STABLE AND EFFICIENT SOLUTION OF WEIGHTED LEAST-SQUARES PROBLEMS WITH APPLICATIONS IN INTERIOR POINT METHODS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Patricia D. Hough

August 1996

CTC96TR256

STABLE AND EFFICIENT SOLUTION OF WEIGHTED LEAST-SQUARES PROBLEMS WITH APPLICATIONS IN INTERIOR POINT METHODS

Patricia D. Hough, Ph.D.

Cornell University 1996

CTC96TR256

In this thesis, we consider two closely related problems. The first is a full-rank weighted least-squares problem with a weight matrix that is positive definite, diagonal, and extremely ill conditioned. The ill-conditioning can cause standard algorithms to compute solutions with, in some cases, no digits of accuracy. Theory suggests the existence of an algorithm that will compute an accurate solution despite the ill-conditioning in the weight matrix. We describe a new algorithm, the Complete Orthogonal Decomposition (COD) Algorithm, for solving the weighted least-squares problem and show that it has this desirable property. In addition, the COD Algorithm is based on standard, well-understood techniques and is straightforward to implement.

A natural application for the weighted least-squares problem described in the previous paragraph is interior point methods for linear programming. We discuss the problem in this context, and describe how the COD algorithm can be extended

and used in this setting. Unlike other algorithms, this one is stable for interior point methods without assuming nondegeneracy in the linear programming instance. Computational experiments indicate that it is more reliable than other algorithms when the problem is near degenerate.

The second problem involves a particular interior point algorithm. In 1994, Vavasis and Ye proposed a new primal-dual path-following interior point method, the layered-step interior point (LIP) method. This algorithm interleaves traditional steps with longer, layered least-squares (LLS) steps. Computation of a LLS step requires solving a weighted least-squares problem similar to the one described above, but the weight matrix also has the property that the weights fall into well-separated groups. This additional structure allows the problem to be broken down into smaller, constrained problems with well-conditioned weight matrices. The smaller problems can then be solved stably with standard algorithms, and the LLS step can be computed.

Vavasis and Ye did not propose a particular algorithm for solving the LLS problem. In this thesis, we present an algorithm based on Cholesky factorization. The algorithm is such that a modified version of the sparse Cholesky code of Ng and Peyton of Oak Ridge National Laboratories can be used. Thus, the theoretical results are straightforward, and this algorithm proves to be accurate and efficient in practice.

BIOGRAPHICAL SKETCH

Not so long ago, in the nearby land of Fairfax, Virginia, Patricia D. Hough was born to Roland and Roberta Hough. Five years later, her family (which included two younger sisters, Beth and Linda) moved to a rural area of Northern Virginia. It was there, near the small, no-stoplight town of Round Hill, where Patty (as she prefers to be called) grew up. Though she was a quiet, country girl, she began to learn the art of music and the sport of softball at a young age. With much hard work, she excelled at the piano and in the outfield, as well as in various academic endeavors.

In 1987, Patty graduated from Loudoun Valley High School and set out into the world, drawn by legends of cities with stoplights. She attended Lynchburg College, where she majored in mathematics with minors in computer science and chemistry. She also found time to play for the school softball team. While all of this was exciting and challenging, Patty began to realize that there must be something more. Driven by her curiosity and ambition, she decided to attend graduate school after completing her undergraduate degree.

In 1991, Patty moved to Ithaca, New York to pursue her doctorate in applied mathematics at Cornell University. After five years of chasing her goal, it seems that she has finally achieved it. Upon completing the requirements for her Ph.D., Patty will move on, leaving behind not only this thesis, but also a legacy of pull-hitting and mischief. (Although, she claims the mischief is really the work of her evil twin.) Her adventures now take her to the far-off land of Livermore, California, where she will begin a postdoctoral position in the Scientific Computing Division of Sandia National Laboratories.

A aquel que me dió la libertad de escribir esta página con my propio estilo.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my thesis advisor, Steve Vavasis. Without his unlimited availability, endless patience, sound advice, and encouraging words, this thesis would surely not exist. I would also like to thank the other members of my committee. It was the enthusiasm of Nick Trefethen in the classroom that secured my interest in numerical analysis and scientific computing, and he has offered many words of advice and encouragement during the last several years. Al Schatz always had a way of calming my nerves when they were raging out of control. His stories and reassuring words reminded me that everyone else (including my committee members!) is as human as I am, and the obstacles that every graduate student must overcome never seemed to loom so large after talking to him.

During my time as a graduate student, I spent two summers away from Cornell. I would like to thank Margaret Wright and David Gay for the opportunity to spend a summer working with them at Bell Labs. Likewise, I thank Rob Schreiber, Sid Chatterjee, and Tom Sheffler for hosting me for a summer at RIACS. (Thanks especially to Tom for the recommendation letters and such.) These experiences proved to be invaluable in more ways than one.

Graduate school can be trying experience, but it was made easier by those who keep the Center for Applied Mathematics running smoothly. My gratitude goes to Dolores Pendell for taking care of all of the administrative headaches, to Kaila Patel for keeping the computers up and running, and to both of them for their friendship and support. I would also like to thank John Guckenheimer and Lou Billera for looking after the needs of the students and for taking an active interest in them as

individuals. Finally, thanks to Ellie, not only for picking up after us, but for being a good friend and substitute mom.

The relationships I have developed over the past five years have played a significant role during my stay at Cornell. I would like to thank all of the CAMsters who have come and gone over the years for being an enjoyable bunch of people to work with. Special thanks to Hal Schenck for all of the back rubs and to my classmates - Don Allers, Jay Austin, Toby Driscoll, Stefano Herzel, Kim-Chuan Toh - without whom I never would have survived the first year, much less all five. I would like to thank the Grad IV folks for the fellowship and the good music, as well as the softball, volleyball, and basketball crowds for giving me a way to blow off steam (and to have fun doing it). Of course, I cannot forget my housemates - Tom Adelman, Ed Bueler, Javier Peña - who have put up with me for the last two years. There is one small group of people who, while they each fall into at least one of the above groups, are in a class by themselves: Jeff Baggett, Tom Bressoud, Rob Ghrist, Noelle Green-Willms (my P-i-C), Katy Simonsen, Stephen Wirkus, Randy Zounes. Words hardly seem adequate...

It is quite possible that I would have never considered graduate school were it not for the professors in the Math and Chemistry Departments of Lynchburg College. I would like to thank them for noticing that the quiet mouse in the back of their classrooms had potential, for putting her in a position to realize it, and for their continued support and encouragement. I would also like to thank all of those involved in the 1990 NSF-REU program in mathematics at the College of William and Mary for a positive experience that reinforced my decision to attend graduate school.

Most importantly, I would like to thank my family for standing behind me every step of the way and for always believing in me. They have been understanding about my infrequent visits, my absence at various gatherings, and forgotten birthdays. They have let me find my own way in life but never let me down when I needed them. They have always put my happiness before their own, and I can only hope that they have some idea of how much all of their love and support means to me.

As with everything else in this world, none of this work would have been possible without funding. Support was received from an NSF Presidential Young Investigator grant, with matching funds received from AT&T and Xerox Corporation. This work was also supported by ONR grant N00014-96-1-0050 and NSF grant DMS-9505155.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction*

1.1 The Problem

We consider solving the problem

$$\min_{\mathbf{y}\in\mathbb{R}^n} \left\| D^{1/2}\left(A\mathbf{y}-\mathbf{b}\right)\right\| \tag{1.1}$$

for $\mathbf{y}$, where $D \in \mathbb{R}^{m\times m}$ is diagonal and positive definite, $A \in \mathbb{R}^{m\times n}$, $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, and $\|\cdot\| = \|\cdot\|_2$. Two equivalent ways to write this problem are

$$A^T D A\mathbf{y} = A^T D\mathbf{b}$$

and

$$\begin{bmatrix} D^{-1} & -A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix},$$

*Portions of this chapter are reprinted from "Complete Orthogonal Decomposition for Weighted Least Squares" by Patricia D. Hough and Stephen A. Vavasis, to appear in SIAM J. Matrix Anal. Appl., Vol. 18, No. 2, 1997. Copyright (C) 1997 Society for Industrial and Applied Mathematics.

which is a special case of an equilibrium system. Applications of equilibrium systems include optimization, finite elements, structural analysis, and electrical networks [37]. These applications are discussed in more detail in §1.2.

The following assumptions are made throughout the thesis.

A1. *A has rank n, i.e., full column rank.*

A1 implies that (1.1) is a full-rank weighted least-squares problem with a unique solution and, when combined with the definition of the problem, allows the use of a norm bound obtained independently by a number of researchers. The first to derive this bound was Dikin in 1974 [9], and the most recent were Stewart [36] and Todd [38]. See Forsgren [10] for a survey of this and related work. The bound in question is given in the following theorem.

Theorem 1.1.1 *Let $\mathcal{D}$ denote the set of all positive definite $m \times m$ real diagonal matrices. Let A be an $m \times n$ real matrix of rank n. If we define χ_A and $\bar{\chi}_A$ as follows:*

$$\begin{aligned} a)\ \chi_A &= \sup\{\left\| \left(A^T D A\right)^{-1} A^T D \right\| : D \in \mathcal{D}\},\ \textit{and} \\ b)\ \bar{\chi}_A &= \sup\{\left\| A \left(A^T D A\right)^{-1} A^T D \right\| : D \in \mathcal{D}\}, \end{aligned}$$

then both χ_A and $\bar{\chi}_A$ are finite.

In this theorem, the norm can be any matrix norm induced by a vector norm. However in this thesis, $\|\cdot\| = \|\cdot\|_2$. Similarly, the condition number of a matrix M is

the condition number of M in the 2-norm, i.e., $\kappa(M) = \kappa_2(M)$. We make one more assumption, which is more in the nature of a reminder than a precise mathematical statement.

A2. *D is very ill conditioned.*

A discussion of the ill-conditioning of D in applications is included in §1.2. This assumption indicates that the coefficient matrix of the least-squares problem can also be ill conditioned. For this reason, the methods typically used to solve least-squares problems can give highly inaccurate solutions $\mathbf{y}$, as argued by Vavasis [41]. Our concern is with the effects of the ill-conditioning of D, so the following definition of stability will be used from this point on.

Definition 1.1.1 *An algorithm for* (1.1) *is* **stable** *if, in the presence of finite-precision arithmetic, an error bound of the form*

$$\|\mathbf{y} - \hat{\mathbf{y}}\| \leq \epsilon \cdot f(A) \cdot \|\mathbf{b}\| \tag{1.2}$$

is satisfied, where $\mathbf{y}$ *is the true solution,* $\hat{\mathbf{y}}$ *is the computed solution,* $f(A)$ *is some function of* A *not depending on* D*, and* $\epsilon > 0$ *is machine roundoff.*

For the purposes of the analysis in Chapter 2, other standard terminology is modified analogously. For example, a well-conditioned matrix is one for which there is an upper bound on the condition number that does not depend on D. In order to show that the proposed algorithm is stable, then, we strive to obtain bounds on norms, condition numbers, and errors that do not depend on D.

1.2 Applications of Stable Weighted Least Squares

In this section we discuss three applications of (1.1) that involve ill-conditioned weight matrices and explain the role of the forward error bound (1.2) in these applications. In two of the three applications, we provide summaries of computational experiments from previous work.

Electrical networks. Perhaps the simplest application of (1.1) is to resistive networks with fixed voltage sources (batteries). In this case, D encodes the resistances of the resistors in the network, $\mathbf{y}$ encodes the voltages of the nodes, A encodes node-wire adjacency (A is a node-arc adjacency matrix; all of its entries are either 0, 1, or -1), and $\mathbf{b}$ encodes battery voltages.

The case when D is ill conditioned arises when there are highly varying resistances in the circuit. Highly varying resistances can occur, for instance, when one tries to model current leakage through insulators as part of the problem.

We have conducted small-scale computational experiments on electrical networks using the COD algorithm described in Chapter 2 and the NSH method [41], which also satisfies (1.2). For node-arc adjacency matrices, χ_A and $\bar{\chi}_A$ are bounded by $O(m)$, so (1.2) states that the voltages should be calculated to within machine precision multiplied by the norm of the battery voltages. When applied to the aforementioned problems, the COD and the NSH algorithms yield solutions with fifteen digits of accuracy, while textbook methods (e.g. [7]) routinely give answers without any significant digits of accuracy.

Finite element methods. A more sophisticated application of (1.1) is to solve boundary value problems of the form

$$\begin{aligned} \nabla \cdot (c \nabla u) &= 0 \quad \text{on } \Omega, \\ u &= g \quad \text{on } \partial\Omega \end{aligned}$$

for u. Here, c is a user-specified conductivity field on the domain Ω that must be positive at every point. The function g is the user-specified Dirichlet boundary data. This boundary value problem arises in many fields of science and engineering; an example application is the heat equilibrium equation, in which case u stands for the steady-state temperature field in the domain and c stands for thermal conductivity.

As argued in [40], it is possible to express the standard piecewise-linear finite element method for this boundary value problem as a weighted least-squares problem. The diagonal weight matrix D encodes the conductivity field c. The matrix A encodes geometric information about the triangulation. The vector $\mathbf{y}$ stands for the solution field u, and the vector $\mathbf{b}$ encodes the Dirichlet boundary data. Thus, the case when D is ill-conditioned corresponds to finite element problems with highly varying conductivity. This in turn corresponds to applications in which the domain is composed of varying materials.

The analysis in [40] shows that variants of $\chi_A, \bar{\chi}_A$ are modest for this geometry matrix A, at least in the case of finite element triangulations with bounded aspect ratio and with all dihedral angles bounded by $\pi/2$. Thus, any method satisfying the stability bound (1.2) should compute the finite element solution $\mathbf{y}$ to all significant digits of accuracy, relative to the boundary data. This solution $\mathbf{y}$ is still inexact for

u because of truncation error that is always present in finite element methods: our analysis addresses roundoff error rather than truncation error.

The computational experiments presented in [40] are larger-scale computations. In a problem with conductivity varying by 15 orders of magnitude, the traditional finite element solution method returns an answer with no significant digits because of roundoff error. In contrast, a variant of the NSH method (which takes advantage of the isotropic nature of the problem) [40] that satisfies our stability bound (1.2) returns an answer which was as accurate as could be hoped for compared to an analytically-derived solution, given the presence of truncation error. The COD method of Chapter 2 would require a similar modification to take advantage of isotropy.

Interior point methods. It is well-known that the system of equations for the Newton step in an interior point method can be expressed as a weighted least-squares problem [17] of the form (1.1). Thus, the COD algorithm can clearly be used in this situation. However, there are additional issues that arise in interior point methods. Since this is the application examined in this thesis, further discussion is postponed until Chapter 3.

1.3 Organization of Thesis

Notice that Theorem 1.1.1 suggests the existence of an algorithm that satisfies a stability bound like (1.2), but how to construct such an algorithm is not apparent from the theorem. The goals of Chapter 2 are to develop an algorithm to solve (1.1) and to show that it satisfies (1.2). The proposed algorithm, known as the complete

orthogonal decomposition (COD) algorithm, is based on standard methods. It is therefore easy to understand and straightforward to implement. A forward error analysis demonstrates that the algorithm does indeed compute a solution satisfying (1.2).

In Chapter 3, we set up the weighted least-squares problem in the context of interior point methods for linear programming. The COD algorithm can be used in this setting, and it can be extended to compute search directions with the accuracy required for interior point methods. Unlike for other algorithms, there are no assumptions about nondegeneracy in the linear programming instance. Numerical tests indicate that this algorithm performs better than others on near-degenerate problems.

Chapter 4 addresses the layered least-squares (LLS) problem. This problem arises in an interior point algorithm proposed by Vavasis and Ye, the layered-step interior point (LIP) algorithm. In this algorithm, traditional interior point steps are interleaved with longer, layered least-squares steps. The weighted least-squares problem that arises from the LLS step is simply (1.1) with weights that fall into well-separated groups. While the COD algorithm can be used to solve the LLS problem, the additional structure allows us to develop a more efficient approach which also satisfies (1.2). The algorithm and the results of computational experiments appear in Chapter 4.

The final chapter highlights of the results of this thesis, and it contains a discussion of problems that remain open for research. The final remarks draw together the main threads of the work and tie them in to the big picture.

Chapter 2

The COD Algorithm*

2.1 The Algorithm

The most common methods for solving weighted least-squares problems are those used to solve unweighted problems. These include QR factorization and solving the normal equations via Cholesky factorization. Such algorithms appear in standard textbooks, e.g. [16]. There are also some specialized algorithms for weighted least squares that appear in the literature. These will be discussed in §2.4. As indicated in Chapter 1, these algorithms may compute solutions with no digits of accuracy when the weight matrix is highly ill conditioned. However, it is possible to modify an existing algorithm in order to obtain one that will compute a solution satisfying (1.2), and thus the complete orthogonal decomposition algorithm is born. A description of the algorithm appears below.

*Portions of this chapter are reprinted from "Complete Orthogonal Decomposition for Weighted Least Squares" by Patricia D. Hough and Stephen A. Vavasis, to appear in SIAM J. Matrix Anal. Appl., Vol. 18, No. 2, 1997. Copyright (C) 1997 Society for Industrial and Applied Mathematics.

Algorithm: Complete Orthogonal Decomposition (COD)

Step 1: QR factor, with column pivoting, $A^T D^{1/2}$ to get

$$A^T D^{1/2} = QRP, \tag{2.1}$$

where Q is an $n \times n$ orthogonal matrix, R is an $n \times m$ upper triangular ("trapezoidal") matrix, and P is an $m \times m$ permutation matrix.

Step 2: Apply reduced QR factorization (without pivoting) to R^T to get

$$R^T = Z_1 U_1, \tag{2.2}$$

where Z_1 is an $m \times n$ matrix with orthonormal columns, and U_1 is an $n \times n$ upper triangular matrix.

Step 3: Solve the following system, via back substitution, for $\bar{\mathbf{y}}$:

$$U_1 \bar{\mathbf{y}} = Z_1^T P D^{1/2} \mathbf{b}. \tag{2.3}$$

Step 4: To obtain $\mathbf{y}$, multiply the result of Step 3 by Q:

$$\mathbf{y} = Q\bar{\mathbf{y}}. \tag{2.4}$$

Both QR factorizations can be computed with Householder reflections, though other

methods are also acceptable. Note that the QR factorization for the least-squares problem occurs in Step 2. The QR factorization in Step 1 is needed to make the algorithm stable. Solution of least-squares problems via QR factorization with column pivoting was introduced by Golub [15]. The term "complete orthogonal decomposition" refers to a factorization of the form QRZ in which Q and Z are orthogonal and R is triangular [16]. Therefore we have chosen this name for the above algorithm, which computes a particular kind of complete orthogonal decomposition.

In exact arithmetic, the complete orthogonal decomposition algorithm solves the weighted least-squares problem given by (1.1). Writing the problem as a system of equations gives

$$D^{1/2}A\mathbf{y} \stackrel{LS}{=} D^{1/2}\mathbf{b}.$$

After performing the QR factorization in Step 1, $D^{1/2}A$ can be replaced by the equivalent quantity $P^T R^T Q^T$. The system of equations becomes

$$P^T R^T Q^T \mathbf{y} \stackrel{LS}{=} D^{1/2}\mathbf{b}$$

or equivalently,

$$R^T Q^T \mathbf{y} \stackrel{LS}{=} P D^{1/2}\mathbf{b}.$$

Letting $\bar{\mathbf{y}} = Q^T\mathbf{y}$ constitutes a change of variables and transforms the above system of equations to

$$R^T \bar{\mathbf{y}} \stackrel{LS}{=} P D^{1/2}\mathbf{b},$$

which is again a least-squares problem. Steps 2 and 3 are a standard method for solving least-squares problems, so the result in exact arithmetic is the solution $\bar{\mathbf{y}}$ to the transformed problem.

Most of this chapter is devoted to analysis of the stability of the COD algorithm. Before giving a rigorous stability analysis of the algorithm, we offer an intuitive explanation of why this algorithm finds an accurate solution. The first step is a QR factorization of a matrix that is well conditioned up to a scaling of the columns. So, the result is a computed upper triangular matrix that is close to the exact upper triangular matrix. It would be useful to know something about the condition number of this matrix as well. To minimize confusion assume, without loss of generality, that $A^T D^{1/2}$ has been "pre-pivoted." This means that the columns of $A^T D^{1/2}$ are ordered in such a way that the norms of the first n columns are, loosely speaking, in decreasing order. In addition, the norms of the first n columns are larger than those of the last $m - n$ columns. One might suspect that this implies that the entries of $D^{1/2}$ are ordered in the same way. In other words, some inequality similar to

$$d_i^{1/2} \geq d_j^{1/2} \text{ for } i \leq j, 1 \leq i \leq n, 1 \leq j \leq m$$

might hold. This ordering becomes significant in the second step of the algorithm.

Recall that

$$R = Q^T A^T D^{1/2}.$$

Notice the $Q^T A^T$ is upper triangular. So let

$$\bar{R} = Q^T A^T.$$

Notice also that R is ill conditioned and that the ill-conditioning arises from $D^{1/2}$. We try to "offset" the effects of $D^{1/2}$ in the following naive way. Let $\bar{D} = D(1:n, 1:n)$,

and consider the following:

$$\bar{D}^{-1/2}R = \bar{D}^{-1/2}\bar{R}D^{1/2} = \begin{bmatrix} \left(\frac{d_1}{d_1}\right)^{1/2}\bar{r}_{11} & \cdots & \left(\frac{d_n}{d_1}\right)^{1/2}\bar{r}_{1n} & \cdots & \left(\frac{d_m}{d_1}\right)^{1/2}\bar{r}_{1m} \\ & \ddots & \vdots & & \vdots \\ & & \left(\frac{d_n}{d_n}\right)^{1/2}\bar{r}_{nn} & \cdots & \left(\frac{d_m}{d_n}\right)^{1/2}\bar{r}_{nm} \end{bmatrix}.$$

Recall that because our focus is on the ill-conditioning in D, "well-conditioned" means that the matrix under consideration has a condition number determined by a property of A independently of D. Thus, $\bar{R}$ is trivially well conditioned since it has the same singular values as A. If the weights are indeed in the order described above, then it is not difficult to show that there are upper bounds on all entries of $\bar{D}^{-1/2}R$. It can also be shown that there are upper bounds on the entries of $\left(\bar{D}^{-1/2}R(:,1:n)\right)^{-1}$. Using this information, it is not difficult to show that $\bar{D}^{-1/2}R$ (and hence $R^T\bar{D}^{-1/2}$) is well conditioned, i.e., R^T is well conditioned up to a scaling of the columns. In the second step, then, we have a least-squares problem with a coefficient matrix that is well conditioned up to a scaling of the columns, namely solve

$$\min_{\bar{\mathbf{y}}\in\mathbb{R}^n} \left\| R^T\bar{\mathbf{y}} - D^{1/2}\mathbf{b} \right\|$$

for $\bar{\mathbf{y}}$. A QR factorization of R^T yields the upper triangular matrix U_1 in (2.2) that is also well conditioned up to scaling of the columns. We show that U_1 is also well conditioned up to scaling of the rows (see (2.16) below). In traditional analysis, U_1 being well conditioned up to a scaling of rows indicates that the standard algorithms for (2.3) of our algorithm gives an accurate solution (e.g. inequality (3.1.1) from [16] combined with Theorem 2.7.3 of [16] shows that scaling rows does not change the error bounds).

