $x_A \in \mathbb{R}^m$ and $x_Z \in \mathbb{R}^{n-m}$, since A is of full rank. Equation 5.4.5 implies that $x_A = (AA^T)^{-1}Cy$ and $x = A^T(AA^T)^{-1}Cy + Zx_Z$. Premultiplying (5.4.4) by x^T and substituting in $Ax = Cy$ gives

$$\lambda(K^{-1}\mathcal{H}) = \frac{x^T H x + y^T C y}{x^T G x + y^T C y}. \tag{5.4.7}$$

There are at most l linearly independent eigenvectors of the form $[x_A^T A + x_Z^T Z^T, y^T]^T$, where $Cy \neq 0$, associated with these eigenvalues.

If, in addition, C is non-singular, then $l = m$. Equation (5.4.5) along with the non-singularity of C implies that $y = C^{-1}Ax$. Substituting this into (5.4.7) gives the required result. □

Remark 5.4.2. In the case of symmetric positive definite matrices C and G, Theorem 5.4.1 is not new, see [6] and [9].

We can improve on the above theorem by considering the case of C being symmetric positive definite and the case of C being symmetric positive semi-definite separately. In the latter case we will assume that C is of rank l with $0 < l < m$, and that we can find a decomposition $C = EDE^T$, where E is m by l with orthogonal columns and D is l by l and invertible. A singular value decomposition is suitable for this (but is also clearly not unique).

Theorem 5.4.3. *Assume that $A \in \mathbb{R}^{m\times n}$ $(m \leq n)$ has full rank, $C \in \mathbb{R}^{m\times m}$ is symmetric and positive definite, and $G, H \in \mathbb{R}^{n\times n}$ are symmetric. Let $\mathcal{H}, K \in \mathbb{R}^{(n+m)\times(n+m)}$ be as defined in Theorem 5.4.1. Then the matrix $K^{-1}\mathcal{H}$ has*

- *an eigenvalue at 1 with multiplicity m, and*
- *n eigenvalues which are defined by the generalized eigenvalue problem $(H + A^T C^{-1} A)x = \lambda(G + A^T C^{-1} A)x$.*

This accounts for all of the eigenvalues.

Proof. The eigenvalues of the preconditioned coefficient matrix $K^{-1}\mathcal{H}$ may be derived by considering the generalized eigenvalue problem

$$\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} G & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{5.4.8}$$

Expanding this out we obtain

$$Hx + A^T y = \lambda Gx + \lambda A^T y, \tag{5.4.9}$$

and

$$Ax - Cy = \lambda Ax - \lambda Cy. \tag{5.4.10}$$

Equation (5.4.10) implies that either $\lambda = 1$ or $Ax - Cy = 0$. If the former holds then (5.4.9) becomes

$$Hx = Gx. \tag{5.4.11}$$

Equation (5.4.11) is trivially satisfied by $x = 0$ and, hence, there are m linearly independent eigenvectors of the form $\begin{bmatrix} 0^T & y^T \end{bmatrix}$ associated with the unit eigenvalue. If there exist any $x \neq 0$ which satisfy (5.4.11), then there will be a i $(0 \leq i \leq n)$ linearly independent eigenvectors of the form $\begin{bmatrix} x^T & y^T \end{bmatrix}$ where the components x arise from the generalized eigenvalue problem $Hx = Gx$.

If $\lambda \neq 1$, then (5.4.10) implies that

$$y = C^{-1}Ax.$$

Substituting this into (5.4.9) yields the generalized eigenvalue problem

$$\left(H + A^T C^{-1} A\right) x = \lambda \left(G + A^T C^{-1} A\right) x. \tag{5.4.12}$$

Thus, the non-unit eigenvalues of $K^{-1}\mathcal{H}$ are defined as the non-unit eigenvalues of (5.4.12). Noting that if (5.4.12) has any unit eigenvalues, then the values of $x(\neq 0)$ which satisfy this are exactly those which arise from the generalized eigenvalue problem $Ax = Gx$, we complete our proof. □

If $H + A^T C^{-1} A$ or $G + A^T C^{-1} A$ are positive definite, then the preconditioned system has real eigenvalues.

Theorem 5.4.4. *Assume that $A \in \mathbb{R}^{m\times n}$ $(m \leq n)$ has full rank, $C \in \mathbb{R}^{m\times m}$ is symmetric and positive definite, $G, H \in \mathbb{R}^{n\times n}$ are symmetric and $G + A^T C^{-1} A^T$. Let $\mathcal{H}, K \in \mathbb{R}^{(n+m)\times(n+m)}$ be as defined in Theorem 5.4.1. Then the matrix $K^{-1}\mathcal{H}$ has $n + m$ eigenvalues as defined in Theorem 5.4.3 and $m + i + j$ linearly independent eigenvectors. There are*

- *m eigenvectors of the form $\begin{bmatrix} 0^T & y^T \end{bmatrix}^T$ that correspond to the case $\lambda =$ 1,*

- *i $(0 \leq i \leq n)$ eigenvectors of the form* $\begin{bmatrix} x^T & y^T \end{bmatrix}^T$ *arising from* $Hx = \sigma Gx$ *for which the i vectors are linearly independent,* $\sigma = 1$, *and* $\lambda = 1$, *and*
- *j $(0 \leq j \leq n)$ eigenvectors of the form* $\begin{bmatrix} x^T & y^T \end{bmatrix}^T$ *that correspond to the case* $\lambda \neq 1$.

Proof. The form of the eigenvectors follows directly from the proof of Theorem 5.4.3. It remains for us to show that the $m+i+j$ eigenvectors are linearly independent, that is, we need to show that

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \\ y_1^{(1)} & \cdots & y_m^{(1)} \end{bmatrix} \begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_m^{(1)} \end{bmatrix} + \begin{bmatrix} x_1^{(2)} & \cdots & x_i^{(2)} \\ y_1^{(2)} & \cdots & y_i^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ \vdots \\ a_i^{(2)} \end{bmatrix} + \begin{bmatrix} x_1^{(3)} & \cdots & x_j^{(3)} \\ y_1^{(3)} & \cdots & y_j^{(3)} \end{bmatrix} \begin{bmatrix} a_1^{(3)} \\ \vdots \\ a_j^{(3)} \end{bmatrix} \tag{5.4.13}$$

implies that the vectors $a^{(k)}$ $(k = 1, 2, 3)$ are zero vectors. Multiplying (5.4.13) by $K^{-1}\mathcal{H}$, and recalling that in the previous equation the first matrix arises from the case $\lambda_k = 1$ $(k = 1, \ldots, m)$, the second matrix from the case $\lambda_k = 1$ and $\sigma_k = 1$ $(k = 1, \ldots, i)$, and the last matrix from $\lambda_k \neq 1$ $(k = 1, \ldots, j)$, gives

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \\ y_1^{(1)} & \cdots & y_m^{(1)} \end{bmatrix} \begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_m^{(1)} \end{bmatrix} + \begin{bmatrix} x_1^{(2)} & \cdots & x_i^{(2)} \\ y_1^{(2)} & \cdots & y_i^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ \vdots \\ a_i^{(2)} \end{bmatrix} + \begin{bmatrix} x_1^{(3)} & \cdots & x_j^{(3)} \\ y_1^{(3)} & \cdots & y_j^{(3)} \end{bmatrix} \begin{bmatrix} \lambda_1 a_1^{(3)} \\ \vdots \\ \lambda_j a_j^{(3)} \end{bmatrix}. \tag{5.4.14}$$

Subtracting (5.4.13) from (5.4.14) we obtain

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} x_1^{(3)} & \cdots & x_j^{(3)} \\ y_1^{(3)} & \cdots & y_j^{(3)} \end{bmatrix} \begin{bmatrix} (\lambda_1 - 1)a_1^{(3)} \\ \vdots \\ (\lambda_j - 1)a_j^{(3)} \end{bmatrix}.$$

The assumption that $G + A^T C^{-1} A$ is positive definite implies that $x_k^{(3)}$ $(k = 1, \ldots, j)$ are linearly independent and thus that $(\lambda_k - 1)a_1^{(3)} = 0$, $(k = 1, \ldots, j)$.

The eigenvalues λ_k $(k = 1, \ldots, j)$ are non-unit which implies that $a_k^{(3)} = 0$ $(k = 1, \ldots, j)$. We also have linear independence of $x_k^{(2)}$ $(k = 1, \ldots, i)$ and thus $a_k^{(2)} = 0$ $(k = 1, \ldots, i)$. Equation (5.4.13) simplifies to

$$\begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & \cdots & 0 \\ y_1^{(1)} & \cdots & y_m^{(1)} \end{bmatrix} \begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_m^{(1)} \end{bmatrix}.$$

However, $y_k^{(1)}$ $(k = 1, \ldots, m)$ are linearly independent and thus $a_k^{(1)} = 0$ $(k = 1, \ldots, m)$. □

We shall now consider the case of C being symmetric and positive semidefinite.

Theorem 5.4.5. *Assume that $A \in \mathbb{R}^{m\times n}$ $(m \leq n)$ has full rank, $C \in \mathbb{R}^{m\times m}$ is symmetric and positive semidefinite with rank l where $0 < l < m$, and $G, H \in \mathbb{R}^{n\times n}$ are symmetric. Also assume that C is factored as EDE^T, where $E \in \mathbb{R}^{m\times l}$ and $D \in \mathbb{R}^{l\times l}$ is nonsingular, $F \in \mathbb{R}^{m\times(m-l)}$ is a basis for the nullspace of E^T and $\begin{bmatrix} E & F \end{bmatrix} \in \mathbb{R}^{m\times m}$ is orthogonal. Let the columns of $N \in \mathbb{R}^{n\times(n-m+l)}$ span the nullspace of F^TA, and $\mathcal{H}, K \in \mathbb{R}^{(n+m)\times(n+m)}$ be as defined in Theorem 5.4.1. Then the matrix $K^{-1}\mathcal{H}$ has*

- *an eigenvalue at 1 with multiplicity $2m - l$, and*
- *$n - m + l$ eigenvalues which are defined by the generalized eigenvalue problem $N^T(H + A^TED^{-1}E^TA)Nz = \lambda N(G + A^TED^{-1}E^TA)Nz$.*

This accounts for all of the eigenvalues.

Proof. Any $y \in \mathbb{R}^m$ can be written as $y = Ey_e + Fy_f$. Substituting this into the generalized eigenvalue problem (5.4.8) and premultiplying by $\begin{bmatrix} I & 0 \\ 0 & E^T \\ 0 & F^T \end{bmatrix}$ we obtain

$$\left[\begin{array}{cc|c} H & A^TE & A^TF \\ E^TA & -D & 0 \\ \hline F^TA & 0 & 0 \end{array}\right] \begin{bmatrix} x \\ y_e \\ y_f \end{bmatrix} = \lambda \left[\begin{array}{cc|c} G & A^TE & A^TF \\ E^TA & -D & 0 \\ \hline F^TA & 0 & 0 \end{array}\right] \begin{bmatrix} x \\ y_e \\ y_f \end{bmatrix}. \tag{5.4.15}$$

Noting that the (3,3) block has dimension $(m - l) \times (m - l)$ and is a zero matrix in both coefficient matrices, we can apply Theorem 2.1 from [55] to obtain that $K^{-1}\mathcal{H}$ has

- an eigenvalue at 1 with multiplicity $2(m-l)$, and
- $n-m+2l$ eigenvalues which are defined by the generalized eigenvalue problem

$$\overline{N}^T \begin{bmatrix} H & A^TE \\ E^TA & -D \end{bmatrix} \overline{N} w_n = \lambda \overline{N}^T \begin{bmatrix} G & A^TE \\ E^TA & -D \end{bmatrix} \overline{N} w_n, \qquad (5.4.16)$$

where $\overline{N}$ is an $(n+l)\times(n-m+2l)$ basis for the nullspace of $\begin{bmatrix} F^TA & 0 \end{bmatrix} \in \mathbb{R}^{(m-l)\times(n+l)}$, and

$$\begin{bmatrix} x \\ y_e \end{bmatrix}^T = \overline{N} w_n + \begin{bmatrix} A^TF \\ 0 \end{bmatrix} w_a.$$

Letting $\overline{N} = \begin{bmatrix} N & 0 \\ 0 & I \end{bmatrix}$, then (5.4.16) becomes

$$\begin{bmatrix} N^THN & N^TA^TE \\ E^TAN & -D \end{bmatrix} \begin{bmatrix} w_{n1} \\ w_{n2} \end{bmatrix} = \lambda \begin{bmatrix} N^TGN & N^TA^TE \\ E^TAN & -D \end{bmatrix} \begin{bmatrix} w_{n1} \\ w_{n2} \end{bmatrix}. \quad (5.4.17)$$

This generalized eigenvalue problem is exactly that of the form considered in Theorem 5.4.3, so (5.4.17) has an eigenvalue at 1 with multiplicity l and the remaining eigenvalues are defined by the generalized eigenvalue problem

$$N^T\left(H + A^TED^{-1}E^TA\right)Nw_{n1} = \lambda N^T\left(G + A^TED^{-1}E^TA\right)Nw_{n1}. \quad (5.4.18)$$

Hence, $K^{-1}\mathcal{H}$ has an eigenvalue at one with multiplicity $2m-l$ and the other eigenvalues are defined by the generalized eigenvalue problem (5.4.18). □

If $N^T(H + A^TED^{-1}E^TA)N$ or $N^T(G + A^TED^{-1}E^TA)N$ are positive definite, then the preconditioned system has real eigenvalues.

Theorem 5.4.6. *Assume that $A \in \mathbb{R}^{m\times n}$ $(m \le n)$ has full rank, $C \in \mathbb{R}^{m\times m}$ is symmetric and positive semidefinite with rank l where $0 < l < m$, and $G, H \in \mathbb{R}^{n\times n}$ are symmetric. Also assume that C is factored as EDE^T, where $E \in \mathbb{R}^{m\times l}$ and $D \in \mathbb{R}^{l\times l}$ is nonsingular, $F \in \mathbb{R}^{m\times(m-l)}$ is a basis for the nullspace of E^T and $\begin{bmatrix} E & F \end{bmatrix} \in \mathbb{R}^{m\times m}$ is orthogonal. Let the columns of $N \in \mathbb{R}^{n\times(n-m+l)}$ span the nullspace of F^TA, and $\mathcal{H}, K \in \mathbb{R}^{(n+m)\times(n+m)}$ be as defined in Theorem 5.4.1. Additionally, assume $N^T(G + A^TED^{-1}E^TA)N$ is positive definite. Then the matrix $K^{-1}\mathcal{H}$ has $n+m$ eigenvalues as defined in Theorem 5.4.5 and $m+i+j$ linearly independent eigenvectors. There are*

- *m eigenvectors of the form* $\begin{bmatrix} 0^T & y^T \end{bmatrix}$ *that correspond to the case* $\lambda = 1$,
- i $(0 \leq i \leq n)$ *eigenvectors of the form* $\begin{bmatrix} x^T & y^T \end{bmatrix}$ *arising from* $Hx = \sigma Gx$ *for which the* i *vectors* x *are linearly independent,* $\sigma = 1$, *and* $\lambda = 1$, *and*
- j $(0 \leq j \leq n)$ *eigenvectors of the form* $\begin{bmatrix} x^T & y^T \end{bmatrix}$ *that correspond to the case* $\lambda \neq 1$.

Proof. Proof of the form and linear independence of the $m + i + j$ eigenvalues obtained in a similar manner to the proof of Theorem 5.4.4. □

Remark 5.4.7. The condition that $N^T(G + A^T E D^{-1} E^T A)N$ is positive definite will be hard to verify for general symmetric matrices C and G. However, C has been assumed to be positive semidefinite, so $A^T E D^{-1} E^T A$ is also positive semidefinite. Many simple choices of G will be positive definite, for example, $G = I$. It is then trivial to show that $N^T(G + A^T E D^{-1} E^T A)N$ is positive definite.

Example 5.4.8. The CUTEr test set [47] provides a set of quadratic programming problems. We shall use the problem CVXQP1_S in the following examples. This problem is very small with $n = 100$ and $m = 50$. We shall set $G = \texttt{diag}\{H\}$, $C = \texttt{diag}\{0, \ldots, 0, 1, \ldots, 1\}$ and vary the number of zeros on the diagonal of C so as to change its rank. In Figure 5.1, we illustrate the change in the eigenvalues of the preconditioned system $K^{-1}\mathcal{H}$ for three different choices of C. The eigenvalues are sorted such that

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{n+m}.$$

When $C = 0$, we expect there to be at least $2m$ unit eigenvalues [55]. We observe that our example has exactly $2m$ eigenvalues at 1. From Theorem 5.4.3, when $C = I$, there will be at least m unit eigenvalues. Our example has exactly m unit eigenvalues, Figure 5.1. When C has rank $\frac{m}{2}$, the preconditioned system $K^{-1}\mathcal{H}$ has at least $\frac{3m}{2}$ unit eigenvalues, Theorem 5.4.5. Once again the number of unit eigenvalues for our example is exactly the lower bound given by Theorem 5.4.5. When the rank of C is greater than zero, the largest eigenvalue

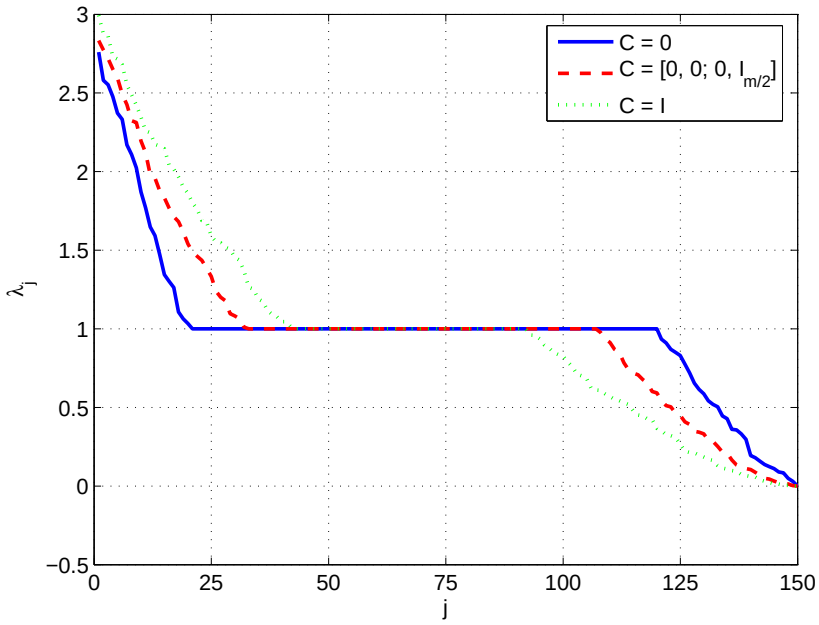

Figure 5.1: Distribution of the eigenvalues of $K^{-1}\mathcal{H}$ for various choices of C.

of $K^{-1}\mathcal{H}$, λ_1, is defined by the generalized eigenvalue problem (5.4.18). As the rank of C varies, this clearly affects the eigenvalues of (5.4.18) and, hence, the largest eigenvalue is changing.

To show that both the lower and upper bounds on the number of linearly independent eigenvectors can be attained we need only consider variations on Examples 2.5 and 2.6 from [55].

Example 5.4.9 (minimum bound)**.** Consider the matrices

$$\mathcal{H} = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 2 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad K = \begin{bmatrix} 1 & 3 & 1 & 0 \\ 3 & 4 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

such that $m = 2$, $n = 2$ and $l = 1$. The preconditioned matrix $K^{-1}\mathcal{H}$ has an eigenvalue at 1 with multiplicity 4, but only two linearly independent eigenvectors which arise from case (1) of Theorem 5.4.6. These eigenvectors may be taken to be $\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T$ and $\begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T$.

Example 5.4.10 (maximum bound)**.** Let $\mathcal{H} \in \mathbb{R}^{3\times 3}$ be as defined in Example 5.4.9, but assume that $G = H$. The preconditioned matrix $K^{-1}\mathcal{H}$ has an eigenvalue at 1 with multiplicity 4 and clearly a complete set of eigenvectors. These may be taken to be the columns of the identity matrix.

In the context of the use of constraint preconditioners, the convergence of an iterative method under preconditioning is not only influenced by the spectral properties of the coefficient matrix, but also by the relationship between m, n and l. We can determine an upper bound on the number of iterations of an appropriate Krylov subspace method by considering minimum polynomials of the coefficient matrix.

Definition 5.4.11. Let $\mathcal{A} \in \mathbb{R}^{(n+m)\times(n+m)}$. The monic polynomial f of minimum degree such that $f(\mathcal{A}) = 0$ is called the *minimum polynomial* of $\mathcal{A}$.

Krylov subspace theory states that iteration with any method with an optimality property, e.g. GMRES, will terminate when the degree of the minimum polynomial is attained [73]. In particular, the degree of the minimum polynomial is equal to the dimension of the corresponding Krylov subspace (for general c) [72, Proposition 6.1]. Again, we shall consider the cases of C symmetric positive definite and C symmetric positive semidefinite separately.

Theorem 5.4.12. *Let the assumptions of Theorem 5.4.4 hold. Then the dimension of the Krylov subspace* $\mathcal{K}(K^{-1}\mathcal{H}, c)$ *is at most* $\min\{n+2, n+m\}$.

Proof. As in the proof of Theorem 5.4.3, the generalized eigenvalue problem is

$$\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} G & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{5.4.19}$$

Suppose that the preconditioned matrix $K^{-1}\mathcal{H}$ takes the form

$$K^{-1}\mathcal{H} = \begin{bmatrix} \Theta_1 & \Theta_3 \\ \Theta_2 & \Theta_4 \end{bmatrix}, \tag{5.4.20}$$

where $\Theta_1 \in \mathbb{R}^{n\times n}$, $\Theta_2 \in \mathbb{R}^{m\times n}$, $\Theta_3 \in \mathbb{R}^{n\times m}$, and $\Theta_4 \in \mathbb{R}^{m\times m}$. Using the facts that $K\left(K^{-1}\mathcal{H}\right) = \mathcal{H}$ and A has full row rank, we obtain $\Theta_3 = 0$ and $\Theta_4 = I$. The precise forms of Θ_1 and Θ_2 are irrelevant for the argument that follows.

From the earlier eigenvalue derivation, it is evident that the characteristic polynomial of the preconditioned linear system (5.4.20) is

$$\left(K^{-1}\mathcal{H} - I\right)^m \prod_{i=1}^{n} \left(K^{-1}\mathcal{H} - \lambda_i I\right).$$

In order to prove the upper bound on the Krylov subspace dimension, we need to show that the order of the minimum polynomial is less than or equal to $\min\{n+2, n+m\}$. Expanding the polynomial $(K^{-1}\mathcal{H} - I)\prod_{i=1}^{n}(K^{-1}\mathcal{H} - \lambda_i I)$ of degree $n+1$, we obtain

$$\begin{bmatrix} (\Theta_1 - I)\prod_{i=1}^{n}(\Theta_1 - \lambda_i I) & 0 \\ \Theta_2 \prod_{i=1}^{n}(\Theta_1 - \lambda_i I) & 0 \end{bmatrix}.$$

Since Θ_1 has a full set of linearly independent eigenvectors, Θ_1 is diagonalizable. Hence,

$$(\Theta_1 - I)\prod_{i=1}^{n}(\Theta_1 - \lambda_i I) = 0.$$

We therefore obtain

$$\left(K^{-1}\mathcal{H} - I\right)\prod_{i=1}^{n}\left(K^{-1}\mathcal{H} - \lambda_i I\right) = \begin{bmatrix} 0 & 0 \\ \Theta_2 \prod_{i=1}^{n}(\Theta_1 - \lambda_i I) & 0 \end{bmatrix}. \tag{5.4.21}$$

If $\Theta_2 \prod_{i=1}^{n}(\Theta_1 - \lambda_i I) = 0$, then the order of the minimum polynomial of $K^{-1}\mathcal{H}$ is less than or equal to $\min\{n+1, n+m\}$. If $\Theta_2 \prod_{i=1}^{n}(\Theta_1 - \lambda_i I) \neq 0$, then the dimension of $\mathcal{K}(K^{-1}\mathcal{H}, c)$ is at most $\min\{n+2, n+m\}$ since multiplication of (5.4.21) by another factor $(K^{-1}\mathcal{H} - I)$ gives the zero matrix. □

Theorem 5.4.13. *Let the assumptions of Theorem 5.4.6 hold. Then the dimension of the Krylov subspace* $\mathcal{K}(K^{-1}\mathcal{H}, c)$ *is at most* $\min\{n-m+l+2, n+m\}$.

Proof. As in the proof of Theorem 5.4.12, the preconditioned matrix $K^{-1}\mathcal{H}$ takes the form

$$K^{-1}\mathcal{H} = \begin{bmatrix} \Theta_1 & 0 \\ \Theta_2 & I \end{bmatrix}, \tag{5.4.22}$$

where $\Theta_1 \in \mathbb{R}^{n\times n}$, and $\Theta_2 \in \mathbb{R}^{m\times n}$. The precise forms of Θ_1 and Θ_2 are irrelevant for the argument that follows.

From the earlier eigenvalue derivation, it is evident that the characteristic polynomial of the preconditioned linear system (5.4.22) is

$$\left(K^{-1}\mathcal{H} - I\right)^{2m-l} \prod_{i=1}^{n-m+l} \left(K^{-1}\mathcal{H} - \lambda_i I\right).$$

In order to prove the upper bound on the Krylov subspace dimension, we need to show that the order of the minimum polynomial is less than or equal to $\min\{n-m+l+2, n+m\}$. Expanding the polynomial

$$\left(K^{-1}\mathcal{H} - I\right) \prod_{i=1}^{n-m+l} \left(K^{-1}\mathcal{H} - \lambda_i I\right)$$

of degree $n-m+l+1$, we obtain

$$\begin{bmatrix} (\Theta_1 - I)\prod_{i=1}^{n-m+l}(\Theta_1 - \lambda_i I) & 0 \\ \Theta_2 \prod_{i=1}^{n-m+l}(\Theta_1 - \lambda_i I) & 0 \end{bmatrix}.$$

Since $G + A^T E D^{-1} E^T A$ is positive definite, Θ_1 has a full set of linearly independent eigenvectors and is diagonalizable. Hence, $(\Theta_1 - I)\prod_{i=1}^{n-m+l}(\Theta_1 - \lambda_i I) = 0$. We therefore obtain

$$\left(K^{-1}\mathcal{H} - I\right) \prod_{i=1}^{n-m+l} \left(K^{-1}\mathcal{H} - \lambda_i I\right) = \begin{bmatrix} 0 & 0 \\ \Theta_2 \prod_{i=1}^{n-m+l}(\Theta_1 - \lambda_i I) & 0 \end{bmatrix}. \quad (5.4.23)$$

If $\Theta_2 \prod_{i=1}^{n-m+l}(\Theta_1 - \lambda_i I) = 0$, then the order of the minimum polynomial of $K^{-1}\mathcal{H}$ is less than or equal to $\min\{n-m+l+1, n+m\}$. If $\Theta_2 \prod_{i=1}^{n-m+l}(\Theta_1 - \lambda_i I) = 0$, then the dimension of $\mathcal{K}\left(K^{-1}\mathcal{H}, c\right)$ is at most $\min\{n-m+l+2, n+m\}$ since multiplication of (5.4.23) by another factor $(K^{-1}\mathcal{H} - I)$ gives the zero matrix.

□

So Theorems 5.2.3, 5.4.12 and 5.4.13 tells us that with preconditioner

$$K = \begin{bmatrix} G & A^T \\ A & -C \end{bmatrix}$$

for

$$\mathcal{H} = \begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}$$

the dimension of the Krylov subspace is no greater than $\min\{n-m+l+2, n+m\}$ (under suitable assumptions). Hence termination (in exact arithmetic) is guaranteed in a number of iterations smaller than this.

5.4.2 Improving the eigenvalue bounds for the reduced-space basis

We recall from (5.4.2) that it is the distribution of the generalized eigenvalues λ for which

$$H_Z\bar{v} = \lambda G_Z\bar{v}$$

that determines the convergence of the Algorithms 5.3.1 and 5.3.2. Suppose that we denote the coefficient matrices of (5.3.4) and (5.3.11) by

$$\overline{\mathcal{H}} = \begin{bmatrix} H & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{bmatrix} \text{ and } \overline{K} = \begin{bmatrix} G & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{bmatrix}$$

respectively. If we recall the definitions (5.3.7) and (5.3.9), the following result is the direct consequence of Theorem 5.2.1.

Corollary 5.4.14. *Suppose that Z is any n by $n-m+l$ basis matrix for the nullspace of $[A\ E]$. Then $\overline{K}^{-1}\overline{\mathcal{H}}$ has $2m$ unit eigenvalues, and the remaining eigenvalues are those of the generalized eigenvalue problem (5.4.2) (or, equivalently, the generalized eigenvalue problem (5.4.18)).*

As in Section 5.2.1, we can improve on Corollary 5.4.14 (that is reduce the upper bound on the number of distinct eigenvalues) by applying Theorems 5.2.5, 5.2.6 and 5.2.7. To do so, let

$$\overline{R} = -A_1^{-1}[A_2\ E],$$

and note that

$$\overline{\mathcal{H}} = \left[\begin{array}{c|cc|c} H_{11} & H_{21}^T & 0 & A_1^T \\ \hline H_{21} & H_{22} & 0 & A_2^T \\ 0 & 0 & D^{-1} & E^T \\ \hline A_1 & A_2 & E & 0 \end{array}\right] \text{ and } \overline{K} = \left[\begin{array}{c|cc|c} G_{11} & G_{21}^T & 0 & A_1^T \\ \hline G_{21} & G_{22} & 0 & A_2^T \\ 0 & 0 & D^{-1} & E^T \\ \hline A_1 & A_2 & E & 0 \end{array}\right].$$

We therefore have the following consequences.

Corollary 5.4.15. *Suppose that G and H are as in (5.2.6) and that (5.2.7) holds. Suppose furthermore that*

$$\begin{bmatrix} H_{22} & 0 \\ 0 & D^{-1} \end{bmatrix} \tag{5.4.24}$$

is positive definite, and let

$$\overline{\rho} = \min\left[\ \eta, \mathit{rank}(H_{21})\ \right] + \min\left[\ \eta, \mathit{rank}(H_{21}) + \min[\eta, \mathit{rank}(H_{11})]\ \right],$$

where $\eta = \mathit{rank}([A_2\ E])$. *Then* $\overline{K}^{-1}\overline{\mathcal{H}}$ *has at most*

$$\mathit{rank}(\overline{R}^T H_{21}^T + H_{21}\overline{R} + \overline{R}^T H_{11}\overline{R}) + 1 \leq \min(\overline{\rho}, n-m+l) + 1 \leq \min(2m, n-m+l) + 1$$

distinct eigenvalues.

Corollary 5.4.16. *Suppose that* G *and* H *are as in (5.2.6) and that (5.2.8) holds. Suppose furthermore that*

$$\begin{bmatrix} H_{22} & 0 \\ 0 & D^{-1} \end{bmatrix} + \overline{R}^T H_{11}^T \overline{R} \tag{5.4.25}$$

is positive definite, and that

$$\overline{\nu} = 2\min\left[\ \eta, \mathit{rank}(H_{21})\ \right],$$

where $\eta = \mathit{rank}([A_2\ E])$. *Then* $\overline{K}^{-1}\overline{\mathcal{H}}$ *has at most*

$$\mathit{rank}(\overline{R}^T H_{21}^T + H_{21}\overline{R}) + 1 \leq \overline{\nu} + 1 \leq \min(2m, n-m+l) + 1$$

distinct eigenvalues.

Corollary 5.4.17. *Suppose that* G *and* H *are as in (5.2.6) and that (5.2.9) holds. Suppose furthermore that*

$$\begin{bmatrix} H_{22} & 0 \\ 0 & D^{-1} \end{bmatrix} + \overline{R}^T H_{21}^T + H_{21}\overline{R} \tag{5.4.26}$$

is positive definite, and that

$$\overline{\mu} = \min\left[\eta, \mathit{rank}(H_{11})\ \right],$$

where $\eta = \mathit{rank}([A_2\ E])$. *Then* $\overline{K}^{-1}\overline{\mathcal{H}}$ *has at most*

$$\mathit{rank}(\overline{R}^T H_{11}\overline{R}) + 1 \leq \overline{\mu} + 1 \leq \min(m, n-m+l) + 1$$

distinct eigenvalues.

As in Section 5.2, the requirements that (5.4.24)—(5.4.26) are positive definite are not as severe as it might first seem, for we have the following corollary to Theorem 5.2.4.

Corollary 5.4.18. *The inertial requirement (5.2.1) holds for a given H if and only if there exists a positive semi-definite matrix $\bar{\Delta}$ such that*

$$\begin{bmatrix} H & 0 \\ 0 & D^{-1} \end{bmatrix} + \begin{bmatrix} A^T \\ E^T \end{bmatrix} \Delta \begin{bmatrix} A & E \end{bmatrix}$$

is positive definite for all Δ for which $\Delta - \bar{\Delta}$ is positive semi-definite. In particular, if (5.2.1) holds, $H + A^T\Delta A$ and $E^T\Delta E + D^{-1}$ are positive definite for all such Δ.

As we did for (5.2.3) and (5.2.4), we can rewrite (5.3.11) as the equivalent

$$\begin{bmatrix} H + A^T\Delta A & A^T\Delta E & A^T \\ E^T\Delta A & E^T\Delta E + D^{-1} & E^T \\ A & E & 0 \end{bmatrix} \begin{bmatrix} x \\ z \\ w \end{bmatrix} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix},$$

where $w = y - \Delta(Ax + Ez) = y - \Delta(Ax - Cy) = y$. Eliminating the variable z, we find that

$$\begin{bmatrix} H + A^T(\Delta - \Delta W\Delta)A & A^TP^T \\ PA & -W \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = -\begin{bmatrix} r \\ 0 \end{bmatrix},$$

where

$$P = I - W\Delta \text{ and } W = E(E^T\Delta E + D^{-1})^{-1}E^T.$$

Hence

$$\begin{bmatrix} H + A^T(\Delta - \Delta W\Delta)A & A^T \\ A & -\overline{C} \end{bmatrix} \begin{bmatrix} x \\ \overline{y} \end{bmatrix} = -\begin{bmatrix} r \\ 0 \end{bmatrix},$$

where

$$\overline{C} = P^{-1}WP^{-T} = (I - W\Delta)^{-1}W(I - \Delta W)^{-1} \text{ and } \overline{y} = P^Ty.$$

Note that $\overline{C}$ is diagonal whenever C is.

In this section and the previous one we have seen how we can apply a projected PCG method to solve saddle point problems of the form given in (5.3.1) and how we can use specific constraint preconditioners to improve the spectral properties of the preconditioned system, and hence improve the rate of convergence of the PPCG method. We will carry out numerical tests with these methods in the following chapters.

5.5 Standard Constraint Preconditioners and Positive Semidefinite C

In our derivation of Algorithm 5.3.1 we firstly added a variable z and then applied the PPCG method for the case $C = 0$ to the system

$$\begin{bmatrix} H & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{bmatrix} \begin{bmatrix} x \\ z \\ y \end{bmatrix} = \begin{bmatrix} r \\ 0 \\ 0 \end{bmatrix}.$$

We chose to apply a preconditioner of the form

$$\begin{bmatrix} G & 0 & A^T \\ 0 & D^{-1} & E^T \\ A & E & 0 \end{bmatrix},$$

but it is possible to relax the preconditioner to take the form

$$\begin{bmatrix} G & 0 & A^T \\ 0 & \widetilde{D}^{-1} & E^T \\ A & E & 0 \end{bmatrix},$$

where $\widetilde{D} \in \mathbb{R}^{l \times l}$ is symmetric, positive definite but does not necessarily satisfy $C = E\widetilde{D}E^T$. The requirement of (5.3.9) being positive definite now corresponds to

$$\widetilde{G}_Z = Z_1^T G Z_1 + Z_2^T \widetilde{D}^{-1} Z_2$$

being positive definite. Suppose that we define

$$\widetilde{D} = \gamma D, \tag{5.5.1}$$

where $\gamma \in (0, 1]$. If G_Z is positive definite and C positive semidefinite, then $\widetilde{G}_Z$ is guaranteed to be positive definite for all $\gamma \in (0, 1]$. Hence, we obtain Algorithm 5.5.1.

As we have already mentioned, we can apply Algorithm 5.5.1 for any $\gamma \in (0, 1]$ when H_Z and G_Z are both assumed to be symmetric, positive definite. In particular, this is true for any positive γ approaching the limit 0. Unfortunately we cannot apply Algorithm 5.5.1 for $\gamma = 0$ because of the requirement of $\frac{1}{\gamma}$ in the preconditioner. However, suppose that, as in Section 5.3, it would be preferable to work directly with C and that at each iteration

$$s = -E^T a, \quad q = -DE^T d \text{ and } h = -DE^T t$$

Algorithm 5.5.1 Projected Preconditioned Conjugate Gradients (variant 3)

Given $x = 0$, $z = 0$ and $s = 0$

Solve $\begin{bmatrix} G & 0 & A^T \\ 0 & \frac{1}{\gamma}D^{-1} & E^T \\ A & E & 0 \end{bmatrix} \begin{bmatrix} g \\ h \\ v \end{bmatrix} = \begin{bmatrix} r \\ s \\ 0 \end{bmatrix}$

Set $(p, q) = -(g, h)$ and $\sigma = g^T r + h^T s$

repeat

Form Hp and $D^{-1}q$

$\alpha = \sigma/(p^T Hp + q^T D^{-1} q)$

$x \leftarrow x + \alpha p$

$z \leftarrow z + \alpha q$

$r \leftarrow r + \alpha Hp$

$s \leftarrow s + \alpha D^{-1} q$

Solve $\begin{bmatrix} G & 0 & A^T \\ 0 & \frac{1}{\gamma}D^{-1} & E^T \\ A & E & 0 \end{bmatrix} \begin{bmatrix} g \\ h \\ v \end{bmatrix} = \begin{bmatrix} r \\ s \\ 0 \end{bmatrix}$

$\sigma_{\text{new}} = g^T r + h^T s$

$\beta = \sigma_{\text{new}}/\sigma$

$\sigma \leftarrow \sigma_{\text{new}}$

$p \leftarrow -r + \beta p$

$q \leftarrow -h + \beta q$

until a termination step is satisfied

for unknown vectors a, d and t—this is clearly the case at the start of the algorithm. Then, letting $w = Ca$, it is straightforward to show that $t = \gamma(v+a)$, and that we can replace our previous algorithm with Algorithm 5.5.2.

Algorithm 5.5.2 Projected Preconditioned Conjugate Gradients (variant 4)

Given $x = 0$, and $a = w = 0$
Solve $\begin{bmatrix} G & A^T \\ A & -\gamma C \end{bmatrix} \begin{bmatrix} g \\ v \end{bmatrix} = \begin{bmatrix} r \\ \gamma w \end{bmatrix}$
Set $p = -g$, $d = -v$ and $\sigma = g^T r$.
repeat
  Form Hp and Cd
  $\alpha = \sigma/(p^T Hp + d^T Cd)$
  $x \leftarrow x + \alpha p$
  $a \leftarrow a + \alpha d$
  $r \leftarrow r + \alpha Hp$
  $w \leftarrow w + \alpha Cd$
  Solve $\begin{bmatrix} G & A^T \\ A & -\gamma C \end{bmatrix} \begin{bmatrix} g \\ v \end{bmatrix} = \begin{bmatrix} r \\ \gamma w \end{bmatrix}$
  $t = \gamma(a + v)$
  $\sigma_{\text{new}} = g^T r + t^T w$
  $\beta = \sigma_{\text{new}}/\sigma$
  $\sigma \leftarrow \sigma_{\text{new}}$
  $p \leftarrow -r + \beta p$
  $d \leftarrow -t + \beta d$
until a termination step is satisfied

Notice that, again, z no longer appears, and that the preconditioning is carried out using the matrix

$$\widetilde{K} = \begin{bmatrix} G & A^T \\ A & -\gamma C \end{bmatrix}. \tag{5.5.2}$$

We can apply this algorithm for all $\gamma \in (0, 1]$ and also the limiting case $\gamma = 0$ because the transformation back to the original variables has removed the singularity. The use of $\gamma = 0$ corresponds to use of a "standard" constraint preconditioner on the system (5.3.4).

In Section 2.2 we noted that if an active set strategy is used along with the described interior point method for solving an inequality constrained optimization problem, then the entries in C converge towards zero, and the entries in the (1,1) block of the saddle point system are also converging towards a matrix with finite real entries. Hence, as we draw near to optimality we might like to apply Algorithm 5.5.2 with $\gamma = 0$ and an unchanging choice of G instead of

Algorithm 5.3.2. Clearly, any factorizations needed for applying the constraint preconditioner would only need to be done once instead of each time that we call Algorithm 5.5.2.

5.5.1 Spectral Properties of $\widetilde{K}^{-1}\mathcal{H}$ for $\gamma = 0$

As we discussed earlier in this chapter, the distribution of the eigenvalues can determine the convergence of an applicable iterative method. We shall use similar methods to those in the previous section proving the following results about the eigenvalues and eigenvectors of the $\widetilde{K}^{-1}\mathcal{H}$ for $\gamma = 0$.

Theorem 5.5.1. *Assume that $A \in \mathbb{R}^{m\times n}$ $(m \leq n)$ has full rank, $C \in \mathbb{R}^{m\times m}$ is symmetric and positive definite, and $G, H \in \mathbb{R}^{n\times n}$ are symmetric. Let $\mathcal{H} \in \mathbb{R}^{(n+m)\times(n+m)}$ be as defined in Theorem 5.4.1 and the preconditioner $\widetilde{K}$ be defined to be a matrix of the form*

$$\widetilde{K} = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix},$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric, and $A \in \mathbb{R}^{m\times n}$ is the same as occurs in $\mathcal{H}$. Then the matrix $\widetilde{K}^{-1}\mathcal{H}$ has $n + m$ eigenvalues which are defined to be the finite eigenvalues of the quadratic eigenvalue problem

$$\lambda^2 A^T C^{-1} A - \lambda \left(G + 2A^T C^{-1} A\right) x + \left(H + A^T C^{-1} A\right) x = 0.$$

Proof. The eigenvalues of the preconditioned coefficient matrix $K^{-1}\mathcal{H}$ may be derived by considering the generalized eigenvalue problem

$$\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{5.5.3}$$

Expanding this out we obtain

$$Hx + A^T y = \lambda G x + \lambda A^T y, \tag{5.5.4}$$

and

$$Ax - Cy = \lambda A x. \tag{5.5.5}$$

If $\lambda = 1$, then (5.5.5) implies that $Cy = 0$. Using the assumption that C is nonsingular we obtain $y = 0$. Equation (5.5.4) x must satisfy $Hx = Gx$, but there is no guarantee that there exists such an x.

Suppose that $\lambda \neq 1$, then rearranging (5.5.5) gives

$$y = (1-\lambda)C^{-1}Ax.$$

Substituting this into (5.5.4) and rearranging we obtain

$$\lambda^2 A^T C^{-1} A x - \lambda \left(G + 2A^T C^{-1} A\right) x + \left(H + A^T C^{-1} C\right) x = 0. \tag{5.5.6}$$

Noting that any unit eigenvalues of $\widetilde{K}^{-1}\mathcal{H}$ will also satisfy this quadratic eigenvalue problem, we complete the proof. □

Theorem 5.5.2. *Assume that $A \in \mathbb{R}^{m\times n}$ $(m \leq n)$ has full rank, $C \in \mathbb{R}^{m\times m}$ is symmetric and positive semidefinite with rank l where $0 < l < m$, and $G, H \in \mathbb{R}^{n\times n}$ are symmetric. Also assume that C is factored as EDE^T, where $E \in \mathbb{R}^{m\times l}$ and $D \in \mathbb{R}^{l\times l}$ is nonsingular, $F \in \mathbb{R}^{m\times(m-l)}$ is a basis for the nullspace of E^T and $\begin{bmatrix} E & F \end{bmatrix} \in \mathbb{R}^{m\times m}$ is orthogonal. Let the columns of $N \in \mathbb{R}^{n\times(n-m+l)}$ span the nullspace of $F^T A$, $\mathcal{H}$ be as defined in Theorem 5.4.1 and the preconditioner $\widetilde{K}$ be defined to be a matrix of the form*

$$\widetilde{K} = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix},$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric, and $A \in \mathbb{R}^{m\times n}$ is the same as occurs in $\mathcal{H}$. Then $\widetilde{K}^{-1}\mathcal{H}$ has

- *an eigenvalue at 1 with multiplicity $2(m-l)$, and*
- *$n - m + 2l$ eigenvalues which are defined as the finite eigenvalues of the quadratic eigenvalue problem*

$$\begin{aligned} 0 \;=\; & \lambda^2 N^T A^T E D^{-1} E^T A N w_{n1} - \lambda N^T \left(G + 2A^T E D^{-1} E^T A\right) N w_{n1} \\ & + N^T \left(H + A^T E D^{-1} E A\right) N w_{n1} = 0. \end{aligned}$$

This accounts for all of the eigenvalues.

Proof. Any $y \in \mathbb{R}^m$ can be written as $y = Ey_e + Fy_f$. Substituting this into the generalized eigenvalue problem (5.5.3) and premultiplying by $\begin{bmatrix} I & 0 \\ 0 & E^T \\ 0 & F^T \end{bmatrix}$ we obtain

$$\left[\begin{array}{cc|c} H & A^TE & A^TF \\ E^TA & -D & 0 \\ \hline F^TA & 0 & 0 \end{array}\right] \begin{bmatrix} x \\ y_e \\ y_f \end{bmatrix} = \lambda \left[\begin{array}{cc|c} G & A^TE & A^TF \\ E^TA & 0 & 0 \\ \hline F^TA & 0 & 0 \end{array}\right] \begin{bmatrix} x \\ y_e \\ y_f \end{bmatrix}. \quad (5.5.7)$$

Noting that the (3,3) block has dimension $(m-l) \times (m-l)$ and is a zero matrix in both coefficient matrices, we can apply Theorem 2.1 from [55] to obtain that $\widetilde{K}^{-1}\mathcal{H}$ has

- an eigenvalue at 1 with multiplicity $2(m-l)$, and
- $n - m + 2l$ eigenvalues which are defined by the generalized eigenvalue problem

$$\overline{N}^T \begin{bmatrix} H & A^TE \\ E^TA & -D \end{bmatrix} \overline{N} w_n = \lambda \overline{N}^T \begin{bmatrix} G & A^TE \\ E^TA & 0 \end{bmatrix} \overline{N} w_n, \quad (5.5.8)$$

where $\overline{N}$ is an $(n+l) \times (n-m+2l)$ basis for the nullspace of $\begin{bmatrix} F^TA & 0 \end{bmatrix} \in \mathbb{R}^{(m-l)\times(n+l)}$, and

$$\begin{bmatrix} x \\ y_e \end{bmatrix} = \begin{bmatrix} A^TF \\ 0 \end{bmatrix} w_a + \overline{N} w_n.$$

Letting $\overline{N} = \begin{bmatrix} N & 0 \\ 0 & I \end{bmatrix}$, then (5.4.16) becomes

$$\begin{bmatrix} N^THN & N^TA^TE \\ E^TAN & -D \end{bmatrix} \begin{bmatrix} w_{n1} \\ w_{n2} \end{bmatrix} = \lambda \begin{bmatrix} N^TGN & N^TA^TE \\ E^TAN & 0 \end{bmatrix} \begin{bmatrix} w_{n1} \\ w_{n2} \end{bmatrix}. \quad (5.5.9)$$

This generalized eigenvalue problem is exactly that of the form considered in Theorem 5.5.1, so the eigenvalues of (5.5.9) are equivalently defined by the quadratic eigenvalue problem

$$\begin{aligned} 0 &= \lambda^2 N^TA^TED^{-1}E^TANw_{n1} - \lambda N^T\left(G + 2A^TED^{-1}E^TA\right)Nw_{n1} \\ &\quad + N^T\left(H + A^TED^{-1}EA\right)Nw_{n1} = 0. \end{aligned} \quad (5.5.10)$$

This defines $2(n-m+l)$ eigenvalues of which $2(n-m+l)-(n-m) = n-m+2l$ are finite [79]. Hence, $\widetilde{K}^{-1}\mathcal{H}$ has an eigenvalue at 1 with multiplicity $2(m-l)$ and the remaining eigenvalues are defined by the quadratic eigenvalue problem (5.5.10). □

Corollary 5.5.3. *Assume that $A \in \mathbb{R}^{m\times n}$ $(m \leq n)$ has full rank, $C \in \mathbb{R}^{m\times m}$ is symmetric and positive semidefinite (possibly nonsingular) with rank l where $0 < l \leq m$, and $G, H \in \mathbb{R}^{n\times n}$ are symmetric. Also assume that C is factored as EDE^T, where $E \in \mathbb{R}^{m\times l}$ and $D \in \mathbb{R}^{l\times l}$ is nonsingular, F is a basis for the nullspace of E^T and $\begin{bmatrix} E & F \end{bmatrix} \in \mathbb{R}^{m\times m}$ is orthogonal. Let the columns of $N \in \mathbb{R}^{n\times(n-m+l)}$ span the nullspace of F^TA, $\mathcal{H}$ be as defined in Theorem 5.4.1 and the preconditioner $\widetilde{K}$ be defined to be a matrix of the form*

$$\widetilde{K} = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix},$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric, and $A \in \mathbb{R}^{m\times n}$ is the same as occurs in $\mathcal{H}$. Then $\widetilde{K}^{-1}\mathcal{H}$ has

- *an eigenvalue at 1 with multiplicity $2(m-l)$, and*
- *$n - m + 2l$ eigenvalues which are defined as the finite eigenvalues of the quadratic eigenvalue problem*

$$\begin{aligned} 0 &= \lambda^2 N^T A^T E D^{-1} E^T A N w_{n1} - \lambda N^T \left(G + 2A^T E D^{-1} E^T A\right) N w_{n1} \\ &\quad + N^T \left(H + A^T E D^{-1} E A\right) N w_{n1}. \end{aligned} \tag{5.5.11}$$

This accounts for all of the eigenvalues.

