ABSTRACT

Title of Dissertation: MATRIX FACTORIZATIONS, TRIADIC MATRICES, AND MODIFIED CHOLESKY FACTORIZATIONS FOR OPTIMIZATION

Haw-ren Fang, Doctor of Philosophy, 2006

Dissertation directed by: Professor Dianne P. O'Leary
Department of Computer Science

This thesis focuses on the Cholesky-related factorizations of symmetric matrices and their application to Newton-type optimization.

A matrix is called triadic if it has at most two nonzero off-diagonal elements in each column. Tridiagonal matrices are a special case of these. We prove that the triadic structure is preserved in the Cholesky-related factorizations We analyze its numerical stability and also present our perturbation analysis.

Newton-like methods solve nonlinear programming problems whose objective function and constraint functions are twice continuously differentiable. At each iteration, a search direction is computed by solving a linear symmetric system $Ax = b$. When A is not positive definite, the computed search direction may not be a descent direction.

Modified Newton methods add a perturbation E to A, so that $A+E$ is positive definite, where E is symmetric positive semidefinite. We study the modified Newton methods in the literature, and develop other stable and efficient algorithms. One of them exploits the merits of triadic structure.

We apply our modified Newton methods to the Euclidean distance matrix completion problem (EDMCP). Given n points in Euclidean space, the Euclidean distance matrix (EDM) is the real symmetric matrix with (i, j) entry being the square of the Euclidean distance between ith and jth points. Given a partial Euclidean distance matrix (i.e., some entries are not specified), the EDMCP is to find a EDM that includes the specified entries. We tackle the EDMCP by transforming it into a global optimization problem and applying modified Newton methods. We also develop a dimensional relaxation method for global minimization and test it on sample problems including protein structure prediction.

MATRIX FACTORIZATIONS, TRIADIC MATRICES, AND MODIFIED CHOLESKY FACTORIZATIONS FOR OPTIMIZATION

by

Haw-ren Fang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Committee:

Professor Dianne P. O'Leary, Chair
Professor Jeffery Cooper
Professor Ramani Duraiswami
Professor Howard Elman
Professor Amitabh Varshney

ACKNOWLEDGEMENTS

First and foremost of all, I am very grateful and deeply indebted to my advisor, Dianne P. O'Leary, for her thoughtful guidance, invaluable discussions, stimulating suggestions, and encouragement in research. She was always available when I needed her advices. Her contribution to all sections of this thesis is immeasurable. But for her this work could not have been completed.

I would like to thank the Department of Computer Science, University of Maryland for giving me permission to commence this thesis in the first instance.

I am appreciative of the financial support by National Science Foundation that made this work possible.

I am grateful to Jeffery Cooper, Ramani Duraiswami, Howard Elman, and Amitabh Varshney, for serving on my dissertation committee and their constructive comments.

With much pleasure I would like to express my gratitude to Che-Rung Lee for many very helpful discussions in several aspects of this work.

I would also like to take this opportunity to recognize several other people for their helpfulness during the preparation of this thesis. I thank Nick Higham for his encouragement in working on triadic matrices and stability analysis, in Chapters 2 and 3, respectively. I am grateful to Simon Schurr for discussing the convergence of interior point methods in convex programming for Chapter 4, and to Betty Eskow for constructive discussions on modified Cholesky algorithms for Chapter 5 and making her SE99 code available to us. For Chapter 6 my thanks go to David Fushman for sharing his expertise in molecular biology, and to Clyde Kruskal for a discussion on an NP-complete problem related to distance matrix completion.

Finally I wish to express my sincere gratitude to my parents, Ing-Teng Fang and Bi-Lan Lee. Only with their endless care and support I could concentrate on and complete this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

This thesis focuses on the Cholesky-related factorizations and their application to Newton-type optimization. The contents include the analysis of the factorization algorithms applied to the symmetric triadic matrices, the backward stability analysis, their use in computing Newton-like directions, and an application to distance matrix completion, as briefly described in the following sections.

1.1 Sparsity of Matrix Factorizations

Matrix factorizations have wide applications in numerical linear algebra, in solving linear systems, computing inertia, and rank estimation, and sparsity is an important consideration. In many cases, a factorization algorithm could ruin the sparsity and result in inefficiency of time and memory. For example, the matrix

$$\begin{bmatrix} 1 & -1 & -1 & \cdots & -1 \\ -1 & 2 & & & \\ -1 & & -2 & & \\ \vdots & & & \ddots & \\ -1 & & & & 2(-1)^n \end{bmatrix} \in R^{n\times n}$$

has $3n-2$ nonzero elements, but its LU factorization and LDL^T factorization are full (i.e., all entries nonzero) due to fill-in. Without pivoting, the LU factorization of a band matrix and the Cholesky factorization of a symmetric band matrix preserve the band structure and thereby the sparsity. However, pivoting can ruin the sparsity. For example, the LU factorization with complete pivoting destroys the sparsity of the tridiagonal matrix

$$\begin{bmatrix} 1 & 1 & & & & & \\ 1 & 1 & 1 & & & & \\ & x_1 & 1 & 1 & & & \\ & & 1 & 1 & 1 & & \\ & & & x_2 & 1 & \ddots & \\ & & & & \ddots & \ddots & \\ & & & & & x_{(n-1)/2} & 1 \end{bmatrix} \in R^{n\times n},$$

where $x_1 > x_2 > \ldots x_{(n-1)/2} > 1$ with n odd. The first $(n-1)/2$ pivots chosen in sequence are $x_1, x_2, \ldots, x_{(n-1)/2}$. The kth column of L has $k+2$ nonzero elements for $k = 1, \ldots, (n-1)/2$. Hence, L has $\Theta(n^2)$ nonzero elements.

A matrix A is called *triadic* if it has at most two nonzero off-diagonal elements. Tridiagonal matrices are a special case of these. In Chapter 2, we study the LL^T, LDL^T and LBL^T factorizations for symmetric matrices. We prove that if A is triadic then the triadic structure of these factorizations is preserved for any diagonal pivoting strategy applied [5, 10, 12, 13, 14]. Therefore, the required memory is $O(n)$ for factorization of $n \times n$ symmetric triadic matrices, whereas $O(n^2)$ is required for general symmetric matrices. The required time for factorization is $O(n^3)$ for the full symmetric matrices, whereas it is between $O(n^2)$ and $O(n)$ for symmetric triadic matrices, depending on the pivoting strategy applied.

We also give sharper bounds on growth factors using the triadic structure, which implies better numerical stability. The perturbation analysis is also presented.

1.2 Numerical Stability

A real number in a computer is stored as a floating point number with a finite number of significant digits. A number of mathematical problems have solutions quite sensitive to rounding errors. For example,

$$\begin{bmatrix} 10^{-20} & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2+10^{-20} \\ 3 \end{bmatrix},$$

has solution $x = 1$ and $y = 2$. After one step of Gaussian Elimination on a machine with IEEE double precision floating point standard (IEEE 754) which carries up to 16 significant digits, the problem is transformed to

$$\begin{bmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ -2\times 10^{20} \end{bmatrix},$$

which gives a solution of $x = 0$ and $y = 2$. Given a numerical algorithm, we need a proof of the stability to show that such an excessive error is impossible.

In 1970s, Bunch *et al.* developed a series of pivoting algorithms for LBL^T factorization [10, 12, 14]. Some others are reported in recent years [5, 13]. In Chapter 3, we give a condition under which the LBL^T factorization is guaranteed to run to completion in inexact arithmetic with inertia preserved, and analyze the stability of its application to rank estimation for symmetric matrices. In addition, we present a new proof of the componentwise backward stability of these factorization algorithms using the inner product formulation, giving a slight improvement of the bounds in Higham's proofs [39, 40], which relied on the

outer product formulation and normwise analysis. Moreover, the improvement in stability bounds when the matrix is triadic is also displayed.

1.3 Modified Cholesky Algorithms for Optimization

Newton-like methods are widely applied to optimization problems involving twice continuously differentiable functions. Newton's method forms a quadratic model of the objective function around the current iterate x. For unconstrained nonlinear programming with a starting point x, a search direction p is found by solving a linear system $Hp = -g$, where H is the Hessian matrix of the objective function and g is its gradient at the current point. When inequality constraints are present, the objective function is replaced by the Lagrangian; i.e., H is the Hessian matrix of the Lagrangian and g is its gradient. A search direction is a descent direction if the objective function or Lagrangian decreases along this direction with a small enough step length.

When H is not positive definite, the computed p is not guaranteed to be a descent direction. Modified Newton methods replace H by a positive definite matrix $H + E$, where E is symmetric positive semidefinite. The linear system is then $(H + E)p = -g$. The computed p is guaranteed to be a descent direction. One objective is to keep E small, so that $H + E$ is close to H. Chapter 4 shows that modified Newton methods produce a descent direction for nonlinear programming with inequality and/or equality constraints.

There are three Cholesky-related factorizations for symmetric matrices: LDL^T, LBL^T and LTL^T. Positive semidefiniteness [38][41, Chapter 10] is required for

the Cholesky factorization. Quasidefiniteness [29, 62] and diagonal dominance [19] guarantee the existence of the LDL^T factorization. The LBL^T factorization [5, 12, 14] and the LTL^T factorization [1, 50] are for indefinite matrices. Modified Newton methods using these factorizations are sometimes called *modified Cholesky algorithms*.

Gill, Murray and Wright introduced one stable algorithm [28, Chapter 4], whereas Schnabel and Eskow gave another [54, 55]. Their algorithms are via the Cholesky factorization. One distinction is that the Gill-Murray-Wright algorithm has a bound on the modification $\|E\|_2 = O(n^2)$, whereas Schnabel-Eskow algorithm further guarantees $\|E\|_2 = O(n)$. In Chapter 5, we present a couple of variants that also ensure $\|E\|_2 = O(n)$. In our experiments, they are as good as the Schnabel-Eskow algorithm.

Cheng and Higham proposed another algorithm [16], via the same approach of Moré and Sorensen [45]. Their methods are based on the LBL^T factorization. The common problem of the algorithms in this class is that the cost in worst cases is $O(n^3)$ more than the standard Cholesky factorization, whereas $O(n^2)$ is expected. In Chapter 5, we present a new algorithm that guarantees the modification cost $O(n^2)$. Our algorithm slightly outperforms the Cheng-Higham algorithm by producing a smaller $\|E\|$ in the experiments.

1.4 Distance Matrix Completion Problems

We denote a point $p \in R^n$ by a column vector. The *Euclidean distance* between two points p_1, p_2 is defined by

$$\|p_1 - p_2\| = \sqrt{(p_1 - p_2)^T (p_1 - p_2)}.$$

A symmetric matrix $D = [d_{ij}] \in R^{n\times n}$ is called a *Euclidean distance matrix* (EDM)[1], if there are points $p_1, p_2, \ldots, p_n$ such that $d_{ij} = \|p_i - p_j\|^2$ for $i, j = 1, 2, \ldots, n$. Apparently a EDM has zero diagonal and nonnegative off-diagonal elements. A matrix $A = [a_{ij}]$ is *symmetric partial* if there are unspecified entries, and a_{ij} is specified and equal to a_{ji} whenever a_{ji} is specified. Let

$$\mathcal{C}(A) := \{D \in R^{n\times n} : d_{ij} = a_{ij} \text{ for all specified entries } a_{ij} \text{ in } A.\}.$$

A matrix D is called a *EDM completion* of A if and only if $D \in \mathcal{C}(A)$ is a EDM. The *Euclidean distance matrix completion problem* (EDMCP) is to find a EDM completion D of a given symmetric partial matrix A, if any.

One prominent application of the EDMCP is protein structure prediction. The interatomic distance information comes from the structural interpretation of nuclear magnetic resonance data.

A well-known approach to solve the EDMCP is via semidefinite programming [3]. In Chapter 6, we transform the EDMCP into a global optimization problem, and apply modified Newton methods for descent directions. We also have developed a dimensional relaxation method for global minimization.

[1]Some authors define a EDM $D = [d_{ij}]$ by $d_{ij} = \|p_i - p_j\|$, so our D is their $D \circ D$, where $\circ$ denotes *Hadamard* (elementwise) product.

Chapter 2

Matrix Factorizations of Symmetric Triadic Matrices

This chapter is mainly from [23].

A matrix is called *triadic* if the number of nonzero off-diagonal elements in each column is bounded by 2. Tridiagonal matrices are a special case of these, but other matrices, such as block diagonal matrices with full 3×3 blocks, and matrices that are tridiagonal except for entries in each corner, are also triadic. These latter matrices arise in solution of differential equations with periodic boundary conditions. Triadic matrices can also be used as preconditioners for iterative methods.

In this chapter we consider the LL^T, LDL^T and LBL^T factorizations of a symmetric triadic matrices. Section 2.1 proves that the triadic structure is preserved in these factorizations, using any diagonal pivoting strategy. Section 2.2 reviews various pivoting strategies for symmetric matrices, and they are applied to triadic matrices in Section 2.3, where a couple of pivoting strategies specific to symmetric tridiagonal matrices without interchanging rows and columns are also presented. Section 2.4 gives the perturbation analysis of these factorizations. Results are summarized in Section 2.5.

2.1 Diagonal Pivoting Preserves Triadic Structure

As illustrated in Section 1.1, sparsity can be ruined in the matrix factorizations due to fill-in. In this section we show that diagonal pivoting preserves sparsity in these factorizations of symmetric triadic matrices. This is a consequence of the property that for any permutation matrix P, PTP^T is symmetric triadic if and only if T is symmetric triadic.

First we consider the sparsity of LDL^T (and thus LL^T) factorizations with no pivoting. The following lemma on the triadic structure of the Schur complements leads to the desired result. Recall that e_k is the column vector that is zero except for a 1 in its kth position.

Lemma 2.1 *Let* $T = \begin{bmatrix} t_{11} & c_1^T \\ c_1 & T_{22} \end{bmatrix}$ *be a symmetric triadic matrix with* $t_{11} \neq 0$. *Then the Schur complement* $\widehat{T} = T_{22} - c_1 c_1^T / t_{11}$ *is symmetric triadic.*

Proof Since T is triadic, c_1 has at most two nonzero elements. The only nontrivial case is for two nonzero elements, denoted by $c_{i1} = \xi$ and $c_{j1} = \eta$. The matrix T_{22} is also triadic and its ith and jth rows have at most one off-diagonal element each. Moreover,

$$c_1 c_1^T = \xi^2 e_i e_i^T + \xi\eta(e_i e_j^T + e_j e_i^T) + \eta^2 e_j e_j^T$$

has at most four nonzero elements. Two of these are on the diagonal, and the others are in positions (i, j) and (j, i). Thus the sum of T_{22} and $-c_1 c_1^T / t_{11}$ is triadic. □

Theorem 2.1 *In the LDL^T factorization of a symmetric triadic matrix, L is triadic.*

Proof The proof is by finite induction. At the kth step, assume that the remaining $(n-k+1)\times(n-k+1)$ matrix $\overline{T}$ is symmetric triadic. Then off-diagonal elements in the next column of L are computed as c_1/t_{11}, where

$$\overline{T} = \begin{bmatrix} t_{11} & c_1^T \\ c_1 & T_{22} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ c_1/t_{11} & I \end{bmatrix} \begin{bmatrix} t_{11} & 0 \\ 0 & \widehat{T} \end{bmatrix} \begin{bmatrix} 1 & c_1^T/t_{11} \\ 0 & I \end{bmatrix},$$

where $\widehat{T} = T_{22} - c_1 c_1^T/t_{11}$ is the Schur complement of T. Notice that c_1 has at most two nonzero elements. By Lemma 2.1, the matrix $\widehat{T}$, which becomes $\overline{T}$ for the next iteration, is triadic, so we can continue the induction. □

Similarly, the triadic structure is also preserved in the Cholesky factorization.

Now we establish the same result for the LBL^T factorization. The algorithm for LBL^T factorization is the same as for LDL^T factorization with diagonal pivoting, except when all diagonal elements of the Schur complement are zeros. In such a case, we diagonally pivot some nonzero off-diagonal element in the lower triangular part to be at the second row and first column in the Schur complement and perform a decomposition with respect to the 2×2 block. As a result, the factorization is denoted by $PTP^T = LBL^T$, where P is a permutation matrix. To control element growth and improve numerical stability, a pivoting algorithm may choose a 2×2 block pivot, even if the diagonal elements of the 2×2 block are not zero [5, 10, 12, 13, 14].

Lemma 2.2 *Let* $T = \begin{bmatrix} T_{11} & T_{21}^T \\ T_{21} & T_{22} \end{bmatrix}$ *be a symmetric triadic matrix, where* $T_{11} =$

$\begin{bmatrix} \sigma_1 & a \\ a & \sigma_2 \end{bmatrix}$ *with* $\det(T_{11}) \neq 0$. *Then the Schur complement* $\widehat{T} = T_{22} - T_{21}T_{11}^{-1}T_{21}^T$ *with respect to the* 2×2 *pivot* T_{11} *is symmetric triadic.*

Proof If $a = 0$, the result is obtained by invoking Lemma 2.1 twice. We consider the case $a \neq 0$. Since $\det(T_{11}) \neq 0$, $T_{11}^{-1} = \frac{1}{\det(T_{11})} \begin{bmatrix} \sigma_2 & -a \\ -a & \sigma_1 \end{bmatrix}$. Since T has at most two nonzero off-diagonal elements in each column and T_{11} already has one nonzero off-diagonal element in each column, T_{21} has at most one nonzero element in each column. The only nontrivial case is for two nonzero elements in T_{21}, denoted by $\begin{bmatrix} \xi e_i & \eta e_j \end{bmatrix}$. Then

$$\begin{aligned} T_{21}T_{11}^{-1}T_{21}^T &= \tfrac{1}{\det(T_{11})} \begin{bmatrix} \xi e_i & \eta e_j \end{bmatrix} \begin{bmatrix} \sigma_2 & -a \\ -a & \sigma_1 \end{bmatrix} \begin{bmatrix} \xi e_i^T \\ \eta e_j^T \end{bmatrix} \\ &= \tfrac{1}{\det(T_{11})}(\sigma_2\xi^2 e_i e_i^T - a\xi\eta e_j e_i^T + \sigma_1\eta^2 e_j e_j^T - a\xi\eta e_i e_j^T). \end{aligned}$$

Thus the only two off-diagonal elements of this matrix are in positions (i, j) and (j, i). Since T is symmetric triadic, T_{22} has at most one nonzero element in each of ith and jth rows, so the sum of T_{22} and $T_{21}T_{11}^{-1}T_{21}^T$ is triadic. □

Theorem 2.2 *In the* LBL^T *factorization of a symmetric triadic matrix,* L *is triadic.*

Proof Again the proof is by finite induction. At the kth step, assume that the remaining matrix $\overline{T}$ is triadic. If the next pivot is 1×1, then Lemma 2.1 and the argument in the proof of Theorem 2.1 shows that the next column of L is triadic, as is the new remaining matrix. If the next pivot is 2×2, then the factorization produces

$$\overline{T} = \begin{bmatrix} T_{11} & T_{21}^T \\ T_{21} & T_{22} \end{bmatrix} = \begin{bmatrix} I_2 & 0 \\ T_{21}T_{11}^{-1} & I_{k-2} \end{bmatrix} \begin{bmatrix} T_{11} & 0 \\ 0 & \widehat{T} \end{bmatrix} \begin{bmatrix} I_2 & T_{11}^{-T}T_{21}^T \\ 0 & I_{k-2} \end{bmatrix},$$

The off-diagonal part of the two new columns of L are

$$\begin{aligned} T_{21}T_{11}^{-1} &= \frac{1}{\det(T_{11})} \begin{bmatrix} \xi e_i & \eta e_j \end{bmatrix} \begin{bmatrix} \sigma_2 & -a \\ -a & \sigma_1 \end{bmatrix} \\ &= \frac{1}{\det(T_{11})} \begin{bmatrix} \sigma_2\xi e_i - a\eta e_j & -a\xi e_i + \sigma_1\eta e_j \end{bmatrix}, \end{aligned}$$

also triadic, and Lemma 2.2 shows that $\widehat{T}$ is triadic, so the induction can be continued. □

Combining Theorems 2.1 and 2.2 with the fact that the triadic property of a matrix is preserved under symmetric permutation, we see that the number of nonzero elements is $O(n)$ in all of these factorizations if diagonal pivoting is used. More precisely, by Lemmas 2.1 and 2.2 at most $n-2$ off-diagonal fill entries can occur.