The remainder of the chapter contains a detailed discussion of the topics mentioned in this section. A discussion in §2.2 of a numerical issue, namely checking for linear dependence among the rows of A, leads into the rigorous forward error analysis of the COD algorithm in §2.3.1 through §2.3.3. In §2.4, the analysis is then compared to the (backward error) analyses of algorithms found in the literature.

2.2 A Note on Numerical Tolerance

In the upcoming analysis, we assume throughout that any occurrence of exact linear dependence among the columns of A^T is always determined correctly in Step 1 of the algorithm (QR factorization with pivoting). This requires the use of a numerical tolerance. To illustrate this point, consider applying the algorithm when $D^{1/2} = \mathrm{diag}(1, 1, 1, 10^{-20})$ and

$$A^T = \begin{pmatrix} 1 & 1 & 0 & 3 \\ 0 & 1 & -1 & 0 \\ 1 & 0 & 1 & 7 \end{pmatrix}.$$

Observe that the third column of A^T is dependent on the first two. If the QR factorization of $A^T D^{1/2}$ were done in exact arithmetic, this dependence would be manifested as a 0 in the $(3,3)$ position of the factored $A^T D^{1/2}$ after the first two QR factorization steps, and column 4 would be chosen for the third pivot.

In finite-precision arithmetic, however, we would expect the $(3,3)$ entry to be on the order of machine epsilon rather than 0. Because column 4 is weighted by 10^{-20}, the unwanted residual in the $(3,3)$ position could cause column 3 to be chosen for

the third column pivot instead of column 4. Thus, without modification, ordinary QR-factorization with column pivoting procedure has missed a linear dependence.

We address this problem as follows: after the kth QR factorization step, we check whether the residual portion (that is, positions $k+1, \ldots, n$) of any uneliminated column has become very small (according to some tolerance level) with respect to the original norm of that column. If so, those entries are changed to zeros. Notice that this test requires very little additional work because the usual QR factorization algorithm with column pivoting already monitors the norms of the residual portions of the columns [15]. In this way, if there are exact dependences among the rows of A, the algorithm does not miss them.

If this numerical test fails to detect exact dependence, then the following stability analysis no longer holds. It can be shown that the test we have proposed will fail only in the case that there is near-dependence among the columns of A^T. However, in this case, the parameter χ_A given by Theorem 1.1.1 is large, and so the stability bound (which depends on χ_A and $\bar{\chi}_A$) is not practically applicable.

2.3 The Analysis

2.3.1 The First QR Factorization

The intuitive discussion in §2.1 asserted that the ordering of the weights produced in Step 1 of the algorithm is important in stabilizing the algorithm. Therefore, it is necessary to establish this order. The pivoting in Step 1 chooses a particular set of rows of $D^{1/2}A$. The corresponding rows of A form a basis for the row space of A.

Thus, we will refer to the rows chosen by the column pivoting as the "basis rows" of A. We must determine how the weight of a particular basis row compares to those of other rows in the same basis and to those of the rows not in that basis. In order to do this, we start with a general result about any set of rows that forms a basis for the row space of A.

Lemma 2.3.1 *Let B be an $n \times n$ matrix whose columns are an arbitrary set of n rows $\mathbf{a}_{i_1}^T, \ldots, \mathbf{a}_{i_n}^T$ of A that form a basis for the row space of A. Then*

$$\max_{1 \le j \le n} \left\| \mathbf{a}_{i_j} \right\| \le (\chi_A \left\| A \right\|) \cdot \min_{1 \le j \le n} \left\| \mathbf{a}_{i_j} \right\| . \tag{2.5}$$

Proof: Let B and $\mathbf{a}_{i_1}^T, \ldots, \mathbf{a}_{i_n}^T$ be as in the lemma. Without loss of generality, suppose that $\left\| \mathbf{a}_{i_n} \right\| = \min_{1 \le j \le n} \left\| \mathbf{a}_{i_j} \right\|$. Then write

$$B = \left[\hat{B}, \mathbf{a}_{i_n} \right]$$

and partition B^{-1} as:

$$B^{-1} = \begin{bmatrix} X \\ \mathbf{v}^T \end{bmatrix} .$$

Then

$$B^{-1}B = I = \begin{bmatrix} X\hat{B} & X\mathbf{a}_{i_n} \\ \mathbf{v}^T\hat{B} & \mathbf{v}^T\mathbf{a}_{i_n} \end{bmatrix} .$$

This means $\mathbf{v}^T\mathbf{a}_{i_n} = 1$. By the Cauchy-Schwarz inequality, $\left\| \mathbf{v} \right\| \ge 1/\left\| \mathbf{a}_{i_n} \right\|$. Also, $\left\| \mathbf{v} \right\| \le \left\| B^{-1} \right\|$ because $\mathbf{v}$ is a row of B^{-1}, and

$$\left\| B^{-1} \right\| \le \chi_A \tag{2.6}$$

(see [41] for the proof of (2.6)). Combining these inequalities yields

$$1/\left\|\mathbf{a}_{i_n}\right\| \leq \chi_A,$$

i.e.,

$$\min_{1\leq j\leq n}\left\|\mathbf{a}_{i_j}\right\| \geq 1/\chi_A.$$

Multiply both sides of this inequality by the inequality $\|A\| \geq \max_{1\leq j\leq n}\left\|\mathbf{a}_{i_j}\right\|$ to obtain

$$\max_{1\leq j\leq n}\left\|\mathbf{a}_{i_j}\right\| \leq (\chi_A \|A\|)\cdot \min_{1\leq j\leq n}\left\|\mathbf{a}_{i_j}\right\|,$$

as required. □

Notice that the above lemma implies that there is a lower bound on the norm of every column of any basis for the row space of A.

Suppose now that $k > 0$ steps of the factorization have been completed. Partition the resulting matrix by rows as follows:

$$AQ_1\cdots Q_k = \bar{A} = \begin{bmatrix} \alpha_{11} & 0 & \cdots & \mathbf{0}^T \\ \vdots & \ddots & \ddots & \vdots \\ \alpha_{k1} & \cdots & \alpha_{kk} & \mathbf{0}^T \\ \alpha_{k+1,1} & \cdots & \alpha_{k+1,k} & \bar{\mathbf{a}}_{k+1}^T \\ \vdots & & \vdots & \vdots \\ \alpha_{m,1} & \cdots & \alpha_{m,k} & \bar{\mathbf{a}}_m^T \end{bmatrix}.$$

Lemma 2.3.1 can be extended so that it applies to the residual portions of the rows of $\bar{A}$, i.e., $\bar{\mathbf{a}}_{k+1}^T, \ldots, \bar{\mathbf{a}}_n^T$, as follows.

Lemma 2.3.2 *Let B be an $n \times n$ matrix whose first k columns are the k rows of A, say $\mathbf{a}_{i_1}^T, \ldots, \mathbf{a}_{i_k}^T$, chosen by the column pivoting in the first k steps of the QR factorization. Let the remaining columns of B be arbitrary rows of A, say $\mathbf{a}_{i_{k+1}}^T, \ldots, \mathbf{a}_{i_n}^T$, such that the columns of B form a basis for the row space of A. As with $\bar{A}$ above, write*

$$Q_k^T \cdots Q_1^T B = \begin{bmatrix} \alpha_{i_1,1} & \cdots & \alpha_{i_k,1} & \alpha_{i_{k+1},1} & \cdots & \alpha_{i_n,1} \\ 0 & \ddots & \vdots & \vdots & & \vdots \\ \vdots & \ddots & \alpha_{i_k k} & \alpha_{i_{k+1} k} & & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \bar{\mathbf{a}}_{i_{k+1}} & \cdots & \bar{\mathbf{a}}_{i_n} \end{bmatrix}.$$

Then

$$\max_{k+1 \le j \le n} \left\| \bar{\mathbf{a}}_{i_j} \right\| \le (\chi_A \, \|A\|) \cdot \min_{k+1 \le j \le n} \left\| \bar{\mathbf{a}}_{i_j} \right\|. \tag{2.7}$$

Proof: Let $Q = Q_1 \cdots Q_k$, and let B be defined as above. Then

$$Q^T B = \begin{bmatrix} R & X \\ 0 & \bar{B} \end{bmatrix},$$

where R is upper triangular. Comparing this matrix to the partitioned one above, we see that the columns of $\bar{B}$ are $\bar{\mathbf{a}}_{i_{k+1}}, \ldots, \bar{\mathbf{a}}_{i_n}$. To prove Lemma 2.3.2, then, we need a lower bound on a typical column of $\bar{B}$. Notice that

$$(Q^T B)^{-1} = \begin{bmatrix} R^{-1} & -R^{-1} X \bar{B}^{-1} \\ 0 & \bar{B}^{-1} \end{bmatrix}.$$

Therefore,

$$\left\| \bar{B}^{-1} \right\| \le \left\| B^{-1} \right\| \le \chi_A.$$

Now, as above, turn an upper bound on $\left\|\bar{B}^{-1}\right\|$ into a lower bound on any column of $\bar{B}$. □

Using the previous two lemmas, we can determine the relationships between the weights of the basis rows and those of the nonbasis rows. For the remainder of the analysis assume, as in the intuitive discussion in §2.1, that the columns of $A^T D^{1/2}$ have been reordered so that no pivoting is necessary. This implies that not only do the first n columns of A^T form a basis for the column space of A^T, but there is also an order that has been imposed on the columns of A^T. Notice that the "pre-pivoting" also implies that $\bar{\mathbf{a}}_{i_j}$ becomes $\bar{\mathbf{a}}_j$ for the remainder of the chapter. So let B be the $n \times n$ matrix whose columns are rows $\mathbf{a}_1^T, \ldots, \mathbf{a}_n^T$ and let $d_1, \ldots, d_m$ denote the (re-ordered) entries of D.

Theorem 2.3.1 *Suppose the first $k \geq 0$ steps of the QR factorization have been completed. If $d_{k+1}^{1/2}$ is the weight assigned to $\mathbf{a}_{k+1} \in B$, and $d_j^{1/2}$ is the weight assigned to $\mathbf{a}_j \notin B$, then*

$$\frac{d_j}{d_{k+1}} \leq \left(\chi_A \left\|A\right\|\right)^4, \tag{2.8}$$

provided $\mathbf{a}_j$ is linearly independent of the first k basis vectors.

Proof: For $k \geq 0$, let $\bar{A}$ be as defined before Lemma 2.3.2. (Notice that when $k = 0$, this is just the matrix A, partitioned into rows.) Since the columns of B form

a basis for the row space of A,

$$\mathbf{a}_j = \sum_{i=1}^{n} c_i \mathbf{a}_i.$$

Assuming $\mathbf{a}_j$ is linearly independent of the first k basis rows implies

$$\mathbf{a}_j - \sum_{i=1}^{k} c_i \mathbf{a}_i = \sum_{i=k+1}^{n} c_i \mathbf{a}_i \neq \mathbf{0},$$

which means $c_i \neq 0$ for at least one i such that $k+1 \leq i \leq n$. Take $Q = Q_1 \cdots Q_k$. Then

$$Q^T \mathbf{a}_j - \sum_{i=1}^{k} c_i Q^T \mathbf{a}_i = \sum_{i=k+1}^{n} c_i Q^T \mathbf{a}_i,$$

and

$$\bar{\mathbf{a}}_j - \sum_{i=1}^{k} c_i \bar{\mathbf{a}}_i = \sum_{i=k+1}^{n} c_i \bar{\mathbf{a}}_i,$$

where $\bar{\mathbf{a}}_i$ is the residual portion of $\mathbf{a}_i$. Notice that $\bar{\mathbf{a}}_i = \mathbf{0}$ for $1 \leq i \leq k$. So

$$\bar{\mathbf{a}}_j = \sum_{i=k+1}^{n} c_i \bar{\mathbf{a}}_i,$$

where $c_i \neq 0$ for at least one i. Let l be such that $k+1 \leq l \leq n$ and $c_l \neq 0$. Then

$$\bar{\mathbf{a}}_l = \frac{1}{c_l} \left(\bar{\mathbf{a}}_j - \sum_{i=k+1, i \neq l}^{n} c_i \bar{\mathbf{a}}_i \right).$$

So, $\bar{B} = \{\bar{\mathbf{a}}_j, \bar{\mathbf{a}}_{k+1}, \ldots, \bar{\mathbf{a}}_{l-1}, \bar{\mathbf{a}}_{l+1}, \ldots, \bar{\mathbf{a}}_n\}$ is a basis for $\{\bar{\mathbf{a}}_{k+1}, \ldots, \bar{\mathbf{a}}_m\}$. Since (2.5) and (2.7) hold for any basis for the row space of A,

$$\|\bar{\mathbf{a}}_j\| \geq \frac{\max\left\{\|\bar{\mathbf{a}}_i\| : \bar{\mathbf{a}}_i \in \bar{B}\right\}}{\chi_A \|A\|}.$$

Recall that the columns of $A^T D^{1/2}$ have been reordered so that no pivoting is necessary. This means that there is an order imposed on the columns of $A^T D^{1/2}$. More specifically, at step $k+1$

$$d_j^{1/2} \|\bar{\mathbf{a}}_j\| \leq d_{k+1}^{1/2} \|\bar{\mathbf{a}}_{k+1}\|.$$

Thus,

$$\begin{aligned}
\frac{d_j}{d_{k+1}} &\leq \left(\frac{\|\bar{\mathbf{a}}_{k+1}\|}{\|\bar{\mathbf{a}}_j\|}\right)^2 \\
&\leq \left(\frac{\chi_A \cdot \|A\| \cdot \max_{k+1\leq i\leq n} \|\bar{\mathbf{a}}_i\|}{\max\left\{\|\bar{\mathbf{a}}_i\| : \bar{\mathbf{a}}_i \in \bar{B}\right\}}\right)^2 \\
&\leq \left(\frac{\chi_A \cdot \|A\| \cdot \max_{k+1\leq i\leq n} \|\bar{\mathbf{a}}_i\|}{\min_{k+1\leq i\leq n} \|\bar{\mathbf{a}}_i\|}\right)^2 \\
&\leq \left(\chi_A \|A\|\right)^4,
\end{aligned}$$

which is (2.8). □

It is also necessary to know the relationships between the weights of the basis rows of A. Suppose $\mathbf{a}_i, \mathbf{a}_j \in B$, where $i < j$. It follows from (2.5), (2.7), and the implicit order indicated by the absence of column pivoting that

$$\frac{d_j}{d_i} \leq \left(\chi_A \|A\|\right)^2 \leq \left(\chi_A \|A\|\right)^4. \tag{2.9}$$

Recall that the intuitive argument given in §2.1 relied on the weights being in the following order:

$$d_i^{1/2} \geq d_j^{1/2} \text{ for } i \leq j, 1 \leq i \leq n, 1 \leq j \leq m.$$

Theorem 2.3.1 indicates, however, that they are not ordered in exactly this way. Instead, this ordering holds up to scaling by a constant, i.e.,

$$d_i^{1/2} \geq \frac{d_j^{1/2}}{\left(\chi_A \|A\|\right)^2} \text{ for } i \leq j, 1 \leq i \leq n, 1 \leq j \leq m.$$

This bound is sufficient for the arguments that follow.

The second step of the algorithm performs a QR factorization on R^T. In order to analyze that step, then, it is necessary to know something about the condition of

R. The relationships between the weights of the rows of A are used in the proof of the following theorem, which states that R is well conditioned up to a scaling of the rows or the columns. Recall that for an $m \times n$ matrix M of rank n, $\kappa(M)$ is the condition number (in the 2-norm) of M, i.e.,

$$\kappa(M) = \|M\| \cdot \left\| \left(M^T M\right)^{-1} M^T \right\|.$$

Theorem 2.3.2 *Let* $C = \bar{D}^a R D^{-1/2-a}$, *where* $a \leq 0$ *and* $\bar{D} = D(1:n, 1:n)$. *If* $\tilde{C} = C(1:k, :)$, *then*

$$\kappa(\tilde{C}) \leq n^4 \cdot (\chi_A \|A\|)^{16a+2} \tag{2.10}$$

for any $1 \leq k \leq n$.

Proof: First, we must find an upper bound on $\left\|\tilde{C}\right\|$. Since $\tilde{C}$ is a submatrix of C, then $\left\|\tilde{C}\right\| \leq \|C\|$. Therefore, it is sufficient to show that there is an upper bound on $\|C\|$. Write C as follows:

$$C = \bar{D}^a R D^{-1/2-a} = \bar{D}^a Q^T A^T D^{1/2} D^{-1/2-a} = \bar{D}^a \bar{R} D^{-a},$$

where $\bar{R} = Q^T A^T = R D^{-1/2}$. If the entries of C are written explicitly,

$$C = \begin{bmatrix} \left(\frac{d_1}{d_1}\right)^a \bar{r}_{11} & \cdots & \left(\frac{d_n}{d_1}\right)^a \bar{r}_{1n} & \cdots & \left(\frac{d_m}{d_1}\right)^a \bar{r}_{1m} \\ & \ddots & \vdots & & \vdots \\ & & \left(\frac{d_n}{d_n}\right)^a \bar{r}_{nn} & \cdots & \left(\frac{d_m}{d_n}\right)^a \bar{r}_{nm} \end{bmatrix}.$$

Consider $\bar{R}_1 = \bar{R}(:, 1:n)$. Again, let B be the basis consisting of the first n columns of A^T. Then $\bar{R}_1 = Q^T B$. So, $\left|\frac{1}{\bar{r}_{ii}}\right| \leq \|B^{-1}\| \leq \chi_A$ for $1 \leq i \leq n$. If $\bar{\mathbf{r}}_i^T$ is the ith row

of $\bar{R}$, then $\left\|\bar{\mathbf{r}}_i^T\right\| \leq \|A\|$ for all $1 \leq i \leq n$. These facts, (2.8), and (2.9) imply the following:

$$\frac{1}{\chi_A} \leq |\bar{r}_{ii}| \leq \|A\|, 1 \leq i \leq n \text{ and} \tag{2.11}$$

$$\left\|d_i^a \bar{\mathbf{r}}_j^T D^{-a}\right\| \leq \|A\| \cdot (\chi_A \|A\|)^{4a}, 1 \leq i \leq j \leq n. \tag{2.12}$$

Recall that Theorem 2.3.1, and thus (2.11), holds only when $\mathbf{a}_j$ is linearly independent of the first $i-1$ basis vectors. We must now consider the case not covered by Theorem 2.3.1. Suppose that B and D are defined as before. For each nonbasis row $\mathbf{a}_j$ there is a $1 \leq k \leq n$ such that $\mathbf{a}_j$ is linearly independent of the first $k-1$ basis vectors, but is linearly dependent on the first k basis vectors. So,

$$\mathbf{a}_j = \sum_{i=1}^{k} c_i \mathbf{a}_i,$$

where $c_k \neq 0$. Now, suppose that k steps of the QR factorization have been completed. Then

$$Q_k^T \cdots Q_1^T \mathbf{a}_j = \sum_{i=1}^{k} c_i Q_k^T \cdots Q_1^T \mathbf{a}_i = \sum_{i=1}^{k} c_i \begin{bmatrix} \alpha_{1i} \\ \vdots \\ \alpha_{ii} \\ \mathbf{0} \end{bmatrix}$$

So, $\bar{\mathbf{a}}_j = \mathbf{0}$ (where $\bar{\mathbf{a}}_j$ is as in Lemma 2.3.2). After this point, transformations act only on $\bar{\mathbf{a}}_j$. This gives

$$Q^T \mathbf{a}_j = \sum_{i=1}^{k} c_i \begin{bmatrix} \alpha_{1i} \\ \vdots \\ \alpha_{ii} \\ \mathbf{0} \end{bmatrix},$$

telling us that $\bar{r}_{ij} = 0$ for $i > k$, so these entries do not contribute to the norm in (2.11). Thus, (2.11) holds even in the case where $\mathbf{a}_j$ is not linearly independent of the first $i-1$ basis vectors.

If $\mathbf{c}_i^T$ is i^{th} row of C, then

$$\begin{aligned} \|C\| &\leq \sum_{i=1}^{n} \left\|\mathbf{c}_i^T\right\| \\ &\leq n \cdot \max_{1\leq i\leq n} \left\|\mathbf{c}_i^T\right\| \\ &= n \cdot \max_{1\leq i\leq n} \left\|d_i^a \bar{\mathbf{r}}_i^T D^{-a}\right\| \\ &\leq n \cdot \|A\| \cdot (\chi_A \|A\|)^{4a}. \end{aligned} \tag{2.13}$$

The third line follows from the definition of C, and the fourth line follows from (2.11). The next step is to find an upper bound on $\left\|\left(\tilde{C}\tilde{C}^T\right)^{-1}\right\|$. Let $\tilde{C}_1 = \tilde{C}(:, 1:k)$. Notice that

$$\left\|\left(\tilde{C}\tilde{C}^T\right)^{-1}\right\| \leq \left\|\tilde{C}_1^{-T}\right\|^2.$$

If $C_1 = C(:, 1:n)$, it is easy to show that $\tilde{C}_1^{-T}$ is a submatrix of C_1^{-T}. To obtain an upper bound on $\left\|C_1^{-T}\right\|$, we use the following fact, which will be proved after the current proof.

Fact: If $C_1 = C(1:n, 1:n)$, then

$$\left\|C_1^{-T}\right\| \leq n \cdot \chi_A \cdot (\chi_A \|A\|)^{4a}.$$

So,

$$\begin{aligned} \left\|\left(\tilde{C}\tilde{C}^T\right)^{-1}\right\| &= \left\|\tilde{C}_1^{-T}\right\|^2 \\ &\leq \left\|C_1^{-T}\right\|^2 \\ &\leq \left[n \cdot \chi_A \cdot (\chi_A \|A\|)^{4a}\right]^2, \end{aligned}$$

and the bound on the condition number is

$$\begin{aligned}\kappa(\tilde{C}) &= \left\|\tilde{C}\right\| \cdot \left\|\tilde{C}^T \left(\tilde{C}\tilde{C}^T\right)^{-1}\right\| \\ &\leq \left\|\tilde{C}\right\|^2 \cdot \left\|\left(\tilde{C}\tilde{C}^T\right)^{-1}\right\| \\ &\leq n^4 \cdot (\chi_A \|A\|)^{16a+2}.\end{aligned}$$

Thus, the theorem is proved. □

The above theorem implies that R is not only well conditioned up to a scaling of the rows, but it is well conditioned up to a scaling of either the rows or the columns. This result will be useful later in the analysis.

Recall that we must still prove the fact used in the above proof. We state it in the form of the following lemma and give the proof below.