Proof. For the case of $0 < l < m$ the definition of the eigenvalues trivially holds from Theorem 5.5.2.

Consider the case $l = m$. Then $2(m-l) = 0$, so we need to show that $\widetilde{K}^{-1}\mathcal{H}$ may have no eigenvalues. This is trivially true from Theorem 5.5.1.

If C is nonsingular, then N will also be nonsingular and can be arbitrarily defined as $N = I$. The assumptions imply that E is an orthogonal matrix. Hence,

$$ED^{-1}E^T = E^{-T}D^{-1}E^{-1} = (EDE^T)^{-1} = C^{-1}.$$

Also, from the proof of Theorem 5.5.2, $x = w_{n1}$ when $N = I$. Thus

$$\begin{aligned} 0 &= \lambda^2 N^T A^T E D^{-1} E^T A N w_{n1} - \lambda N^T \left(G + 2A^T E D^{-1} E^T A\right) N w_{n1} \\ &\quad + N^T \left(H + A^T E D^{-1} E A\right) N w_{n1} \\ &= \lambda^2 A^T C^{-1} A N x - \lambda \left(G + 2A^T C^{-1} A\right) x + \left(H + A^T C^{-1} A\right) x. \end{aligned}$$

This is exactly the same quadratic eigenvalue problem as given in Theorem 5.5.1 and, hence, the eigenvalues of $\widetilde{K}^{-1}$ can be equivalently defined by either of the quadratic eigenvalue problems. □

We note that the quadratic eigenvalue problem (5.5.11) can have negative and complex eigenvalues. The following theorem gives sufficient conditions for general quadratic eigenvalue problems to have real and positive eigenvalues and a full set of linearly independent eigenvectors.

Theorem 5.5.4. *Consider the quadratic eigenvalue problem*

$$\left(\lambda^2 K - \lambda L + M\right) x = 0, \tag{5.5.12}$$

where $M, L, \in \mathbb{R}^{n\times n}$ are symmetric positive definite, and $K \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite. Define $\gamma(M, L, K)$ to be

$$\gamma(M, L, K) = \min\left\{(x^T Lx)^2 - 4(x^T Mx)(x^T Kx) : \|x\|_2 = 1\right\}.$$

If $\gamma(M, L, K) > 0$, then the eigenvalues λ are real and positive, and there are n linearly independent eigenvectors associated with the n largest (n smallest) eigenvalues.

Proof. From [79, Section 1] we know that under our assumptions the quadratic eigenvalue problem

$$\left(\mu^2 M + \mu L + K\right) x = 0$$

has real and negative eigenvalues, and that there are n linearly independent eigenvectors associated with the n largest (n smallest) eigenvalues.. Suppose we divide this equation by μ^2 and set $\lambda = -1/\mu$. The quadratic eigenvalue problem (5.5.12) is obtained, and since μ is real and negative, λ is real and positive. □

We would like to be able to use the above theorem to show that, under suitable assumptions, all the eigenvalues of $\widetilde{K}^{-1}\mathcal{H}$ are real and positive. Let

$$\widetilde{D} = N^T A^T E D^{-1} E^T A N, \tag{5.5.13}$$

where D and E are as defined in Corollary 5.5.3. If we assume that $N^THN+\widetilde{D}$ is positive definite, then we may write $N^THN+\widetilde{D} = R^TR$ for some nonsingular matrix R. The quadratic eigenvalue problem (5.5.11) is similar to

$$\left(\lambda^2 R^{-T}\widetilde{D}R^{-1} - \lambda R^{-T}(N^TGN + 2\widetilde{D})R^{-1} + I\right) z = 0,$$

where $z = Rw_{n1}$. Thus, if we assume that $N^TGN + 2\widetilde{D}$ and $N^THN + \widetilde{D}$ are positive definite, and can show that

$$\gamma(I, R^{-T}(N^TGN + 2\widetilde{D})R^{-1}, R^{-T}\widetilde{D}R^{-1}) > 0,$$

where $\gamma(\cdot,\cdot,\cdot)$ is as defined in Theorem 5.5.4, then we can apply the above theorem to show that (5.5.11) has real and positive eigenvalues.

Let us assume that $\|z\|_2 = 1$. Then

$$\begin{aligned}
&\left(z^TR^{-T}\left(N^TGN + 2\widetilde{D}\right)R^{-1}z\right)^2 - 4z^Tzz^TR^{-T}\widetilde{D}R^{-1}z \\
= &\left(z^TR^{-T}N^TGNR^{-1}z + 2z^TR^{-T}\widetilde{D}R^{-1}z\right)^2 - 4z^TR^{-T}\widetilde{D}R^{-1}z \\
= &\left(z^TR^{-T}N^TGNR^{-1}z\right)^2 + 4z^TR^{-T}\widetilde{D}R^{-1}z\left(z^TR^{-T}N^TGNR^{-1}z + z^TR^{-T}\widetilde{D}R^{-1}z - 1\right) \\
= &\left(w_{n1}^TN^TGNw_{n1}\right)^2 + 4w_{n1}^T\widetilde{D}w_{n1}\left(w_{n1}^TN^TGNw_{n1} + w_{n1}^T\widetilde{D}w_{n1} - 1\right), \qquad (5.5.14)
\end{aligned}$$

where $1 = \|z\|_2 = \|Rw_{n1}\|_2 = \|w_{n1}\|_{N^THN+\widetilde{D}}$. Clearly, we can guarantee that (5.5.14) is positive if

$$w_{n1}^TN^TGNw_{n1} + w_{n1}^T\widetilde{D}w_{n1} > 1 \quad \text{for all } w_{n1} \text{ such that } \|w_{n1}\|_{N^THN+\widetilde{D}} = 1,$$

that is

$$\frac{w_{n1}^TN^TGNw_{n1} + w_{n1}^T\widetilde{D}w_{n1}}{w_{n1}^T\left(N^THN + \widetilde{D}\right)w_{n1}} > \frac{w_{n1}^T\left(N^THN + \widetilde{D}\right)w_{n1}}{w_{n1}^T\left(N^THN + \widetilde{D}\right)w_{n1}} \quad \text{for all } w_{n1} \neq 0.$$

Rearranging we find that we require

$$w_{n1}^TN^TGNw_{n1} > w_{n1}^TN^THNw_{n1}$$

for all $Nw_{n1} \neq 0$. Thus we need only scale any positive definite G such that $\frac{w_{n1}^TN^TGNw_{n1}}{w_{n1}^TN^TNw_{n1}} > \|H\|_2^2$ for all $Nw_{n1} \neq 0$ to guarantee that (5.5.14) is positive for all w_{n1} such that $\|w_{n1}\|_{N^THN+\widetilde{D}} = 1$. For example, we could choose $G = \alpha I$, where $\alpha > \|H\|_2^2$.

Using the above in conjunction with Corollary 5.5.3 we obtain:

Theorem 5.5.5. *Suppose that A, C, D, E, G, H and N are as defined in Corollary 5.5.3 and $\widetilde{D}$ is as defined in (5.5.13). Further, assume that $N^T HN + \widetilde{D}$ and $N^T GN + 2\widetilde{D}$ are symmetric positive definite, $\widetilde{D}$ is symmetric positive semidefinite and*

$$\min\left\{(z^T N^T GNz)^2 + 4(z^T\widetilde{D}z)(z^T N^T GNz + z^T\widetilde{D}z - 1) : \|z\|_{N^T HN+\widetilde{D}} = 1\right\} > 0,$$

then all the eigenvalues of $\widetilde{K}^{-1}\mathcal{H}$ are real and positive. The matrix $\widetilde{K}^{-1}\mathcal{H}$ also has $m - l + i + j$ linearly independent eigenvectors. There are

1. *$m - l$ eigenvectors of the form $\begin{bmatrix} 0^T & y_f^T \end{bmatrix}^T$ that correspond to the case $\lambda = 1$,*

2. *i $(0 \le i \le n)$ eigenvectors of the form $\begin{bmatrix} x^T & 0^T & y_f^T \end{bmatrix}^T$ arising from $Hx = \sigma Gx$ for which the i vectors x are linearly independent, $\sigma = 1$, and $\lambda = 1$, and*

3. *j $(0 \le j \le n - m + 2l)$ eigenvectors of the form $\begin{bmatrix} 0^T & w_{n1}^T & w_{n2}^T & y_f^T \end{bmatrix}^T$ corresponding to the eigenvalue of $\widetilde{K}^{-1}\mathcal{H}$ not equal to 1, where the components w_{n1} arise from the quadratic eigenvalue problem*

$$\begin{aligned} 0 &= \lambda^2 N^T A^T E D^{-1} E^T A N w_{n1} - \lambda N^T \left(G + 2A^T E D^{-1} E^T A\right) N w_{n1} \\ &\quad + N^T \left(H + A^T E D^{-1} E A\right) N w_{n1}, \end{aligned}$$

with $\lambda \neq 1$, and $w_{n2} = (1 - \lambda) D^{-1} E^T A N w_{n1}$.

Proof. It remains for us to prove the form of the eigenvectors and that they are linearly independent. We will consider the case $l = m$ and $0 < l < m$ separately.

Consider $l = m$. Then, from the proof of Theorem 5.5.1, when $\lambda = 1$ the eigenvectors must take the form $\begin{bmatrix} x^T & 0^T \end{bmatrix}^T$, where $Hx = \sigma Gx$ for which the i vectors x are linearly independent, $\sigma = 1$. Hence, any eigenvectors corresponding to a unit eigenvalue fall into the second statement of the theorem and there are i $(0 \le i \le n)$ such eigenvectors which are linearly independent. The proof of Theorem 5.5.1 also shows that the eigenvectors corresponding to $\lambda \neq 1$ take the form $\begin{bmatrix} x^T & y^T \end{bmatrix}^T$, where x corresponds to the quadratic eigenvalue problem (5.5.11) and $y = (1 - \lambda) C^{-1} A x = (1 - \lambda) D^{-1} E A N x$ (since we can set $D = C$ and $E = I$). Clearly, there are at most $n + m$ such eigenvectors.

By our assumptions, all of the vectors x defined by the quadratic eigenvalue problem (5.5.11) are linearly independent. Also, if x is associated with two eigenvalues, then these eigenvalues must be distinct [79]. By setting $w_{n1} = x$ and $w_{n2} = y$ we obtain j $(0 \leq j \leq n + m)$ eigenvectors of the form given in statement 3 of the proof.

It remains for us to prove that the $i + j$ eigenvectors defined above are linearly independent. Hence, we need to show that

$$\begin{bmatrix} x_1^{(1)} & \cdots & x_i^{(1)} \\ 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_i^{(1)} \end{bmatrix} + \begin{bmatrix} x_1^{(2)} & \cdots & x_j^{(2)} \\ y_1^{(2)} & \cdots & y_j^{(2)} \end{bmatrix} \begin{bmatrix} a_1^{(2)} \\ \vdots \\ a_j^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \quad (5.5.15)$$

implies that the vectors $a^{(1)}$ and $a^{(2)}$ are zero vectors. Multiplying (5.5.15) by $\widetilde{K}^{-1}\mathcal{H}$, and recalling that in the previous equation the first matrix arises from $\lambda_p = 1$ $(p = 1, \cdots, i)$ and the second matrix from $\lambda_p \neq 1$ $(p = 1, \cdots, j)$ gives

$$\begin{bmatrix} x_1^{(1)} & \cdots & x_i^{(1)} \\ 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} a_1^{(1)} \\ \vdots \\ a_i^{(1)} \end{bmatrix} + \begin{bmatrix} x_1^{(2)} & \cdots & x_j^{(2)} \\ y_1^{(2)} & \cdots & y_j^{(2)} \end{bmatrix} \begin{bmatrix} \lambda_1^{(2)} a_1^{(2)} \\ \vdots \\ \lambda_j^{(2)} a_j^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (5.5.16)$$

Subtracting (5.5.15) from (5.5.16) we obtain

$$\begin{bmatrix} x_1^{(2)} & \cdots & x_j^{(2)} \\ y_1^{(2)} & \cdots & y_j^{(2)} \end{bmatrix} \begin{bmatrix} (\lambda_1^{(2)} - 1)a_1^{(2)} \\ \vdots \\ (\lambda_j^{(2)} - 1)a_j^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (5.5.17)$$

Some of the eigenvectors $x(= w_{n1})$ defined by the quadratic eigenvalue problem (5.5.11) will be associated with two (non-unit) eigenvalues; let us assume that there are k such eigenvectors. By our assumptions, these eigenvalues must be distinct. Without loss of generality, assume that $x_p^{(2)} = x_{k+p}^{(2)}$ for $p = 1, \ldots, k$. The vectors $x_p^{(2)}$ $(p = k + 1, \ldots, j)$ are linearly independent and $\lambda_p^{(2)} \neq 1$ (p=2k+1,...,j), which gives rise to $a_p^{(2)} = 0$ for $p = 2k + 1, \ldots, j$. Equation (5.5.17) becomes

$$\begin{bmatrix} x_1^{(2)} & \cdots & x_k^{(2)} & x_1^{(2)} & \cdots & x_k^{(2)} \\ y_1^{(2)} & \cdots & y_k^{(2)} & y_{k+1}^{(2)} & \cdots & y_{2k}^{(2)} \end{bmatrix} \begin{bmatrix} (\lambda_1^{(2)} - 1)a_1^{(2)} \\ \vdots \\ (\lambda_j^{(2)} - 1)a_2 k^{(2)} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \quad (5.5.18)$$

The vectors $x_p^{(2)}$ $(p = 1, \ldots, k)$ are linearly independent. Hence

$$(\lambda_1^{(2)} - 1)a_p^{(2)} x_p^{(2)} + (\lambda_1^{(2)} - 1)a_{p+k}^{(2)} x_p^{(2)} = 0, \quad p = 1, \ldots, k,$$

and

$$a_p^{(2)} = -a_{p+k}^{(2)} \frac{1-\lambda_{p+k}^{(2)}}{1-\lambda_p^{(2)}}, \quad p = 1, \ldots, k.$$

Now $y_p^{(2)} = (1-\lambda_p^{(2)})C^{-1}Ax_p^{(2)}$ for $p = 1, \ldots, 2k$. Hence, we require

$$(\lambda_1^{(2)}-1)^2 a_p^{(2)} C^{-1}Ax_p^{(2)} + (\lambda_1^{(2)}-1)^2 a_{p+k}^{(2)} C^{-1}Ax_p^{(2)} = 0, \quad p = 1, \ldots, k.$$

Substituting in $a_p^{(2)} = -a_{p+k}^{(2)} \frac{1-\lambda_{p+k}^{(2)}}{1-\lambda_p^{(2)}}$ and rearranging gives $(\lambda_p^{(2)}-1)a_p^{(2)} = (\lambda_{p+k}^{(2)}-1)a_{p+k}^{(2)}$ for $p = 1, \ldots, k$. Since these eigenvalues are non-unit and $\lambda_p^{(2)} \neq \lambda_{p+k}^{(2)}$ for $p = 1, \ldots, k$ we conclude that $a_p^{(2)} = 0$ $(p = 1, \ldots, j)$.

We also have linear independence of $x_p^{(1)}$ $(p = 1, \ldots, i)$, and thus $a_p^{(1)} = 0$ $(p = 1, \ldots, i)$.

Consider the case $0 \leq l \leq m$. From the proof of Theorem 5.5.2, the generalized eigenvalue problem can be expressed as

$$\left[\begin{array}{cc|c} H & A^TE & A^TF \\ E^TA & -D & 0 \\ \hline F^TA & 0 & 0 \end{array}\right] \left[\begin{array}{c} x \\ y_e \\ y_f \end{array}\right] = \lambda \left[\begin{array}{cc|c} G & A^TE & A^TF \\ E^TA & 0 & 0 \\ \hline F^TA & 0 & 0 \end{array}\right] \left[\begin{array}{c} x \\ y_e \\ y_f \end{array}\right].$$

we can apply Theorem 2.3 from [55] to obtain that $\widetilde{K}^{-1}\mathcal{H}$ has the following $m-l+i+j$ linearly independent eigenvectors:

1. $m-l$ eigenvectors of the form $\left[\begin{array}{ccc} 0^T & y_f^T \end{array}\right]^T$ corresponding to the eigenvalue 1 of $\widetilde{K}^{-1}\mathcal{H}$;

2. i $(0 \leq i \leq n+l)$ eigenvectors of the form $\left[\begin{array}{ccc} x^T & y_e^T & y_f^T \end{array}\right]^T$ corresponding to the eigenvalue 1 of $\widetilde{K}^{-1}\mathcal{H}$, where the components x and y_e arise from the generalized eigenvalue problem

$$\left[\begin{array}{cc} H & A^TE \\ E^TA & -D \end{array}\right] \left[\begin{array}{c} x \\ y_e \end{array}\right] = \left[\begin{array}{cc} G & A^TE \\ E^TA & -D \end{array}\right] \left[\begin{array}{c} x \\ y_e \end{array}\right];$$

3. j $(0 \leq j \leq n-m+2l)$ eigenvectors of the form $\left[\begin{array}{ccc} 0^T & w_n^T & y_f^T \end{array}\right]^T$ corresponding to the eigenvalues of $\widetilde{K}^{-1}\mathcal{H}$ not equal to 1, where the components w_n arise from the generalized eigenvalue problem

$$\overline{N}^T \left[\begin{array}{cc} H & A^TE \\ E^TA & -D \end{array}\right] \overline{N}w_n = \lambda \overline{N}^T \left[\begin{array}{cc} G & A^TE \\ E^TA & 0 \end{array}\right] \overline{N}w_n, \qquad (5.5.19)$$

where $\overline{N}$ is an $(n+l) \times (n-m+2l)$ basis for the nullspace of $\left[\begin{array}{cc} F^TA & 0 \end{array}\right] \in \mathbb{R}^{(m-l)\times(n+l)}$, with $\lambda \neq 1$.

Let us consider the i $(0 \leq i \leq n+l)$ eigenvectors defined in the second statement above. If $\lambda = 1$, then $y_e = 0$ since D is nonsingular. Thus, there will be a maximum of n linearly independent eigenvectors taking this form which correspond to the generalized eigenvalue problem $Hx = \sigma Gx$, where $\sigma = 1$. Thus $0 \leq i \leq n$.

Consider the j $(0 \leq j \leq n-m+2l)$ eigenvectors defined in the third statement above. From the proof of Theorem 5.5.2 the eigenvalues are equivalently defined by (5.5.11) and

$$w_n = \begin{bmatrix} w_{n1} \\ (1-\lambda)D^{-1}E^TANw_{n1} \end{bmatrix}.$$

Hence, the j $(0 \leq j \leq n-m+2l)$ eigenvectors corresponding to the non-unit eigenvalues of $\widetilde{K}^{-1}\mathcal{H}$ take the form $\begin{bmatrix} 0^T & w_{n1}^T & w_{n2}^T & y_f^T \end{bmatrix}^T$.

Proof of the linear independence of these eigenvectors follows similarly to the case of $l = m$. □

We discussed in Sections 5.2 and 5.4.1 the fact that the convergence of an iterative method when a constraint preconditioner is used is influenced by the relationship between m, n, and l, as well as the spectral properties of the coefficient matrix. This is also the case for the dimension of the Krylov subspace $\mathcal{K}(\widetilde{K}^{-1}\mathcal{H}, c)$ for all c :

Theorem 5.5.6. *Let the assumptions of Theorem 5.5.5 hold. Then the dimension of the Krylov subspace* $\mathcal{K}(\widetilde{K}^{-1}\mathcal{H}, c)$ *is at most* $\min\{n-m+2l+2, n+m\}$.

Proof. As in the proof to Theorem 5.5.2, the generalized eigenvalue problem can be written as

$$\left[\begin{array}{cc|c} H & A^TE & A^TF \\ E^TA & -D & 0 \\ \hline F^TA & 0 & 0 \end{array}\right] \begin{bmatrix} x \\ y_e \\ y_f \end{bmatrix} = \lambda \left[\begin{array}{cc|c} G & A^TE & A^TF \\ E^TA & 0 & 0 \\ \hline F^TA & 0 & 0 \end{array}\right] \begin{bmatrix} x \\ y_e \\ y_f \end{bmatrix}.$$

Hence, the preconditioned matrix $\widetilde{K}^{-1}\mathcal{H}$ is similar to

$$\widehat{K}^{-1}\widehat{\mathcal{H}} = \begin{bmatrix} \Theta_1 & 0 \\ \Theta_2 & I \end{bmatrix}, \tag{5.5.20}$$

where the precise forms of $\Theta_1 \in \mathbb{R}^{(n+l)\times(n+l)}$ and $\Theta_2 \in \mathbb{R}^{(m-l)\times(n+l)}$ are irrelevant here. The remainder of the proof follows similarly to that of Theorem 5.4.13. □

5.5.2 Clustering of eigenvalues when $\|C\|$ is small

When using interior point methods to solve inequality constrained optimization problems, the matrix C is diagonal and of full rank. In this case, Theorem 5.5.6 would suggest that there is little advantage in using a constraint preconditioner of the form $\widetilde{K}$ over using any other preconditioner. However, when an active set strategy is used, the entries of C also become small as we draw near to optimality and, hence, $\|C\|$ is small. In the following we shall assume that the norm considered is the ℓ_2 norm, but the results could be generalized to other norms.

Lemma 5.5.7. *Let $\zeta > 0$, $\delta \geq 0$, $\varepsilon \geq 0$ and $\delta^2 + 4\zeta(\delta - \varepsilon) \geq 0$. Then the roots of the quadratic equation*

$$\lambda^2\zeta - \lambda(\delta + 2\zeta) + \varepsilon + \zeta = 0$$

satisfy

$$\lambda = 1 + \frac{\delta}{2\zeta} \pm \mu, \quad \mu \leq \sqrt{2}\max\left\{\frac{\delta}{2\zeta}, \sqrt{\frac{|\delta - \varepsilon|}{\zeta}}\right\}.$$

Proof. The roots of the quadratic equation satisfy

$$\begin{aligned}
\lambda &= \frac{\delta + 2\zeta \pm \sqrt{(\delta + 2\zeta)^2 - 4\zeta(\varepsilon + \zeta)}}{2\zeta} \\
&= 1 + \frac{\delta}{2\zeta} \pm \frac{\sqrt{\delta^2 + 4\zeta(\delta - \varepsilon)}}{2\zeta} \\
&= 1 + \frac{\delta}{2\zeta} \pm \sqrt{\left(\frac{\delta}{2\zeta}\right)^2 + \frac{\delta - \varepsilon}{\zeta}}
\end{aligned}$$

If $\frac{\delta-\varepsilon}{\zeta} \geq 0$, then

$$\begin{aligned}
\sqrt{\left(\frac{\delta}{2\zeta}\right)^2 + \frac{\delta - \varepsilon}{\zeta}} &\leq \sqrt{2\max\left\{\left(\frac{\delta}{2\zeta}\right)^2, \frac{\delta - \varepsilon}{\zeta}\right\}} \\
&= \sqrt{2}\max\left\{\frac{\delta}{2\zeta}, \sqrt{\frac{\delta - \varepsilon}{\zeta}}\right\}.
\end{aligned}$$

If $\frac{\delta-\varepsilon}{\zeta} \leq 0$, then the assumption $\delta^2 + 4\zeta(\delta - \varepsilon) \geq 0$ implies that

$$\left(\frac{\delta}{2\zeta}\right)^2 \geq \frac{\varepsilon - \delta}{\zeta} \geq 0.$$

Hence,

$$\begin{aligned}\sqrt{\left(\frac{\delta}{2\zeta}\right)^2+\frac{\delta-\varepsilon}{\zeta}} &\leq \frac{\delta}{2\zeta}\\ &< \sqrt{2}\max\left\{\frac{\delta}{2\zeta},\sqrt{\frac{\varepsilon-\delta}{\zeta}}\right\}.\end{aligned}$$

□

Remark 5.5.8. The important point to notice is that if $\zeta \gg \delta$ and $\zeta \gg \varepsilon$, then $\lambda \approx 1$ in Theorem 5.5.7.

Theorem 5.5.9. *Let $\mathcal{H}, \widehat{K} \in \mathbb{R}^{(n+m)\times(n+m)}$ and their sub-blocks be as defined in Theorem 5.5.5 using the same notation and assumptions. We shall assume that A, G, and H remain fixed, but C may change so long as E also remains fixed. Further, assume that $N^TGN + 2\widetilde{D}$ and $N^THN + \widetilde{D}$ are symmetric positive definite, $\widetilde{D}$ is symmetric positive semidefinite and*

$$\min\left\{(z^TN^TGNz)^2 + 4(z^T\widetilde{D}z)(z^TN^TGNz + z^T\widetilde{D}z - 1) : \|z\|_{N^THN+\widetilde{D}} = 1\right\} > 0.$$

Then all the eigenvalues of $\widetilde{K}^{-1}\mathcal{H}$ are real and positive.

The eigenvalues λ of (5.5.11) subject to $E^TANw_{n1} \neq 0$, will also satisfy

$$|\lambda - 1| \approx c\sqrt{\|C\|} \quad \textit{for small} \quad \|C\|,$$

where c is a constant.

Proof. That the eigenvalues of $\widetilde{K}^{-1}\mathcal{H}$ are real and positive follows directly from Theorem 5.5.5.

Suppose that $C = EDE^T$ is a reduced singular value decomposition of C, where the columns of $E \in \mathbb{R}^{m\times l}$ are orthogonal and $D \in \mathbb{R}^{l\times l}$ is diagonal with entries d_j that are non-negative and in non-increasing order.

In the following, $\|\cdot\| = \|\cdot\|_2$, so that

$$\|C\| = \|D\| = d_1.$$

Premultiplying the quadratic eigenvalue problem (5.5.11) by w_{n1}^T gives

$$\begin{aligned}0 &= \lambda^2 w_{n1}^T\widetilde{D}w_{n1} - \lambda(w_{n1}^TN^TGNw_{n1} + 2w_{n1}^T\widetilde{D}w_{n1})\\ &\quad +(w_{n1}^TN^THNw_{n1} + w_{n1}^T\widetilde{D}w_{n1}).\end{aligned} \tag{5.5.21}$$

Assume that $v = E^T AN w_{n1}$ and $\|v\| = 1$, where w_{n1} is an eigenvalue of the above quadratic eigenvector problem. Then

$$\begin{aligned} w_{n1}^T \widetilde{D} w_{n1} &= v^T D^{-1} v \\ &= \frac{v_1^2}{d_1} + \frac{v_2^2}{d_2} + \ldots + \frac{v_m^2}{d_m} \\ &\geq \frac{v^T v}{d_1} \\ &= \frac{1}{\|C\|}. \end{aligned}$$

Hence,

$$\frac{1}{w_{n1}^T \widetilde{D} w_{n1}} \leq \|C\|.$$

Let $\zeta = w_{n1}^T \widetilde{D} w_{n1}$, $\delta = w_{n1}^T N^T G N w_{n1}$ and $\varepsilon = w_{n1}^T N^T H N w_{n1}$. Then (5.5.21) becomes

$$\lambda^2 \zeta - \lambda(\delta + 2\zeta) + \varepsilon + \zeta = 0.$$

From Lemma 5.5.7, λ must satisfy

$$\lambda = 1 + \frac{\delta}{2\zeta} \pm \mu, \quad \mu \leq \sqrt{2} \max\left\{ \frac{\delta}{2\zeta}, \sqrt{\frac{|\delta - \varepsilon|}{\zeta}} \right\}.$$

Now $\delta \leq c \left\| N^T G N \right\|$, $\varepsilon \leq c \left\| N^T H N \right\|$, where c is an upper bound on $\|w_{n1}\|$ and w_{n1} are eigenvectors of (5.5.11) subject to $E^T AN w_{n1} \neq 0$ and $\left\| E^T AN w_{n1} \right\| = 1$. Hence, the eigenvalues of (5.5.11) subject to $E^T AN w_{n1} \neq 0$ satisfy

$$|\lambda - 1| \approx c\sqrt{\|C\|}$$

for small $\|C\|$, where c is a constant. □

This clustering of part of the spectrum of $\widetilde{K}^{-1}\mathcal{H}$ will often translate into a speeding up of the convergence of a selected Krylov subspace method [5, Section 1.3]. We can therefore see the advantage of Algorithm 5.5.2 when we the interior point method is drawing near to the optimal point.

5.5.3 Numerical Examples

We shall verify our theoretical results by considering some simple saddle point systems.

Example 5.5.10 (C nonsingular)**.** Consider the matrices

$$\mathcal{H} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & -1 \end{bmatrix}, \quad \widetilde{K} = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix},$$

so that $m = l = 1$ and $n = 2$. The preconditioned matrix $\widetilde{K}^{-1}\mathcal{H}$ has eigenvalues at $\frac{1}{2}$, $2 - \sqrt{2}$ and $2 + \sqrt{2}$. The corresponding eigenvectors are $\begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$, $\begin{bmatrix} 1 & 0 & (\sqrt{2}-1) \end{bmatrix}^T$ and $\begin{bmatrix} 1 & 0 & -(\sqrt{2}+1) \end{bmatrix}^T$ respectively. The preconditioned system $\widetilde{K}^{-1}\mathcal{H}$ has all non-unit eigenvalues, but this does contradict Corollary 5.5.3 because $m - l = 0$. With our choices of $\mathcal{H}$ and $\widetilde{K}$, and setting $D = I$ and $E = I$ ($C = EDE^T$), the quadratic eigenvalue problem (5.5.11) is

$$\left(\lambda^2 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix} + \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\right) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0.$$

This quadratic eigenvalue problem has three finite eigenvalues which are $\lambda = 2 - \sqrt{2}$, $\lambda = 2 + \sqrt{2}$ and $\lambda = \frac{1}{2}$.

Example 5.5.11 (C semidefinite)**.** Consider the matrices

$$\mathcal{H} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \quad \widetilde{K} = \begin{bmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

so that $m = 2$, $n = 2$ and $l = 1$. The preconditioned matrix $\widetilde{K}^{-1}\mathcal{H}$ has two unit eigenvalues and a further two at $\lambda = 2 - \sqrt{2}$ and $\lambda = 2 + \sqrt{2}$. There is just one linearly independent eigenvector associated with the unit eigenvector; specifically this is $\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T$. For the non-unit eigenvalues, the eigenvectors are $\begin{bmatrix} 0 & 1 & 0 & (\sqrt{2}-1) \end{bmatrix}^T$ and $\begin{bmatrix} 0 & 1 & 0 & -(\sqrt{2}+1) \end{bmatrix}^T$ respectively.

Since $2(m - l) = 2$, we correctly expected there to be at least two unit eigenvalues, Corollary 5.5.3. The remaining eigenvalues will be defined by the quadratic eigenvalue problem (5.5.11):

$$\left(\lambda^2 - 4\lambda + 2\right) w_{n1} = 0,$$

where $D = [1]$ and $E = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$ are used as factors of C. This quadratic eigenvalue problem has two finite eigenvalues; these are $\lambda = 2 - \sqrt{2}$ and $\lambda = 2 + \sqrt{2}$.

Example 5.5.12 (C with small entries)**.** Suppose that $\mathcal{H}$ and $\widetilde{K}$ are as in Example 5.5.10, but $C = [10^{-a}]$ for some positive real number a. Setting $D = 10^{-a}I$ and $E = I$ ($C = EDE^T$), the quadratic eigenvalue problem (5.5.11) is

$$\left(\lambda^2 \begin{bmatrix} 10^a & 0 \\ 0 & 0 \end{bmatrix} - \lambda \begin{bmatrix} 2 + 2 \times 10^a & 0 \\ 0 & 2 \end{bmatrix} + \begin{bmatrix} 1 + 10^a & 0 \\ 0 & 1 \end{bmatrix}\right) \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0.$$

This quadratic eigenvalue problem has three finite eigenvalues which are defined as $\lambda = 1 + 10^{-a} \pm 10^{-a}\sqrt{1 + 10^a}$ and $\lambda = \frac{1}{2}$. Just the first two eigenvalues satisfy $E^T ANx \neq 0$. For large values of a, $\lambda \approx 1 + 10^{-a} \pm 10^{-\frac{a}{2}}$; these eigenvalues will be close to 1.

We shall use the problem CVXQP2_M from the `CUTEr` test set [47] in the following examples. This problem is very small with $n = 1000$ and $m = 250$. "Barrier" penalty terms (in this case 1.1) are added to the diagonal of H to simulate systems that might arise during and iteration of an interior-point method for such problems. We shall set $G = \texttt{diag}\{H\}$, and $C = \alpha \times \texttt{diag}\{0, \ldots, 0, 1, \ldots, 1\}$, where α is a positive, real parameter that we will change.

All tests carried out in this chapter were performed on a dual Intel Xeon 3.20GHz machine with hyperthreading and 2GiB of RAM. It was running Fedora Core 2 (Linux kernel 2.6.8) with MATLAB® 7.0. We use the projected preconditioned conjugate gradient method, Algorithm 5.5.2, and terminate the iteration when the value of residual is reduced by at least a factor of 10^{-8}.

In Figure 5.2 we compare the performance (in terms of iteration count) between using a preconditioner of the form $\widetilde{K}$ with $\gamma = 0$ and one of the form K, Equations (5.5.2) and (5.3.10) respectively. The matrix C used in this set of results takes the form αI. As α becomes smaller, we expect the difference between the number of iterations required to become less between the two preconditioners. We observe that, in this example, once $\alpha \leq 10^{-1}$ there is little benefit in reproducing C in the preconditioner.

In Figure 5.3 we also compare the performance (in terms of iteration count) between using a preconditioner of the form $\widetilde{K}$ with $\gamma = 0$ and one of the form K, Equations (5.5.2) and (5.3.10) respectively. However, we have now set $C = \alpha \times \texttt{diag}\{0, \ldots, 0, 1, \ldots, 1\}$, where $\text{rank}(C) = \lfloor m/2 \rfloor$. We observe that the convergence is faster in the second figure — this is as we would expect

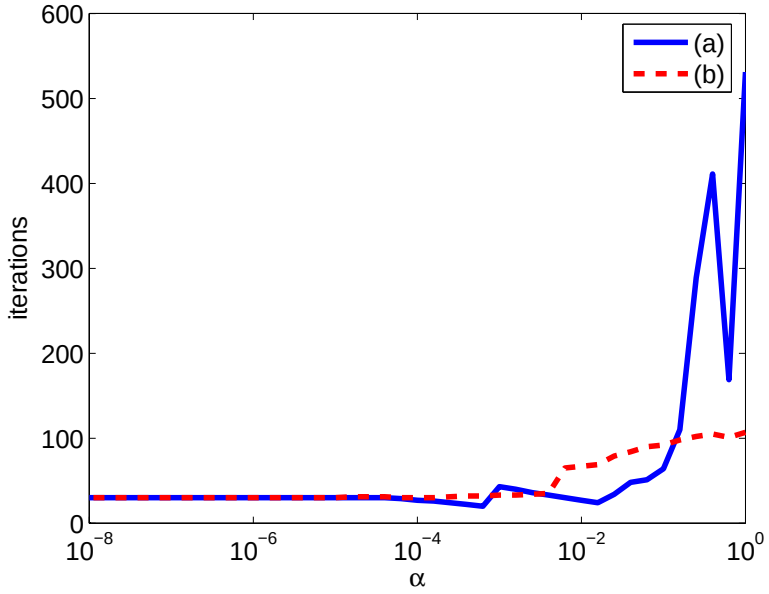

Figure 5.2: Number of PPCG iterations when either (a) $\widetilde{K}$ or (b) K are used as preconditioners for $C = \alpha I$.

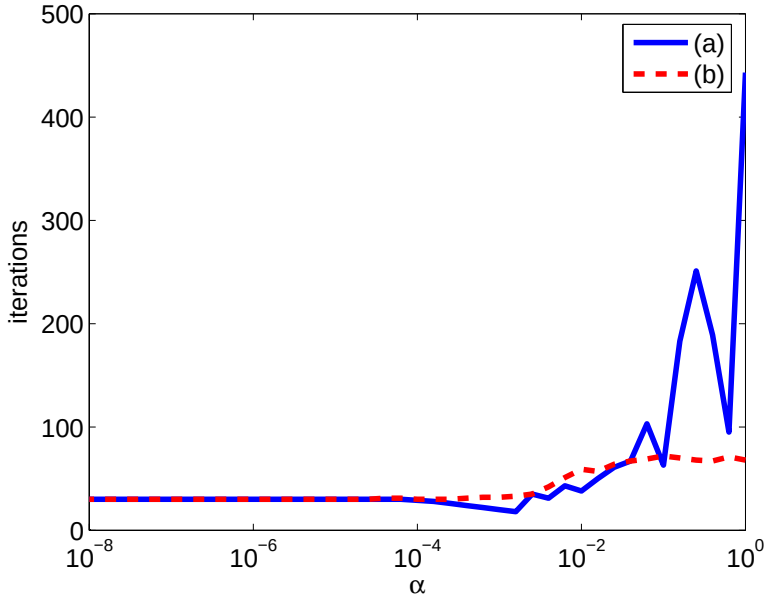

Figure 5.3: Number of PPCG iterations when either (a) $\widetilde{K}$ or (b) K are used as preconditioners for $C = \alpha \times diag(0, \ldots, 0, 1, \ldots, 1)$, where $\text{rank}(C) = \lfloor m/2 \rfloor$.

because of there now being a guarantee of at least 250 unit eigenvalues in the preconditioned system compared to the possibility of none.

5.6 Other possible ways to generate a preconditioned conjugate gradient style method to solve saddle point systems

In the previous sections of the chapter we have shown how we can use projected conjugate gradient methods to solve saddle point systems. Indefinite preconditioners known as constraint preconditioners (or extended constraint preconditioners) were used with these methods. To generate these methods we used a nullspace projection and applied the preconditioned conjugate gradient method to this reduced problem. By changes in variables we were able to expand the method so that the full system instead of this reduced system was used.

In this brief section, we explore another idea for generating preconditioned conjugate gradient methods: this is motivated by the work of Rozložník and Simoncini [70]. They showed that the preconditioned conjugate gradient method can be applied to solve systems of the form

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix},$$

where $H \in \mathbb{R}^{n \times n}$ is symmetric, $A \in \mathbb{R}^{m \times n}$ and a preconditioner of the form

$$K = \begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix}$$

is used. In Section 5.6.2 we will see the importance of the zero block in the right-hand side.

We will be closely following the arguments of Rozložník and Simoncini [70] to show that the preconditioned conjugate gradient method can be applied to solve systems of the form

$$\underbrace{\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix},$$

where $H \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$ are symmetric and positive semidefinite, $A \in \mathbb{R}^{m\times n}$ and a constraint preconditioner of the form

$$K = \begin{bmatrix} G & A^T \\ A & -C \end{bmatrix}$$

is used.

5.6.1 More properties of the constraint preconditioner

In the previous sections we arbitrarily assumed that the constraint preconditioner was applied on the left, but let us consider what happens when we apply it on the right. For simplicity we shall assume that G is symmetric positive definite, but the idea can be extended to symmetric positive semidefinite G by using the augmented Lagrangian ideas at the end of Section 5.4.2.

The eigenvalues of $K^{-1}\mathcal{H}$ and $\mathcal{H}K^{-1}$ are equal, so from Theorem **??** and appropriate assumptions, the eigenvalues of $\mathcal{H}K^{-1}$ are all real and positive. We can use the Schur complement factorization to express the inverse of K :

$$\begin{aligned} K^{-1} &= \begin{bmatrix} I & -G^{-1}A^T \\ 0 & I \end{bmatrix} \begin{bmatrix} G^{-1} & 0 \\ 0 & -S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -AG^{-1} & I \end{bmatrix} \\ &= \begin{bmatrix} G^{-1} - G^{-1}A^TS^{-1}AG^{-1} & G^{-1}A^TS^{-1} \\ S^{-1}AG^{-1} & -S^{-1} \end{bmatrix}, \end{aligned}$$

where $S = C + AG^{-1}A^T$, so that

$$\mathcal{H}K^{-1} = \begin{bmatrix} HG^{-1} + (I - HG^{-1})\, A^TS^{-1}AG^{-1} & (HG^{-1} - I)\, A^TS^{-1} \\ 0 & I \end{bmatrix}.$$

When K^{-1} and $\mathcal{H}K^{-1}$ are applied to a vector of the form $\begin{bmatrix} v^T & 0^T \end{bmatrix}^T$ it follows that

$$K^{-1}\begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} \left(G^{-1} - G^{-1}A^TS^{-1}AG^{-1}\right) v \\ S^{-1}AG^{-1}v \end{bmatrix} \tag{5.6.1}$$

and

$$\mathcal{H}K^{-1}\begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} \left(HG^{-1} + (I - HG^{-1})\, A^TS^{-1}AG^{-1}\right) v \\ 0 \end{bmatrix}. \tag{5.6.2}$$

Remark 5.6.1. It is important to note that the last m entries of $\mathcal{H}K^{-1}w$ are zero if the last m entries of w are zero.

5.6.2 Application of the Bi-CG method and its relationship to PCG

Let us consider the implementation of the simplified Bi-CG as a short recurrence approach. The classical right preconditioned recurrence is as follows:

Given r_0 and $\widetilde{r}_0$.
Set $p_0 = r_0$ and $\widetilde{p}_0 = \widetilde{r}_0$.
for $k = 0, 1, \ldots$ **do**
$\alpha_k = \langle \widetilde{r}_k, r_k \rangle \,/\, \langle \widetilde{p}_k, \mathcal{H}K^{-1}p_k \rangle$,
$t_{k+1} = t_k + \alpha_k p_k$,
$r_{k+1} = r_k - \alpha_k \mathcal{H}K^{-1}p_k$,
$\widetilde{r}_{k+1} = \widetilde{r}_k - \alpha_k K^{-1}\mathcal{H}\widetilde{p}_k$,
$\beta_k = \langle \widetilde{r}_{k+1}, r_{k+1} \rangle \,/\, \langle \widetilde{r}_k, r_k \rangle$,
$p_{k+1} = r_{k+1} + \beta_k p_k$,
$\widetilde{p}_{k+1} = \widetilde{r}_{k+1} + \beta_k \widetilde{p}_k$.
end for

Using J-symmetry[3] with $J = K^{-1}$ and the initial settings of $\widetilde{r}_0 = K^{-1}r_0$ and $r_0 = \begin{bmatrix} s_0^T & 0^T \end{bmatrix}^T$ we obtain that $\widetilde{r}_k = K^{-1}r_k$ for all subsequent $k \geq 1$. Similarly, $\widetilde{p}_k = K^{-1}p_k$ for all subsequent $k \geq 1$. Since we can explicitly compute $\widetilde{p}_k$ and $\widetilde{r}_k$ we can omit the "tilde" recurrence from the method. If $r_0 = \begin{bmatrix} s_0^T & 0^T \end{bmatrix}$, then using (5.6.2) it is straightforward to show that r_{k+1} can be written in the form $r_{k+1} = \begin{bmatrix} s_{k+1}^T & 0^T \end{bmatrix}^T$ for all $k \geq 0$. We can also show that

$$\mathcal{H}\widetilde{p}_{k+1} = \begin{bmatrix} H\widetilde{p}_{k+1}^{(1)} + A^T\widetilde{p}_{k+1}^{(2)} \\ 0 \end{bmatrix}.$$

Hence, we obtain the K^{-1}-symmetric Bi-CG($\mathcal{H}K^{-1}$) algorithm, Algorithm 5.6.1.

If a general choice of r_0 was used, then the values of $\widehat{\alpha}_k$ and $\widehat{\beta}_k$ may be negative. However, fixing the bottom m entries of r_0 (and, consequently for all r_k, $k \geq 1$) to be zero, forces $\widehat{\alpha}_k$ and $\widehat{\beta}_k$ to be positive. We observe that Algorithm 5.6.1 is therefore equivalent to the PCG method for $r_0 = \begin{bmatrix} s_0^T & 0^T \end{bmatrix}^T$, Algorithm 5.6.2, which in itself is an example of Algorithm 5.3.2. However, as far as we are aware, there is no straightforward way to use this form of derivation to reproduce Algorithm 5.5.2.

[3] A matrix H is called J-symmetric if there exists a nonsingular J such that $H^T J = JH$.

Algorithm 5.6.1 K^{-1}-symmetric Bi-CG($\mathcal{H}K^{-1}$) Algorithm

Given $\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$ such that $\begin{bmatrix} b \\ 0 \end{bmatrix} - \mathcal{H}\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} s_0 \\ 0 \end{bmatrix}$.

Set $\begin{bmatrix} u_0 \\ v_0 \end{bmatrix} = \begin{bmatrix} s_0 \\ 0 \end{bmatrix}$.

for $k = 0, 1, \ldots$ **do**

$$\widehat{\alpha}_k = \frac{\left\langle \begin{bmatrix} s_k \\ 0 \end{bmatrix}, K^{-1}\begin{bmatrix} s_k \\ 0 \end{bmatrix} \right\rangle}{\left\langle \mathcal{H}K^{-1}\begin{bmatrix} u_k \\ v_k \end{bmatrix}, K^{-1}\begin{bmatrix} u_k \\ v_k \end{bmatrix} \right\rangle},$$

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \widehat{\alpha}_k K^{-1}\begin{bmatrix} u_k \\ v_k \end{bmatrix},$$

$$\begin{bmatrix} s_{k+1} \\ 0 \end{bmatrix} = \begin{bmatrix} s_k \\ 0 \end{bmatrix} - \widehat{\alpha}_k \mathcal{H}K^{-1}\begin{bmatrix} u_k \\ v_k \end{bmatrix},$$

$$\widehat{\beta}_k = \frac{\left\langle \begin{bmatrix} s_{k+1} \\ 0 \end{bmatrix}, K^{-1}\begin{bmatrix} s_{k+1} \\ 0 \end{bmatrix} \right\rangle}{\left\langle \begin{bmatrix} s_k \\ 0 \end{bmatrix}, K^{-1}\begin{bmatrix} s_k \\ 0 \end{bmatrix} \right\rangle},$$

$$K^{-1}\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = K^{-1}\begin{bmatrix} s_{k+1} \\ 0 \end{bmatrix} + \widehat{\beta}_k K^{-1}\begin{bmatrix} x_k \\ y_k \end{bmatrix}.$$

end for

Algorithm 5.6.2 PCG($\mathcal{H}$) Algorithm

Given $t_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$ such that $r_0 = \begin{bmatrix} b \\ 0 \end{bmatrix} - \mathcal{H}\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} s_0 \\ 0 \end{bmatrix}$.

Set $p_0 = K^{-1}r_0$.

for $k = 0, 1, \ldots$ **do**

$\alpha_k = \frac{\langle r_k, K^{-1}r_k \rangle}{\langle p_k, \mathcal{H}p_k \rangle}$,

$t_{k+1} = t_k + \alpha_k p_k$,

$r_{k+1} = r_k - \alpha_k \mathcal{H}p_k$,

$\beta_k = \frac{\langle r_{k+1}, K^{-1}r_{k+1} \rangle}{\langle r_k, K^{-1}r_k \rangle}$,

$p_{k+1} = K^{-1}r_{k+1} + \beta_k p_k$.

end for

Chapter 6

The Schilders Factorization

In the projected preconditioned conjugate gradient algorithms presented in Chapter 5, we observe the need to solve the preconditioning step

$$\underbrace{\begin{bmatrix} G & A^T \\ A & -C \end{bmatrix}}_{K} \begin{bmatrix} g \\ v \end{bmatrix} = \begin{bmatrix} r \\ w \end{bmatrix}$$

during each iteration. There is no obvious reason why solving such systems should be any easier than solving

$$\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix},$$

the system for which our iterative method is trying to find an approximate solution. In Table 7.5 we see that even simple choices such as $G = I$ can be prohibitive in terms of the memory requirements for sparse direct solution. The standard method for using constraint preconditioners, at least in the optimization community [9, 37, 56], has been to choose G and then factor the resulting constraint preconditioners with available codes such as the `HSL` routines `MA27` and `MA57` [22, 24]. However, we propose an alternative to this: *implicit factorization constraint preconditioners,* which we demonstrate to be very effective as well as being cheap to apply.

Suppose that we consider constraint preconditioners of the form

$$K = PBP^T,$$

where solutions with each of the matrices P, B, and P^T are easily obtained. The key idea is that rather than obtaining P and B from a given K, K is *derived implicitly from specially chosen P and B.*

In this chapter we will consider the case of $C = 0$ and two possible implicit factorization constraint preconditioners: the variable reduction method and the Schilders factorization method. In the following chapter we will consider the case $C \neq 0$.

6.1 Variable Reduction Method

As in Section 5.2.1, let us express the constraint preconditioner in a block 3 by 3 structure:

$$K = \begin{bmatrix} G_{11} & G_{21}^T & A_1^T \\ G_{21} & G_{22} & A_2^T \\ A_1 & A_2 & 0 \end{bmatrix}, \tag{6.1.1}$$

where $G_{11} \in \mathbb{R}^{m\times m}$, $G_{21} \in \mathbb{R}^{(n-m)\times m}$, $G_{22} \in \mathbb{R}^{(n-m)\times(n-m)}$, $A_1 \in \mathbb{R}^{m\times m}$ and $A_2 \in \mathbb{R}^{m\times(n-m)}$. We shall assume that A_1 and its transpose are easily invertible: we shall consider how to reorder A to produce such an A_1 in Chapter 8.