Theorem 2.3 *If we factor a symmetric triadic matrix using LL^T, LDL^T or LBL^T factorization with any diagonal pivoting, then L is triadic.*

Although the columns of L are sparse, the number of nonzero elements in each row of L is bounded only by n; if T is symmetric tridiagonal, for example, and

$$\widetilde{Z} = \begin{bmatrix} 0 & & & 1 \\ 1 & \ddots & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \tag{2.1}$$

is the circular shift-down matrix, then the last row of L in the factorization $\widetilde{Z}^T T \widetilde{Z} = LDL^T$ is generally full.

2.2 Diagonal Pivoting Strategies for Symmetric Indefinite Matrices

Traditionally, the stability analysis of LDL^T and LBL^T factorizations of a symmetric matrix $A \in R^{n\times n}$ involves the *growth factor*

$$\rho(A) = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|}, \tag{2.2}$$

where a_{ij} and $a_{ij}^{(k)}$ are the (i,j) entries of A and the kth Schur complement, respectively. In some applications, such as computing a Newton-like direction for optimization, a bound on the elements in L is also required.

If the symmetric matrix $A \in R^{n\times n}$ is positive semidefinite [19][41, Section 10.3] or diagonally dominant [19][41, Section 9.5] (i.e., $|a_{ii}| \geq \sum_{j\neq i} |a_{ij}|$ for $i = 1, \ldots, n$), then the largest magnitude element will appear on the diagonal. Each Schur complement inherits the property of positive semidefiniteness or diagonal dominance. Therefore, with pivoting in either case, the elements of L in the LDL^T factorization are bounded in magnitude by 1. With or without pivoting, the growth factor is $\rho(A) = 1$ if A is symmetric positive semidefinite, and $\rho(A) \leq 2$ if A is diagonally dominant. Quasidefiniteness[1] also guarantees the existence of the LDL^T factorization [29, 62].

A symmetric indefinite matrix is factorized in the LBL^T form. Pivoting can control the element growth and bound the elements in L. There are three kinds of pivoting strategies in the literature: Bunch-Parlett [14] (complete pivoting);

[1]We say that a symmetric matrix $A \in R^{n\times n}$ is symmetric quasidefinite if there exists a permutation matrix P such that PAP^T has the form $\begin{bmatrix} H & K^T \\ K & -G \end{bmatrix}$ where H and G are positive definite.

fast Bunch-Parlett and bounded Bunch-Kaufman [5] (rook pivoting); and Bunch-Kaufman [12] (partial pivoting). For full matrices, complete pivoting requires $O(n^3)$ comparisons, partial pivoting requires $O(n^2)$, and the cost of rook pivoting varies between $O(n^2)$ and $O(n^3)$. Therefore, it is interesting to uncover the advantages of the more expensive strategies. We consider each strategy in turn, applying each to the current Schur complement matrix denoted by A. Note that all the pivoting strategies have a preset constant $0 < \alpha < 1$.

2.2.1 Complete Pivoting

Bunch and Parlett [14] devised the pivoting strategy presented in Algorithm 2.1.

Algorithm 2.1 Bunch-Parlett pivot selection.

Let a_{kk} be the largest magnitude diagonal element.

Let a_{ij} $(i < j)$ be the largest magnitude off-diagonal element.

if $|a_{kk}| \geq \alpha |a_{ij}|$ **then**

Use a_{kk} as a 1×1 pivot.

else

Use $\begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}$ as a 2×2 block pivot.

end if

The process continues until $a_{kk} = a_{ij} = 0$ or the factorization completes. The resulting pivot satisfies the following *strong condition*:

1. If a 1×1 pivot a_{kk} is chosen, then $|a_{kk}| \geq \alpha |a_{pk}|$ for $p \neq k$.
2. If a 2×2 block pivot $\begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}$ is chosen, then $|a_{ii}| < \alpha |a_{ij}|$, $|a_{jj}| < \alpha |a_{ij}|$, and a_{ij} is the element of maximum magnitude in both columns i and j.

For any algorithm satisfying the strong condition, the elements in L are bounded and the element growth during the factorization is well controlled, as we will show in Sections 2.2.4 and 2.2.5.

2.2.2 Rook Pivoting

The cost for finding a pivot satisfying the strong condition can be reduced by the iterative process in Algorithm 2.2 by Ashcraft, Grimes, and Lewis [5].

Algorithm 2.2 Pivot selection by rook pivoting, given an initial index i.

Find the index $j \neq i$ such that $|a_{ji}| = \max\{|a_{pi}| : p \neq i\}$.

if $|a_{ii}| \geq \alpha |a_{ji}|$ **then**

Use a_{ii} as a 1×1 pivot.

else

Find the index $k \neq j$ such that $|a_{kj}| = \max\{|a_{pj}| : p \neq j\}$.

repeat

if $|a_{jj}| \geq \alpha |a_{kj}|$ **then**

Use a_{jj} as a 1×1 pivot.

else if $|a_{ij}| = |a_{kj}|$ **then**

Use $\begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}$ as a 2×2 pivot.

else

Set $i := j$ and $j := k$.

Find index $k \neq j$ such that $|a_{kj}| = \max\{|a_{pj}| : p \neq j\}$.

end if

until a pivot is chosen.

end if

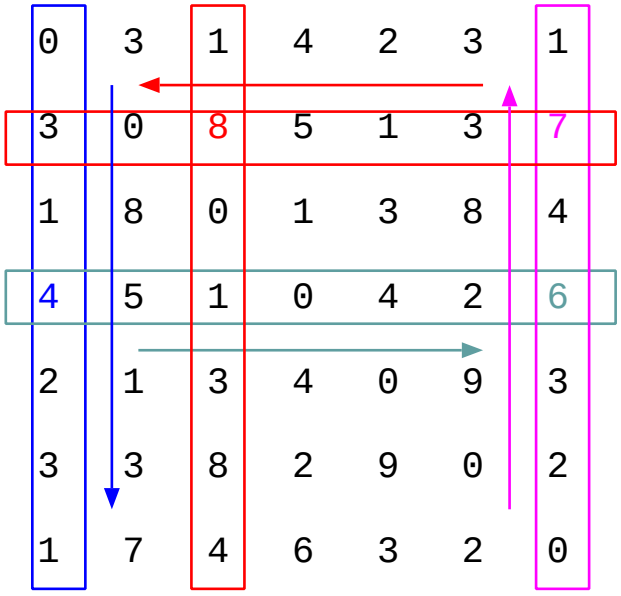

Figure 2.1: An example of rook pivoting.

An example with zero diagonal is illustrated in Figure 2.1. We search the first column and find the largest element 4, whose row is then searched for the largest element 6 in the last column, where we find the largest element 7 in the second position. Finally we find 8 the largest element in the second row. It is also the largest element in the third column. By rook pivoting, we choose the 2×2 block pivot $\begin{bmatrix} 0 & 8 \\ 8 & 0 \end{bmatrix}$. Note that this block pivot satisfies the strong condition, although 8 is not the largest element in the whole matrix.

If the initial pivot index $i := 1$, this is called *bounded Bunch-Kaufman* pivot selection, while if a_{ii} is the maximal magnitude diagonal element, it is called *fast Bunch-Parlett* pivot selection [5]. Note that for fast Bunch-Parlett selection, we do not need to test whether a_{jj} is a 1×1 pivot or not in the loop, because if the initial maximum magnitude diagonal element a_{ii} failed to be a pivot at the beginning, $|a_{jj}|$ is at most $|a_{ii}|$, and $|a_{ij}|$ is increasing in the loop. Foster [26] gave the probabilistic analysis of unsymmetric rook pivoting for LU factorization.

2.2.3 Partial Pivoting

Bunch and Kaufman [12] devised the efficient pivoting strategy shown in Algorithm 2.3.

Algorithm 2.3 Bunch-Kaufman pivot selection, given an initial index i.

{Given initial pivot index i.}

Find the index $j \neq i$ such that $|a_{ji}| = \max\{|a_{ki}| : k \neq i\} =: \lambda$.

if $|a_{ii}| \geq \alpha\lambda$, **then**

Use a_{ii} as a 1×1 pivot.

else

(*) Compute $\sigma := \max\{|a_{kj}| : k \neq j\} \geq \lambda$.

if $|a_{ii}|\sigma \geq \alpha\lambda^2$ **then**

Use a_{ii} as a 1×1 pivot.

else if $|a_{jj}| \geq \alpha\sigma$ **then**

Use a_{jj} as a 1×1 pivot.

else

Use $\begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}$ as a 2×2 pivot.

end if

end if

Bunch-Kaufman pivoting does not guarantee the strong condition, but satisfies the following *weak condition*:

1. If a 1×1 pivot a_{kk} is chosen, then
   - $|a_{kk}| \max\{|a_{pq}| : p \neq q \text{ and } (a_{pk} \neq 0 \text{ or } q = k)\} \geq \alpha \max_{p\neq k} |a_{pk}|^2$.
2. If a 2×2 block pivot $\begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}$ is chosen $(i < j)$, then

- $|a_{ii}| < \alpha\lambda$,
- $|a_{ii}|\sigma < \alpha\lambda^2$,
- $|a_{jj}| < \alpha\sigma$,

where $\lambda = \max_{k \neq i} |a_{ki}|$ and $\sigma = \max_{k \neq j} |a_{kj}|$.

We compare the weak condition with the strong condition. For 1×1 pivots, $\max\{|a_{pq}| : p \neq q \text{ and } (a_{pk} \neq 0 \text{ or } q = k)\} \geq \max_{p \neq k} |a_{pk}|$ so the strong condition guarantees the weak condition. For 2×2 block pivots, the weak condition meets the strong condition if $\sigma = \lambda$. We conclude that the strong condition implies the weak condition.

The natural choice of the initial pivot index i in Algorithm 2.3 is $i := 1$, which achieves the least cost to satisfy the weak condition [12].

Ashcraft, Grimes and Lewis [5] argued that a bounded L can improve stability. We can improve the probability that the Bunch-Kaufman algorithm has a bounded L by choosing the largest magnitude diagonal entry as the search starting point at each pivot step [12]. The additional number of comparisons is $\frac{n^2}{2} + O(n)$, so the total comparison count remains $O(n^2)$. By making this change, we usually find a 1×1 pivot at the very first test at each step of pivot selection. The strong condition usually holds, but it is not guaranteed as shown in the following example [39]:

$$A = \begin{bmatrix} \epsilon^2 & \epsilon & \epsilon \\ \epsilon & 0 & 1 \\ \epsilon & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & & \\ \frac{1}{\epsilon} & 1 & \\ \frac{1}{\epsilon} & 0 & 1 \end{bmatrix} \begin{bmatrix} \epsilon^2 & & \\ & -1 & \\ & & -1 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{\epsilon} & \frac{1}{\epsilon} \\ & 1 & 0 \\ & & 1 \end{bmatrix} = LBL^T,$$

where L is unbounded as $\epsilon \to 0$.

2.2.4 The Weak Condition Controls the Growth Factor

In summary, the Bunch-Parlett, fast Bunch-Parlett, and bounded Bunch-Kaufman pivoting strategies satisfy the strong condition, whereas the Bunch-Kaufman pivoting strategy satisfies the weak condition.

The weak condition controls the element growth during the factorization, as shown by an argument similar to those in [5, 12, 14, 39] [41, Chapter 11]. The growth factor is defined in (2.2).

When a 1×1 pivot is chosen, we have

$$\begin{aligned} & \frac{\max\{|a_{pk}| : p \neq k\}^2}{|a_{kk}|} \\ \leq\ & \frac{1}{\alpha}\max\{|a_{pq}| : p \neq q \text{ and } (a_{pk} \neq 0 \text{ or } q = k)\} \qquad (2.3) \\ \leq\ & \frac{1}{\alpha}\max_{p\neq q}|a_{pq}|. \end{aligned}$$

Therefore, the element growth is bounded by $1 + \frac{1}{\alpha}$.

If a 2×2 block pivot is chosen, the weak condition guarantees $|a_{ii}a_{jj}| < \alpha^2\lambda^2$. Then

$$|\det(\begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix})| = |a_{ij}^2 - a_{ii}a_{jj}| > (1-\alpha^2)\lambda^2 \qquad (2.4)$$

Since $0 < \alpha < 1$,

$$|\begin{bmatrix} a_{ii} & a_{ij} \\ a_{ji} & a_{jj} \end{bmatrix}^{-1}| < \frac{1}{(1-\alpha^2)\lambda^2}\begin{bmatrix} |a_{jj}| & \lambda \\ \lambda & |a_{ii}| \end{bmatrix}.$$

Therefore, the increase of each element in magnitude for the 2×2 block decomposition is bounded by

$$\frac{1}{(1-\alpha^2)\lambda^2}\begin{bmatrix} \lambda & \sigma \end{bmatrix}\begin{bmatrix} |a_{jj}| & \lambda \\ \lambda & |a_{ii}| \end{bmatrix}\begin{bmatrix} \lambda \\ \sigma \end{bmatrix}$$

$$
\begin{aligned}
&= \frac{1}{(1-\alpha^2)\lambda^2}(\lambda^2(|a_{jj}|+\sigma)+(\lambda^2+\sigma|a_{ii}|)\sigma) \\
&< \frac{1}{(1-\alpha^2)\lambda^2}(\lambda^2(\alpha\sigma+\sigma)+(\lambda^2+\alpha\lambda^2)\sigma) \\
&= \frac{2(1+\alpha)\sigma}{1-\alpha^2} = \frac{2\sigma}{1-\alpha}, \qquad (2.5)
\end{aligned}
$$

and the element growth for the 2×2 block decomposition is bounded by $1+\frac{2}{1-\alpha}$.

Therefore, the element growth is bounded by

$$
g = \max\left\{1+\frac{1}{\alpha}, \sqrt{1+\frac{2}{1-\alpha}}\right\}.
$$

The minimum of g is $\frac{1+\sqrt{17}}{2} \approx 2.562$, which is attained when $\alpha = \frac{1+\sqrt{17}}{8} \approx 0.640$. Thus

$$
\rho(A) \le g^{n-1}. \qquad (2.6)
$$

The attainability of the last inequality is a research problem [41, Problem 11.10].

With complete pivoting (Bunch-Parlett pivoting strategy), we can obtain a smaller bound on the growth factor of $A \in R^{n\times n}$ as

$$
\rho(A) \le 3nf(n), \text{ where } f(n) = (\prod_{k=2}^{n} k^{1/(k-1)})^{1/2} \le 1.8n^{(\ln n)/4} \qquad (2.7)
$$

with the pivoting argument $\alpha = \frac{1+\sqrt{17}}{8}$. This was shown by Bunch [9] with an analysis similar to Wilkinson's for Gaussian elimination with complete pivoting [65].

We note that the bounds on element increases in (2.3) and (2.5) are in terms of off-diagonal elements. Therefore, the growth factor $\bar{\rho}(A)$ for off-diagonal elements is bounded by g^{n-2}, i.e.,

$$
\bar{\rho}(A) = \frac{\max_{i\neq j,k}|a_{ij}^{(k)}|}{\max_{i\neq j}|a_{ij}|} \le g^{n-2} \qquad (2.8)
$$

for $n > 1$.

The bound on $\bar{\rho}(A)$ (2.8) is attainable, for example, applying Bunch-Kaufman pivoting strategy with $\alpha = \frac{1+\sqrt{17}}{8}$ on

$$A = \begin{bmatrix} -\alpha & 1 & 1 & \cdots & 1 \\ 1 & -\alpha g - \frac{1}{\alpha} & 1 & \cdots & 1 \\ 1 & 1 & -\alpha g^2 - \frac{g}{\alpha} - \frac{1}{\alpha} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 1 \\ 1 & 1 & \cdots & 1 & -\alpha g^{n-1} - \frac{g^{n-2}}{\alpha} - \frac{g^{n-3}}{\alpha} - \cdots - \frac{1}{\alpha} \end{bmatrix}.$$

The weak condition is stronger than necessary to bound the growth factor; we need only

$$|a_{kk}| \max_{p \neq q} |a_{pq}| \geq \alpha \max_{p \neq k} |a_{pk}|^2$$

for 1×1 pivots, but our version of the weak condition is useful for the triadic case considered in Section 2.3.2.

Sorensen and Van Loan [21, Section 5.3.2] suggested a variant of the Bunch-Kaufman pivoting strategy by modifying (*) in Algorithm 2.3 to be:

$$\sigma := \max_{k} |a_{kj}| \geq \lambda. \tag{2.9}$$

This small change ensures that for a positive definite matrix no interchanges are done and only 1×1 pivots are used. The bound on the growth factor (2.6) still holds; however, there is no bound on the off-diagonal factor. For example,

$$A = \begin{bmatrix} \epsilon & 1 & 1 \\ 1 & \frac{1}{\epsilon} & 0 \\ 1 & 0 & \frac{1}{\epsilon} \end{bmatrix} = \begin{bmatrix} 1 & & \\ \frac{1}{\epsilon} & 1 & \\ \frac{1}{\epsilon} & 0 & 1 \end{bmatrix} \begin{bmatrix} \epsilon & & \\ & 0 & -\frac{1}{\epsilon} \\ & -\frac{1}{\epsilon} & 0 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{\epsilon} & \frac{1}{\epsilon} \\ & 1 & 0 \\ & & 1 \end{bmatrix} = LBL^T.$$

The growth factor is $\rho(A) = 1$ but the off-diagonal growth factor is $\bar{\rho}(A) = \frac{1}{\epsilon}$, which is unbounded as $\epsilon \to 0$.