Lemma 2.3.3 *Let $C_1 = C(:, 1:n)$, where C is defined as in the previous theorem. Then*

$$\left\|C_1^{-T}\right\| \leq n \cdot \chi_A \cdot (\chi_A \|A\|)^{4a}. \tag{2.14}$$

Proof: Recall that

$$C = \bar{D}^a \bar{R} D^{-a},$$

where $\bar{D} = D(1:n, 1:n)$ and $\bar{R} = Q^T A^T = RD^{-1/2}$. So,

$$C_1 = \bar{D}^a \bar{R}(:, 1:n)\bar{D}^{-a}.$$

Notice that

$$\bar{R}(:, 1:n) = Q^T B,$$

where B is the row space basis consisting of the first n rows of A. If $L = Q^T B^{-T}$, then

$$\begin{aligned} C_1^{-T} &= \bar{D}^{-a} L \bar{D}^a \\ &= \begin{bmatrix} \left(\frac{d_1}{d_1}\right)^a l_{11} & & & \\ \left(\frac{d_2}{d_1}\right)^a l_{21} & \left(\frac{d_2}{d_2}\right)^a l_{22} & & \\ \vdots & \vdots & \ddots & \\ \left(\frac{d_n}{d_1}\right)^a l_{n1} & \left(\frac{d_n}{d_2}\right)^a l_{n2} & \cdots & \left(\frac{d_n}{d_n}\right)^a l_{nn} \end{bmatrix}. \end{aligned}$$

If $\mathbf{c}_i^T$ is the i^{th} row of C_1^{-T}, then

$$\begin{aligned} \left\|C_1^{-T}\right\| &\leq \sum_{i=1}^{n} \left\|\mathbf{c}_i^T\right\| \\ &\leq n \cdot \max_{1\leq i\leq n} \left\|\mathbf{c}_i^T\right\| \\ &\leq n \cdot (\chi_A \|A\|)^{4a} \cdot \max_{1\leq i\leq n} \left\|\mathbf{l}_i^T\right\| \\ &\leq n \cdot (\chi_A \|A\|)^{4a} \cdot \left\|Q^T B^{-T}\right\| \\ &= n \cdot (\chi_A \|A\|)^{4a} \cdot \left\|B^{-T}\right\| \\ &\leq n \cdot \chi_A \cdot (\chi_A \|A\|)^{4a}, \end{aligned}$$

as claimed. Notice that the third line follows from the definition of C_1 and (2.11). The fourth line follows from the definition of L, and the last line uses (2.6). □

Next, we show that the computed $\hat{R}$ in Step 1 is close to the true R using a forward error analysis.

Theorem 2.3.3 *Let $\mathbf{r}_j^T$ and $\hat{\mathbf{r}}_j^T$ be the j^{th} rows of R and $\hat{R}$, respectively. Then*

$$\frac{\left\|\mathbf{r}_j^T - \hat{\mathbf{r}}_j^T\right\|}{\left\|\mathbf{r}_j^T\right\|} \leq c\epsilon \cdot \chi_A \left\|A\right\| \cdot \left\{ \sum_{i=j}^{m} \left[1 + (\chi_A \left\|A\right\|)^6\right] \right\}^{1/2} + O(\epsilon^2) \qquad (2.15)$$

where c is a small constant.

Remark. For this proof we assume that Householder transformations are used for the QR factorization in Step 1, although Modified Gram-Schmidt or Givens rotations could also be used. In addition, we assume that the diagonal entries of R have the same signs as corresponding entries of $\hat{R}$. (Recall that QR factorization is uniquely determined only if an assumption is made about the signs of the diagonal entries of R.)

Proof: Let B be the $n \times n$ matrix of basis columns of A^T and consider first the factorization $B = QR_0$. Note that this Q is the same as Q in Step 1, and that R_0 denotes the first n columns of $RD^{-1/2}$. Let $\hat{Q}$ and $\hat{R}_0$ denote the computed versions of these matrices. It follows from standard backward error analysis (see Theorem 18.4 of [21]) that there exists an exactly orthogonal matrix $\tilde{Q}$ such that $\left\|\tilde{Q} - \hat{Q}\right\| \leq c\epsilon$ and such that $\tilde{Q}\hat{R}_0 = QR_0 + E$, where $\left\|E\right\| \leq c\epsilon \left\|B\right\|$, where c is a small constant. We must change this to a forward error bound on the computed factors. If we multiply on the left by Q^T and the right by R_0^{-1}, we obtain $Q^T\tilde{Q}\hat{R}_0R_0^{-1} = I + Q^TER_0^{-1}$. Let $E' = Q^TER_0^{-1}$. By multiplying the preceding equation by its transpose, we obtain

$$(\hat{R}_0R_0^{-1})^T(\hat{R}_0R_0^{-1}) = (I + E')^T(I + E') = I + E' + (E')^T + (E')^TE'.$$

Notice that the left-hand side is a Cholesky factorization, i.e., $\hat{R}_0 R_0^{-1}$ is an upper triangular matrix with positive diagonal entries. The Cholesky factorization is uniquely determined if the signs of diagonal entries are positive.

The right-hand side can be written $I + E' + (E')^T + O(\epsilon^2)$. Write $E' + (E')^T = U^T + D + U$ where U is strictly upper triangular and D is diagonal. Then we observe that $(I + D/2 + U)^T(I + D/2 + U) = I + E' + (E')^T + O(\epsilon^2)$ and that $I + D/2 + U$ is upper triangular. Thus, by uniqueness of the Cholesky factorization, we have $\hat{R}_0 R_0^{-1} = I + D/2 + U + O(\epsilon^2)$, i.e.,

$$\hat{R}_0 = (I + D/2 + U)R_0 + O(\epsilon^2)$$

i.e.,

$$\hat{R}_0 - R_0 = (D/2 + U)R_0 + O(\epsilon^2).$$

Recall that $D/2 + U$ is part of $E' + (E')^T$, so

$$\|D/2 + U\| \leq 2\|E'\| \leq 2\|E\| \cdot \left\|R_0^{-1}\right\|.$$

Recall that $\|E\|$ is bounded by $c\epsilon\|B\|$ and $\left\|R_0^{-1}\right\| = \|B^{-1}\|$, $\|R_0\| = \|B\|$. Thus,

$$\left\|\hat{R}_0 - R_0\right\| \leq c\epsilon \cdot \|B\| \cdot \kappa(B) + O(\epsilon^2)$$

(with a different c). Recall from (2.6) that $\kappa(B) \leq \chi_A \|A\|$, hence

$$\left\|\hat{R}_0 - R_0\right\| \leq c\epsilon\|A\| \cdot (\chi_A \|A\|) + O(\epsilon^2).$$

A similar analysis, starting from the fact that $Q^T\tilde{Q} = \hat{R}_0 R_0^{-1} + E' = I + D/2 + U + E' + O(\epsilon^2)$, gives a bound on $\left\|\hat{Q} - Q\right\|$ of $c\epsilon\chi_A\|A\|$. We will stop writing $O(\epsilon^2)$ for now.

Next we consider the actual QR factorization of $A^T D^{1/2}$ in Step 1 of the COD algorithm. Since we have now proved that $\hat{Q}$ is close to Q, we conclude that $\hat{R}$ which is $\hat{Q} A^T D^{1/2}$, is close to the true R on a column-by-column basis. In other words, let us define $\mathbf{v}_i$ to be the ith column of R and $\hat{\mathbf{v}}_i$ the corresponding column of $\hat{R}$; then

$$\|\mathbf{v}_i - \hat{\mathbf{v}}_i\| \leq c \cdot \epsilon \cdot \chi_A \|A\| \cdot \|\mathbf{v}_i\|, 1 \leq i \leq m.$$

This gives a bound on the elementwise error, namely

$$|r_{ji} - \hat{r}_{ji}| \leq \|\mathbf{v}_i - \hat{\mathbf{v}}_i\| \leq c\epsilon\chi_A \|A\| \cdot \|\mathbf{v}_i\|, 1 \leq i \leq m, 1 \leq j \leq n.$$

Notice that if the ith row $\mathbf{a}_i^T$ of A is not linearly independent of the first $j-1$ basis rows, then $r_{ji} = 0$ if $i \geq j$. Because of the dependency test in the COD algorithm, $\hat{r}_{ji} = 0$ also. This point will be important later in the proof. We can now find a bound on the normwise error of the rows of R. Let $\mathbf{r}_j^T$ and $\hat{\mathbf{r}}_j^T$ be the jth rows of R and the computed matrix $\hat{R}$. Then

$$\frac{\left\|\mathbf{r}_j^T - \hat{\mathbf{r}}_j^T\right\|^2}{\left\|\mathbf{r}_j^T\right\|^2} \leq \frac{\sum_{i=j}^{m} (r_{ji} - \hat{r}_{ji})^2}{\sum_{i=j} r_{ji}^2} \leq \frac{c^2 \epsilon^2 (\chi_A \|A\|)^2 \cdot \sum_{i=j}^{m} \|\mathbf{v}_i\|^2}{r_{jj}^2}.$$

Based on the argument above, there will be no contribution (in the sum) from columns $\mathbf{v}_i$ if $\mathbf{a}_i^T$ is dependent only on the first $j-1$ basis rows. Now consider all other $\frac{\|\mathbf{v}_i\|^2}{r_{jj}^2}$. Suppose that $j-1, j \geq 1$ steps of the QR factorization have been completed. Let X denote $A^T D^{-1/2}$ and $\mathbf{x}_1, \ldots, \mathbf{x}_m$ the columns of X. Then

$$Q_{j-1}^T \cdots Q_1^T A^T D^{-1/2} = \begin{bmatrix} r_{11} & \cdots & r_{1,j-1} & r_{1j} & \cdots & r_{1m} \\ 0 & & \vdots & \vdots & & \vdots \\ \vdots & & r_{j-1,j-1} & r_{j-1,j} & \cdots & r_{j-1,m} \\ \mathbf{0} & \cdots & \mathbf{0} & \bar{\mathbf{x}}_j & \cdots & \bar{\mathbf{x}}_m \end{bmatrix}.$$

Then $r_{jj}^2 = \|\bar{\mathbf{x}}_j\|^2$. Since no pivoting is necessary, the columns of A are ordered such that $\|\bar{\mathbf{x}}_i\|^2 \leq \|\bar{\mathbf{x}}_j\|^2$ for $i \geq j$. Recall that $\bar{R} = Q^T A^T = RD^{1/2}$. So for $i \geq j$,

$$\begin{aligned}
\frac{\|\mathbf{v}_i\|^2}{r_{jj}^2} &= \frac{\|\bar{\mathbf{x}}_i\|^2 + r_{1i}^2 + \cdots + r_{j-1,i}^2}{r_{jj}^2} \\
&\leq 1 + \frac{r_{1i}^2 + \cdots + r_{j-1,i}^2}{r_{jj}^2} \\
&= 1 + \frac{d_i\left(\bar{r}_{1i}^2 + \cdots + \bar{r}_{j-1,i}^2\right)}{d_j \bar{r}_{jj}^2} \\
&\leq 1 + \frac{d_i \|\bar{\mathbf{r}}_i\|^2}{d_j \bar{r}_{jj}^2} \\
&\leq 1 + \chi_A^2 \cdot \|\mathbf{a}_i\|^2 \cdot (\chi_A \|A\|)^4 \\
&\leq 1 + (\chi_A \|A\|)^6 ,
\end{aligned}$$

where $\bar{\mathbf{r}}_i$ in the fourth line is the ith column of $\bar{R}$. The fifth line follows from (2.8), (2.9) and the lower bound of (2.11) and holds only when $\mathbf{a}_i$ is linearly independent of $\mathbf{a}_1, \ldots, \mathbf{a}_{j-1}$. Recall that we need not be concerned with the other case since, as discussed above, there is no contribution from such terms. Thus,

$$\frac{\left\|\mathbf{r}_j^T - \hat{\mathbf{r}}_j^T\right\|^2}{\left\|\mathbf{r}_j^T\right\|^2} \leq c^2\epsilon^2(\chi_A \|A\|)^2 \cdot \left(\sum_{i=j}^{m}\left[1 + (\chi_A \|A\|)^6\right]\right)^2 .$$

Taking the square root of both sides gives the required result. □

We have now established that the QR factorization in the first step of the algorithm gives an upper triangular matrix R that is well conditioned up to a scaling of the rows or the columns. It also yields a computed matrix $\hat{R}$ whose rows are close to those of R. With these results in hand, we move on to the analysis of the second step of the algorithm.

2.3.2 The Second QR Factorization

Recall that in this step we use the "skinny" QR factorization,

$$R^T = Z_1 U_1,$$

where Z_1 is an $m \times n$ matrix with orthonormal columns and U_1 is an $n \times n$ upper triangular matrix. The results of §2.3.1 imply that R^T is well conditioned up to a scaling of the columns. Thus, the QR factorization in this step gives an upper triangular matrix $\hat{U}_1$ that is close to the exact upper triangular matrix U_1. In addition, the results (concerning the condition number of R) of §2.3.1 can be used to prove similar results about the condition number of U_1. Again, let $D = \mathrm{diag}(d_1, \ldots, d_m)$ and $\bar{D} = D(1:n, 1:n)$. Since U_1 is the coefficient matrix of the system of equations in the back substitution step, the following theorem concerning the condition of U_1 will be quite useful.

Theorem 2.3.4 *Let U_1 and $\bar{D}$ be defined as above. Then*

$$\kappa\left(\bar{D}^a U_1 \bar{D}^{-1/2-a}\right) \leq n^{10} \cdot \left(\chi_A \|A\|\right)^{4a+26} \tag{2.16}$$

for any $-\frac{1}{2} \leq a \leq \frac{1}{2}$.

Proof: We separately bound $\left\|\bar{D}^a U_1 \bar{D}^{-1/2-a}\right\|$ and $\left\|\bar{D}^{a+1/2} U_1^{-1} \bar{D}^{-a}\right\|$, starting with the second norm. Notice that

$$R^T U_1^{-1} = Z_1.$$

Let $\mathbf{v}_k = U_1^{-1}(1:k,k)$, $\tilde{R} = R(1:k,:)$, and $\tilde{D} = D(1:k,1:k)$. It follows from the fact that $U_1^{-T} R R^T U_1^{-1} = I$ that

$$\frac{1}{u_{kk}} \tilde{R}\tilde{R}^T \mathbf{v}_k = \mathbf{e}_k,$$

where $\mathbf{e}_k$ is the kth column of the $k \times k$ identity matrix. So,

$$\begin{aligned}
\mathbf{v}_k &= u_{kk} \left(\tilde{R}\tilde{R}^T\right)^{-1} \mathbf{e}_k \\
&= \left(\mathbf{z}_k^T \mathbf{r}_k\right) \tilde{D}^{-1/2} \left(\tilde{D}^{-1/2}\tilde{R}\tilde{R}^T\tilde{D}^{-1/2}\right)^{-1} \tilde{D}^{-1/2}\mathbf{e}_k \\
&= \left(\mathbf{z}_k^T \mathbf{r}_k\right) d_k^{-1/2} \tilde{D}^{-1/2}\mathbf{x},
\end{aligned}$$

where $\mathbf{z}_k$ is the kth column of Z_1, $\mathbf{r}_k$ is the kth column of R^T and $\mathbf{x}$ is the last column of $\left(\tilde{D}^{-1/2}\tilde{R}\tilde{R}^T\tilde{D}^{-1/2}\right)^{-1}$. Multiplying both sides by $d_k^{-a}\tilde{D}^{a+1/2}$ yields

$$d_k^{-a}\tilde{D}^{a+1/2}\mathbf{v}_k = \left(\mathbf{z}_k^T \mathbf{r}_k\right) d_k^{-1/2-a}\tilde{D}^a\mathbf{x}.$$

We show that there is an upper bound on the right-hand side as follows:

$$\begin{aligned}
\left\| \left(\mathbf{z}_k^T \mathbf{r}_k\right) d_k^{-1/2-a}\tilde{D}^a\mathbf{x} \right\| &= d_k^{-1/2} \cdot \left|\mathbf{z}_k^T \mathbf{r}_k\right| \cdot \left\| d_k^{-a}\tilde{D}^a\mathbf{x} \right\| \\
&\leq d_k^{-1/2} \cdot \|\mathbf{r}_k\| \cdot \left\| \tilde{D}^a \left(\tilde{D}^{-1/2}\tilde{R}\tilde{R}^T\tilde{D}^{-1/2}\right)^{-1} \tilde{D}^{-a} \right\| \\
&\leq \|A\| \cdot (\chi_A \|A\|)^2 \cdot \left\| \left(\tilde{D}^{-1/2+a}\tilde{R}\tilde{R}^T\tilde{D}^{-1/2-a}\right)^{-1} \right\| \\
&= \|A\| \cdot (\chi_A \|A\|)^2 \cdot \frac{\kappa\left(\tilde{D}^{-1/2+a}\tilde{R}\tilde{R}^T\tilde{D}^{-1/2-a}\right)}{\left\|\tilde{D}^{-1/2+a}\tilde{R}\tilde{R}^T\tilde{D}^{-1/2-a}\right\|} \\
&\leq \|A\| \cdot (\chi_A \|A\|)^2 \cdot \frac{\kappa\left(\tilde{D}^{-1/2+a}\tilde{R}D^{-a}D^a\tilde{R}^T\tilde{D}^{-1/2-a}\right)}{\bar{r}_{11}^2} \\
&\leq \chi_A \cdot (\chi_A \|A\|)^3 \cdot \kappa\left(\tilde{D}^{-1/2+a}\tilde{R}D^{-a}\right) \cdot \kappa\left(D^a\tilde{R}^T\tilde{D}^{-1/2-a}\right).
\end{aligned}$$

The third line is an application of (2.11) with $a = -1/2$. In the fifth line of the above inequality, $\bar{r}_{11}$ is the (1,1) entry of $\bar{R} = Q^T A^T$. Notice that if $-\frac{1}{2} \leq a \leq \frac{1}{2}$, then Theorem 2.3.2 applies. So,

$$\begin{aligned}\left\| d_k^{-a} \tilde{D}^{-1/2-a} \mathbf{v}_k \right\| &\leq \chi_A \cdot (\chi_A \|A\|)^3 \cdot \kappa\left(\tilde{D}^{-1/2+a} \tilde{R} D^{-a}\right) \cdot \kappa\left(D^a \tilde{R}^T \tilde{D}^{-1/2-a}\right) \\ &\leq n^8 \cdot \chi_A \cdot (\chi_A \|A\|)^{23}.\end{aligned}$$

Now,

$$\begin{aligned}\left\| \bar{D}^{a+1/2} U_1^{-1} \bar{D}^{-a} \right\| &\leq \sum_{i=1}^{n} \left\| d_i^{-a} \bar{D}^{a+1/2} \mathbf{v}_i \right\| \\ &\leq n \cdot \max_{1 \leq i \leq n} \left\| d_i^{-a} \tilde{D}^{a+1/2} \mathbf{v}_i \right\| \\ &\leq n^9 \cdot \chi_A \cdot (\chi_A \|A\|)^{23}, \end{aligned} \tag{2.17}$$

for $-\frac{1}{2} \leq a \leq \frac{1}{2}$. In order to find an upper bound on $\left\| \bar{D}^a U_1 \bar{D}^{-1/2-a} \right\|$, we proceed as follows:

$$\begin{aligned}\left\| \bar{D}^a U_1 \bar{D}^{-1/2-a} \right\| &\leq \sum_{i=1}^{n} \left\| d_i^{-1/2-a} \bar{D}^a \mathbf{u}_i \right\| \\ &\leq n \cdot \max_{1 \leq i \leq n} \left\| d_i^{-1/2-a} \bar{D}^a \mathbf{u}_i \right\| \\ &\leq n \cdot (\chi_A \|A\|)^{4a} \cdot \max_{1 \leq i \leq n} \left\| d_i^{-1/2} Z_1^T \mathbf{r}_i \right\| \\ &\leq n \cdot (\chi_A \|A\|)^{4a} \cdot \max_{1 \leq i \leq n} \left\| d_i^{-1/2} \mathbf{r}_i \right\| \\ &= n \cdot (\chi_A \|A\|)^{4a} \cdot \left\| d_i^{-1/2} D^{1/2} \bar{\mathbf{r}}_i \right\| \\ &\leq n \cdot \|A\| \cdot (\chi_A \|A\|)^{4a+2}, \end{aligned}$$

where $\mathbf{u}_i$ is the ith column of U_1, $\mathbf{r}_i$ is the ith column of R^T, and $\bar{\mathbf{r}}_i$ is the ith column of $\bar{R}^T$. The third line of the inequality is derived from the second using (2.9), the fact

that U_1 is upper triangular, and $U_1 = Z_1^T R^T$. The fourth line uses $\|Z_1\| = 1$. The last line is obtained by applying (2.11) with $a = -1/2$. Thus for $-\frac{1}{2} \leq a \leq \frac{1}{2}$,

$$\begin{aligned} \kappa\left(\bar{D}^a U_1 \bar{D}^{-1/2-a}\right) &= \left\|\bar{D}^a U_1 \bar{D}^{-1/2-a}\right\| \cdot \left\|\bar{D}^{a+1/2} U_1^{-1} \bar{D}^{-a}\right\| \\ &\leq n^{10} \cdot \left(\chi_A \|A\|\right)^{4a+26}. \end{aligned}$$

□

Now that we know that $\bar{D}^a U_1 \bar{D}^{-1/2-a}$ is well-conditioned for $-\frac{1}{2} \leq a \leq \frac{1}{2}$, we move on to the analysis of the remainder of the algorithm.

2.3.3 Finding the Solution y

In analyzing the remainder of the algorithm, we first show that the error introduced in the back substitution step is small. In Step 3 of the algorithm, the upper triangular system

$$U_1 \bar{\mathbf{y}} = Z_1^T D^{1/2} \mathbf{b}$$

is solved for $\bar{\mathbf{y}}$. (Note that this is slightly different from the system given in Step 3 of the algorithm as presented in §2.1 since the columns of $A^T D^{1/2}$ have been "pre-pivoted.") Instead of working with the system given above, consider the following system:

$$\bar{D}^{-1/2} U_1 \bar{\mathbf{y}} = \bar{D}^{-1/2} Z_1^T D^{1/2} \mathbf{b},$$

where $D = \mathrm{diag}(d_1, d_2, \ldots, d_m)$ and $\bar{D} = D(1:n, 1:n)$ as before. In working through the steps of back substitution, one can see that solving this system is equivalent to solving the original one, even in floating point arithmetic. (In other words,

a rescaling of the rows does not change the numerical bounds.) Recall from the last section that $\bar{D}^a U_1 \bar{D}^{-1/2-a}$ is well-conditioned for $-\frac{1}{2} \leq a \leq \frac{1}{2}$. Therefore, standard techniques for analyzing back substitution can be used to show that the error at this step is small. The following theorem states that error bound.