The variable reduction technique (sometimes called the reduced gradient technique) uses an algebraic description of the nullspace method to factor the constraint preconditioner. The fundamental basis is used to define a matrix, Z, whose columns form a basis of the nullspace of A :

$$Z = \begin{bmatrix} R \\ I \end{bmatrix}, \text{ where } R = -A_1^{-1}A_2.$$

In Section 3.4 we saw that a saddle point problem can be reduced to the block triangular form

$$\mathcal{Y}^T K \mathcal{Y} = \begin{bmatrix} Y^TGY & Y^TGZ & Y^TA^T \\ Z^TGY & Z^TGZ & 0 \\ AY & 0 & 0 \end{bmatrix}, \tag{6.1.2}$$

where

$$\mathcal{Y} = \begin{bmatrix} Y & Z & 0 \\ 0 & 0 & I \end{bmatrix}$$

for some $Y \in \mathbb{R}^{n\times m}$ such that $\begin{bmatrix} Y & Z \end{bmatrix}$ spans $\mathbb{R}^n$. Note that when using the fundamental basis we can choose $Y = [I\ 0]^T$. With these choices, the matrix $\mathcal{Y}$ is block triangular so its inverse can also be expressed in a block form:

$$\mathcal{Y}^{-1} = \begin{bmatrix} Y & Z & 0 \\ 0 & 0 & I \end{bmatrix}^{-1} = \begin{bmatrix} I & R & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}^{-1} = \begin{bmatrix} I & -R & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}.$$

We therefore obtain the following factorization for K :

$$\begin{aligned} K &= \begin{bmatrix} I & 0 & 0 \\ -R^T & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} Y^TGY & Y^TGZ & Y^TA^T \\ Z^TGY & Z^TGZ & 0 \\ AY & 0 & 0 \end{bmatrix} \begin{bmatrix} I & -R & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} I & 0 & 0 \\ -R^T & I & 0 \\ 0 & 0 & I \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} G_{11} & D_{21}^T & A_1^T \\ D_{21} & D_{22} & 0 \\ A_1 & 0 & 0 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} I & -R & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix}}_{P^T}, \end{aligned} \tag{6.1.3}$$

where

$$\begin{aligned} D_{21} &= G_{21} + R^TG_{11}, \\ D_{22} &= G_{22} + R^TG_{11}R + G_{21}R + (G_{21}R)^T. \end{aligned}$$

Using this factorization we can solve preconditioning steps of the form

$$\begin{bmatrix} G_{11} & G_{21}^T & A_1^T \\ G_{21} & G_{22} & A_2^T \\ A_1 & A_2 & 0 \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ v \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ w \end{bmatrix} \tag{6.1.4}$$

by carrying out the following steps:

$$\begin{aligned} \psi_1 &= r_1, \\ \psi_2 &= r_2 + R^T\psi_1, \\ \psi_3 &= w, \\ \varphi_1 &= A_1^{-1}\psi_3, \\ \varphi_2 &= D_{22}^{-1}(\psi_2 - D_{21}\varphi_1), \\ \varphi_3 &= A_1^{-T}(\psi_1 - G_{11}\varphi_1 - D_{21}^T\varphi_2), \\ g_2 &= \varphi_2, \\ g_1 &= \varphi_1 + Rg_2, \\ v &= \varphi_3. \end{aligned}$$

This can be simplified to just

$$\begin{aligned} \psi_2 &= r_2 - A_2^TA_1^{-T}r_1, \\ \varphi_1 &= A_1^{-1}w, \\ g_2 &= D_{22}^{-1}\left(\psi_2 - D_{21}\varphi_1\right), \\ v &= A_1^{-T}(r_1 - G_{11}\varphi_1 - D_{21}^T\varphi_2), \\ g_1 &= \varphi_1 - A_1^{-1}A_2g_2. \end{aligned}$$

If we assume that G is chosen and then D_{21} and D_{22} appropriately defined; that all the sub-blocks in the factorization (6.1.3) are dense; and that Gaussian Elimination is used to factor any of the blocks for which systems must be solved, then the number of flops used will be

$$\text{work} = \mathcal{O}\left(\frac{2}{3}(m^3 + (n-m)^3) + 2(i+1)(n^2 + 4mn + 4m^2)\right),$$

where i is the number of iterations carried out by our iterative method. The cost of forming D_{22} has been excluded from this calculation, but it is clearly likely to be expensive in terms of CPU time and memory. We note that, in practice, a symmetric factorization will be used for blocks such as D_{22} and that many blocks will be sparse. This implies that the above expression for the number of flops used by the preconditioning steps could be a lot higher than what happens in reality.

As noted earlier, the standard method for using constraint preconditioners involves the selection of a matrix G and then the factorization of the resulting K. We can choose such a G and then use the factorization (6.1.3) but we observe that even simple choices of G can result in the matrices D_{21} and D_{22} becoming dense. We only need carry out matrix-vector products with D_{21}, so we are not concerned about this matrix being dense because we will not explicitly form D_{21}. However, we need to solve systems involving D_{22} and, hence, we would ideally like to keep D_{22} sparse.

Importantly, Equation (6.1.3) can be used to define a constraint preconditioner implicitly. We choose a G_{11}, D_{21} and D_{22} such that D_{22} is symmetric and nonsingular. We then know that multiplying together the factors will give us a constraint preconditioner. In this way, we can choose D_{22} such that systems involving it can be efficiently solved.

If we assume that G_{11} and D_{21} are allowed to be dense, but D_{22} is diagonal (and nonsingular), then the total work in solving the preconditioning steps when i iterations are required can be expressed as

$$\text{work} = \mathcal{O}\left(\frac{2}{3}m^3 + 2(i+1)(4mn + m^2)\right).$$

We note that this bound in the work is substantially lower than that given earlier when G is chosen and K then factored with (6.1.3).

In the following section we shall propose an alternative implicit factorization constraint preconditioner: the Schilders factorization.

6.2 The Schilders Factorization

The Schilders factorization was originally derived by considering models for electronic circuits. We will start by considering how it was derived and then turn to how it can be used as an implicit factorization constraint preconditioner.

6.2.1 Derivation of the Factorization

We shall firstly restrict ourselves to a special class of matrices A, namely those which have the property

$$A_{i,j} \in \{-1, 0, 1\}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

We shall also assume that each column of A contains at most two nonzero elements which are of opposite sign:

$$\sum_{i=1}^{m} |A_{i,j}| \leq 2,$$

$$-1 \leq \sum_{i=1}^{m} A_{i,j} \leq 1.$$

As previously, we assume that the matrix A has full rank. The matrix A can be considered to be the difference of two incidence matrices.

Let $\Pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ be a permutation with the property that

$$A_{i,\Pi(i)} \neq 0,$$

and

$$A_{j,\Pi(i)} = 0 \quad \text{for} \quad j > i.$$

Hence, A is permuted to an upper trapezoidal form, meaning that the left m by m sub-block of A is upper triangular. It is straight-forward to show that such a permutation may not exist for such matrices A, but it is also relatively easy to show that if a row permutation, $\widehat{\Pi}$, is also carried out, then A can be permuted to be upper trapezoidal. In the following we will assume that $\widehat{\Pi}(i) = i$, but the generalization is straightforward. Let us define a permutation matrix Q by

$$Q = \left[e_{\Pi(1)}, e_{n+1}, \ldots, e_{\Pi(m)}, e_{n+m}, \ldots, e_{\Pi(n)} \right].$$

After the permutation of rows and columns, we obtain the matrix

$$\widetilde{\mathcal{H}} = Q^T \mathcal{H} Q.$$

In order to find a suitable preconditioner, Schilders proposes an incomplete decomposition for the system $\widetilde{\mathcal{H}}$. Let us define the preconditioner, $\widetilde{K}$, to be

$$\widetilde{K} = (\widetilde{L} + \widetilde{D})\widetilde{D}^{-1}(\widetilde{L} + \widetilde{D})^T,$$

where $\widetilde{L}$ and $\widetilde{D}$ take the following block forms:

$$\widetilde{L} = \left[\begin{array}{ccc|ccc} 0 & \cdots & 0 & 0 & \cdots & 0 \\ \widetilde{L}_{2,1} & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & 0 & 0 & \cdots & 0 \\ \hline \widetilde{L}_{m+1,1} & \cdots & \widetilde{L}_{m+1,m} & 0 & \cdots & 0 \\ \vdots & & \vdots & \ddots & \ddots & \vdots \\ \widetilde{L}_{n,1} & \cdots & \widetilde{L}_{n,m-1} & \cdots & \widetilde{L}_{n,n-1} & 0 \end{array}\right], \tag{6.2.1}$$

$$\widetilde{D} = \left[\begin{array}{ccc|ccc} \widetilde{D}_{1,1} & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots & & \vdots \\ \vdots & \ddots & \widetilde{D}_{m,m} & 0 & \cdots & 0 \\ \hline 0 & \cdots & 0 & \widetilde{D}_{m+1,m+1} & \ddots & \vdots \\ \vdots & & \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \cdots & 0 & \widetilde{D}_{n,n} \end{array}\right], \tag{6.2.2}$$

$$L_{i,j} \in \begin{cases} \mathbb{R}^{2\times 2}, & 1 \le j < i \le m, \\ \mathbb{R}^{1\times 2}, & m < i \le n, \quad 1 \le j \le m, \\ \mathbb{R}^{1\times 1}, & m < j < i \le n, \end{cases} \tag{6.2.3}$$

$$D_{i,i} \in \begin{cases} \mathbb{R}^{2\times 2}, & i = 1, \ldots, m, \\ \mathbb{R}^{1\times 1}, & i = m+1, \ldots, n. \end{cases} \tag{6.2.4}$$

The non-zero sub-blocks of $\widetilde{L}$ are defined to be the equal to the associated sub-blocks in $\widetilde{\mathcal{H}}$, i.e., for $1 \le j < i \le m$

$$\widetilde{L}_{i,j} = \begin{bmatrix} H_{\Pi(i),\Pi(j)} & A^T_{\Pi(i),j} \\ A_{i,\Pi(j)} & 0 \end{bmatrix}.$$

We shall say that $\widetilde{L} =$ "lower"$(\widetilde{\mathcal{H}})$.

The matrices $\widetilde{D}_1, \ldots, \widetilde{D}_m$ and scalars $\widetilde{D}_{m+1}, \ldots, \widetilde{D}_n$ are required such that

$$\text{“diag”}\left((\widetilde{L}+\widetilde{D})\widetilde{D}^{-1}(\widetilde{L}+\widetilde{D})^T\right) = \text{“diag”}(\widetilde{\mathcal{H}}), \tag{6.2.5}$$

where “diag” means the block diagonal in terms of the diagonal blocks given in (6.2.2), i.e.,

$$\widetilde{\mathcal{H}} = \text{“diag”}(\widetilde{\mathcal{H}}) + \text{“lower”}(\widetilde{\mathcal{H}}) + \left(\text{“lower”}(\widetilde{\mathcal{H}})\right)^T.$$

Theorem 6.2.1. *Assume that A, Π and Q are as described at the beginning of this section, $\widetilde{L} =$ “lower”$(\widetilde{\mathcal{H}})$ and Equation 6.2.5 holds. Then*

$$Q(\widetilde{L}+\widetilde{D})\widetilde{D}^{-1}(\widetilde{L}+\widetilde{D})^T Q^T = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix},$$

where $G \in \mathbb{R}^{n\times n}$ is symmetric.

Proof. See [74] and [80, Section 3.3]. □

The assumption that A is the difference of two incidence matrices (where both incidence matrices have at most one nonzero in each column) and upper trapezoidal is a big restriction. However, Schilders [75] shows that this can be generalized to any A that is of full rank. In particular, let us use a QR decomposition of A :

$$A\Pi = Q^T\widehat{A},$$

where Π is now an n by n permutation matrix, Q is an m by m orthogonal matrix, and $\widehat{A}$ is of upper trapezoidal form. Furthermore, we assume that the first m columns of $\widehat{A}$ are linearly independent. Note: such decompositions are always available.

Suppose that

$$\mathcal{Q} = \begin{bmatrix} \Pi^T & 0 \\ 0 & Q \end{bmatrix},$$

and

$$\widehat{H} = \Pi^T H \Pi.$$

Then,

$$\mathcal{Q}\mathcal{H}\mathcal{Q}^T = \begin{bmatrix} \widehat{H} & \widehat{A}^T \\ \widehat{A} & 0 \end{bmatrix},$$

and the following theorem holds:

Theorem 6.2.2. *Let $\widehat{H}$ and $\widehat{A}$ be as above, and write $\widehat{A} = \left[\widehat{A}_1, \widehat{A}_2\right]$, where $\widehat{A_1} \in \mathbb{R}^{m\times m}$ is upper triangular and nonsingular. Assume that $\widehat{H}$ is symmetric and positive definite. Then there exists a diagonal matrix $D_1 \in \mathbb{R}^{m\times m}$, a diagonal matrix $D_2 \in \mathbb{R}^{(n-m)\times(n-m)}$, a strictly lower triangular matrix $L_1 \in \mathbb{R}^{m\times m}$, a unit lower triangular matrix $L_2 \in \mathbb{R}^{(n-m)\times(n-m)}$, and a matrix $M \in \mathbb{R}^{(n-m)\times m}$ such that*

$$\begin{bmatrix} \widehat{H} & \widehat{A}^T \\ \widehat{A} & 0 \end{bmatrix} = \begin{bmatrix} \widehat{A}_1^T & 0 & L_1 \\ \widehat{A}_2^T & L_2 & M \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} D_1 & 0 & I \\ 0 & D_2 & 0 \\ I & 0 & 0 \end{bmatrix} \begin{bmatrix} \widehat{A}_1 & \widehat{A}_2 & 0 \\ 0 & L_2^T & 0 \\ L_1^T & M^T & I \end{bmatrix}. \quad (6.2.6)$$

Proof. From [75, Section 3]. Working out the expression in the right hand side, and writing

$$\widehat{H} = \begin{bmatrix} \widehat{H}_{11} & \widehat{H}_{21}^T \\ \widehat{H}_{21} & \widehat{H}_{22} \end{bmatrix},$$

we find that the following must hold:

$$\begin{aligned} \widehat{H}_{11} &= \widehat{A}_1^T D_1 \widehat{A}_1 + L_1 \widehat{A}_1 + \widehat{A}_1^T L_1^T, & (6.2.7)\\ \widehat{H}_{21} &= \widehat{A}_2^T D_1 \widehat{A}_1 + M \widehat{A}_1 + \widehat{A}_2^T L_1^T, & (6.2.8)\\ \widehat{H}_{22} &= L_2 D_2 L_2^T + \widehat{A}_2^T D_1 \widehat{A}_2 + \widehat{A}_2^T M^T + M \widehat{A}_2. & (6.2.9) \end{aligned}$$

Multiplying equation (6.2.7) from the left by $\widehat{A}_1^{-T}$ and from the right by $\widehat{A}_1^{-1}$ gives

$$D_1 + L_1^T \widehat{A}_1^{-1} + \widehat{A}_1^{-T} L_1 = \widehat{A}_1^{-T} \widehat{H}_{11} \widehat{A}_1^{-1}.$$

Using the fact that $\widehat{A}_1$ is upper triangular we obtain the following expressions for D_1 and L_1 :

$$\begin{aligned} D_1 &= \operatorname{diag}(\widehat{A}_1^{-T} \widehat{H}_{11} \widehat{A}_1^{-1}), \\ L_1 &= \widehat{A}_1^T \text{“lower”}(\widehat{A}_1^{-T} \widehat{H}_{11} \widehat{A}_1^{-1}). \end{aligned}$$

Having found D_1 and L_1 and using (6.2.8), the matrix M is given by

$$M = \left(\widehat{H}_{21} - \widehat{A}_2^T L_1^T\right) \widehat{A}_1^{-1} - \widehat{A}_2^T D_1.$$

It remains for us to show that matrices D_2 and L_2 exist such that (6.2.9) is satisfied. Firstly observe that

$$M \widehat{A}_2 = \left(\widehat{H}_{21} - \widehat{A}_2^T L_1\right) \widehat{A}_1^{-1} \widehat{A}_2 - \widehat{A}_2^T D_1 \widehat{A}_2.$$

Substituting this into (6.2.9) and making use of the expressions for D_1 and L_1, we find that

$$\begin{aligned} L_2 D_2 L_2^T &= \widehat{H}_{22} + \widehat{A}_2^T \widehat{A}_1^{-T} \widehat{H}_{11} \widehat{A}_1^{-1} \widehat{A}_2 - \widehat{H}_{21} \widehat{A}_1^{-1} \widehat{A}_2 - \widehat{A}_2^T \widehat{A}_1^{-T} \widehat{H}_{21}^T \\ &= \begin{bmatrix} -\widehat{A}_2^T \widehat{A}_1^{-T} & I \end{bmatrix} \begin{bmatrix} \widehat{H}_{11} & \widehat{H}_{21}^T \\ \widehat{H}_{21} & \widehat{H}_{22} \end{bmatrix} \begin{bmatrix} -\widehat{A}_1^{-1} \widehat{A}_2 \\ I \end{bmatrix}. \end{aligned}$$

Now $\widehat{H}$ is assumed to the symmetric and positive definite, so D_2 and L_2 of appropriate forms can be found.

□

6.2.2 Generalization of the Factorization

We previously saw that by carrying out a transformation of A we can factor the resulting saddle point problem using this Schilders factorization, Theorem 6.2.2. However, some of the sub-blocks have been specifically chosen to be diagonal and others upper triangular. This is not necessary, we only wish for systems with coefficient matrix D_2 to be easy to solve and for matrix-vector multiplications with D_1 to be reasonably cheap. As in Section 6.1, we would also like to be able to use the factorization to implicitly define a constraint preconditioner.

Theorem 6.2.3. *Let $A \in \mathbb{R}^{m \times n}$ $(m \leq n)$ be a full rank matrix which we can partition as $A = [A_1, A_2]$, where $A_1 \in \mathbb{R}^{m \times m}$ is nonsingular. Suppose that $D_1 \in \mathbb{R}^{m \times m}$ and $D_2 \in \mathbb{R}^{(n-m) \times (n-m)}$ are symmetric, and, in addition, D_2 is nonsingular. Furthermore, assume that $L_1 \in \mathbb{R}^{m \times m}$, $L_2 \in \mathbb{R}^{(n-m) \times (n-m)}$ and $M \in \mathbb{R}^{(n-m) \times m}$, where L_2 is assumed to be nonsingular. Then*

$$\begin{bmatrix} G_{11} & G_{21}^T & A_1^T \\ G_{21} & G_{22} & A_2^T \\ A_1 & A_2 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} A_1^T & 0 & L_1 \\ A_2^T & L_2 & M \\ 0 & 0 & I \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} D_1 & 0 & I \\ 0 & D_2 & 0 \\ I & 0 & 0 \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} A_1 & A_2 & 0 \\ 0 & L_2^T & 0 \\ L_1^T & M^T & I \end{bmatrix}}_{P^T}, \tag{6.2.10}$$

where

$$\begin{aligned} G_{11} &= L_1 A_1 + A_1^T L_1^T + A_1^T D_1 A_1, \\ G_{21} &= A_2^T L_1^T + M A_1 + A_2^T D_1 A_1, \\ G_{22} &= L_2 D_2 L_2^T + M A_2 + A_2^T M^T + A_2^T D_1 A_2. \end{aligned}$$

Proof. Multiplying out the factors reveals the result. □

Remark 6.2.4. The matrix D_2 in the above will be identical to the value of D_{22} in (6.1.3) for the same values of G. This factorization can therefore be considered to be an example of a nullspace method.

It follows trivially from Silvester's law of inertia that

$$D_2 \text{ must be positive definite} \tag{6.2.11}$$

for $\mathcal{P}$ to be a meaningful preconditioner [44]. Similarly, D_{22} in (6.1.3) must be positive definite for $K = PBP^T$ to be a meaningful preconditioner, where P and B are defined in Theorem 6.2.3.

It is reasonable to assume that $L_1 = 0$ and $L_2 = I$ since this doesn't restrict the possible forms of G produced but simplifies solves with P, B, and P^T. Under this assumption, and the additional assumption that D_2 is diagonal, the cost of using the implicit factorization preconditioner (6.2.10) is

$$\text{work} = \mathcal{O}\left(\frac{2}{3}m^3 + 2(i+1)(4mn - m^2)\right),$$

where i is the number of iterations carried out by our iterative method. We observe that this upper bound is lower than that of the variable reduction method (the constant involved in the $\mathcal{O}$-notation can be assumed to be the same in each case), so we expect to favour the Schilders factorization for general implicit factorization choices.

One example of this for specific choices in our implicit factorizations is to require that $G_{11} = 0$, $G_{21} = 0$, and $G_{22} = H_{22}$, as in Theorem 5.2.5. We shall assume that H_{22} is positive definite so that a Cholesky factorization can be carried out: the preconditioner is meaningful in terms of the PPCG method. The constraint preconditioner has a block triangular form

$$K = \begin{bmatrix} 0 & 0 & A_1^T \\ 0 & G_{22} & A_2^T \\ A_1 & A_2 & 0 \end{bmatrix},$$

which can be taken advantage of when solving systems of the form

$$K \begin{bmatrix} g_1 \\ g_2 \\ v \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ w \end{bmatrix}.$$

Specifically, we would carry out the following steps:

$$\begin{array}{rcll} v & = & A_1^{-T} r_1 & \mathcal{O}(2m^2) \text{ flops} \\ g_2 & = & H_{22}^{-1}\left(r_2 - A_2^T v\right) & \mathcal{O}(2n^2 - 2mn) \text{ flops} \\ g_1 & = & A_1^{-1}\left(w - A_2 g_2\right) & \mathcal{O}(2mn) \text{ flops} \end{array}$$

giving an upper bound on the work of

$$\text{work} = \mathcal{O}\left(\frac{1}{3}(m^3 + 2(n-m)^3) + 2(i+1)(m^2+n^2)\right) \text{ flops.}$$

Now let us consider how the variable reduction method would solve such a preconditioning step:

$$\begin{array}{rcll} x_2 & = & r_2 - A_2^T A_1^{-T} r_1 & \mathcal{O}(2mn) \text{ flops} \\ v & = & A_1^{-T} r_1 & \mathcal{O}(2m^2) \text{ flops} \\ g_2 & = & H_{22}^{-1} x_2 & \mathcal{O}(2n^2 - 4mn + 2m^2) \text{ flops} \\ y_1 & = & A_1^{-1} w & \mathcal{O}(2m^2) \text{ flops} \\ g_1 & = & y_1 - A_1^{-1} A_2 g_2 & \mathcal{O}(2mn) \text{ flops} \end{array}$$

giving an upper bound on the work of

$$\text{work} = \mathcal{O}\left(\frac{1}{3}(m^3 + 2(n-m)^3) + 2(i+1)(3m^2+n^2)\right) \text{ flops.}$$

If we assume that $L_1 = 0$ and $L_2 = I$ in the Schilders factorization, then we obtain

$$\begin{array}{rcll} v & = & A_1^{-T} r_1 & \mathcal{O}(2m^2) \text{ flops} \\ x_2 & = & r_2 - A_2^T v & \mathcal{O}(2mn) \text{ flops} \\ g_2 & = & H_{22}^{-1} x_2 & \mathcal{O}(2n^2 - 4mn + 2m^2) \text{ flops} \\ g_1 & = & A_1^{-1}(w - A_2 g_2) & \mathcal{O}(2mn) \text{ flops} \end{array}$$

giving an upper bound on the work of

$$\text{work} = \mathcal{O}\left(\frac{1}{3}(m^3 + 2(n-m)^3) + 2(i+1)(m^2+n^2)\right) \text{ flops.}$$

We see that the Schilders factorization carries out the expected steps according to the structure of K, but the variable reduction method is less efficient. We will therefore concentrate on the Schilders factorization from this point onwards.

Remark 6.2.5. The bounds on the number of flops given above are all based on dense matrices. However, it is clear from just observing that the variable reduction method requires two more solves involving A_1 (A_1^T) that the Schilders factorization will be preferable in this case.

6.3 Numerical Experiments

In this section we indicate that, in some cases, the Schilders factorization constraint preconditioners are very effective in practice when applied to equality constrained quadratic programming problems.

We consider the set of quadratic programming examples from the `CUTEr` test set examined in Section 5.2.1. For each we use the projected preconditioned conjugate gradient method of Algorithm 5.1.3 to solve the resulting quadratic programming problem

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} x^T Q x + b^T x \text{ subject to } Ax - d = 0.$$

Firstly a feasible point $x = x_0$ is determined. Thereafter, iterates $x_0 + s$ generated by the conjugate gradient method are constrained to satisfy $As = 0$ by means of the constraint preconditioning system (5.2.3). Since, as frequently happens in practice, $f(x_0 + s)$ may be unbounded from below, a trust-region constraint $\|s\| \leq \Delta$ is also imposed, and the Generalized Lanczos Trust-Region (GLTR) method [46], as implemented in the `GALAHAD` library [48], is used to solve the resulting problem

$$\min_{x \in \mathbb{R}^n} f(x_0 + s) \text{ subject to } As = 0 \text{ and } \|s\| \leq \Delta; \tag{6.3.1}$$

a large value of $\Delta = 10^{10}$ is used so as not to cut off the unconstrained solution for convex problems.

In Tables 6.1 and 6.2, we compare four preconditioning strategies for (approximately) solving the problem (6.3.1). The tables contain the results for the test problems in Table 5.2. The column "fact." contains the CPU time (in seconds) used to factorize the preconditioner, the column "iter." contains the number of PPCG iterations used to reach the required tolerance, and the column "total" contains the total CPU time (in seconds) used to solve the problem. Full listings for all the test problems in the `CUTEr` test set can be found in Appendix B. We consider both low and high(er) accuracy solutions. For the former, we terminate as soon as the norm of the (preconditioned) gradient of $f(x_0 + s)$ has been reduced more than 10^{-2} from that of $f(x_0)$, while the latter requires a 10^{-8} reduction; these are intended to simulate the levels of accuracy required within a nonlinear programming solver in early (global) and later (asymptotic) phases of the solution process.

We consider two explicit factorizations, one using exact factors ($G = H$), and the other using the simple projection ($G = I$). The `HSL` package `MA57` [22] (version 2.2.1) is used to factorize K and subsequently solve (5.2.3); by way of comparison we also include times for exact factorization with the earlier `MA27` [24], since this is still widely used. Two implicit factorizations of the form (6.2.10) are considered. In the first, the simplest, we set $G_{11} = 0$, $G_{21} = 0$ and $G_{22} = I$, i.e. $L_2 = I$ and $D_2 = I$ are used. The second uses $G_{11} = 0$, $G_{21} = 0$ and $G_{22} = H_{22}$. In particular, we exploit one of `MA57`'s options to make modest modifications [76] of the diagonals of H_{22} to ensure that G_{22} is positive definite if H_{22} fails to be—this proved only to be necessary for the `BLOWEY*` problems.

All of our experiments in this section were performed using a single processor of a 3.05Mhz Dell Precision 650 Workstation with 4 Gbytes of RAM. Our codes were written in double precision Fortran 90, compiled using the Intel ifort 8.1 compiler, and wherever possible made use of tuned ATLAS BLAS [85] for core computations. A single iteration of iterative refinement is applied, as necessary, when applying the preconditioner (5.2.3) to try to ensure small relative residuals. All of the problems have been pre-processed by the method described in Section 5.2.1 to ensure that A_1 is nonsingular.

For each option tested, we record the time taken to compute the (explicit or implicit) factors, the number of GLTR iterations performed (equivalently, the number of preconditioned systems solved), and the total CPU time taken to solve the quadratic programming problem (including the factorization). The initial feasible point x_0 is found by solving

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}$$

using the factors of K. Occasionally—in particular when $b = 0$ and $G = H$—such a point solves EQP, and the resulting iteration count is zero. In a few cases, the problems are so ill-conditioned that the trust-region constraint is activated, and more than one GLTR iteration is required to solve EQP even when $G = H$. Furthermore, in some problems, rank deficiency of A violated our assumption of A_1 being nonsingular and resulted in unacceptably large residuals in (5.2.3) and subsequent failure of GLTR when $G = H$, even after iterative refinement.

In Table 6.3 we record the total number of nonzero entries in the factors required by the different preconditioners. For the explicit factorization methods

Table 6.1: The time taken to compute the factors, the number of GLTR iterations performed to achieve a residual decrease of at least 10^{-2}, and the total CPU time taken (including the factorization) to solve various CUTEr QP problems with implicit and explicit preconditioners are shown — times given in seconds

	Explicit factors									Implicit factors					
	$G = H$						$G = I$			$G_{22} = I$			$G_{22} = H_{22}$		
	MA27			MA57			MA57			MA57			MA57		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.08	1	0.13	0.47	1	0.54	0.46	1	0.53	0.04	125	1.54	0.25	125	2.01
BLOCKQP1	0.06	0	0.08	0.21	0	0.23	0.23	1	0.26	0.33	2	0.35	0.39	2	0.41
CVXQP1	579.20	0	580.15	3.99	0	4.11	0.20	3	0.24	0.21	57	0.56	0.24	55	0.69
KSIP	0.01	1	0.02	0.05	1	0.06	0.04	3	0.05	0.02	3	0.03	0.02	3	0.03
PRIMAL1	0.01	1	0.01	0.01	1	0.02	0.03	5	0.03	0.00	15	0.01	0.00	27	0.02
STCQP2	9.76	0	9.84	0.87	0	0.92	0.14	3	0.17	0.03	3	0.05	0.11	1	0.13
UBH1	0.02	0	0.03	0.12	0	0.14	0.11	0	0.13	0.02	0	0.03	0.04	0	0.05

Table 6.2: The time taken to compute the factors, the number of GLTR iterations performed to achieve a residual decrease of at least 10^{-8}, and the total CPU time taken (including the factorization) to solve various CUTEr QP problems with implicit and explicit preconditioners are shown — times given in seconds

	Explicit factors									Implicit factors					
	$G = H$						$G = I$			$G_{22} = I$			$G_{22} = H_{22}$		
	MA27			MA57			MA57			MA57			MA57		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.08	1	0.13	0.47	1	0.54	0.46	1	0.53	0.04	866	10.35	0.25	872	12.50
BLOCKQP1	0.06	0	0.08	0.21	0	0.23	0.23	1	0.26	0.33	3	0.37	0.39	3	0.43
CVXQP1	579.20	0	580.15	3.99	0	4.11	0.20	5	0.27	0.21	211	1.49	0.24	207	1.94
KSIP	0.01	1	0.02	0.05	1	0.06	0.04	5	0.05	0.02	18	0.05	0.02	10	0.04
PRIMAL1	0.01	1	0.01	0.01	1	0.02	0.03	8	0.03	0.00	153	0.08	0.00	158	0.09
STCQP2	9.76	0	9.84	0.87	0	0.92	0.14	7	0.20	0.03	7	0.07	0.11	1	0.13
UBH1	0.02	0	0.03	0.12	0	0.14	0.11	0	0.13	0.02	0	0.03	0.04	0	0.05

Table 6.3: CUTEr QP problems — total number of nonzero entries in factors

	Explicit factors MA57		Implicit factors MA57	
name	$G = H$	$G = I$	$G_{22} = I$	$G_{22} = H_{22}$
AUG2DCQP	35061	35061	8004	8004
BLOCKQP1	110080	105080	25022	25022
CVXQP1	5872175	120118	25354	40155
KSIP	25236	25236	2042	2042
PRIMAL1	10458	10458	820	820
STCQP2	288794	24952	19480	35419
UBH1	66009	66009	24006	24006

the HSL subroutine MA57 is used to factor the preconditioner K into LDL^T — we record the number of nonzeros in L and D. In our implicit method we need to factor A_1 and D_2 — the number of nonzeros in the resulting factors are recorded. We observe that in many cases the implicit factorization methods have a dramatic reduction in the number of nonzeros in the factors, hence the amount of memory required will be a lot lower.

It is difficult to analyze which preconditioner is performing better overall by just looking at the tables. We shall therefore use performance profiles to illustrate the results over the whole CUTEr test set. To explain the idea, let Φ represent the set of preconditioners that we wish to compare. Suppose that using a given preconditioner $i \in \Phi$ the quadratic programming problem is solved in total CPU time $t_{ij} \geq 0$ for example j from the test set $\mathcal{T}$. For all the problems $j \in \mathcal{T}$, we want to compare the performance of the use of preconditioner i with the performance of the fastest algorithm using a preconditioner from the set Φ. For $j \in \mathcal{T}$, let $t_j^{\text{MIN}} = \min\{t_{ij}; i \in \Phi\}$. Then for $\alpha \geq 1$ and each $i \in \Phi$ we define

$$k(t_{ij}, t_j^{\text{MIN}}, \alpha) = \begin{cases} 1 & \text{if } t_{ij} \leq \alpha t_j^{\text{MIN}} \\ 0 & \text{otherwise.} \end{cases}$$

The *performance profile* of the algorithm using preconditioner i is then given by the function

$$p_i(\alpha) = \frac{\sum_{j \in \mathcal{T}} k(t_{ij}, t_j^{\text{MIN}}, \alpha)}{|\mathcal{T}|}, \alpha \geq 1.$$

Hence, $p_i(1)$ gives the fractions of problems for which the algorithm using preconditioner i is most effective according to the total CPU time used, $p_i(2)$ gives the fraction for which the algorithm using preconditioner i is within a factor of 2 of the best, and $\lim_{\alpha \to \infty} p_i(\alpha)$ gives the fraction for which the algorithm succeeded.

In Figures 6.1 and 6.2 (see Tables B.1 and B.2 for the raw data), we compare the five different preconditioning strategies. We see that if low accuracy solutions suffice, the Schilders factorization with $G_{22} = I$ is significantly more effective at reducing the residual than its explicit counterparts. For higher accuracy solutions the explicit factorization with $G = I$ is generally more cost effective than the Schilders factorizations. For the majority of the test problems, it is clear that the use of MA57 is preferable to MA27.

We must admit to being slightly disappointed that the more sophisticated Schilders factorization with $G_{22} = H_{22}$ seemed to show few advantages over

the cheaper $G_{22} = I$, but this might reflect the nature of H in our test set. In Chapter 8 we will see the advantage of the choice $G_{22} = H_{22}$ when solving mixed constraint optimization problems.

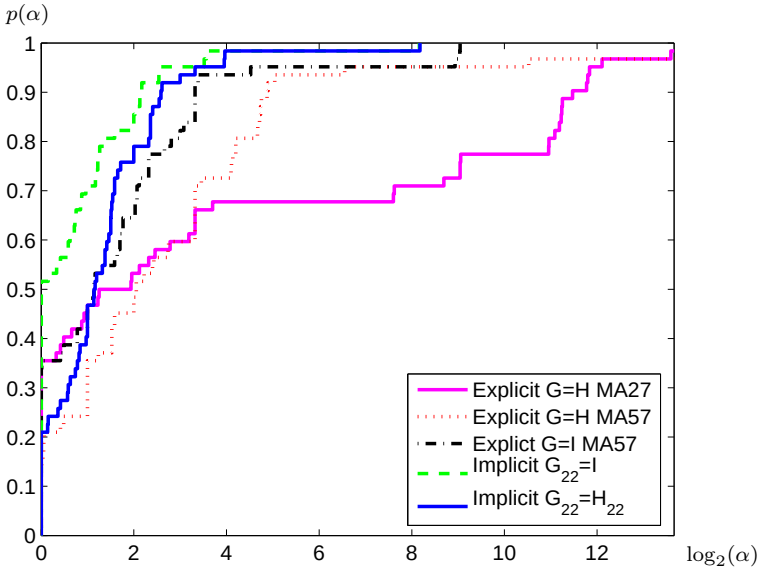

Figure 6.1: Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-2}.

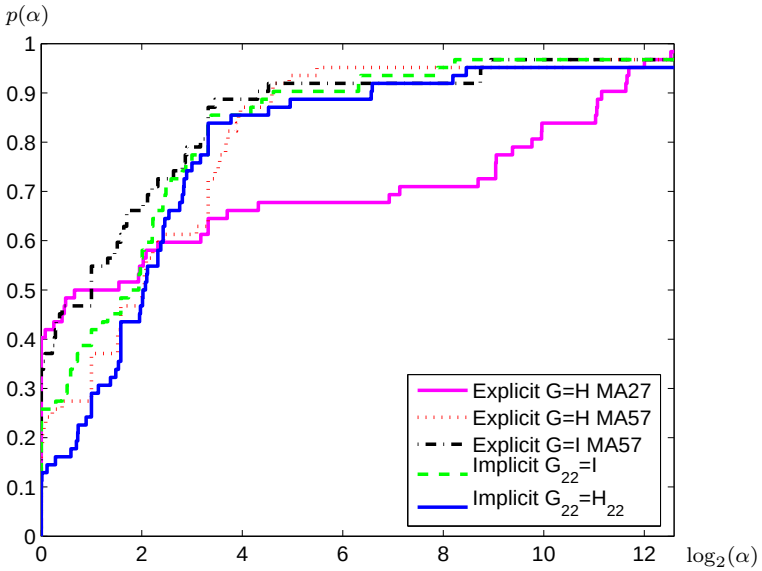

Figure 6.2: Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-8}.

Chapter 7

Implicit Factorization Constraint Preconditioners

In Chapter 6 we considered how to use the Schilders factorization (and variable reduction method) to generate constraint preconditioners for the case $C = 0$. In this chapter we will extend this idea to the use of implicit factorization constraint preconditioners for $C \neq 0$.

7.1 Generating implicit factorization constraint preconditioners

Let K be a constraint preconditioner of the form

$$K = \begin{bmatrix} G & A^T \\ A & -C \end{bmatrix}, \tag{7.1.1}$$

where $G \in \mathbb{R}^{n \times n}$ approximates, but is not the same as H [see Section 5.4]. Again, let us expand out K in a similar manner to that done in (6.1.1):

$$K = \begin{bmatrix} G_{11} & G_{21}^T & A_1^T \\ G_{21} & G_{22} & A_2^T \\ A_1 & A_2 & -C \end{bmatrix}, \tag{7.1.2}$$

where $G_{11} \in \mathbb{R}^{m \times m}$, $G_{21} \in \mathbb{R}^{(n-m) \times m}$, $G_{22} \in \mathbb{R}^{(n-m) \times (n-m)}$, $A_1 \in \mathbb{R}^{m \times m}$ and $A_2 \in \mathbb{R}^{m \times (n-m)}$. As in the previous chapter, we shall assume that A_1 and its transpose are easily invertible: we shall consider how to partition A to produce such an A_1 in Chapter 8.

Again, we wish to consider preconditioners of the form

$$K = PBP^T, \tag{7.1.3}$$

where solutions with each of the matrices P, B and P^T are easily obtained. In particular, rather than obtaining P and B from a given K, K is derived from specially chosen P and B.

Suppose that

$$P = \begin{bmatrix} P_{11} & P_{12} & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & B_{21}^T & B_{31}^T \\ B_{21} & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{bmatrix}, \tag{7.1.4}$$

where the corner sub-blocks are all m by m in dimension. For the factorization PBP^T to be a constraint preconditioner we require that

$$\begin{aligned} A_1 &= (P_{31}B_{11} + P_{32}B_{21})\,P_{11}^T + \left(P_{31}B_{21}^T + P_{32}B_{22}\right)P_{12}^T \\ &\quad + P_{33}\left(B_{31}P_{11}^T + B_{32}P_{12}^T\right) + \left(P_{31}B_{31}^T + P_{32}B_{32}^T\right)A_1 \\ &\quad + P_{33}B_{33}A_1 \end{aligned} \tag{7.1.5}$$

$$\begin{aligned} A_2 &= (P_{31}B_{11} + P_{32}B_{21})\,P_{21}^T + \left(P_{31}B_{21}^T + P_{32}B_{22}\right)P_{22}^T \\ &\quad + P_{33}\left(B_{31}P_{21}^T + B_{32}P_{22}^T\right) + \left(P_{31}B_{31}^T + P_{32}B_{32}^T\right)A_2 \\ &\quad + P_{33}B_{33}A_2 \end{aligned} \tag{7.1.6}$$

$$\begin{aligned} -C &= (P_{31}B_{11} + P_{32}B_{21})\,P_{31}^T + \left(P_{31}B_{21}^T + P_{32}B_{22}\right)P_{32}^T \\ &\quad + P_{33}\left(B_{31}P_{31}^T + B_{32}P_{32}^T\right) + \left(P_{31}B_{31}^T + P_{32}B_{32}^T\right)P_{33}^T \\ &\quad + P_{33}B_{33}P_{33}^T. \end{aligned} \tag{7.1.7}$$

Pragmatically, we are only interested in the case where one of the three possibilities

$$P_{11} = 0, \quad P_{12} = 0 \quad \text{and} \quad P_{32} = 0, \tag{7.1.8}$$

$$\text{or} \quad P_{11} = 0, \quad P_{12} = 0 \quad \text{and} \quad P_{21} = 0, \tag{7.1.9}$$

$$\text{or} \quad P_{12} = 0, \quad P_{32} = 0 \quad \text{and} \quad P_{33} = 0, \tag{7.1.10}$$

(as well as nonsingular P_{31} and P_{22}) holds, since P will then be easily block-invertible. Likewise, we restrict ourselves to the three general cases

$$B_{21} = 0, \quad B_{31} = 0 \quad \text{and} \quad B_{32} = 0, \tag{7.1.11}$$

$$\text{or} \quad B_{32} = 0, \quad B_{33} = 0 \ \text{ with easily invertible } \ B_{31} \ \text{ and } \ B_{22}, \tag{7.1.12}$$

$$\text{or} \quad B_{11} = 0, \quad B_{21} = 0 \ \text{ with easily invertible } \ B_{31} \ \text{ and } \ B_{22}, \tag{7.1.13}$$

so that B is block invertible. B will also be easily block invertible if

$$B_{21} = 0 \ \text{ and } \ B_{32} = 0 \ \text{ with easily invertible } \begin{bmatrix} B_{11} & B_{31}^T \\ B_{31} & B_{33} \end{bmatrix} \ \text{ and } \ B_{22}, \tag{7.1.14}$$

so we will also consider this possibility.

We shall examine in detail the case where (7.1.8) and (7.1.11) hold. The rest of the cases will be examined in Appendix C and the resulting implicit-factorization families are summarized in Tables 7.1 and 7.2.

If (7.1.8) and (7.1.11) hold, then P_{31}, P_{22}, B_{11}, B_{22} and B_{33} are required to be nonsingular, and

$$\begin{aligned} A_1 &= P_{33}B_{33}A_1, && (7.1.15)\\ A_2 &= P_{31}B_{11}P_{21}^T + P_{33}B_{33}A_2, && (7.1.16)\\ -C &= P_{31}B_{11}P_{31}^T + P_{33}B_{33}P_{33}^T. && (7.1.17) \end{aligned}$$

Equation (7.1.15) implies that

$$P_{33}B_{33} = I \tag{7.1.18}$$

and, hence, that P_{33} is symmetric. Equation (7.1.16) forces $P_{31}B_{11}P_{21}^T = 0$, and thus that

$$P_{21} = 0$$

since P_{31} and B_{11} are nonsingular. Finally, (7.1.17) becomes

$$-C = P_{31}B_{11}P_{31}^T + P_{33}. \tag{7.1.19}$$

We therefore have

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{bmatrix}, \tag{7.1.20}$$

where

$$B_{11} = -P_{31}^{-1}\left(C + P_{33}\right)P_{31}^{-T} \quad \text{and} \quad B_{33} = P_{33}^{-1}. \tag{7.1.21}$$

We shall refer to this as Family 1.

Remark 7.1.1. We note that there is no restriction on the matrices P_{22} and B_{22} in Family 1. This is also the same for all of the families in Tables 7.1 and 7.2.

Table 7.1: Possible implicit factors for the preconditioner (7.1.3). We give the P and B factors and any necessary restrictions on their entries. We also associate a family number with each class of implicit factors, and indicate where each is derived in Appendix C.

Family/ reference	P	B	conditions
1. (C.0.6) –(C.0.7)	$\begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix}$	$\begin{bmatrix} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{bmatrix}$	$B_{11} = -P_{31}^{-1}\left(C + P_{33}\right)P_{31}^{-T}$ $B_{33} = P_{33}^{-1}$
2. (C.0.16) –(C.0.17)	$\begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix}$	$\begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & 0 \end{bmatrix}$	$B_{11} = -P_{31}^{-1}\left(C + P_{33} + P_{33}^T\right)P_{31}^{-T}$ $B_{31} = P_{31}^{-T}$
3. (C.0.18) –(C.0.19)	$\begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & -C \end{bmatrix}$	$\begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & 0 \end{bmatrix}$	$B_{11} = P_{31}^{-1}CP_{31}^{-T}$ $B_{31} = P_{31}^{-T}$
4. (C.0.25) –(C.0.26)	$\begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & 0 \end{bmatrix}$	$P_{21} = -P_{22}CB_{32}P_{31}$ $B_{31} = P_{31}^{-T}$ $C = -\left(P_{33} + P_{33}^T\right)$
5. (C.0.27) –(C.0.28)	$\begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$	$B_{32} = -B_{31}P_{21}^TP_{22}^{-T}$ $B_{31} = \left(I - B_{33}P_{33}^T\right)P_{31}^{-T}$ $-C = P_{33} + P_{33}^T - P_{33}B_{33}P_{33}^T$
6. (C.0.35) –(C.0.36)	$\begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & P_{32} & P_{33} \end{bmatrix}$	$\begin{bmatrix} B_{11} & B_{21}^T & B_{31}^T \\ B_{21} & B_{22} & 0 \\ B_{31} & 0 & 0 \end{bmatrix}$	$P_{32} = -P_{31}B_{21}^TB_{22}^{-1}$ $B_{31} = P_{31}^{-T}$ $-C = P_{31}\left(B_{11} - B_{31}^TB_{22}^{-1}B_{21}\right)P_{31}^T$ $+P_{33} + P_{33}^T$
7. (C.0.45) –(C.0.46)	$\begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & P_{32} & P_{33} \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$	$P_{32} = -P_{33}B_{32}B_{22}^{-1}$ $P_{31} = \left(I - P_{32}B_{32}^T - P_{33}B_{33}^T\right)B_{31}^{-T}$ $-C = P_{33}\left(B_{33} - B_{32}B_{22}^{-1}B_{32}^T\right)P_{33}^T$ $+P_{33} + P_{33}^T$

Table 7.2: Possible implicit factors for the preconditioner (7.1.3). We give the P and B factors and any necessary restrictions on their entries. We also associate a family number with each class of implicit factors, and indicate where each is derived in Appendix C.

Family/ reference	P	B	conditions
8. (C.0.47)	$\begin{bmatrix} A_1^T & 0 & A_1^T \\ A_2^T & P_{22} & A_2^T \\ -C & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -C^{-1} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{bmatrix}$	C invertible
9. (C.0.51) –(C.0.52)	$\begin{bmatrix} P_{11} & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} B_{11} & B_{21}^T & B_{31}^T \\ B_{21} & B_{22} & 0 \\ B_{31} & 0 & 0 \end{bmatrix}$	$B_{11} = -P_{31}^{-1} C P_{31}^{-T}$ $B_{31} = P_{31}^{-T} - M B_{11}$ $B_{21} = P_{22}^{-1}\left(P_{21} - A_2^T\right) B_{11}$ $P_{11} = A_1^T M$ for some invertible M
10. (C.0.53)	$\begin{bmatrix} P_{11} & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$	$C = 0$ $B_{31} = P_{31}^{-T}$
11. (C.0.60) –(C.0.61)	$\begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & -C \end{bmatrix}$	$\begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{bmatrix}$	C invertible $P_{31}^T = B_{11}^{-1} B_{31}^T C$ $B_{33} = \left(B_{31} P_{31}^T - I\right) C^{-1}$
11. (C.0.60), (C.0.65) –(C.0.66)	$\begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & -C \end{bmatrix}$	$\begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{bmatrix}$	$B_{11} = P_{31}^{-1} C P_{31}^{-T}$ $B_{31} = P_{31}^{-T}$ $B_{33} C = 0$
13. (C.0.68) –(C.0.69)	$\begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix}$	$\begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{bmatrix}$	$P_{31} = (I - P_{33} B_{33}) B_{31}^{-T}$ $B_{11} = P_{31}^{-1}\left(P_{33} B_{33} P_{33}^T\right.$ $\left. = -C - P_{33} - P_{33}^T\right) P_{31}^{-T}$
14. (C.0.76) –(C.0.77)	$\begin{bmatrix} P_{11} & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{bmatrix}$	$B_{11} = -P_{31}^{-1} C P_{31}^{-T}$ $B_{31} = P_{31}^{-T} - M B_{11}$ $P_{11} = A_1^T M$ $P_{21} = A_2^T M$ for some invertible M

7.2 Reproducing H

Having described families of preconditioners which are capable of reproducing the required components A and C of K, we now examine what form the resulting G takes. In particular, we consider which submatrices of G can be defined to completely reproduce the associated submatrix of H; we say that a component G_{ij}, $i, j \in \{1, 2\}$, is *complete* if it is possible to choose it such that $G_{ij} = H_{ij}$. We give the details in Appendix D, and summarize our findings for each of the 14 families from Section 7.1 in Table 7.3. In Table 7.3 the superscript [1] indicates that the value of G_{21} is dependent on the choice of G_{11}. If G_{ij}, $i, j \in \{1, 2\}$, is a zero matrix, then a superscript [2] is used. The superscript [3] means that G_{21} is dependent on the choice of G_{11} when $C = 0$, but complete otherwise, whilst the superscript [4] indicates that G_{11} is only guaranteed to be complete when $C = 0$.