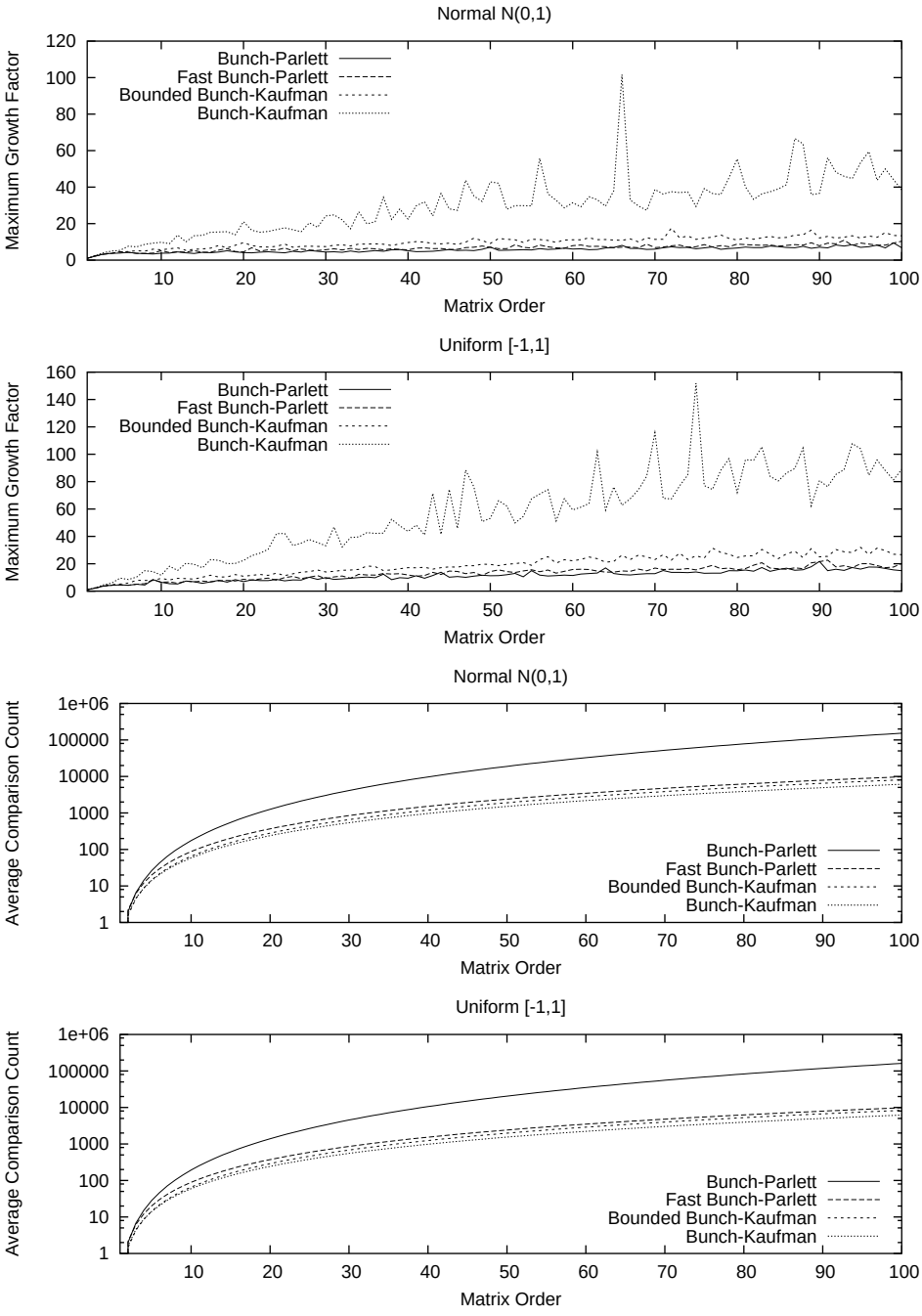

Figure 2.2: Experimental maximum growth factor and average number of comparisons for factoring a symmetric matrix, 20,000 matrices for each method and matrix size.

In practice, the average growth factors for both tridiagonal and full matrices are far from this bound. Figure 2.2 shows the results of an experiment for the maximum growth factor and the average number of comparisons of $20,000$ random symmetric $n \times n$ matrices for each $n = 1, \ldots, 100$ with $\alpha = \frac{1+\sqrt{17}}{8} \approx 0.640$. All the matrices are either from the normal $N(0,1)$ distribution or from the uniform $[-1,1]$ distribution.

Although $\alpha \approx 0.640$ minimizes the bound on the growth factor, our experiments show that the best α to minimize the average growth factor is usually between 0.74 and 0.78, as shown in Figure 2.3, where $20,000$ random matrices are generated for each matrix size, each type of distribution and each α.

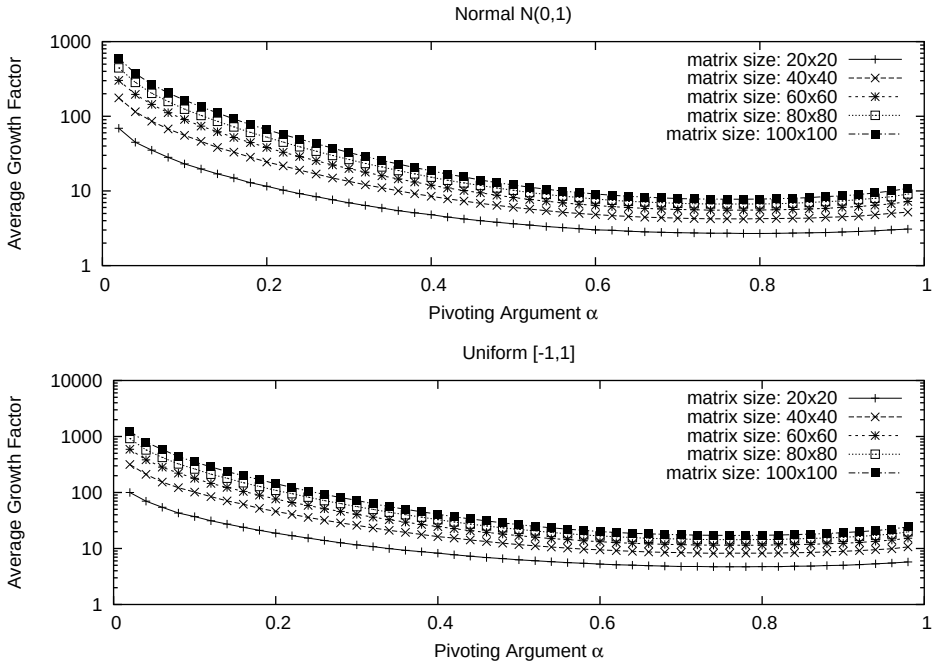

Figure 2.3: Experimental average growth factor for factoring a symmetric matrix using the Bunch-Kaufman pivoting strategy, 20,000 matrices for each method and matrix size.

2.2.5 The Strong Condition Bounds Elements in L

The weak condition does not bound elements in L. For example [39][41, Section 11.1.2],

$$A = \begin{bmatrix} 0 & \epsilon & \\ \epsilon & 0 & 1 \\ & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ 0 & 1 & \\ \frac{1}{\epsilon} & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \epsilon & \\ \epsilon & 0 & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \frac{1}{\epsilon} \\ & 1 & 0 \\ & & 1 \end{bmatrix} = LBL^T,$$

when the Bunch-Kaufman pivoting strategy is applied. As $\epsilon \to 0$, L is unbounded.

In contrast, the strong condition does ensure a bound on elements in L. When a 1×1 pivot is chosen, then the magnitude of elements in the pivot column of L is bounded by $\frac{1}{\alpha}$. If a 2×2 block pivot is chosen, the strong condition implies $\lambda = \sigma$ and therefore the two columns of L corresponding to this 2×2 block pivot have elements bounded by

$$\begin{aligned} \frac{1}{(1-\alpha^2)\lambda^2} \begin{bmatrix} \lambda & \sigma \end{bmatrix} \begin{bmatrix} |a_{jj}| & \lambda \\ \lambda & |a_{ii}| \end{bmatrix} &= \frac{1}{(1-\alpha^2)\lambda^2} \begin{bmatrix} \lambda & \lambda \end{bmatrix} \begin{bmatrix} \alpha\lambda & \lambda \\ \lambda & \alpha\lambda \end{bmatrix} \\ &= \frac{1+\alpha}{1-\alpha^2} \begin{bmatrix} 1 & 1 \end{bmatrix} = \frac{1}{1-\alpha} \begin{bmatrix} 1 & 1 \end{bmatrix}. \end{aligned}$$

Therefore, the elements in L are bounded in magnitude by

$$\gamma = \max\left\{\frac{1}{\alpha}, \frac{1}{1-\alpha}\right\}.$$

2.2.6 The Growth Factor and Element Bounds

We summarize our results on element growth in the following theorem.

Theorem 2.4 *For LBL^T factorization of a symmetric matrix $A \in R^{n\times n}$, if the weak condition holds, then the growth factor $\rho(A)$ defined in (2.2) is bounded by*

$$\rho(A) \le g^{n-1},$$

where

$$g = \max\left\{1 + \frac{1}{\alpha}, \sqrt{1 + \frac{2}{1-\alpha}}\right\},$$

with α the parameter in the factorization algorithm. If the strong condition holds, then the elements in L are bounded in magnitude by

$$\gamma = \max\left\{\frac{1}{\alpha}, \frac{1}{1-\alpha}\right\}.$$

As shown above, $\alpha = \frac{1+\sqrt{17}}{8}$ minimizes the bound g on element growth, but $\alpha = 0.5$ minimizes the bound γ on the elements of L. The consequences of each of these choices are summarized in Table 2.1.

Table 2.1: The element growth bound g and the bound γ for L (when complete or rook pivoting is used) with two optimal choices of α.

	α	g	γ
minimize g	$\frac{1+\sqrt{17}}{8} \approx 0.640$	$\frac{1+\sqrt{17}}{2} \approx 2.562$	$\frac{7+\sqrt{17}}{4} \approx 2.781$
minimize γ	$\frac{1}{2}$	3	2

2.3 Diagonal Pivoting Strategies for Symmetric Triadic Matrices

In Section 2.1, we showed that sparsity is preserved in LL^T, LDL^T and LBL^T factorizations of a symmetric triadic matrix with any diagonal pivoting strategy. In this section, we study two pivoting strategies particular to symmetric tridiagonal matrices [10, 13] and also apply the pivoting strategies from the previous section to triadic matrices.

2.3.1 Pivoting Strategies for Symmetric Tridiagonal Matrices

One pivoting strategy has been proposed for LBL^T factorizations of irreducible tridiagonal matrices. Consider the variant proposed by Higham [40] of the algorithm of Bunch [10] represented in Algorithm 2.4. The great advantage is that there are *no* interchanges of rows and columns, yet the growth factor is bounded by

$$\rho(A) = \max\{1 + \frac{1}{\alpha}, \frac{1}{1-\alpha}\},$$

whose minimum is $\frac{\sqrt{5}+3}{2} \approx 2.618$, achieved by choosing $\alpha = \frac{\sqrt{5}-1}{2}$.

Algorithm 2.4 Bunch's pivot selection.

$\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$

$\sigma =$ the maximum magnitude of the elements in the initial matrix.

if $|a_{11}|\sigma \geq \alpha |a_{21}|^2$ **then**

Use a_{11} as a 1×1 pivot.

else

Use $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ as a 2×2 block pivot.

end if

In Algorithm 2.4, choosing the pivot size requires knowing *a priori* the largest element in magnitude σ of the initial symmetric tridiagonal matrix. In some applications, such as solving indefinite symmetric systems using Lanczos method, it is favored to factor a symmetric tridiagonal matrix without knowing the whole matrix in advance. Bunch and Marcia [13] devised a pivoting strategy to achieve this goal and preserve the same bound on the growth factor $\rho(A) \leq \frac{\sqrt{5}+3}{2} \approx 2.618$. Their method is presented in Algorithm 2.5.

Algorithm 2.5 Bunch-Marcia pivot selection.

$\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$

Compute $\Delta := a_{11}a_{22} - a_{21}^2$.

if $|\Delta| \leq \alpha |a_{11}a_{31}|$ or $|a_{21}\Delta| \leq \alpha |a_{11}^2 a_{31}|$ **then**

Use a_{11} as a 1×1 pivot.

else

Use $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ as a 2×2 block pivot.

end if

Both pivoting strategies are excellent for applications relying on B (e.g., computing inertia), but there is no element bound on L. For example, both Bunch's pivoting strategy and Bunch-Marcia pivoting strategy produce the following LBL^T factorization for $\epsilon \neq 0$,

$$A = \begin{bmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ 0 & 1 & \\ 1/\epsilon & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \epsilon & \\ \epsilon & 0 & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1/\epsilon \\ & 1 & 0 \\ & & 1 \end{bmatrix} = LBL^T, \quad (2.10)$$

which is presented in [39] as a factorization using Bunch-Kaufman pivoting strategy. Hence, both algorithms are not well suited to computing Newton-like directions or solving circulant systems of equations. Nevertheless, Higham proved the stability of Bunch's pivoting strategy [40]. Bunch and Marcia also demonstrated that their method is normwise backward stable [13].

2.3.2 Pivoting Strategies from Those for Dense Matrices

All the pivoting strategies from Section 2.2 can be applied to a symmetric triadic matrix $A \in R^{n \times n}$. The growth factor is constrained because of the triadic struc-

ture, and we obtain a sharper result for $\rho(A)$ than that of Theorem 2.4, whereas the bound γ on the elements of L remains the same.

Theorem 2.5 *For LBL^T factorization of a symmetric triadic $A \in R^{n\times n}$, the growth factor of off-diagonal elements, defined in (2.8), is bounded as*

$$\bar{\rho}(A) \leq \begin{cases} 2g^{\lfloor \lg(n-1) \rfloor} \leq 2(n-1)^{\lg g} & \textit{if the strong condition holds,} \\ 2g^{\lfloor (n-1)/2 \rfloor} & \textit{if the weak condition holds.} \end{cases}$$

for $n > 1$, where

$$g = \max\{\frac{1}{\alpha}, \frac{1}{1-\alpha^2}\}. \tag{2.11}$$

If we choose $\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$ to minimize g to be $\frac{\sqrt{5}+1}{2} \approx 1.618$, then $\lg g \approx 0.694$, and therefore the bound for strong condition is sub-linear.

Proof Without loss of generality, we assume the required interchanges of rows and columns for pivoting are done prior to the factorization. Let $S_k(A)$ be the Schur complement of A after reducing k rows and k columns, and let

$$A^{(k+1)} = \begin{array}{c} \\ k \\ n-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & S_k(A) \end{bmatrix} \end{array}.$$

By Lemmas 2.1 and 2.2, at most two diagonal and two off-diagonal elements are changed in the Schur complement. We denote them by $a_{ii}^{(k+1)}$, $a_{jj}^{(k+1)}$, $a_{ij}^{(k+1)}$ and $a_{ji}^{(k+1)}$. In addition to $a_{ij}^{(k+1)}$ and $a_{ji}^{(k+1)}$, $A^{(k+1)}$ has at most one nonzero off-diagonal element in each of ith and jth rows, inherited from $A^{(k-p)}$, where $p = 1$ or $p = 2$ when the previous pivot is 1×1 or 2×2, respectively.

Assume for now that

$$a_{ij}^{(k+1-p)} = a_{ji}^{(k+1-p)} = 0 \tag{2.12}$$

for each k. Later we will show that if this assumption breaks, the bounds on the off-diagonal growth factor are at most doubled.

For a 1×1 pivot, (2.12) implies that the weak condition coincides with the strong condition. Therefore,

$$|a_{ij}^{(k+1)}| = \frac{|a_{ik}^{(k)}||a_{jk}^{(k)}|}{|a_{kk}^{(k)}|} \leq \frac{1}{\alpha}\min\{|a_{ik}^{(k)}|, |a_{jk}^{(k)}|\} \leq g\min\{|a_{ik}^{(k)}|, |a_{jk}^{(k)}|\} \tag{2.13}$$

For a 2×2 pivot $\begin{bmatrix} a_{k-1,k-1}^{(k-1)} & a_{k-1,k}^{(k-1)} \\ a_{k,k-1}^{(k-1)} & a_{kk}^{(k-1)} \end{bmatrix}$, there are at most two nonzero off-diagonal elements under the pivot, denoted by $a_{i,k-1}^{(k-1)}$ and $a_{jk}^{(k-1)}$. If $i = j$, then the only element changed in $A^{(k+1)}$ from $A^{(k-1)}$ is $a_{ii}^{(k+1)}$. In this case, the matrix size is reduced without increasing the off-diagonal elements. In order to maximize $\bar{\rho}(A)$, we assume $i \neq j$. The weak condition ensures (2.4). Therefore,

$$\begin{aligned} |a_{ij}^{(k+1)}| &\leq \frac{1}{(1-\alpha^2)|a_{k,k-1}^{(k-1)}|^2} \begin{bmatrix} |a_{i,k-1}^{(k-1)}| & 0 \end{bmatrix} \begin{bmatrix} |a_{kk}^{(k-1)}| & |a_{k-1,k}^{(k-1)}| \\ |a_{k,k-1}^{(k-1)}| & |a_{k-1,k-1}^{(k-1)}| \end{bmatrix} \begin{bmatrix} 0 \\ |a_{jk}^{(k-1)}| \end{bmatrix} \\ &= \frac{|a_{i,k-1}^{(k-1)}||a_{jk}^{(k-1)}|}{(1-\alpha^2)|a_{k,k-1}^{(k-1)}|} \\ &\leq \begin{cases} g\min\{|a_{i,k-1}^{(k-1)}|, |a_{jk}^{(k-1)}|\} & \text{if the strong condition holds,} \\ g|a_{jk}^{(k-1)}| & \text{if the weak condition holds.} \end{cases} \end{aligned} \tag{2.14}$$

Since all the Schur complements are symmetric, we consider the elements in the lower triangle. Let $G(m) = g^m \max_{i\neq j}|a_{ij}|$.

Consider the case that the strong condition holds. By (2.14) for a 2×2 pivot, an off-diagonal element of size $G(m)$ requires three $G(m-1)$ elements: $|a_{i,k-1}^{(k-1)}|$, $|a_{jk}^{(k-1)}|$, and $|a_{k,k-1}^{(k-1)}|$. Note that the strong condition guarantees $|a_{k,k-1}^{(k-1)}| \geq |a_{i,k-1}^{(k-1)}|$. By (2.13) for a 1×1 pivot, if $|a_{ij}^{(k+1)}| \geq G(m)$, then $|a_{ik}^{(k)}|, |a_{jk}^{(k)}| \geq G(m-1)$. In other words, an off-diagonal element of size $G(m)$ requires two off-diagonal supporting

elements of size $G(m-1)$. Therefore, the bound on element growth using 1×1 pivots is higher than that using 2×2 pivots. We see by induction that a $G(m)$ element requires $2^m G(0)$ elements, using 1×1 pivots.

The following diagram is an illustration for obtaining a $G(3)$ element with the smallest number of pivots. Note that $G(0)$ elements are from the original matrix A, whereas $G(1)$, $G(2)$, and $G(3)$ elements are fill-in entries during the factorization. The last column indicates the Schur complements as the sources of the two off-diagonal elements in each row, if they were not present initially.