Theorem 2.3.5 *Let* $\bar{\mathbf{y}}$ *be the exact solution to* $\bar{D}^{-1/2} U_1 \bar{\mathbf{y}} = \bar{D}^{-1/2} Z_1^T D^{1/2} \mathbf{b}$, *and let* $\check{\mathbf{y}}$ *be the computed solution. Then*

$$\|\bar{\mathbf{y}} - \check{\mathbf{y}}\| \leq \epsilon \cdot n^{29} \cdot \chi_A \cdot (\chi_A \|A\|)^{78} \cdot \|\mathbf{b}\| + O(\epsilon^2). \tag{2.18}$$

Proof: Let $\check{\mathbf{y}}$ be the computed solution to the above system. Then $\check{\mathbf{y}}$ is the exact solution to the nearby system of equations,

$$(\bar{D}^{-1/2} U_1 + E)\check{\mathbf{y}} = \bar{D}^{-1/2} Z_1^T D^{1/2} \mathbf{b}.$$

The matrix E accounts for errors during the back substitution and $|E| \leq \epsilon \cdot \left|\bar{D}^{-1/2} U_1\right|$, where ϵ is machine roundoff [16]. So,

$$\bar{D}^{-1/2} U_1 \bar{\mathbf{y}} - (\bar{D}^{-1/2} U_1 + E)\check{\mathbf{y}} = \mathbf{0},$$

or

$$\bar{\mathbf{y}} - \check{\mathbf{y}} = (\bar{D}^{-1/2} U_1)^{-1} E \check{\mathbf{y}}.$$

Substituting for $\check{\mathbf{y}}$ on the right-hand side yields

$$\bar{\mathbf{y}} - \check{\mathbf{y}} = (D^{-1/2} U)^{-1} E (D^{-1/2} U + E)^{-1} \bar{D}^{-1/2} Z_1^T D^{1/2} \mathbf{b}.$$

Thus,

$$\begin{aligned}
\|\bar{\mathbf{y}} - \check{\mathbf{y}}\| &\leq \left\|(\bar{D}^{-1/2}U_1)^{-1}\right\| \cdot \|E\| \cdot \left\|(\bar{D}^{-1/2}U_1 + E)^{-1}\right\| \cdot \left\|\bar{D}^{-1/2}Z_1^T D^{1/2}\right\| \cdot \|\mathbf{b}\| \\
&\leq \epsilon \cdot \left\|(\bar{D}^{-1/2}U_1)^{-1}\right\| \cdot \left\|\bar{D}^{-1/2}U_1\right\| \cdot \left\|(\bar{D}^{-1/2}U_1 + E)^{-1}\right\| \cdot \\
&\quad \left\|\bar{D}^{-1/2}U_1^{-T}RD^{1/2}\right\| \cdot \|\mathbf{b}\| \\
&\leq \epsilon \cdot \left\|(\bar{D}^{-1/2}U_1)^{-1}\right\| \cdot \left\|\bar{D}^{-1/2}U_1\right\| \cdot (\|I\| + \left\|(\bar{D}^{-1/2}U_1)^{-1}E\right\| \\
&\quad + \left\|(\bar{D}^{-1/2}U_1)^{-1}E\right\|^2 + \left\|(\bar{D}^{-1/2}U_1)^{-1}E\right\|^3 + \cdots) \cdot \\
&\quad \left\|(\bar{D}^{-1/2}U_1)^{-1}\right\| \cdot \left\|\bar{D}^{-1/2}U_1^{-T}\bar{D}\bar{D}^{-1}RD^{1/2}\right\| \cdot \|\mathbf{b}\| \\
&\leq \epsilon \cdot \kappa(\bar{D}^{-1/2}U_1) \cdot \left\|(\bar{D}^{-1/2}U_1)^{-1}\right\| \cdot \left\|\bar{D}^{-1/2}U_1^{-T}\bar{D}\right\| \cdot \left\|\bar{D}^{-1}RD^{1/2}\right\| \cdot \|\mathbf{b}\| \\
&\quad + O(\epsilon^2) \\
&\leq \epsilon \cdot n^{29} \cdot \chi_A \cdot (\chi_A \|A\|)^{78} \cdot \|\mathbf{b}\| + \mathrm{O}(\epsilon^2),
\end{aligned}$$

as claimed. The last line is obtained by applying (2.16), (2.17), and (2.13) with the appropriate values of a. □

In the theorem above, the errors in the computation of U_1 itself (which also contribute to the error in $\bar{\mathbf{y}}$) are not included, but could be accounted for as a somewhat larger perturbation matrix E. As we we have already argued in the proof of Theorem 2.3.3, the errors in computing the factors are small. A similar analysis could be applied to the second factorization, showing that errors made in forming each row of U_1 are small with respect to the norm of that row. Therefore, the perturbation matrix E is small with respect to $\bar{D}^{-1/2}U_1$. Explicitly including this analysis in the previous theorem would make the proof more complicated, but the bound would be qualitatively the same.

The final step is to obtain $\mathbf{y}$ by multiplying $\bar{\mathbf{y}}$ by Q. Let $\hat{\mathbf{y}}$ be the computed result. Assume that $\hat{\mathbf{y}}$ accounts for the errors during both this step and the previous step. Then (2.13), (2.17), and (2.18) are used to obtain the following error bound:

$$\begin{aligned}
\|\mathbf{y} - \hat{\mathbf{y}}\| &\leq \epsilon \cdot n \cdot \|\mathbf{y}\| + \|\bar{\mathbf{y}} - \check{\mathbf{y}}\| \\
&\leq \epsilon \cdot n \cdot \left\| \left(\bar{D}^{-1/2} U_1\right)^{-1} \right\| \cdot \left\| \bar{D} U_1^{-1} \bar{D}^{-1/2} \right\| \cdot \left\| \bar{D}^{-1} R D^{1/2} \right\| \cdot \|\mathbf{b}\| + \|\bar{\mathbf{y}} - \check{\mathbf{y}}\| \\
&\leq \epsilon \cdot \left[n^{19} \cdot \chi_A \cdot (\chi_A \|A\|)^{51} + n^{29} \cdot \chi_A \cdot (\chi_A \|A\|)^{78} \right] \cdot \|\mathbf{b}\| + \mathrm{O}(\epsilon^2). \quad (2.19)
\end{aligned}$$

Notice that the error bound is of the form

$$\|\mathbf{y} - \hat{\mathbf{y}}\| \leq \epsilon \cdot f(A) \cdot \|\mathbf{b}\| .$$

Thus, the complete orthogonal decomposition algorithm satisfies the stability bound (1.2).

2.4 Related Work

As discussed in §1.2, weighted least-squares problems arise in a number of applications. Consequently, there are numerous algorithms in the existing literature that are specialized for weighted least squares. In this section, we give a brief overview of backward error analysis, which is used in the stability analyses of these algorithms, and describe the relationship to the forward error analysis of this chapter. In doing so, we show that there are difficulties when trying to obtain stability bound (1.2) in such a setting, and we explain how the COD algorithm and its analysis avoid these problems. We also take a closer look at a couple of algorithms for which a forward analysis has been done or for which proving a forward bound appears possible.

As just mentioned, there are many algorithms that appear in the literature. Such algorithms are presented in Barlow [1], Björck and Duff [5], Golub [15], Gulliksson [19], Gulliksson and Wedin [20], Paige [31], Peters and Wilkinson [32], Powell and Reid [33], Van Loan [39], and Vavasis [41]. These algorithms are based on standard algorithms for solving unweighted least-squares problems, and special techniques are employed to exploit structure and to deal with widely varying weights. None of these works, except [41], proves a forward stability bound for their algorithms. Many of these papers were published before Theorem 1.1.1 appeared, so the absence of forward error bound is not surprising. Recall that a forward error bound has relevance for the applications described in §1.2.

Since Vavasis's NSH method [41] is the only other algorithm that proves a forward stability bound, we discuss it first. The NSH algorithm employs nonstandard techniques, particularly when choosing the nullspace basis for $A^T D$. In contrast, the COD algorithm of this chapter uses standard techniques that are well understood, namely QR decomposition and back substitution. Also, our algorithm is more efficient than the NSH algorithm. The NSH method solves an $m \times m$ system of equations and thus requires $O(m^3)$ flops. The work for the QR factorizations dominates the work required for the complete orthogonal algorithm, so this algorithm requires $O(mn^2)$ flops. Since $n < m$ (and n could be much smaller than m), the COD algorithm requires less work.

In considering the other algorithms, the question arises whether a forward error bound can be derived from the error analyses given by some of the authors mentioned above. Several, e.g. [1], [5], [19], [31], and [33], prove a backward stability bound, i.e.,

they prove that their algorithms compute a solution $\hat{\mathbf{y}}$ that solves a nearby problem given by

$$\min_{\hat{\mathbf{y}} \in \mathbb{R}^n} \left\| (D + \Delta D)^{1/2} \left[(A + \Delta A)\, \hat{\mathbf{y}} - (\mathbf{b} + \Delta \mathbf{b}) \right] \right\|,$$

(or something similar) where $\|\Delta D\| \leq \epsilon \cdot c_1 \cdot \|D\|$, $\|\Delta A\| \leq \epsilon \cdot c_2 \cdot \|A\|$, and $\|\Delta \mathbf{b}\| \leq \epsilon \cdot c_3 \cdot \|\mathbf{b}\|$. Here c_1, c_2, and c_3 are small constants, and ϵ is machine precision. It may seem at first that a forward error bound can be obtained by applying Theorem 1.1.1 to the backward error bounds. Since all of the backward error analyses of previous authors introduce a perturbation ΔA into the coefficient matrix A, the forward error bound will involve $\chi_{A+\Delta A}$. The result is a forward bound of the form

$$\|\mathbf{y} - \hat{\mathbf{y}}\| \leq \epsilon \cdot c \cdot \chi_{A+\Delta A} \cdot \|\mathbf{b}\|, \tag{2.20}$$

where c is a constant for these other algorithms. Here a difficulty arises: χ_A is not continuous with respect to perturbations of A, as observed by [36]. In fact, there exists a matrix A such that $\chi_A < 3$ but for any $\epsilon > 0$

$$\sup\{\chi_{A+\Delta A} : \|\Delta A\| \leq \epsilon\} = \infty.$$

In particular, the example is

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Note that $\lim_{n\to\infty} \chi_{A_n} = \infty$ if we define

$$A_n = \begin{bmatrix} 1 + 1/n & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Thus, the supremum of the right-hand side of (2.20) over all arbitrarily small perturbations of A is infinity for this particular A. Notice that the matrix A has the special property that its first two rows are parallel. Indeed, it is a consequence of [36] and [30] that χ_A is discontinuous at A whenever A has an $n \times n$ singular submatrix. (This fact also follows from (2.6).) A "randomly chosen" matrix would never have an $n \times n$ singular submatrix, but singular submatrices occur often in practice (e.g. consider a node-arc adjacency matrix or a linear programming problem that includes two simple-bound constraints of the form $y_i \geq c_1$ and $y_i \leq c_2$). Thus, (1.2) is not established for such an A, i.e., it cannot be established in general using the kind of backward error analysis in the existing literature.

The most difficult question with respect to previous algorithms is the following: Can a forward error bound be derived for an algorithm in the previous literature by using a completely new analysis? We claim that any algorithm purporting to satisfy a forward error bound like (1.2) must have an explicit test for singularity. To illustrate this, we consider the impact of a singular submatrix upon algorithms for weighted least squares (WLS). Consider the following two WLS problems:

$$\min \left\| \mathrm{diag}(10^{30}, 10^{30}, 1) \left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \mathbf{y} - \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \right) \right\| \tag{2.21}$$

and

$$\min \left\| \mathrm{diag}(10^{30}, 10^{30}, 1) \left(\begin{bmatrix} 1+10^{-15} & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \mathbf{y} - \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \right) \right\|. \tag{2.22}$$

Observe that although the numerical data in (2.21) is very close to (2.22), the two WLS problems have sharply different solutions. In (2.21), the first two rows are parallel and are minimized (in the absence of the third row) by any vector on the line $y_1 + y_2 = 1.5$. The third row is minimized by any vector with $y_2 = 3$. Thus, the solution to the problem is $(-1.5, 3)$.

In contrast, for (2.22) the first two rows are not parallel. Because of their huge weights, both must be satisfied nearly as equations at the WLS solution, i.e., the solution to the second problem will be very close to

$$\begin{bmatrix} 1 + 10^{-15} & 1 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 2 \end{bmatrix},$$

which will obviously be very large since the matrix being inverted is nearly singular.

Thus, any algorithm purporting to satisfy a forward error bound like (1.2) for the example problem (2.21) must preserve the dependence between the first two rows of A even if there is roundoff error. In fact, this is exactly how COD operates. The COD algorithm, when it encounters rows of A that are nearly dependent, will perturb them so that they become exactly dependent. This perturbation is the topic of §2.2. Ordinarily, it is considered undesirable for a numerical algorithm to make a yes/no decision about linear dependence. It is apparent from the contrast between (2.21) and (2.22), however, that any algorithm that is supposed to satisfy stability bound (1.2) must "know" that the first two rows of A in (2.21) are parallel.

Thus, we conclude that forward-stable algorithms, that is algorithms satisfying (1.2), must include a test for linear dependence among the rows with heaviest weights. Most algorithms in the previous literature do not. One exception is Björck and Duff's

modification of Peters and Wilkinson's algorithm. This method is an elimination-based decomposition of the normal equations. A special type of pivoting is used to preserve sparsity. We suspect that a modification of this pivoting and an appropriate test for linear dependence among the rows of A will yield an algorithm for which a forward stability bound can be established. The only approach we know of to establish this forward error bound would be an analysis similar to the preceding analysis of COD.

Additionally, many interior-point implementations of Cholesky factorization effectively perform a test for linear dependence, e.g. [47]. If a Cholesky pivot is very small, then the algorithm treats that column differently. Although a singularity test is present, we do not expect that a method based on normal equations could be stable for weighted least squares. Too much information about rows of A corresponding to small entries of D is lost when forming $A^T DA$ (e.g. consider what happens to the last row of A in the normal equations of (2.21)). Nevertheless, interior point practitioners report successes with this approach.

Chapter 3

Weighted Least Squares in Interior Point Methods

3.1 The Application

As mentioned in Chapter 1, the weighted least-squares problem (1.1) arises in a number of applications. This chapter contains a discussion of the problem in the context of one of those applications, namely interior point methods for linear programming. Thus, we consider solving the following primal-dual pair of linear programming problems, given in standard form:

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned} \tag{3.1}$$

and

$$\begin{array}{ll} \text{maximize} & \mathbf{b}^T\mathbf{y} \\ \text{subject to} & A^T\mathbf{y} + \mathbf{s} = \mathbf{c}, \\ & \mathbf{s} \geq \mathbf{0}. \end{array} \tag{3.2}$$

In this setting, $A \in \mathbb{R}^{m\times n}$ is known and has rank m, $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^n$ are given, and $\mathbf{x}, \mathbf{s} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ are unknown. The focus of this chapter is on the linear algebra computations that arise in primal-dual path-following interior point algorithms. For an overview of such methods, see Gonzaga [17], M. Wright [44], or S. Wright [45].

The optimality conditions for the linear programming problems are given by:

$$\begin{aligned} A\mathbf{x} &= \mathbf{b}, \\ A^T\mathbf{y} + \mathbf{s} &= \mathbf{c}, \\ SX\mathbf{e} &= \mathbf{0} \\ \mathbf{x} \geq \mathbf{0} \quad &, \quad \mathbf{s} \geq \mathbf{0}, \end{aligned}$$

where $S = \text{diag}(\mathbf{s})$, $X = \text{diag}(\mathbf{x})$, and $\mathbf{e}$ is the vector of all ones. Introducing a parameter $\mu > 0$ into the third equation perturbs the optimality conditions to the following:

$$\begin{aligned} A\mathbf{x} &= \mathbf{b}, \\ A^T\mathbf{y} + \mathbf{s} &= \mathbf{c}, \\ SX\mathbf{e} &= \mu\mathbf{e}, \\ \mathbf{x} > \mathbf{0} \quad &, \quad \mathbf{s} > \mathbf{0}. \end{aligned} \tag{3.3}$$

If both (3.1) and (3.2) have strictly interior points, then this system of equations has a unique solution for every $\mu > 0$. The set of all solutions as μ ranges from 0 to ∞ is the *central path* of the linear programming problem. The central path was first defined by Bayer and Lagarias [2], [3], [4], Megiddo [23], and Sonnevend [35]. Note that $\mu = \infty$ corresponds to the analytic center of the feasible region, and $\mu = 0$ corresponds to the optimal solution. In path-following algorithms, the goal is to approach the optimal solution by following the central path. This is accomplished by solving (3.3) with a sequence of values for μ that tend to zero.

Given $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ that approximates a point on the central path, it is necessary to solve (3.3) with a smaller value of μ for the next approximate central path point $(\mathbf{x} + \Delta\mathbf{x}, \mathbf{y} + \Delta\mathbf{y}, \mathbf{s} + \Delta\mathbf{s})$, i.e., solve

$$\begin{aligned} A(\mathbf{x} + \Delta\mathbf{x}) &= \mathbf{b}, \\ A^T(\mathbf{y} + \Delta\mathbf{y}) + (\mathbf{s} + \Delta\mathbf{s}) &= \mathbf{c}, \\ (S + \Delta S)(X + \Delta X)\mathbf{e} &= \bar{\mu}\mathbf{e}, \\ \mathbf{x} + \Delta\mathbf{x} > \mathbf{0} \quad , \quad \mathbf{s} + \Delta\mathbf{s} &> \mathbf{0}, \end{aligned}$$

where $\bar{\mu} < \mu$. Solving for the unknowns $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$, applying (3.3), and linearizing the third condition yield the Newton equations:

$$A\Delta\mathbf{x} = \mathbf{0}, \tag{3.4a}$$

$$A^T\Delta\mathbf{y} + \Delta\mathbf{s} = \mathbf{0}, \text{ and} \tag{3.4b}$$

$$X\Delta\mathbf{s} + S\Delta\mathbf{x} = \bar{\mu}\mathbf{e} - XS\mathbf{e}. \tag{3.4c}$$

Notice that the solution $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$ is uniquely determined by this system of equations. The remainder of this chapter focuses on the stable computation of the solution

to this system of equations. The next section contains the derivation of the weighted least-squares problem that must be solved to compute $\Delta\mathbf{y}$. This turns out to be a problem of the type described in Chapter 1, and it can, therefore, be solved using the COD algorithm of Chapter 2. A discussion of numerical problems that may arise in the computations of $\Delta\mathbf{x}$ and $\Delta\mathbf{s}$ appears in §3.3. We propose computing scaled versions of these directions using one of the orthogonal factors computed by the COD algorithm. Numerical experiments in which the extended COD algorithm was incorporated into two different interior point methods were done to test its performance in practice. The results of these experiments appear in §3.4.

We now move on to the discussion of how to obtain the search directions.

3.2 Computing $\Delta\mathbf{y}$

The dual step $\Delta\mathbf{y}$ can be obtained by computing the solution to a weighted least-squares problem similar to the one described in Chapter 1. The derivation follows. First, the other variables must be eliminated from the system of equations. Solving (3.4c) for $\Delta\mathbf{s}$ and substituting into (3.4b) yields

$$A^T\Delta\mathbf{y} = -\bar{\mu}X^{-1}\mathbf{e} + \mathbf{s} + X^{-1}S\Delta\mathbf{x}.$$

Multiplying both sides of the previous equation by $AS^{-1}X$ and applying (3.4a) gives

$$AS^{-1}XA^T\Delta\mathbf{y} = AS^{-1}X(\mathbf{s} - \bar{\mu}X^{-1}\mathbf{e}).$$

Since $S^{-1}X$ is a diagonal matrix, we write

$$ADA^T\Delta\mathbf{y} = AD(\mathbf{s} - \bar{\mu}X^{-1}\mathbf{e}), \tag{3.5}$$

where $D = S^{-1}X$. Notice that these are the normal equations for the weighted least-squares problem given by

$$\min_{\Delta \mathbf{y} \in \mathbb{R}^m} \left\| D^{1/2}(A^T \Delta \mathbf{y} - \mathbf{v}) \right\|, \tag{3.6}$$

where $\mathbf{v} = \mathbf{s} - \bar{\mu} X^{-1}\mathbf{e}$. Notice also that the matrix D is positive definite since $(\mathbf{x}, \mathbf{s}) > \mathbf{0}$. A final point to note is that some entries of $\mathbf{s}$ and some entries of $\mathbf{x}$ approach zero near the boundary of the feasible region, so D becomes extremely ill conditioned near the boundary. So (3.6) is exactly the problem defined in Chapter 1, except that A has been transposed to be consistent with standard linear programming notation. Thus, the COD Algorithm can be used to compute $\Delta\mathbf{y}$ accurately with respect to $\mathbf{s}$. (The other term $\bar{\mu} X^{-1}\mathbf{e}$ in (3.6) is on the same order as $\mathbf{s}$ because of proximity to the central path).

Since Chapter 2 contains a detailed discussion of the stability of the algorithm and other numerical issues that arise, we forego any further discussion of the computation of $\Delta\mathbf{y}$. Instead, we shift the focus to determining $\Delta\mathbf{x}$ and $\Delta\mathbf{s}$. The following section contains a discussion of numerical issues that arise and a description of how the COD algorithm can be extended to stably compute these directions.

3.3 Computing $\Delta\mathbf{x}$ and $\Delta\mathbf{s}$

Once $\Delta\mathbf{y}$ has been obtained, it is apparent from the central path equations that computing $\Delta\mathbf{x}$ and $\Delta\mathbf{s}$ is straightforward. Solving (3.4b) and (3.4c) yields the following:

$$\Delta\mathbf{s} = A^T \Delta\mathbf{y}, \text{ and} \tag{3.7a}$$

$$\Delta\mathbf{x} = \bar{\mu} S^{-1}\mathbf{e} - S^{-1} X \Delta\mathbf{s}. \tag{3.7b}$$

However, this approach is not numerically stable. The forward error bound (1.2) is not sufficiently strong to get the requisite accuracy bound on $\Delta\mathbf{x}$ or $\Delta\mathbf{s}$, because some components are very small as convergence is achieved [46], or more generally, as the boundary of the feasible region is approached. This means that there is a demand for more accuracy in some components of $A^T\Delta\mathbf{y}$ than what could be obtained from (1.2). To clarify this, let us consider the following two-variable example with box constraints, given in dual form:

$$\begin{aligned} \text{maximize} \quad & 100y_1 - y_2 \\ \text{subject to} \quad & l \leq y_1, y_2 \leq u \end{aligned}$$

Since there is significantly more emphasis on y_1, and interior point method will first maximize with respect to y_1, then with respect to y_2. The feasible region and the central path are shown in Figure 3.1. Notice that near the boundary (e.g. at point P on the central path), the distance s_1 to the boundary is small. As a result, Δs_1 will also be small (since it cannot exceed the magnitude of s_1 and remain feasible). So, $\Delta\mathbf{s}$ must be computed in such a way that there is a small relative error in Δs_1. In other words, the error in Δs_1 should be small relative to s_1. However, the step from point P has a very small change in the y_1 direction and a large change in the y_2 direction. Obtaining $\Delta\mathbf{s}$ via (3.7a) implies that the contributions to Δs_1 from Δy_1 and Δy_2 are treated as "equal". This means that any error in Δy_2 will be introduced into Δs_1. While the normwise error in $\Delta\mathbf{s}$ may be acceptable, the relative error in Δs_1 may be too large since Δy_2 is large compared to s_1. This large relative error is then passed on to the entries of $\Delta\mathbf{x}$ in (3.7b). Thus, it is necessary to find a way to

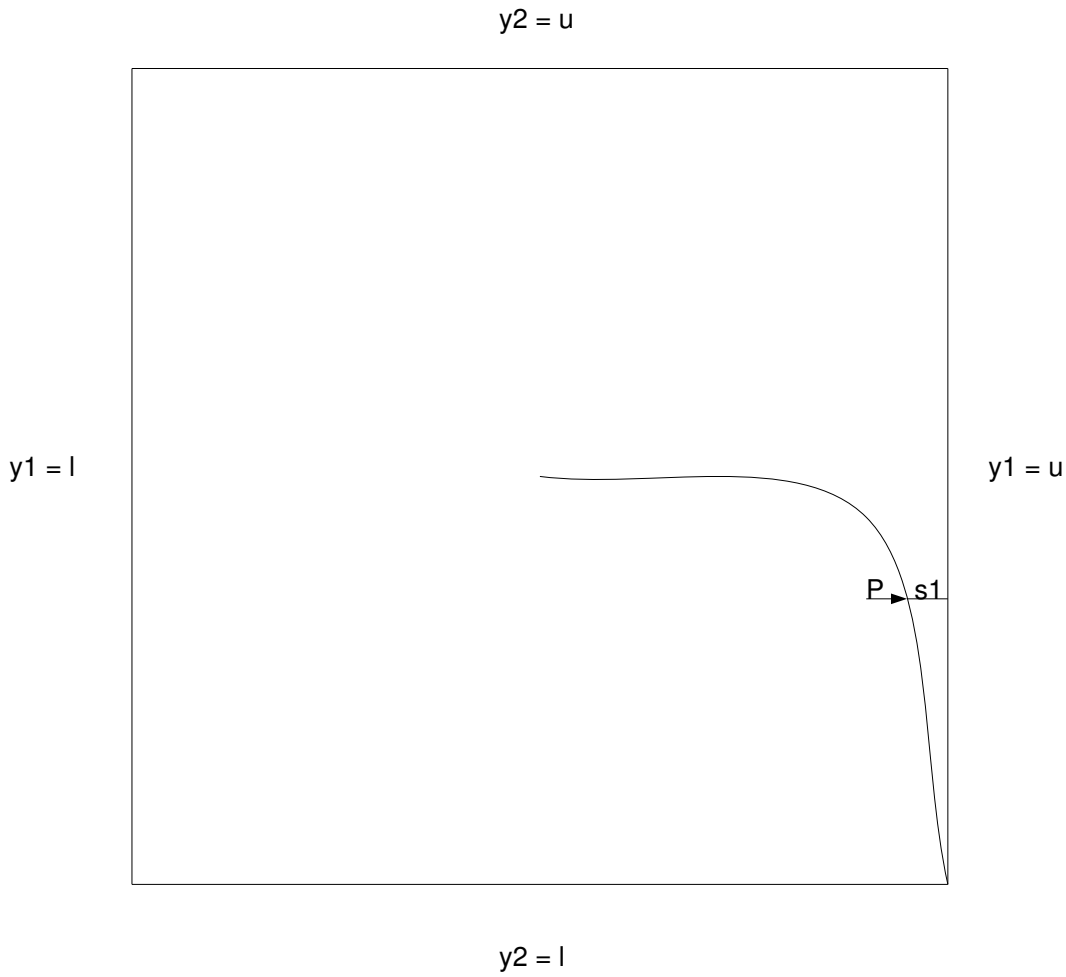

Figure 3.1: The feasible region and the central path for the given example appear below. Notice that the central path approaches the y_1 boundary first and then proceeds to the optimal solution. Notice that the magnitudes of the changes in the two directions can vary by a significant amount near the boundary.

compute these directions with more accuracy than can be guaranteed by using (3.7a) and (3.7b).