Some of the submatrices in the factors P and B can be arbitrarily chosen without changing the completeness of the family. We shall call these *free blocks*. For example, consider Family 2 from Table 7.1. The matrix G produced by this family always satisfies $G_{11} = 0$, $G_{21} = 0$, and $G_{22} = P_{22}B_{22}P_{22}^T$. Hence, P_{22} can be defined as any nonsingular matrix of suitable dimension, and B_{22} can be subsequently chosen so that $G_{22} = H_{22}$. The simplest choice for P_{22} is the identity matrix. We observe, that the choice of the remaining submatrices in P and B will not affect the completeness of the factorization, and are only required to satisfy the conditions given in Table 7.1. The simplest choices for these submatrices will be $P_{31} = I$, and $B_{11} = 0$, giving $P_{33} = -\frac{1}{2}C$, and $B_{31} = I$. Using these simple choices we obtain:

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ 0 & I & A_2^T \\ I & 0 & -\frac{1}{2}C \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 0 & I \\ 0 & B_{22} & 0 \\ I & 0 & 0 \end{bmatrix}.$$

The simplest choice of the free blocks may result in some of the families having the same factors as other families. This is indicated in the "Comments" column of the table. Table 7.3 also gives the conditions that C must satisfy to use the family, and whether the family is feasible to use, i.e., are the conditions on the blocks given in Tables 7.1 and 7.2 easily satisfied?

Table 7.4 gives some guidance towards which families from Tables 7.1 and 7.2 should be used in the various cases of G given in Section 5.4.2. We also suggest simple choices for the free blocks. In our view, although Table 7.3

Family	Completeness G_{11}	G_{21}	G_{22}	Conditions on C	Feasible to use	Comments
1.	✓	$\times^1$	✓	any C	✓	
2.	$\times^2$	$\times^2$	✓	any C	✓	
3.	$\times^2$	✓	✓	any C	✓	
4.	$\times^2$	$\times^2$	✓	any C	✓	Simplest choice of free blocks is the same as that for Family 2.
5.	✓	$\times^1$	✓	any C	$C = 0$	
6.	$\times^2$	$\times^2$	✓	any C	✓	Simplest choice of free blocks is the same as that for Family 2.
7.	✓	✓^{3}	✓	any C	$C = 0$	If $C = 0$ and use simplest choice of free blocks, then same as that for Family 5 with $C = 0$.
8.	✓	$\times^1$	✓	nonsingular	✓	
9.	✓	✓	✓	any C	$C = 0$	
10.	✓	✓	✓	$C = 0$	✓	Generalization of factorization suggested by Schilders, see Chapter 6.
11.	✓	✓	✓	nonsingular	✓	
12.	✓^{4}	✓	✓	any C	diagonal C	$C = 0$ gives example of Family 10. C nonsingular gives Family 3.
13.	✓	$\times^1$	✓	any C	✓	
14.	✓	$\times^1$	✓	any C	✓	$C = 0$ gives example of Family 10.

Table 7.3: Blocks of G for the families of preconditioners given in Tables 7.1 and 7.2. The superscripts used are defined in the first paragraph of Section 7.2.

Sub-blocks of G	Conditions on C	Family	Free block choices
$G_{22} = H_{22}$, $G_{11} = 0$, $G_{21} = 0$	any C	2	$P_{22} = I$, $P_{31} = I$, $B_{11} = 0$
$G_{22} = H_{22}$, $G_{11} = H_{11}$, $G_{21} = 0$	$C = 0$	10	$B_{21} = 0$, $P_{22} = I$, $P_{31} = I$
$G_{22} = H_{22}$, $G_{11} = H_{11}$, $G_{21} = 0$	C non-singular	11	$P_{22} = I$, $P_{31} = I$
$G_{22} = H_{22}$, $G_{21} = H_{21}$, $G_{11} = 0$	any C	3	$P_{22} = I$, $P_{31} = I$

Table 7.4: Guidance towards which family to use to generate the various choices of G given in Section 5.4.2.

indicates that it is theoretically possible to reproduce all of H using (e.g.) Family 9, in practice this will often be unviable because of the resulting density of some of the matrices that need to be factorized.

7.3 Numerical Experiments

In this section we examine how effective implicit factorization preconditioners might be when compared to explicit ones. We consider problems generated using the complete set of quadratic programming examples from the `CUTEr` test set examined in Section 5.2.1. All inequality constraints are converted to equations by adding slack variables, and a suitable "barrier" penalty term (in this case, 1.1) is added to the diagonal of the Hessian for each bounded or slack variable to simulate systems that might arise during an iteration of an interior-point method for such problems. The resulting equality constrained quadratic programs are then of the form

$$\min_{x \in \mathbb{R}^n} f(x) = \frac{1}{2} x^T H x + b^T x \text{ subject to } Ax = 0. \tag{7.3.1}$$

Given the data H and A, two illustrative choices of diagonal C are considered, namely

$$c_{ii} = 1 \quad \text{for} \quad 1 \leq i \leq m, \tag{7.3.2}$$

and

$$c_{ii} = \begin{cases} 0 & \text{for} \quad 1 \leq i \leq \lceil \frac{m}{2} \rceil, \\ 1 & \text{for} \quad \lceil \frac{m}{2} \rceil + 1 \leq i \leq m, \end{cases} \tag{7.3.3}$$

in practice such C may be thought of as regularization terms for some or all of the constraints in (7.3.1). We wish to solve the resulting saddle point systems (5.3.1) using Algorithm 5.3.2.

We consider two explicit factorization preconditioners, one using exact factors ($G = H$), and the other using a simple projection ($G = I$). A Matlab interface to the HSL package `MA57` [22] (version 2.2.1) is used to factorize K and subsequently solve (5.3.12). Three implicit factorizations of the form (7.1.3) are also considered. The first is from Family 1 (Table 7.1), and aims for simplicity by choosing $P_{31} = I$, $P_{33} = I = B_{33}$ and $B_{22} = I = P_{22}$, and this leads to $B_{11} = -(C + I)$; such a choice does not necessarily reproduce any of H, but is inexpensive to use. The remaining implicit factorizations are from Family 2 (Table 7.1). The former (marked (a) in the Figures) selects $G_{22} = H_{22}$ while the latter (marked (b) in the Figures) chooses $G_{22} = I$; for simplicity we chose $P_{31} = I = B_{31}$, $B_{11} = 0$, $P_{22} = I$ and $P_{33} = -\frac{1}{2}C$ (see Section 7.2), and thus we merely require that $B_{22} = H_{22}$ for case (a) and $B_{22} = I$ for case (b)—we use `MA57` to factorize H_{22} in the former case.

Given A, a suitable basis matrix A_1 is found by finding a sparse LU factorization of A^T using the built-in Matlab function `lu`. An attempt to correctly identify rank is controlled by tight threshold column pivoting, in which any pivot may not be smaller than a factor $\tau = 2$ of the largest entry in its (uneliminated) column [38].

Although such a strategy may not be as robust as, say, a singular value decomposition or a QR factorization with pivoting, both our and experiences others' [38] indicate it to be remarkably reliable and successful in practice. Having found A_1, the factors are discarded, and a fresh LU decomposition of A_1, with a looser threshold column pivoting factor $\tau = 100$, is computed using `lu` in order to try to encourage sparse factors.

All of our experiments in this chapter were performed using a dual processor Intel Xeon 3.2GHz Workstation with hyper-threading and 2 Gbytes of RAM. Our codes were written and executed in Matlab 7.0 Service Pack 1.

In Tables 7.5 and 7.6 we compare our five preconditioning strategies for (approximately) solving the problem (5.3.1) using Algorithm 5.3.2 when C is given by (7.3.2). We consider both low and high(er) accuracy solutions. For the former, we terminate as soon as the residual σ has been reduced by more than 10^{-2} from its original value, while the latter requires a 10^{-8} reduction. The column "fact." contains the CPU time (in seconds) used to factorize the preconditioner, the column "iter." contains the number of PPCG iterations used to reach the required tolerance, and the column "total" contains the total CPU time (in seconds) used to solve the problem. Tables containing the full

Table 7.5: Time taken to compute the factors, number of PPCG iterations performed to achieve a residual decrease of at least 10^{-2}, and total CPU time taken (including the factorization) to solve (5.3.1) using Algorithm 5.3.2 when $C = I$ for various CUTEr QP problems with implicit and explicit preconditioners — times given in seconds

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.38	1	0.45	0.12	1	0.18	0.07	13	0.19	0.07	267	1.94	0.02	36	0.30
BLOCKQP1	5.03	1	33.28	4.98	1	33.15	0.14	1	28.20	0.06	1	28.18	0.06	1	28.09
CONT5-QP	ran out of memory			ran out of memory			0.91	1	6.63	0.24	1	5.94	0.25	1	5.92
CVXQP1	48.16	1	50.38	139.83	1	142.34	0.18	2	0.36	0.13	1310	43.43	0.12	1258	42.24
KSIP	0.50	1	1.04	0.50	1	1.02	0.06	2	0.59	0.01	2	0.55	0.01	2	0.55
PRIMAL1	0.11	1	0.76	0.11	1	0.12	0.05	19	0.08	0.01	8	0.03	0.01	2	0.02
STCQP2	0.86	1	0.96	1.47	1	1.62	0.05	4	0.12	0.09	1	0.13	0.09	2622	38.05
UBH1	0.34	1	0.52	0.33	1	0.52	0.13	2	0.29	0.05	1	0.17	0.05	4	0.23

Table 7.6: Time taken to compute the factors, number of PPCG iterations performed to achieve a residual decrease of at least 10^{-8}, and total CPU time taken (including the factorization) to solve (5.3.1) using Algorithm 5.3.2 when $C = I$ for various CUTEr QP problems with implicit and explicit preconditioners — times given in seconds

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.38	1	0.44	0.12	1	0.18	0.07	157	1.11	0.07	1220	8.48	0.02	89	0.66
BLOCKQP1	5.03	1	33.06	4.98	1	33.11	0.14	1	28.22	0.06	2	28.20	0.06	2	28.14
CONT5-QP	ran out of memory			ran out of memory			0.91	1	6.71	0.24	110	30.72	0.25	147	37.19
CVXQP1	48.16	1	50.39	139.83	1	142.36	0.18	51	1.56	0.13	9237	305.76	0.12	4165	138.96
KSIP	0.50	1	1.02	0.50	1	1.03	0.06	8	0.61	0.01	6	0.57	0.01	6	0.57
PRIMAL1	0.11	1	0.12	0.11	1	0.12	0.05	172	0.30	0.01	21	0.05	0.01	31	0.07
STCQP2	0.86	1	0.96	1.47	1	1.62	0.05	92	1.12	0.09	1	0.13	0.09	6140	89.14
UBH1	0.34	1	0.53	0.33	1	0.52	0.13	30	0.87	0.05	472	10.12	0.05	47	1.13

Table 7.7: Time taken to compute the factors, number of PPCG iterations performed to achieve a residual decrease of at least 10^{-2}, and total CPU time taken (including the factorization) to solve (5.3.1) using Algorithm 5.3.2 when C given by (7.3.3) for various CUTEr QP problems with implicit and explicit preconditioners — times given in seconds

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.13	1	0.19	0.13	1	0.19	0.06	19	0.21	0.02	43	0.35	0.02	1	0.06
BLOCKQP1	4.94	2	32.95	4.93	2	33.05	0.14	1	28.14	0.06	1	28.07	0.07	1	28.10
CONT5-QP	ran out of memory			ran out of memory			0.87	1	6.50	0.25	1	5.87	0.25	1	5.83
CVXQP1	0.48	2	0.75	0.46	2	0.79	0.14	2	0.33	0.06	1	0.22	0.06	1	0.22
KSIP	0.50	2	1.03	0.50	1	1.03	0.03	1	0.56	0.01	2	0.54	0.01	1	0.52
PRIMAL1	0.12	2	0.13	0.12	1	0.13	0.04	18	0.07	0.02	9	0.04	0.02	2	0.03
STCQP2	0.13	1	0.19	0.14	1	0.20	0.05	4	0.12	0.03	255	3.08	0.03	4	0.09
UBH1	0.34	1	0.52	0.34	1	0.52	0.14	2	0.29	0.05	1	0.17	0.05	4	0.23

Table 7.8: Time taken to compute the factors, number of PPCG iterations performed to achieve a residual decrease of at least 10^{-8}, and total CPU time taken (including the factorization) to solve (5.3.1) using Algorithm 5.3.2 when C given by (7.3.3) for various CUTEr QP problems with implicit and explicit preconditioners — times given in seconds

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.13	6	0.26	0.13	7	0.27	0.06	163	1.12	0.02	1158	7.94	0.02	233	1.61
BLOCKQP1	4.94	2	33.05	4.93	2	32.97	0.14	1	28.12	0.06	2	28.18	0.07	2	28.17
CONT5-QP	ran out of memory			ran out of memory			0.87	1	6.55	0.25	36	13.90	0.25	36	13.30
CVXQP1	0.48	124	6.61	0.46	73	4.99	0.14	97	2.63	0.06	21	0.74	0.06	22	0.76
KSIP	0.50	6	1.05	0.50	8	1.07	0.03	15	0.61	0.01	6	0.57	0.01	5	0.54
PRIMAL1	0.12	6	0.14	0.12	9	0.15	0.04	166	0.28	0.02	15	0.05	0.02	30	0.07
STCQP2	0.13	57	1.07	0.14	51	1.14	0.05	67	0.79	0.03	6029	71.04	0.03	5989	66.54
UBH1	0.34	6	0.76	0.34	5	0.67	0.14	28	0.82	0.05	31	0.81	0.05	24	0.65

results for the CUTEr collection can be found in Appendix E. Tables 7.7 and 7.8 (c.f., Tables E.3–E.4) repeat the experiments when C is given by (7.3.3).

As in the previous chapter, as well as presenting tables of data in this chapter we use performance profiles, see Section 6.3. Figures 7.1 and 7.2 correspond to Tables E.1 and E.2 respectively.

We see that if low accuracy solutions suffice, then the implicit factorizations appear to be significantly more effective at reducing the residual than their explicit counterparts. In particular, the implicit factorization from Family 1 seems to be the most effective. Of interest is that for Family 2, the cost of applying the more accurate implicit factorization that reproduces H_{22} generally does not pay off relative to the cost of the cheaper implicit factorizations. This was also observed when we used the Schilders factorization, see Section 6.3. For higher accuracy solutions, the leading implicit factorization still slightly outperforms the explicit factors, although the remaining implicit factorizations are now less effective.

Figures 7.3 and 7.4 correspond to Tables E.3 and E.4 respectively. Once again the implicit factorizations seem very effective, with a shift now to favour those from Family 2, most especially the less sophisticated of these.

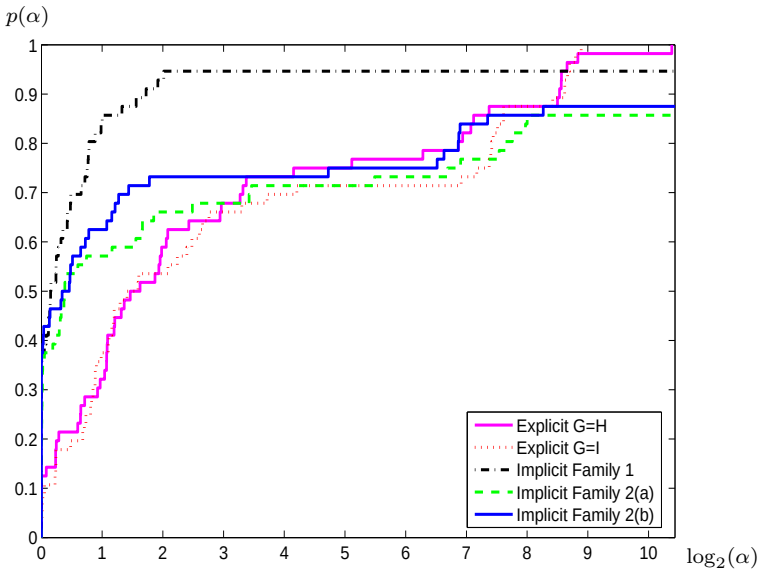

Figure 7.1: Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-2}, when C is given by (7.3.2).

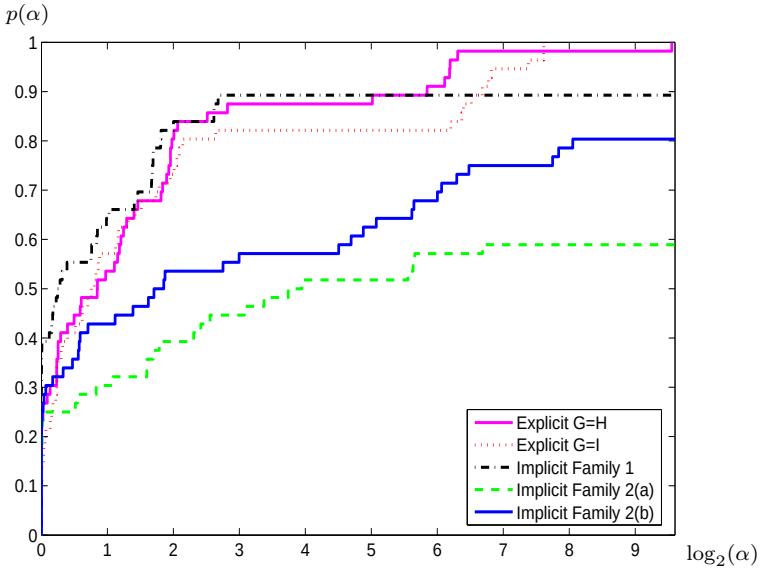

Figure 7.2: Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-8}, when C is given by (7.3.2).

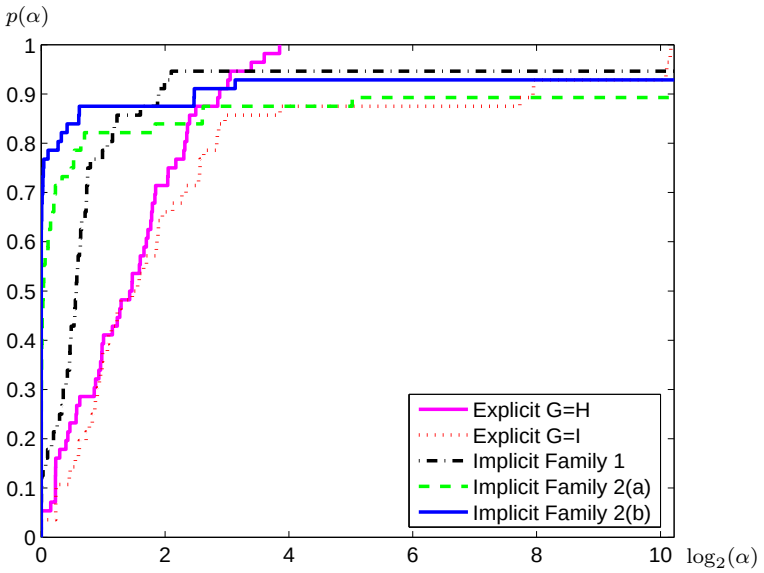

Figure 7.3: Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-2}, when C is given by (7.3.3).

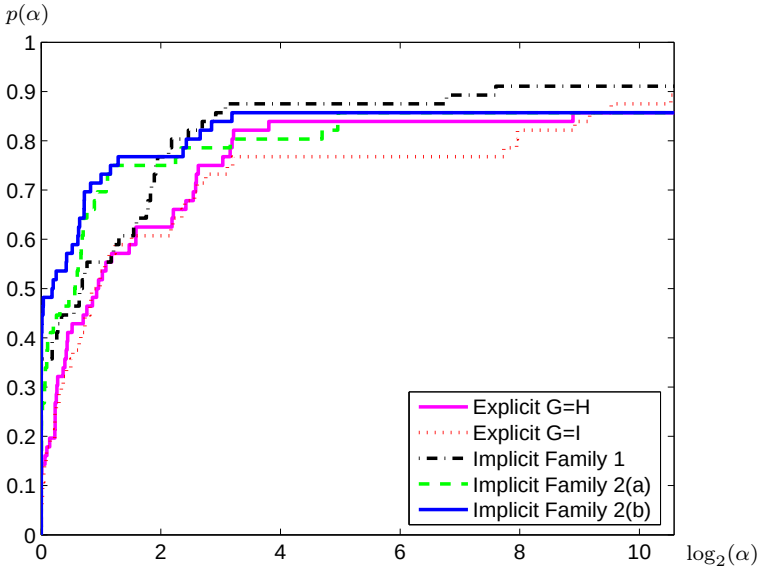

Figure 7.4: Performance profile, $p(\alpha)$: CPU time (seconds) to reduce relative residual by 10^{-8}, when C is given by (7.3.3).

Chapter 8

Permutations and the Nonsingularity of A_1

In Chapters 5, 6 and 7 we assumed that we can express the block 2 by 2 saddle point system

$$\underbrace{\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}$$

in a block 3 by 3 structure

$$\begin{bmatrix} H_{1,1} & H_{1,2} & A_1^T \\ H_{2,1} & H_{2,2} & A_2^T \\ A_1 & A_2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ y \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ d \end{bmatrix},$$

where $H_{11} \in \mathbb{R}^{m\times m}$, $H_{21} \in \mathbb{R}^{(n-m)\times m}$, $H_{22} \in \mathbb{R}^{(n-m)\times(n-m)}$, $A_1 \in \mathbb{R}^{m\times m}$, $A_2 \in \mathbb{R}^{m\times(n-m)}$ and A_1 is nonsingular. In practice, we know that A is of full rank but we certainly cannot assume that the first m columns of A are linearly independent. However, by the assumption that A is of full rank and $m \leq n$, we can always find an n by n permutation matrix Π such that

$$A\Pi = \widehat{A}$$

and the first m columns of $\widehat{A}$ are linearly independent. Letting

$$\widehat{H} = \Pi^T H \Pi,$$

we solve

$$\underbrace{\begin{bmatrix} \widehat{H} & \widehat{A}^T \\ \widehat{A} & -C \end{bmatrix}}_{\widehat{\mathcal{H}}} \begin{bmatrix} \widehat{x} \\ y \end{bmatrix} = \begin{bmatrix} \Pi^T b \\ d \end{bmatrix} \tag{8.0.1}$$

and set

$$x = \Pi\widehat{x}.$$

8.1 Desirable properties of the permutation

The choice of permutation, Π, such that the first m columns of $\widehat{A} = A\Pi$ are linearly independent is clearly not unique. Let us define $\widehat{A}_1 \in \mathbb{R}^{m\times m}$ and $\widehat{A}_2 \in \mathbb{R}^{m\times(n-m)}$ in a similar manner to that of A_1 and A_2, i.e.

$$A\Pi = \widehat{A} = \begin{bmatrix} \widehat{A}_1 & \widehat{A}_2 \end{bmatrix}. \tag{8.1.1}$$

We need to consider what sort of properties of $\widehat{\mathcal{H}}$ we would like to be induced from our choice of Π. Let us firstly consider the case of $C = 0$.

8.1.1 Permutations for the case $C = 0$

When applying the projected preconditioned conjugate gradient method (PPCG), Algorithm 5.1.3, to solve systems of the form (8.0.1) we know that the convergence of the method is determined by the $n - m$ eigenvalues λ defined by the generalized eigenvalue problem

$$\widehat{Z}^T\widehat{H}\widehat{Z}\widehat{x}_z = \lambda\widehat{Z}^T\widehat{G}\widehat{Z}\widehat{x}_z, \tag{8.1.2}$$

where $\widehat{Z}$ is an n by $n - m$ basis for the nullspace of $\widehat{A}$ and the preconditioner

$$\widehat{K} = \begin{bmatrix} \widehat{G} & \widehat{A}^T \\ \widehat{A} & 0 \end{bmatrix}$$

is used, see Section 5.2. Using the fundamental basis $\widehat{Z}$ (5.2.5) we find that

$$\begin{aligned} \widehat{Z}^T\widehat{G}\widehat{Z} &= \widehat{G}_{22} + \widehat{R}^T\widehat{G}_{21}^T + \widehat{G}_{21}\widehat{R} + \widehat{R}^T\widehat{G}_{11}\widehat{R} \\ \text{and}\quad \widehat{Z}^T\widehat{H}\widehat{Z} &= \widehat{H}_{22} + \widehat{R}^T\widehat{H}_{21}^T + \widehat{H}_{21}\widehat{R} + \widehat{R}^T\widehat{H}_{11}\widehat{R}, \end{aligned}$$

where $\widehat{R} = -\widehat{A}_1^{-1}\widehat{A}_2$. In Theorems 5.2.5–5.2.7 we saw that, under the appropriate assumptions, the number of distinct eigenvalues is bounded from above by an expression involving the rank of $\widehat{A}_2$, so one possible requirement of Π would be to minimize the rank of $\widehat{A}_2$.

When using Algorithm 2.3.1 to solve the mixed constraint optimization problems of the form given in Section 2.3, we saw that some of the entries on the diagonal of H grow like $\mathcal{O}(\mu_k^{-1})$ as $\mu_k \to 0$ and the others will be $\mathcal{O}(1)$ for general quadratic programming problems, see Section 2.4.2.1. Another possibility would be for Π to take this into account. Suppose that $\widehat{H}_{22}$ contains the diagonal entries of H which grow like $\mathcal{O}(\mu_k^{-1})$, but the diagonal entries of

$\widehat{H}_{11}$ are $\mathcal{O}(1)$. Using a preconditioner of the form used in Theorem 5.2.5 will result in the $n-m$ eigenvalues of (8.1.2) clustering around 1 as we grow close to the optimal value of x. We may have to slightly reshuffle the permutation to obtain a nonsingular $\widehat{A}_1$, but we hope that the majority of the large diagonal entries can be kept in $\widehat{H}_{22}$. By clustering the eigenvalues of (8.1.2) around 1 we hope that the number of iterations required by the PPCG method will be greatly reduced [5, Section 1.3].

In using our implicit factorization preconditioners we are required to solve systems involving $\widehat{A}_1$. From Chapter 3 we know that choosing $\widehat{A}_1$ to be sparse and well conditioned would be an advantage. The problem of finding the sparsest $\widehat{A}_1$ is NP complete [16, 17], so we just wish to find a sparse $\widehat{A}_1$ but not necessarily the sparsest. We also wish for the condition number of $\widehat{A}_1$ to be relatively low.

We can therefore conclude that ideally our permutation Π should

- produce a sparse and well conditioned $\widehat{A}_1$,
- produce a low rank $\widehat{A}_2$,
- move the largest diagonal entries of H into $\widehat{H}_{22}$

Let us consider the different possibilities for carrying out some of these ideal requirements. We will firstly consider how the condition number of $\widehat{A}_1$ can affect the number of iterations carried out by the PPCG method when used inside Algorithm 2.3.1.

8.1.1.1 The effect of the condition number and sparsity of $\widehat{A}_1$

Some of the problems in the `CUTEr` collection of quadratic programming problems [47] already have the first m columns of A linearly independent, for example, the `CVXQP` problems all satisfy this. We will compare the condition number and sparsity of $\widehat{A}_1$ when we use different permutations that are easily available within MATLAB®. We terminate the loop in Algorithm 2.3.1 when $\|Ax_k - d\|_2 < 10^{-8}$ and $\mu_k < 10^{-8}$. The PPCG method uses a stopping tolerance of 10^{-10}. Both algorithms are implemented in MATLAB® with an interface to the fortran code `MA57` when symmetric systems are factored for the implicit-factorization. Algorithm 2.3.1 is implemented using Mehrotra's predictor-corrector method to find the search direction, Section 2.3.2. The

Schilders factorization is used to produce a preconditioner of the form considered in Theorem 5.2.5. We will consider three different methods for generating the permutation Π, namely

- None: $\Pi = I$;
- LU: Π is generated using the MATLAB® command `[l,u,Π,q]=lu(A',0.5)`;
- QR: Π is generated using the MATLAB® command `[q,r,Π]=qr(full(A))`.

See Appendix F for descriptions of these functions.

We record the following details when using these different permutations:

- $\kappa(\widehat{A}_1)$: The condition number of $\widehat{A}_1$;
- $nnz(\widehat{A}_1)$: The number of nonzero entries that $\widehat{A}_1$ has;
- k : The number of interior point iterations carried out;
- Total PPCG Its 1: The total number of PPCG iterations used to find the affine directions, Section 2.3.2;
- Total PPCG Its 2: The total number of PPCG iterations used to find the search directions, Section 2.3.2;
- Total CPU Time: The total CPU time required to solve the quadratic programming problem in seconds;
- % Permutation Time: The percentage of the total CPU time used to find Π.

The results for different quadratic programming problems are given in Tables 8.1–8.3. If the condition number and number of nonzero entries is given for a particular $\widehat{A}_1$ but no further results, then the interior point method failed to converge. If no results are given for a particular permutation, then the resulting $\widehat{A}_1$ is singular.

We firstly observe that the different permutations can make a large difference to the total number of interior point iterations required to solve the problem. We also observe that even when the number of interior point iterations is roughly unchanged, the total number of PPCG iterations can vary greatly. Comparing the examples in Tables 8.1–8.3 for which the first m columns of A

	CVXQP1_M ($n = 1000$, $m = 500$)			CVXQP2_M ($n = 1000$, $m = 250$)		
Permutation	None	LU	QR	None	LU	QR
$\kappa(\widehat{A}_1)$	71279	4165	1854	964	142	33
$nnz(\widehat{A}_1)$	1048	996	1298	411	309	428
rank$(\widehat{A}_2)$	200	200	200	200	218	218
k	22	15	11	26	12	12
Total PPCG Its 1	7995	6125	1024	2490	2652	505
Total PPCG Its 2	8200	6130	1060	3787	2761	514
Total CPU Time	56.2	39.7	9.07	25.9	20.1	6.22
% Permutation Time	0	0.01	5.8	0	0.05	4.0

Table 8.1: The effect of different permutations on the number of iterations and the time to solve the mixed constraint quadratic programming problem.

	CVXQP3_M ($n = 1000$, $m = 750$)			KSIP ($n = 1021$, $m = 1001$)		
Permutation	None	LU	QR	None	LU	QR
$\kappa(\widehat{A}_1)$	$\approx 10^9$	58315	5111	—	1	850
$nnz(\widehat{A}_1)$	1048	1880	2147	—	1001	4998
rank$(\widehat{A}_2)$	100	100	100	—	20	20
k	—	—	10	—	15	16
Total PPCG Its 1	—	—	875	—	51	87
Total PPCG Its 2	—	—	904	—	56	86
Total CPU Time	—	—	8.36	—	12.5	16.1
% Permutation Time	—	—	12.6	—	0.08	13.0

Table 8.2: The effect of different permutations on the number of iterations and the time to solve the mixed constraint quadratic programming problem.

	MOSARQP1 ($n = 3200$, $m = 700$)			PRIMAL1 ($n = 410$, $m = 85$)		
Permutation	None	LU	QR	None	LU	QR
$\kappa(\widehat{A}_1)$	167	4	3.5	—	1	191
$nnz(\widehat{A}_1)$	3372	700	1953	—	85	3078
rank$(\widehat{A}_2)$	700	700	700	—	85	85
k	21	14	22	—	13	13
Total PPCG Its 1	13004	10089	17574	—	1821	1278
Total PPCG Its 2	19298	12509	16690	—	1804	1301
Total CPU Time	796.7	182.6	715.1	—	6.85	6.39
% Permutation Time	0	0.01	0.55	—	0.15	0.16

Table 8.3: The effect of different permutations on the number of iterations and the time to solve the mixed constraint quadratic programming problem.

are originally linearly independent, we observe that there is always an advantage in reordering the columns of A. The permutation obtained using the QR factorization does not take advantage of the sparse format of A : this permutation will be inefficient to generate when the sizes of m and n are large and A is sparse. For example, 13% of the overall time spent in solving the `KSIP` problem was in finding the permutation, compared to just 0.1% when the `LU` function was used to generate Π. The `KSIP` problem is still relatively small, but for larger problems the inefficiency of the `QR` will be more exaggerated in terms of memory requirements and time.

The large difference in the number of PPCG iterations (and hence the total CPU time) does not appear to be completely down to the condition number and sparsity of $\widehat{A}_1$ or the rank of $\widehat{A}_2$. Let us plot what happens to the diagonal entries of $\widehat{H}$ as the interior point method progresses and see if the iterations numbers can be explained by the distribution of the large diagonal entries.

Figures 8.1–8.3 show the diagonal entries of $\widehat{H}$ at each iteration k of the interior point method when applied to the `CVXQP1_M` quadratic programming problem from the `CUTEr` test set [47]. We see that the large diagonal values are distributed in an almost uniform manner when no permutation is used or the permutation is generated with the `LU` function, Figures 8.1 and 8.2. When the permutation is generated using the `QR` function, the majority of the large diagonal values have been moved into $\widehat{H}_{22}$, Figure 8.3. We will therefore expect many of the eigenvalues to be clustered around 1 and, hence,

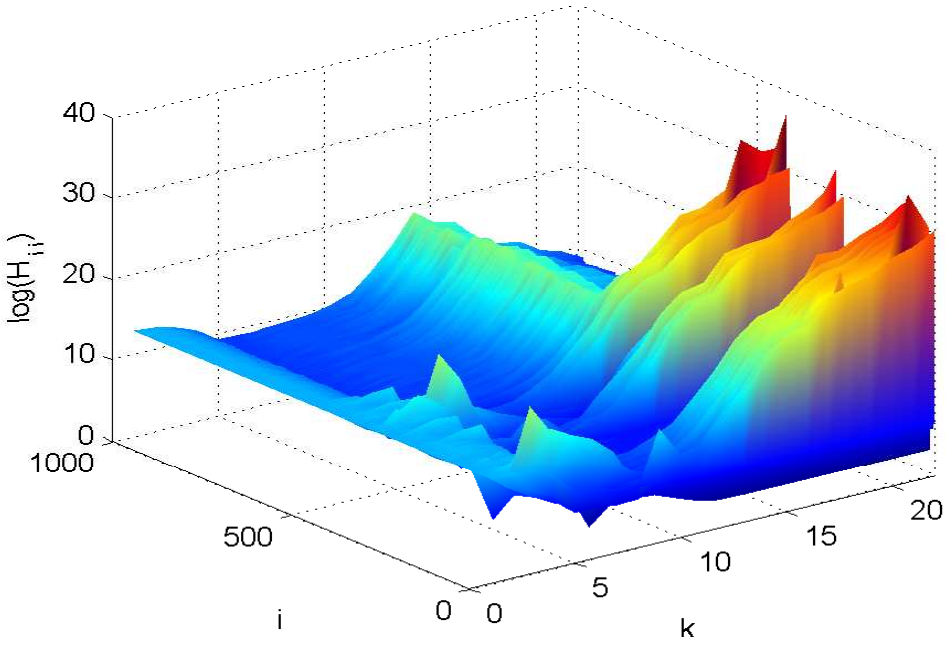

Figure 8.1: `CVXQP1_M`: Diagonal entries of $\widehat{H}$ as the interior point method progresses when no permutation is used.

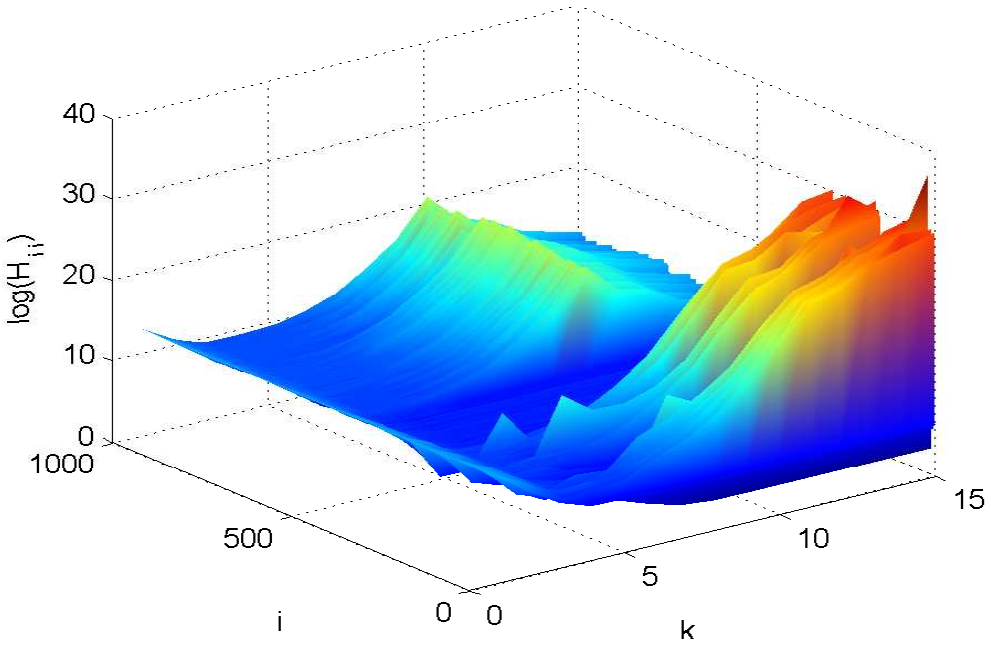

Figure 8.2: `CVXQP1_M`: Diagonal entries of $\widehat{H}$ as the interior point method progresses when Π is derived using the `LU` function.

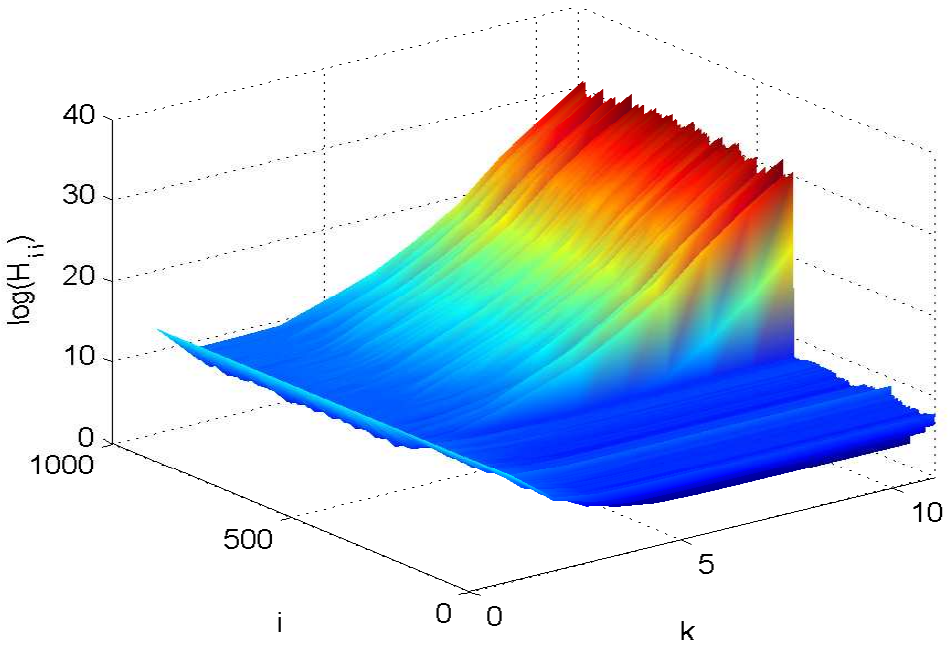

Figure 8.3: `CVXQP1_M`: Diagonal entries of $\widehat{H}$ as the interior point method progresses when Π is derived using the `QR` function.

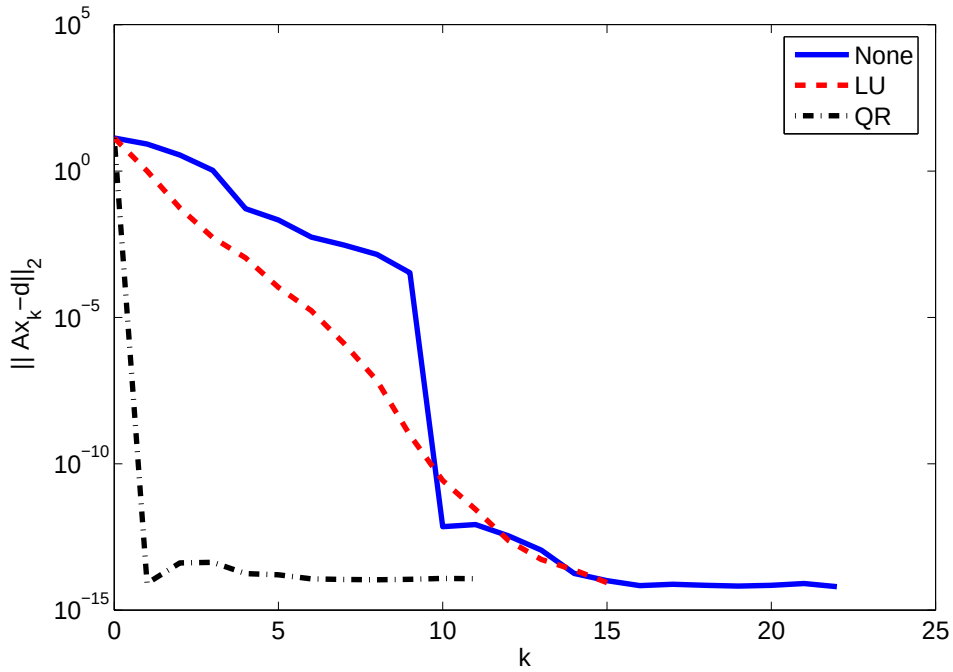

Figure 8.4: `CVXQP1_M`: Comparison of the convergence of $\|Ax_k - d\|_2$ for the different permutations.

a significant decrease in the number of PPCG iterations required over the other two permutations: this is what we see in practice.

We also observe that when no permutation or the `LU` generated permutation are used, the magnitude of the diagonal entries of $\widehat{H}$ start to increase and then some drop again. The ill-conditioning of $\widehat{A}_1$ is resulting in the convergence of $Ax_k - d$ to 0 being delayed compared to when the permutation generated by the `QR` factorization is used. We compare the values of $\|Ax_k - d\|_2$ in Figure 8.4. The first two permutations are trying to minimize the problem $\frac{1}{2}x^T Hx + c^T x$ subject to $x \geq 0$ before the requirement $Ax = d$ is close to being obtained, so some of the entries in x are firstly moving towards 0 (giving large diagonal entries in $\widehat{H}$). These entries then move away from 0 in order for the constraint $Ax = d$ to be satisfied. In comparison, the permutation generated by the `QR` function produces a much more favourably conditioned $\widehat{A}_1$ and just one interior point iteration is required for the constraint $Ax = d$ to be closely satisfied. This means that the entries of x which wrongly started moving towards 0 with the other permutations will not do this. The total number of interior point iterations required is also reduced for this reason.

8.1.1.2 The effect of the location of the diagonal values of $\widehat{H}$

In the previous section we saw that not only is it important to keep the condition number of $\widehat{A}_1$ low (and also aim to reduce the rank of A_2), but it's the distribution of the diagonal entries of $\widehat{H}$ that makes a large (and dominating) difference to the total time used to solve a quadratic programming problem with mixed constraints. We will firstly use the preconditioner as described in Theorem 5.2.5, and then consider other choices for the constraint preconditioner afterwards.

From Section 8.1.1 we know that we would like to try to move the large diagonal entries of H into $\widehat{H}_{22}$, but we'd also like to obtain an $\widehat{A}_1$ which is well-conditioned. For convenience, we'd also like the method to be cheap to apply and to use built in MATLAB® functions. Let us try the following method:

1. Find a permutation Π_1 that sorts the diagonal entries of H;
2. Find a permutation Π_2 so that the first m columns of $\widehat{A} = A\Pi_1\Pi_2$ are linearly independent but the effect of Π_1 isn't completely lost;
3. Set $\Pi = \Pi_1\Pi_2$.

We shall compare the differences between choosing Π_1 to firstly sort the diagonal entries of H into ascending and descending order: to do this we will use the MATLAB® commands

```
[yy,ii]=sort(diag(H));
```

and

```
[yy,ii]=sort(-diag(H));
```

respectively. We then set $\Pi_1 =$ `I(:,ii)`. In our tables of results we shall refer to these methods of generating the permutation Π as `LUA` and `LUD` respectively.

To find Π_2 we use the `LU` function applied to $\texttt{Ap} = A\Pi_1$:

```
[l,u,P]=lu(Ap',0.5).
```

We give results for both of these strategies, and an additional strategy. In this additional strategy we try both the above strategies and compare them to see which one is "best" at moving the large entries into $\widehat{H}_{22}$: this is a hybrid of the other two methods. Specifically, if the first strategy returns $\widetilde{\Pi}$ and the second strategy returns $\overline{\Pi}$, then we carry out the following steps:

$d_1 = \texttt{diag}(\widetilde{\Pi}^T H \widetilde{\Pi})$
$d_2 = \texttt{diag}(\overline{\Pi}^T H \overline{\Pi})$
$r_1 = \frac{\texttt{mean}(d_1(m+1{:}m+n))}{\texttt{mean}(d_1(1{:}m))}$
$r_2 = \frac{\texttt{mean}(d_2(m+1{:}m+n))}{\texttt{mean}(d_2(1{:}m))}$
if $r_1 > r_2$ **then**
  $\Pi = \widetilde{\Pi}$
else
  $\Pi = \overline{\Pi}$
end if

In our tables of results the above method is referred to as `LUH`. All of the results are run on the same machine and to the same specifications as in the previous section. However, this time we generate a new permutation for each interior point iteration. The results for different quadratic programming problems are given in Tables 8.4–8.6.

We firstly observe that trying to take the diagonal entries of H into account when generating Π is advantageous over just using the `LU` method in the previous section. We also observe that the `LUD` method is generally better than the `LUA` in generating a permutation that reduces the number of PPCG iterations required, but the difference is normally fairly small; this appears to be due to the `LU` function moving columns to the end of A when they are swapped out

	CVXQP1_M ($n = 1000$, $m = 500$)			CVXQP2_M ($n = 1000$, $m = 250$)		
Permutation	LUA	LUD	LUH	LUA	LUD	LUH
k	13	12	12	12	12	12
Total PPCG Its 1	1700	1244	1375	375	308	311
Total PPCG Its 2	1789	1241	1439	382	308	309
Total CPU Time	14.6	11.2	12.9	5.44	4.70	4.95
% Permutation Time	3.5	3.1	7.0	2.2	2.3	6.1

Table 8.4: The effect of different permutations on the number of iterations and the time to solve the mixed constraint quadratic programming problem.

	CVXQP3_M ($n = 1000$, $m = 750$)			KSIP ($n = 1021$, $m = 1001$)		
Permutation	LUA	LUD	LUH	LUA	LUD	LUH
k	12	10	10	12	13	12
Total PPCG Its 1	1190	1020	998	34	37	35
Total PPCG Its 2	1259	1069	991	39	45	40
Total CPU Time	12.8	10.4	12.2	15.8	17.6	21.7
% Permutation Time	18.8	14.7	30.1	35.4	36.6	52.6

Table 8.5: The effect of different permutations on the number of iterations and the time to solve the mixed constraint quadratic programming problem.

	MOSARQP1 ($n = 3200$, $m = 700$)			PRIMAL1 ($n = 410$, $m = 85$)		
Permutation	LUA	LUD	LUH	LUA	LUD	LUH
k	66	12	12	13	13	13
Total PPCG Its 1	82833	3840	3932	1480	1374	1475
Total PPCG Its 2	98205	4008	4008	1480	1386	1448
Total CPU Time	2986	58.7	58.7	7.62	7.25	7.91
% Permutation Time	0.04	0.39	0.39	4.1	4.7	8.1

Table 8.6: The effect of different permutations on the number of iterations and the time to solve the mixed constraint quadratic programming problem.

of position. There is a big difference between using the LUA and LUD methods for the MOSARQP1 problem. In Figures 8.5—8.8 we compare the distribution of the diagonal values for the different LU* permutations. The permutation generated by the LUA method is moving many of the large diagonal entries of H into $\widehat{H}_{11}$ and not $\widehat{H}_{22}$, so the average number of PPCG iterations required per interior point iteration is a lot higher than when we use the other iterations. The LUD and LUH are successfully moving the majority of the large diagonal entries into $\widehat{H}_{22}$; there are still some large entries in $\widehat{H}_{11}$ but far fewer than when the LU and LUA methods are used to generate the permutation. For this larger problem there is little difference in using the hybrid method LUH and LUD methods. For the smaller problems there is generally a small overhead when using the LUH method, particularly when $n - m$ is small, but in practice much larger problems will be solved using the PPCG method, so in Table 8.7 we compare the different methods for a larger test problem from the CUTEr quadratic programming test set. We observe the advantage of using the LUH method over the other permutations.

	GOULDQP2 ($n = 19999$, $m = 9999$)			
Permutation	LU	LUA	LUD	LUH
k	8	15	29	35
Total PPCG Its 1	12305	56862	23907	12660
Total PPCG Its 2	12431	58027	28581	8711
Total CPU Time	2092	7264	2859	1350
% Permutation Time	0.001	0.02	0.13	0.80

Table 8.7: The effect of different permutations on the number of iterations and the time to solve the mixed constraint quadratic programming problem.