$*$		$G(0)$					$G(0)$	
	$*$	$G(0)$				$G(0)$		
$G(0)$	$G(0)$	$*$				$G(1)$	$G(1)$	$A^{(2)}, A^{(1)}$
			$*$		$G(0)$		$G(0)$	
				$*$	$G(0)$	$G(0)$		
			$G(0)$	$G(0)$	$*$	$G(1)$	$G(1)$	$A^{(5)}, A^{(4)}$
	$G(0)$	$G(1)$		$G(0)$	$G(1)$	$*$	$G(2)$	$G(2)$	$A^{(6)}, A^{(3)}$
			$G(0)$		$G(1)$	$G(2)$	$*$	$G(3)$	$A^{(7)}$
$G(0)$		$G(1)$				$G(2)$	$G(3)$	$*$	

Consider the elements in the lower triangle. Each pivot can produce a $G(k)$ element from two $G(k-1)$ elements for some $1 < k \leq m$. The number of pivots required to obtain a $G(m)$ element is $2^{m-1} + 2^{m-2} + \ldots + 2^0 = 2^m - 1$. The last 2×2 Schur complement, with or without a row/column reduced afterward, cannot constitute off-diagonal element growth. The least matrix size required to have a $G(m)$ off-diagonal element is $(2^m - 1) + 2 = 2^m + 1$. If the matrix size of a symmetric matrix $A \in R^{n\times n}$ is less than $2^m + 1$ but larger than 2^{m-1},

then off-diagonal elements in the Schur complements are at most $G(m-1)$ in magnitude. In other words,

$$\bar{\rho}(A) \leq g^{\lfloor \lg(n-1) \rfloor} \leq (n-1)^{\lg g}. \tag{2.15}$$

Consider the case that the weak condition holds. Recall that for a 1×1 pivot, the weak condition coincides with the strong condition, and an off-diagonal element $G(m)$ requires two $G(m-1)$ elements. By (2.14) for a 2×2 pivot, an off-diagonal element of size $G(m)$ requires only one $G(m-1)$ element. From $G(0)$ to $G(1)$ we use a 1×1 pivot for maximum growth. Otherwise, the bound on element growth using 2×2 pivots is at least as big as that using 1×1 pivots. The bound can increase by a factor of g every two rows/columns reduced during the decomposition, except from $G(0)$ to $G(1)$ (one row/column reduced). The last 1×1 and 2×2 Schur complements, if any, cannot constitute off-diagonal element growth. Therefore,

$$\bar{\rho}(A) \leq g^{\lfloor (n-1)/2 \rfloor}, \tag{2.16}$$

where $A \in R^{n \times n}$ is symmetric triadic.

So far we assume (2.12) holds. Now we show that if (2.12) breaks, the bounds in (2.15) and (2.16) are at most doubled. If $a_{ij}^{(k+1-p)} = a_{ji}^{(k+1-p)} \neq 0$, then there are no other off-diagonal elements in the ith and jth rows and columns in $A^{(k+1)}$, where $p = 1, 2$ stands for 1×1, 2×2 pivots, respectively. As a result, $A^{(k+1)}$ is a reducible matrix. After diagonally interchanging rows and columns, $A^{(k+1)}$ consists of two diagonal blocks: $\begin{bmatrix} a_{ii}^{(k+1)} & a_{ij}^{(k+1)} \\ a_{ji}^{(k+1)} & a_{jj}^{(k+1)} \end{bmatrix}$ and the remaining matrix, in which all the elements are taken from $A^{(k+1-p)}$. The bound on $a_{ji}^{(k+1)}$ in the 2×2 block is at most doubled, since it is a sum of two terms each of which is bounded as (2.15) or (2.16), depending on whether the condition satisfied is strong or

weak. Note that no off-diagonal element growth afterward in this 2×2 block, and the other block is intact. Therefore, we obtain the result by safely declaring that the bounds in (2.15) and (2.16) are at most doubled if (2.12) breaks. □

Theorem 2.6 *For LBL^T factorization of a symmetric triadic $A \in R^{n\times n}$, consider the growth factor, defined in (2.2). If the weak condition holds,*

$$\rho(A) \leq \begin{cases} \frac{4g(g^{(n-3)/2}-1)}{g-1} + 2(g^{(n-1)/2} + g^{(n+1)/2}) + 1 & \text{if } n \text{ odd,} \\ \frac{4g(g^{(n-2)/2}-1)}{g-1} + 2g^{n/2} + 1 & \text{if } n \text{ even.} \end{cases}$$

That is, $\rho(A) = O(g^{n/2})$. If the strong condition holds,

$$\rho(A) \leq 2ng^{\lfloor \lg(n-1) \rfloor} \leq 2n(n-1)^{\lg g} = O(n^{1+\lg g}),$$

for $n > 1$, where

$$g = \max\{\frac{1}{\alpha}, \frac{1}{1-\alpha^2}\}.$$

If we choose $\alpha = \frac{\sqrt{5}-1}{2}$ to minimize g to be $\frac{\sqrt{5}+1}{2}$, then $\lg g \approx 0.694$, and therefore the bound for strong condition is sub-quadratic.

Proof The major difference between $\rho(A)$ and $\bar{\rho}(A)$ is that the diagonal element increases can accumulate, whereas the accumulation of two off-diagonal element increases results in a reducible Schur complement, so further accumulation is impossible. Therefore, the diagonal element growth factor is bounded by the sum of n elements, each of which is bounded by Theorem 2.6. So we obtain the bound on $\rho(A)$ for the strong condition. Though this approach also gives a bound for the weak condition, a tighter bound can be obtained, as follows.

The proof of Theorem 2.5 shows that the off-diagonal element bound in the Schur complement depends on the number of rows/columns reduced. We follow the notation in the proof of Theorem 2.5.

If the weak condition holds, the off-diagonal elements $a_{ij}^{(k+1)}$ in $A^{(k+1)}$ (after reducing k rows/columns) are bounded as $|a_{ij}^{(k+1)}| \leq 2g^{\lfloor (k+1)/2 \rfloor} \max |a_{ij}|$ for $i \neq j$ and k from 1 to $n-2$. This is also the bound on the diagonal element increase of A_{k+1} from the previous iteration. We sum up all the relative element increases during the decomposition to obtain a bound on $\rho(A)$, where $A \in R^{n \times n}$ is symmetric triadic:

$$\begin{aligned} \rho(A) &\leq \underline{1} + 2g^{\lfloor 2/2 \rfloor} + 2g^{\lfloor 3/2 \rfloor} + \cdots + 2g^{\lfloor (n-1)/2 \rfloor} + \underline{2g^{\lfloor (n-1)/2 \rfloor + 1}} \\ &= \begin{cases} \frac{4g(g^{(n-3)/2}-1)}{g-1} + 2(g^{(n-1)/2} + g^{(n+1)/2}) + 1 & \text{if } n \text{ odd}, \\ \frac{4g(g^{(n-2)/2}-1)}{g-1} + 2g^{n/2} + 1 & \text{if } n \text{ even}. \end{cases} \end{aligned}$$

The first $\underline{1}$ is underlined because each diagonal element in the initial A can be $G(0)$. The reason for the last term $\underline{2g^{\lfloor (n-1)/2 \rfloor + 1}}$ is as follows.

If a 1×1 pivot is chosen in the last 2×2 Schur complement or a 2×2 pivot is chosen in the last 3×3 Schur complement, the reduction can still increase the very last diagonal element, but no off-diagonal element growth occurs there. Similarly, if (2.12) breaks, the reduced 2×2 block can exhibit diagonal element growth, but no off-diagonal element growth. This case is also taken into account in $\underline{2g^{\lfloor (n-1)/2 \rfloor + 1}}$.

In a similar vein, we can also obtain a slightly tighter bound for the strong condition, but it is also $O(n^{1+\lg g})$:

$$\rho(A) \leq 1 + 2g^{\lfloor \lg 2 \rfloor} + 2g^{\lfloor \lg 3 \rfloor} + \cdots + 2g^{\lfloor \lg (n-1) \rfloor} + 2g^{\lfloor \lg (n-1) \rfloor + 1} = O(n^{1+\lg g}). \quad \square$$

Now we investigate the attainability of the bounds on the growth factor in Theorem 2.6. If we choose $\alpha = \frac{\sqrt{5}-1}{2}$ to minimize g to be $\frac{\sqrt{5}+1}{2}$, then $\lg g \approx 0.694$, and therefore the bound for the strong condition is $O(n^{1.695})$, which is sub-quadratic.

Even linear growth is rare, but it is possible. For example, if

$$A = \begin{bmatrix} -1 & 1 & & & & 1 \\ 1 & -2 & 1 & & & \\ & 1 & -2 & \ddots & & \\ & & \ddots & \ddots & 1 & \\ & & & 1 & -2 & \end{bmatrix} \in R^{n\times n},$$

then $\rho(A) = \frac{n}{2} + O(1)$.

For the weak condition, exponential growth is achievable. For example,

$$A = \begin{bmatrix} -a & -1 & & 1 & & & & & & \\ -1 & -a & & & & & & & & 1 \\ & & -a & -1 & & 1 & & & & \\ 1 & & -1 & (g-1)a & & & & & & 0 \\ & & & & -a & -1 & & \ddots & & \\ & & 1 & & -1 & (g-1)a & & & & 0 \\ & & & & & & \ddots & \ddots & \vdots & \\ & & & & \ddots & & & \ddots & \ddots & 0 \\ & 1 & & 0 & & 0 & \cdots & 0 & 1 & \end{bmatrix} \in R^{n\times n},$$

where n is odd, $|a| < \alpha = \frac{\sqrt{5}-1}{2}$ and therefore $g = \frac{7+\sqrt{17}}{4}$. Applying the Bunch-Kaufman pivoting strategy, the pivots are all 2×2 without interchanging rows and columns. When $|a| \to \alpha^-$, the $(n, 2j)$ entry becomes g^{j-1} after $(j-1)$ 2×2 pivots for $j = 1, \ldots, \frac{n-1}{2}$ and therefore $\rho(A) = O(g^{n/2})$. The explicit zeros indicate where the growth is maximal. Despite these examples, in our experiments, the growth factor of the LBL^T factorization of symmetric tridiagonal matrices with optional additional corner elements is almost always bounded by a constant for any pivoting strategy in Section 2.2.

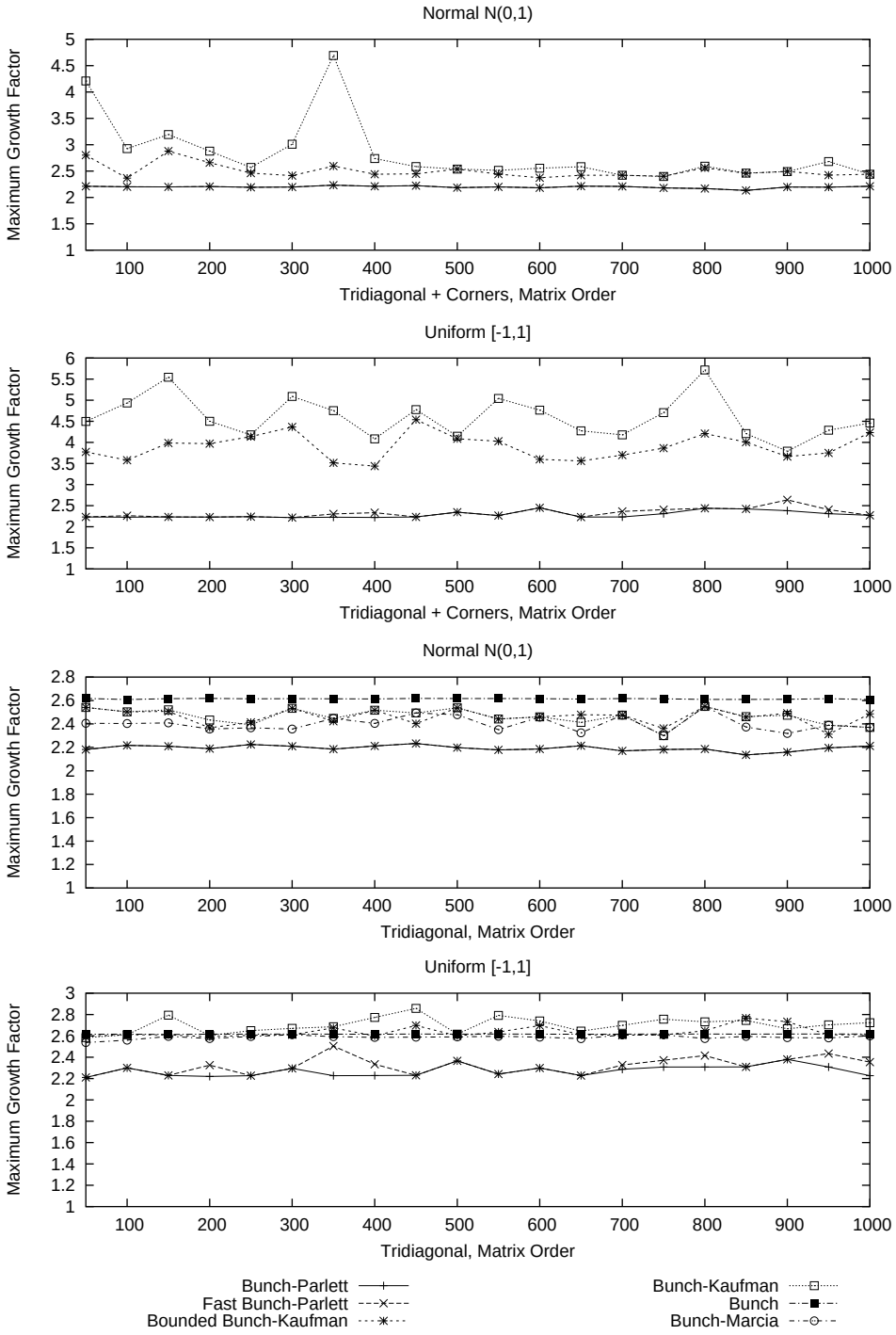

Figure 2.4: Experimental maximum growth factor for factoring a symmetric tridiagonal matrix with optional additional corner elements, 20,000 matrices for each method and matrix size.

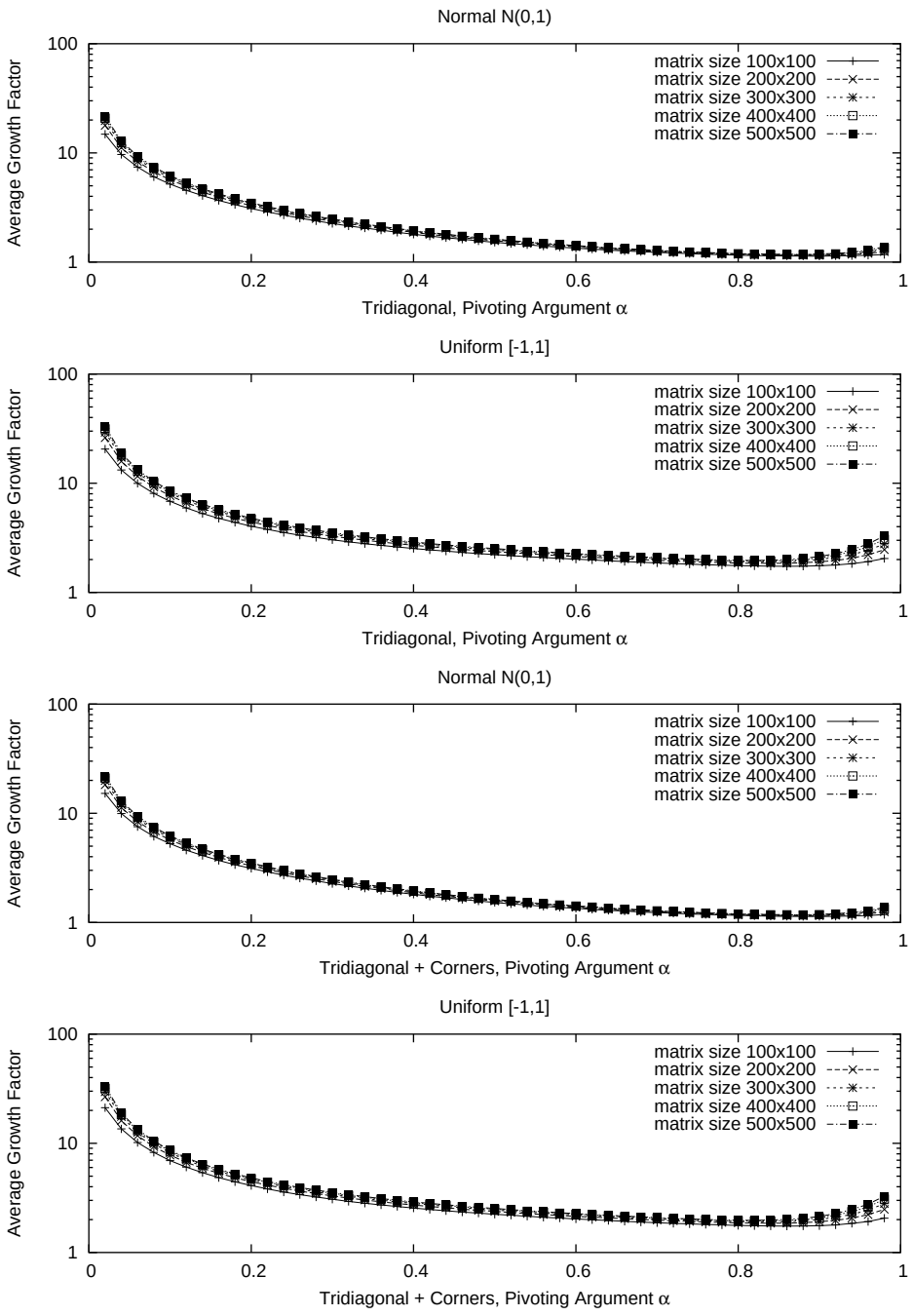

Figure 2.5: Experimental average growth factor for factoring a symmetric tridiagonal matrix with optional additional corner elements using the Bunch-Kaufman pivoting strategy, 20,000 matrices for each method and matrix size.

Figure 2.4 shows the maximum growth factor of $20,000$ random symmetric matrices for each $n = 50, 100, \ldots, 1000$ in each plot. Although $\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$ minimizes the bound on the relative element increase, our experiments show that the best α to minimize the average growth factor is usually between 0.82 and 0.86, as shown in Figure 2.5, where $20,000$ random matrices are generated for each matrix size and α.

2.3.3 Pivoting Cost

Now we discuss the pivoting cost for LBL^T factorizations of triadic matrices.

When the Bunch-Parlett algorithm is applied, it is natural to search the whole matrix instead of only the lower (or upper) triangular part due to the usual data structure for sparse matrices. So the number of comparisons is at most $3k+O(1)$ to select a pivot in a $k \times k$ Schur complement. Therefore, the total number of comparisons is bounded by $\frac{3}{2}n^2 + O(n)$ for a symmetric triadic $A \in R^{n\times n}$, which is more expensive than the $O(n)$ cost of the factorization.

The Bunch-Kaufman algorithm requires at most $5n + O(1)$ comparisons for a symmetric triadic $A \in R^{n\times n}$.

For the fast Bunch-Parlett and bounded Bunch-Kaufman pivoting strategies, the number of comparisons in worst cases is the same as that of Bunch-Parlett pivoting. The average number of element comparisons is between those for the Bunch-Kaufman and Bunch-Parlett pivoting strategies.