Rather than computing $\Delta\mathbf{s}$ and $\Delta\mathbf{x}$, we propose computing scaled versions of these directions. This approach makes use of an orthogonal projection formed by orthogonal factors computed by the COD algorithm. The computation of the scaled $\Delta\mathbf{s}$ is derived as follows. The normal equations (3.5) imply

$$\Delta\mathbf{y} = (ADA^T)^{-1}AD(\mathbf{s} - \bar{\mu}X^{-1}\mathbf{e}).$$

Substituting this into (3.7a) and multiplying both sides by $D^{1/2}$ give

$$D^{1/2}\Delta\mathbf{s} = -D^{1/2}A^T\Delta\mathbf{y} = D^{1/2}A^T(ADA^T)^{-1}AD(\bar{\mu}X^{-1}\mathbf{e} - \mathbf{s}).$$

Notice that D is embedded in a matrix that must be inverted. When D is ill conditioned, this cannot be done accurately, but this problem can be avoided. Working through the steps of the COD algorithm shows that $D^{1/2}A^T = P^TZ_1U_1Q^T$. Substituting for $D^{1/2}A^T$ in the previous equation yields the following:

$$D^{1/2}\Delta\mathbf{s} = PZ_1Z_1^TP^TD^{1/2}(\bar{\mu}X^{-1}\mathbf{e} - \mathbf{s}). \tag{3.8}$$

The goal is to obtain a forward error bound on the entries of $D^{1/2}\Delta\mathbf{s}$, so it is necessary to determine forward error bounds for the factors on the right-hand side. We start by showing that the error in Z_1 is small. Note that this is asserted in Chapter 2 but not proved. The theorem and proof appear below.

Theorem 3.3.1 *Let Z_1 be as in step 3 of the COD algorithm, and let $\hat{Z}_1$ be the corresponding computed matrix. Then*

$$\left\| Z_1 - \hat{Z}_1 \right\| \leq \epsilon \cdot k \cdot f(A) + O(\epsilon^2), \tag{3.9}$$

where k is a small constant, ϵ is machine roundoff, and $f(A)$ is a function of A not depending on D.

Proof: Recall from Step 3 of the COD algorithm that $R^T = Z_1 U_1$. In order to use the results of the previous chapter, we consider instead $R^T D^{-1/2} = Z_1 U_0$, where $U_0 = U_1 D^{-1/2}$. Let $\hat{Z}_1$ and $\hat{U}_0$ be the corresponding computed matrices, and assume that the signs of the diagonal entries of $\hat{U}_0$ are the same as those of the corresponding entries of U_0. Standard backward error analysis implies that there exists a matrix $\tilde{Z}_1$ with exactly orthonormal columns such that $\left\| \tilde{Z}_1 - \hat{Z}_1 \right\| \leq c\epsilon$ and such that $\tilde{Z}_1 \hat{U}_0 = Z_1 U_0 + E$, where $\|E\| \leq c\epsilon \left\| R^T D^{-1/2} \right\|$, and c is a small constant. We proceed as follows. Multiplying both sides on the right by U_0^{-1} gives

$$\tilde{Z}_1 \hat{U}_0 U_0^{-1} = Z_1 + E U_0^{-1}. \tag{3.10}$$

We now multiply this equation by its transpose to obtain

$$(\tilde{Z}_1 \hat{U}_0 U_0^{-1})^T (\tilde{Z}_1 \hat{U}_0 U_0^{-1}) = (Z_1 + E U_0^{-1})^T (Z_1 + E U_0^{-1}),$$

or

$$(\hat{U}_0 U_0^{-1})^T (\hat{U}_0 U_0^{-1}) = I + Z_1^T E U_0^{-1} + (Z_1^T E U_0^{-1})^T + O(\epsilon^2).$$

Notice that this is the Cholesky factorization of some matrix.

Now let $E_1 = Z_1^T E U_0^{-1}$. Notice that $E_1 + E_1^T$ can be written as

$$E_1 + E_1^T = L + W + L^T,$$

where L is strictly lower triangular and W is diagonal. It is not difficult to see that

$$(I + W/2 + L^T)^T (I + W/2 + L^T) = I + E_1 + E_1^T + O(\epsilon^2).$$

Also, $I + W/2 + L^T$ is upper triangular, so by uniqueness of the Cholesky factorization

$$\hat{U}_0 U_0^{-1} = I + W/2 + L^T + O(\epsilon^2).$$

Substituting this into (3.10) yields

$$\tilde{Z}_1 (I + W/2 + L^T + O(\epsilon^2)) = Z_1 + E U_0^{-1}$$

and thus

$$Z_1 - \tilde{Z}_1 = \tilde{Z}_1 (W/2 + L^T) + E U_0^{-1} + O(\epsilon^2).$$

Since $W/2 + L^T$ is part of $E_1 + E_1^T$, then

$$\left\| W/2 + L^T \right\| \leq 2 \left\| E_1 \right\| \leq 2 \cdot \|E\| \cdot \left\| U_0^{-1} \right\|.$$

So

$$\begin{aligned} \left\| Z_1 - \hat{Z}_1 \right\| &\leq 3 \cdot \|E\| \cdot \left\| U_0^{-1} \right\| + O(\epsilon^2) + c\epsilon \\ &\leq 3 \cdot c \cdot \epsilon \cdot \left\| R^T D^{-1/2} \right\| \cdot \left\| U_0^{-1} \right\| + c \cdot \epsilon + O(\epsilon^2). \end{aligned}$$

Recall from Chapter 2 that $\left\| R^T D^{-1/2} \right\| \leq f_1(A)$ and that $\left\| U_0^{-1} \right\| = \left\| D^{1/2} U_1^{-1} \right\| \leq f_2(A)$, where $f_1(A)$ and $f_2(A)$ do not depend on D. Substituting these bounds into the above inequality and combining terms yield the final result:

$$\left\| Z_1 - \hat{Z}_1 \right\| \leq \epsilon \cdot k \cdot f(A) + O(\epsilon^2).$$

(Notice that the right-hand side of the error bound does not depend on the weight matrix D.) $\square$

We now return to the computation of $D^{1/2}\Delta\mathbf{s}$. Recall (3.8):

$$D^{1/2}\Delta\mathbf{s} = PZ_1Z_1^TP^TD^{1/2}(\bar{\mu}X^{-1}\mathbf{e} - \mathbf{s}).$$

Theorem 3.3.1 indicates that the error in the computation of the orthogonal projection $PZ_1Z_1^TP^T$ is bounded by a function that is independent of the weight matrix D. In order to say something useful about the factor $D^{1/2}(\bar{\mu}X^{-1}\mathbf{e} - \mathbf{s})$, we use the relationships $D^{1/2} = S^{-1/2}X^{1/2}$ and $XS\mathbf{e} \approx \mu\mathbf{e}$. These two facts imply that $D^{1/2}(\bar{\mu}X^{-1}\mathbf{e} - \mathbf{s}) \approx \mu^{1/2}\mathbf{e}$. Since forming the right-hand side involves scaling and subtraction (to obtain $\mu^{1/2}\mathbf{e}$) and then multiplication by an orthogonal projection, it is not difficult to show that

$$\left\| D^{1/2}\Delta\mathbf{s} - D^{1/2}\Delta\hat{\mathbf{s}} \right\| \leq \epsilon \cdot \mu^{1/2} \cdot c \cdot g(A),$$

where $D^{1/2}\Delta\mathbf{s}$ is the exact result, $D^{1/2}\Delta\hat{\mathbf{s}}$ is the computed result, $\epsilon > 0$ is machine precision, c is a constant, and $g(A)$ is a function of A that does not depend on D. An entrywise error bound can now be obtained as follows. Notice that $(D^{1/2}\Delta\mathbf{s})_i = d_i\Delta s_i$. So

$$\begin{aligned} d_i^{1/2}\left|\Delta s_i - \Delta\hat{s}_i\right| &= \left| d_i^{1/2}\Delta s_i - d_i^{1/2}\Delta\hat{s}_i \right| \\ &\leq \left\| D^{1/2}\Delta\mathbf{s} - D^{1/2}\Delta\hat{\mathbf{s}} \right\| \\ &\leq \epsilon \cdot \mu^{1/2} \cdot c \cdot g(A). \end{aligned}$$

Recall that $D^{1/2} = S^{-1/2}X^{1/2}$ and $XS\mathbf{e} \approx \mu\mathbf{e}$. Thus, $d_i^{1/2} \approx \frac{\mu^{1/2}}{s_i}$ and

$$\frac{\mu^{1/2}\,|\Delta s_i - \Delta\hat{s}_i|}{s_i} \approx d_i^{1/2}\,|\Delta s_i - \Delta\hat{s}_i| \leq \epsilon \cdot \mu^{1/2} \cdot c \cdot g(A), \tag{3.11}$$

and thus,

$$\frac{|\Delta s_i - \Delta\hat{s}_i|}{s_i} \leq \epsilon \cdot c \cdot g(A) \tag{3.12}$$

for $1 \leq i \leq n$. Again, the key point to note is that the right-hand side of (3.12) is independent of D.

Computing $\Delta\mathbf{x}$ with the required accuracy is now straightforward. First, recall that (3.7b) says

$$\Delta\mathbf{x} = \bar{\mu}S^{-1}\mathbf{e} - \mathbf{x} - S^{-1}X\Delta\mathbf{s}.$$

Recall that $D = S^{-1}X$. Substituting this into the previous equation and multiplying both sides by $D^{-1/2}$ yields

$$D^{-1/2}\Delta\mathbf{x} = D^{-1/2}(\bar{\mu}S^{-1}\mathbf{e} - \mathbf{x}) - D^{1/2}\Delta\mathbf{s}.$$

It is also not difficult to show that the entries of both terms on the right-hand side are of similar magnitude. Since only scaling and subtraction are involved, it is easy to show that the entries of $D^{-1/2}\Delta\mathbf{x}$ are computed accurately. Using an argument similar to the one for $\Delta\mathbf{s}$, we find that the entries of $\Delta\mathbf{x}$ also satisfy an error bound of the form

$$\frac{|\Delta x_i - \Delta\hat{x}_i|}{x_i} \leq \epsilon \cdot c \cdot g(A), \tag{3.13}$$

where $\epsilon > 0$ is machine precision, c is a constant, and $g(A)$ is a function of A. These error bounds demonstrate that despite contributions from directions of different magnitudes, this method computes $\Delta\mathbf{x}$ and $\Delta\mathbf{s}$ with suitable relative accuracy in each component.

Now that the stability results have been established, it remains to be seen how the extended COD algorithm performs in practice. The results of computational experiments appear in the next section.

3.4 Numerical Results

In order to determine the effectiveness of the extended COD algorithm in practice, it has been implemented and incorporated into two primal-dual path-following interior point methods. The first is a feasible predictor-corrector method proposed by Mizuno, Todd, and Ye [26], and the second is the infeasible predictor-corrector method of Mehrotra [25] (as implemented by Zhang in LIPSOL [47]). Each algorithm was also implemented with some version of Cholesky factorization for comparison purposes. The algorithms and the results of the experiments are described in the following subsections.

Two types of test problems were used. The first group includes small shortest path problems designed to be "torture" tests for the algorithms. Even though there are specialized algorithms for solving shortest path problems, they make nice test problems for two reasons. First, the matrix A for these problems is a reduced node-arc incidence (RNAI) matrix. Vavasis showed that χ_A and $\bar{\chi}_A$ are bounded by the size of the matrix for RNAI matrices [41]. Thus, it is easy to evaluate the strength of the error bounds in this case. Secondly, it is easy to determine the solution by observation, so it is possible to determine whether or not the algorithms are performing according to expectations.

The other set of problems consists of the small NETLIB problems. These were tested in two ways. First, the COD implementations of the algorithms were used to compute the solutions to the problems. These solutions were compared to known solutions in order to determine whether or not the COD algorithm computed the correct solution. The results were also compared to those obtained using the Cholesky implementations. The problems were then modified in the following way: near-degeneracies were introduced by making some inactive constraints at the solution near active and then modifying the original problems accordingly. This was done in such a way that the solutions are still known. The results obtained using both the COD and the Cholesky solvers were compared in order to determine which solver performed better.

3.4.1 The Algorithm of Mizuno, Todd, and Ye

The first interior point algorithm used for computational experiments is a predictor-corrector method proposed by Mizuno, Todd, and Ye [26]. Briefly, each main iteration consists of the following steps. A "predictor" step is computed by solving (3.4a), (3.4b), and (3.4c) with $\bar{\mu} = 0$. Then a line search is performed to determine the longest step length that will keep the iterate within a specified distance $\alpha \in (0,1)$ of the central path. For these experiments, the parameter α is chosen to be fairly large to allow for longer steps rather than shorter, more centered steps. After $\mathbf{x}, \mathbf{y}, \mathbf{s}$, and μ have been updated according to the predicted step, the iterates are centered. This requires repeatedly solving (3.4a), (3.4b), and (3.4c) with the new value of μ and updating $\mathbf{x}, \mathbf{y}$, and $\mathbf{s}$ until the result is within $\beta \in (0,1)$ of the central path,

where $\beta < \alpha$. In contrast to α, β is fairly small since the goal is to bring the iterates in close to the central path. (Notice that this implies that β should be chosen such that $\beta < \alpha$.) An outline of the algorithm is given below.

Algorithm: Mizuno, Todd, and Ye

Given an approximately centered starting point $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)}, \mathbf{s}^{(0)})$ and $\mu^{(0)}$, proceed as follows.

While $(\mathbf{x}^{(k)})^T \mathbf{s}^{(k)} > tol$

(predictor step)

compute $\Delta \mathbf{x}_p^{(k)}, \Delta \mathbf{y}_p^{(k)}, \Delta \mathbf{s}_p^{(k)}$

determine largest θ such that $\left\| \frac{(\mathbf{x}^{(k)} + \theta \Delta \mathbf{x}_p^{(k)}).*(\mathbf{s}^{(k)} + \theta \Delta \mathbf{s}_p^{(k)})}{\mu^{(k)}} - 1 \right\| < \alpha$

set $\bar{\mathbf{x}} = \mathbf{x}^{(k)} + \theta \Delta \mathbf{x}_p^{(k)}, \bar{\mathbf{y}} = \mathbf{y}^{(k)} + \theta \Delta \mathbf{y}_p^{(k)}, \bar{\mathbf{s}} = \mathbf{s}^{(k)} + \theta \Delta \mathbf{s}_p^{(k)}, \mu^{(k)} = (1 - \theta)\mu^{(k)}$

While $\left\| \frac{\bar{\mathbf{x}}.*\bar{\mathbf{s}}}{\mu^{(k)}} - 1 \right\| > \beta$

(corrector step)

compute $\Delta \mathbf{x}_c^{(k)}, \Delta \mathbf{y}_c^{(k)}, \Delta \mathbf{s}_c^{(k)}$

set $\bar{\mathbf{x}} = \bar{\mathbf{x}} + \Delta \mathbf{x}_c^{(k)}, \bar{\mathbf{y}} = \bar{\mathbf{y}} + \Delta \mathbf{y}_c^{(k)}, \bar{\mathbf{s}} = \bar{\mathbf{s}} + \Delta \mathbf{s}_c^{(k)}$

end

set $k = k + 1, \mathbf{x}^{(k)} = \bar{\mathbf{x}}, \mathbf{y}^{(k)} = \bar{\mathbf{y}}, \mathbf{s}^{(k)} = \bar{\mathbf{s}}$

end

The question that remains is that of how to compute the feasible starting point $(\mathbf{x}^{(0)}, \mathbf{y}^{(0)}, \mathbf{s}^{(0)})$ and μ_0. In order to do this, the algorithm is implemented in a Phase

I - Phase II manner. The linear programming problem is solved in Phase II, but the initial point is determined in Phase I. The procedure is as follows. Artificial variables are introduced into the original problem in order to construct a larger linear programming problem for which a feasible starting point is known. This larger problem can be solved to optimality, and it would be possible to obtain the optimal solution to the original problem. However, this would require more work than solving the original problem, so the interior point method is run on the larger problem only until a suitable starting point for the original problem can be extracted. Once this has been done, the interior point method is used on the original problem with this starting point. The results of the computational tests are taken from Phase II, so we go into no further detail about Phase I.

An example from the shortest path test problems is given in Figure 3.2. The matrix A is the reduced node-arc incidence matrix for this graph, the entries of the vector $\mathbf{c}$ are the edge weights, and the vector $\mathbf{b}$ contains a -1 in the fifth entry and a 1 in the second entry (thus describing the locations of the source and the sink). When $\delta = 10^{-6}$, the Cholesky implementation of the Mizuno, Todd, and Ye Algorithm reaches a point where the coefficient matrix of the normal equations is no longer recognized by MATLAB as being positive definite. The algorithm stops at this point without finding the shortest path. On the other hand, the COD implementation computes the shortest path with accuracy on the order of machine precision.

Recall that the next set of examples includes some of the small problems in NETLIB test set. The purpose of these tests was to verify that the implementations work correctly for problems where the solution is already known. The results are

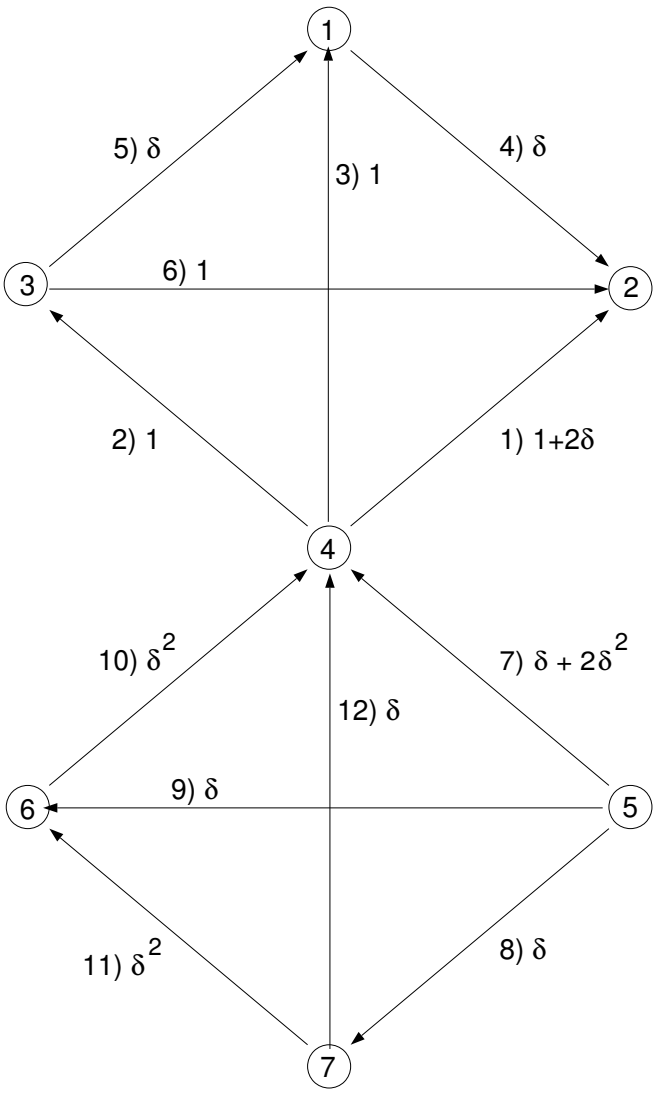

Figure 3.2: The graph below illustrates a near-degenerate shortest path problem. It is clear that the shortest path from the source (node 5) to the sink (node 2) consists of edges 9, 10, 3, and 4. However, the Cholesky implementations fail with small values of δ while the COD implementations find the correct path.

Table 3.1: This table contains a summary of the numerical results for the NETLIB test problems solved using the Mizuno, Todd, and Ye algorithm. Notice that the performance of the two implementations is comparable.

	problem	number of calls to Cholesky	number of calls to COD
NETLIB problems unchanged	afiro	21	22
	sc50a	30	30
	sc50b	27	27
with near-degeneracy	afiro	42	43
	sc50a	51	52
	sc50b	25	25
with near-degeracy & permutation	afiro	42	42
	sc50a	52	52
	sc50b	15	25

shown in Table 3.1. Notice that both the Cholesky and the COD implementations compute the correct solution, and the same number of function calls is made in each case. This is not surprising since the Cholesky solvers are quite robust, and we expect the real advantage of the COD algorithm to be seen in near-degenerate problems.

When near-degeneracy is introduced into the problems, we again find that both implementations make the same number of function calls in converging to the solution. (See Table 3.1.) Cholesky factorization is known to be sensitive to permutations in the rows of A, so experiments were run in which permutations were also introduced into the near-degenerate problems. Once again, the performance of the two implementations is comparable. Similar results were obtained for other NETLIB test problems used for numerical tests. It is unclear at this point why the Cholesky algorithm performs as well as the COD algorithm on the near-degenerate NETLIB

problems. However, while these results are inconclusive, the results obtained for the shortest path problems (like the one in Figure 3.2) indicate that there are cases in which it is advantageous to use the COD algorithm.