8.1.1.3 Numerical examples

There is generally little advantage in trying to get the PPCG method to converge with a high accuracy during the first few iterations of the interior point method. Instead, whilst $\mu > 10^{-1}$ we will use a stopping tolerance of 10^{-4}, but when $\mu \leq 10^{-1}$ a stopping tolerance of 10^{-10} is used. We shall compare the behaviour of different preconditioning strategies for solving quadratic programming problems of the form

$$\min_x \frac{1}{2} x^T Q x + c^T x \quad \text{subject to} \quad Ax - d = 0 \text{ and } x \geq 0.$$

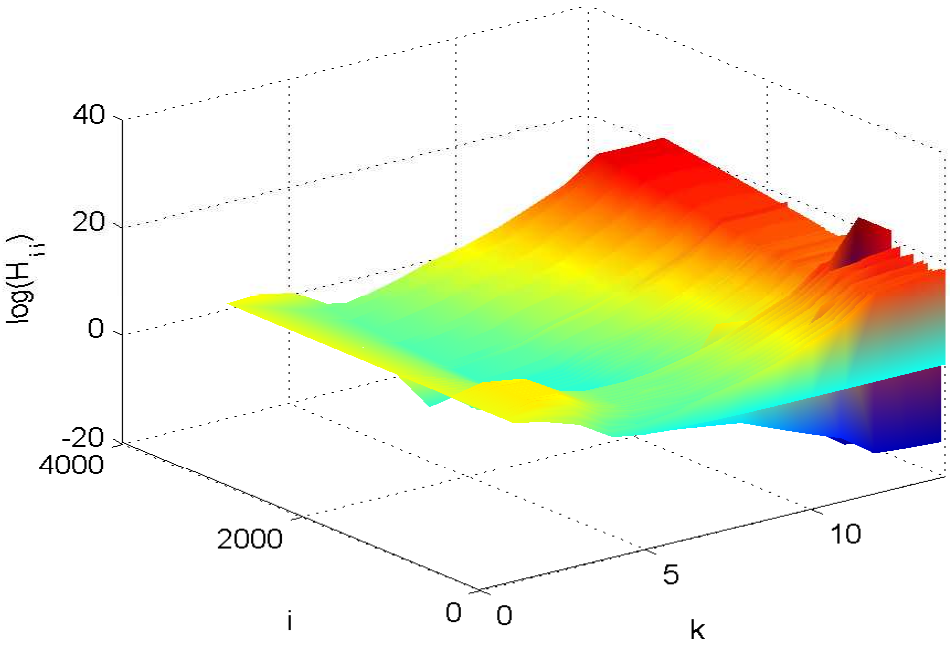

Figure 8.5: `MOSARQP1:` Diagonal entries of $\widehat{H}$ as the interior point method progresses when Π is derived using the `LU` method.

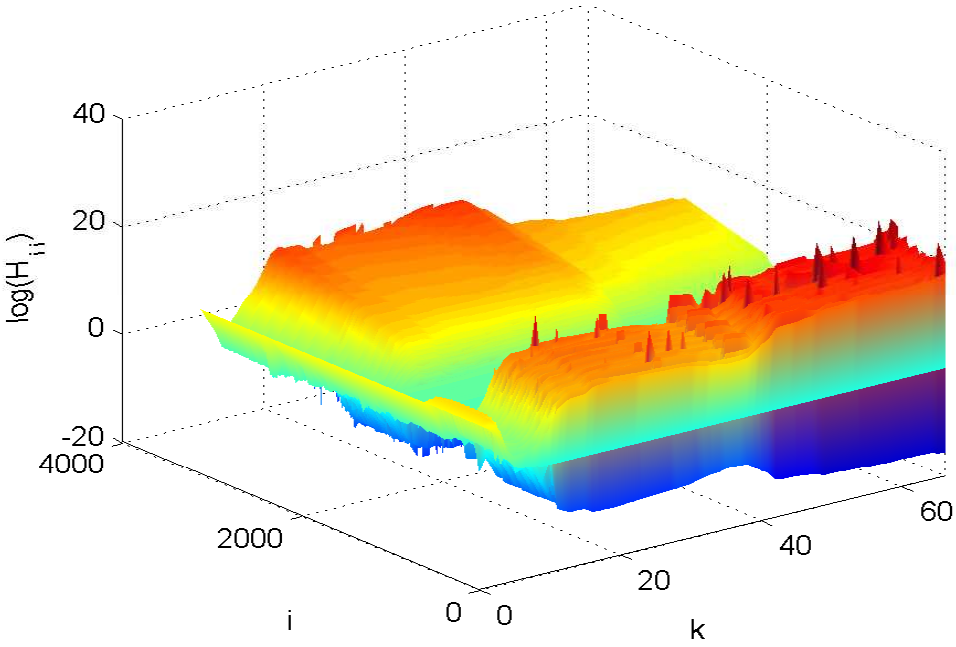

Figure 8.6: `MOSARQP1:` Diagonal entries of $\widehat{H}$ as the interior point method progresses when Π is derived using the `LUA` method.

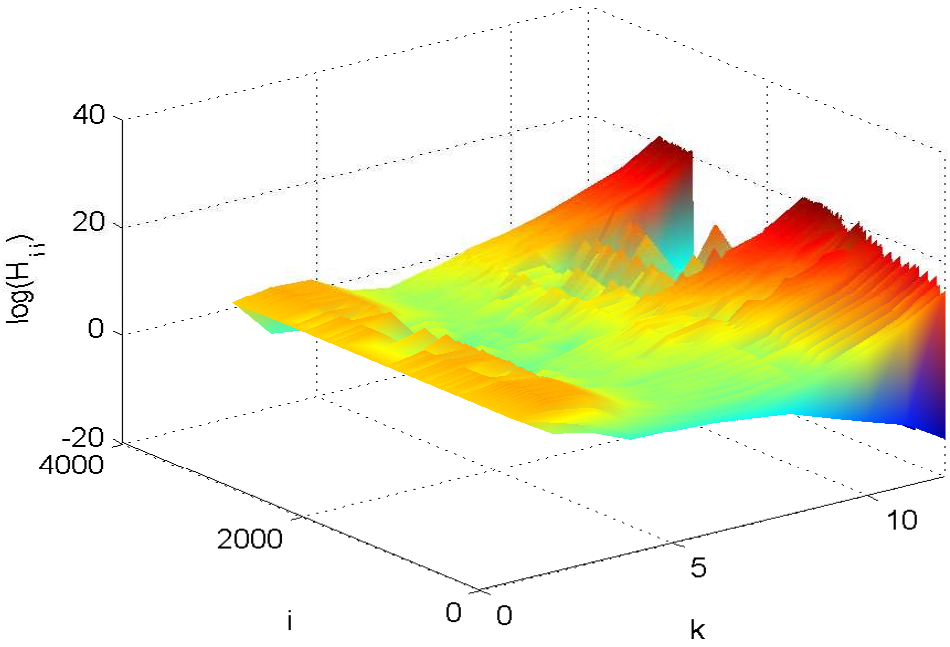

Figure 8.7: `MOSARQP1:` Diagonal entries of $\widehat{H}$ as the interior point method progresses when Π is derived using the `LUD` method.

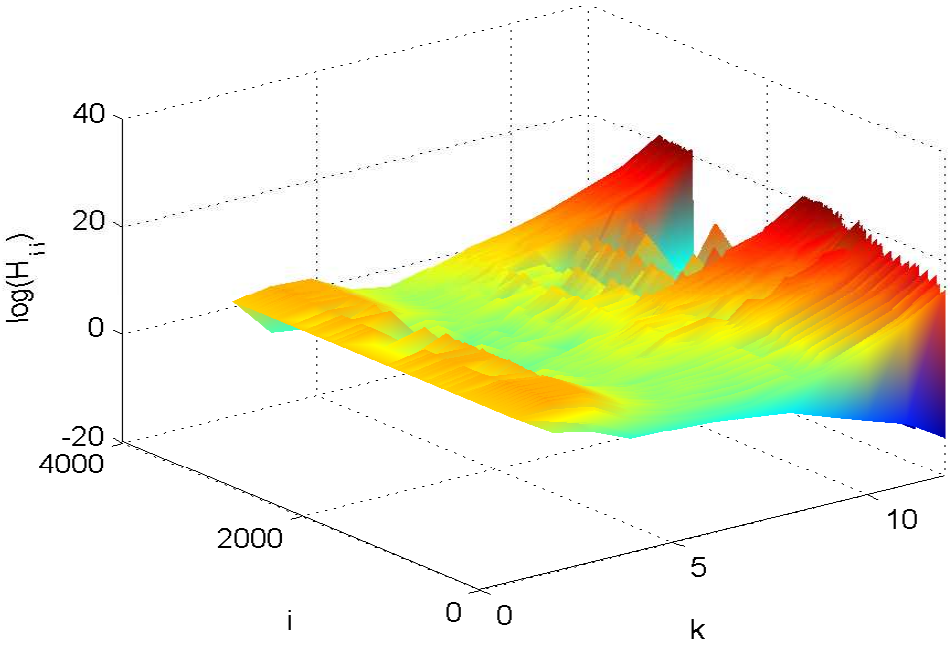

Figure 8.8: `MOSARQP1:` Diagonal entries of $\widehat{H}$ as the interior point method progresses when Π is derived using the `LUH` method.

As in all our previous numerical tests, we shall consider problems for the `CUTEr` test set [47]. The following preconditioners will be considered:

`Expl1` No permutation used and `MA57` is used to factor the constraint preconditioner with $G = \texttt{diag}\{H\}$. This is the same as the AS-preconditioner considered in [9].

`Expl2` No permutation used and `MA57` is used to factor the constraint preconditioner with $G = I$. This is the same as the preconditioner presented in [70].

`Impl1` Permutation of the form `LUH` is used with a preconditioner generated with the Schilders factorization where $G_{11} = 0$, $G_{21} = 0$ and $G_{22} = H_{22}$.

`Impl2` Permutation of the form `LUH` is used with a preconditioner generated with the Schilders factorization where $G_{11} = 0$, $G_{21} = 0$ and $G_{22} = \texttt{diag}\{H_{22}\}$.

`Impl3` Permutation of the form `LUH` is used with a preconditioner generated with the Schilders factorization where $G_{11} = 0$, $G_{21} = H_{21}$ and $D_2 = H_{22}$.

`Impl4` Permutation of the form `LUH` is used with a preconditioner generated with the Schilders factorization where $G_{11} = 0$, $G_{12} = 0$ and $G_{22} = I$.

For many of quadratic programming problems, the submatrix H_{22} will not be diagonal, so we will expect the preconditioner `Impl2` to use more PPCG iterations than when the preconditioner `Impl1` is used. However, the matrix D_2 in the Schilders factorization will take longer to be factorized when `Impl1` is used, so it is difficult to predict which of these two preconditioners will perform better (i.e. use the least CPU time) overall.

In exact arithmetic, the preconditioners `Impl1` and `Impl3` will produce the same values of $Z^T H Z$ and $Z^T G Z$, Section 5.2.1. However, the latter preconditioner will carry out more operations, so we expect the timings to be longer for this. We might also expect there to be a bigger effect from rounding errors for the preconditioner `Impl3` compared to `Impl1`, so the number of PPCG iterations might also be larger.

As we approach optimality some of the diagonal entries of H will grow in size: by using the `LUH` permutation we hope to move the majority of these large entries into H_{22}. The preconditioner `Impl4` does not take any of these

large entries into account, so we expect the number of PPCG iterations to grow very large as we approach optimality. We therefore predict that all of the other implicit preconditioners will perform a lot better than when `Impl4` is used.

By setting $G = \texttt{diag}\{H\}$ in `Expl1`, we replicate one of the preconditioners considered in [9]. As the problems to be solved grow large, we will expect the choice of preconditioner `Impl1` to be better than `Expl1`.

There is no consideration of the diagonal entries of H when the preconditioner `Expl2` is used. We therefore predict that it will require a large number of PPCG iterations and will produce very poor CPU timings compared to the other preconditioners.

Table 8.8 contains results for some problems found in the `CUTEr` test set. More results from the `CUTEr` test set can be found in Appendix G. We observe that for small problems there is little advantage in using the Schilders implicit factorizations compared to the explicit factorizations (in fact, the explicit factorizations `Expl1` are normally favourable in these cases). However, as the dimensions of the problems being solved increase the implicit factorizations (excluding `Impl4`) appear to become preferable. For some of the problems, the dimensions and memory requirements have become large enough for the explicit factorizations to be unusable because of the computer used not having enough memory. Clearly, whichever machine is used, we will encounter problems for which the explicit factorizations will be infeasible to use for memory reasons, but implicit factorizations can still be extremely effective.

Table 8.8: CUTEr QP problems—Number of iterations used

Problem		Expl1	Expl2	Impl1	Impl2	Impl3	Impl4
AUG3DCQP_M	k	11	11	17	17	15	16
$n = 3873$	Total Its 1	11	9869	1475	1475	1297	20762
$m = 1000$	Total Its 2	11	9689	1568	1568	1336	20507
	Total CPU	23.02	23.11	26.06	26.06	35.43	253.90
	% Permutation	0	0	1.80	1.80	1.35	0.18
CONT1-10	k	memory	—	6	6	6	6
$n = 10197$	Total Its 1	—	—	37	37	37	56
$m = 9801$	Total Its 2	—	—	38	38	38	56
	Total CPU	—	—	67.92	67.92	80.27	70.30
	% Permutation	—	—	50.25	50.25	42.74	48.58
CVXQP2_M	k	12	14	13	13	13	20
$n = 1000$	Total Its 1	260	7310	256	656	256	13859
$m = 250$	Total Its 2	263	6945	265	675	260	13815

Table 8.8: CUTEr QP problems—Number of iterations used (continued)

Problem		Expl1	Expl2	Impl1	Impl2	Impl3	Impl4
	Total CPU	7.24	71.05	4.04	7.18	4.98	132.39
	% Permutation	0	0	4.70	2.65	3.61	0.19
DUALC2	k	7	8	41	16	10	10
$n = 235$	Total Its 1	23	40	244	107	34	40
$m = 229$	Total Its 2	24	39	277	108	33	38
	Total CPU	0.65	0.71	3.56	2.36	0.99	1.04
	% Permutation	0	0	6.18	4.23	3.03	3.85
STCQP2	k	16	16	16	16	16	16
$n = 8193$	Total Its 1	63	3645	16	63	16	3541
$m = 4095$	Total Its 2	71	3626	16	71	16	3508
	Total CPU	9.47	165.79	8.38	8.64	9.08	107.92
	% Permutation	0	0	18.38	18.26	16.52	1.33

As in previous chapters, we compare the preconditioning strategies using a performance profile, Figure 8.9. The data from Appendix G is used for this performance profile but any problems with $n < 1000$ are now excluded because these problems are too small to be effective tests for the various preconditioners. We observe that the `Impl1` preconditioner generally outperforms all of the other choices of preconditioner. As expected, the `Expl2` and `Impl4` preconditioners perform badly because they do not take into account the large entries that are forming on the diagonal of H.

We noted that `Impl1` and `Impl3` are different implementations of the same choice of G. We were expecting `Impl1` to be more efficient and this is confirmed by our results. `Expl1` and `Impl2` perform very similarly for many of the problems, but as the problems grow large the implicit factorization generally becomes preferable out of the two because it does not suffer from the memory problems. We therefore conclude that the implicit-factorization preconditioner `Impl1` is preferable for such problems.

8.1.2 Permutations for the case $C \neq 0$

Similarly to the case of $C = 0$, we can use Corollaries 5.4.15—5.4.17 to find that ideally our permutation should produce:

- a sparse and well conditioned $\widehat{A}_1$;
- a low rank $\widehat{A}_2$.

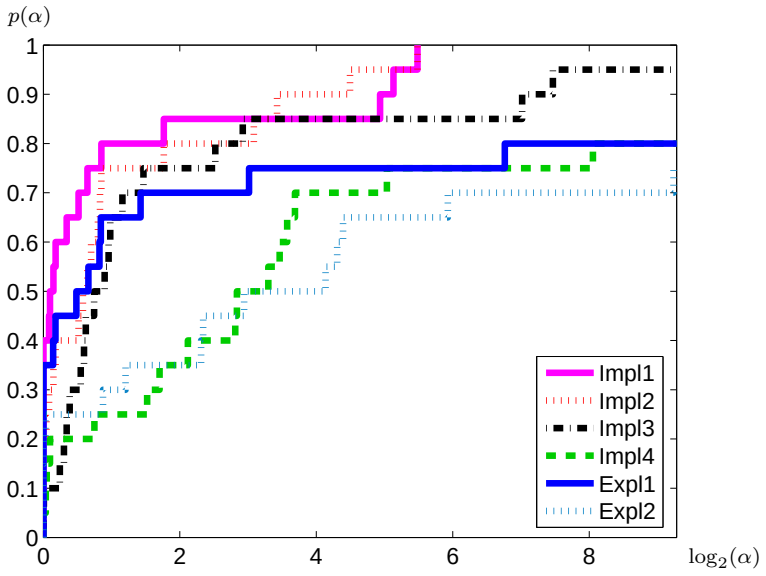

Figure 8.9: Performance profile, $p(\alpha)$: CPU time (seconds) to solve QP programming problems.

We no longer worry about the ordering of the diagonal entries of H because an active set strategy is used within the interior point methods when solving inequality constrained problems, as described in Section 2.2.

Let us consider how we might perform a permutation to reduce the rank of $\widehat{A}_2$ but maintain the nonsingularity of $\widehat{A}_1$. The approach we shall use is based on the computation of a block triangular form of A^T. This method takes a matrix A^T and finds a row and column permutation such that

$$\widehat{A}^T = \begin{bmatrix} \widehat{A}_h & X & X \\ 0 & \widehat{A}_s & X \\ 0 & 0 & \widehat{A}_v \end{bmatrix},$$

where $\widehat{A}_h$ is underdetermined, $\widehat{A}_s$ is square, $\widehat{A}_v$ is overdetermined, and Xs denote possibly nonzero matrices of appropriate dimension. However, since A has full row rank in our case, there will not exist such a $\widehat{A}_h$. We hope that for many of our test problems the dimensions of $\widehat{A}_s$ will be large so that the rank of $\widehat{A}_2$ will be small.

Algorithms for computing such row and column permutations are generally based on a canonical decomposition for bipartite graphs as discovered by Dulmage and Mendelsohn [26]. Essentially, a subset of the bipartite graph corresponding to A^T is found and this is then used to divide the rows into three

groups and the columns into a further three groups. These groups then give us the rows and columns used in $\widehat{A}_h$, $\widehat{A}_s$ and $\widehat{A}_v$. Further details can be found in [68]. If τ is the number of nonzero entries in A^T, then such a decomposition can be found in $\mathcal{O}(\tau\sqrt{n})$ flops. The function `dmperm` can be used to compute such a permutation in MATLAB®.

Unfortunately, the resulting $\widehat{A}_1$ is not guaranteed to be nonsingular so, as in the previous section where we initially carried out a permutation to order the diagonal entries of H, we might still need to carry out a further permutation (using the `lu` function as before) to guarantee that $\widehat{A}_1$ is nonsingular. We shall call this the BTF permutation. Tables 8.9—8.11 compare the permutations LU, QR and BTF for some of the test problems from the `CUTEr` test set. These problems are transformed to give inequality constrained problems of the form

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^T\left(Q + 1.1I\right)x + b^T x \text{ subject to } Ax \geq d,$$

and the interior point method Algorithm 2.2.1 is used in conjunction with an active set strategy to solve these problems. The PPCG method, Algorithm 5.3.2, is used to solve the saddle point problems arising within the interior point method: we use Family 2 from Table 7.1 with $B_{11} = 0$, $B_{22} = Q_{22} + 1.1I$, $P_{22} = I$ and $P_{31} = I$ as the implicit factorization constraint preconditioner. The PPCG method is terminated when the residual has been reduced by a factor of 10^{-6} and we terminate the interior point method once $\left\|c(x^k) - \mu^k[Y^k]^{-1}e\right\|_2 < 10^{-10}$ and $\mu_k < 10^{-10}$ in Algorithm 2.2.1.

	`CVXQP1_M` ($n = 1000$, $m = 500$)			`CVXQP2_M` ($n = 1000$, $m = 250$)		
Permutation	LU	QR	BTF	LU	QR	BTF
k	120	37	34	25	24	24
Total PPCG Its	33369	5263	5348	8114	1111	1214
Total CPU Time	148.24	45.48	25.57	36.58	12.68	8.10
% Permutation Time	0.30	38.83	1.11	0.19	38.17	1.41

Table 8.9: The effect of different permutations on the number of iterations and the time to solve the inequality constrained quadratic programming problem.

We observe that the time spent in finding the solution is dramatically reduced when the BTF permutation is used compared to the LU permutation: this is down to the greatly reduced number of PPCG iterations. As expected, the QR permutation is too inefficient to use for the large problems. As a result

	GOULDQP2_S ($n = 699$, $m = 349$)			KSIP ($n = 1021$, $m = 1001$)		
Permutation	LU	QR	BTF	LU	QR	BTF
k	8	14	18	15	6	6
Total PPCG Its	165	286	447	76	22	11
Total CPU Time	0.95	4.29	2.42	13.89	16.85	7.27
% Permutation Time	1.05	60.84	1.65	1.58	65.22	21.05

Table 8.10: The effect of different permutations on the number of iterations and the time to solve the inequality constrained quadratic programming problem.

	MOSARQP1 ($n = 3200$, $m = 700$)			PRIMAL1 ($n = 410$, $m = 85$)		
Permutation	LU	QR	BTF	LU	QR	BTF
k	15	10	10	6	9	8
Total PPCG Its	185	120	105	130	161	156
Total CPU Time	5.45	40.09	3.71	0.60	0.99	0.90
% Permutation Time	1.65	90.15	2.96	5.00	12.12	15.56

Table 8.11: The effect of different permutations on the number of iterations and the time to solve the inequality constrained quadratic programming problem.

of these tests we will use the BTF permutation in the following numerical examples.

8.1.2.1 Numerical examples

We shall compare the behaviour of different preconditioning strategies for solving inequality constrained quadratic programming problems:

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T \left(Q + 1.1I\right) x + b^T x \text{ subject to } Ax \geq d.$$

The interior point method Algorithm 2.2.1 is used in conjunction with an active set strategy to solve these problems. The PPCG method, Algorithm 5.3.2, is used to solve the saddle point problems arising within the interior point method. As in all our previous tests, we shall consider problems from the CUTEr test to give us the matrices and vectors A, Q, b, and d. The PPCG method is terminated when the residual has been reduced by a factor of 10^{-6} and we terminate the interior point method once $\left\| c(x^k) - \mu^k [Y^k]^{-1} e \right\|_2 < 10^{-10}$ and $\mu_k < 10^{-10}$ in Algorithm 2.2.1.

The following preconditioners will be compared:

`Expl1` No permutation used and `MA57` is used to factor the constraint preconditioner with $G = \texttt{diag}\{H\}$.

`Expl2` No permutation used and `MA57` is used to factor the constraint preconditioner with $G = I$.

`Fam1a` Permutation of the form `BTF` is used with a preconditioner of the form Family 1 (Table 7.1) with $P_{31} = I$, $P_{33} = B_{33} = I$, and $P_{22} = B_{22} = I$.

`Fam1b` Permutation of the form `BTF` is used with a preconditioner of the form Family 1 (Table 7.1) with $P_{31} = I$, $P_{33} = B_{33} = I$, $P_{22} = I$, and $B_{22} = Q_{22} + 1.1I$.

`Fam1c` Permutation of the form `BTF` is used with a preconditioner of the form Family 1 (Table 7.1) with $P_{31} = I$, $P_{33} = B_{33} = I$, $P_{22} = I$, and $B_{22} = \texttt{diag}\{Q_{22}\} + 1.1I$.

`Fam2a` Permutation of the form `BTF` is used with a preconditioner of the form Family 2 (Table 7.1) with $P_{31} = B_{31} = I$, $B_{11} = 0$, $P_{22} = I$, and $B_{22} = Q_{22} + 1.1I$.

`Fam2b` Permutation of the form `BTF` is used with a preconditioner of the form Family 2 (Table 7.1) with $P_{31} = B_{31} = I$, $B_{11} = 0$, $P_{22} = I$, and $B_{22} = H_{22}$.

`Fam2c` Permutation of the form `BTF` is used with a preconditioner of the form Family 2 (Table 7.1) with $P_{31} = B_{31} = I$, $B_{11} = 0$, $P_{22} = I$, and $B_{22} = \texttt{diag}\{Q_{22}\} + 1.1I$.

`Fam2d` Permutation of the form `BTF` is used with a preconditioner of the form Family 2 (Table 7.1) with $P_{31} = B_{31} = I$, $B_{11} = 0$, and $P_{22} = B_{22} = I$.

`Fam3a` Permutation of the form `BTF` is used with a preconditioner of the form Family 3 (Table 7.1) with $P_{31} = B_{31} = I$, $G_{21} = Q_{21}$, and $G_{22} = Q_{22} + 1.1I$.

`Fam3b` Permutation of the form `BTF` is used with a preconditioner of the form Family 3 (Table 7.1) with $P_{31} = B_{31} = I$, $G_{21} = Q_{21}$, and $B_{22} = I$.

We expect that the preconditioners `Fam1b` and `Fam1c` will require fewer PPCG iterations than when `Fam1a` is used of there being no inclusion of H in G for the latter choice. `Fam1b` should use fewer PPCG iterations than `Fam1c`, but the latter will be more efficient to form and use.

Similarly, we expect `Fam2b` to use fewer iterations than `Fam2a,` but `Fam2a` will be far more efficient to form and solve. `Fam2c` is likely to use more iterations than `Fam2a`, but the efficiency in forming and applying the preconditioner may make it favourable. The preconditioner `Fam2d` does not take the entries of H into account, so we expect this to perform badly.

`Fam3a` takes more account of the entries of H in G than any of our other implicit factorization preconditioners. However, this will be at a cost of the efficiency. `Fam3b` will be more efficient than `Fam3a` to apply, but it will generally require more iterations.

For large problems, the explicit factorizations `Expl1` and `Expl2` may become unviable to use because of memory problems.

Table 8.12 contains results for some problems found in the `CUTEr` test set. More results from the `CUTEr` test set can be found in Appendix H. We observe that for small problems there is little advantage in using implicit factorization constraint preconditioners over the explicit factorizations (in fact, the explicit factorization `Expl1` is often preferable in this case). Looking at the tables, for larger problems the implicit factorizations appear to become preferable. As in previous chapters, we shall also use a performance profiling technique to compare the results. As in the previous section, we shall exclude any test problems with $n < 1000$ from the profiles.

A figure profiling all of the preconditioners can be found in Appendix H. We observe that the choices `Fam3a` and `Fam3b` are performing very poorly, so we have removed them in the profile found in Figure 8.10: there is a high cost in trying to reproduce H_{21} and H_{22} in G. The preconditioners generated by Family 1 are generally outperforming those of Family 2 (as was the case in Figures 7.1 and 7.2), Figure 8.11. The explicit factorization constraint preconditioners are generally being outperformed by the implicit factorizations generated by both Family 1 and Family 2. In Figure 8.12 we profile the explicit factorization preconditioners with just those generated through Family 1. Clearly, the implicit factorization constraint preconditioners `Fam1a`—`Fam1c` are outperforming both of the explicit factorization preconditioners. As predicted, `Fam1a` is performing the poorest out of those generated with Family 1.

Table 8.12: CUTEr QP problems—Number of iterations used

Problem		Expl1	Expl2	Fam1a	Fam1b	Fam1c	Fam2a	Fam2b	Fam2c	Fam2d	Fam3a	Fam3b
AUG3DC-	k	26	26	97	105	95	111	200	105	79	88	85
QP_M	Its	507	473	3258	3616	3052	3817	4522	3293	3046	2816	2769
	CPU	26.77	26.04	85.76	93.97	83.39	99.83	175.32	92.49	73.12	82.34	79.75
CONT1-10	k	17	17	25	18	24	27	54	24	25	25	25
	Its	835	677	2227	1517	2096	2015	1502	2061	2243	2082	2245
	CPU	9193	8973	13172	8960	12108	13478	25933	12542	13345	12647	12354
CVXQP2_M	k	24	27	27	24	24	24	23	25	26	—	—
	Its	2161	3596	4452	1233	3800	1214	1281	8331	4254	—	—
	CPU	13.73	14.62	12.71	8.25	10.60	8.10	8.32	23.02	12.34	—	—
DUALC2	k	200	13	200	27	200	200	200	52	7	82	13
	Its	409	2277	10431	3526	12773	17601	743	6882	851	12673	590
	CPU	10.38	3.01	22.78	5.75	26.73	31.69	8.84	10.62	1.32	25.83	1.56
STCQP2	k	74	74	73	73	73	73	74	73	74	93	75
	Its	1146	2216	4355	5541	5729	5485	5196	14471	4378	32656	5639
	CPU	702.32	807.47	316.91	396.21	332.36	398.94	434.13	450.43	324.74	2802.36	354.24

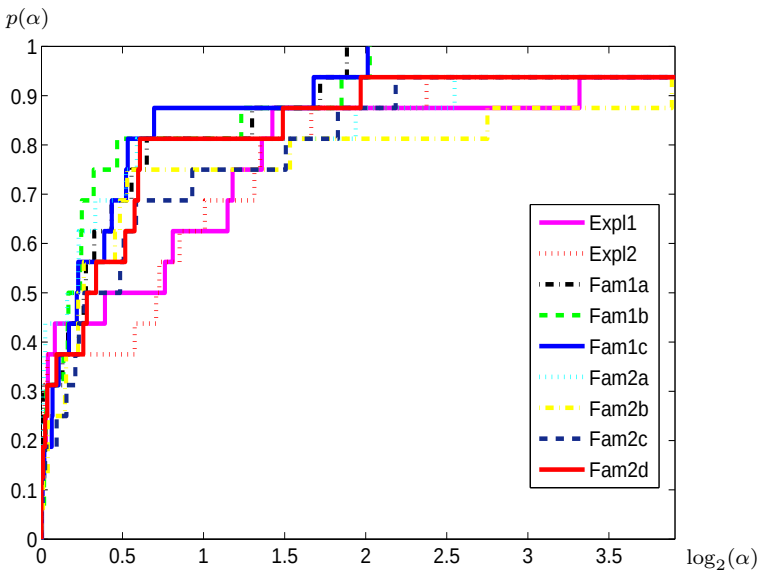

Figure 8.10: Performance profile, $p(\alpha)$: CPU time (seconds) to solve QP programming problems.

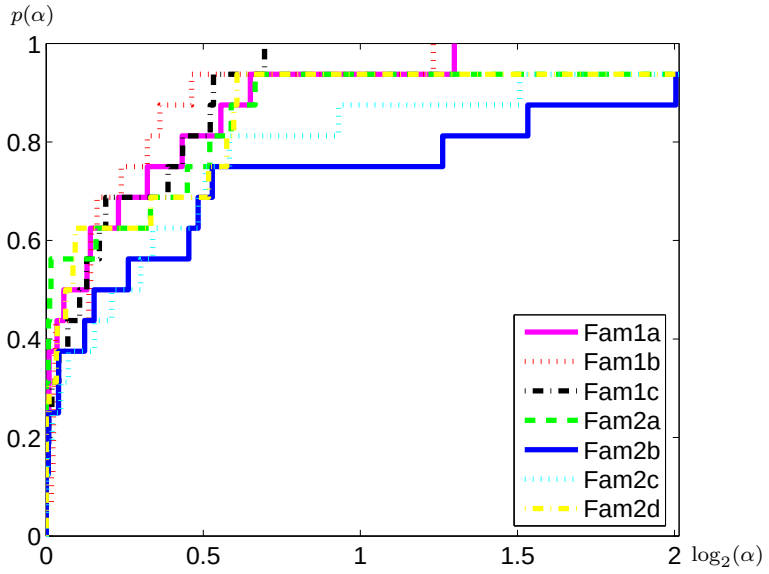

Figure 8.11: Performance profile, $p(\alpha)$: CPU time (seconds) to solve QP programming problems.

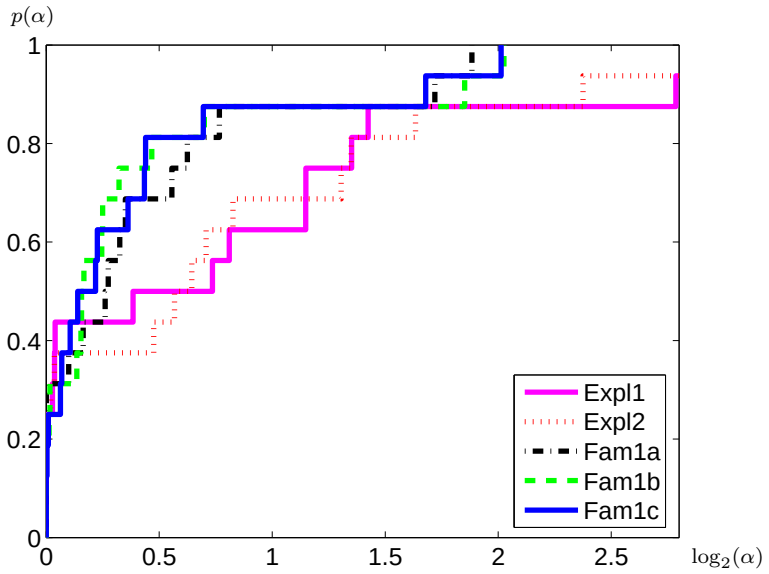

Figure 8.12: Performance profile, $p(\alpha)$: CPU time (seconds) to solve QP programming problems.

There is little difference between `Fam1b` and `Fam1c` in the performance profile; but `Fam1c` will use less memory so we recommend this preconditioner.

The factorization times for our test problems are so fast for the implicit methods generated through Family 1 (for example, see Appendix E) that there is little point in applying Algorithm 5.5.2 with $\gamma = 0$ and a fixed G for the latter interior point iterations. However, for saddle point problems where A_1 is inefficient to factor, then we expect this to be a good method to follow and suggest this as future work.

Chapter 9

Other Preconditioners

In the previous chapters we have concentrated on the use of constraint preconditioners in partnership with projected preconditioned conjugate methods to solve systems of the form

$$\underbrace{\begin{bmatrix} H & A^T \\ A & -C \end{bmatrix}}_{\mathcal{H}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}, \tag{9.0.1}$$

where $H \in \mathbb{R}^{n\times n}$, $C \in \mathbb{R}^{m\times m}$ are symmetric and $A \in \mathbb{R}^{m\times n}$ with $m \leq n$. However, there are a variety of other preconditioners that could be used although a different iterative method to the PPCG method would have to be used. Many of these preconditioners have been constructed with specific applications such as incompressible fluid flow and magnetostatics in mind, but they can equally be applied to constrained optimization problems. We will briefly consider some such preconditioners and compare them with the implicit factorization constraint preconditioners presented in earlier chapters.

9.1 Block diagonal preconditioners

The symmetry of the saddle point problem naturally leads to the consideration of symmetric preconditioners. The constraint preconditioners that we previously considered are indefinite, but, in contrast, several authors have opted for positive definite preconditioners [31, 65, 66, 77]. One choice is the block diagonal matrix

$$K = \begin{bmatrix} I & 0 \\ 0 & AA^T + C \end{bmatrix}.$$

When preconditioning $\mathcal{H}$ with K we obtain

$$K^{-\frac{1}{2}}\mathcal{H}K^{-\frac{1}{2}} = \begin{bmatrix} H & \widehat{A}^T \\ \widehat{A} & -\widehat{C} \end{bmatrix},$$

where $\widehat{A} = \left(AA^T + C\right)^{-\frac{1}{2}} A$ and $\widehat{C} = \left(AA^T + C\right)^{-\frac{1}{2}} C \left(AA^T + C\right)^{-\frac{1}{2}}$. Perugia and Simoncini [65] prove the following results relating to some of the spectral properties of $K^{-\frac{1}{2}}\mathcal{H}K^{-\frac{1}{2}}$:

Theorem 9.1.1. *Let A, C, $\widehat{A}$ and $\widehat{C}$ be as above. Then*

- $\widehat{A}\widehat{A}^T + C = I$
- $A^T(AA^T + C)^{-1}A$ *is a projection and so it has eigenvalues* $\lambda \in \{0, 1.\}$
- *The largest singular value of $\widehat{A}$ is equal to 1.*

Thus, using this in conjunction with Theorem 2.4.4, we obtain

Corollary 9.1.2. *Assume that H is symmetric positive definite and A has full rank. Let λ_1 and λ_n denote the largest and smallest eigenvalues of H respectively, and let σ_m denote the smallest eigenvalue of $\widehat{A}$, where $\widehat{A}$ is defined above. Let $\lambda(K^{-\frac{1}{2}}\mathcal{H}K^{-\frac{1}{2}})$ denote the spectrum of $K^{-\frac{1}{2}}\mathcal{H}K^{-\frac{1}{2}}$. Then*

$$\lambda(K^{-\frac{1}{2}}\mathcal{H}K^{-\frac{1}{2}}) \subset I^- \cup I^+,$$

where

$$\begin{aligned} I^- &= \left[\frac{1}{2}\left(\lambda_n - \left\|\widehat{C}\right\| - \sqrt{\left(\lambda_n + \left\|\widehat{C}\right\|\right)^2 + 4}\right), \frac{1}{2}\left(\lambda_1 - \sqrt{\lambda_1^2 + 4\sigma_m^2}\right)\right], \\ I^+ &= \left[\lambda_n, \frac{1}{2}\left(\lambda_1 + \sqrt{\lambda_1^2 + 4}\right)\right], \end{aligned}$$

and $\widehat{C}$ is as defined above.

Remark 9.1.3. For the case of the quadratic programming problems as considered in Sections 2.3.1 and 2.4.2.1, the upper bound on the eigenvalues will still grow like $\mathcal{O}(\frac{1}{\mu_k})$ as the interior point method draws near to the optimal solution, and the lower bound on the absolute value of the eigenvalues will remain constant. Hence, following on from our results in Chapter 8, we do not expect this preconditioner to be very effective for these problems.

Perugia and Simoncini [65] also consider an approximation to this preconditioner which takes the form

$$K_I = \begin{bmatrix} I & 0 \\ 0 & S_I \end{bmatrix}, \tag{9.1.1}$$

where S_I is a symmetric positive definite approximation to $AA^T + C$. Bounds on the spectral properties can be found in [65].

An alternative which takes into the account the entries of H has been suggested by Murphy, Golub and Wathen when $C = 0$ and H is nonsingular [58]. Suppose that we precondition

$$\mathcal{H} = \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix}$$

by

$$K = \begin{bmatrix} H & 0 \\ 0 & AH^{-1}A^T \end{bmatrix}; \tag{9.1.2}$$

then $K^{-1}\mathcal{H}$ is diagonalizable and has at most four distinct eigenvalues

$$0, \quad 1, \quad \frac{1}{2} \pm \frac{\sqrt{5}}{2}.$$

If $K^{-1}\mathcal{H}$ is nonsingular, then it has the three nonzero eigenvalues. However, as discussed in Section 3.3, the block $AH^{-1}A^T$ may be full and too expensive to compute or factor except in the special case when H is diagonal, and, hence, an approximation to this is often used. It is hoped that the approximation will not dramatically affect the distribution of the eigenvalues. For various discretization schemes when applied to steady incompressible Navier-Stokes equations, Elman shows that

$$X^{-1} = \left(AA^T\right)^{-1} \left(AHA^T\right) \left(AA^T\right)^{-1} \tag{9.1.3}$$

accurately approximates the inverse of the Schur complement [27]. Of course, we may find this to be a bad approximation when applied to our constrained optimization problems.

If $C \neq 0$, then preconditioning $\mathcal{H}$ by the analogous

$$K = \begin{bmatrix} H & 0 \\ 0 & AH^{-1}A^T + C \end{bmatrix} \tag{9.1.4}$$

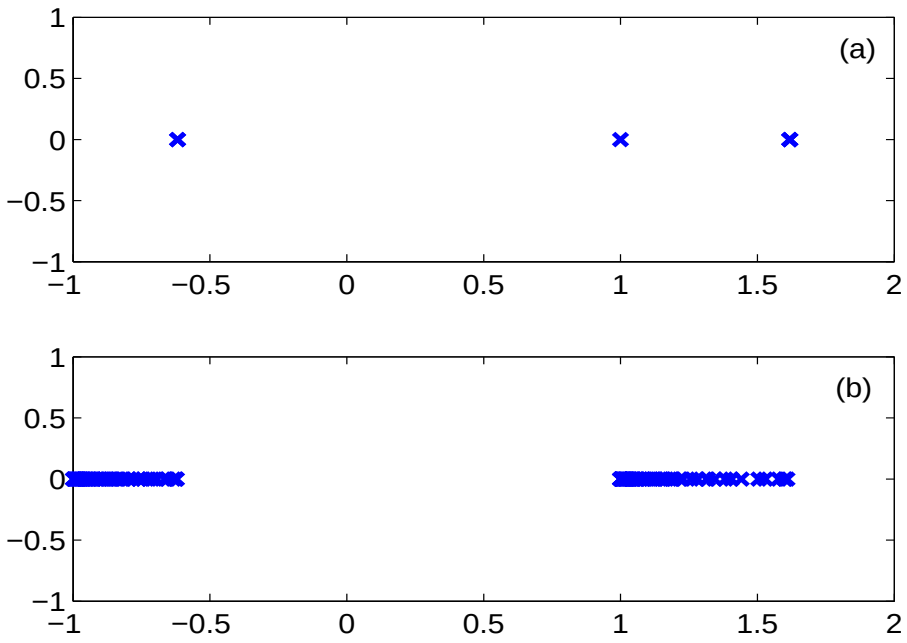

Figure 9.1: Eigenvalues of $K^{-1}\mathcal{H}$ where K is given by (9.1.4) with (a) $C = 0$ (b) $C \neq 0$.

unfortunately does not produce a system with at most four distinct eigenvalues. This is illustrated in Figure 9.1 for the problem `CVXQP1_S` (the value of 1.1 is added to all the diagonal entries of H). We compare the case $C = 0$, and that when C is set to be diagonal with random positive entries.

There are many other block diagonal preconditioners that can be applied: see the work of Gill, Murray, Ponceleón and Saunders for comparisons of some different block diagonal preconditioners which are applied within a linear programming context [37].

9.2 The Hermitian and Skew-Hermitian Splitting Preconditioner

The saddle point system (9.0.1) can be rewritten in the equivalent form

$$\underbrace{\begin{bmatrix} H & A^T \\ -A & C \end{bmatrix}}_{\mathcal{M}} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ -d \end{bmatrix} = \mathcal{M}u = c. \tag{9.2.1}$$

By splitting $\mathcal{M}$ into its symmetric and skew-symmetric parts we obtain

$$\mathcal{M} = \begin{bmatrix} H & 0 \\ 0 & C \end{bmatrix} + \begin{bmatrix} 0 & A^T \\ -A & 0 \end{bmatrix} = \mathcal{M}_H + \mathcal{M}_S. \tag{9.2.2}$$

Let us consider the preconditioning matrix

$$K = \frac{1}{2\alpha} \left(\mathcal{M}_H + \alpha I\right) \left(\mathcal{M}_S + \alpha I\right), \tag{9.2.3}$$

where $\alpha > 0$ is some chosen constant. Note that if C and H are diagonal and positive semidefinite, then $\mathcal{M}_H + \alpha I$ is positive definite. Also, $\mathcal{M}_S + \alpha I$ will be nonsingular. Although the scaling factor $\frac{1}{2\alpha}$ in (9.2.3) has no impact on the preconditioned system it is retained as a normalization factor.

Simoncini and Benzi [78] prove various bounds for the case $C = 0$ on the eigenvalues produced by the associated preconditioned matrix, that is,

$$\left(\mathcal{M}_H + \mathcal{M}_S\right) u = \lambda \frac{1}{2\alpha} \left(\mathcal{M}_H + \alpha I\right) \left(\mathcal{M}_S + \alpha I\right) u. \tag{9.2.4}$$

Let $\Im(\theta)$ and $\Re(\theta)$ denote the imaginary and real parts of θ, respectively. The following theorem gives bounds for the eigenvalues defined by the eigenvalue problem (9.2.4):

Theorem 9.2.1. *Assume that H is symmetric and positive semidefinite with λ_1 and λ_n its largest and smallest eigenvalues, respectively. Let σ_1 and σ_m be the largest and smallest singular values of A, respectively. The eigenvalues of (9.2.4) are such that the following hold:*

1. *If $\Im(\lambda) \neq 0$, then*

$$\begin{aligned} \frac{\left(\alpha + \frac{1}{2}\lambda_n\right)\lambda_n}{3\alpha^2} &< \Re(\lambda) < \min\left\{2, \frac{4\alpha}{\alpha + \lambda_n}\right\}, \\ \frac{\lambda_n^2}{3\alpha^2 + \frac{1}{4}\lambda_n^2} &< |\lambda|^2 \leq \frac{4\alpha}{\alpha + \alpha\left(1 + \frac{\sigma_1^2}{\alpha^2}\right)^{-1} + \lambda_n}. \end{aligned}$$

2. *If $\Im(\lambda) = 0$, then*

$$\min\left\{\frac{2\lambda_n}{\alpha + \lambda_n}, \frac{2\frac{\sigma_m^2}{\varrho}}{\alpha + \frac{\sigma_m^2}{\varrho}}\right\} \leq \lambda \leq \frac{2\rho}{\alpha + \rho} < 2,$$

where $\varrho = \lambda_1(1 + \frac{\sigma_m^2}{\alpha^2})$ and $\rho = \lambda_1(1 + \frac{\sigma_1^2}{\alpha^2})$.

Proof. See [78, Theorem 2.2]. □

In addition, the following theorem holds:

Theorem 9.2.2. *Assume that the hypotheses and notation of Theorem 9.2.1 hold and also assume that H is symmetric positive definite. If $\alpha \leq \frac{1}{2}\lambda_n$, then all the eigenvalues of (9.2.4) are real.*

Proof. See [78, Theorem 3.1]. □

Remark 9.2.3. If the hypotheses and notation of Theorem 9.2.2 hold, and we consider the saddle point problems produced whilst solving quadratic programming problems with interior point methods, Sections 2.3.1 and 2.4.2.1, then for fixed α the lower bound on the eigenvalues will satisfy

$$\begin{aligned}
\min\left\{\frac{2\lambda_n}{\alpha+\lambda_n}, \frac{2\frac{\sigma_m^2}{\varrho}}{\alpha+\frac{\sigma_m^2}{\varrho}}\right\} &= \min\left\{\frac{2\lambda_n}{\alpha+\lambda_n}, \frac{2\sigma_m^2}{\varrho\alpha+\sigma_m^2}\right\} \\
&\approx \min\left\{\frac{2\lambda_n}{\alpha+\lambda_n}, \frac{2\sigma_m^2}{\frac{\zeta}{\mu}\alpha+\sigma_m^2}\right\} \quad \text{for some positive constant } \zeta \\
&= \mathcal{O}(\mu) \quad \text{as} \quad \mu \to 0.
\end{aligned}$$

Hence, we would like α to scale with μ as $\mu \to 0$, that is, $\alpha = \epsilon\mu$ for some (small) $\epsilon > 0$ such that the eigenvalues will cluster around 2.

9.3 Numerical Examples

In this Section we shall compare the above preconditioners with the implicit-factorization constraint preconditioner `Impl1` from Section 8.1.1.3. We wish to solve problems of the form

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}x^TQx + b^Tx \text{ subject to } Ax - d = 0 \text{ and } x \geq 0$$

using the interior point method Algorithm 2.3.1 with Mehrotra's Predictor-Corrector Method. When solving the resulting saddle point problems we shall use the following preconditioners with associated iterative methods:

`Impl1` Implicit factorization constraint preconditioner as defined in Section 8.1.1.3 using the PPCG method.

`BD1` Block diagonal preconditioner as defined in Equation 9.1.1 using the Bi-CGSTAB method.

`BD1a` Block diagonal preconditioner as defined in Equation 9.1.1 but AA^T is approximated by an incomplete Cholesky factorization (Bi-CGSTAB method).

`BD2` Block diagonal preconditioner as defined in Equation 9.1.2 using the Bi-CGSTAB method.

`BD2a` Block diagonal preconditioner as defined in Equation 9.1.2 but the inverse of the Schur complement is approximated by X^{-1} given in (9.1.3) (Bi-CGSTAB method).

`HSS1` The HSS preconditioner as defined in Equation 9.2.3 with $\alpha = 0.01$ using Bi-CGSTAB.

`HSS2` The HSS preconditioner as defined in Equation 9.2.3 with

$$\alpha = \max\left(10^{-8}, \min\left(10^{-2}, 10^{-2}\mu_k\right)\right)$$

using Bi-CGSTAB.

We could use the MINRES method when applying the block diagonal preconditioners but, in practice, there was little difference in the CPU times compared to the Bi-CGSTAB method. We shall terminate the interior point method when $\|Ax_k - d\|_2 < 10^{-6}$ and $\mu_k < 10^{-6}$. (Note: this is different to the termination condition used in Chapter 8.) Whilst $\mu > 10^{-1}$ we will use a stopping tolerance of 10^{-4}, but when $\mu \leq 10^{-1}$ a stopping tolerance of 10^{-10} is used for terminating either PPCG or Bi-CGSTAB.

In Tables 9.1–9.4 we give iteration and timing results for four of the smaller problems (mixed constraints) from the `CUTEr` test set. We observe that the implicit factorization constraint preconditioner `Impl1` and block diagonal preconditioner `BD2` consistently produce results which allow the interior point method to correctly terminate, but the remainder of the preconditioners perform badly in general. However, `Impl1` is tending to perform faster than `BD2`, and as the dimension of the problems increases, we will expect the implicit factorization preconditioner to be several orders of magnitude faster (and, of course, to use far less memory). We are already seeing the large difference in CPU time costs in the relatively small problem `CVXQP2_M`.