The Bunch and Bunch-Marcia pivoting selections specific to symmetric tridiagonal matrices require at most $2n + O(1)$ comparisons.

Figure 2.6 shows the average number of comparisons for $20,000$ symmetric matrices for each pivoting method and $n = 50, 100, \ldots, 1000$.

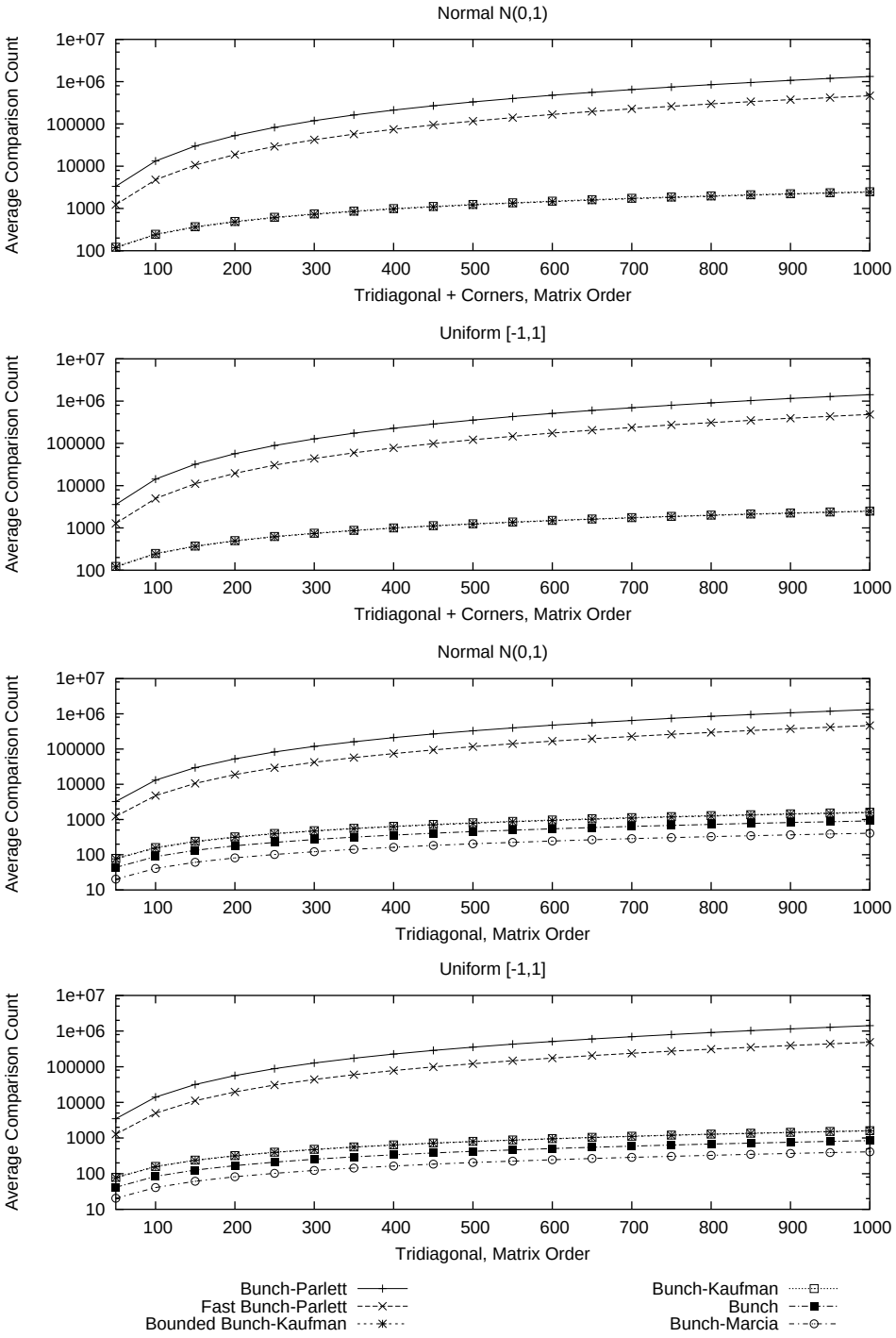

Figure 2.6: Experimental average number of comparisons to factor a symmetric tridiagonal matrix with optional corner elements, 20,000 matrices for each method and matrix size.

2.4 Perturbation Theory

Higham [38] gave the perturbation analysis of the Cholesky factorization of a positive semidefinite symmetric matrix with complete pivoting. In this section we analyze the perturbation of LDL^T and LBL^T factorizations.

Theorem 2.7 *Let $S_k(A)$ be the Schur complement appearing in an LL^T, LDL^T or LBL^T factorization of a symmetric matrix A after processing the first k columns and k rows, $k < n$. Suppose there is a symmetric perturbation in A, denoted by E. Partition A as*

$$A = \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix}$$

where $A_{11} \in R^{k\times k}$, and partition E accordingly. If both A_{11} and $A_{11} + E_{11}$ are nonsingular, then

$$S_k(A+E) - S_k(A) = E_{22} - (E_{21}W + W^T E_{21}^T) + W^T E_{11} W + O(\|E\|^2),$$

so

$$\|S_k(A+E) - S_k(A)\| \le \|E\|(1 + \|W\|^2)^2 + O(\|E\|^2),$$

where $W = A_{11}^{-1}A_{21}^T$, and $\|\cdot\|$ is a p-norm, ∞-norm or Frobenius norm.

Proof The factorization takes the form

$$A = \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & \\ L_{21} & I_{n-k} \end{bmatrix} \begin{bmatrix} X & \\ & S_k(A) \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & I_{n-k} \end{bmatrix},$$

where $L_{11} \in R^{k\times k}$ is lower triangular. The matrix X is either the identity, a diagonal matrix, or a block diagonal matrix with each block of order 1 or 2, depending on the factorization being LL^T, LDL^T, or LBL^T. In any case,

$A_{11} = L_{11}XL_{11}^T$ and $A_{21} = L_{21}XL_{11}^T$. Therefore, $W = A_{11}^{-1}A_{21}^T = L_{11}^{-T}L_{21}^T$ and $A_{22} = S_k(A) + L_{21}XL_{21}^T$. We also know that $S_k(A) = A_{22} - A_{21}A_{11}^{-1}A_{21}^T$ and $(A_{11} + E_{11})^{-1} = (I + A_{11}^{-1}E_{11})^{-1}A_{11}^{-1} = (I - A_{11}^{-1}E_{11})A_{11}^{-1} + O(\|E_{11}\|^2)$. The result is obtained by substituting the previous two equations into $S_k(A + E) = (A + E)_{22} - (A_{21} + E_{21})(A_{11} + E_{11})^{-1}(A_{21} + E_{21})^T$ and collecting the $O(\|E\|^2)$ terms. □

Theorem 2.7 shows that $W = A_{11}^{-1}A_{21}^T = L_{11}^{-T}L_{21}^T$ governs the sensitivity of $S_k(A)$ to perturbation for LL^T, LDL^T and LBL^T factorizations. For general symmetric matrices, a bound on $\|W\|_{2,F}$ is given in (2.17) obtained by Lemma 2.3. For symmetric triadic matrices, a bound on $\|W\|_{2,F}$ is given in Lemma 2.6, with proof via Lemmas 2.4 and 2.5.

Lemma 2.3 *If* $L = \begin{bmatrix} L_{11} & \\ L_{21} & L_{22} \end{bmatrix} \in R^{n\times n}$ *where* $L_{11} \in R^{k\times k}$ *is unit lower triangular with off-diagonal elements bounded in magnitude by* γ*, then*

$$|W| = |L_{11}^{-T}L_{21}^T| \leq \gamma y e^T \in R^{k\times(n-k)},$$

where $y = [(1+\gamma)^{k-1}, (1+\gamma)^{k-2}, \ldots, 1]^T$ *and* $e \in R^{n-k}$ *is a vector of ones.*

Proof Let $w = [w_1, \ldots, w_k]^T \in R^k$ be a column of W. Since the matrix W satisfies $L_{11}^T W = L_{21}^T$, $|w_i| \leq \gamma(1 + \sum_{j=i+1}^k |w_j|)$ for $i = 1, \ldots, k$. The solution to this recurrence relation is $|w_i| \leq \gamma(1+\gamma)^{k-i}$ for $i = 1, \ldots, k$, and the result follows. □

The consequent result of Lemma 2.3 is

$$\|W\|_{2,F} \leq \sqrt{\frac{\gamma}{\gamma+2}(n-k)((1+\gamma)^{2k} - 1)}\,. \tag{2.17}$$

For comparison, Higham [38] showed that with complete pivoting, the Cholesky factorization of a symmetric positive semidefinite matrix satisfies

$$\|W\|_{2,F} \leq \sqrt{\frac{1}{3}(n-k)(4^k-1)}\,,$$

a particular case of (2.17) with $\gamma = 1$ due to positive semidefiniteness and complete pivoting.

Lemma 2.4 *Let* $F_\gamma(n) = \sum_{i=1}^{\lceil n/2 \rceil} \binom{n-i}{i-1} \gamma^{n-i}$ *and* $\Phi_\gamma = \frac{1+\sqrt{1+4/\gamma}}{2}\gamma$. *Then*

$$\frac{1}{1+(1/\gamma)}\Phi_\gamma^{n-1} \leq F_\gamma(n) \leq \Phi_\gamma^{n-1}$$

for $n \in N$ *and* $\gamma > 0$.

Proof $F_\gamma(n)$ satisfies the recurrence relation $F_\gamma(n) = \gamma(F_\gamma(n-1) + F_\gamma(n-2))$ for $n > 2$ with base case $F_\gamma(1) = 1$ and $F_\gamma(2) = \gamma$. Note that $\gamma + \gamma\Phi_\gamma = \Phi_\gamma^2$. The result can be obtained by mathematical induction. □

Lemma 2.5 *Let* $C \in R^{m\times n}$ *be a matrix with all elements nonnegative and* $n \geq 2$. *Then* $\|C\|_p \leq \|C\hat{I}\|_p$, *where* $1 \leq p \leq \infty$ *or* $p = F$, *and* $\hat{I} \in R^{n\times(n-1)}$ *is the identity matrix of size* $n-1$ *with the last row repeated.*

Proof When $0 \leq p < \infty$, $\|C\|_p = \max_{\|x\|_p=1} \|Cx\|_p = \|Cz\|_p$ for some z with $\|z\|_p = 1$. Note that $z_i \geq 0$ for $i = 1, \ldots, n$, since all the elements of C are nonnegative.

Let $\hat{z} = [z_1, \ldots, z_{n-2}, \max(z_{n-1}, z_n)]^T$. Then $\|\hat{z}\|_p \leq 1$, and

$$\|C\|_p = \|Cz\|_p \leq \|C\hat{I}\hat{z}\|_p \leq \|C\hat{I}(\hat{z}/\|\hat{z}\|_p)\|_p \leq \max_{\|x\|_p=1} \|C\hat{I}x\|_p = \|C\hat{I}\|_p.$$

The cases of $p = F$ (Frobenius-norm) and $p = \infty$ (∞-norm) are trivial. □

Lemma 2.6 *The LBL^T factorization for symmetric triadic matrices has*

$$\|L_{11}^{-T}L_{21}^T\|_{2,F} \leq \frac{2\gamma\Phi_\gamma}{\Phi_\gamma - 1}\sqrt{\frac{\Phi_\gamma^{2k}-1}{\Phi_\gamma^2-1}} = O(\Phi_\gamma^k),$$

where $\gamma \geq 1$ is the off-diagonal element bound of L and $\Phi_\gamma = \frac{1+\sqrt{1+4/\gamma}}{2}\gamma$.

Proof The proof of Lemma 2.4 shows $F_\gamma(i) = \gamma(F_\gamma(i-1) + F_\gamma(i-2))$ for $i > 2$. Observing the elements in $L_{11}^{-1}L_{11} = I$, we obtain

$$|L_{11}^{-1}| \leq \sum_{i=1}^{k} F_\gamma(k)Z^{k-1} = \begin{bmatrix} F_\gamma(1) & & & & \\ F_\gamma(2) & F_\gamma(1) & & & \\ F_\gamma(3) & F_\gamma(2) & F_\gamma(1) & & \\ \vdots & \ddots & \ddots & \ddots & \\ F_\gamma(k) & \cdots & F_\gamma(3) & F_\gamma(2) & F_\gamma(1) \end{bmatrix}, \qquad (2.18)$$

where $Z \in R^{k\times k}$ is the shift-down matrix. Note that this bound is attainable with $L_{11} = I - \gamma Z - \gamma Z^2$. By Lemma 2.4,

$$|L_{11}^{-T}|e \leq [\frac{\Phi_\gamma^k - 1}{\Phi_\gamma - 1}, \frac{\Phi_\gamma^{k-1} - 1}{\Phi_\gamma - 1}, \ldots, 1]^T. \qquad (2.19)$$

Since L is triadic, each row of L_{21}^T has at most two nonzero elements. Let $|L_{21}^T| = R_1 + R_2$, where R_1 and R_2 contain the first and the second nonzero elements in each row, respectively. Then

$$\|L_{11}^{-T}L_{21}^T\| \leq \||L_{11}^{-T}||L_{21}^T|\| \leq \||L_{11}^{-T}|R_1\| + \||L_{11}^{-T}|R_2\|.$$

By Lemma 2.5,

$$\begin{aligned} \||L_{11}^{-T}|R_1\| &\leq \||L_{11}^{-T}|R_1\hat{I}_{n-k}\| \\ &\leq \||L_{11}^{-T}|R_1\hat{I}_{n-k}\hat{I}_{n-k-1}\| \\ &\leq \cdots \leq \||L_{11}^{-T}|R_1\hat{I}_{n-k}\hat{I}_{n-k-1}\cdots\hat{I}_2\| \\ &\leq \||L_{11}^{-T}|(\gamma e)\| = \gamma\||L_{11}^{-T}|e\|. \end{aligned}$$

Similarly, $\||L_{11}^{-T}|R_2\| \leq \gamma\||L_{11}^{-T}|e\|$. By (2.19),

$$\begin{aligned}
\|L_{11}^{-T}L_{21}^{T}\|_{2,F} &\leq 2\gamma\||L_{11}^{-T}|e\|_{2,F} \\
&\leq \frac{2\gamma}{\Phi_\gamma - 1}\sqrt{\Phi_\gamma^2\frac{\Phi_\gamma^{2k}-1}{\Phi_\gamma^2-1} - 2\Phi_\gamma\frac{\Phi_\gamma^{k}-1}{\Phi_\gamma-1} + k} \\
&\leq \frac{2\gamma\Phi_\gamma}{\Phi_\gamma - 1}\sqrt{\frac{\Phi_\gamma^{2k}-1}{\Phi_\gamma^2-1}}
\end{aligned}$$

for $\gamma \geq 1$. Note that this bound is halved when $n - k = 1$. □

As displayed in Table 2.1, in the LBL^T factorization of a symmetric triadic matrix with complete pivoting or rook pivoting, γ can be 2 or $\frac{7+\sqrt{17}}{8} \approx 2.781$, to minimize the element bound of matrix L or the growth factor, respectively.

2.5 Summary

We have studied various pivoting strategies in computing the LBL^T factorizations of symmetric triadic matrices. We denote the strategies as **Bunch** (Bunch), **BM** (Bunch-Marcia), **BP** (Bunch-Parlett), **FBP** (fast Bunch-Parlett), **BBK** (bounded Bunch-Kaufman), and **BK** (Bunch-Kaufman). The results are summarized as follows:

1. **BK**, **BBK**, **BP**, and **FBP** are applicable to general symmetric matrices, whereas **Bunch** and **BM** are specific for symmetric tridiagonal matrices.

2. **BBK**, **BP**, and **FBP** satisfy the strong condition, whereas **BK** satisfies the weak condition.

3. Both the strong and the weak conditions control the growth factor (see Section 2.2.4), but only the strong condition guarantees a bounded L (see Section 2.2.5).

4. The triadic structure is preserved in LL^T, LDL^T, and LBL^T factorizations with any diagonal pivoting strategy (see Theorem 2.3).

5. Previously, the pivoting parameter for general symmetric matrices was suggested to be $\alpha = \frac{1+\sqrt{17}}{8} \approx 0.640$. We presented a new choice $\alpha = 0.5$ that results in a sharper bound on the elements in L (see Table 2.1), and another $\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$ that better controls growth factor for triadic matrices (see Theorem 2.6).

6. For LDL^T factorization of a positive definite symmetric matrix A with complete pivoting, the magnification factor in the error bound for the Schur complement after k steps is bounded by $\sqrt{\frac{1}{3}(n-k)(4^k-1)}$ if A is full [40], and $O((\frac{1+\sqrt{5}}{2})^k)$ if A is triadic (see Lemma 2.6).

7. For two pivoting strategies $\mathcal{P}$ and $\mathcal{Q}$, we will say $\mathcal{P} \succ \mathcal{Q}$, $\mathcal{P} \succeq \mathcal{Q}$, and $\mathcal{P} \simeq \mathcal{Q}$, if $\mathcal{P}$ is better than, slightly better than, or similar to $\mathcal{Q}$, respectively. Our experimental results with pivoting argument $\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$ are as follows. For LBL^T factorizations of tridiagonal matrices, the maximum growth factors satisfy **BP** $\succeq$ **FBP** $\succ$ **BBK** $\succeq$ **BK** and **BM** $\succ$ **Bunch**, as shown in Figure 2.4, whereas the average number of comparisons satisfies **BM** $\succ$ **Bunch** $\succ$ **BK** $\simeq$ **BBK** $\succ$ **FBP** $\succ$ **BP**, as shown in Figure 2.6.

Chapter 3

Backward Error Analysis of Cholesky-Related Factorizations

Parts of this chapter are drawn from material in [22].

Positive semidefiniteness is required for Cholesky factorization. Quasidefiniteness guarantees the existence of the LDL^T factorization. LBL^T factorizations are for indefinite matrices.

LDL^T factorization is guaranteed for quasidefinite matrices that arise in regularized linear programming and regularized least norm problems. Gill, Saunders and Shinnerl [29] analyzed the backward stability by transforming a quasidefinite matrix to an unsymmetric positive definite matrix. We prove that an LDL^T factorization is numerically backward stable, as long as the growth factor is modest.

For indefinite symmetric matrices, we use the LBL^T factorization, where pivoting is incorporated for stability. We give a condition under which LBL^T factorization will run to completion in inexact arithmetic with inertia preserved. In addition, we present a new proof of the componentwise backward stability of the factorization using the inner product formulation, whereas other proofs in the literature replied on the outer product formulation.

Slapničar analyzed the Bunch-Parlett pivoting strategy [58, Section 7]. The methods of Bunch-Kaufman, Bunch, and Bunch-Marcia may lead to unbounded L. Hence the reliability could be questioned. Nevertheless, Higham proved the stability of the Bunch-Kaufman pivoting strategy [39] and Bunch's method [40]. His proofs consist of componentwise backward error analysis using an outer product formulation and normwise analysis. In a similar vein, Bunch and Marcia proved the normwise backward stability of their method. All these componentwise analyses relied on the outer product formulation. In this chapter, we present a new proof of componentwise backward stability using an inner product formulation. In addition, we give a sufficient condition under which the LBL^T factorization of a symmetric matrix is guaranteed to run to completion numerically and preserve inertia.