3.4.2 LIPSOL

LIPSOL is Yin Zhang's implementation of Mehrotra's interior point method. This is an infeasible second-order predictor-corrector method. Because it is infeasible, the iterates move toward feasibility as they move toward the solution, and no linearization occurs in the centering equation since the method is second order. Therefore, the system of equations to be solved is somewhat different that (3.4a), (3.4b), and (3.4c). Instead, it is necessary to compute $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$ such that

$$\begin{aligned} A\Delta\mathbf{x} &= \mathbf{b} - A\mathbf{x}, \\ A^T\Delta\mathbf{y} + \Delta\mathbf{s} &= \mathbf{c} - A^T\mathbf{y} - \mathbf{s}, \\ X\Delta\mathbf{s} + S\Delta\mathbf{x} &= \mu\mathbf{e} - XS\mathbf{e} - \Delta X\Delta S\mathbf{e}. \end{aligned} \tag{3.14}$$

This system is broken down into two steps. The first is the predictor step, which finds the affine scaling search direction. This is done by solving

$$\begin{aligned} A\Delta\mathbf{x}_p &= \mathbf{b} - A\mathbf{x}, \\ A^T\Delta\mathbf{y}_p + \Delta\mathbf{s}_p &= \mathbf{c} - A^T\mathbf{y} - \mathbf{s}, \\ X\Delta\mathbf{s}_p + S\Delta\mathbf{x}_p &= -XS\mathbf{e}. \end{aligned} \tag{3.15}$$

Notice that the nonzero terms on the right-hand side introduce additional terms into the least-squares problem defined in §3.2. Following a derivation similar to that in

§3.2 yields the following normal equations for a weighted least-squares problem:

$$ADA^T\Delta\mathbf{y}_p = AD\left[(c - A^T\mathbf{y}) + D^{-1}\mathbf{p}\right], \tag{3.16}$$

where $D = S^{-1}X, \mathbf{p} = A^T\mathbf{q}$, and $AA^T\mathbf{q} = \mathbf{b} - A\mathbf{x}$. Notice that both terms on the right-hand side have the same order of magnitude.

We take a moment here to point out that the Cholesky algorithm used in LIPSOL is the sparse Cholesky algorithm of Ng and Peyton [29], with the addition of the ad hoc rank determination described in §2.4. Thus, this implementation is expected to be quite robust.

Similarly, the equations defining $\Delta\mathbf{x}_p$ and $\Delta\mathbf{s}_p$ are somewhat different. Continuing with the derivation as in §3.3 yields the following equation for the scaled version of $\Delta\mathbf{s}_p$:

$$D^{1/2}\Delta\mathbf{s}_p = D^{1/2}(\mathbf{c} - A^T\mathbf{y} - \mathbf{s}) - PZ_1Z_1^TP^T\left[D^{1/2}(\mathbf{c} - A^T\mathbf{y}) + D^{-1/2}\mathbf{p}\right]. \tag{3.17}$$

We offer an intuitive explanation as to why this computes $\Delta\mathbf{s}_p$ that satisfies (3.12). As the solution is approached and D becomes ill conditioned, the first term on the right-hand side tends to zero and makes a negligible contribution. In the second term, $\mathbf{c} - A^T\mathbf{y}$ tends to $\mathbf{s}$, and $\mathbf{p}$ looks like $\Delta\mathbf{x}_p$. So arguments similar to those in §3.3 imply that the nonorthogonal factor in the second term looks like $\mu^{1/2}\mathbf{e}$, and an error bound like (3.12) is obtained. Arguments similar to those in §3.3 imply that $\Delta\mathbf{x}_p$ computed by

$$D^{-1/2}\Delta\mathbf{x}_p = -D^{-1/2}\mathbf{x} - D^{1/2}\Delta\mathbf{s}_p \tag{3.18}$$

satisfies an error bound like (3.13).

After the predictor step is determined, the corrector step finds the other component of the search direction. This takes into account the centering and the second-order term. The values of $\Delta\mathbf{x}_p$ and $\Delta\mathbf{s}_p$ computed in the previous step are used to "predict" the second-order term, and the system of equations solved at this step is given by

$$\begin{aligned} A\Delta\mathbf{x}_c &= \mathbf{0}, \\ A^T\Delta\mathbf{y}_c + \Delta\mathbf{s}_c &= \mathbf{0}, \\ X\Delta\mathbf{s}_c + S\Delta\mathbf{x}_c &= \mu\mathbf{e} - \Delta X_p \Delta S_p \mathbf{e}. \end{aligned} \tag{3.19}$$

Again, the equations look somewhat different from those for a feasible method, but the stability arguments are similar. Thus, the specific equations and discussions are omitted here.

To obtain the solution to (3.14), the solutions to (3.15) and (3.19) are combined as follows:

$$\begin{aligned} \Delta\mathbf{x} &= \Delta\mathbf{x}_p + \Delta\mathbf{x}_c, \\ \Delta\mathbf{y} &= \Delta\mathbf{y}_p + \Delta\mathbf{y}_c, \\ \Delta\mathbf{s} &= \Delta\mathbf{s}_p + \Delta\mathbf{s}_c. \end{aligned}$$

There was one additional change made to the LIPSOL code. In LIPSOL, there are constant upper bounds imposed on the entries of the weight matrix D. These prevent D becoming arbitrarily ill conditioned, and it is not known how these caps affect the convergence of the interior point algorithm (since the systems of equations are therefore solved with a perturbed D). As we are interested in the performance of algorithms when D becomes arbitrarily ill conditioned, the caps were removed.

Table 3.2: This table contains a summary of the numerical results for the NETLIB test problems solved using LIPSOL. Notice that while the performances of the Cholesky implementation and the COD implementation are comparable for the unchanged problems, the COD implementation is more reliable for the near-degenerate problems.

	problem	number of iterations, Cholesky	number of iterations COD
NETLIB problems unchanged	afiro	7	7
	sc50a	9	9
	sc50b	8	8
with near-degeneracy	afiro	n.c.	11
	sc50a	n.c.	12
	sc50b	n.c.	9

LIPSOL was run on all of the same test problems as the Mizuno, Todd, and Ye algorithm. For the example illustrated in Figure 3.2 with $\delta = 10^{-8}$, the Cholesky implementation converges to $\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & .2 & .8 & 0 & .8 & .8 & 0 \end{bmatrix}^T$, which is not even a valid path. On the other hand, the COD implementation finds the correct shortest path.

Table 3.2 contains the results for the NETLIB problems. For the unmodified problems, both implementations converge to the solution in the same number of iterations (with two function calls per iteration). For the near-degenerate problems, however, the Cholesky implementation halts without converging, but the COD implementation converges to the solution. Thus, the advantage of using the extended COD algorithm is clear in these experimental results.

3.5 Related Work

The problem of stably solving the ill-conditioned linear systems that arise in interior point methods has received a fair amount of attention [13]. Among those who have studied such problems are Coleman and Liu [8], Forsgren, Gill, and Shinnerl [11], Gill, Saunders, and Shinnerl [14], Gould [18], Murray [27], Nash and Sofer [28], M. Wright [43], S. Wright [46]. One difference between these other works and ours may be summarized as follows. These other works typically look at the more general problem $\min \|H^{-1/2}(A\mathbf{y} - \mathbf{b})\|$ where H is symmetric, positive definite, but not necessarily diagonal. This is a problem that we currently cannot address with our techniques. In some recent work, Forsgren [10] has derived a result similar to Theorem 1.1.1 for such matrices H that are also diagonally dominant. This is accomplished by defining a new type of diagonal decomposition. If this decomposition can be found accurately, then the work of this thesis applies. However, the applicability of the extended COD algorithm to the general case has not yet been determined.

When specialized to diagonal weight matrices D, these authors consider a more restricted problem in that they all make an assumption that the large and small entries on the diagonal of D have some correlation with the columns of A^T. This corresponds to a nondegeneracy assumption about the underlying optimization problem. In contrast, our method does not involve any restrictions about where "large" versus "small" entries of D can appear. As hoped, numerical results indicate that the COD method is more reliable when there is degeneracy or near-degeneracy in the underlying optimization problem.

Chapter 4

Efficient Solution of the Layered Least-Squares Problem*

The problem under consideration in this chapter is still that of solving a primal-dual pair of linear programs, given again here in standard form:

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \geq \mathbf{0}, \end{aligned}$$

and the dual problem:

$$\begin{aligned} \text{maximize} \quad & \mathbf{b}^T\mathbf{y} \\ \text{subject to} \quad & A^T\mathbf{y} + \mathbf{s} = \mathbf{c}, \\ & \mathbf{s} \geq \mathbf{0}. \end{aligned}$$

*This chapter represents joint work with John R. Gilbert and Stephen A. Vavasis.

As before, $A \in \mathbb{R}^{m \times n}$ $(\text{rank}(A) = m)$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{c} \in \mathbb{R}^n$ are given, and $\mathbf{x}, \mathbf{s} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbb{R}^m$ are unknown. The previous chapter contained a discussion of numerical issues that arise in traditional primal-dual path-following interior point algorithms. It also included a description of how a standard method for solving weighted least-squares problems can be modified and extended to compute search directions accurately despite ill-conditioning that arises from near-degeneracy in the linear programming problem. In this chapter, we look at a new numerical linear algebra problem that arises in an interior point algorithm recently proposed by Vavasis and Ye. The goal is to apply standard algorithms to solve this problem in a stable and efficient manner.

4.1 The LIP Algorithm

The interior point algorithm under consideration is the layered-step interior point (LIP) algorithm that was recently proposed by Vavasis and Ye [42]. This algorithm interleaves traditional path-following steps, such as those described in Chapter 3, with longer layered least-squares (LLS) steps. The essential observation made by Vavasis and Ye is that the central path for a linear programming problem is composed of at most $O(n^2)$ alternating curved and approximately straight segments, and that the approximately straight segments may be traversed in a single step if the LLS direction is followed.

As an example, consider the linear programming problem

$$\begin{aligned} \text{minimize} \quad & y_1 - y_2/100 \\ \text{subject to} \quad & 0 \leq y_1, y_2 \leq 1, \\ & y_1 + y_2 \leq 1.5. \end{aligned}$$

The feasible region for this problem, which is a pentagon and the central path are illustrated in Figure 4.1. Observe that the initial portion of the central path appears to be a straight segment, nearly horizontal. There is a sharp turn near the point $(0, 0.5)$, and then the central path again appears to be a straight segment in a nearly vertical direction. The LIP algorithm would need a single step to follow each of the two straight portions of the central path and would use more traditional path-following, such as one of the methods in Chapter 3, to negotiate the bend near $(0, 0.5)$.

The complexity of this new interior point method, unlike that of its predecessors, is independent of the objective function and the right-hand side vector (and thus is a step closer to "strongly polynomial time"). In other words, the complexity depends only on the constraint matrix A. The novelty in this method is the *layered least-squares* computation mentioned in the previous paragraph. This new numerical linear algebra problem can be regarded as a weighted least-squares problem with infinite weight gaps. The COD Algorithm of Chapter 2 can be used to solve this problem, but it is not well-suited for large sparse problems. In addition, the structure of the LLS problem suggests that the problem can be broken down into smaller problems on which standard algorithms can be used. Vavasis and Ye did not propose any particular algorithm for the LLS problem. The remainder of this chapter contains a

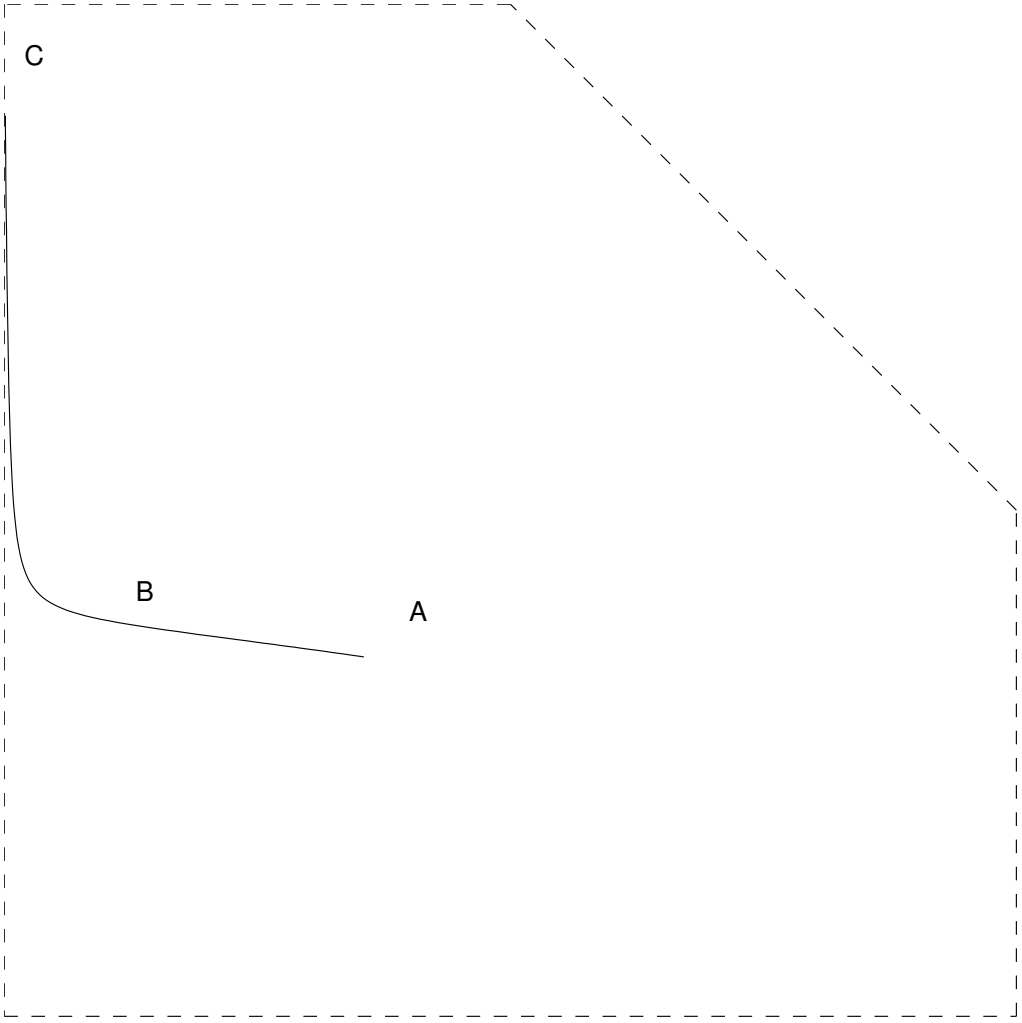

Figure 4.1: This figure shows the feasible region and the central path for the example given in the discussion of the LIP algorithm. Each of the nearly-straight segments is computed using a layered least-squares step, while the curved section beginning at B is traversed by traditional interior point steps.

description of a stable (in the sense of (1.2)) and efficient algorithm based on standard sparse Cholesky methods. An unusual feature is that arithmetic on infinitesimals occurs in the development of the algorithm, but the infinitesimals do not appear in the algorithm itself. A description of some of the implementation details and results of computational experiments are also presented.

Since the LIP Algorithm is a primal-dual method, the LLS problem has both a primal and a dual formulation. We examine the dual formulation first.

4.2 The Dual LLS Problem

As the name suggests, the problem in question must first be partitioned into layers. Let $(\mathbf{x}, \mathbf{y}, \mathbf{s})$ be an approximate central path point with centering parameter μ. Now let

$$\delta_i = \mu x_i / s_i$$

for $i = 1, \ldots, n$, and let

$$D = \text{diag}(\boldsymbol{\delta}) = \mu S^{-1} X,$$

where $S = \text{diag}(\mathbf{s})$ and $X = \text{diag}(\mathbf{x})$. We now define partitions of the index set $\{1, \ldots, n\}$. Take $\boldsymbol{\sigma}$ to be the permutation that sorts the entries of $\boldsymbol{\delta}$ into descending order, i.e.,

$$\delta_{\sigma(1)} \geq \delta_{\sigma(2)} \geq \cdots \geq \delta_{\sigma(n)}.$$

If $g > 1$ is a gap size parameter depending on the matrix A, and i is the smallest index such that $\delta_{\sigma(i)}/\delta_{\sigma(i+1)} > g$, then the first partition is defined by $J_1 = \{\sigma(1), \ldots, \sigma(i)\}$. The next partition is defined by $J_2 = \{\sigma(i+1), \ldots, \sigma(i+k)\}$, where k is the smallest

index such that $\delta_{\sigma(i+k)}/\delta_{\sigma(i+k+1)} > g$. The remaining partitions $J_3, \ldots, J_p$ are defined in a similar manner.

The matrices and vectors used in the dual LLS step $(A, D, \mathbf{s})$ are partitioned according to $J_1, \ldots, J_p$ as follows. Let $A_1, \ldots, A_p$ be the columns of A indexed by $J_1, \ldots, J_p$, respectively, and let $D_1, \ldots, D_p$ be the diagonal matrices whose diagonal entries are the entries of D indexed by $J_1, \ldots, J_p$. Similarly, $\mathbf{s}$ is partitioned into $\mathbf{s}_1, \ldots, \mathbf{s}_p$. Notice that in the interior point setting, A is fixed throughout, but $\mathbf{s}$, D, and $J_1, \ldots, J_p$ vary from one iteration to the next.

Using these partitioned matrices and vectors, the dual LLS problem is defined as follows. Let $V_0 = \mathbb{R}^m$. For $k = 1, \ldots, p$, define

$$V_k^D = \{\text{minimizers } \Delta\mathbf{y} \text{ of } \left\| D_k^{1/2}(A_k^T \Delta\mathbf{y}^{(k)} - \mathbf{s}_k) \right\| \text{ subject to } \Delta\mathbf{y} \in V_{k-1}^D\}. \quad (4.1)$$

Notice that $V_0^D \supset V_1^D \supset \cdots \supset V_p^D$, and the solution $\Delta\mathbf{y}^*$ is the unique element of V_p. Note also that if l is the minimum index such that $[A_1, \ldots, A_l]$ is full rank, then the solution is uniquely determined at the lth step. For example, if $A_1 A_1^T$ has rank m, then the solution is the unique minimizer of $\left\| D_1^{1/2}(A_1^T \Delta\mathbf{y}^{(1)} - \mathbf{s}_1) \right\|$.

The definition of the LLS problem in the previous paragraph indicates that a sequence of constrained weighted least-squares problems must be solved in order to obtain $\Delta\mathbf{y}^*$. The Lagrange multipler conditions for these problems are:

$$\begin{aligned} A_1 D_1 A_1^T \Delta\mathbf{y}^* - A_1 D_1 \mathbf{s}_1 &= \mathbf{0}, \\ A_2 D_2 A_2^T \Delta\mathbf{y}^* - A_2 D_2 \mathbf{s}_2 &= A_1 \boldsymbol{\lambda}_{2,1}, \\ &\vdots \\ A_p D_p A_p^T \Delta\mathbf{y}^* - A_p D_p \mathbf{s}_p &= A_1 \boldsymbol{\lambda}_{p,1} + \cdots + A_{p-1} \boldsymbol{\lambda}_{p,p-1}, \end{aligned} \quad (4.2)$$

for some Lagrange multipliers $\boldsymbol{\lambda}_{i,j}$. Vavasis and Ye showed that these conditions uniquely define $\Delta\mathbf{y}^*$ and used this result in proving that the LLS problem is the limiting case of an ordinary weighted least squares problem. In particular, let $\Xi(\epsilon)$ denote the $n \times n$ diagonal matrix depending on $\epsilon > 0$ whose entries are as follows. For $k = 1, \ldots, p$ and for indices $i \in J_k$, the (i,i) entry is ϵ^k. Let $\Delta\mathbf{y}^*$ be the LLS solution, and let $\Delta\mathbf{y}(\epsilon)$ be the minimizer of $\left\| \Xi(\epsilon) D^{1/2} (A^T \Delta\mathbf{y} - \mathbf{s}) \right\|$. Then $\lim_{\epsilon \to 0+} \Delta\mathbf{y}(\epsilon) = \Delta\mathbf{y}^*$. Special cases of this result were known earlier; see e.g. [22, Ch. 22]. It is this weighted least-squares problem that can be solved using the COD algorithm; however, we propose a more efficient method for computing the LLS step.

In order to reduce the amount of notation, we assume that the weight matrix D is the identity matrix (or equivalently, we replace $D^{1/2}A^T$ by A^T and $D^{1/2}\mathbf{s}$ with $\mathbf{s}$). This assumption is valid for the following reason. In the LIP method, the particular choice of partitions arising in the LLS step is such that each submatrix $D_1, \ldots, D_p$ has an upper bound (in terms of A) on its condition number, even though there is no prior upper bound on the condition of D. Note that the algorithm presented here is unstable if any D_k is not well conditioned.

We now derive the algorithm for the dual LLS problem. For simplicity, the algorithm is presented for the case where there are only two layers. It is not difficult to extend it to the general case, as the algorithm is recursive in nature. First, consider the normal equations for minimizing $\left\| A_1^T \Delta\mathbf{y}^{(1)} - \mathbf{s}_1 \right\|$, i.e.,

$$A_1 A_1^T \Delta\mathbf{y}^{(1)} = A_1 \mathbf{s}_1.$$

An $L\bar{D}L^T$ factorization is performed on $A_1 A_1^T$, where L is a lower triangular matrix,

and $\bar{D}$ is diagonal. Since the rank of A_1 may be less than m and is not known *a priori*, the $L\bar{D}L^T$ algorithm must test for rank deficiency. In exact arithmetic, rank deficiency is manifested as a zero on the diagonal during the factorization. Note that because $A_1A_1^T$ is positive semidefinite, a zero on the diagonal means that the entire row and column of the uneliminated portion of the matrix must be zero. (In finite precision, rank deficiency would be manifested as small entries on the diagonal. Such entries are changed to exact zeros, as are the corresponding uneliminated row and column.) The proposed algorithm skips such a row and column and continues with the factorization. This yields a factorization $A_1A_1^T = L^{(1)}\bar{D}^{(1)}L^{(1)T}$, where some diagonal entries of $\bar{D}^{(1)}$ are zeros, and the corresponding columns of $L^{(1)}$ are columns of the identity matrix.

Let $\epsilon > 0$ denote an infinitesimally small number. Since the LLS problem is the limiting case of an ordinary weighted least-squares problem, it is reasonable to represent the factorization in terms of infinitesimals (although there are no infinitesimals in the actual algorithm). Assuming that A_1 is not full rank, we now consider the factorization of $A_1A_1^T + \epsilon A_2A_2^T$. This can be written as

$$A_1A_1^T + \epsilon A_2A_2^T = L^{(1)}(\bar{D}^{(1)} + \epsilon W^{(2)})L^{(1)T},$$

where

$$W^{(2)} = (L^{(1)})^{-1}A_2A_2^T(L^{(1)})^{-T}$$

and is found by triangular substitution. The next step is to factor the symmetric positive semidefinite matrix $\bar{D}^{(1)} + \epsilon W^{(2)}$. To avoid confusion, let us assume that the nonzero diagonal entries of $\bar{D}^{(1)}$ are the first k entries, and the remaining $m - k$

entries are zeros. Let I_1 be the set of indices in which $\bar{D}^{(1)}$ is not zero. Then we can write $\bar{D}^{(1)} + \epsilon W^{(2)}$ in block matrix form and carry out block elimination:

$$\begin{aligned}
\bar{D}^{(1)} + \epsilon W^{(2)} &= \begin{bmatrix} \bar{D}_1 + \epsilon W_{11}^{(2)} & \epsilon (W_{21}^{(2)})^T \\ \epsilon W_{21}^{(2)} & \epsilon W_{22}^{(2)} \end{bmatrix} \\
&= \begin{bmatrix} I & 0 \\ \epsilon (W_{21}^{(2)})(\bar{D}_1^{(1)} + \epsilon W_{11}^{(2)})^{-1} & I \end{bmatrix} \begin{bmatrix} \bar{D}_1^{(1)} + \epsilon W_{11}^{(2)} & 0 \\ 0 & \epsilon W_{22}^{(2)} \end{bmatrix} \\
&\quad \begin{bmatrix} I & \epsilon (\bar{D}_1^{(1)} + \epsilon W_{11}^{(2)})^{-1} (W_{21}^{(2)})^T \\ 0 & I \end{bmatrix},
\end{aligned}$$

which we write as $M^{(1)} B^{(2)} M^{(1)T}$. Dropping higher-order infinitesimals yields

$$M^{(1)} = \begin{bmatrix} I & 0 \\ \epsilon (W_{21}^{(2)})(\bar{D}_1^{(1)})^{-1} & I \end{bmatrix}, \text{ and } B^{(2)} = \begin{bmatrix} \bar{D}_1^{(1)} & 0 \\ 0 & \epsilon W_{22}^{(2)} \end{bmatrix}.$$

The next step is an $L\bar{D}L^T$ factorization of $B^{(2)}$. This yields

$$B^{(2)} = L^{(2)} \bar{D}^{(2)} L^{(2)T}$$

and thus,

$$A_1 A_1^T + \epsilon A_2 A_2^T = L^{(1)} M^{(1)} L^{(2)} \bar{D}^{(2)} L^{(2)T} M^{(1)T} L^{(1)T}, \tag{4.3}$$

where $L^{(1)}, L^{(2)}$, and $M^{(1)}$ are lower triangular, and $\bar{D}^{(2)}$ is a diagonal matrix whose first block of entries are $O(1)$ and second block are $O(\epsilon)$. We can now proceed to solving the LLS problem.