	Impl1	BD1	BD1a	BD2	BD2a	HSS1	HSS2
k	6	—	—	6	—	—	—
Total Its 1	72	—	—	20	—	—	—
Total Its 2	73	—	—	19	—	—	—
Total CPU	0.49	—	—	0.83	—	—	—

Table 9.1: CVXQP1_S: Comparison of different preconditioning methods

	Impl1	BD1	BD1a	BD2	BD2a	HSS1	HSS2
k	11	—	—	11	—	—	—
Total Its 1	220	—	—	42	—	—	—
Total Its 2	225	—	—	39	—	—	—
Total CPU	3.66	—	—	40.79	—	—	—

Table 9.2: CVXQP2_M: Comparison of different preconditioning methods

	Impl1	BD1	BD1a	BD2	BD2a	HSS1	HSS2
k	9	7	—	7	7	7	7
Total Its 1	25	25.5	—	22.5	25.5	29.5	24.5
Total Its 2	25	25.5	—	23.5	25.5	30	24.5
Total CPU	0.77	1.43	—	1.34	1.41	2.10	1.91

Table 9.3: DUAL1: Comparison of different preconditioning methods

	Impl1	BD1	BD1a	BD2	BD2a	HSS1	HSS2
k	11	—	—	11	—	11	32
Total Its 1	610	—	—	47.5	—	2953	1167
Total Its 2	623	—	—	40.5	—	2463	585
Total CPU	4.24	—	—	3.07	—	584.35	2229.92

Table 9.4: PRIMAL1: Comparison of different preconditioning methods

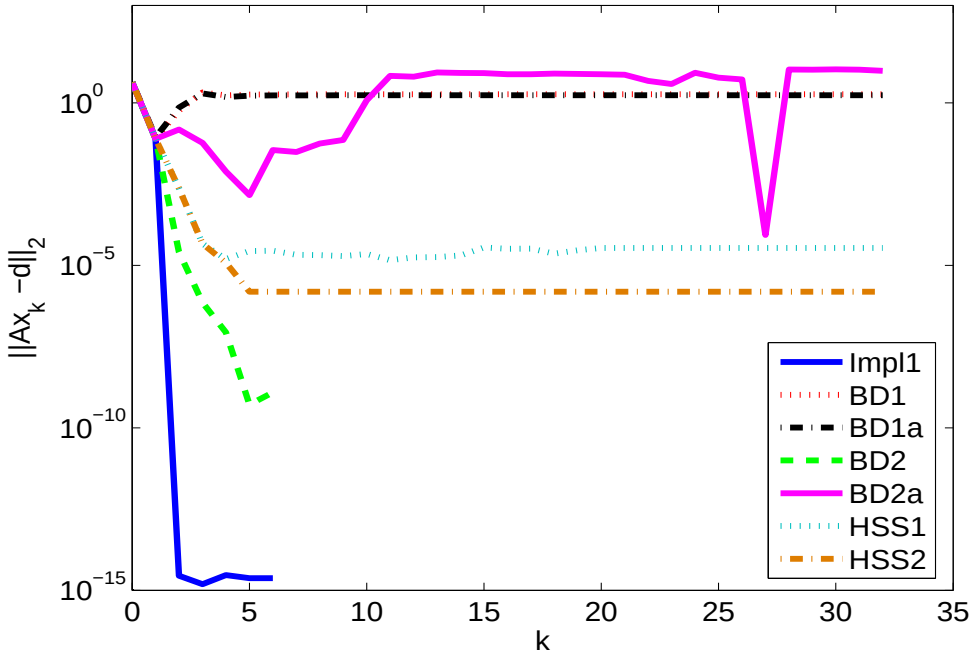

Figure 9.2: CVXQP1_M: Convergence of $\|Ax_k - d\|_2$ for the different preconditioners.

When the preconditioner `BD1a` is applied, the interior point method is always failing to terminate because the requirement $\|Ax_k - d\|_2 < 10^{-6}$ is not satisfied. Similarly, if the interior point method fails to converge when the other preconditioners are employed, we find that it is due to the requirement $\|Ax_k - d\|_2 < 10^{-6}$ failing to be satisfied. We give the plot of the convergence of $\|Ax_k - d\|_2$ for the problem CVXQP1_M in Figure 9.2.

In Chapter 5 we saw that when problems of the form

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix}$$

are (approximately) solved with the PPCG method, Algorithm 5.3.2, then an approximate solution $\begin{bmatrix} \widetilde{\Delta x}^T & \widetilde{\Delta y}^T \end{bmatrix}^T$ is produced for which $\left\| A\widetilde{\Delta x} - \delta_2 \right\| \approx 0$ no matter what tolerance is used in terminating the PPCG method. This results in the value $\|Ax_k - d\|_2$ converging towards zero as the interior point method progresses. The only other preconditioner that we might expect to consistently produce an approximate solution with $\left\| A\widetilde{\Delta x} - \delta_2 \right\| \approx 0$ is `BD2` because of its spectral properties. However, as we have already mentioned, this preconditioner will be very inefficient to apply as the dimension of the problem increases.

Hence, the constraint preconditioners (with the PPCG method) are allowing the constraints to be satisfied earlier during the interior point method and are also cheaper to apply than the other preconditioners considered. Therefore, out of the preconditioners tested, the implicit factorization constraint preconditioner is our preconditioner of choice for the classes of optimization problems considered in this work.

Chapter 10

Concluding Remarks

10.1 Conclusions

In this thesis we have discussed constrained optimization problems and existing methods for finding the optimal solution: interior point methods. We saw that we are required to solve at least one saddle point system during each iteration of these methods and that this is the major cost factor when solving such problems. In Chapters 3 and 4 we considered how direct methods and iterative methods can be employed to solve the arising saddle point systems. We noted that as the size of the problems has increased over time, the currently available routines involving direct methods have become too inefficient so iterative methods are now commonly used.

Although the saddle point systems are symmetric and indefinite, we showed that we can apply projected preconditioned conjugate gradient (PPCG) methods to find (approximate) solutions, Chapter 5. To use such methods, we require constraint preconditioners: we extended the use of these preconditioners to allow for nonzero entries in the (2,2) sub-block of the saddle point system (we have used the notation C for this block throughout this thesis) and have proved relevant results about the spectral properties of the resulting preconditioned system.

In Chapter 6 we assumed that $C = 0$ and firstly noted that choosing a specific constraint preconditioner and then factorizing it for use within the PPCG method can be prohibitively expensive even for the simplest of choices: we have referred to these preconditioners as explicit factorization constraint preconditioners. We then introduced the key idea of implicit factorization constraint preconditioners which allow us to apply constraint preconditioners

with far lower overheads in terms of memory and time. We analyzed two possible implicit factorization methods (the variable reduction method and the Schilders factorization) and found the Schilders factorization to be preferable. Numerical examples in Chapters 6 and 8 substantially support the use of implicit factorization constraint preconditioners instead of explicit factorization constraint preconditioners.

We extend the use of implicit factorization constraint preconditioners to the case of $C \neq 0$ in Chapter 7. Fourteen families of such preconditioners are derived, but we find that some of them will be equivalent when the natural choices for some of the blocks are used and, as such, just three of the families are really worth testing for $C \neq 0$. As for the case of $C = 0$, our numerical examples reveal that the implicit factorization constraint preconditioners can be far more efficient to use when compared to the explicit factorization constraint preconditioners when applied to constrained optimization test problems.

A fundamental assumption was made about the form of the saddle point problem for us to apply our implicit factorization constraint preconditioners, namely, the first m columns of the (2,1) sub-block (the matrix A in our notation) must be linearly independent. We considered different permutations to achieve this assumption in Chapter 8 and found that the diagonal entries of the (1,1) sub-block (the matrix H) should be taken into account when finding such a permutation for mixed constraint optimization problems.

Finally, in Chapter 9, we compared constraint preconditioners with some other preconditioners (and associated iterative methods) that are often used when solving saddle point systems arising in different applications. We found that constraint preconditioners (with the PPCG method) were favourable for the constrained optimization test problems that we considered because they allowed the constraints to be satisfied earlier during the interior point method and were cheaper to apply.

10.2 Future work

There are many possible avenues of interest still to explore. In this section we shall indicate just a few of the possibilities.

10.2.1 Permutations to obtain a nonsingular A_1

The investigations carried out in Chapter 8 were not designed to be very thorough, we just wanted to find a permutation that was cheap to apply and reasonably effective at reducing the number of PPCG iterations required in solving the saddle point problems. However, we saw how important it can be to take the entries of H into account when choosing our permutation. Graph based methods which weight the nodes according to the diagonal entries of H is just one possibility for obtaining suitable permutations that needs exploring.

10.2.2 PDE constrained optimization problems

The methods that we have proposed in Chapter 5 perform an initial projection step which forces the iterates x_k and y_k to satisfy $Ax_k - Cy_k = 0$. We therefore require the constraint to be exact for subsequent iterations to satisfy this requirement. When the matrix A has a PDE-like structure we are often required to handle it approximately: the PPCG methods with constraint preconditioners would not be applicable. In the recent work of Forsgren, Gill and Griffin [33] they assume that C is positive definite and consider solving the *doubly augmented system*

$$\begin{bmatrix} H + 2A^TC^{-1}A & -A^T \\ -A & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b - 2A^TC^{-1}d \\ -d \end{bmatrix} \tag{10.2.1}$$

instead of the equivalent system (3.0.1). If $H + 2A^TC^{-1}A$ is positive definite, then the PCG method can be applied to solve (10.2.1). Forsgren *et al.* suggest using preconditioners of the form

$$\underbrace{\begin{bmatrix} G + 2\widetilde{A}^TC^{-1}\widetilde{A} & -\widetilde{A}^T \\ -\widetilde{A} & C \end{bmatrix}}_{\widetilde{K}},$$

where G is an approximation to H, and $\widetilde{A}$ indicates that the systems involving A can be solved approximately. In addition, they suggest applying the preconditioning step $Kv = r$ to the equivalent system

$$\begin{bmatrix} G & \widetilde{A}^T \\ \widetilde{A} & -C \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} r_1 + 2\widetilde{A}^TC^{-1}r_2 \\ -r_2 \end{bmatrix}$$

to avoid computations involving $G + 2\widetilde{A}^TC^{-1}\widetilde{A}$. The use of implicit factorization constraint preconditioners for this preconditioning step is an exciting prospect.

10.2.3 Other application areas

As we noted in Chapter 2, there are many application areas other than constrained optimization problems for which the solution of saddle point systems is crucial. A further avenue would be to explore the use of implicit factorization constraint preconditioners with the PPCG method for some of these application areas.

Appendices

Appendix A

Improved eigenvalue bounds with reduced-space basis for $C = 0$: Complete tables

The following appendix gives complete results corresponding to the numerical experiments carried out in Section 5.2.1. The ranks of the submatrices in the resulting saddle point systems are given along with results which show that reproducing parts of H in G can dramatically reduce the number of distinct eigenvalues compared to using some general choice for G.

Table A.1: NETLIB LP problems

			rank				iteration bound		
							any G	exact H_{22} & H_{21}	
name	n	m	A	A_2	H_{11}	H_{12}		$\mu + 1$	upper
25FV47	1876	821	820	725	820	0	1057	726	822
80BAU3B	12061	2262	2262	2231	2262	0	9800	2232	2263
ADLITTLE	138	56	56	53	56	0	83	54	57
AFIRO	51	27	27	21	27	0	25	22	25
AGG2	758	516	516	195	516	0	243	196	243
AGG3	758	516	516	195	516	0	243	196	243
AGG	615	488	488	123	488	0	128	124	128
BANDM	472	305	305	161	305	0	168	162	168
BCDOUT	7078	5414	5412	1102	2227	0	1667	1103	1667
BEACONFD	295	173	173	116	173	0	123	117	123
BLEND	114	74	74	37	74	0	41	38	41
BNL1	1586	643	642	458	642	0	945	459	644
BNL2	4486	2324	2324	1207	2324	0	2163	1208	2163
BOEING1	726	351	351	314	351	0	376	315	352
BOEING2	305	166	166	109	166	0	140	110	140
BORE3D	334	233	231	73	231	0	104	74	104
BRANDY	303	220	193	98	193	0	111	99	111
CAPRI	482	271	271	144	261	0	212	145	212
CYCLE	3371	1903	1875	1272	1868	0	1497	1273	1497
CZPROB	3562	929	929	732	929	0	2634	733	930
D2Q06C	5831	2171	2171	2059	2171	0	3661	2060	2172
D6CUBE	6184	415	404	403	404	0	5781	404	416

Table A.1: NETLIB LP problems (continued)

			rank				iteration bound		
							any G	exact H_{22} & H_{21}	
name	n	m	A	A_2	H_{11}	H_{12}		$\mu+1$	upper
DEGEN2	757	444	442	295	442	0	316	296	316
DEGEN3	2604	1503	1501	1052	1501	0	1104	1053	1104
DFL001	12230	6071	6058	5313	6058	0	6173	5314	6072
E226	472	223	223	186	223	0	250	187	224
ETAMACRO	816	400	400	341	400	0	417	342	401
FFFFF800	1028	524	524	290	524	0	505	291	505
FINNIS	1064	497	497	456	497	0	568	457	498
FIT1D	1049	24	24	24	24	0	1026	25	25
FIT1P	1677	627	627	627	627	0	1051	628	628
FIT2D	10524	25	25	25	25	0	10500	26	26
FIT2P	13525	3000	3000	3000	3000	0	10526	3001	3001
FORPLAN	492	161	161	100	161	0	332	101	162
GANGES	1706	1309	1309	397	1309	0	398	398	398
GFRD-PNC	1160	616	616	423	616	0	545	424	545
GOFFIN	101	50	50	50	0	0	52	1	51
GREENBEA	5598	2392	2389	2171	2389	0	3210	2172	2393
GREENBEB	5598	2392	2389	2171	2386	0	3210	2172	2393
GROW15	645	300	300	300	300	0	346	301	301
GROW22	946	440	440	440	440	0	507	441	441
GROW7	301	140	140	140	140	0	162	141	141
SIERRA	2735	1227	1217	768	1217	0	1519	769	1228
ISRAEL	316	174	174	142	174	0	143	143	143
KB2	68	43	43	25	43	0	26	26	26
LINSPANH	97	33	32	32	32	0	66	33	34
LOTFI	366	153	153	110	153	0	214	111	154
MAKELA4	61	40	40	21	40	0	22	22	22
MAROS-R7	9408	3136	3136	3136	3136	0	6273	3137	3137
MAROS	1966	846	846	723	846	0	1121	724	847
MODEL	1557	38	38	11	38	0	1520	12	39
MODSZK1	1620	687	686	667	684	0	935	668	688
NESM	3105	662	662	568	662	0	2444	569	663
OET1	1005	1002	1002	3	1000	0	4	4	4
OET3	1006	1002	1002	4	1000	0	5	5	5
PEROLD	1506	625	625	532	562	0	882	533	626
PILOT4	1123	410	410	367	333	0	714	334	411
PILOT87	6680	2030	2030	1914	2030	0	4651	1915	2031
PILOT-JA	2267	940	940	783	903	0	1328	784	941
PILOTNOV	2446	975	975	823	975	0	1472	824	976
PILOT	4860	1441	1441	1354	1441	0	3420	1355	1442
PILOT-WE	2928	722	722	645	662	0	2207	646	723
PT	503	501	501	2	499	0	3	3	3
QAP8	1632	912	853	697	853	0	780	698	780
QAP12	8856	3192	3089	2783	3089	0	5768	2784	3193
QAP15	22275	6330	6285	5632	6285	0	15991	5633	6331
QPBD_OUT	442	211	211	176	211	0	232	177	212
READING2	6003	4000	4000	2001	2001	0	2004	2002	2004
RECIPELP	204	91	91	78	91	0	114	79	92
SC105	163	105	105	58	105	0	59	59	59
SC205	317	205	205	112	205	0	113	113	113
SC50A	78	50	50	28	50	0	29	29	29
SC50B	78	50	50	28	50	0	29	29	29
SCAGR25	671	471	471	199	471	0	201	200	201

Table A.1: NETLIB LP problems (continued)

			rank				iteration bound		
							any G	exact H_{22} & H_{21}	
name	n	m	A	A_2	H_{11}	H_{12}		$\mu+1$	upper
SCAGR7	185	129	129	56	129	0	57	57	57
SCFXM1	600	330	330	217	330	0	271	218	271
SCFXM2	1200	660	660	440	660	0	541	441	541
SCFXM3	1800	990	990	660	990	0	811	661	811
SCORPION	466	388	388	77	388	0	79	78	79
SCRS8	1275	490	490	341	490	0	786	342	491
SCSD1	760	77	77	77	77	0	684	78	78
SCSD6	1350	147	147	147	147	0	1204	148	148
SCSD8	2750	397	397	397	397	0	2354	398	398
SCTAP1	660	300	300	246	300	0	361	247	301
SCTAP2	2500	1090	1090	955	1090	0	1411	956	1091
SCTAP3	3340	1480	1480	1264	1480	0	1861	1265	1481
SEBA	1036	515	515	479	515	0	522	480	516
SHARE1B	253	117	117	72	117	0	137	73	118
SHARE2B	162	96	96	65	96	0	67	66	67
SHELL	1777	536	535	489	535	0	1243	490	537
SHIP04L	2166	402	360	343	360	0	1807	344	403
SHIP04S	1506	402	360	256	360	0	1147	257	403
SHIP08L	4363	778	712	679	712	0	3652	680	779
SHIP08S	2467	778	712	406	712	0	1756	407	779
SHIP12L	5533	1151	1042	828	1042	0	4492	829	1152
SHIP12S	2869	1151	1042	451	1042	0	1828	452	1152
SIERRA	2735	1227	1217	768	1217	0	1519	769	1228
SIPOW1M	2002	2000	2000	2	2000	0	3	3	3
SIPOW1	2002	2000	2000	2	1999	0	3	3	3
SIPOW2M	2002	2000	2000	2	2000	0	3	3	3
SIPOW2	2002	2000	2000	2	1999	0	3	3	3
SIPOW3	2004	2000	2000	4	1999	0	5	5	5
SIPOW4	2004	2000	2000	4	1999	0	5	5	5
SSEBLIN	218	72	72	72	72	0	147	73	73
STAIR	614	356	356	249	356	0	259	250	259
STANDATA	1274	359	359	283	359	0	916	284	360
STANDGUB	1383	361	360	281	360	0	1024	282	362
STANDMPS	1274	467	467	372	467	0	808	373	468
STOCFOR1	165	117	117	48	117	0	49	49	49
STOCFOR2	3045	2157	2157	888	2157	0	889	889	889
STOCFOR3	23541	16675	16675	6866	16675	0	6867	6867	6867
TFI2	104	101	101	3	100	0	4	4	4
TRUSS	8806	1000	1000	1000	1000	0	7807	1001	1001
TUFF	628	333	302	207	301	0	327	208	327
VTP-BASE	346	198	198	86	198	0	149	87	149
WOOD1P	2595	244	244	244	244	0	2352	245	245
WOODW	8418	1098	1098	1098	1098	0	7321	1099	1099

Table A.2: CUTEr QP problems

			rank				iteration bound				
							any G	exact H_{22}		exact H_{22} & H_{21}	
name	n	m	A	A_2	H_{11}	H_{12}		$\rho+1$	upper	$\mu+1$	upper
AUG2DCQP	20200	10000	10000	10000	10000	0	10201	10001	10201	10001	10001
AUG2DQP	20200	10000	10000	10000	10000	0	10201	10001	10201	10001	10001

Table A.2: CUTEr QP problems (continued)

			rank				iteration bound				
							any G	exact H_{22}		exact H_{22} & H_{21}	
name	n	m	A	A_2	H_{11}	H_{12}		$\rho+1$	upper	$\mu+1$	upper
AUG3DCQP	27543	8000	8000	7998	8000	0	19544	7999	16001	7999	8001
AUG3DQP	27543	8000	8000	7998	8000	0	19544	7999	16001	7999	8001
BLOCKQP1	10011	5001	5001	5001	5001	5000	5011	5011	5011	5002	5002
BLOCKQP2	10011	5001	5001	5001	5001	5000	5011	5011	5011	5002	5002
BLOCKQP3	10011	5001	5001	5001	5001	5000	5011	5011	5011	5002	5002
BLOWEYA	4002	2002	2002	2000	2002	2000	2001	2001	2001	2001	2001
BLOWEYB	4002	2002	2002	2000	2002	2000	2001	2001	2001	2001	2001
BLOWEYC	4002	2002	2002	2000	2002	2000	2001	2001	2001	2001	2001
CONT-050	2597	2401	2401	192	2401	0	197	193	197	193	197
CONT-101	10197	10098	10098	99	10098	0	100	100	100	100	100
CONT-201	40397	40198	40198	199	40198	0	200	200	200	200	200
CONT5-QP	40601	40200	40200	401	40200	0	402	402	402	402	402
CONT1-10	10197	9801	9801	392	9801	0	397	393	397	393	397
CONT1-20	40397	39601	39601	792	39601	0	797	793	797	793	797
CONT-300	90597	90298	90298	299	90298	0	300	300	300	300	300
CVXQP1	10000	5000	5000	2000	5000	2000	5001	4001	5001	2001	5001
CVXQP2	10000	2500	2500	2175	2500	1194	7501	3370	5001	2176	2501
CVXQP3	10000	7500	7500	1000	7500	2354	2501	2001	2501	1001	2501
DEGENQP	125050	125025	125024	26	125024	0	27	27	27	27	27
DUALC1	223	215	215	8	215	0	9	9	9	9	9
DUALC2	235	229	229	6	229	0	7	7	7	7	7
DUALC5	285	278	278	7	278	0	8	8	8	8	8
DUALC8	510	503	503	7	503	0	8	8	8	8	8
GOULDQP2	19999	9999	9999	9999	9999	0	10001	10000	10001	10000	10000
GOULDQP3	19999	9999	9999	9999	9999	9999	10001	10001	10001	10000	10000
KSIP	1021	1001	1001	20	1001	0	21	21	21	21	21
MOSARQP1	3200	700	700	700	700	3	2501	704	1401	701	701
NCVXQP1	10000	5000	5000	2000	5000	2000	5001	4001	5001	2001	5001
NCVXQP2	10000	5000	5000	2000	5000	2000	5001	4001	5001	2001	5001
NCVXQP3	10000	5000	5000	2000	5000	2000	5001	4001	5001	2001	5001
NCVXQP4	10000	2500	2500	2175	2500	1194	7501	3370	5001	2176	2501
NCVXQP5	10000	2500	2500	2175	2500	1194	7501	3370	5001	2176	2501
NCVXQP6	10000	2500	2500	2175	2500	1194	7501	3370	5001	2176	2501
NCVXQP7	10000	7500	7500	1000	7500	2354	2501	2001	2501	1001	2501
NCVXQP8	10000	7500	7500	1000	7500	2354	2501	2001	2501	1001	2501
NCVXQP9	10000	7500	7500	1000	7500	2354	2501	2001	2501	1001	2501
POWELL20	10000	5000	5000	4999	5000	0	5001	5000	5001	5000	5001
PRIMALC1	239	9	9	9	9	0	231	10	19	10	10
PRIMALC2	238	7	7	7	7	0	232	8	15	8	8
PRIMALC5	295	8	8	8	8	0	288	9	17	9	9
PRIMALC8	528	8	8	8	8	0	521	9	17	9	9
PRIMAL1	410	85	85	85	85	0	326	86	171	86	86
PRIMAL2	745	96	96	96	96	0	650	97	193	97	97
PRIMAL3	856	111	111	111	111	0	746	112	223	112	112
PRIMAL4	1564	75	75	75	75	0	1490	76	151	76	76
QPBAND	75000	25000	25000	25000	25000	0	50001	25001	50001	25001	25001
QPNBAND	75000	25000	25000	25000	25000	0	50001	25001	50001	25001	25001
QPCBOEI1	726	351	351	314	351	0	376	315	376	315	352
QPCBOEI2	305	166	166	109	166	0	140	110	140	110	140
QPCSTAIR	614	356	356	249	356	0	259	250	259	250	259
QPNBOEI1	726	351	351	314	351	0	376	315	376	315	352
QPNBOEI2	305	166	166	109	166	0	140	110	140	110	140

Table A.2: CUTEr QP problems (continued)

			rank				iteration bound				
							any G	exact H_{22}		exact H_{22} & H_{21}	
name	n	m	A	A_2	H_{11}	H_{12}		$\rho+1$	upper	$\mu+1$	upper
QPNSTAIR	614	356	356	249	356	0	259	250	259	250	259
SOSQP1	5000	2501	2501	2499	2501	2499	2500	2500	2500	2500	2500
STCQP1	8193	4095	1771	0	1771	317	6423	1	6423	1	4096
STCQP2	8193	4095	4095	0	4095	1191	4099	1	4099	1	4096
STNQP1	8193	4095	1771	0	1771	317	6423	1	6423	1	4096
STNQP2	8193	4095	4095	0	4095	1191	4099	1	4099	1	4096
UBH1	9009	6000	6000	3003	6	0	3010	7	3010	7	3010
YAO	4002	2000	2000	2000	2000	0	2003	2001	2003	2001	2001

Appendix B

CUTEr QP problems: Complete tables

The following appendix gives complete results corresponding to the numerical experiments carried out in Section 6.3. The factorization times "fact," iteration counts "it" and total CPU times "total" are given for different choices of preconditioner.

Table B.1: CUTEr QP problems—residual decrease of at least 10^{-2}

	Explicit factors									Implicit factors					
	$G = H$						$G = I$			$G_{22} = I$			$G_{22} = H_{22}$		
	MA27			MA57			MA57			MA57			MA57		
name	fact	it	total	fact	it	total	fact	it	total	fact	it	total	fact	it	total
AUG2DCQP	0.08	1	0.13	0.47	1	0.54	0.46	1	0.53	0.04	125	1.54	0.25	125	2.01
AUG2DQP	0.08	1	0.13	0.47	1	0.54	0.46	2	0.53	0.04	120	1.49	0.25	125	2.03
AUG3DCQP	1.56	1	1.66	1.54	1	1.67	1.45	1	1.57	0.05	41	0.71	0.79	41	1.59
AUG3DQP	1.59	1	1.69	1.29	1	1.42	1.46	2	1.59	0.05	43	0.71	0.78	40	1.56
BLOCKQP1	0.06	0	0.08	0.21	0	0.23	0.23	1	0.26	0.33	2	0.35	0.39	2	0.41
BLOCKQP2	0.06	0	0.08	0.21	0	0.23	0.23	2	0.26	0.33	2	0.36	0.39	2	0.41
BLOCKQP3	0.06	0	0.08	0.21	0	0.23	0.23	1	0.25	0.33	2	0.35	0.38	2	0.41
BLOWEYA	26.50	1	26.60	0.04	1	0.05	0.05	35	0.21	0.03	50	0.13	0.04	50	0.15
BLOWEYB	26.29	1	26.39	0.04	1	0.05	0.05	13	0.11	0.03	32	0.09	0.04	32	0.11
BLOWEYC	26.27	1	26.36	0.04	1	0.05	0.05	36	0.21	0.03	50	0.12	0.04	50	0.15
CONT-050	0.17	1	0.19	0.12	1	0.14	0.12	1	0.14	0.09	3	0.10	0.09	3	0.11
CONT-101	3.03	1	3.18	0.73	2	0.85	0.70	2	0.82	0.86	2	0.91	0.86	2	0.91
CONT-201	35.96	4	38.38	5.78	5	6.99	5.63	6	7.04	10.14	2	10.41	10.10	2	10.37
CONT5-QP	33.89	1	34.59	3.37	1	3.83	3.35	2	3.80	20.01	39	22.36	19.94	37	22.20
CONT1-10	2.81	1	2.95	0.68	1	0.80	0.66	1	0.77	0.90	3	0.97	0.91	3	0.99
CONT1-20	30.94	1	31.65	6.85	1	7.46	6.67	2	7.28	10.83	3	11.22	10.86	3	11.26
CONT-300	140.10	9	146.23	19.33	5	22.26	18.33	5	21.25	40.82	2	41.46	41.00	2	41.64
CVXQP1	579.20	0	580.15	3.99	0	4.11	0.20	3	0.24	0.21	57	0.56	0.24	55	0.69
CVXQP2	139.11	0	139.48	1.70	0	1.78	0.10	3	0.12	0.01	14	0.07	0.10	14	0.23
CVXQP3	1354	0	1355	9.93	0	10.13	0.32	3	0.38	0.33	44	0.64	0.34	43	0.68
DEGENQP	3.85	1	4.14	14.36	1	14.72	0.01	2	0.01	2.43	3	2.87	2.45	3	2.89
DUALC1	0.01	5	0.01	0.00	2	0.01	0.00	1	0.00	0.00	8	0.00	0.00	8	0.00
DUALC2	0.01	9	0.01	0.00	1	0.01	0.01	2	0.01	0.00	6	0.00	0.00	6	0.01
DUALC5	0.01	8	0.02	0.01	1	0.01	0.01	2	0.01	0.00	6	0.01	0.00	6	0.01
DUALC8	0.11	5	0.13	0.01	2	0.01	0.20	0	0.23	0.01	7	0.01	0.01	7	0.01

Table B.1: CUTEr QP problems—residual decrease of at least 10^{-2} (continued)

	Explicit factors									Implicit factors					
	$G = H$						$G = I$			$G_{22} = I$			$G_{22} = H_{22}$		
	MA27			MA57			MA57			MA57			MA57		
name	fact	it	total	fact	it	total	fact	it	total	fact	it	total	fact	it	total
GOULDQP2	0.05	0	0.07	0.23	0	0.27	0.20	2	0.25	0.03	0	0.05	0.08	0	0.10
GOULDQP3	0.07	1	0.11	0.32	1	0.40	0.05	5	0.06	0.03	6	0.11	0.08	6	0.17
KSIP	0.01	1	0.02	0.05	1	0.06	0.04	3	0.05	0.02	3	0.03	0.02	3	0.03
MOSARQP1	0.02	1	0.03	0.04	1	0.04	0.20	3	0.24	0.06	6	0.07	0.07	6	0.08
NCVXQP1	573.69	0	574.65	4.10	0	4.22	0.20	3	0.24	0.21	55	0.54	0.24	55	0.68
NCVXQP2	584.17	0	585.14	4.02	0	4.14	0.20	3	0.24	0.20	55	0.54	0.24	56	0.70
NCVXQP3	573.04	0	573.98	4.15	0	4.28	0.11	3	0.13	0.20	54	0.53	0.23	55	0.69
NCVXQP4	138.52	0	138.90	1.71	0	1.79	0.10	3	0.12	0.01	14	0.07	0.10	13	0.22
NCVXQP5	130.26	0	130.64	1.69	0	1.76	0.10	3	0.13	0.01	14	0.06	0.10	14	0.24
NCVXQP6	139.37	0	139.75	1.70	0	1.79	0.32	3	0.38	0.01	14	0.06	0.10	14	0.24
NCVXQP7	1364	0	1365	10.03	0	10.23	0.33	3	0.39	0.33	43	0.64	0.34	43	0.67
NCVXQP8	1387	0	1388	10.07	0	10.26	0.33	3	0.38	0.33	43	0.63	0.34	43	0.67
NCVXQP9	1358	0	1359	10.12	0	10.32	0.09	2	0.11	0.33	44	0.64	0.34	43	0.67
POWELL20	0.03	0	0.05	0.09	0	0.11	0.00	5	0.01	0.01	2	0.03	0.07	2	0.08
PRIMALC1	0.00	1	0.00	0.00	1	0.01	0.00	3	0.00	0.00	11	0.00	0.00	6	0.00
PRIMALC2	0.00	1	0.00	0.00	1	0.01	0.00	6	0.01	0.00	5	0.00	0.00	5	0.00
PRIMALC5	0.00	1	0.00	0.00	1	0.01	0.01	4	0.01	0.00	6	0.00	0.00	5	0.00
PRIMALC8	0.01	1	0.01	0.01	1	0.01	0.01	8	0.02	0.00	11	0.01	0.00	7	0.01
PRIMAL1	0.01	1	0.01	0.01	1	0.02	0.03	5	0.03	0.00	15	0.01	0.00	27	0.02
PRIMAL2	0.01	1	0.01	0.03	1	0.03	0.06	4	0.07	0.00	13	0.01	0.01	21	0.02
PRIMAL3	0.03	1	0.03	0.06	1	0.06	0.03	3	0.04	0.01	18	0.04	0.01	26	0.06
PRIMAL4	0.04	1	0.04	0.03	1	0.03	14.34	2	14.69	0.01	12	0.03	0.02	15	0.04
QPBAND	0.16	1	0.30	1.08	1	1.28	1.84	2	1.99	0.09	2	0.19	0.40	2	0.54
QPNBAND	0.17	1	0.30	1.07	1	1.27	1.83	3	2.03	0.09	3	0.24	0.41	2	0.55
QPCBOEI1	0.01	1	0.01	0.02	2	0.02	0.01	3	0.01	0.00	12	0.01	0.00	12	0.01
QPCBOEI2	0.00	1	0.01	0.00	1	0.01	0.00	3	0.01	0.00	12	0.00	0.00	12	0.00
QPCSTAIR	0.01	1	0.01	0.02	1	0.02	0.01	3	0.02	0.00	12	0.01	0.00	14	0.01
QPNBOEI1	0.01	1	0.01	0.02	2	0.02	0.01	3	0.01	0.01	12	0.01	0.00	12	0.01
QPNBOEI2	0.00	1	0.00	0.00	1	0.01	0.00	3	0.01	0.00	12	0.00	0.00	12	0.00
QPNSTAIR	0.01	1	0.01	0.02	1	0.02	0.01	3	0.02	0.00	12	0.01	0.00	12	0.01
SOSQP1	0.01	0	0.01	0.04	0	0.04	0.04	0	0.05	0.03	1	0.04	0.05	1	0.05
STCQP1	rank deficient A			rank deficient A			20.67	3	21.01	0.02	3	0.04	0.09	1	0.10
STCQP2	9.76	0	9.84	0.87	0	0.92	0.14	3	0.17	0.03	3	0.05	0.11	1	0.13
STNQP1	113.27	0	113.59	rank deficient A			20.75	3	21.09	0.02	3	0.04	0.09	1	0.11
STNQP2	9.64	0	9.72	0.87	0	0.92	0.14	3	0.17	0.03	3	0.05	0.11	1	0.13
UBH1	0.02	0	0.03	0.12	0	0.14	0.11	0	0.13	0.02	0	0.03	0.04	0	0.05
YAO	0.01	1	0.01	0.03	1	0.04	0.03	6	0.05	0.01	21	0.04	0.02	21	0.06

Table B.2: CUTEr QP problems—residual decrease of at least 10^{-8}

	Explicit factors									Implicit factors					
	$G = H$						$G = I$			$G_{22} = I$			$G_{22} = H_{22}$		
	MA27			MA57			MA57			MA57			MA57		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.08	1	0.13	0.47	1	0.54	0.46	1	0.53	0.04	866	10.35	0.25	872	12.50
AUG2DQP	0.08	1	0.13	0.47	1	0.54	0.46	4	0.60	0.04	882	10.67	0.25	855	12.33
AUG3DCQP	1.56	1	1.66	1.54	1	1.67	1.45	1	1.57	0.05	378	6.04	0.79	377	8.18
AUG3DQP	1.59	1	1.69	1.29	1	1.42	1.46	5	1.75	0.05	381	5.90	0.78	380	8.27
BLOCKQP1	0.06	0	0.08	0.21	0	0.23	0.23	1	0.26	0.33	3	0.37	0.39	3	0.43
BLOCKQP2	0.06	0	0.08	0.21	0	0.23	0.23	2	0.26	0.33	3	0.37	0.39	3	0.43
BLOCKQP3	0.06	0	0.08	0.21	0	0.23	0.23	1	0.25	0.33	5	0.39	0.38	4	0.44
BLOWEYA	26.50	1	26.60	0.04	1	0.05	$>$ 10000 iterations			$>$ 10000 iterations			$>$ 10000 iterations		
BLOWEYB	26.29	1	26.39	0.04	1	0.05	0.05	216	0.99	0.03	668	1.23	$>$ 10000 iterations		
BLOWEYC	26.27	1	26.36	0.04	1	0.05	$>$ 10000 iterations			$>$ 10000 iterations			$>$ 10000 iterations		

Table B.2: CUTEr QP problems—residual decrease of at least 10^{-8} (continued)

	Explicit factors									Implicit factors					
	$G = H$						$G = I$			$G_{22} = I$			$G_{22} = H_{22}$		
		MA27			MA57			MA57			MA57			MA57	
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
CONT-050	0.17	1	0.19	0.12	1	0.14	0.12	1	0.14	0.09	7	0.12	0.09	7	0.13
CONT-101	3.03	1	3.18	0.73	4	1.11	0.70	5	1.15	0.86	10	1.09	0.86	10	1.10
CONT-201	35.96	4	38.38	5.78	8	9.39	5.63	13	11.26	10.14	11	11.48	10.10	11	11.43
CONT5-QP	33.89	1	34.59	3.37	1	3.83	3.35	2	3.95	20.01	113	26.81	19.94	98	25.89
CONT1-10	2.81	1	2.95	0.68	1	0.80	0.66	1	0.77	0.90	10	1.13	0.91	10	1.16
CONT1-20	30.94	1	31.65	6.85	1	7.46	6.67	5	9.08	10.83	12	12.29	10.86	12	12.34
CONT-300	140.10	27	174.66	19.33	26	45.80	18.33	40	58.01	40.82	15	44.98	41.00	15	45.16
CVXQP1	579.20	0	580.15	3.99	0	4.11	0.20	5	0.27	0.21	211	1.49	0.24	207	1.94
CVXQP2	139.11	0	139.48	1.70	0	1.78	0.10	5	0.14	0.01	51	0.21	0.10	51	0.59
CVXQP3	1353.52	0	1355.13	9.93	0	10.13	0.32	5	0.42	0.33	183	1.62	0.34	178	1.71
DEGENQP	3.85	1	4.14	14.36	1	14.72	0.01	11	0.01	2.43	3	3.00	2.45	7	3.52
DUALC1	0.01	5	0.01	0.00	11	0.01	0.00	1	0.00	0.00	8	0.01	0.00	8	0.00
DUALC2	0.01	9	0.01	0.00	1	0.01	0.01	4	0.01	0.00	6	0.00	0.00	6	0.01
DUALC5	0.01	145	0.20	0.01	1	0.01	0.01	5	0.01	0.00	7	0.01	0.00	7	0.01
DUALC8	0.11	5	0.13	0.01	7	0.02	0.20	0	0.23	0.01	7	0.01	0.01	7	0.01
GOULDQP2	0.05	0	0.07	0.23	0	0.27	0.20	5	0.31	0.03	0	0.05	0.08	0	0.10
GOULDQP3	0.07	1	0.11	0.32	1	0.40	0.05	21	0.08	0.03	1614	18.95	0.08	1579	23.38
KSIP	0.01	1	0.02	0.05	1	0.06	0.04	5	0.05	0.02	18	0.05	0.02	10	0.04
MOSARQP1	0.02	1	0.03	0.04	1	0.04	0.20	5	0.27	0.06	36	0.10	0.07	35	0.13
NCVXQP1	573.69	0	574.65	4.10	0	4.22	0.20	5	0.27	0.21	215	1.51	0.24	204	1.89
NCVXQP2	584.17	0	585.14	4.02	0	4.14	0.20	6	0.28	0.20	212	1.50	0.24	212	2.00
NCVXQP3	573.04	0	573.98	4.15	0	4.28	0.11	5	0.14	0.20	210	1.46	0.23	204	1.92
NCVXQP4	138.52	0	138.90	1.71	0	1.79	0.10	5	0.14	0.01	51	0.20	0.10	51	0.60
NCVXQP5	130.26	0	130.64	1.69	0	1.76	0.10	6	0.15	0.01	51	0.20	0.10	50	0.59
NCVXQP6	139.37	0	139.75	1.70	0	1.79	0.32	5	0.42	0.01	51	0.21	0.10	51	0.61
NCVXQP7	1363.85	0	1365.49	10.03	0	10.23	0.33	5	0.43	0.33	189	1.69	0.34	176	1.67
NCVXQP8	1386.80	0	1388.45	10.07	0	10.26	0.33	5	0.42	0.33	191	1.69	0.34	176	1.70
NCVXQP9	1357.68	0	1359.31	10.12	0	10.32	0.09	20	0.23	0.33	193	1.69	0.34	179	1.71
POWELL20	0.03	0	0.05	0.09	0	0.11	0.00	11	0.01	0.01	40	0.21	0.07	40	0.31
PRIMALC1	0.00	1	0.00	0.00	1	0.01	0.00	4	0.00	0.00	25	0.01	0.00	12	0.01
PRIMALC2	0.00	1	0.00	0.00	1	0.01	0.00	10	0.01	0.00	9	0.00	0.00	9	0.00
PRIMALC5	0.00	1	0.00	0.00	1	0.01	0.01	7	0.01	0.00	15	0.01	0.00	10	0.01
PRIMALC8	0.01	1	0.01	0.01	1	0.01	0.01	14	0.02	0.00	20	0.01	0.00	10	0.01
PRIMAL1	0.01	1	0.01	0.01	1	0.02	0.03	8	0.03	0.00	153	0.08	0.00	158	0.09
PRIMAL2	0.01	1	0.01	0.03	1	0.03	0.06	6	0.07	0.00	86	0.06	0.01	92	0.08
PRIMAL3	0.03	1	0.03	0.06	1	0.06	0.03	5	0.04	0.01	74	0.14	0.01	80	0.15
PRIMAL4	0.04	1	0.04	0.03	1	0.03	14.34	2	14.80	0.01	41	0.07	0.02	44	0.09
QPBAND	0.16	1	0.30	1.08	1	1.28	1.84	5	2.19	0.09	7	0.46	0.40	5	0.78
QPNBAND	0.17	1	0.30	1.07	1	1.27	1.83	6	2.24	0.09	8	0.51	0.41	6	0.84
QPCBOEI1	0.01	1	0.01	0.02	5	0.02	0.01	5	0.02	0.00	47	0.03	0.00	47	0.03
QPCBOEI2	0.00	1	0.01	0.00	1	0.01	0.00	5	0.01	0.00	38	0.01	0.00	37	0.01
QPCSTAIR	0.01	1	0.01	0.02	1	0.02	0.01	8	0.02	0.00	40	0.02	0.00	52	0.03
QPNBOEI1	0.01	1	0.01	0.02	5	0.03	0.01	5	0.01	0.01	48	0.03	0.00	47	0.03
QPNBOEI2	0.00	1	0.00	0.00	1	0.01	0.00	5	0.01	0.00	37	0.01	0.00	37	0.01
QPNSTAIR	0.01	1	0.01	0.02	1	0.02	0.01	8	0.02	0.00	40	0.02	0.00	56	0.03
SOSQP1	0.01	0	0.01	0.04	0	0.04	0.04	0	0.05	0.03	1	0.04	0.05	1	0.05
STCQP1	rank deficient A			rank deficient A			20.67	6	21.35	0.02	6	0.05	0.09	1	0.10
STCQP2	9.76	0	9.84	0.87	0	0.92	0.14	7	0.20	0.03	7	0.07	0.11	1	0.13
STNQP1	113.27	0	113.59	rank deficient A			20.75	6	21.43	0.02	6	0.05	0.09	1	0.11
STNQP2	9.64	0	9.72	0.87	0	0.92	0.14	8	0.22	0.03	8	0.08	0.11	1	0.13
UBH1	0.02	0	0.03	0.12	0	0.14	0.11	0	0.13	0.02	0	0.03	0.04	0	0.05
YAO	0.01	1	0.01	0.03	1	0.04	0.03	26	0.11	0.01	107	0.18	0.02	106	0.23

Appendix C

Generation of the implicit factorization families

We examine each of the sub-cases mentioned in Section 7.1 in detail. Note that for general P and B as partitioned in (7.1.4), we have

$$\begin{aligned}
A_1 &= (P_{31}B_{11} + P_{32}B_{21})\,P_{11}^T + \left(P_{31}B_{21}^T + P_{32}B_{22}\right)P_{12}^T \\
&\quad + P_{33}\left(B_{31}P_{11}^T + B_{32}P_{12}^T\right) + \left(P_{31}B_{31}^T + P_{32}B_{32}^T\right)A_1 \\
&\quad + P_{33}B_{33}A_1 \\
A_2 &= (P_{31}B_{11} + P_{32}B_{21})\,P_{21}^T + \left(P_{31}B_{21}^T + P_{32}B_{22}\right)P_{22}^T \\
&\quad + P_{33}\left(B_{31}P_{21}^T + B_{32}P_{22}^T\right) + \left(P_{31}B_{31}^T + P_{32}B_{32}^T\right)A_2 \\
&\quad + P_{33}B_{33}A_2 \\
-C &= (P_{31}B_{11} + P_{32}B_{21})\,P_{31}^T + \left(P_{31}B_{21}^T + P_{32}B_{22}\right)P_{32}^T \\
&\quad + P_{33}\left(B_{31}P_{31}^T + B_{32}P_{32}^T\right) + \left(P_{31}B_{31}^T + P_{32}B_{32}^T\right)P_{33}^T \\
&\quad + P_{33}B_{33}P_{33}^T.
\end{aligned}$$

Case 1: (7.1.8) and (7.1.11) hold

If (7.1.8) and (7.1.11) hold, then P_{31}, P_{22}, B_{11}, B_{22} and B_{33} are required to be nonsingular, and

$$A_1 = P_{33}B_{33}A_1, \tag{C.0.1}$$

$$A_2 = P_{31}B_{11}P_{21}^T + P_{33}B_{33}A_2, \tag{C.0.2}$$

$$-C = P_{31}B_{11}P_{31}^T + P_{33}B_{33}P_{33}^T. \tag{C.0.3}$$

Equation C.0.1 implies that

$$P_{33}B_{33} = I \tag{C.0.4}$$

and, hence, that P_{33} is symmetric. Equation C.0.2 forces $P_{31}B_{11}P_{21}^T = 0$, and thus that

$$P_{21} = 0$$

since P_{31} and B_{11} are nonsingular. Finally, (C.0.3) becomes

$$-C = P_{31}B_{11}P_{31}^T + P_{33}. \tag{C.0.5}$$

We therefore have

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{bmatrix}, \tag{C.0.6}$$

where

$$B_{11} = -P_{31}^{-1}\left(C + P_{33}\right)P_{31}^{-T} \quad \text{and} \quad B_{33} = P_{33}^{-1}. \tag{C.0.7}$$

Case 2: (7.1.8) and (7.1.12) hold

If (7.1.8) and (7.1.12) hold, then P_{31}, P_{22}, B_{31} and B_{22} are required to be nonsingular, and

$$\begin{aligned} A_1 &= P_{31}B_{31}^T A_1, && \text{(C.0.8)} \\ A_2 &= P_{31}B_{31}^T A_2 + P_{31}B_{11}P_{21}^T + P_{31}B_{21}^T P_{22}^T + P_{33}B_{31}P_{21}^T, && \text{(C.0.9)} \\ -C &= P_{31}B_{31}^T P_{33}^T + P_{31}B_{11}P_{31}^T + P_{33}B_{31}P_{31}^T. && \text{(C.0.10)} \end{aligned}$$

Equation C.0.8 implies that

$$P_{31}B_{31}^T = I, \tag{C.0.11}$$

holds. It then follows from (C.0.10) and (C.0.11) that

$$P_{33} + P_{33}^T + P_{31}B_{11}P_{31}^T = -C. \tag{C.0.12}$$

The remaining requirement (C.0.9) implies that $P_{31}B_{11}P_{21}^T + P_{31}B_{21}^T P_{22}^T + P_{33}B_{31}P_{21}^T = 0$, which is most easily guaranteed if either

$$B_{21} = 0 \quad \text{and} \quad P_{21} = 0 \tag{C.0.13}$$

or

$$B_{21} = 0 \quad \text{and} \quad P_{31}B_{11} = -P_{33}B_{31} \tag{C.0.14}$$

or

$$B_{21} = 0, B_{11} = 0 \quad \text{and} \quad P_{33} = 0. \tag{C.0.15}$$

When (C.0.13) holds, it follows from (C.0.11) and (C.0.12) that

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & 0 \end{bmatrix}, \tag{C.0.16}$$

where

$$B_{31} = P_{31}^{-1} \quad \text{and} \quad P_{33} + P_{33}^T + P_{31}B_{11}P_{31}^T = -C. \tag{C.0.17}$$

In the case of (C.0.14),

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & 0 & -C \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & 0 \end{bmatrix}, \tag{C.0.18}$$

where

$$B_{31} = P_{31}^{-1} \quad \text{and} \quad P_{31}B_{11}P_{31}^T = C, \tag{C.0.19}$$

as then

$$P_{33} + P_{31}B_{11}P_{31}^T = P_{33} - P_{33}B_{31}P_{31}^T = P_{33} - P_{33} = 0$$

from (C.0.11) and (C.0.14) and hence $P_{33} = P_{33}^T = -C$ from (C.0.12). Finally, (C.0.15) can only hold when $C = 0$, and is a special instance of (C.0.18)—(C.0.19).