With complete pivoting, Cholesky factorization can be applied to rank estimation. The stability is analyzed by Higham [38]. Given $A \in R^{n\times n}$ of rank r, the LU factorization needs $2(3n-2r)r^2/3$ flops, whereas Cholesky factorization needs $(3n-2r)r^2/3$ flops but requires symmetric positive semidefiniteness. To estimate the rank of a symmetric indefinite matrix, we can use LBL^T factorization with the Bunch-Parlett (complete pivoting), fast Bunch-Parlett or bounded Bunch-Kaufman (rook pivoting) pivoting strategy. The number of required flops is still $(3n-2r)r^2/3$. In this chapter we analyze the stability, generalizing results of Higham for the symmetric semidefinite case with complete pivoting. Moreover, we display the improvement in stability bounds when the matrix is triadic.

Section 3.1 and Section 3.2 give the componentwise backward error analysis of LDL^T and LBL^T factorizations, and the application to solve symmetric linear systems, respectively. In Section 3.3 we discuss the stability using normwise

analysis. Section 3.4 analyzes rank estimation for symmetric indefinite matrices by LBL^T factorization with the Bunch-Parlett, fast Bunch-Parlett, or bounded Bunch-Kaufman pivoting, as well as rank estimation for positive definite or diagonally dominant matrices by LDL^T factorization with complete pivoting. Section 3.5 gives the concluding remarks for this chapter.

Throughout the chapter, without loss of generality, we assume the required interchanges for any diagonal pivoting are done prior to the factorization, so that $A := PAP^T$, where P is the permutation matrix for pivoting.

3.1 Componentwise Analysis

The stability of Cholesky factorization in LL^T form, which requires a positive definite or semidefinite matrix, is well studied in [38] and [41, Chapter 10]. In this chapter, we focus on LDL^T factorization and LBL^T factorization. The improvement of the stability because of the triadic structure is also discussed. We begin with basics for rounding error analysis.

3.1.1 Floating Point Arithmetic

We use $fl(\cdot)$ to denote the computed value of a given expression, and follow the standard model

$$fl(x\ op\ y) = (x\ op\ y)(1+\delta),\ \text{for } |\delta| \le u \text{ and } op = +, -, \times, /,$$

where u is the unit roundoff. This model holds in most computers, including those using IEEE standard arithmetic. Lemma 3.1 gives the basic tool for rounding error analysis [41, Lemma 3.1].

Lemma 3.1 *If* $|\delta_i| \le u$ *and* $\sigma_i = \pm 1$ *for* $i = 1, \ldots, k$ *then if* $ku < 1$,

$$\prod_{i=1}^{k}(1+\delta_i)^{\sigma_i} = 1 + \theta_k,\ |\theta_k| \le \epsilon_k,$$

where

$$\epsilon_k = \frac{ku}{1-ku}.$$

The function ϵ_k defined in Lemma 3.1 has two useful properties[1]:

$$\epsilon_m + \epsilon_n + 2\epsilon_m\epsilon_n \le \epsilon_{m+n} \text{ for } m, n \ge 0,$$

and

$$c\epsilon_n \le \epsilon_{cn} \text{ for } c \ge 1.$$

Since we assume $ku < 1$ for all practical k,

$$\epsilon_k = ku + ku\epsilon_k = ku + O(u^2).$$

These properties are used frequently to derive inequalities in this chapter.

3.1.2 LDL^T Factorization

An LDL^T factorization is guaranteed for symmetric positive semidefinite matrices [38], symmetric diagonally dominant matrices [19], and symmetric quasidefinite matrices [29, 62]. We now investigate its stability. The factorization is denoted by $A = LDL^T \in R^{n\times n}$, where $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$ and the (i,j) entries of A and L are a_{ij} and l_{ij}, respectively. Note that $a_{ij} = a_{ji}$ and $l_{ij} = 0$ for all $1 \le i < j \le n$, and $l_{ii} = 1$ for $1 \le i \le n$. Algorithm 3.1 is computationally equivalent to the LDL^T factorization in inner product form[2].

[1]The two properties were listed in [39] and [41, Lemma 3.3], but with '$2\epsilon_m\epsilon_n$' replaced by '$\epsilon_m\epsilon_n$'. Here we give a slightly tighter inequality.

[2]In practice, we store the values of $d_k l_{ik}$ in an array for the computations in (*). Hence the factorization requires only $n^3/3$ flops, the same as Cholesky factorization. This remark also applies to Algorithm 3.3.

Algorithm 3.1 LDL^T factorization of $A \in R^{n\times n}$ in inner product form.

for $i = 1, \ldots, n$ **do**

for $j = 1, \ldots, i-1$ **do**

$(*)$ $l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} d_k l_{ik} l_{jk})/d_j$

end for

$(**)$ $d_i = a_{ii} - \sum_{k=1}^{i-1} d_k l_{ik}^2$

end for

For a positive definite symmetric matrix $A = LL^T \in R^{n\times n}$,

$$|A - \hat{L}\hat{L}^T| \leq \epsilon_{n+1}|\hat{L}||\hat{L}^T|,$$

where we assume the Cholesky factorization runs to completion and $\hat{L}$ is the computed version of L [41, Theorem 10.3]. Here and throughout this chapter, we use a hat to denote computed quantities, inequality and absolute value for matrices are defined elementwise, and A is a symmetric matrix with floating point numbers.

Due to potential rounding errors in forming or storing A, A may not be exactly the same as the matrix under consideration. Hence we write

$$A = \widetilde{A} + \Delta\widetilde{A},$$

where $\widetilde{A}$ is the matrix under consideration. If each element in A is rounded to the closest floating point number, then $|a_{ij} - \tilde{a}_{ij}| \leq u\tilde{a}_{ij}$ and therefore

$$|A - \widetilde{A}| \leq u|\widetilde{A}| \leq \frac{u}{1-u}|A| = \epsilon_1|A|.$$

The overall backward error of the Cholesky factorization is

$$|\widetilde{A} - \hat{L}\hat{L}^T| \leq |\widetilde{A} - A| + |A - \hat{L}\hat{L}^T| \leq \epsilon_1|A| + \epsilon_{n+1}|\hat{L}||\hat{L}^T|.$$

In general, we assume A is close to $\widetilde{A}$ and therefore $\Delta\widetilde{A}$ is small relative to A. For simplicity, we usually consider only $|A - \hat{L}\hat{L}^T|$. For the same reason, we consider only $|A - \hat{L}\hat{D}\hat{L}^T|$ and $|A - \hat{L}\hat{B}\hat{L}^T|$ for LDL^T and LBL^T factorizations, respectively. This remark also applies to the backward error analysis of solving symmetric linear systems (discussed in Section 3.2) and rank estimation (discussed in Section 3.4), using the LDL^T or LBL^T factorization.

We begin with developing a bound on $|A - \hat{L}\hat{D}\hat{L}^T|$ for LDL^T factorization given in Theorem 3.1 with proof via Lemma 3.2. The result is extended to LBL^T factorization in Section 3.1.3.

Lemma 3.2 *Let $y = (s - \sum_{k=1}^{n-1} a_k b_k c_k)/d$. No matter what the order of evaluation, the computed $\hat{y}$ satisfies*

$$\hat{y}d + \sum_{k=1}^{n-1} a_k b_k c_k = s + \Delta s,$$

where

$$|\Delta s| \leq \epsilon_n(|\hat{y}d| + \sum_{k=1}^{n-1} |a_k b_k c_k|).$$

Proof The proof is analogous to that of [41, Lemma 8.4]. Using Lemma 3.1, one may obtain

$$\hat{y}d(1 + \theta_n^{(0)}) = s - \sum_{k=1}^{n-1} a_k b_k c_k (1 + \theta_n^{(k)}),$$

where $|\theta_n^{(k)}| \leq \epsilon_n$ for $k = 0, 1, \ldots, n-1$. The result follows immediately. □

Theorem 3.1 *If the LDL^T factorization of a symmetric matrix $A \in R^{n\times n}$ runs to completion, then the computed $\hat{L}\hat{D}\hat{L}^T$ satisfies*[3]

$$|A - \hat{L}\hat{D}\hat{L}^T| \leq \epsilon_n |\hat{L}||\hat{D}||\hat{L}^T|.$$

[3]Slapničar [58] presented a comparable bound using the factorization in outer product form.

Proof By Lemma 3.2, no matter what the order of evaluation in $(*)$ and $(**)$ in Algorithm 3.1,

$$|a_{ij} - \sum_{k=1}^{j} \hat{d}_k \hat{l}_{ik} \hat{l}_{jk}| \leq \epsilon_j \sum_{k=1}^{j} |\hat{d}_k \hat{l}_{ik} \hat{l}_{jk}| \tag{3.1}$$

for $1 \leq j \leq i \leq n$, where we define $\hat{l}_{ii} := 1$ for $i = 1, \ldots, n$ to simplify the notation. The rest of the proof is by collecting terms in (3.1) into one matrix presentation. $\square$

Theorem 3.1 shows that an LDL^T factorization is stable if $|\hat{L}||\hat{D}||\hat{L}^T|$ is suitably bounded relative to A. However, even with pivoting, the LDL^T factorization of a given symmetric matrix may not exist (e.g., $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$), and $|\hat{L}||\hat{D}||\hat{L}^T|$ could be catastrophically large. It is a well-known fact that LDL^T factorization is not generally stable. See Section 3.3.1 for a sufficient condition for the stability of LDL^T factorization.

The LDL^T factorization of a symmetric triadic matrix has L triadic, but the last row in L can be full[4]. Therefore, the bounding coefficient ϵ_n in Theorem 3.1 cannot be reduced with the triadic structure. Instead, we bound C defined by

$$|A - \hat{L}\hat{D}\hat{L}^T| = C \circ (|\hat{L}||\hat{D}||\hat{L}^T|), \tag{3.2}$$

where $\circ$ denotes *Hadamard* (elementwise) product. Let $\|C\|_S = \sum_{i,j} |c_{ij}|$, where c_{ij} denotes the (i, j) entry of C.

To show the improvement of stability because of the triadic structure, we compare $\|C\|_S$ for a general symmetric matrix with that for a symmetric triadic matrix.

[4]See the example with the circular shift down matrix (2.1) in Section 2.1.

By (3.1), we obtain $c_{ij} = c_{ji} = \epsilon_j$ for $1 \le j \le i \le n$. Therefore,

$$|A - \hat{L}\hat{D}\hat{L}^T| \le \begin{bmatrix} \epsilon_1 & \epsilon_1 & \epsilon_1 & \cdots & \epsilon_1 \\ \epsilon_1 & \epsilon_2 & \epsilon_2 & \cdots & \epsilon_2 \\ \epsilon_1 & \epsilon_2 & \epsilon_3 & \cdots & \epsilon_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \epsilon_1 & \epsilon_2 & \epsilon_3 & \cdots & \epsilon_n \end{bmatrix} \circ (|\hat{L}||\hat{D}||\hat{L}^T|). \tag{3.3}$$

Then

$$\begin{aligned} \|C\|_S &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} = \sum_{i=0}^{n-1}(2i+1)\epsilon_{n-i} \\ &\le \sum_{i=0}^{n-1} \epsilon_{(2i+1)(n-i)} \le \epsilon_{\sum_{i=0}^{n-1}(2i+1)(n-i)} \\ &= \epsilon_{\frac{1}{6}n(n+1)(2n+1)} = \frac{1}{6}(2n^3+3n^2+n)u + O(u^2). \end{aligned} \tag{3.4}$$

Note that (3.3) is approximately tight allowing any order of evaluation in $(*)$ and $(**)$ in Algorithm 3.1. Before investigating the case where A is a symmetric triadic matrix, we introduce the following lemma.

Lemma 3.3 *For any triadic and lower triangular matrix $L \in R^{n\times n}$, LL^T has at most $7n-14$ nonzero elements for $n \ge 4$, evaluated using at most $9n-13$ nonzero terms. The bound, $7n-14$, is attained by $L = Z^3 + Z + I$, where $Z \in R^{n\times n}$ is the shift-down matrix*[5].

Proof Let $L = \tilde{L} + \tilde{D}$, where $\tilde{L}$ and $\tilde{D}$ are the off-diagonal and the diagonal part of L, respectively. Then $LL^T = (\tilde{L}+\tilde{D})(\tilde{L}+\tilde{D})^T = \tilde{L}\tilde{L}^T + \tilde{L}\tilde{D} + \tilde{D}\tilde{L}^T + \tilde{D}^2$, in which $\tilde{L}\tilde{D}$ contributes at most $2n-3$ nonzero elements in the lower triangular

[5]Note that though LL^T remains sparse, L^TL can be full (e.g., when the elements in the last column of L are all nonzero and elsewhere zero).

part. Now we inspect $\tilde{L}\tilde{L}^T$, in which each column of $\tilde{L}$ multiplying $\tilde{L}^T$ may contribute one off-diagonal element in the lower triangular part, except the last two columns of $\tilde{L}$. Therefore, $\tilde{L}\tilde{L}^T$ contributes at most $n-2$ nonzero off-diagonal elements in the lower triangular part. There are at most $(2n-3)+(n-2) = 3n-5$ off-diagonal terms in the lower triangular part. However, the two nonzero off-diagonal elements contributed by the third to last and fourth to last columns in $\tilde{L}$ must be in the bottom-rightmost 3×3 block of LL^T, which have collisions if the bottom-rightmost 3×3 block of $\tilde{L}$ is full. As a result, there are at most $(3n-5)-2 = 3n-7$ nonzero off-diagonal elements in the lower triangular part for $n \geq 4$. Along with the n diagonal elements, there are at most $2(3n-7)+n = 7n-14$ nonzero elements for $n \geq 4$. Note that $\tilde{D}^2$ and $\tilde{L}\tilde{L}^T$ can contribute n and $2n-3$ nonzero terms to the diagonal of LL^T, respectively. Overall, there are at most $2(3n-5)+(3n-3) = 9n-13$ nonzero terms for $n \geq 4$. □

If $A \in R^{n\times n}$ is symmetric triadic, then A has at most $3n$ nonzero elements. and so does its factorization LDL^T (or LL^T). However, because of rounding errors, the computed $\hat{L}\hat{D}\hat{L}^T$ (or $\hat{L}\hat{L}^T$) may have more nonzero elements than A. Nevertheless, by Lemma 3.3, the number of nonzero elements in $\hat{L}\hat{L}^T$ or $\hat{L}\hat{D}\hat{L}^T$ is bounded by $7n-14$ for $n \geq 4$.

For $1 \leq j \leq i \leq n$, c_{ij} depends on the number of nonzero terms $\hat{d}_k\hat{l}_{ik}\hat{l}_{jk}$ in (3.1). By Lemma 3.3, there are at most $9n-13$ nonzero terms in $\hat{L}\hat{D}\hat{L}^T$ for $n \geq 4$. Therefore,

$$\|C\|_S = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \leq \epsilon_{9n-13} = 9nu + O(u^2). \tag{3.5}$$

Comparing (3.5) with (3.4), we see the improvement of componentwise backward error because of the triadic structure. Note that the analysis is independent of the order of evaluation in $(*)$ and $(**)$ in Algorithm 3.1.

3.1.3 LBL^T Factorization

Now we analyze the LBL^T factorization of a symmetric and possibly indefinite matrix $A \in R^{n\times n}$. The factorization is denoted by $A = LBL^T \in R^{n\times n}$, where

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_m \end{bmatrix} \in R^{n\times n}$$

and

$$L = \begin{bmatrix} L_{11} & & & \\ L_{21} & L_{22} & & \\ \vdots & & \ddots & \\ L_{m1} & L_{m2} & \cdots & L_{mm} \end{bmatrix} \in R^{n\times n}.$$

Each B_i is a 1×1 or 2×2 block, with $L_{ii} = 1$ or $L_{ii} = I_2$, respectively. The rest of L is partitioned accordingly. Algorithm 3.2 is computationally equivalent to the LBL^T factorization in inner product form[6].

Algorithm 3.2 LBL^T factorization of $A \in R^{n\times n}$ in inner product form.

for $i = 1, \ldots, m$ **do**

for $j = 1, \ldots, i-1$ **do**

(*) $L_{ij} = (A_{ij} - \sum_{k=1}^{j-1} L_{ik}B_kL_{jk}^T)B_j^{-1}$

end for

(**) $B_i = A_{ii} - \sum_{k=1}^{i-1} L_{ik}B_kL_{ik}^T$

end for

[6]In practice, we store the values of $L_{ik}B_k$ in an array for the computations in (*). Hence the factorization requires only $n^3/3$ flops, the same as Cholesky factorization. This remark also applies to Algorithm 3.4.

In Algorithm 3.2, each multiplication by B_j^{-1} in $(*)$ with $B_j \in R^{2\times 2}$ can be computed by solving a 2×2 linear system, denoted by $Ey = z$ (i.e., $E := B_j$). We assume the linear system is solved successfully with computed $\hat{y}$ satisfying

$$|\Delta E| \leq \epsilon_c |E|, \text{ where } (E + \Delta E)\hat{y} = z \tag{3.6}$$

for some constant ϵ_c.

Higham [39] showed that with Bunch-Kaufman pivoting with pivoting argument $\alpha = \frac{1+\sqrt{17}}{8} \approx 0.640$, if the system is solved by GEPP, then $\epsilon_c = \epsilon_{12}$; if it is solved by explicit inverse of E with scaling (as implemented in both LAPACK and LINPACK), $\epsilon_c = \epsilon_{180}$. In a similar vein, the assumption (3.6) also holds with the other suggested pivoting argument $\alpha = 0.5$ to minimize the elements in magnitude in L (see Table 2.1) and $\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$ for triadic matrices (see Theorem 2.6).

Higham's analysis [39] also applies to the variant by Sorensen and Van Loan [21, Section 5.3.2] (see (2.9) for the change they made). Since Bunch-Parlett, fast Bunch-Parlett and bounded Bunch-Kaufman pivoting strategies satisfy stronger conditions than the Bunch-Kaufman, condition (3.6) still holds.

Higham [40] also showed that with Bunch's pivoting strategy for symmetric tridiagonal matrices [10], if a 2×2 linear system is solved by GEPP, then $\epsilon_c = \epsilon_{6\sqrt{5}}$. For the 2×2 linear system solved by explicit inverse with scaling, a constant ϵ_c can also be obtained. For the Bunch-Marcia algorithm, $\epsilon_c = \epsilon_{500}$ with the 2×2 linear system solved by explicit inverse with scaling [13].

We conclude that all pivoting strategies in the literature [5, 10, 12, 13, 14] satisfy condition (3.6).