To obtain the solution, first form $\mathbf{r}_1 = A_1 \mathbf{s}_1$ and $\mathbf{r}_2 = A_2 \mathbf{s}_2$. Then compute $L^{-1}\mathbf{r}_1 + \epsilon L^{-1}\mathbf{r}_2$, where $L = L^{(1)} M^{(1)} L^{(2)}$. Notice that the two terms must be computed separately because of the large difference in magnitude. If this quantity is not formed

in the way, necessary $O(\epsilon)$ information will be lost. The result is a block vector whose entries are $O(1)$ in I_1 and $O(\epsilon)$ in I_2. To see this, consider performing the forward substitution one step at a time. For example, consider substitution on $\mathbf{r}_1$. Notice that $(L^{(1)})^{-1}\mathbf{r}_1$ has nonzeros only in I_1 because $\mathbf{r}_1$ lies in the range of A_1. (In finite precision arithmetic, this has to be enforced by inserting zeros at this step.) To clarify this, consider the relationship between Cholesky factorization and QR factorization. If $A_1 A_1^T = L^{(1)}\bar{D}^{(1)}(L^{(1)})^T$, then $A_1^T = Q\tilde{D}^{(1)}(L^{(1)})^T$, where $Q = [Q_1, 0]$, Q_1 has rank(A_1) orthonormal columns, and $\tilde{D}^{(1)} = \text{diag}((\bar{D}_1^{(1)})^{1/2}, I)$. Thus,

$$\begin{aligned}(L^{(1)})^{-1}\mathbf{r}_1 &= (L^{(1)})^{-1}L^{(1)}\tilde{D}^{(1)}Q^T\mathbf{s}_1 \\ &= \begin{bmatrix} (\bar{D}_1^{(1)})^{1/2}Q_1^T\mathbf{s}_1 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{r}}_1 \\ \mathbf{0} \end{bmatrix},\end{aligned}$$

where $\bar{\mathbf{r}}_1 = (\bar{D}_1^{(1)})^{1/2}Q_1^T\mathbf{s}_1$. Then forward substitution involving $M^{(1)}$ creates a new vector that is $O(\epsilon)$ in I_2. In particular,

$$(M^{(1)})^{-1}\begin{bmatrix} \bar{\mathbf{r}}_1 \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{r}}_1 \\ -\epsilon W_{21}^{(2)}(D_1^{(1)})^{-1}\bar{\mathbf{r}}_1 \end{bmatrix}.$$

Next, notice that $L^{(2)}$ has the form

$$L^{(2)} = \begin{bmatrix} I & 0 \\ 0 & L_{22}^{(2)} \end{bmatrix},$$

so it operates only on I_2. The second term $\epsilon L^{(-1)}\mathbf{r}_2$ is $O(\epsilon)$, so $L^{-1}\mathbf{r}_1 + \epsilon L^{-1}\mathbf{r}_2$ is $O(1)$ in I_1 and $O(\epsilon)$ in I_2 as claimed. When $\bar{D}^{-1}$ is applied to $L^{-1}\mathbf{r}_1 + \epsilon L^{-1}\mathbf{r}_2$, the infinitesimals cancel the infinities in $\bar{D}^{-1}$, and the result is a vector whose entries are $O(1)$. Finally, we apply L^{-T}, via back substitution, to obtain the solution.

As mentioned previously, generalizing this algorithm to the case of more than two layers is straightforward. If $[A_1, A_2]$ is not full rank, then the Cholesky factorization of B_2 will find zeros on the diagonal. In this case, the third layer is incorporated and so on until the coefficient matrix is full rank. So for each k, I_k is the set of indices not in $I_1 \cup \cdots \cup I_{k-1}$ such that $\bar{D}^{(k)}$ is not zero in these positions. The procedure described above is continued until we obtain the following factorization:

$$A_1 A_1^T + \epsilon A_2 A_2^T + \cdots + \epsilon^{l-1} A_l A_l^T = L \bar{D} L^T,$$

where

$$L = L^{(1)} M^{(1)} L^{(2)} M^{(2)} \cdots L^{(l-1)} M^{(l-1)} L^{(l)}$$

and $\bar{D}$ is a diagonal matrix whose diagonal entries are $O(1)$ in I_1, $O(\epsilon)$ in I_2, and so on. Here l is the first index such that the Cholesky factorization finds no zeros on the diagonal. To solve the LLS problem, we form

$$\Delta \mathbf{y} = L^{-T} \bar{D}^{-1} (L^{-1} \mathbf{r}_1 + \epsilon L^{-1} \mathbf{r}_2 + \cdots + \epsilon^{l-1} L^{-1} \mathbf{r}_l).$$

Notice that $L^{-1}\mathbf{r}_1$ has nonzero entries only in I_1, $\epsilon L^{-1}\mathbf{r}_2$ has nonzero entries in $I_1 \cup I_2$, and so on. The QR factorization argument used in the two-layer case easily extends to this case. Thus, a generalization of the discussion regarding the formation of and the structure of the right-hand side of the two-layer case applies here.

The following theoretical results hold for this procedure. In exact arithmetic, it returns the exact solution to the LLS problem. In finite precision arithmetic, if we assume that no errors are made in determining which Cholesky pivots are zero, then the computed solution satisfies an error bound of the form (1.2). We forego a rigorous

analysis as it is merely an application of standard forward error analysis of the Cholesky algorithm to a sequence of problems involving well-conditioned matrices.

Now that $\Delta\mathbf{y}^*$ has been computed, it is necessary to compute $\Delta\mathbf{s}^*$. Recall that the straightforward way to do this is the following:

$$\Delta\mathbf{s}^* = -A^T\Delta\mathbf{y}^*.$$

However, recall that the discussion in Chapter 2 states that this approach is numerically unstable, but since all quantities in a layer are of the same order of magnitude, computing $\Delta\mathbf{s}_k^* = A_k^T\Delta\mathbf{y}^{(k)}$ for $k = 1, \ldots, l$ can be done accurately. If $\Delta\mathbf{y}^{(k)}$ can be computed in such a way that $A_k^T(\Delta\mathbf{y}^* - \Delta\mathbf{y}^{(k)}) = \mathbf{0}$, where $\Delta\mathbf{s}_k$ and $\Delta\mathbf{y}^{(k)}$ depend only on $\mathbf{s}_1, \ldots, \mathbf{s}_k$, then $\Delta\mathbf{s}^*$ can be determined at each layer. This can, in fact, be done by using the partial $L\bar{D}L^T$ factorization to solve the system of equations at each layer. We again demonstrate the method for the two-layer case. The generalization is straightforward.

Recall that at the first layer, we have $A_1A_1^T = L^{(1)}\bar{D}^{(1)}(L^{(1)})^T$. Now partition $L^{(1)}$ and $\bar{D}^{(1)}$ as follows:

$$L^{(1)} = \begin{bmatrix} L_{11}^{(1)} & 0 \\ L_{21}^{(1)} & I \end{bmatrix} \text{ and } \bar{D}^{(1)} = \begin{bmatrix} \bar{D}_1 & 0 \\ 0 & 0 \end{bmatrix}.$$

The right-hand side of the first layer can be written as

$$A_1\mathbf{s}_1 = \mathbf{r}_1 = \begin{bmatrix} \mathbf{r}_{11} \\ \\ \mathbf{r}_{12} \end{bmatrix}.$$

Now $\Delta\mathbf{y}^{(1)}$ is computed as follows:

$$\Delta\mathbf{y}^{(1)} = \begin{bmatrix} (L_{11}^{(1)})^{-T}\bar{D}_1^{-1}(L_{11}^{(1)})^{-1}\mathbf{r}_{11} \\ \\ \mathbf{0} \end{bmatrix}.$$

The zero block in $\Delta\mathbf{y}^{(1)}$ can be explained by the relationship between Cholesky factorization and QR factorization, as previously discussed. The question that must now be answered is whether or not $\Delta\mathbf{y}^* - \Delta\mathbf{y}^{(1)}$ is in the nullspace of A_1^T.

In order to show the $A_1^T(\Delta\mathbf{y}^* - \Delta\mathbf{y}^{(1)}) = \mathbf{0}$, we again use the QR factorization $A_1^T = [Q_1, 0]\tilde{D}^{(1)}(L^{(1)})^T$. First, $\Delta\mathbf{y}^*$ is written in terms of the partitioned matrices and vectors shown above and the $\Delta\mathbf{y}^{(1)}$ component is removed. This leaves a vector of the form

$$\Delta\mathbf{y}^* - \Delta\mathbf{y}^{(1)} = \begin{bmatrix} (L_{11}^{(1)})^{-T}(L_{21}^{(1)})^T\mathbf{v} \\ -\mathbf{v} \end{bmatrix},$$

where $\mathbf{v}$ is the appropriate combination of the factors and the right-hand sides. We now multiply by A_1^T, one factor at a time. It is easy to see that multiplication by $(L^{(1)})^T$ leaves zeros in the I_1 entries of the resulting vector. Multiplication by $\tilde{D}^{(1)}$ has no effect, and multiplication by $[Q_1, 0]$ zeroes out the I_2 component. Thus, $A_1^T(\Delta\mathbf{y}^* - \Delta\mathbf{y}^{(1)}) = \mathbf{0}$. This approach extends to the general case in the obvious way.

As in the discussion of the method for finding $\Delta\mathbf{y}^*$, we forego any formal discussion of the stability of this computation. A straightforward error analysis will show that the entries of $\Delta\mathbf{s}^*$ are computed accurately with respect to the corresponding entries of $\mathbf{s}$, i.e., the entries of $\Delta\mathbf{s}^*$ satisfy an error bound of the form (3.12).

This concludes the discussion of the dual LLS problem. We now move on to the primal problem.

4.3 The Primal LLS Problem

The formulation of the primal LLS problem is quite similar to that of the dual LLS problem. Thus, the discussion will be somewhat shorter and contain less detail than that of the dual problem. The main differences between the two problems are the following: in the primal problem, we work with the nullspace of A rather than the range of A^T and D^{-1} instead of D. Also, the order of the partitions is opposite that for the dual problem. As with the dual problem, the LLS problem is defined by a sequence of nested subspaces. Let $V_p^P = N(A)$. For $k = p, p-1, \ldots, 1$, define

$$V_{k-1}^P = \{\text{minimizers } \Delta\mathbf{x} \text{ of } \left\| D_k^{-1/2}(\Delta\mathbf{x}_k + \mathbf{x}_k) \right\| \text{ subject to } \Delta\mathbf{x} \in V_k^P\}. \tag{4.4}$$

So $V_0^P \subset V_1^P \subset \cdots \subset V_p^P$, and the solution to the primal LLS problem is the unique element $\Delta\mathbf{x}^*$ of V_0^P. Like the dual LLS step, it is possible that $\Delta\mathbf{x}^*$ is uniquely determined at layer l, where $l > 0$. The Lagrange multiplier conditions for this sequence of constrained minimization problems are given by the following:

$$\begin{aligned} A\Delta\mathbf{x}^* &= \mathbf{0} \\ D_p^{-1}\Delta\mathbf{x}_p^* + D_p^{-1}\mathbf{x}_p &= A_p^T\boldsymbol{\lambda}_p \\ D_{p-1}^{-1}\Delta\mathbf{x}_{p-1}^* + D_{p-1}^{-1}\mathbf{x}_{p-1} &= A_{p-1}^T\boldsymbol{\lambda}_{p-1} \\ &\vdots \\ D_1^{-1}\Delta\mathbf{x}_1^* + D_1^{-1}\mathbf{x}_1 &= A_1^T\boldsymbol{\lambda}_1 \end{aligned} \tag{4.5}$$

As with the dual problem, these equations uniquely define the primal LLS step $\Delta\mathbf{x}^*$, and Vavasis and Ye showed that the primal LLS problem is also the limiting case of an ordinary weighted least-squares problem. As before, assume that D is the identity

matrix, and we introduce infinitesimals $\epsilon > 0$ into the problem. First the Lagrange multiplier conditions are rewritten in matrix-vector form, and the infinitesimals are introduced. This yields the following:

$$\begin{bmatrix} A_1^T \\ \epsilon A_2^T \\ \vdots \\ \epsilon^{p-1} A_p^T \end{bmatrix} \boldsymbol{\lambda} = \begin{bmatrix} \Delta \mathbf{x}_1^* + \mathbf{x}_1 \\ \Delta \mathbf{x}_2^* + \mathbf{x}_2 \\ \vdots \\ \Delta \mathbf{x}_p^* + \mathbf{x}_p \end{bmatrix}.$$

Multiplying both sides of the previous equation by A gives the primal LLS problem:

$$(A_1 A_1^T + \epsilon A_2 A_2^T + \cdots + \epsilon^{p-1} A_p A_p^T)\boldsymbol{\lambda} = A_1 \mathbf{x}_1 + A_2 \mathbf{x}_2 + \cdots + A_p \mathbf{x}_p. \tag{4.6}$$

Notice that the coefficient matrix is the same as that for the dual problem. Thus, we can use the $L\bar{D}L^T$ factorization computed for the dual case. However, the form of the right-hand side is different from that of the dual problem in the sense that it contains no infinitesimal factors. It is therefore necessary to use a different method to compute the solution.

For simplicity, we again demonstrate the algorithm for the two-layer case. Let $\mathbf{r}_1 = A_1 \mathbf{x}_1$ and $\mathbf{r}_2 = A_2 \mathbf{x}_2$. After the factorization is complete, i.e., once

$$A_1 A_1^T + \epsilon A_2 A_2^T = L^{(1)} M^{(1)} L^{(2)} \bar{D}^{(2)} (L^{(2)})^T (M^{(1)})^T (L^{(1)})^T$$

has been computed, we can determine $\boldsymbol{\lambda}$. Details of the calculation are excluded to avoid unsightly notation. In solving for $\boldsymbol{\lambda}$, the triangular substitution must be done on $\mathbf{r}_1$ and $\mathbf{r}_2$ separately. We label the results $\boldsymbol{\lambda}^{(1)}$ and $\boldsymbol{\lambda}^{(2)}$. Computing $\boldsymbol{\lambda}^{(1)}$ is straightforward, i.e.,

$$\boldsymbol{\lambda}^{(1)} = L^{-T} \bar{D}^{-1} L^{-1} \mathbf{r}_1,$$

where $L = L^{(1)}M^{(1)}L^{(2)}$. Higher-order infinitesimals are dropped from the result.

The computation of $\boldsymbol{\lambda}^{(2)}$ must be done more carefully. Notice that after the forward substitution and scaling, $(L^{(1)})^T\boldsymbol{\lambda}^{(2)}$ has been computed and is $O(1)$ in I_1 and $O(1/\epsilon)$ in I_2. In performing the back substitution, necessary $O(1)$ information will be lost, so $(L^{(1)})^T\boldsymbol{\lambda}^{(2)}$ is split into the sum of two vectors with the following form:

$$(L^{(1)})^T\boldsymbol{\lambda}^{(2)} = \begin{bmatrix} O(1) \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ O(1/\epsilon) \end{bmatrix}.$$

The back substitution is performed separately on the two terms to obtain $\boldsymbol{\lambda}_1^{(2)} + \frac{1}{\epsilon}\boldsymbol{\lambda}_2^{(2)}$. Thus, $\boldsymbol{\lambda} = \boldsymbol{\lambda}^{(1)} + \boldsymbol{\lambda}_1^{(2)} + \frac{1}{\epsilon}\boldsymbol{\lambda}_2^{(2)}$. Again, higher-order infinitesimals are dropped.

It is now possible to compute $\Delta\mathbf{x}_1^*$ and $\Delta\mathbf{x}_2^*$ as follows:

$$\begin{aligned} \Delta\mathbf{x}_1^* &= A_1^T(\boldsymbol{\lambda}^{(1)} + \boldsymbol{\lambda}_1^{(2)}) + \frac{1}{\epsilon}A_1^T\boldsymbol{\lambda}_2^{(2)} - \mathbf{x}_1, \text{ and} \\ \Delta\mathbf{x}_2^* &= \epsilon A_2^T(\boldsymbol{\lambda}^{(1)} + \boldsymbol{\lambda}_1^{(2)}) + A_2^T\boldsymbol{\lambda}_2^{(2)} - \mathbf{x}_2. \end{aligned}$$

Using arguments similar to those used in the computation of $\Delta\mathbf{s}^*$, i.e., writing out $\boldsymbol{\lambda}_2^{(2)}$ explicitly and using the QR factorization of A_1^T, it is easy to see that $A_1^T\boldsymbol{\lambda}_2^{(2)} = \mathbf{0}$. (In finite-precision arithmetic, this quantity must be set to $\mathbf{0}$.) So $\Delta\mathbf{x}_1^*$ can be computed accurately. Since the first layer does not contribute to the second, finding $\Delta\mathbf{x}_2^*$ is straightforward, and higher-order infinitesimals are dropped. It is not difficult to show that the solution computed by the method just described will satisfy an error bound of the form (3.13). The discussion of this method extends to the general case in a straightforward manner.

Clearly, using the same $L\bar{D}L^T$ factorization for the primal and the dual LLS problems makes the overall computation of the LLS step more efficient than if two

factorizations were required. There are other ways to increase the efficiency of the algorithm, and these are discussed in the following section.

4.4 Efficiency of the LLS Algorithm

There are two observations that allow simplifications in the implementation of the algorithm described in the previous sections. The first pertains to the matrix that performs the block Gaussian elimination at kth layer $M^{(k)}$. The structure of the problem and the computations are such that it is not necessary to form $M^{(k)}$, nor is it necessary to store any information in addition to the Cholesky factors. For example, in the two-layer case, $W_{21}^{(2)}$ and the appropriate identity matrix make up the blocks of $M^{(1)}$. Recall that

$$W^{(2)} = (L^{(1)})^{-1} A_2 A_2^T (L^{(1)})^{-T}.$$

So $W_{21}^{(2)}$ can be reconstructed from L and A, which have already been stored. Thus, whenever $M^{(1)}$ is needed for a computation it is necessary only to obtain the appropriate pieces of $(L^{(1)})^{-1}$ and A_2 and perform the desired operation. It should be noted that the only computations for which $M^{(1)}$ is required are matrix-vector multiplications that arise during the triangular substitution phases of finding the solution. It is not needed for incorporating later layers (which would involve matrix-matrix multiplications) because it introduces only higher-order infinitesimal terms that will be ignored. So the increase in the amount of work arising from operations involving $M^{(1)}$ is not significant. Not having to store any information also simplifies storing the

lower-triangular factors computed at each layer. For example, $L^{(2)}$ can be "inserted" into the columns of $L^{(1)}$ corresponding to the identity.

In addition to conserving space by not saving each $M^{(k)}$, it is possible to increase efficiency by taking advantage of the sparsity and the structure of the LLS problem. This is done in the following way. A preliminary step is to select a good sparse order, such as nested dissection or symmetric minimum degree, for the entire matrix AA^T [12]. This order is chosen to minimize the fill-in that occurs during the computation of the Cholesky factor of AA^T. Notice that the nonzero pattern of each of the submatrices $A_k A_k^T$ is a subset of that of AA^T, so the sparse ordering will also be suitable for the subproblems. Recall that in the interior point context, A is fixed from iteration to iteration, so this preliminary step can be done once for the whole algorithm. Also, all of the successive Cholesky factors $L^{(1)}, L^{(2)}, \ldots, L^{(l)}$ fit together inside the data structure reserved for the Cholesky factor of AA^T. Therefore, all required space can be allocated at once. Finally, it is clear that taking advantage of sparsity in the actual computations greatly increases the speed since it decreases the number of required operations.

To summarize, the proposed algorithm for solving the LLS problem requires the same amount of space and the same sparse data structures, all of which only need to be allocated once at the beginning of the algorithm, as would the sparse Cholesky algorithm applied to the original problem. Also, since the final result is an $L\bar{D}L^T$ factorization of the original problem, computed one piece at a time, this algorithm requires approximately the same number of floating point operations as the sparse Cholesky methods applied to the original problem.

4.5 Description of Implementation

The core of the implementation of the LLS algorithm is the sparse Cholesky code written by Ng and Peyton of Oak Ridge National Laboratories. (This includes the minimum degree ordering code of Liu.) Recall that the LLS algorithm is somewhat different from a straightforward Cholesky factorization. For this reason, some slight modifications are required. In addition, we have added a MATLAB interface in order to facilitate the manipulation of data according to the partitions. These modifications are described here.

The MATLAB program takes as input the constraint matrix A, a vector containing the diagonal entries of D, and the right-hand side vectors $\mathbf{x}$ and $\mathbf{s}$. After the sparse ordering is determined and the data structures are allocated, the MATLAB function extracts the data of the first layer. In computing $A_1 D_1 A_1^T = L^{(1)} \bar{D}^{(1)} (L^{(1)})^T$, we see the first modification of the Oak Ridge code. The original code returns the Cholesky factor and a flag that indicates whether or not the factorization was successful, i.e., whether or not the matrix to be factored is positive definite. The modified code returns instead a scaled Cholesky factor, a vector containing the diagonal entries, and a 0-1 vector that indicates which diagonal entries are zero. Notice that this vector corresponds to the index set discussed in §4.2. In finite-precision arithmetic, the zeros on the diagonal are not exact, and a numerical tolerance is used to determine which diagonal entries are too small. This tolerance will be adjusted according to the results of computational experiments. The only additional work incurred by these modifications is that of scaling the Cholesky factor by the diagonal entries.

After the factorization for the first layer is computed, $\Delta \mathbf{s}_1$ is determined in the manner described in §4.2. Recall that this requires finding $\Delta \mathbf{y}^{(1)}$ via triangular substitution. This is accomplished by a slightly modified version of the Oak Ridge triangular solvers, written to be compatible with the Cholesky code. The solvers handle only one right-hand side at a time. While this is sufficient for the first layer, recall that in most instances in the LLS application, it is necessary to perform the substitution on multiple right-hand sides. This modification is straightforward. Another small modification is necessary. Since the factorization is actually an $L\bar{D}L^T$ factorization as opposed to a Cholesky factorization, it is necessary to add a scaling step between the forward substitution and the back substitution. This can be done quickly and easily in the MATLAB code. In fact, it is more convenient than scaling within the solvers for two reasons. First, the scaling is not always necessary, and secondly, it is often necessary to scale by only some of the submatrices of D.