Case 3: (7.1.8) and (7.1.13) hold

If (7.1.8) and (7.1.13) hold, then P_{31}, P_{22}, B_{31} and B_{22} are required to be nonsingular, and

$$A_1 = P_{31}B_{31}^T A_1 + P_{33}B_{33}A_1, \tag{C.0.20}$$

$$A_2 = P_{31}B_{31}^T A_2 + P_{33}B_{33}A_2 + P_{33}\left(B_{31}P_{21}^T + B_{32}P_{22}^T\right), \tag{C.0.21}$$

$$-C = P_{31}B_{31}^T P_{33}^T + P_{33}B_{33}P_{33}^T + P_{33}B_{31}P_{31}^T. \tag{C.0.22}$$

Equation C.0.20 implies that either (C.0.11) holds and either $P_{33} = 0$ or $B_{33} = 0$, or

$$P_{33}B_{33} = I - P_{31}B_{31}^T \tag{C.0.23}$$

with nonzero P_{33} and B_{33}. It is easy to see that requirement (C.0.22) cannot hold when $P_{33} = 0$ unless $C = 0$, this will then be a special case of Family

10. So suppose instead that (C.0.11) holds and that $B_{33} = 0$. In this case, the requirement (C.0.22) is simply that

$$P_{33} + P_{33}^T = -C,$$

while (C.0.21) additionally requires that

$$B_{31}P_{21}^T + B_{32}P_{22}^T = 0. \tag{C.0.24}$$

This results in

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & 0 \end{bmatrix}, \tag{C.0.25}$$

where

$$B_{31} = P_{31}^{-T}, \;\; P_{21} = -P_{22}B_{32}^T B_{31}^{-T} \quad \text{and} \quad P_{33} + P_{33}^T = -C. \tag{C.0.26}$$

Suppose that (C.0.23) holds with nonzero P_{33} and B_{33}. Then the requirement (C.0.22) is that

$$-C = (I - P_{33}B_{33})\,P_{33}^T + P_{33}B_{33}P_{33}^T + P_{33}\,(I - B_{33}P_{33}) = P_{33} + P_{33}^T - P_{33}B_{33}P_{33}^T$$

while once again (C.0.24) holds since $P_{22} \neq 0$. Thus, we have

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & 0 \end{bmatrix}, \tag{C.0.27}$$

where

$$B_{31} = \left(I - B_{33}P_{33}^T\right) P_{31}^{-T}, \;\; B_{33} = P_{33}^{-1} + P_{33}^{-T} + P_{33}^{-1}CP_{33}^{-T} \quad \text{and} \quad B_{32} = -B_{31}P_{21}^T P_{22}^{-T}. \tag{C.0.28}$$

Case 4: (7.1.9) and (7.1.11) hold

If (7.1.9) and (7.1.11) hold, then P_{31}, P_{22}, B_{11}, B_{22} and B_{33} are required to be nonsingular, and

$$A_1 = P_{33}B_{33}A_1, \tag{C.0.29}$$

$$A_2 = P_{33}B_{33}A_2 + P_{32}B_{22}P_{22}^T, \tag{C.0.30}$$

$$-C = P_{33}B_{33}P_{33}^T + P_{32}B_{22}P_{32}^T + P_{31}B_{11}P_{31}^T. \tag{C.0.31}$$

As in case 1, (C.0.29) implies that (C.0.4) holds (and thus P_{33} is symmetric). Requirement (C.0.30) then forces $P_{32}B_{22}P_{22}^T = 0$, and thus that $P_{32} = 0$ since P_{22} and B_{22} are nonsingular. Requirement (C.0.31) then leads to (C.0.5), and hence exactly the same conclusions as for case 1.

Case 5: (7.1.9) and (7.1.12) hold

If (7.1.9) and (7.1.12) hold, then P_{31}, P_{22}, B_{31} and B_{22} are required to be nonsingular, and

$$\begin{aligned} A_1 &= P_{31}B_{31}^T A_1, && \text{(C.0.32)} \\ A_2 &= P_{331}B_{31}^T A_2 + \left(P_{31}B_{21}^T + P_{32}B_{22}\right)P_{22}^T, && \text{(C.0.33)} \\ -C &= P_{31}B_{31}^T P_{33}^T + \left(P_{31}B_{21}^T + P_{32}B_{22}\right)P_{32}^T + P_{33}B_{31}P_{31}^T \\ &\quad + \left(P_{31}B_{11} + P_{32}B_{21}\right)P_{31}^T. && \text{(C.0.34)} \end{aligned}$$

As in case 2, (C.0.32) implies that (C.0.11) holds. Requirement (C.0.33) and the nonsingularity of P_{22} together imply that

$$P_{31}B_{21}^T + P_{32}B_{22} = 0.$$

Thus wither

$$B_{21} = 0 \quad \text{and} \quad P_{32} = 0$$

or

$$P_{32} = -P_{31}B_{21}^T B_{22}^{-1} \quad \text{with nonzero} \quad B_{21} and P_{32}$$

since B_{31} and P_{22} are nonsingular. The first of these two cases is identical to (C.0.6)–(C.0.17) under the requirement (C.0.34). Under the same requirement, simple manipulation for the second case gives

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{bmatrix}, \qquad \text{(C.0.35)}$$

where

$$B_{31} = P_{31}^{-T}, \;\; P_{33}+P_{33}^T = -C-P_{31}\left(B_{11} - B_{21}^T B_{22}^{-1} B_{21}\right)P_{31}^T \quad \text{and} \quad P_{32} = -P_{31}B_{21}^T B_{22}^{-1}. \qquad \text{(C.0.36)}$$

Case 6: (7.1.9) and (7.1.13) hold

If (7.1.9) and (7.1.13) hold, then P_{31}, P_{22}, B_{31} and B_{22} are required to be nonsingular, and

$$\begin{aligned} A_1 &= \left(P_{31}B_{31}^T + P_{32}B_{32}^T + P_{33}B_{33}\right) A_1, && \text{(C.0.37)} \\ A_2 &= \left(P_{31}B_{31}^T + P_{32}B_{32}^T + P_{33}B_{33}\right) A_2 + \left(P_{32}B_{22} + P_{33}B_{32}\right) P_{22}^T, && \text{(C.0.38)} \\ -C &= \left(P_{31}B_{31}^T + P_{32}B_{32}^T + P_{33}B_{33}\right) P_{33}^T + \left(P_{32}B_{22} + P_{33}B_{32}\right) P_{32}^T \\ &\quad + P_{33}B_{31}P_{31}^T. && \text{(C.0.39)} \end{aligned}$$

The requirement (C.0.37) implies that either

$$P_{31}B_{31}^T + P_{32}B_{32}^T = I, \tag{C.0.40}$$

and either $P_{33} = 0$ or $B_{33} = 0$, or

$$P_{31}B_{31} = I - P_{32}B_{32}^T - P_{33}B_{33} \tag{C.0.41}$$

with nonzero P_{33} and B_{33}. As in case 3, it is easy to see that it is not possible for the requirement (C.0.39) to hold when $P_{33} = 0$ unless $C = 0$, and this is just a case of Family 10. Suppose instead that (C.0.40) holds and that $B_{33} = 0$. Then the nonsingularity of P_{22} and B_{22} and the requirement (C.0.38) together imply that

$$P_{32} = -P_{33}B_{32}B_{22}^{-1}.$$

Requirement (C.0.39) then becomes

$$P_{33} + P_{33}^T + P_{33}B_{32}B_{22}^{-1}B_{32}^T P_{33}^T = -C. \tag{C.0.42}$$

This then results in

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & 0 \end{bmatrix}, \tag{C.0.43}$$

where

$$\begin{aligned} &P_{33} + P_{33}^T + P_{33}B_{32}B_{22}^{-1}B_{32}^T P_{33}^T = -C, \\ &P_{32} = -P_{33}B_{32}B_{22}^{-1}, \text{ and } P_{31} = \left(I - P_{32}B_{32}^T\right) B_{31}^{-T}. \end{aligned} \tag{C.0.44}$$

Note that although (C.0.42) restricts the choice of B_{32} and B_{22}, it is easily satisfied, for example, when $B_{32} = 0$.

Suppose that (C.0.41) holds with nonzero P_{33} and B_{33}. Then, once again, the nonsingularity of P_{22} and B_{22} and the requirement (C.0.38) together imply that (C.0.42) holds, while (C.0.41) and (C.0.42) show that the requirement (C.0.39) holds whenever

$$\begin{aligned} -C &= P_{33} + P_{33}^T + P_{32}B_{22}P_{32}^T - P_{33}B_{33}P_{33}^T \\ &= P_{33} + P_{33}^T + P_{33}\left(B_{32}B_{22}^{-1}B_{32}^T - B_{33}\right)P_{33}^T. \end{aligned}$$

Hence, we obtain

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ 0 & P_{22} & A_2^T \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{bmatrix}, \tag{C.0.45}$$

where

$$\begin{aligned} &P_{33} + P_{33}^T + P_{33}\left(B_{32}B_{22}^{-1}B_{32}^T - B_{33}\right)P_{33}^T = -C, \\ &P_{32} = -P_{33}B_{32}B_{22}^{-1}, \text{ and } P_{31} = \left(I - P_{32}B_{32}^T - P_{33}B_{33}\right)B_{31}^{-T}. \end{aligned} \tag{C.0.46}$$

Observe that (C.0.43)–(C.0.44) is the special case $B_{33} = 0$ of (C.0.45)–(C.0.46).

Case 7: (7.1.10) and (7.1.11) hold

If (7.1.10) and (7.1.11) hold, then P_{31}, P_{22}, B_{22} and B_{33} are required to be nonsingular, and

$$A_1 = P_{31}B_{11}P_{11}^T, \;\; A_2 = P_{31}B_{11}P_{21}^T \quad \text{and} \quad -C = P_{31}B_{11}P_{31}^T.$$

This then leads to the requirement that C is nonsingular and

$$P_{11} = A_1^T, \;\; P_{21} = A_2^T, \;\; P_{31} = -C \quad \text{and} \quad B_{11} = -C^{-1}.$$

Hence,

$$P = \begin{bmatrix} A_1^T & 0 & A_1^T \\ A_2^T & P_{22} & A_2^T \\ -C & P_{32} & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} -C^{-1} & 0 & 0 \\ 0 & B_{22} & 0 \\ 0 & 0 & B_{33} \end{bmatrix}. \tag{C.0.47}$$

Case 8: (7.1.10) and (7.1.12) hold

If (7.1.10) and (7.1.12) hold, then P_{31}, P_{22}, B_{31} and B_{22} are required to be nonsingular, and

$$\begin{aligned} A_1 &= P_{31}B_{31}^T A_1 + P_{31}B_{11}P_{11}^T, && \text{(C.0.48)} \\ A_2 &= P_{31}B_{31}^T A_2 + P_{31}B_{11}P_{21}^T + P_{31}B_{21}^T A_2, && \text{(C.0.49)} \\ -C &= P_{31}B_{11}^T P_{31}^T. && \text{(C.0.50)} \end{aligned}$$

In reverse order, these give

$$\begin{aligned} B_{11} &= -P_{31}^{-1}CP_{31}^{-T}, \\ B_{31} &= P_{31}^{-T} - A_1^{-T}P_{11}B_{11}, \\ \text{and} \quad B_{21} &= P_{22}^{-1}\left(P_{21} - A_2^T A_1^{-T}P_{11}\right)B_{11}. \end{aligned}$$

There is very little reason to believe that B_{31} will be easily invertible in general, but if $P_{11} = A_1^T M$ for some diagonal M and if P_{31} and B_{11} are also diagonal, then it may be. This leads to

$$P = \begin{bmatrix} P_{11} & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & B_{21}^T & B_{31}^T \\ B_{21} & B_{22} & 0 \\ B_{31} & 0 & 0 \end{bmatrix}, \qquad \text{(C.0.51)}$$

where

$$\begin{aligned} &B_{11} = -P_{31}^{-1}CP_{31}^{-T}, \;\; B_{31} = P_{31}^{-T} - MB_{11}, \\ &B_{21} = P_{22}^{-1}\left(P_{21} - A_2^T M\right)B_{11} \quad \text{and} \quad P_{11} = A_1^T M \end{aligned} \qquad \text{(C.0.52)}$$

for some suitable M.

Case 9: (7.1.10) and (7.1.13) hold

If (7.1.10) and (7.1.13) hold, then P_{31}, P_{22}, B_{31} and B_{22} are required to be nonsingular, and

$$A_1 = P_{31}B_{31}^T A_1, \;\; A_2 = P_{31}B_{31}^T A_2 \quad \text{and} \quad -C = 0.$$

Thus, the matrix C is required to be equal to the zero matrix of appropriate size. This gives

$$P = \begin{bmatrix} P_{11} & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ B_{31}^{-T} & 0 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} 0 & 0 & B_{31}^T \\ 0 & B_{22} & B_{32}^T \\ B_{31} & B_{32} & B_{33} \end{bmatrix}. \qquad \text{(C.0.53)}$$

Case 10: (7.1.8) and (7.1.14) hold

If (7.1.8) and (7.1.14) hold, then P_{31}, P_{22} and B_{22} are required to be nonsingular, and

$$\begin{aligned} A_1 &= P_{31}B_{31}^T A_1 + P_{33}B_{33}A_1, && \text{(C.0.54)} \\ A_2 &= P_{31}B_{31}^T A_2 + P_{33}B_{33}A_2 + P_{31}B_{11}P_{21}^T + P_{33}B_{31}P_{21}^T, && \text{(C.0.55)} \\ -C &= P_{31}B_{31}^T P_{33}^T + P_{33}B_{33}P_{33}^T + P_{31}B_{11}P_{31}^T + P_{33}B_{31}P_{31}^T. && \text{(C.0.56)} \end{aligned}$$

Requirement (C.0.54) implies that

$$P_{31}B_{31}^T + P_{33}B_{33} = I, \tag{C.0.57}$$

whilst (C.0.55) gives either

$$P_{31}B_{11} + P_{33}B_{31} = 0 \tag{C.0.58}$$

or

$$P_{31}B_{11} + P_{33}B_{31} \neq 0 \quad \text{and} \quad P_{21} = 0. \tag{C.0.59}$$

If (C.0.57) and (C.0.58) hold, requirement (C.0.56) is simply $P_{33} = -C$. If C is invertible, this leads to

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{bmatrix}, \tag{C.0.60}$$

where

$$P_{33} = -C, \;\; P_{31}^T = -B_{11}^{-1}B_{31}^T P_{33}^T \quad \text{and} \quad B_{33} = \left(I - B_{31}P_{31}^T\right) P_{33}^{-T}. \tag{C.0.61}$$

However, since solves with B simply involve B_{22} and

$$\begin{bmatrix} B_{11} & B_{31}^T \\ B_{31} & B_{33} \end{bmatrix} = \begin{bmatrix} B_{11} & 0 \\ B_{31} & I \end{bmatrix} \begin{bmatrix} B_{11}^{-1} & 0 \\ 0 & C^{-1} \end{bmatrix} \begin{bmatrix} B_{11}^T & B_{31}^T \\ 0 & I \end{bmatrix}, \tag{C.0.62}$$

the block form of (C.0.62) indicates that only products with C, and not its inverse, are required when solving with B, and that B_{33} need not be formed.

If C is singular, then (C.0.57) and (C.0.58) give that

$$P_{33} = -C, \;\; B_{31} = (I + B_{33}C)\, P_{31}^{-T} \quad \text{and} \quad B_{11} = P_{31}^{-1}\,(C + CP_{33}C)\, P_{31}^{-T}. \tag{C.0.63}$$

As before, solves with B simply involve B_{22} and

$$\begin{bmatrix} B_{11} & B_{31}^T \\ B_{31} & B_{33} \end{bmatrix} = \begin{bmatrix} P_{31}^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} CB_{33}C + C & I + CB_{33} \\ I + B_{33}C & B_{33} \end{bmatrix} \begin{bmatrix} P_{31}^{-T} & 0 \\ 0 & I \end{bmatrix},$$

and, hence, we need to ensure that

$$\begin{bmatrix} CB_{33}C + C & I + CB_{33} \\ I + B_{33}C & B_{33} \end{bmatrix} = \begin{bmatrix} C & I \\ I & 0 \end{bmatrix} + \begin{bmatrix} C \\ I \end{bmatrix} B_{33} \begin{bmatrix} C & I \end{bmatrix} \quad \text{(C.0.64)}$$

is non-singular (and has the correct inertia). The possibility $B_{33} = 0$ is that given by (C.0.18) in Case 2, but an interesting alternative is when B_{33} is chosen so that

$$B_{33}C = 0. \quad \text{(C.0.65)}$$

In this case, (C.0.63) becomes

$$P_{33} = -C, \;\; B_{31} = P_{31}^{-T} \quad \text{and} \quad B_{11} = P_{31}^{-1} C P_{31}^{-T}, \quad \text{(C.0.66)}$$

and (C.0.64) gives

$$\begin{bmatrix} CB_{33}C + C & I + CB_{33} \\ I + B_{33}C & B_{33} \end{bmatrix} = \begin{bmatrix} C & I \\ I & B_{33} \end{bmatrix} = \begin{bmatrix} I & 0 \\ B_{33} & I \end{bmatrix} \begin{bmatrix} C & I \\ I & 0 \end{bmatrix} \quad \text{(C.0.67)}$$

which is clearly (block) invertible.

If (C.0.57) and (C.0.59) hold, then the requirement (C.0.56) becomes

$$-C = P_{33}^T + P_{31} B_{11} P_{31}^T + P_{33} B_{31} P_{31}^T$$

which leads to

$$P = \begin{bmatrix} 0 & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & P_{33} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{bmatrix}, \quad \text{(C.0.68)}$$

where

$$P_{31} = (I - P_{33}B_{33})\, B_{31}^{-T} \quad \text{and} \quad B_{11} = P_{31}^{-1} \left(P_{33} B_{33} P_{33}^T - C - P_{33} - P_{33}^T \right) P_{31}^{-T}. \quad \text{(C.0.69)}$$

Case 11: (7.1.9) and (7.1.14) hold

If (7.1.9) and (7.1.14) hold, then P_{31}, P_{22} and B_{22} are required to be nonsingular, and

$$\begin{aligned} A_1 &= P_{31}B_{31}^T A_1 + P_{33}B_{33}A_1, && \text{(C.0.70)} \\ A_2 &= P_{31}B_{31}^T A_2 + P_{33}B_{33}A_2 + P_{32}B_{22}P_{22}^T, && \text{(C.0.71)} \\ -C &= \begin{array}{l} P_{31}B_{31}^T P_{33}^T + P_{33}B_{33}P_{33}^T + P_{31}B_{11}P_{31}^T \\ +P_{33}B_{31}P_{31}^T + P_{32}B_{22}P_{32}^T. \end{array} && \text{(C.0.72)} \end{aligned}$$

To satisfy the requirement (C.0.70), then (C.0.57) must hold. Requirement (C.0.71) and the nonsingularity of P_{22} and B_{22} force $P_{32} = 0$. This case is then simply a subcase of the previous one.

Case 12: (7.1.10) and (7.1.14) hold

If (7.1.10) and (7.1.14) hold, then P_{31}, P_{22} and B_{22} are required to be nonsingular, and

$$\begin{aligned} A_1 &= P_{31}B_{31}^T A_1 + P_{31}B_{11}P_{11}^T, && \text{(C.0.73)} \\ A_2 &= P_{31}B_{31}^T A_2 + P_{31}B_{11}P_{21}^T, && \text{(C.0.74)} \\ -C &= P_{31}B_{11}P_{31}^T. && \text{(C.0.75)} \end{aligned}$$

As in case 8, the requirements (C.0.73) and C.0.75 respectively imply that

$$B_{11} = -P_{31}^{-1}CP_{31}^{-T} \quad \text{and} \quad B_{31} = P_{31}^{-T} - A_1^{-T}P_{11}B_{11}.$$

Requirement (C.0.74) imposes that $B_{11}\left(P_{21}^T - P_{11}^T A_1^{-1}A_2\right) = 0$, which is certainly satisfied when

$$P_{21}^T = P_{11}^T A_1^{-1}A_2.$$

The latter is true, for example, if $P_{11} = A_1^T M$ and $P_{21} = A_2^T M$ for a given matrix M. In general, we thus have that

$$P = \begin{bmatrix} P_{11} & 0 & A_1^T \\ P_{21} & P_{22} & A_2^T \\ P_{31} & 0 & 0 \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & 0 & B_{31}^T \\ 0 & B_{22} & 0 \\ B_{31} & 0 & B_{33} \end{bmatrix}, \qquad \text{(C.0.76)}$$

where

$$B_{11} = -P_{31}^{-1}CP_{31}^{-T}, \quad B_{31} = P_{31}^{-T} - A_1^{-T}P_{11}B_{11} \quad \text{and} \quad P_{21}^T = P_{11}^T A_1^{-1}A_2.$$

The warnings concerning the easy invertibility of B_{31} we mentioned for the case 8 equally apply here, so we actually require

$$B_{11} = -P_{31}^{-1} C P_{31}^{-T}, \;\; B_{31} = P_{31}^{-T} - M B_{11}, \;\; P_{11} = A_1^T M \quad \text{and} \quad P_{21}^T = A_2^T M, \tag{C.0.77}$$

for some suitable (diagonal) M.

Appendix D

Implicit factorization families and the matrix G which arises

Here we examine the matrix G which arises for each of the families mentioned in Section 7.1. Note that for general P and B partitioned as is (7.1.4), we have

$$\begin{aligned}
G_{11} &= P_{11}B_{11}P_{11}^T + P_{11}B_{21}^TP_{12}^T + P_{11}B_{31}^TA_1 + P_{12}B_{21}P_{11}^T + P_{12}B_{22}P_{12}^T \\
&\quad + P_{12}B_{32}^TA_1 + A_1^TB_{31}P_{11}^T + A_1^TB_{32}P_{12}^T + A_1^TB_{33}A_1, \\
G_{21} &= P_{21}B_{11}P_{11}^T + P_{21}B_{21}^TP_{12}^T + P_{21}B_{31}^TA_1 + P_{22}B_{21}P_{11}^T + P_{22}B_{22}P_{12}^T \\
&\quad + P_{22}B_{32}^TA_1 + A_2^TB_{31}P_{11}^T + A_2^TB_{32}P_{12}^T + A_2^TB_{33}A_1, \\
G_{22} &= P_{21}B_{11}P_{21}^T + P_{21}B_{21}^TP_{22}^T + P_{21}B_{31}^TA_2 + P_{22}B_{21}P_{21}^T + P_{22}B_{22}P_{22}^T \\
&\quad + P_{22}B_{32}^TA_2 + A_2^TB_{31}P_{21}^T + A_2^TB_{32}P_{22}^T + A_2^TB_{33}A_2.
\end{aligned}$$

Family 1: (7.1.8) and (7.1.11) hold

In this case

$$\begin{aligned}
G_{11} &= A_1^TB_{33}A_1, \\
G_{21} &= A_2^TB_{33}A_1, \\
G_{22} &= P_{21}B_{11}P_{21}^T + P_{22}B_{22}P_{22}^T + A_2^TB_{33}A_2.
\end{aligned}$$

Since $P_{21} = 0$ for Family 1, G_{22} becomes

$$G_{22} = P_{22}B_{22}P_{22}^T + A_2^TB_{33}A_2.$$

Families 2 and 3: (7.1.8) and (7.1.12) hold

In this case

$$\begin{aligned} G_{11} &= 0, \\ G_{21} &= P_{21}B_{31}^T A_1, \\ G_{22} &= P_{21}B_{11}P_{21}^T + P_{21}B_{21}^T P_{22}^T + P_{21}B_{31}^T A_2 + P_{22}B_{21}P_{21}^T + P_{22}B_{22}P_{22}^T \\ &\quad + A_2^T B_{31}P_{21}^T. \end{aligned}$$

For Family 2, $B_{21} = 0$ and $P_{21} = 0$, so that G_{21} and G_{22} become

$$\begin{aligned} G_{21} &= 0, \\ G_{22} &= P_{22}B_{22}P_{22}^T. \end{aligned}$$

For Family 3, $B_{21} = 0$, so that G_{22} becomes

$$G_{22} = P_{21}B_{11}P_{21}^T + P_{21}B_{31}^T A_2 + P_{22}B_{22}P_{22}^T + A_2^T B_{31}P_{21}^T.$$

Families 4 and 5: (7.1.8) and (7.1.13) hold

In this case

$$\begin{aligned} G_{11} &= A_1^T B_{33}A_1, \\ G_{21} &= P_{21}B_{31}^T A_1 + P_{22}B_{32}^T A_1 + A_2^T B_{33}A_1, \\ G_{22} &= P_{21}B_{31}^T A_2 + P_{22}B_{22}P_{22}^T + P_{22}B_{32}^T A_2 + A_2^T B_{31}P_{21}^T + A_2^T B_{32}P_{22}^T \\ &\quad + A_2^T B_{33}A_2. \end{aligned}$$

For both families, (C.0.24) holds, and thus

$$\begin{aligned} G_{21} &= A_2^T B_{33}A_1, \\ G_{22} &= P_{22}B_{22}P_{22}^T + A_2^T B_{33}A_2. \end{aligned}$$

In addition, for Family 4, $B_{33} = 0$, so

$$G_{22} = P_{22}B_{22}P_{22}^T.$$

Family 6: (7.1.9) and (7.1.12) hold

In this case

$$\begin{aligned} G_{11} &= 0, \\ G_{21} &= 0, \\ G_{22} &= 0P_{22}B_{22}P_{22}^T. \end{aligned}$$

Family 7: (7.1.9) and (7.1.13) hold

In this case

$$\begin{aligned} G_{11} &= A_1^T B_{33} A_1, \\ G_{21} &= P_{22} B_{32}^T A_1 + A_2^T B_{33} A_1, \\ G_{22} &= P_{22} B_{22} P_{22}^T + P_{22} B_{32}^T A_2 + A_2^T B_{32} P_{22}^T + A_2^T B_{33} A_2. \end{aligned}$$

Family 8: (7.1.10) and (7.1.11) hold

In this case

$$\begin{aligned} G_{11} &= P_{11} B_{11} P_{11}^T + A_1^T B_{33} A_1, \\ G_{21} &= P_{21} B_{11} P_{11}^T + A_2^T B_{33} A_1, \\ G_{22} &= P_{21} B_{11} P_{21}^T + P_{22} B_{22} P_{22}^T + A_2^T B_{33} A_2. \end{aligned}$$

But since $P_{11} = A_1^T$, $P_{21} = A_2^T$ and $B_{11} = -C^{-1}$, we have

$$\begin{aligned} G_{11} &= A_1^T \left(B_{33} - C^{-1} \right) A_1, \\ G_{21} &= A_2^T \left(B_{33} - C^{-1} \right) A_1, \\ G_{22} &= P_{22} B_{22} P_{22}^T + A_2^T \left(B_{33} - C^{-1} \right) A_2. \end{aligned}$$

Family 9: (7.1.10) and (7.1.12) hold

In this case

$$\begin{aligned} G_{11} &= P_{11} B_{11} P_{11}^T + P_{11} B_{31}^T A_1 + A_1^T B_{31} P_{11}^T, \\ G_{21} &= P_{21} B_{11} P_{11}^T + P_{21} B_{31}^T A_1 + P_{22} B_{21} P_{11}^T + A_2^T B_{31} P_{11}^T, \\ G_{22} &= P_{21} B_{11} P_{21}^T + P_{21} B_{21}^T P_{22}^T + P_{21} B_{31}^T A_2 + P_{22} B_{21} P_{21}^T + P_{22} B_{22} P_{22}^T \\ &\quad + A_2^T B_{31} P_{21}^T. \end{aligned}$$

Family 10: (7.1.10) and (7.1.13) hold

In this case

$$\begin{aligned} G_{11} &= P_{11} B_{31}^T A_1 + A_1^T B_{31} P_{11}^T + A_1^T B_{33} A_1, \\ G_{21} &= P_{21} B_{31}^T A_1 + P_{22} B_{32}^T A_1 + A_2^T B_{31} P_{11}^T + A_2^T B_{33} A_1, \\ G_{22} &= P_{21} B_{31}^T A_2 + P_{22} B_{22} P_{22}^T + P_{22} B_{32}^T A_2 + A_2^T B_{31} P_{21}^T + A_2^T B_{32} P_{22}^T \\ &\quad + A_2^T B_{33} A_2. \end{aligned}$$

Families 11, 12 and 13: (7.1.8) and (7.1.14) hold

In this case

$$\begin{aligned} G_{11} &= A_1^T B_{33} A_1, \\ G_{21} &= P_{21} B_{31}^T A_1 + P_{22} B_{32}^T A_1 + A_2^T B_{31} P_{11}^T + A_2^T B_{33} A_1, \\ G_{22} &= P_{21} B_{11} P_{21}^T + P_{21} B_{31}^T A_2 + P_{22} B_{22} P_{22}^T + A_2^T B_{31} P_{21}^T + A_2^T B_{33} A_2. \end{aligned}$$

For Family 13, $P_{21} = 0$, and thus

$$\begin{aligned} G_{21} &= P_{22} B_{32}^T A_1 + A_2^T B_{31} P_{11}^T + A_2^T B_{33} A_1, \\ G_{22} &= P_{22} B_{22} P_{22}^T + A_2^T B_{33} A_2. \end{aligned}$$

Family 14: (7.1.10) and (7.1.14) hold

In this case

$$\begin{aligned} G_{11} &= P_{11} B_{11} P_{11}^T + P_{11} B_{31}^T A_1 + A_1^T B_{31} P_{11}^T + A_1^T B_{33} A_1, \\ G_{21} &= P_{21} B_{11} P_{11}^T + P_{21} B_{31}^T A_1 + A_2^T B_{31} P_{11}^T + A_2^T B_{33} A_1, \\ G_{22} &= P_{21} B_{11} P_{21}^T + P_{21} B_{31}^T A_2 + P_{22} B_{22} P_{22}^T + A_2^T B_{31} P_{21}^T + A_2^T B_{33} A_2. \end{aligned}$$

Appendix E

CUTEr QP problems: Complete tables

The following appendix gives complete results corresponding to the numerical experiments carried out in Section 7.3. The factorization times "fact," iteration counts "iter" and total CPU times "total" are given for different choices of preconditioner.

Table E.1: CUTEr QP problems—residual decrease of at least 10^{-2} and $C = I$

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.38	1	0.45	0.12	1	0.18	0.07	13	0.19	0.07	267	1.94	0.02	36	0.30
AUG2DQP	0.12	1	0.18	0.12	1	0.18	0.06	13	0.17	0.03	268	1.89	0.02	36	0.29
AUG3DCQP	0.16	1	0.24	0.16	1	0.24	0.06	24	0.24	0.03	96	0.74	0.03	37	0.33
AUG3DQP	0.16	1	0.24	0.16	1	0.24	0.07	25	0.26	0.03	96	0.75	0.03	37	0.33
BLOCKQP1	5.03	1	33.28	4.98	1	33.15	0.14	1	28.20	0.06	1	28.18	0.06	1	28.09
BLOCKQP2	4.98	1	33.13	5.02	1	33.14	0.14	1	28.22	0.06	1	28.17	0.07	1	28.16
BLOCKQP3	4.96	1	33.03	5.04	1	33.13	0.14	1	28.08	0.06	1	28.10	0.06	1	28.07
BLOWEYA	0.27	1	0.33	16.58	1	16.69	0.06	1	0.10	0.05	1	0.09	0.05	1	0.09
BLOWEYB	0.28	1	0.34	16.52	1	16.63	0.06	1	0.10	0.05	1	0.09	0.05	1	0.08
BLOWEYC	0.29	1	0.35	14.75	1	14.87	0.07	1	0.11	0.05	1	0.09	0.05	1	0.09
CONT-050	0.36	1	0.53	0.61	1	0.81	0.05	1	0.17	0.01	1	0.13	0.01	1	0.13
CONT-101	2.17	1	3.33	4.39	1	5.84	0.20	0	1.02	0.06	0	0.87	0.05	0	0.87
CONT-201	13.18	1	21.04	29.63	1	38.82	0.86	0	6.94	0.24	0	6.27	0.24	0	6.30
CONT1-10	2.16	1	3.35	4.22	1	5.57	0.20	1	1.08	0.05	1	0.93	0.05	1	0.92
CONT1-20	13.33	1	21.64	29.07	1	38.37	0.85	0	7.48	0.25	0	6.84	0.25	0	6.78
CONT5-QP	ran out of memory			ran out of memory			0.91	1	6.63	0.24	1	5.94	0.25	1	5.92
CVXQP1	48.16	1	50.38	139.83	1	142.34	0.18	2	0.36	0.13	1310	43.43	0.12	1258	42.24
CVXQP2	27.27	1	28.66	29.21	1	30.67	0.11	3	0.21	0.30	523	21.98	0.30	130	5.63
CVXQP3	53.57	1	56.33	78.67	1	82.48	0.45	1	0.89	0.16	1	0.59	0.06	1	0.48
DUALC1	0.13	1	0.23	0.03	1	0.06	0.01	1	0.03	0.01	1	0.02	0.01	3	0.03
DUALC2	0.03	1	0.05	0.03	1	0.05	0.01	1	0.04	0.01	1	0.03	0.01	0	0.02
DUALC5	0.04	1	0.07	0.13	1	0.16	0.02	10	0.06	0.01	1	0.03	0.01	563	0.81
DUALC8	0.10	1	0.23	0.10	1	0.20	0.04	1	0.15	0.01	1	0.11	0.01	7	0.12
GOULDQP2	0.85	1	1.04	0.79	1	1.04	0.27	4	0.45	0.12	13	0.60	0.11	14	0.65
GOULDQP3	0.81	1	0.99	0.92	1	1.23	0.27	4	0.46	0.11	13	0.78	0.11	15	0.98
KSIP	0.50	1	1.04	0.50	1	1.02	0.06	2	0.59	0.01	2	0.55	0.01	2	0.55
MOSARQP1	0.09	1	0.13	0.09	1	0.13	0.04	7	0.09	0.02	5	0.08	0.02	5	0.08
NCVXQP1	117.67	1	119.93	141.30	1	143.77	0.14	2	0.32	9.66	1420	69.96	9.64	676	38.51
NCVXQP2	129.22	1	131.48	141.86	1	144.34	0.14	3	0.35	9.64	1	9.82	9.65	31	11.12
NCVXQP3	128.41	1	130.66	129.05	1	131.54	0.14	2	0.32	9.64	1197	60.17	8.36	1047	52.77
NCVXQP4	89.54	1	90.99	93.00	1	94.49	0.12	2	0.20	19.55	564	51.00	19.44	4	19.73
NCVXQP5	89.18	1	90.61	84.34	1	85.80	0.14	3	0.25	19.56	790	62.90	22.38	9	22.94
NCVXQP6	88.20	1	89.58	91.74	1	93.18	0.14	3	0.24	18.91	522	47.40	18.91	161	27.79
NCVXQP7	628.16	1	632.70	82.54	1	86.56	0.34	1	0.78	0.17	1	0.61	0.06	1	0.47
NCVXQP8	61.02	1	64.21	84.22	1	88.60	0.28	1	0.73	0.19	1	0.65	0.07	1	0.52
NCVXQP9	54.51	1	57.44	85.05	1	88.96	0.29	1	0.74	2.20	1	2.66	2.10	1	2.53
POWELL20	0.34	1	0.47	0.32	1	0.46	0.14	1	0.20	0.06	13	0.31	0.06	14	0.32

Table E.1: CUTEr QP problems—residual decrease of at least 10^{-8} (continued)

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
PRIMAL1	0.11	1	0.76	0.11	1	0.12	0.05	19	0.08	0.01	8	0.03	0.01	2	0.02
PRIMAL2	0.16	1	0.17	0.16	1	0.17	0.05	18	0.09	0.02	2	0.03	0.03	2	0.04
PRIMAL3	0.57	1	0.60	0.59	1	0.62	0.04	25	0.14	0.02	2	0.04	0.01	2	0.03
PRIMAL4	0.30	1	0.32	0.30	1	0.32	0.05	15	0.11	0.01	2	0.03	0.02	2	0.03
PRIMALC1	0.01	1	0.02	0.02	1	0.02	0.02	2	0.03	0.01	248	0.25	0.01	30	0.04
PRIMALC2	0.02	1	0.02	0.03	1	0.03	0.03	2	0.03	0.01	245	0.24	0.01	245	0.24
PRIMALC5	0.02	1	0.02	0.02	1	0.02	0.05	2	0.05	0.02	7	0.03	0.02	7	0.03
PRIMALC8	0.04	1	0.04	0.03	1	0.04	0.02	2	0.03	0.02	64	0.12	0.01	6	0.02
QPBAND	0.42	1	0.55	0.42	1	0.57	0.19	2	0.26	0.15	16	0.77	0.15	14	0.71
QPNBAND	0.53	1	0.70	0.43	1	0.59	0.18	3	0.27	0.15	12	0.61	0.15	12	0.63
QPCBOEI1	0.05	1	0.09	0.06	1	0.10	0.02	16	0.09	0.01	2	0.04	0.01	2	0.04
QPCBOEI2	0.09	1	0.11	0.09	1	0.11	0.02	2	0.03	0.02	1	0.02	0.02	1	0.03
QPNBOEI1	0.43	1	0.47	0.05	1	0.09	0.03	16	0.09	0.01	3	0.05	0.01	2	0.05
QPNBOEI2	0.11	1	0.12	0.09	1	0.11	0.02	2	0.03	0.01	1	0.02	0.01	1	0.02
QPCSTAIR	0.05	1	0.11	0.05	1	0.11	0.02	3	0.08	0.01	13	0.09	0.01	5	0.07
QPNSTAIR	0.05	1	0.11	0.06	1	0.12	0.02	4	0.08	0.01	9	0.08	0.01	5	0.07
SOSQP1	0.12	1	0.17	0.12	1	0.18	0.07	1	0.10	0.03	1	0.06	0.03	1	0.06
STCQP2	0.86	1	0.96	1.47	1	1.62	0.05	4	0.12	0.09	1	0.13	0.09	2622	38.05
STNQP2	63.37	1	63.72	66.32	1	66.82	0.12	8	0.38	6.52	247	17.24	0.27	15	0.86
UBH1	0.34	1	0.52	0.33	1	0.52	0.13	2	0.29	0.05	1	0.17	0.05	4	0.23

Table E.2: CUTEr QP problems—residual decrease of at least 10^{-2} and $C = I$

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.38	1	0.44	0.12	1	0.18	0.07	157	1.11	0.07	1220	8.48	0.02	89	0.66
AUG2DQP	0.12	1	0.18	0.12	1	0.18	0.06	165	1.14	0.03	1271	8.83	0.02	670	4.60
AUG3DCQP	0.16	1	0.24	0.16	1	0.24	0.06	106	0.75	0.03	1684	11.91	0.03	2621	18.50
AUG3DQP	0.16	1	0.24	0.16	1	0.24	0.07	106	0.76	0.03	273	1.98	0.03	50	0.42
BLOCKQP1	5.03	1	33.06	4.98	1	33.11	0.14	1	28.22	0.06	2	28.20	0.06	2	28.14
BLOCKQP2	4.98	1	33.02	5.02	1	33.31	0.14	1	28.27	0.06	2	28.15	0.07	2	28.13
BLOCKQP3	4.96	1	33.06	5.04	1	33.15	0.14	1	28.24	0.06	2	28.17	0.06	2	28.23
BLOWEYA	0.27	1	0.32	16.58	1	16.69	0.06	1	0.09	0.05	1	0.09	0.05	1	0.09
BLOWEYB	0.28	1	0.33	16.52	1	16.63	0.06	1	0.10	0.05	1	0.09	0.05	1	0.08
BLOWEYC	0.29	1	0.35	14.75	1	14.87	0.07	1	0.11	0.05	1	0.09	0.05	1	0.09
CONT-050	0.36	1	0.53	0.61	1	0.81	0.05	9	0.23	0.01	28	0.34	0.01	29	0.35
CONT-101	2.17	1	3.33	4.39	1	5.82	0.20	0	1.03	0.06	0	0.86	0.05	0	0.84
CONT-201	13.18	1	21.05	29.63	1	38.79	0.86	0	6.97	0.24	0	6.25	0.24	0	6.34
CONT1-10	2.16	1	3.34	4.22	1	5.57	0.20	6	1.26	0.05	28	1.80	0.05	30	1.90
CONT1-20	13.33	1	21.66	29.07	1	38.44	0.85	0	7.52	0.25	0	6.76	0.25	0	6.77
CONT5-QP	ran out of memory			ran out of memory			0.91	1	6.71	0.24	110	30.72	0.25	147	37.19
CVXQP1	48.16	1	50.39	139.83	1	142.36	0.18	51	1.56	0.13	9237	305.76	0.12	4165	138.96
CVXQP2	27.27	1	28.67	29.21	1	30.67	0.11	369	7.39	0.30	9400	380.46	0.30	1353	55.55
CVXQP3	53.57	1	56.32	78.67	1	82.50	0.45	3	0.98	0.16	9291	369.75	0.06	5782	224.38
DUALC1	0.13	1	0.14	0.03	1	0.05	0.01	16	0.05	0.01	2	0.02	0.01	15	0.04
DUALC2	0.03	1	0.05	0.03	1	0.05	0.01	9	0.05	0.01	1	0.03	0.01	0	0.02
DUALC5	0.04	1	0.07	0.13	1	0.16	0.02	11	0.06	0.01	135	0.22	0.01	563	0.82
DUALC8	0.10	1	0.20	0.10	1	0.21	0.04	13	0.18	0.01	997	2.16	0.01	14	0.13
GOULDQP2	0.85	1	1.04	0.79	1	1.04	0.27	17	0.87	0.12	317	11.63	0.11	715	25.62
GOULDQP3	0.81	1	0.99	0.92	1	1.23	0.27	18	0.93	0.11	96	4.63	0.11	673	31.52
KSIP	0.50	1	1.02	0.50	1	1.03	0.06	8	0.61	0.01	6	0.57	0.01	6	0.57
MOSARQP1	0.09	1	0.13	0.09	1	0.13	0.04	50	0.27	0.02	295	2.07	0.02	952	6.58
NCVXQP1	117.67	1	119.93	141.30	1	143.77	0.14	61	1.73	9.66	9557	416.62	9.64	1802	86.77
NCVXQP2	129.22	1	131.49	141.86	1	144.36	0.14	9942	236.88	9.64	1	9.82	9.65	31	11.10
NCVXQP3	128.41	1	130.67	129.05	1	131.54	0.14	62	1.78	9.64	9877	443.35	8.36	2618	119.47
NCVXQP4	89.54	1	90.98	93.00	1	94.49	0.12	8373	163.19	19.55	7877	438.09	19.44	416	42.25
NCVXQP5	89.18	1	90.59	84.34	1	85.79	0.14	8263	160.94	19.56	1069	78.17	22.38	61	25.79
NCVXQP6	88.20	1	89.60	91.74	1	93.18	0.14	9041	175.54	18.91	9394	531.64	18.91	248	32.56
NCVXQP7	628.16	1	632.76	82.54	1	86.49	0.34	3	0.84	0.17	9736	373.63	0.06	4650	180.96
NCVXQP8	61.02	1	64.21	84.22	1	88.69	0.28	3	0.81	0.19	9994	426.71	0.07	5460	215.20
NCVXQP9	54.51	1	57.41	85.05	1	89.06	0.29	3	0.79	2.20	1790	81.11	2.10	1124	50.49
POWELL20	0.34	1	0.47	0.32	1	0.46	0.14	1	0.20	0.06	3581	54.93	0.06	82	1.37
PRIMAL1	0.11	1	0.12	0.11	1	0.12	0.05	172	0.30	0.01	21	0.05	0.01	31	0.07
PRIMAL2	0.16	1	0.17	0.16	1	0.17	0.05	132	0.31	0.02	23	0.08	0.03	37	0.12
PRIMAL3	0.57	1	0.60	0.59	1	0.62	0.04	117	0.44	0.02	36	0.16	0.01	37	0.16
PRIMAL4	0.30	1	0.32	0.30	1	0.32	0.05	62	0.27	0.01	13	0.08	0.02	53	0.23
PRIMALC1	0.01	1	0.02	0.02	1	0.02	0.02	6	0.03	0.01	248	0.25	0.01	132	0.14
PRIMALC2	0.02	34	0.05	0.03	1	0.03	0.03	4	0.03	0.01	245	0.23	0.01	245	0.24
PRIMALC5	0.02	1	0.02	0.02	1	0.02	0.05	5	0.05	0.02	16	0.04	0.02	14	0.03

Table E.2: CUTEr QP problems—residual decrease of at least 10^{-8} (continued)

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
PRIMALC8	0.04	1	0.04	0.03	1	0.04	0.02	5	0.03	0.02	536	0.81	0.01	46	0.08
QPBAND	0.42	1	0.55	0.42	1	0.57	0.19	7	0.37	0.15	50	1.97	0.15	224	8.36
QPNBAND	0.53	1	0.70	0.43	1	0.58	0.18	7	0.36	0.15	30	1.24	0.15	159	5.98
QPCBOEI1	0.05	1	0.09	0.06	1	0.10	0.02	113	0.28	0.01	222	0.51	0.01	23	0.09
QPCBOEI2	0.09	1	0.11	0.09	1	0.11	0.02	4	0.03	0.02	2	0.03	0.02	2	0.03
QPNBOEI1	0.43	1	0.47	0.05	1	0.09	0.03	114	0.29	0.01	20	0.08	0.01	24	0.09
QPNBOEI2	0.11	1	0.12	0.09	1	0.11	0.02	4	0.03	0.01	2	0.02	0.01	2	0.02
QPCSTAIR	0.05	1	0.11	0.05	1	0.11	0.02	144	0.35	0.01	142	0.35	0.01	38	0.14
QPNSTAIR	0.05	1	0.11	0.06	1	0.12	0.02	145	0.35	0.01	135	0.34	0.01	28	0.12
SOSQP1	0.12	1	0.17	0.12	1	0.18	0.07	3	0.13	0.03	18	0.27	0.03	34	0.46
STCQP2	0.86	1	0.96	1.47	1	1.62	0.05	92	1.12	0.09	1	0.13	0.09	6140	89.14
STNQP2	63.37	1	63.72	66.32	1	66.82	0.12	5141	129.65	6.52	4747	177.13	0.27	5966	207.81
UBH1	0.34	1	0.53	0.33	1	0.52	0.13	30	0.87	0.05	472	10.12	0.05	47	1.13

Table E.3: CUTEr QP problems—residual decrease of at least 10^{-2} and C given by (7.3.3)