Lemma 3.4 *Let $Y = (S - \sum_{k=1}^{m-1} A_k B_k C_k)E^{-1}$, where E and each B_k are either 1×1 or 2×2, such that the matrix operations are well-defined. If E is a 2×2*

matrix, we assume condition (3.6) holds. Then the computed $\hat{Y}$ *satisfies*

$$\hat{Y}E + \sum_{k=1}^{m-1} A_k B_k C_k = S + \Delta S,$$

where

$$|\Delta S| \leq \max\{\epsilon_c, \epsilon_{4m-3}\}(|S| + |\hat{Y}||E| + \sum_{k=1}^{m-1} |A_k||B_k||C_k|).$$

If E *is an identity, then* $\max\{\epsilon_c, \epsilon_{4m-3}\}$ *can be replaced by* ϵ_{4m-3}*, since* $\epsilon_c = 0$*.*

Proof Let $Z = S - \sum_{k=1}^{m-1} A_k B_k C_k$ and hence $Y = ZE^{-1}$. If B_k is a 2×2 matrix, then each element in $A_k B_k C_k$ can be represented in the form $\sum_{i=1}^{4} a_i b_i c_i$. Therefore, each element in $\sum_{k=1}^{m-1} A_k B_k C_k$ sums at most $4(m-1)$ terms. By Lemma 3.2,

$$|\Delta Z| \leq \epsilon_{4m-3}(|S| + \sum_{k=1}^{m-1} |A_k||B_k||C_k|), \text{ where } \hat{Z} = Z + \Delta Z. \tag{3.7}$$

We use $X^{(i)}$ to denote the row vector formed by the ith row of matrix X. If E is a 2×2 matrix, then applying (3.6) by substituting $y = \hat{Y}^{(i)}$ and $z = \hat{Z}^{(i)}$, we obtain

$$\hat{Y}^{(i)}(E + \Delta E_i) = \hat{Z}^{(i)}, \text{ where } |\Delta E_i| \leq \epsilon_c |E| \tag{3.8}$$

for $i = 1$ and $i = 2$ (if any). By (3.7) and (3.8),

$$|\Delta S| = |\hat{Y}E - (S - \sum_{k=1}^{m-1} A_k B_k C_k)| = |\hat{Y}E - \hat{Z} + \hat{Z} - Z| = |\hat{Y}E - \hat{Z} + \Delta Z|,$$

and then

$$\begin{aligned} |\Delta S^{(i)}| &= |-\hat{Y}^{(i)}\Delta E_i + \Delta Z^{(i)}| \leq |\hat{Y}^{(i)}||\Delta E_i| + |\Delta Z^{(i)}| \\ &\leq \epsilon_c |\hat{Y}^{(i)}||E| + \epsilon_{4m-3}(|S^{(i)}| + \sum_{k=1}^{m-1} |A_k^{(i)}||B_k||C_k|) \\ &\leq \max\{\epsilon_c, \epsilon_{4m-3}\}(|S^{(i)}| + |\hat{Y}^{(i)}||E| + \sum_{k=1}^{m-1} |A_k^{(i)}||B_k||C_k|). \end{aligned}$$

Combining the cases $i = 1, 2$ (if any) we obtain the result for E being 2×2.

If E is a 1×1 matrix, then we apply Lemma 3.2 and obtain

$$|\Delta S| \leq \epsilon_{4m-3}(|\hat{Y}||E| + \sum_{k=1}^{m-1} |A_k||B_k||C_k|). \quad \square$$

Theorem 3.2 *If the LBL^T factorization of a symmetric matrix $A \in R^{n\times n}$ runs to completion, then the computed $\hat{L}\hat{B}\hat{L}^T$ satisfies*[7]

$$|A - \hat{L}\hat{B}\hat{L}^T| \leq \max\{\epsilon_c, \epsilon_{4m-3}\}(|A| + |\hat{L}||\hat{B}||\hat{L}^T|),$$

where we assume condition (3.6) holds for all linear systems involving 2×2 pivots, and m is the number of blocks in B, $m \leq n$.

Proof Applying Lemma 3.4 to $(*)$ and $(**)$ in Algorithm 3.2, we obtain

$$|A_{ij} - \hat{L}_{ij}\hat{B}_j - \sum_{k=1}^{j-1} \hat{L}_{ik}\hat{B}_k\hat{L}_{jk}^T| \leq \max\{\epsilon_c, \epsilon_{4j-3}\}(|A_{ij}| + |\hat{L}_{ij}||\hat{B}_j| + \sum_{k=1}^{j-1} |\hat{L}_{ik}||\hat{B}_k||\hat{L}_{jk}^T|).$$

Therefore,

$$|A_{ij} - \sum_{k=1}^{j} \hat{L}_{ik}\hat{B}_k\hat{L}_{jk}^T| \leq \max\{\epsilon_c, \epsilon_{4j-3}\}(|A_{ij}| + \sum_{k=1}^{j} |\hat{L}_{ik}||\hat{B}_k||\hat{L}_{jk}^T|) \tag{3.9}$$

for $1 \leq j \leq i \leq m$, where $\hat{L}_{jj} = 1$ or I_2, depending on whether B_j is 1×1 or 2×2 for $i = 1, \ldots, m$. The result is obtained by collecting terms in (3.9) into one matrix presentation. $\square$

Similar to the coefficient ϵ_n in Theorem 3.1 for LDL^T factorization, the bounding coefficient $\max\{\epsilon_c, \epsilon_{4m-3}\}$ in Theorem 3.2 for LBL^T factorization can hardly be reduced when triadic structure is present. Instead, we bound $\|C\|_S$, where

$$|A - \hat{L}\hat{B}\hat{L}^T| = C \circ (|A| + |\hat{L}||\hat{B}||\hat{L}^T|). \tag{3.10}$$

[7]Using the outer product formulation, Higham [39, Theorem 4.1][41, Theorem 11.3] and Slapničar [58, Theorem 7.1] gave bounds of the same order for Bunch-Kaufman and Bunch-Parlett pivoting strategies, respectively.

Each c_{ij} depends on the number of blocks before itself, which is at most $j-1$ for $1 \le j \le i \le n$. By (3.9), $c_{ij} \le \max\{\epsilon_c, \epsilon_{4j-3}\}$ for $1 \le j \le i \le n$. Therefore,

$$\begin{aligned} \|C\|_S &\le \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \\ &= n^2\epsilon_c + \sum_{i=0}^{n-1}(2i+1)\epsilon_{4(n-i)-3} + O(u^2) \\ &\le n^2\epsilon_c + \epsilon_{\frac{1}{3}n(4n^2-3n+2)} + O(u^2) \\ &= \frac{1}{3}(4n^3 + 3(c-1)n^2 + 2n)u + O(u^2). \end{aligned} \tag{3.11}$$

Because of the rounding errors, the computed $\hat{L}\hat{B}\hat{L}^T$ of a symmetric triadic matrix may have more nonzero elements than LBL^T. The triadic structure is preserved in the L (see Section 2.1). By Lemma 3.3, the number of nonzero blocks in $\hat{L}\hat{B}\hat{L}^T$ is bounded by $7m-14$, and the proof shows that there are at most $9m-13$ block terms for $m \ge 4$, where m is the number of blocks in B. By (3.9), for $m \ge 4$,

$$\|C\|_S \le 4((7m-14)\epsilon_c + \epsilon_{4(9m-13)}) \le 4(7c+36)nu + O(u^2). \tag{3.12}$$

Therefore, $\hat{L}\hat{B}\hat{L}^T$ will still be sparse, but not necessarily triadic.

Comparing (3.12) with (3.11), we see the improvement of componentwise backward error because of the triadic structure. Note that the analysis is independent of the order of evaluation in $(*)$ and $(**)$ in Algorithm 3.2.

3.2 Solving Symmetric Linear Systems

In this section we use LBL^T factorization to solve a symmetric linear system $Ax = b$. After a possible permutation, which is omitted for notational convenience, we obtain $LBL^Tx = b$. Then we may solve three simplified systems, $Ly = b$ for y, $Bz = y$ for z, and $L^Tx = z$ for x.

If A is triadic, then each column of L has at most two off-diagonal elements and we can solve $Ly = b$ and $L^T x = z$, traversing columns of L.

3.2.1 LDL^T Factorization

The computed solution $\hat{x}$ to an $n \times n$ symmetric positive definite system $Ax = b$ using LL^T factorization satisfies [41, Theorem 10.4],

$$(A + \Delta A)\hat{x} = b,\ |\Delta A| \leq \epsilon_{3n+1}|\hat{L}||\hat{L}^T|.$$

Theorem 3.3 gives this bound for LDL^T factorization, with proof via Lemmas 3.5 and 3.6. The result is extended to LBL^T factorization in Section 3.2.2.

Lemma 3.5 *Let $\hat{y}$ be the computed solution to the lower triangular system $Ly = b$ by forward substitution with any ordering of arithmetic operations, where $L \in R^{n\times n}$ is nonsingular. Then*

$$(L + \Delta L)\hat{y} = b,\ |\Delta L| \leq \epsilon_n |L|.$$

If L is unit lower triangular, then there is no division so $|\Delta L| \leq \epsilon_{n-1}|L|$. The bounds for upper triangular systems are the same.

Proof Similar to the derivation leading to [41, Theorem 8.5]. □

Lemma 3.6 *For any $m, n, k > 0$ with $m + n + k < 1/u$,*

$$\epsilon_m + \epsilon_n + \epsilon_k + \epsilon_m\epsilon_n + \epsilon_n\epsilon_k + \epsilon_m\epsilon_k + \epsilon_m\epsilon_n\epsilon_k \leq \epsilon_{m+n+k}.$$

Proof Without loss of generality, let $k \leq m$. Then

$$\begin{aligned} & \epsilon_m + \epsilon_n + \epsilon_k + \epsilon_m\epsilon_n + \epsilon_n\epsilon_k + \epsilon_m\epsilon_k + \epsilon_m\epsilon_n\epsilon_k \\ \leq\ & (\epsilon_m + \epsilon_n + 2\epsilon_m\epsilon_n) + \epsilon_k + \epsilon_m\epsilon_k + \epsilon_m\epsilon_n\epsilon_k \\ \leq\ & \epsilon_{m+n} + \epsilon_k + \epsilon_{m+n}\epsilon_k + \epsilon_{m+n}\epsilon_k \leq \epsilon_{m+n+k}. \quad \square \end{aligned}$$

Theorem 3.3 *Suppose the LDL^T factorization of a symmetric matrix $A \in R^{n\times n}$ runs to completion and produces a computed solution $\hat{x}$ to $Ax = b$. Then*

$$(A + \Delta A)\hat{x} = b,\ |\Delta A| \leq \epsilon_{3n-1}|\hat{L}||\hat{D}||\hat{L}^T|.$$

Proof By Theorem 3.1, $A + \Delta A_1 = \hat{L}\hat{D}\hat{L}^T$ with $|\Delta A_1| \leq \epsilon_n|\hat{L}||\hat{D}||\hat{L}^T|$. By Lemma 3.5,

$$\begin{aligned}
(\hat{L} + \Delta L)\hat{y} = b \quad &, \quad |\Delta L| \leq \epsilon_{n-1}|\hat{L}|, \\
(\hat{D} + \Delta D)\hat{z} = \hat{y} \quad &, \quad |\Delta D| \leq \epsilon_1|\hat{D}|, \qquad (3.13) \\
(\hat{L}^T + \Delta R)\hat{x} = \hat{z} \quad &, \quad |\Delta R| \leq \epsilon_{n-1}|\hat{L}^T|. \qquad (3.14)
\end{aligned}$$

Then

$$\begin{aligned}
b &= (\hat{L} + \Delta L)(\hat{D} + \Delta D)(\hat{L}^T + \Delta R)\hat{x} \\
&= (\hat{L}\hat{D}\hat{L}^T + \Delta L\hat{D}\hat{L}^T + \hat{L}\Delta D\hat{L}^T + \hat{L}\hat{D}\Delta R + \\
&\quad + \hat{L}\Delta D\Delta R + \Delta L\hat{D}\Delta R + \Delta L\Delta D\hat{L}^T + \Delta L\Delta D\Delta R)\hat{x}.
\end{aligned}$$

Since $\hat{L}\hat{D}\hat{L}^T = A + \Delta A_1$,

$$\begin{aligned}
|\Delta A| &= |\Delta A_1 + \Delta L\hat{D}\hat{L}^T + \hat{L}\Delta D\hat{L}^T + \hat{L}\hat{D}\Delta R + \\
&\quad + \hat{L}\Delta D\Delta R + \Delta L\hat{D}\Delta R + \Delta L\Delta D\hat{L}^T + \Delta L\Delta D\Delta R| \\
&\leq |\Delta A_1| + |\Delta L||\hat{D}||\hat{L}^T| + |\hat{L}||\Delta D||\hat{L}^T| + |\hat{L}||\hat{D}||\Delta R| \\
&\quad + |\hat{L}||\Delta D||\Delta R| + |\Delta L||\hat{D}||\Delta R| + |\Delta L||\Delta D||\hat{L}^T| + |\Delta L||\Delta D||\Delta R| \\
&\leq (\epsilon_n + \epsilon_{n-1} + \epsilon_1 + \epsilon_{n-1} + 2\epsilon_1\epsilon_{n-1} + \epsilon_{n-1}\epsilon_{n-1} + \epsilon_1\epsilon_{n-1}\epsilon_{n-1})|\hat{L}||\hat{D}||\hat{L}^T| \\
&\leq (\epsilon_n + \epsilon_{2n-1})|\hat{L}||\hat{D}||\hat{L}^T| \leq \epsilon_{3n-1}|\hat{L}||\hat{D}||\hat{L}^T|.
\end{aligned}$$

The second to last inequality is derived by invoking Lemma 3.6. □

To show the improvement of stability because of triadic structure, we bound $\|C\|_S$, where C is defined by

$$(A + \Delta A)\hat{x} = b,\ |\Delta A| = C \circ |\hat{L}||\hat{D}||\hat{L}^T|. \tag{3.15}$$

We follow the notation in the proof of Theorem 3.3. Let

$$|\Delta A_1| = C_1 \circ (|\hat{L}||\hat{D}||\hat{L}^T|).$$

By (3.2) and (3.4),

$$\|C_1\|_S \leq \frac{1}{6}(2n^3 + 3n^2 + n)u + O(u^2).$$

In the unit lower triangular system $\hat{L}y = b$, $\hat{L}(1\!:\!k, 1\!:\!k)y(1\!:\!k) = b(1\!:\!k)$ is a $k \times k$ unit lower triangular system for $k = 1, \ldots, n$. Repeatedly applying Lemma 3.5, we obtain

$$(\hat{L} + \Delta L)\hat{y} = b,\ |\Delta L| \leq \mathrm{diag}(\epsilon_0, \epsilon_1, \ldots, \epsilon_{n-1})|\hat{L}|.$$

Therefore,

$$\begin{aligned}
&|\Delta A| = |\Delta A_1 + \Delta L\hat{D}\hat{L}^T + \hat{L}\Delta D\hat{L}^T + \hat{L}\hat{D}\Delta R| + O(u^2) \\
\leq\ & |\Delta A_1| + |\Delta L||\hat{D}||\hat{L}^T| + |\hat{L}||\Delta D||\hat{L}^T| + |\hat{L}||\hat{D}||\Delta R| + O(u^2) \\
\leq\ & (C_1 + \mathrm{diag}(\epsilon_0, \epsilon_1, \ldots, \epsilon_{n-1})ee^T + (\epsilon_1 + \epsilon_{n-1})ee^T) \circ (|\hat{L}||\hat{D}||\hat{L}^T|) + O(u^2).
\end{aligned}$$

Finally,

$$\begin{aligned}
\|C\|_S &\leq \|C_1\|_S + \|\mathrm{diag}(\epsilon_0, \epsilon_1, \ldots, \epsilon_{n-1})ee^T\|_S + \epsilon_n\|ee^T\|_S + O(u^2) \\
&= \frac{1}{6}(11n^3 + n)u + O(u^2).
\end{aligned} \tag{3.16}$$

Now suppose that $A \in R^{n\times n}$ is symmetric triadic. Following the notation in the proof of Theorem 3.3, the bound (3.14) can be reduced to be $|\Delta R| \leq \epsilon_2|\hat{L}^T|$, and therefore the bound in Theorem 3.3 becomes

$$(A + \Delta A)\hat{x} = b,\ |\Delta A| \leq \epsilon_{2n+2}|\hat{L}||\hat{D}||\hat{L}^T|.$$

The bound on $\|C\|_S$ in (3.15) can be tightened with the triadic structure as follows.

$$\begin{aligned} |\Delta A| &\leq |\Delta A_1| + |\Delta L||\hat{D}||\hat{L}^T| + |\hat{L}||\Delta D||\hat{L}^T| + |\hat{L}||\hat{D}||\Delta R| + O(u^2) \\ &\leq |\Delta A_1| + \epsilon_{n-1}|\hat{L}||\hat{D}||\hat{L}^T| + \epsilon_1|\hat{L}||\hat{D}||\hat{L}^T| + \epsilon_2|\hat{L}||\hat{D}||\hat{L}^T| + O(u^2) \\ &\leq C_1 \circ |\hat{L}||\hat{D}||\hat{L}^T| + \epsilon_{n+2}|\hat{L}||\hat{D}||\hat{L}^T| + O(u^2). \end{aligned}$$

By (3.5), $\|C_1\|_S \leq 9nu + O(u^2)$. By Lemma 3.3, there are at most $7n-14$ nonzero elements in $\hat{L}\hat{D}\hat{L}^T$ for $n \geq 4$. Therefore,

$$\|C\|_S \leq \|C_1\|_S + (7n-14)\epsilon_{n+2} + O(u^2) \leq (7n^2+9n)u + O(u^2). \tag{3.17}$$

Comparing (3.17) with (3.16), we see the improvement of componentwise backward error because of the triadic structure. The analysis is independent of the order of evaluation in $(*)$ and $(**)$ in Algorithm 3.1 and the order of substitution to solve the unit triangular systems.

3.2.2 LBL^T Factorization

Now we extend Theorem 3.3 for LDL^T factorization to Theorem 3.4 for LBL^T factorization.

Theorem 3.4 *Suppose the LBL^T factorization of a symmetric matrix $A \in R^{n\times n}$ runs to completion and produces a computed solution $\hat{x}$ to $Ax = b$. Then*

$$(A+\Delta A)\hat{x} = b,\ |\Delta A| \leq (\max\{\epsilon_c, \epsilon_{4n-3}\} + \epsilon_{2n+c-2})(|A| + |\hat{L}||\hat{B}||\hat{L}^T|),$$

where we assume condition (3.6) holds for all linear systems involving 2×2 pivots.

Proof The proof is analogous to that of Theorem 3.3 but with two differences. First, since condition (3.6) holds, (3.13) is replaced by

$$(\hat{B}+\Delta B)\hat{z} = \hat{y},\ |\Delta B| \leq \epsilon_c|\hat{B}|.$$

Second, we invoke Theorem 3.2 instead of Theorem 3.1 and obtain

$$|\Delta A_1| \leq \max\{\epsilon_c, \epsilon_{4n-3}\}(|A| + |\hat{L}||\hat{B}||\hat{L}^T|).$$

The result is not difficult to see after a little thought. □

Theorem 3.2 and Theorem 3.4 coincide with a theorem by Higham [39, Theorem 4.1][41, Theorem 11.3] described as follows.