If A_1 is not full rank, the second layer is incorporated, and a process similar to the one above is repeated as necessary. When the factorization is complete, both the dual and the primal problems are solved according to the descriptions given in §4.2 and §4.3. (It should be noted that this is a slightly simplified description of the triangular substitution process.) Notice that triangular substitution involving the Cholesky factors obtained at each layer is interleaved with block Gaussian elimination. Recall that the Gaussian elimination is accomplished by a matrix-vector multiplication. This multiplication step can be performed by a FORTRAN subroutine, or it can be done in MATLAB, which is capable of taking advantage of sparsity. While the MATLAB version may be easier to implement, the FORTRAN version may be faster.

This is a trade-off that must be studied further. After the computations are complete, the MATLAB program returns the LLS step $(\Delta\mathbf{x}^*, \Delta\mathbf{y}^*, \Delta\mathbf{s}^*)$.

The MATLAB interface is used mainly for manipulating data throughout the algorithm, while all of the major computations are done by the Oak Ridge sparse Cholesky code. As we saw in the previous paragraphs, some modifications to the code are necessary so that it fits into the LLS setting, but those changes are few, minor, and easy to implement. Thus, the LLS algorithm is not only efficient in theory, but it allows the use of existing code which is already known to be efficient and robust in practice.

4.6 Numerical Results

We have begun computational experiments using the implementation of the LLS algorithm described in the previous section. The test problems are individual layered least-squares steps, where the algorithm is given the constraint matrix A, a vector containing the diagonal entries of D, and the right-hand side vectors $\mathbf{x}$ and $\mathbf{s}$, and the output is the search direction $(\Delta\mathbf{x}, \Delta\mathbf{y}, \Delta\mathbf{s})$. The tests are done in this way because the algorithm currently stands alone, i.e., it has not yet been implemented as part of an interior point method. Recall that the LLS problem can also be solved by using the COD algorithm with large gaps. Since this algorithm has strong theoretical results and has performed well in practice, it is used as the standard. Thus, the results obtained using the LLS algorithm will be compared to those obtained using the COD algorithm.

Table 4.1: The entries in this table show the accuracy of the search directions computed by the LLS algorithm. The error in $\Delta\mathbf{y}$ is given in the 2-norm, while the maximum error over all entries i is reported for $\Delta\mathbf{s}$ and $\Delta\mathbf{x}$. The exact steps are computed by the COD algorithm.

	approximate gap in $\mathbf{s}$	$\frac{\|\Delta\mathbf{y}^* - \Delta\hat{\mathbf{y}}\|}{\|\mathbf{s}\|}$	$\frac{\|\Delta s_i^* - \Delta\hat{s}_i\|}{s_i}$	$\frac{\|\Delta x_i^* - \Delta\hat{x}_i\|}{x_i}$
2 layers	1×10^4	1.2×10^{-16}	3.7×10^{-13}	5.0×10^{-11}
	1×10^8	7.4×10^{-17}	2.4×10^{-13}	2.4×10^{-13}
	1×10^{16}	6.1×10^{-17}	4.0×10^{-13}	4.0×10^{-13}
5 layers	1×10^4	6.5×10^{-17}	1.3×10^{-13}	2.1×10^{-11}
	1×10^8	6.5×10^{-17}	1.3×10^{-14}	1.4×10^{-14}
	1×10^{16}	4.0×10^{-17}	8.9×10^{-15}	8.4×10^{-15}

Table 4.1 shows the results from one set of tests. In these tests, A is a 23×136 RNAI matrix. The vectors $\mathbf{x}$ and $\mathbf{s}$ are chosen so that $XS\mathbf{e} = \mu\mathbf{e}$, where $\mu > 0$ is given. These vectors are also chosen in such a way that there are either two or five layers (with gaps of various sizes between the layers), and $D = \mu S^{-1}X$, as in §4.2. The gap introduced for the use of the COD algorithm is of size $1/\epsilon$, where ϵ is machine precision. Recall that we are trying to demonstrate that the LLS algorithm computes search directions that satisfy error bounds of the form (1.2), (3.12), and (3.13). Thus, the corresponding errors are shown in the table, where the exact solutions are those computed by the COD algorithm. Notice that the search directions have been computed with high relative accuracy, as predicted by the error bounds, despite ill-conditioning in the weight matrix. These preliminary results are encouraging for future experiments involving interior point methods.

Other computational experiments are in the development stages. We are currently

in the process of incorporating the LLS algorithm into the Mizuno, Todd, and Ye algorithm described in §3.4.1. The results will be an LIP-like interior point algorithm. Many interior point implementations have various parameters that may affect the accuracy to which the search directions are computed, so we will be better able to assess the performance of the algorithm in applications. However, there are still some issues that must be resolved with respect to the inclusion of this algorithm in an interior point method. These and other relevant questions are discussed in Chapter 5.

Chapter 5

Summary and Future Research

This thesis has addressed stability and efficiency issues for three closely related problems:

- solution of weighted least-squares problems with ill-conditioned weight matrices,
- computation of search directions in interior point methods for linear programming, and
- solution of the layered least-squares problem that arises in the LIP Algorithm of Vavasis and Ye.

We now summarize the main results and present questions that are open for future research.

5.1 Weighted Least Squares

The weighted least-squares problem in question is given by:

$$\min_{\mathbf{y} \in R^n} \| D^{-1/2} (A\mathbf{y} - \mathbf{b}) \|,$$

where $D \in \mathbb{R}^{m \times m}$ is a diagonal, positive definite, ill-conditioned matrix; $A \in \mathbb{R}^{m \times n}$ has full column rank; $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^m$. Such a problem is a special case of an equilibrium system, which arises in such applications as optimization, finite elements, structural analysis, and electrical networks. Because D is ill conditioned, the standard methods for solving least-squares problems may not find an accurate solution. We have employed a type of complete orthogonal decomposition to solve this type of problem. The COD algorithm is based on QR factorization, and column pivoting proves to be the key to the stability of the algorithm. Recall that the primary concern is with numerical problems caused by the ill-conditioning of the weight matrix, so by stable, we mean that the algorithm computes a solution that satisfies and error bound of the form (1.2). By examining the structure of the linear system of equations within the context of the algorithm, we find that a standard column pivoting strategy orders the weights in such a way that the ill-conditioning can be "offset".

The (forward) error analysis of the COD algorithm uses a particular norm bound, proved independently by a number of researchers, to derive an error bound that is independent of the ill-conditioned weight matrix. The only previously known stable algorithm is from [41] but is much more complicated and also requires more flops than the COD method. Other previous algorithms do not satisfy a forward error bound of

the form we seek. In contrast to the bound for the COD algorithm, these bounds do depend on the weight matrix. It is also not difficult to illustrate that these algorithms do not satisfy stability bounds strong enough to guarantee an accurate solution. In fact, it is possible to construct small examples for which the exact solutions can be determined by observation, but many algorithms will compute inaccurate solutions. (In fact, some find solutions with no significant digits of accuracy.) On the other hand, the COD algorithm computes a solution with an error on the order of machine precision, as expected.

There is still much work to be done on the COD algorithm. Chapter 2 contains a forward error analysis of the algorithm. The final bound, which is (2.19), involves some very large factors. These factors appear to be an artifact of the forward analysis rather than a feature of the algorithm. The alternative is a backward error analysis. For many other problems in numerical linear algebra, backward error analysis is successful in ultimately producing the best known forward error bounds [16]. As pointed out in §2.4, the straightforward approach to backward error analysis for weighted least squares could not yield the kind of forward error bound we seek. So the question that remains is whether or not it is possible to do a specialized backward error analysis of this algorithm and whether such an analysis yield better forward bounds. It appears that such a backward error analysis must be restricted to some special class of structured perturbations to A.

Another question open for research concerns sparsity. In the applications mentioned in §1.2, the coefficient matrix A is typically sparse. The COD Algorithm, however, is based on dense methods. Particularly for large problems, it would be

beneficial to take advantage of sparsity. Gilbert, Schreiber, and Vavasis began work on a sparse implementation, but this proved to be difficult for even the simplest model problem. While there is still hope for a sparse implementation, there are more promising approaches to solving the problem. One approach is to use an iterative method. It is well known that iterative methods are more efficient than direct methods for large sparse problems, but the usual conjugate gradient algorithm fails to converge when D is ill conditioned. Bobrovnikova and Vavasis have been working on a modified conjugate gradient method in which they take steps to restore the conjugacy of the search directions and the orthogonality of the residuals, both of which are lost because D is ill conditioned [6]. Experimental results are promising. The other approach to handling sparsity occurs in the context of interior point methods and will be discussed later.

The final open question for the COD Algorithm is that of generalizing it to the case where the weight matrix is positive definite and symmetric, but not necessarily diagonal. While this occurs in all of the applications mentioned mentioned in §1.2, it is discussed only in the context of interior point methods in the next section.

5.2 Interior Point Methods

The most natural setting for the weighted least-squares problem described in the previous section is interior point methods for linear programming. Each iteration of an interior point method requires the solution of a system of equations that can be expressed as a such a weighted least-squares problem, and the weight matrix is always

ill conditioned close to the solution of the optimization problem. Ill-conditioning in the weight matrix can also arise as a result of near-degeneracy in the linear programming instance. In primal-dual path-following algorithms, the dual step $\Delta\mathbf{y}$ is obtained by solving the weighted least-squares problem. It is also necessary to compute the primal step and the change in the dual slack variables. As discussed in §3.3, numerical problems can arise here also. The COD Algorithm can be extended to compute $\Delta\mathbf{x}$ and $\Delta\mathbf{s}$ accurately in an entrywise relative sense. The forward error bounds are given by (3.12) and (3.13). The stability analysis makes no assumptions about nondegeneracy in the linear programming instance. On the other hand, many other authors do make assumptions that correspond to a nondegeneracy assumptions about the underlying optimization problem. Thus, the COD Algorithm is more general than others in this respect, and it is reasonable to expect that it will have less difficulty when there is degeneracy or near-degeneracy in the underlying optimization problem.

In order to test the COD algorithm in practice, it has been integrated into two different interior point algorithms. The first is a predictor-corrector method proposed by Mizuno, Todd, and Ye. The other method is Mehrotra's predictor-corrector method, as implemented by Zhang in LIPSOL. These same interior point methods were also implemented using Cholesky factorization to solve the linear systems, and these algorithms were run on a number of near-degenerate linear programming problems. As asserted, the results indicate that it is advantageous in many cases to use the COD implementation rather than the Cholesky.

Recall from the discussion in §3.5 that the applicability of the extended COD

algorithm in nonlinear programming applications has not yet been determined. If it is indeed applicable, it would be necessary to modify it to also compute directions of negative curvature when necessary. It would also be interesting to compare the results to those obtained using the various modified Cholesky algorithms that are now commonly used in interior point methods for nonlinear programming problems.

We have studied how the COD algorithm fits into and performs in the interior point setting, but it has not yet been studied in the context of other applications, namely finite element methods, electrical networks, and structural analysis. As with the interior point study, it would be necessary to tune the implementation and stability analysis to the application and to compare numerical results to those obtained using existing methods.

5.3 Layered Least Squares

Vavasis and Ye recently proposed the LIP method for linear programming. This is a primal-dual interior point method with a new step, the layered least squares (LLS) step. This step requires solving a problem that can be regarded as a weighted least-squares problem with infinite weight gaps. As with traditional path-following steps, the weight matrix can be ill conditioned. There are both primal and dual formulations of the LLS problem, but the approaches to solving them are similar. The original problem is broken down into smaller, constrained least-squares problems with well-conditioned weight matrices. The weights of the first layer are "infinitely" higher than those of the second layer, which are "infinitely" higher than those of

the third layer and so on. The procedure is as follows. The weighted least-squares problem of the first layer is solved as an unconstrained problem. If necessary, the following process is repeated until the unique solution is reached. The weighted least-squares problem of layer i is solved, with the constraint that the solution also solves the problem of layer $i-1$.

Vavasis and Ye did not propose a particular algorithm to solve the LLS problem. The COD algorithm of Chapter 2 can be used to solve this problem, but it is not well suited for large sparse problems, so we have proposed an algorithm based on Cholesky factorization. The development of the algorithm involves arithmetic on infinitesimals, but these infinitesimals do not actually appear in the algorithm. Since the problem is broken down into smaller least-squares problems with well-conditioned weight matrices and then solved one layer at a time, it is easy to show that the errors in the search directions satisfy (1.2), (3.12), and (3.13).

The LLS algorithm has been tested in some small-scale computational experiments. The implementation uses a slightly modified version of the sparse Cholesky code written by Ng and Peyton [29] of Oak Ridge National Laboratories. The algorithm must still be tested with actual interior point algorithms. Because linear programming problems do not usually give rise to problems that are as nicely constructed as those used in this work, it may also be necessary to experiment with different tolerances for determining which Cholesky pivots are zero.

Another question that remains with regard to incorporating the LLS algorithm into interior point methods is that of determining the layers. Recall that the gap g between the layers is defined in terms of χ_A and $\bar{\chi}_A$, so these constants must be

known *a priori*. However, there is no known way to compute them in polynomial time, nor is there a satisfactory way of estimating them. Megiddo, Mizuno, and Tsuchiya propose an alternate approach to determining the gaps [24]. Rather than defining g in terms of A, they determine all possible partitions as g ranges from 1 to infinity. The candidates for the layered least-squares step are computed, and the step which reduces μ the most is chosen. By computing the layered least-squares step in this way, there is no need to know χ_A and $\bar{\chi}_A$. However, the various approaches must still be tested in practice.

There are a number of sparse ordering heuristics that can be used for the LLS algorithm. The question is which one is best. This is a question that has been studied extensively with respect to sparse Cholesky factorization. In particular, Rothberg and Hendrickson have done a systematic study of ordering heuristics for interior point algorithms [34], the results of which could be used to improve the LLS algorithm.

5.4 Closing Remarks

There are two themes that appear over and over again in the work of this thesis. The first pertains to structure. The problems discussed involve computations with highly ill-conditioned matrices. This ill-conditioning can cause numerical problems in standard algorithms, thus causing them to compute inaccurate solutions. However, the ill-conditioning is structured, and it is possible to take advantage of this structure. For example, the weighted least-squares problem that arises in the LLS problem can be broken down into smaller problems with well-conditioned weight matrices.

Standard algorithms can then be used to solve these smaller problems. So taking advantage of the structured ill-conditioning allows the use of standard algorithms which might not otherwise be effective. It also yields nice theoretical and algorithmic results that might not be immediately obvious from looking at the problems. Thus, it is worthwhile to make the effort to identify and take advantage of any structure present in a problem.

Another theme that occurs throughout is that of modifying standard algorithms to solve problems such as those mentioned in the previous paragraph. These algorithms have been studied extensively, are well understood, and are usually easy to implement. Often they work quite well, even on problems with ill-conditioned matrices, though it is not always clear why. They have been modified so that they are more efficient and take advantage of sparsity, and robust implementations of these algorithms are available. In summary, a great deal of work has gone into these standard algorithms, and it would not be beneficial to ignore all that has already been done. It is worthwhile to take the time to see if they can be modified or "patched" to solve problems such as the ones described in this thesis. For example, the COD algorithm is based on QR factorization and employs some standard techniques in a special way to handle the ill-conditioning that arises. It is likely that similar ideas will be equally effective for other problems and other algorithms.

BIBLIOGRAPHY

[1] J. L. Barlow. Stability analysis of the G-algorithm and a note on its application to sparse least squares problems. *BIT*, 25:507–520, 1985.

[2] D. A. Bayer and J. C. Lagarias. The nonlinear geometry of linear programming. I. Affine and projective scaling trajectories. *Transactions of the AMS*, 314:499–526, 1989.

[3] D. A. Bayer and J. C. Lagarias. The nonlinear geometry of linear programming. II. Legendre transform coordinates and central trajectories. *Transactions of the AMS*, 314:527–581, 1989.

[4] D. A. Bayer and J. C. Lagarias. The nonlinear geometry of linear programming. III. Projective Legendre transform coordinates and Hilbert geometry. *Transactions of the AMS*, 320:193–225, 1990.

[5] Å. Björck and I. S. Duff. A direct method for the solution of sparse linear least squares problems. *Linear Alg. and Appl.*, 34:43–67, 1980.

[6] E. Bobrovnikova and S. Vavasis. Conjugate gradient method for weighted and layered least squares. In progress, 1996.

[7] L. O. Chua, C. A. Desoer, and E. S. Kuh. *Linear and Nonlinear Circuits.* McGraw-Hill, New York, 1987.

[8] T. F. Coleman and J. Liu. An interior Newton method for quadratic programming. Technical Report CTC93TR153, Advanced Computing Research Institute, Cornell University, Ithaca, NY, 1993.

[9] I. I. Dikin. On the speed of an iterative process. *Upravlyaemye Sistemi*, 12:54–60, 1974.

[10] A. L. Forsgren. On linear least-squares problems with diagonally dominant weight matrices. Report TRITA-MAT-1995-OS2, Optimization and Systems Theory, Department of Mathematics, Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1995.

[11] A. L. Forsgren, P. E. Gill, and J. R. Shinnerl. Stability of symmetric ill-conditioned systems arising in interior methods for constrained optimization. Report NA 94-1, Department of Mathematics, University of California, San Diego, CA, 1994. To appear in *SIAM J. Matrix Anal. Appl.*, 1995.

[12] A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice–Hall, Englewood Cliffs, N.J., 1981.

[13] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, London, 1981.

[14] P. E. Gill, M. A. Saunders, and J. R. Shinnerl. On the stability of the Cholesky factorization for symmetric quasi-definite systems. To appear in *SIAM J. Matrix Anal. Appl.*, 1995.

[15] G. Golub. Numerical methods for solving linear least squares problems. *Numer. Math.*, 7:206–216, 1965.

[16] G. H. Golub and C. F. Van Loan. *Matrix Computations, 2nd Edition.* Johns Hopkins Univ. Press, Baltimore, 1989.

[17] C. C. Gonzaga. Path-following methods for linear programming. *SIAM Review*, 34:167–224, 1992.

[18] N. Gould. On the accurate determination of search directions for simple differentiable penalty functions. *IMA Journal of Numerical Analysis*, 6:357–372, 1986.

[19] M. Gulliksson. Backward error analysis for the constrained and weighted linear least squares problem when using the weighted QR factorization. *SIAM J. Matrix Anal. Appl.*, 16:675–687, 1995.

[20] M. Gulliksson and P. -Å. Wedin. Modifying the QR decomposition to constrained and weighted linear least squares. *SIAM J. Matr. Anal. Appl.*, 13:1298–1313, 1992.

[21] N. Higham. *Accuracy and Stability of Numerical Algorithms.* SIAM Press, 1996.

[22] C. L. Lawson and R. J. Hanson. *Solving Least Squares Problems.* Prentice Hall, Englewood Cliffs, NJ, 1974.

[23] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior Point and Related Method*, pages 131–158. Springer, New York, 1989.

[24] N. Megiddo, S. Mizuno, and T. Tsuchiya. A modified layered-step interior-point algorithm for linear programming. Technical Report RJ 10028, IBM Research Division, Almaden Research Center, San Jose, CA, 1996.

[25] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. Optimization*, pages 575–601, 1992.

[26] S. Mizuno, M. J. Todd, and Y. Ye. A surface of analytic centers and infeasible interior-point algorithms for linear programming. *Mathematics of Operations Research*, 20:135–162, 1995.

[27] W. Murray. Analytical expressions for the eigenvalues and eigenvectors of the Hessian matrices of barrier and penalty functions. *J. Optimization Theory and Applications*, 7:189–196, 1971.

[28] S. G. Nash and A. Sofer. A barrier method for large-scale constrained optimization. *ORSA Journal on Computing*, 5:40–53, 1993.

[29] E. Ng and B. W. Peyton. Block sparse Cholesky algorithms on advanced uniprocessor computers. *SIAM J. Scientific Computing*, 14:1034–1056, 1993.

[30] D. P. O'Leary. On bounds for scaled projections and pseudoinverses. *Linear Algebra and its Applications*, 132:115–117, 1990.

[31] C. C. Paige. Fast numerically stable computations for generalized least squares problems. *SIAM J. Numer. Anal.*, 16:165–171, 1979.

[32] G. Peters and J. H. Wilkinson. The least squares problem and pseudo-inverses. *The Computer Journal*, 13:309–316, 1970.

[33] M. J. D. Powell and J. K. Reid. On applying Householder transformations to linear least squares problems. In *Proc. IFIP Congress, 1968*, pages 122–126, Amsterdam, 1969. North-Holland.

[34] E. Rothberg and B. Hendrickson. Sparse matrix ordering methods for interior point linear programming. Submitted to INFORM J. Comput., 1996.

[35] G. Sonnevend. An "analytic center" for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming. In A. Prekopa, J. Szelezsan, and B. Strazicky, editors, *System Modelling and Optimization: Proceedings of the 12th IFIP–Conference held in Budapest, Hungary, September 1985*, volume 84 of *Lecture Notes in Control and Information Sciences*, pages 866–876. Springer Verlag, Berlin, Germany, 1986.

[36] G. W. Stewart. On scaled projections and pseudoinverses. *Linear Algebra and its Applications*, 112:189–193, 1989.

[37] G. Strang. A framework for equilibrium equations. *SIAM Review*, 30:283–297, 1988.

[38] M. J. Todd. A Dantzig-Wolfe-like variant of Karmarkar's interior-point linear programming algorithm. *Operations Research*, 38:1006–1018, 1990.

[39] C. F. Van Loan. On the method of weighting for equality-constrained least-squares problems. *SIAM J. Numer. Anal.*, 22:851–864, 1985.

[40] S. A. Vavasis. Stable finite elements for problems with wild coefficients. Technical Report TR-93-1364, Department of Computer Science, Cornell University, Ithaca, N. Y., 1993. To appear in *SIAM J. Num. Anal.*

[41] S. A. Vavasis. Stable numerical algorithms for equilibrium systems. *SIAM J. Matrix Anal. Appl.*, 15:1108–1131, 1994.

[42] S. A. Vavasis and Y. Ye. An accelerated interior point method whose running time depends only on A. Technical Report 93-1391, Department of Computer Science, Cornell University, Ithaca, NY, 1993. To appear in *Math. Progr.* under the title "A primal-dual interior point method whose running time depends only on the constraint matrix".

[43] M. H. Wright. Determining subspace information from the Hessian of a barrier function. Technical Report 92–02, AT&T Bell Laboratories Numerical Analysis Manuscript, 1992.

[44] M. H. Wright. Interior methods for constrained optimization. In *Acta Numerica 1992*. Cambridge University Press, Cambridge, 1992.

[45] S. J. Wright. *Primal-dual Interior-point Methods*. SIAM. to appear in 1997.

[46] S. J. Wright. Stability of linear algebra computations in interior-point methods for linear programming. Technical Report MCS-P446-0694, Mathematics and Computer Science Division, Argonne National Laboratory, Chicago, IL, 1994.

[47] Y. Zhang. Solving large-scale linear programs by interior-point methods under the MATLAB environment. Technical Report TR96-01, Department of Mathematics and Statistics, University of Maryland Baltimore County, Baltimore, MD, 1996.