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.13	1	0.19	0.13	1	0.19	0.06	19	0.21	0.02	43	0.35	0.02	1	0.06
AUG2DQP	0.12	1	0.18	0.12	1	0.18	0.06	19	0.21	0.02	43	0.35	0.02	1	0.06
AUG3DCQP	0.16	2	0.25	0.17	2	0.25	0.06	14	0.19	0.03	12	0.15	0.03	10	0.14
AUG3DQP	0.17	2	0.25	0.17	2	0.26	0.06	14	0.18	0.03	13	0.16	0.03	10	0.14
BLOCKQP1	4.94	2	32.95	4.93	2	33.05	0.14	1	28.14	0.06	1	28.07	0.07	1	28.10
BLOCKQP2	4.93	2	32.84	4.87	2	33.00	0.14	1	28.11	0.07	1	28.11	0.07	1	27.99
BLOCKQP3	4.94	2	32.87	4.81	2	32.99	0.14	1	28.20	0.06	1	28.11	0.06	1	28.02
BLOWEYA	0.25	1	0.31	14.64	1	14.73	0.06	1	0.09	0.02	1	0.06	0.02	1	0.06
BLOWEYB	0.26	1	0.31	14.72	1	14.82	0.06	1	0.09	0.03	1	0.06	0.02	1	0.06
BLOWEYC	0.26	1	0.31	12.99	1	13.09	0.06	1	0.10	0.03	1	0.07	0.03	1	0.06
CONT-050	0.36	1	0.53	0.60	1	0.79	0.05	1	0.17	0.01	1	0.13	0.01	1	0.13
CONT-101	2.17	1	3.33	4.33	1	5.72	0.20	0	1.03	0.05	0	0.86	0.05	0	0.84
CONT-201	13.18	1	20.95	29.39	1	38.16	0.87	0	6.95	0.23	0	6.46	0.23	0	6.25
CONT1-10	2.16	1	3.34	4.14	1	5.47	0.20	1	1.11	0.05	1	0.93	0.05	1	0.93
CONT1-20	13.24	1	21.52	28.77	1	38.13	0.88	0	7.48	0.22	0	6.80	0.22	0	6.72
CONT5-QP	ran out of memory			ran out of memory			0.87	1	6.50	0.25	1	5.87	0.25	1	5.83
CVXQP1	0.48	2	0.75	0.46	2	0.79	0.14	2	0.33	0.06	1	0.22	0.06	1	0.22
CVXQP2	0.16	2	0.29	0.16	2	0.29	0.11	2	0.20	0.07	1	0.13	0.07	1	0.13
CVXQP3	0.71	2	1.29	505.39	2	506.87	0.16	1	0.59	0.05	1	0.47	0.05	1	0.47
DUALC1	0.03	1	0.04	0.03	1	0.05	0.01	2	0.03	0.01	2	0.02	0.01	1	0.02
DUALC2	0.02	2	0.04	0.02	2	0.04	0.02	2	0.04	0.01	89	0.13	0.01	150	0.20
DUALC5	0.03	2	0.06	0.03	2	0.06	0.02	2	0.05	0.01	9	0.04	0.01	26	0.07
DUALC8	0.07	2	0.17	0.07	1	0.17	0.02	1	0.12	0.01	3	0.12	0.01	2	0.12
GOULDQP2	0.48	2	0.72	0.47	2	0.78	0.26	4	0.45	0.12	1	0.20	0.12	1	0.21
GOULDQP3	0.47	2	0.71	0.46	2	0.76	0.26	4	0.47	0.12	1	0.21	0.12	1	0.21
KSIP	0.50	2	1.03	0.50	1	1.03	0.03	1	0.56	0.01	2	0.54	0.01	1	0.52
MOSARQP1	0.09	3	0.15	0.09	3	0.15	0.04	6	0.09	0.03	2	0.06	0.02	2	0.06
NCVXQP1	0.41	2	0.69	0.46	2	0.79	0.14	2	0.32	0.06	1	0.21	0.06	1	0.21
NCVXQP2	0.41	14	1.23	0.46	10	1.29	0.15	1	0.30	0.06	1	0.22	0.06	1	0.22
NCVXQP3	0.41	42	2.65	0.46	15	1.59	0.13	2	0.32	0.06	1	0.24	0.06	1	0.22
NCVXQP4	0.16	3	0.33	0.16	3	0.33	0.11	2	0.20	0.07	1	0.13	0.08	1	0.14
NCVXQP5	0.17	13	0.66	0.17	13	0.66	0.15	4	0.27	0.09	1	0.15	0.07	1	0.13
NCVXQP6	0.16	26	0.98	0.16	19	0.77	0.14	2	0.22	0.08	1	0.14	0.07	1	0.13
NCVXQP7	0.83	2	1.40	527.57	3	529.12	0.19	1	0.63	0.05	1	0.46	0.07	1	0.47
NCVXQP8	0.87	11	2.00	528.25	8	530.43	0.17	1	0.62	0.05	1	0.49	0.06	1	0.48
NCVXQP9	0.72	66	4.99	539.13	16	542.62	0.16	1	0.58	0.05	1	0.47	0.06	1	0.48
POWELL20	0.30	1	0.42	0.32	1	0.45	0.13	1	0.19	0.08	2	0.16	0.06	1	0.12
PRIMAL1	0.12	2	0.13	0.12	1	0.13	0.04	18	0.07	0.02	9	0.04	0.02	2	0.03
PRIMAL2	0.16	1	0.17	0.15	1	0.16	0.05	19	0.09	0.02	2	0.03	0.01	2	0.02
PRIMAL3	0.58	1	0.60	0.58	1	0.61	0.03	24	0.13	0.02	2	0.04	0.02	2	0.05
PRIMAL4	0.30	1	0.32	0.30	1	0.32	0.03	15	0.09	0.03	2	0.04	0.03	2	0.04
PRIMALC1	0.02	1	0.02	0.02	1	0.02	0.02	3	0.03	0.02	248	0.25	0.01	97	0.11
PRIMALC2	0.01	1	0.02	0.01	1	0.01	0.06	2	0.06	0.01	245	0.24	0.01	132	0.13
PRIMALC5	0.02	1	0.02	0.03	1	0.03	0.03	3	0.03	0.02	7	0.03	0.01	8	0.02
PRIMALC8	0.03	1	0.04	0.04	1	0.04	0.03	3	0.04	0.01	16	0.03	0.01	12	0.03
QPBAND	0.26	2	0.40	0.24	2	0.41	0.19	2	0.27	0.11	12	0.44	0.10	11	0.41
QPNBAND	0.24	1	0.33	0.24	2	0.42	0.19	2	0.27	0.10	10	0.39	0.12	8	0.34
QPCBOEI1	0.05	6	0.10	0.05	4	0.10	0.03	2	0.06	0.01	2	0.04	0.01	2	0.04
QPCBOEI2	0.10	6	0.12	0.10	3	0.12	0.02	2	0.03	0.01	1	0.02	0.01	1	0.02
QPNBOEI1	0.05	7	0.11	0.05	5	0.11	0.04	2	0.07	0.02	2	0.05	0.01	2	0.05
QPNBOEI2	0.11	6	0.13	0.09	3	0.11	0.02	2	0.03	0.01	1	0.02	0.01	1	0.02

Table E.3: CUTEr QP problems—residual decrease of at least 10^{-8} (continued)

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
QPCSTAIR	0.05	4	0.12	0.05	3	0.12	0.02	3	0.08	0.01	1	0.06	0.01	1	0.06
QPNSTAIR	0.06	4	0.12	0.06	3	0.12	0.02	4	0.09	0.01	1	0.06	0.01	1	0.06
SOSQP1	0.11	1	0.16	0.12	1	0.18	0.07	1	0.10	0.03	1	0.06	0.03	1	0.06
STCQP2	0.13	1	0.19	0.14	1	0.20	0.05	4	0.12	0.03	255	3.08	0.03	4	0.09
STNQP2	0.31	1	0.43	0.32	5	0.58	0.12	4	0.28	0.06	1	0.14	0.06	5	0.23
UBH1	0.34	1	0.52	0.34	1	0.52	0.14	2	0.29	0.05	1	0.17	0.05	4	0.23

Table E.4: CUTEr QP problems—residual decrease of at least 10^{-8} and C given by (7.3.3)

	Explicit factors						Implicit factors								
	$G = H$			$G = I$			Family 1			Family 2(a)			Family 2(b)		
name	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total	fact.	iter.	total
AUG2DCQP	0.13	6	0.26	0.13	7	0.27	0.06	163	1.12	0.02	1158	7.94	0.02	233	1.61
AUG2DQP	0.12	6	0.24	0.12	7	0.26	0.06	159	1.09	0.02	72	0.55	0.02	247	1.74
AUG3DCQP	0.16	7	0.32	0.17	7	0.33	0.06	109	0.79	0.03	88	0.69	0.03	65	0.53
AUG3DQP	0.17	8	0.34	0.17	8	0.34	0.06	107	0.75	0.03	91	0.71	0.03	65	0.52
BLOCKQP1	4.94	2	33.05	4.93	2	32.97	0.14	1	28.12	0.06	2	28.18	0.07	2	28.17
BLOCKQP2	4.93	2	33.02	4.87	2	32.90	0.14	1	28.07	0.07	2	28.17	0.07	2	28.14
BLOCKQP3	4.94	2	32.92	4.81	3	32.82	0.14	1	28.11	0.06	2	28.09	0.06	2	28.16
BLOWEYA	0.25	5	0.36	14.64	4	14.82	0.06	1	0.09	0.02	1	0.06	0.02	1	0.06
BLOWEYB	0.26	5	0.37	14.72	4	14.90	0.06	1	0.09	0.03	1	0.06	0.02	1	0.06
BLOWEYC	0.26	5	0.37	12.99	4	13.18	0.06	1	0.10	0.03	1	0.07	0.03	1	0.06
CONT-050	0.36	7	0.67	0.60	8	1.03	0.05	8	0.22	0.01	19	0.27	0.01	19	0.27
CONT-101	2.17	6	3.96	4.33	7	6.97	0.20	0	1.02	0.05	0	0.86	0.05	0	0.84
CONT-201	13.18	6	24.10	29.39	6	43.18	0.87	0	6.92	0.23	0	6.37	0.23	0	6.30
CONT1-10	2.16	6	3.99	4.14	8	6.80	0.20	8	1.33	0.05	19	1.54	0.05	19	1.53
CONT1-20	13.24	6	24.70	28.77	7	43.92	0.88	0	7.53	0.22	0	6.72	0.22	0	6.75
CONT5-QP	ran out of memory			ran out of memory			0.87	1	6.55	0.25	36	13.90	0.25	36	13.30
CVXQP1	0.48	124	6.61	0.46	73	4.99	0.14	97	2.63	0.06	21	0.74	0.06	22	0.76
CVXQP2	0.16	116	3.66	0.16	85	2.72	0.11	180	3.77	0.07	27	0.71	0.07	15	0.45
CVXQP3	0.71	134	9.71	505.39	67	513.12	0.16	4	0.70	0.05	19	1.14	0.05	20	1.14
DUALC1	0.03	8	0.05	0.03	7	0.05	0.01	11	0.05	0.01	130	0.18	0.01	13	0.04
DUALC2	0.02	5	0.05	0.02	3	0.04	0.02	8	0.04	0.01	163	0.22	0.01	228	0.29
DUALC5	0.03	10	0.07	0.03	8	0.07	0.02	9	0.06	0.01	13	0.05	0.01	55	0.11
DUALC8	0.07	9	0.19	0.07	4	0.18	0.02	12	0.14	0.01	51	0.21	0.01	71	0.26
GOULDQP2	0.48	5	0.95	0.47	5	1.01	0.26	18	0.91	0.12	42	1.68	0.12	30	1.21
GOULDQP3	0.47	5	0.94	0.46	5	1.00	0.26	18	0.94	0.12	34	1.42	0.12	51	2.09
KSIP	0.50	6	1.05	0.50	8	1.07	0.03	15	0.61	0.01	6	0.57	0.01	5	0.54
MOSARQP1	0.09	14	0.26	0.09	13	0.25	0.04	50	0.26	0.03	14	0.12	0.02	14	0.11
NCVXQP1	0.41	9898	483.79	0.46	5925	352.31	0.14	91	2.46	0.06	21	0.73	0.06	22	0.74
NCVXQP2	0.41	9929	465.07	0.46	9929	582.50	0.15	4966	120.38	0.06	23	0.78	0.06	23	0.78
NCVXQP3	0.41	9997	466.65	0.46	8242	492.05	0.13	92	2.48	0.06	21	0.72	0.06	21	0.72
NCVXQP4	0.16	9489	296.79	0.16	8756	277.30	0.11	2693	52.80	0.07	28	0.74	0.08	16	0.48
NCVXQP5	0.17	9990	319.96	0.17	9973	320.34	0.15	9970	195.47	0.09	28	0.75	0.07	15	0.44
NCVXQP6	0.16	7284	209.70	0.16	9835	287.43	0.14	4658	85.75	0.08	27	0.72	0.07	15	0.44
NCVXQP7	0.83	9906	598.40	527.57	6192	1120.71	0.19	4	0.76	0.05	19	1.12	0.07	20	1.16
NCVXQP8	0.87	9918	640.50	528.25	9756	1523.06	0.17	4	0.76	0.05	19	1.20	0.06	20	1.17
NCVXQP9	0.72	9997	578.21	539.13	9884	1467.28	0.16	4	0.70	0.05	19	1.13	0.06	20	1.15
POWELL20	0.30	1	0.41	0.32	1	0.46	0.13	1	0.19	0.08	317	5.01	0.06	10	0.26
PRIMAL1	0.12	6	0.14	0.12	9	0.15	0.04	166	0.28	0.02	15	0.05	0.02	30	0.07
PRIMAL2	0.16	6	0.19	0.15	7	0.18	0.05	133	0.31	0.02	23	0.08	0.01	11	0.04
PRIMAL3	0.58	6	0.63	0.58	6	0.63	0.03	120	0.44	0.02	10	0.07	0.02	9	0.07
PRIMAL4	0.30	5	0.34	0.30	5	0.34	0.03	62	0.25	0.03	8	0.07	0.03	7	0.06
PRIMALC1	0.02	5	0.03	0.02	4	0.02	0.02	7	0.03	0.02	248	0.26	0.01	248	0.26
PRIMALC2	0.01	4	0.02	0.01	4	0.02	0.06	6	0.06	0.01	245	0.25	0.01	12	0.02
PRIMALC5	0.02	4	0.03	0.03	4	0.03	0.03	6	0.03	0.02	10	0.03	0.01	11	0.02
PRIMALC8	0.03	5	0.05	0.04	4	0.05	0.03	6	0.04	0.01	21	0.04	0.01	56	0.10
QPBAND	0.26	5	0.54	0.24	4	0.51	0.19	8	0.40	0.11	489	12.58	0.10	141	3.66
QPNBAND	0.24	9	0.69	0.24	6	0.61	0.19	7	0.38	0.10	131	3.43	0.12	76	2.06
QPCBOEI1	0.05	21	0.15	0.05	17	0.15	0.03	103	0.26	0.01	23	0.09	0.01	24	0.09
QPCBOEI2	0.10	19	0.15	0.10	18	0.15	0.02	4	0.03	0.01	2	0.03	0.01	2	0.03
QPNBOEI1	0.05	21	0.15	0.05	17	0.15	0.04	104	0.28	0.02	22	0.10	0.01	24	0.09
QPNBOEI2	0.11	18	0.16	0.09	18	0.15	0.02	4	0.03	0.01	2	0.02	0.01	2	0.02
QPCSTAIR	0.05	20	0.17	0.05	19	0.18	0.02	137	0.34	0.01	15	0.09	0.01	970	2.03
QPNSTAIR	0.06	20	0.18	0.06	19	0.18	0.02	125	0.32	0.01	15	0.09	0.01	11	0.09
SOSQP1	0.11	2	0.18	0.12	2	0.20	0.07	5	0.15	0.03	8	0.15	0.03	59	0.77
STCQP2	0.13	57	1.07	0.14	51	1.14	0.05	67	0.79	0.03	6029	71.04	0.03	5989	66.54
STNQP2	0.31	2402	87.38	0.32	529	18.89	0.12	930	22.62	0.06	1	0.14	0.06	5	0.24
UBH1	0.34	6	0.76	0.34	5	0.67	0.14	28	0.82	0.05	31	0.81	0.05	24	0.65

Appendix F

MATLAB® command descriptions

In the following appendix we describe some of the MATLAB® functions which we use in Chapter 8. The descriptions are adapted from those provided in the documentation available with MATLAB®.

F.1 The LU function

The command `[L,U,P] = LU(X)` returns a unit lower triangular matrix `L`, upper triangular matrix `U`, and permutation matrix `P` so that `P*X = L*U`.

A column permutation can also be introduced: `[L,U,P,Q] = LU(X)` returns unit lower triangular matrix `L`, upper triangular matrix `U`, a permutation matrix `P` and a column reordering matrix `Q` so that `P*X*Q = L*U` for sparse non-empty `X`.

`[L,U,P] = LU(X,THRESH)` controls pivoting in sparse matrices, where `THRESH` is a pivot threshold in [0,1]. Pivoting occurs when the diagonal entry in a column has magnitude less than `THRESH` times the magnitude of any sub-diagonal entry in that column. `THRESH = 0` forces diagonal pivoting. `THRESH = 1` is the default.

`[L,U,P,Q] = LU(X,THRESH)` also controls pivoting, but in addition a column permutation is returned, where `THRESH` is a pivot threshold in [0,1]. Given a pivot column j, the sparsest candidate pivot row i is selected such that the absolute value of the pivot entry is greater than or equal to `THRESH` times the largest entry in the column j. The magnitude of entries in `L` is limited to `1/THRESH`. A value of 1.0 results in conventional partial pivoting. The default value is 0.1. Smaller values tend to lead to sparser `LU` factors, but the solution

can become inaccurate. Larger values can lead to a more accurate solution (but not always), and usually an increase in the total work.

F.2 The `QR` function

The MATLAB® command `[Q,R] = QR(A)`, where `A` is m by n, produces an m by n upper triangular matrix `R` and an m by m unitary matrix `Q` so that `A = Q*R`.

If A is full then the command `[Q,R,E] = QR(A)` can be used. This produces unitary `Q`, upper triangular `R` and a permutation matrix `E` so that `A*E = Q*R`. The column permutation `E` is chosen so that `ABS(DIAG(R))` is decreasing.

Appendix G

CUTEr QP problems: Complete tables

The following appendix gives complete results corresponding to the numerical experiments carried out in Section 8.1.1.3. The total number of interior point iterations "k", the total number of PPCG iterations to solve the predictor step "Total Its 1", the total number of PPCG iterations to solve the corrector step "Total Its 2", the total amount of CPU time used "Total CPU" and the percentage of the CPU time which was spent finding the permutation "% Permutation" are given for different preconditioners.

Table G.1: CUTEr QP problems—Number of iterations used

Problem		Impl1	Impl2	Impl3	Impl4	Expl1	Expl2
AUG2DCQP_M	k	139	139	45	109	21	20
$n = 3280$	Total Its 1	6051	6051	2740	12060	30	2118
$m = 1600$	Total Its 2	6518	6518	3011	13846	21	2293
	Total CPU	108.00	108.00	165.60	195.04	94.62	59.93
	% Permutation	3.25	3.25	2.05	1.39	0	0
AUG2DQP_M	k	143	143	85	97	25	24
$n = 3280$	Total Its 1	23037	23037	24091	16426	40	3007
$m = 1600$	Total Its 2	24174	24174	29405	19020	25	3007
	Total CPU	299.88	299.88	506.05	253.40	156.09	88.25
	% Permutation	1.20	1.20	0.71	0.92	0	0
AUG3DCQP_M	k	17	17	15	16	11	11
$n = 3873$	Total Its 1	1475	1475	1297	20762	11	9869
$m = 1000$	Total Its 2	1568	1568	1336	20507	11	9689
	Total CPU	26.06	26.06	35.43	253.90	23.02	23.11
	% Permutation	1.80	1.80	1.35	0.18	0	0
AUG3DQP_M	k	14	14	14	15	12	12
$n = 3873$	Total Its 1	2612	2612	2817	20077	12	4058
$m = 1000$	Total Its 2	2722	2722	2822	20927	12	4324

Table G.1: CUTEr QP problems—Number of iterations used (continued)

Problem		Impl1	Impl2	Impl3	Impl4	Expl1	Expl2
	Total CPU	37.09	37.09	52.92	233.08	23.72	117.70
	% Permutation	1.02	1.02	0.74	0.18	0	0
BLOCKQP1	k	17	9	—	—	10	9
$n = 10011$	Total Its 1	43	23	—	—	10	14
$m = 5001$	Total Its 2	43	24	—	—	10	14
	Total CPU	6991.59	4499.18	—	—	220.36	199.50
	% Permutation	94.49	95.61	—	—	0	0
CONT-050	k	6	6	6	6	6	6
$n = 2597$	Total Its 1	35	35	35	51	6	42
$m = 2401$	Total Its 2	35	35	35	51	6	41
	Total CPU	7.88	7.88	9.99	8.43	63.72	14.45
	% Permutation	34.64	34.64	27.68	32.86	0	0
CONT-101	k	10	10	10	10	—	—
$n = 10197$	Total Its 1	36	36	36	33	—	—
$m = 10098$	Total Its 2	37	37	37	37	—	—
	Total CPU	83.29	83.29	98.25	75.19	—	—
	% Permutation	37.11	37.11	31.76	37.17	—	—
CONT-201	k	9	9	9	9	23	—
$n = 10197$	Total Its 1	43	43	43	72	49	—
$m = 10098$	Total Its 2	31	31	31	74	57	—
	Total CPU	859.61	859.61	864.26	880.43	2306.97	—
	% Permutation	57.23	57.23	57.35	55.52	0	—
CONT-300	k	9	9	9	9	memory	memory
$n = 90597$	Total Its 1	46	46	46	65	—	—
$m = 90298$	Total Its 2	43	43	43	65	—	—
	Total CPU	3791.34	3791.34	4047.43	3811.77	—	—
	% Permutation	63.84	63.87	60.28	63.58	—	—
CONT5-QP	k	18	18	18	—	9	9
$n = 40601$	Total Its 1	16452	16452	16452	—	9	17
$m = 40200$	Total Its 2	17532	17532	17532	—	9	17
	Total CPU	28564.53	28564.53	82627.57	—	638.08	651.66
	% Permutation	4.84	4.84	1.70	—	0	0
CONT1-10	k	6	6	6	6	memory	—
$n = 10197$	Total Its 1	37	37	37	56	—	—
$m = 9801$	Total Its 2	38	38	38	56	—	—
	Total CPU	67.92	67.92	80.27	70.30	—	—
	% Permutation	50.25	50.25	42.74	48.58	—	—
CVXQP1_M	k	12	12	13	14	11	11
$n = 1000$	Total Its 1	1066	1405	1140	8257	108	6754
$m = 500$	Total Its 2	1093	1415	1267	8238	109	6752
	Total CPU	9.50	15.45	17.22	105.70	8.88	68.26
	% Permutation	5.05	3.17	3.31	0.51	0	0

Table G.1: CUTEr QP problems—Number of iterations used (continued)

Problem		Impl1	Impl2	Impl3	Impl4	Expl1	Expl2
CVXQP2_M	k	13	13	13	20	12	14
$n = 1000$	Total Its 1	256	656	256	13859	260	7310
$m = 250$	Total Its 2	265	675	260	13815	263	6945
	Total CPU	4.04	7.18	4.98	132.39	7.24	71.05
	% Permutation	4.70	2.65	3.61	0.19	0	0
CVXQP3_M	k	11	11	11	11	10	10
$n = 1000$	Total Its 1	825	1132	765	3827	79	2362
$m = 750$	Total Its 2	871	1124	783	3688	79	2335
	Total CPU	9.27	15.05	14.01	40.23	12.96	47.02
	% Permutation	22.00	13.42	14.70	4.87	0	0
DUALC1	k	9	12	13	13	11	12
$n = 223$	Total Its 1	28	61	45	89	35	112
$m = 215$	Total Its 2	29	80	48	90	36	103
	Total CPU	0.78	1.75	1.31	1.87	0.99	1.22
	% Permutation	7.69	4.57	6.11	3.74	0	0
DUALC2	k	41	16	10	10	7	8
$n = 235$	Total Its 1	244	107	34	40	23	40
$m = 229$	Total Its 2	277	108	33	38	24	39
	Total CPU	3.56	2.36	0.99	1.04	0.65	0.71
	% Permutation	6.18	4.23	3.03	3.85	0	0
DUALC5	k	13	16	22	7	8	7
$n = 285$	Total Its 1	24	95	40	39	44	48
$m = 278$	Total Its 2	25	89	41	36	43	48
	Total CPU	1.21	2.47	2.24	1.05	1.01	0.87
	% Permutation	6.61	2.43	5.36	4.76	0	0
DUALC8	k	9	10	11	11	9	6
$n = 510$	Total Its 1	35	42	41	56	39	28
$m = 503$	Total Its 2	38	45	49	59	40	29
	Total CPU	1.97	2.63	2.74	2.93	2.25	1.47
	% Permutation	4.57	4.18	4.38	3.41	0	0
GOULDQP2	k	54	55	125	52	25	13
$n = 19999$	Total Its 1	351	24835	1498	20225	61538	72441
$m = 9999$	Total Its 2	2430	19989	15529	20312	50204	69779
	Total CPU	195.38	1657.18	1480.50	1396.20	21221.31	11900.73
	% Permutation	4.31	0.51	1.53	0.58	0	0
GOULDQP3	k	64	82	57	194	10	10
$n = 19999$	Total Its 1	4889	1175	17521	74205	34	78278
$m = 9999$	Total Its 2	5354	1852	21927	84847	33	77861
	Total CPU	578.96	203.27	3352.42	5033.66	18.93	11406.81
	% Permutation	1.99	7.37	0.31	0.70	0	0
KSIP	k	15	15	15	15	13	25
$n = 1021$	Total Its 1	29	29	29	220	13	378

Table G.1: CUTEr QP problems—Number of iterations used (continued)

Problem		Impl1	Impl2	Impl3	Impl4	Expl1	Expl2
$m = 1001$	Total Its 2	34	34	34	247	13	540
	Total CPU	19.99	19.99	20.46	23.51	13.99	32.25
	% Permutation	35.12	35.12	34.41	30.11	0	0
MOSARQP1	k	12	12	12	15	12	20
$n = 3200$	Total Its 1	4104	4116	4100	22282	34	36452
$m = 700$	Total Its 2	4080	4134	4089	21732	35	31205
	Total CPU	56.58	66.69	74.71	313.54	44.64	937.51
	% Permutation	0.44	0.37	0.33	0.10	0	0
PRIMALC1	k	34	34	34	—	28	—
$n = 239$	Total Its 1	104	104	104	—	28	—
$m = 9$	Total Its 2	125	125	125	—	28	—
	Total CPU	1.84	1.84	2.15	—	1.69	—
	% Permutation	7.61	5.61	7.44	—	0	—
PRIMALC2	k	46	46	46	—	23	—
$n = 238$	Total Its 1	71	71	71	—	23	—
$m = 7$	Total Its 2	101	101	101	—	23	—
	Total CPU	2.41	2.41	2.47	—	1.24	—
	% Permutation	8.71	8.71	10.53	—	0	—
PRIMALC5	k	44	44	44	—	21	65
$n = 295$	Total Its 1	300	300	300	—	21	12504
$m = 8$	Total Its 2	278	278	278	—	21	13272
	Total CPU	2.69	2.69	3.20	—	1.28	37.24
	% Permutation	8.55	8.55	7.81	—	0	0
PRIMALC8	k	42	42	42	—	24	—
$n = 528$	Total Its 1	213	213	213	—	24	—
$m = 8$	Total Its 2	206	206	206	—	24	—
	Total CPU	3.31	3.31	4.45	—	2.50	—
	% Permutation	8.76	8.76	7.64	—	0	—
PRIMAL1	k	13	13	13	13	13	13
$n = 410$	Total Its 1	1311	1311	1311	2604	13	1913
$m = 85$	Total Its 2	1322	1322	1322	2607	13	1882
	Total CPU	7.01	7.01	11.82	20.70	2.47	10.71
	% Permutation	4.71	4.71	2.62	1.74	0	0
PRIMAL2	k	12	12	12	12	12	12
$n = 745$	Total Its 1	832	832	832	3193	12	2415
$m = 96$	Total Its 2	805	805	805	3225	12	2438
	Total CPU	6.84	6.84	10.55	34.50	3.44	18.88
	% Permutation	8.63	4.82	5.50	1.62	0	0
PRIMAL3	k	12	12	12	12	12	12
$n = 856$	Total Its 1	1739	1739	1739	5562	12	2846
$m = 111$	Total Its 2	1705	1705	1705	5534	12	2831
	Total CPU	20.05	20.05	30.81	78.69	8.77	40.93

Table G.1: CUTEr QP problems—Number of iterations used (continued)

Problem		Impl1	Impl2	Impl3	Impl4	Expl1	Expl2
	% Permutation	6.48	6.48	4.25	1.65	0	0
PRIMAL4	k	10	10	10	11	10	10
$n = 10197$	Total Its 1	1436	1436	1436	7473	10	3722
$m = 10098$	Total Its 2	1407	1407	1407	7443	10	3721
	Total CPU	15.38	15.38	20.85	92.49	4.80	44.97
	% Permutation	6.37	6.37	4.84	1.21	0	0
QPCSTAIR	k	433	433	433	733	33	172
$n = 614$	Total Its 1	255952	255952	255952	425570	33	111810
$m = 356$	Total Its 2	234820	234820	234820	426534	33	105678
	Total CPU	1686.25	1686.25	2928.70	7220.83	12.38	1266.04
	% Permutation	0.36	0.36	0.22	0.15	0	0
QPNSTAIR	k	798	798	798	113	32	297
$n = 614$	Total Its 1	438152	438152	438152	58502	32	161547
$m = 356$	Total Its 2	372670	372670	372670	57028	32	169140
	Total CPU	2486.57	2486.57	4407.28	1214.57	12.64	2583.85
	% Permutation	0.49	0.49	0.28	0.14	0	0
SOSQP1	k	10	10	15	—	—	—
$n = 5000$	Total Its 1	10	10	16	—	—	—
$m = 2501$	Total Its 2	11	11	19	—	—	—
	Total CPU	3.66	3.66	6.82	—	—	—
	% Permutation	12.84	12.84	9.53	—	—	—
STCQP2	k	16	16	16	16	16	16
$n = 8193$	Total Its 1	16	63	16	3541	63	3645
$m = 4095$	Total Its 2	16	71	16	3508	71	3626
	Total CPU	8.38	8.64	9.08	107.92	9.47	165.79
	% Permutation	18.38	18.26	16.52	1.33	0	0

Appendix H

CUTEr Inequality constrained QP problems: Complete tables

The following appendix gives complete results corresponding to the numerical experiments carried out in Section 8.1.2.1. The total number of interior point iterations "k", the total number of PPCG iterations to solve the resulting saddle point system "Its", and the total amount of CPU time used "CPU" are given for different preconditioners. Performance profiles based on this data are also provided.

Table H.1: CUTEr QP problems—Number of iterations used

Problem		Expl1	Expl2	Fam1a	Fam1b	Fam1c	Fam2a	Fam2b	Fam2c	Fam2d	Fam3a	Fam3b
AUG2DC-QP_M	k	13	13	13	13	13	13	13	13	13	13	13
	Its	13	24	1000	941	892	895	895	895	972	897	975
	CPU	26.81	26.30	31.51	31.26	30.77	30.76	30.76	30.86	31.46	32.93	33.50
AUG2D-QP_M	k	13	13	13	13	13	13	13	13	13	13	13
	Its	13	41	1044	941	840	871	871	989	1026	839	1029
	CPU	26.09	26.45	31.57	30.95	30.36	30.59	30.53	31.35	31.68	32.28	33.95
AUG3DC-QP_M	k	26	26	97	105	95	111	200	105	79	88	85
	Its	507	473	3258	3616	3052	3817	4522	3293	3046	2816	2769
	CPU	26.77	26.04	85.76	93.97	83.39	99.83	175.32	92.49	73.12	82.34	79.75
AUG3D-QP_M	k	14	14	47	52	52	76	200	57	50	75	68
	Its	93	108	2109	2318	2227	3124	5451	2591	2192	3199	2783
	CPU	12.52	12.84	46.21	50.97	50.53	73.26	185.38	56.93	49.02	78.59	69.97
CONT-050	k	13	13	13	13	13	13	14	13	13	13	13
	Its	483	352	538	583	567	564	278	545	532	571	537
	CPU	243.93	236.23	95.62	96.47	96.26	95.54	97.68	95.14	95.06	101.45	100.73
CONT-101	k	—	—	5	5	5	—	—	—	—	—	—
	Its	—	—	5	5	5	—	—	—	—	—	—
	CPU	—	—	3313	3351	3566	—	—	—	—	—	—
CONT1-10	k	17	17	25	18	24	27	54	24	25	25	25
	Its	835	677	2227	1517	2096	2015	1502	2061	2243	2082	2245
	CPU	9193	8973	13172	8960	12108	13478	25933	12542	13345	12647	12354
CVXQP1_M	k	27	31	45	42	47	34	48	51	47	—	—
	Its	1802	3256	9522	6493	10460	5348	7471	13744	9991	—	—
	CPU	27.07	42.31	34.51	30.17	36.72	25.57	35.75	48.73	36.58	—	—

Table H.1: CUTEr QP problems—Number of iterations used (continued)

Problem		Expl1	Expl2	Fam1a	Fam1b	Fam1c	Fam2a	Fam2b	Fam2c	Fam2d	Fam3a	Fam3b
CVXQP2_M	k	24	27	27	24	24	24	23	25	26	—	—
	Its	2161	3596	4452	1233	3800	1214	1281	8331	4254	—	—
	CPU	13.73	14.62	12.71	8.25	10.60	8.10	8.32	23.02	12.34	—	—
CVXQP3_M	k	71	101	157	122	146	121	135	148	141	—	—
	Its	2229	5489	16085	16842	15452	15028	16007	16102	15297	—	—
	CPU	92.36	150.83	103.46	103.66	96.50	93.86	102.15	98.55	93.84	—	—
DUALC2	k	200	13	200	27	200	200	200	52	7	82	13
	Its	409	2277	10431	3526	12773	17601	743	6882	851	12673	590
	CPU	10.38	3.01	22.78	5.75	26.73	31.69	8.84	10.62	1.32	25.83	1.56
DUALC5	k	27	200	200	200	9	200	200	200	200	200	26
	Its	200	24460	5518	4088	271	11104	605	6843	9489	3335	962
	CPU	1.40	54.22	18.98	17.49	0.89	25.15	10.55	18.45	21.17	16.83	3.10
KSIP	k	76	15	22	22	14	7	7	7	7	7	7
	Its	76	41	103	157	51	12	12	13	13	12	13
	CPU	80.20	16.17	19.78	18.89	11.63	8.12	8.04	8.05	8.09	8.18	8.15
MOSARQP1	k	18	28	6	7	10	10	10	8	11	12	12
	Its	72	390	138	87	227	105	105	107	134	159	89
	CPU	6.55	12.66	2.44	2.68	3.95	3.71	3.52	2.82	3.69	4.75	3.81
PRIMAL1	k	11	8	7	7	7	11	11	13	8	13	8
	Its	200	31	404	302	411	187	187	174	134	168	135
	CPU	1.55	0.99	1.13	0.97	1.17	0.96	0.95	1.05	0.69	1.23	0.81
PRIMAL2	k	8	8	7	7	7	8	8	7	8	8	16
	Its	124	80	520	368	555	149	149	150	187	151	813
	CPU	1.71	1.54	1.93	1.65	2.10	1.15	1.16	1.07	1.25	1.32	4.22
PRIMAL3	k	13	13	9	9	9	8	8	6	8	8	8
	Its	438	293	1652	1118	1623	195	195	182	233	197	234
	CPU	8.74	6.87	8.74	6.57	8.77	2.55	2.56	2.07	2.74	2.83	2.99
PRIMAL4	k	21	12	8	9	8	6	6	6	6	6	6
	Its	310	46	1499	1416	1547	150	150	194	192	145	193
	CPU	8.36	3.53	7.22	7.08	7.36	1.74	1.75	1.93	1.92	1.87	2.07
QPBAND	k	13	7	9	10	7	7	7	7	9	7	7
	Its	73	35	191	171	163	151	151	367	337	200	147
	CPU	683.91	390.09	488.90	539.15	391.10	392.88	392.93	396.28	492.66	395.95	391.72
QPCSTAIR	k	56	64	60	59	60	59	63	61	63	60	59
	Its	1190	2806	3319	3116	3614	3068	2764	5326	3138	3605	3264
	CPU	25.90	36.24	12.51	11.68	12.86	11.44	13.71	17.14	12.19	15.26	14.35
SOSQP1	k	6	6	8	9	8	9	9	9	8	6	15
	Its	6	6	11	12	11	12	13	12	11	15	33
	CPU	77.93	88.51	59.76	66.44	59.71	66.29	66.01	66.09	59.40	46.50	105.80
STCQP2	k	74	74	73	73	73	73	74	73	74	93	75
	Its	1146	2216	4355	5541	5729	5485	5196	14471	4378	32656	5639
	CPU	702.32	807.47	316.91	396.21	332.36	398.94	434.13	450.43	324.74	2802.36	354.24

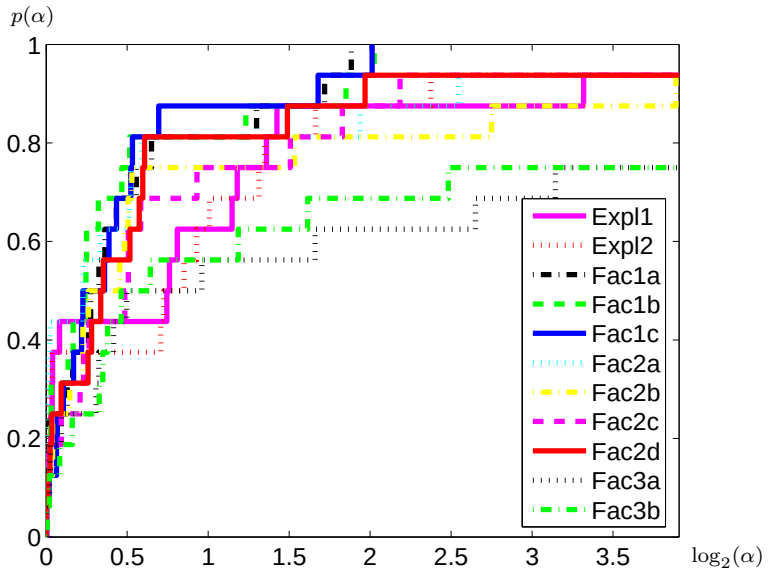

Figure H.1: Performance profile, $p(\alpha)$: CPU time (seconds) to solve QP programming problems.

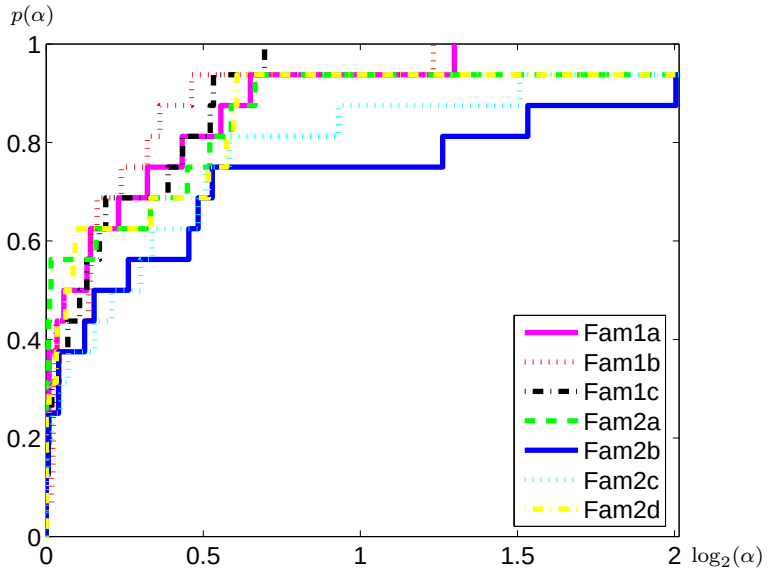

Figure H.2: Performance profile, $p(\alpha)$: CPU time (seconds) to solve QP programming problems.

References

[1] M. Arioli, *The use of QR factorization in sparse quadratic programming and backward error issues*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 825–839.

[2] M. Arioli and L. Baldini, *A backward error analysis of a null space algorithm in sparse quadratic programming*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 425–442.

[3] Kenneth J. Arrow, Leonid Hurwicz, and Hirofumi Uzawa, *Studies in linear and non-linear programming*, With contributions by H. B. Chenery, S. M. Johnson, S. Karlin, T. Marschak, R. M. Solow. Stanford Mathematical Studies in the Social Sciences, vol. II, Stanford University Press, Stanford, Calif., 1958.

[4] M. Avriel, *Nonlinear Programming: Analysis and Methods*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[5] O. Axelsson and V. A. Barker, *Finite element solution of boundary value problems. Theory and computation*, vol. 35 of Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. Reprint of the 1984 original.

[6] O. Axelsson and M. Neytcheva, *Preconditioning methods for linear systems arising in constrained optimization problems*, Numer. Linear Algebra Appl., 10 (2003), pp. 3–31. Dedicated to the 60th birthday of Raytcho Lazarov.

[7] R. H. Bartels and G. H. Golub, *The simplex method of linear programming using LU decompositions*, Communications of the ACM, 12 (1969), pp. 266–268.

[8] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numerica, 14 (2005), pp. 1–137.

[9] L. Bergamaschi, J. Gondzio, and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Comput. Optim. Appl., 28 (2004), pp. 149–171.

[10] G. Biros and O. Ghattas, *A Lagrange-Newton-Krylov-Schur method for PDE-constrained optimization*, SIAG/Optimization Views-and-News, 11 (2000), pp. 12–18.

[11] James R. Bunch, *Equilibration of symmetric matrices in the max-norm*, J. Assoc. Comput. Mach., 18 (1971), pp. 566–572.

[12] ——, *Partial pivoting strategies for symmetric matrices*, SIAM J. Numer. Anal., 11 (1974), pp. 521–528.

[13] J. R. Bunch and B. N. Parlett, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., 8 (1971), pp. 639–655.

[14] Y. Chabrillac and Crouzeix J.-P., *Definiteness and semidefiniteness of quadratic forms revisited.*, Linear Algebra and its Applications, 63 (1984), pp. 283–292.

[15] T. F. Coleman, *Linearly constrained optimization and projected preconditioned conjugate gradients*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, J. Lewis, ed., SIAM, Philadelphia, USA, 1994, pp. 118–122.

[16] T. F. Coleman and A. Pothen, *The null space problem. I. Complexity*, SIAM J. Algebraic Discrete Methods, 7 (1986), pp. 527–537.

[17] ——, *The null space problem. II. Algorithms*, SIAM J. Algebraic Discrete Methods, 8 (1987), pp. 544–563.

[18] Thomas F. Coleman and Arun Verma, *A preconditioned conjugate gradient approach to linear equality constrained minimization*, Comput. Optim. Appl., 20 (2001), pp. 61–72.

[19] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, The MIT Electrical Engineering and Computer Science Series, MIT Press, Cambridge, MA, 1990.

[20] G. Debreu., *Definite and semidefinite quadratic forms*, Econometrica, 20 (1952), pp. 295–300.

[21] H. S. Dollar, N. I. M. Gould, and A. J. Wathen, *On implicit-factorization constraint preconditioners*, in Large-Scale Nonlinear Optimization, Gianni G. Di Pillo and M. Roma, eds., vol. 83 of Nonconvex Optimization and Its Applications, Heidelberg, Berlin, New York, 2006, Springer Verlag, pp. 61–82.

[22] I. S. Duff, *MA57 - a code for the solution of sparse symmetric definite and indefinite systems*, ACM Transactions on Mathematical Software, 30 (2004), pp. 118–144.

[23] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct methods for sparse matrices*, Monographs on Numerical Analysis, The Clarendon Press Oxford University Press, New York, second ed., 1989. Oxford Science Publications.

[24] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Transactions on Mathematical Software, 9 (1983), pp. 302–325.

[25] ——, *The design of MA48: a code for the direct solution of sparse unsymmetric linear systems of equations*, ACM Transactions on Mathematical Software, 22 (1996), pp. 187–226.

[26] A. L. Dulmage and N. S. Mendelsohn, *Two algorithms for bipartite graphs*, J. Soc. Indust. Appl. Math., 11 (1963), pp. 183–194.

[27] H. C. Elman, *Preconditioning for the steady-state Navier-Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299–1316.

[28] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: with Applications in Incompressible Fluid Dynamics*, Oxford University Press, Oxford, 2005.

[29] Mark Embree, *The tortoise and the hare restart GMRES*, SIAM Rev., 45 (2003), pp. 259–266.

[30] J. A. Fessler and S. D. Booth, *Conjugate-gradient preconditioning methods for shift-variant PET image reconstruction*, IEEE Trans. Image Process., 8 (1999), pp. 688–699.

[31] B. Fischer, A. Ramage, D. J. Silvester, and A. J. Wathen, *Minimum residual methods for augmented systems*, BIT, 38 (1998), pp. 527–543.

[32] J. J. H. Forrest and J. A. Tomlin, *Updated triangular factors of the basis to maintain sparsity in the product form simplex method*, Math. Programming, 2 (1972), pp. 263–278.

[33] A. Forsgren, P. E. Gill, and J. D. Griffin, *Iterative solution of augmented systems arising in interior methods*, Tech. Report NA-05-03, University of California, San Diego, August 2005.

[34] A. Forsgren, P. E. Gill, and M. H. Wright, *Interior methods for nonlinear optimization*, SIAM Rev., 44 (2002), pp. 525–597 (2003).

[35] D. M. Gay, *Electronic mail distribution of linear programming test problems*. Mathematical Programming Society COAL Newsletter, December 1985. See `http://www.netlib.org/lp/data/`.

[36] J. R. Gilbert, C. Moler, and R. Schreiber, *Sparse matrices in Matlab: design and implementation*, SIAM Journal on Matrix Analysis and Applications, 13 (1992), pp. 333–356.

[37] P. E. Gill, W. Murray, D. B. Ponceleón, and M. A. Saunders, *Preconditioners for indefinite systems arising in optimization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 292–311.

[38] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: an SQP algorithm for large-scale constrained optimization*, SIAM J. Optim., 12 (2002), pp. 979–1006.

[39] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1981.

[40] G. H. Golub and C. Greif, *On solving block-structured indefinite linear systems*, SIAM J. Sci. Comput., 24 (2003), pp. 2076–2092.

[41] G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, third ed., 1996.

[42] N.I.M. Gould, *On practical conditions for the existence and uniqueness of solutions to the general equality quadratic-programming problem.*, Mathematical Programming, 32 (1985), pp. 90–99.

[43] N. I. M. Gould, *Iterative methods for ill-conditioned linear systems from optimization*, in Nonlinear Optimization and Related Topics, G. DiPillo and F. Giannessi, eds., Dordrecht, The Netherlands, 1999, Kluwer Academic Publishers, pp. 123–142.

[44] N. I. M. Gould, M. E. Hribar, and J. Nocedal, *On the solution of equality constrained quadratic programming problems arising in optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 1376–1395.

[45] N. I. M. Gould and S. Leyffer, *An introduction to algorithms for nonlinear optimization*, Tech. Report RAL-TR-2002-031, Rutherford Appleton Laboratory, 2002.

[46] N. I. M. Gould, S. Lucidi, M. Roma, and P. L. Toint, *Solving the trust-region subproblem using the Lanczos method*, SIAM J. Optim., 9 (1999), pp. 504–525.

[47] N. I. M. Gould, D. Orban, and P. L. Toint, *CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited*, ACM Transactions on Mathematical Software, 29 (2003).

[48] N. I. M. Gould, D. Orban, and P. L. Toint, *GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization*, ACM Trans. Math. Software, 29 (2003), pp. 353–372.

[49] A. Greenbaum, *Iterative methods for solving linear systems*, vol. 17 of Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[50] C. Greif, G. H. Golub, and J. M. Varah, *Augmented Lagrangian techniques for solving saddle point linear systems*, tech. report, Computer Science Department, University of British Columbia, Vancouver, Canada, 2004.

[51] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436 (1953).

[52] N. J. Higham, *Accuracy and stability of numerical algorithms*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second ed., 2002.

[53] HSL, *A collection of Fortran codes for large-scale scientific computation.* See http://hsl.rl.ac.uk/, 2004.

[54] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, Combinatorica, 4 (1984), pp. 373–395.

[55] C. Keller, N. I. M. Gould, and A. J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.

[56] L. Lukšan and J. Vlček, *Indefinitely preconditioned inexact Newton method for large sparse equality constrained non-linear programming problems*, Numer. Linear Algebra Appl., 5 (1998), pp. 219–247.

[57] S. Mehrotra, *On finding a vertex solution using interior point methods*, Linear Algebra Appl., 152 (1991), pp. 233–253. Interior point methods for linear programming.

[58] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.

[59] B. A. Murtagh and M. A. Saunders, *Large-Scale linearly constrained optimization*, Math. Programming, 14 (1978), pp. 41–72.

[60] ——, *A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints*, Math. Programming Stud., (1982), pp. 84–117.

[61] N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 778–795.

[62] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer Series in Operations Research, Springer-Verlag, New York, 1999.

[63] A. R. L. Oliveira, *A New Class of Preconditioners for Large-Scale Linear Systems from Interior Point Methods for Linear Programming*, Doctor of Philosophy, Rice University, Houston, Texas, April 1997.

[64] C. C. Paige and M. A. Saunders, *Solutions of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[65] I. Perugia and V. Simoncini, *Block-diagonal and indefinite symmetric preconditioners for mixed finite element formulations*, Numer. Linear Algebra Appl., 7 (2000), pp. 585–616.

[66] I. Perugia, V. Simoncini, and M. Arioli, *Linear algebra methods in a mixed approximation of magnetostatic problems*, SIAM J. Sci. Comput., 21 (1999), pp. 1085–1101.

[67] B. T. Polyak, *The conjugate gradient method in extremal problems*, U.S.S.R. Computational Mathematics and Mathematical Physics, 9 (1969), pp. 94–112.

[68] Alex Pothen and Chin-Ju Fan, *Computing the block triangular form of a sparse matrix*, ACM Trans. Math. Software, 16 (1990), pp. 303–324.

[69] J. K. Reid, *On the method of conjugate gradients for the solution of large sparse systems of linear equations*, in Large sparse sets of linear equations (Proc. Conf., St. Catherine's Coll., Oxford, 1970), Academic Press, London, 1971, pp. 231–254.

[70] M. Rozložník and V. Simoncini, *Krylov subspace methods for saddle point problems with indefinite preconditioning*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 368–391.

[71] T. Rusten and R. Winther, *A preconditioned iterative method for saddlepoint problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–904.

[72] Y. Saad, *Iterative methods for sparse linear systems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, second ed., 2003.

[73] Y. Saad and M. H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[74] W. H. A. Schilders, *A preconditioning technique for indefinite linear systems*, Technical Report RANA00-18, Technische Universiteit Eindhoven, 2000.

[75] ——, *Solution of indefinite linear systems using the LQ decomposition*. Unpublished, 2003.

[76] Robert B. Schnabel and Elizabeth Eskow, *A revised modified Cholesky factorization algorithm*, SIAM J. Optim., 9 (1999), pp. 1135–1148.

[77] D. J. Silvester and A. J. Wathen, *Fast iterative solution of stabilised Stokes systems. II. Using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367.

[78] V. Simoncini and M. Benzi, *Spectral properties of the Hermitian and skew-Hermitian splitting preconditioner for saddle point problems*, SIAM J. Matrix Anal. Appl., 26 (2004/05), pp. 377–389.

[79] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Rev., 43 (2001), pp. 235–286.

[80] K.-H. Tran, *Investigation of new preconditioning techniques*, master's thesis, Technische Universiteit Eindhoven, 2004.

[81] Lloyd N. Trefethen and David Bau, III, *Numerical linear algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[82] Lloyd N. Trefethen and Mark Embree, *Spectra and pseudospectra*, Princeton University Press, Princeton, NJ, 2005.

[83] H. A. van der Vorst, *Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[84] ——, *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, United Kingdom, 1 ed., 2003.

[85] R. C. Whaley, A. Petitet, and J. Dongarra, *Automated empirical optimization of software and the atlas project*, Parallel Computing, 27 (2001), pp. 3–35.

[86] M. H. Wright, *Ill-conditioning and computational error in interior methods for nonlinear programming*, SIAM J. Optim., 9 (1999), pp. 84–111.

[87] S. J. Wright, *Primal-dual interior-point methods*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

Index