Theorem 3.5 (Higham) *Suppose the LBL^T factorization of a symmetric matrix $A \in R^{n\times n}$ runs to completion and produces a computed solution to $Ax = b$. Without loss of generality, we assume all the interchanges are done with the Bunch-Kaufman pivoting strategy (i.e., $A := PAP^T$, where P is the permutation matrix for pivoting). Let $\hat{L}\hat{B}\hat{L}^T$ be the computed factorization and $\hat{x}$ be the computed solution. Assuming condition (3.6) holds for all linear systems involving 2×2 pivots, then*

$$(A + \Delta A_1) = \hat{L}\hat{B}\hat{L}^T \text{ and } (A + \Delta A_2)\hat{x} = b,$$

where

$$|\Delta A_i| \leq p(n)u(|A| + |\hat{L}||\hat{B}||\hat{L}^T|) + O(u^2),\ i = 1, 2,$$

with $p(n)$ a linear polynomial.

Three remarks are in order. First, Higham's proof is via the LBL^T factorization in outer product form [39], whereas our proof uses inner product form. Second, we give a precise bounding coefficient. Third, Lemma 3.6 eliminates the $O(u^2)$ terms. The result is also true for the Bunch-Parlett, fast Bunch-Parlett, and bounded Bunch-Kaufman pivoting strategies, because they satisfy stronger conditions than the Bunch-Kaufman.

We may also bound $\|C\|_S$, where

$$(A + \Delta A)\hat{x} = b,\ |\Delta A| = C \circ (|A| + |\hat{L}||\hat{B}||\hat{L}^T|). \tag{3.18}$$

Let

$$|A - \hat{L}\hat{B}\hat{L}^T| = C_1 \circ (|A| + |\hat{L}||\hat{B}||\hat{L}^T|).$$

By (3.11),

$$\|C_1\|_S \leq \frac{1}{3}(4n^3 + 3(c-1)n^2 + 2n)u + O(u^2).$$

Similar to (3.16), we find that

$$\begin{aligned} \|C\|_S &\leq \|C_1\|_S + \|\mathrm{diag}(\epsilon_0, \epsilon_1, \ldots, \epsilon_{n-1})ee^T\|_S + (\epsilon_c + \epsilon_{n-1})\|ee^T\|_S + O(u^2) \\ &= (\frac{17}{6}n^3 + \frac{4c-5}{2}n^2 + 2n)u + O(u^2). \end{aligned} \tag{3.19}$$

Now suppose that $A \in R^{n\times n}$ is symmetric triadic. By (3.12),

$$\|C_1\|_S = 4(7c + 36)nu + O(u^2).$$

By Lemma 3.3, there are at most $7n - 14$ nonzero blocks in $\hat{L}\hat{D}\hat{L}^T$ for $n \geq 4$. Each block has at most 4 elements. In the similar vein as (3.17),

$$\begin{aligned} \|C\|_S &\leq \|C_1\|_S + 4(7n - 14)\epsilon_{n+c+1} + O(u^2) \\ &\leq (28n^2 + 56cn + 144n)u + O(u^2). \end{aligned} \tag{3.20}$$

Comparing (3.20) with (3.19), we see the improvement of componentwise backward error because of the triadic structure. Note that the analysis is independent of the order of evaluation in $(*)$ and $(**)$ in Algorithm 3.2 and the order of substitution for solving each unit triangular system.

By Theorems 3.1, 3.2, 3.3, and 3.4, the stability of LDL^T and LBL^T factorizations relies on the bounds on $|\hat{L}||\hat{D}||\hat{L}^T|$ and $|\hat{L}||\hat{B}||\hat{L}^T|$, respectively.

3.3 Normwise Analysis

In this section, we focus on bounding $|||L||D||L^T|||$ and $|||L||B||L^T|||$ in terms of $\|A\|$ to analyze the stability of LDL^T factorization and LBL^T factorization, respectively. We also give a sufficient condition for the success of LBL^T factorization with inertia preserved.

3.3.1 LDL^T Factorization

Theorem 3.1 and Theorem 3.3 imply that the LDL^T factorization of a symmetric matrix $A \in R^{n\times n}$ and its application to solve $Ax = b$ are backward stable, if $|||\hat{L}||\hat{D}||\hat{L}^T|||$ is suitably bounded relative to $\|A\|$. For simplicity, we begin by bounding $|||L||D||L^T|||$ instead of $|||\hat{L}||\hat{D}||\hat{L}^T|||$.

If $A \in R^{n\times n}$ is symmetric positive definite, its LL^T factorization shares the properties with LDL^T factorization, including

$$|||L||D||L^T|||_2 = |||LD^{\frac{1}{2}}||D^{\frac{1}{2}}L^T|||_2 = |||LD^{\frac{1}{2}}|||_2^2 \leq n\|LD^{\frac{1}{2}}\|_2^2 = n\|A\|_2. \quad (3.21)$$

If $A \in R^{n\times n}$ is symmetric and diagonally dominant, its LDL^T factorization inherits the properties of LU factorization of a diagonally dominant matrix. Diagonal dominance guarantees that $|L|$ is diagonally dominant by columns, which implies $|||L^{-T}||L^T|||_\infty \leq 2n-1$ [41, Lemma 8.8]. Therefore,

$$\begin{aligned} |||L||D||L^T|||_\infty &= |||LD||L^T|||_\infty = |||AL^{-T}||L^T|||_\infty \\ &\leq |||A|||_\infty|||L^{-T}||L^T|||_\infty \leq (2n-1)\|A\|_\infty. \end{aligned}$$

The derivation is adapted from that for LU factorization of a diagonally dominant matrix.

We conclude that if A is positive definite or diagonally dominant, its LDL^T factorization and the factorization's use in solving linear system are backward

stable even without pivoting. A weaker condition for the stability of LDL^T factorization can be obtained by [41, Theorem 9.5] described as follows.

Theorem 3.6 *The LU factorization of $A \in R^{n\times n}$ satisfies*

$$\||L||U|\|_\infty \leq (1 + 2(n^2-n)\rho_n)\|A\|_\infty,$$

where ρ_n is the growth factor (the largest element in magnitude in all Schur complements divided by the largest element in $|A|$). This statement is independent of the pivoting strategy applied, if any.

Proof The proof is similar to that of a theorem by Wilkinson. See the discussion of [41, Theorem 9.5] for details. □

When A is symmetric, DL^T in its LDL^T factorization plays the role of U in the LU factorization. Therefore, Theorem 3.6 is applicable to $\||L||D||L^T|\|_\infty$, because $|L||D||L^T| = |L||DL^T| = |LU|$. As a result, the stability of LDL^T factorization is assured as long as the growth factor ρ_n is modest. By Theorems 3.1 and 3.6,

$$\||\hat{L}||\hat{D}||\hat{L}^T|\|_\infty \leq \frac{1 + 2(n^2-n)\rho_n}{1 - (1 + 2(n^2-n)\rho_n)\epsilon_n}\|A\|_\infty.$$

From this point of view, both positive definiteness and diagonal dominance guarantee the stability, because their growth factors are bounded by 1 and 2, respectively. Unfortunately, $|L||D||L^T|$ could be catastrophically large for general matrices. For example, if $A = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix}$, then the corresponding $|L||D||L^T| = \begin{bmatrix} \epsilon & 1 \\ 1 & 2/\epsilon \end{bmatrix}$ is unbounded as $\epsilon \to 0$. This is an illustration of the well-known fact that LDL^T factorization is not generally stable.

In particular, quasidefiniteness guarantees the existence of the LDL^T factorization. Gill, Saunders and Shinnerl [29] analyzed the stability of the LDL^T factorization of a symmetric quasidefinite matrix via the LU factorization of an unsymmetric positive-definite matrix. Their result is as follows.

Theorem 3.7 (Gill, Saunders and Shinnerl) *Given a symmetric quasidefinite matrix* $A = \begin{bmatrix} H & K^T \\ K & -G \end{bmatrix}$ *with* H *and* G *positive definite, the factorization* $PAP^T = LDL^T$ *is stable for any permutation* P*, if* $\theta(A)$ *is not too large, where*

$$\theta(A) = (\frac{\|K\|_2}{\max\{\|G\|_2, \|H\|_2\}})^2 \max\{\kappa_2(G), \kappa_2(H)\}.$$

3.3.2 LBL^T Factorization

Theorems 3.2 and 3.4 imply that the LBL^T factorization of a symmetric matrix $A \in R^{n\times n}$ and its application to solving $Ax = b$ are backward stable, if all linear 2×2 systems are solved with condition (3.6) satisfied and $\||\hat{L}||\hat{B}||\hat{L}^T|\|$ is suitably bounded relative to $\|A\|$. For simplicity, we begin with bounding $\||L||B||L^T|\|$ instead of $\||\hat{L}||\hat{B}||\hat{L}^T|\|$. In other words, our objective is to find a modest c_n such that

$$\||L||B||L^T|\| \leq c_n \|LBL^T\| \tag{3.22}$$

in some proper norm. Using the ∞-norm, we obtain

$$\||L||B||L^T|\|_\infty \leq \||L|\|_\infty \||B|\|_\infty \||L^T|\|_\infty = \|L\|_\infty \|B\|_\infty \|L^T\|_\infty.$$

Therefore, if $\|L\|_\infty \|B\|_\infty \|L^T\|_\infty$ is modest relative to $\|A\|_\infty$ and condition (3.6) holds, then the corresponding LBL^T factorization is normwise backward stable. The same statement can also be obtained from a suitable modification of error

analysis for block LU factorization in [19]. For the Bunch-Parlett, fast Bunch-Parlett, and bounded Bunch-Kaufman pivoting strategies, the element growth of B is well-controlled and the elements in L are bounded, so they are norm-wise backward stable methods. Note that the element growth of B is up to the growth factor ρ_n, whose bound is displayed in Table 3.1. The bounds for general symmetric matrices are from (2.6) and (2.7). The bounds for symmetric triadic matrices are from Theorem 2.6.

Table 3.1: Bounds on growth factor ρ_n.

Symmetric Matrix	α	**BK**	**BBK** and **FBP**	**BP**
general	$\frac{1+\sqrt{17}}{8}$	2.57^{n-1}		$5.4n^{1+(\ln n)/4}$
triadic	$\frac{\sqrt{5}-1}{2}$	$O(1.28^n)$	$O(n^{1.7})$	

The Bunch-Kaufman pivoting strategy [12], Bunch's pivoting strategy [10], and the Bunch-Marcia pivoting strategy [13] may result in unbounded L, so we cannot prove the stability by bounding $\|L\|_\infty\|B\|_\infty\|L^T\|_\infty$. All these strategies do have growth factors well-controlled. However, unlike LU factorization and LDL^T factorization, $\||L||B||L^T|\|$ cannot be bounded in terms of the growth factor and $\|A\|$. For example, without pivoting, we have

$$A = \begin{bmatrix} 1 & 1 & \\ 1 & 1+\epsilon^2 & -\epsilon \\ & -\epsilon & 2 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ \frac{1}{\epsilon} & -\frac{1}{\epsilon} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & \\ 1 & 1+\epsilon^2 & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & & \frac{1}{\epsilon} \\ & 1 & -\frac{1}{\epsilon} \\ & & 1 \end{bmatrix} = LBL^T.$$

The factorization has modest $\|A\|$ for small $\epsilon \neq 0$ and no element growth, but $\||L||B||L^T|\|$ is unbounded when $\epsilon \to 0$.

Higham [39] showed that using Bunch-Kaufman pivoting strategy with the pivoting argument $\alpha = \frac{1+\sqrt{17}}{8} \approx 0.640$, the LBL^T factorization of a symmetric

matrix $A \in R^{n\times n}$ satisfies

$$\begin{aligned} \||L||B||L^T|\|_M &\leq \max\{\frac{1}{\alpha}, \frac{(3+\alpha^2)(3+\alpha)}{(1-\alpha^2)^2}\} n\rho_n \|A\|_M \\ &\approx 35.674 n\rho_n \|A\|_M < 36 n\rho_n \|A\|_M, \end{aligned} \tag{3.23}$$

where ρ_n is the growth factor and $\|\cdot\|_M$ is the largest magnitude element in the given matrix. These bounds also hold for the other suggested pivoting arguments $\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$ for triadic matrices (see Theorem 2.6) and $\alpha = 0.5$ to minimize the element bound on L (see Table 2.1), as well as a variant by Sorensen and Van Loan [21, Section 5.3.2] (see (2.9) for the change they made). The Bunch-Parlett, fast Bunch-Parlett and bounded Bunch-Kaufman pivoting strategies satisfy a stronger condition than the Bunch-Kaufman, so they also satisfy (3.23). By Theorem 3.2 and (3.23),

$$\||\hat{L}||\hat{B}||\hat{L}^T|\|_M \leq 36 n\rho_n \frac{1+\max\{\epsilon_c, \epsilon_{4n-3}\}}{1-36n\rho_n \max\{\epsilon_c, \epsilon_{4n-3}\}} \|A\|_M. \tag{3.24}$$

Higham [40] proved that with Bunch's pivoting strategy,

$$\||L||B||L^T|\|_M \leq 42\|A\|_M, \tag{3.25}$$

where A is symmetric tridiagonal. So a bound on $\||\hat{L}||\hat{B}||\hat{L}^T|\|_M$ in terms of $\|A\|_M$ can be obtained similarly. Bunch and Marcia [13] also showed that the normwise backward error bound (3.25) is preserved with their pivoting method.

In summary, all pivoting strategies for LBL^T factorization in the literature [5, 10, 12, 13, 14] are stable methods. Wilkinson [66] showed that the Cholesky factorization of a symmetric positive definite matrix $A \in R^{n\times n}$ is guaranteed to run to completion if $20n^{3/2}\kappa_2(A)u \leq 1$. We give a sufficient condition for the success of LBL^T factorization with inertia preserved in Theorem 3.9, with proof invoking a theorem by Weyl [42, page 181].

Theorem 3.8 (Weyl) *Let A, B be two $n \times n$ Hermitian matrices and $\lambda_k(A)$, $\lambda_k(B)$, $\lambda_k(A+B)$ be the eigenvalues of A, B, and $A+B$ arranged in increasing order for $k = 1, \ldots, n$. Then for $k = 1, \ldots, n$, we have*

$$\lambda_k(A) + \lambda_1(B) \leq \lambda_k(A+B) \leq \lambda_k(A) + \lambda_n(B).$$

Theorem 3.9 *With the Bunch-Parlett, Bunch-Kaufman, fast Bunch-Parlett or bounded Bunch-Kaufman pivoting strategy, the LBL^T factorization of a symmetric matrix $A \in R^{n \times n}$ succeeds with inertia preserved if $f(n)\kappa_2(A) < 1$ (i.e., A is not too ill conditioned), where*

$$f(n) = 36n^2\rho_n \max\{\epsilon_c, \epsilon_{4n-3}\} \frac{1 + \max\{\epsilon_c, \epsilon_{4n-3}\}}{1 - 36n\rho_n \max\{\epsilon_c, \epsilon_{4n-3}\}} = 144n^3\rho_n u + O(u^2).$$

Proof The proof is by finite induction. Consider Algorithm 3.2. Let

$$A + \Delta A := \hat{L}\hat{B}\hat{L}^T, \; \hat{A}_k := A_k + \Delta A_k,$$

where $A_k := A(1:k, 1:k)$ and $\Delta A_k := \Delta A(1:k, 1:k)$. The process of LBL^T factorization is to iteratively factor A_k, increasing k from 1 to n. Obviously, the first stage succeeds for a nonsingular matrix. Suppose the factorization of A_{k-p} is successfully completed with inertia preserved (i.e., the inertia of $\hat{A}_{k-p}$ is the same as that of A_{k-p}), where $p = 1$ or 2 denotes whether the next pivot is 1×1 or 2×2. Since the inertia of $\hat{A}_{k-p}$ is preserved, all the pivots in $\hat{A}_{k-p}$ are full rank, so the factorization of A_k succeeds (i.e., with no division by zero). The rest of the proof is to show that the inertia is preserved in $\hat{A}_k$ so the induction can be continued.

By Theorem 3.2, (3.23) and (3.24), the componentwise backward error satisfies

$$\|\Delta A_k\|_2 \leq 36k^2\rho_k \max\{\epsilon_c, \epsilon_{4k-3}\} \frac{1 + \max\{\epsilon_c, \epsilon_{4k-3}\}}{1 - 36k\rho_k \max\{\epsilon_c, \epsilon_{4k-3}\}} \|A_k\|_2 =: f(k)\|A_k\|_2$$

for all possible $1 \leq k \leq n$. Note that $f(k) = 144k^3\rho_k u + O(u^2)$. Let

$$\lambda_*(A_k) := \min_{1 \leq i \leq k} |\lambda_i(A_k)|.$$

Assume $f(n)\kappa_2(A_n) < 1$. By Theorem 3.8, if $\lambda_i(A_k) > 0$,

$$\begin{aligned} \lambda_i(A_k + \Delta A_k) &\geq \lambda_i(A_k) - \|\Delta A_k\|_2 \geq \lambda_*(A_k) - f(k)\|A_k\|_2 \\ &= \lambda_*(A_k)(1 - f(k)\kappa_2(A_k)) \geq \lambda_*(A_n)(1 - f(n)\kappa_2(A_n)) > 0. \end{aligned}$$

Similarly, if $\lambda_i(A_k) < 0$,

$$\lambda_i(A_k + \Delta A_k) \leq \lambda_i(A_k) + \|\Delta A_k\|_2 \leq -\lambda_*(A_k) + f(k)\|A_k\|_2 < 0.$$

Therefore, $\lambda_i(A_k + \Delta A_k)$ and $\lambda_i(A_k)$ have the same sign for $i = 1, \ldots, k$. So the pivoting guarantees that the inertia of $\hat{A}_k$ is preserved. By induction, the factorization is guaranteed running to completion with inertia preserved. □

3.4 Rank Estimation

Cholesky factorization for symmetric positive semidefinite matrices with complete pivoting can be used for rank estimation. The stability is well studied in [38]. In this section, we discuss the stability of the Bunch-Parlett, fast Bunch-Parlett, or bounded Bunch-Kaufman pivoting strategy applied to the rank estimation of symmetric indefinite matrices.

Recall that we use A to denote a symmetric matrix of floating point numbers. It is unrealistic to assume A is singular because of the potential rounding errors. Let $\widetilde{A}$ be the exact singular matrix of rank $r < n$ under consideration and $A = \widetilde{A} + \Delta\widetilde{A}$ is its stored matrix of floating point numbers. The overall backward error is $\widetilde{A} - \hat{L}\hat{B}\hat{L}^T$, where $\hat{L} \in R^{n \times r}$ and $\hat{B} \in R^{r \times r}$. For simplicity of discussion we consider only $A - \hat{L}\hat{B}\hat{L}^T$ and say that A is rank r, assuming A is close to $\widetilde{A}$.

3.4.1 LDL^T Factorization

Given a symmetric matrix $A \in R^{n\times n}$ of rank $r < n$, we assume the necessary interchanges are done so that $A(1\!:\!r, 1\!:\!r)$ is nonsingular. The LDL^T factorization, assuming it completes the first r steps, has $L = \begin{bmatrix} L_{11} \\ L_{21} \end{bmatrix}$, where $L_{11} \in R^{r\times r}$ is unit lower triangular, $L_{21} \in R^{(n-r)\times r}$, and $D \in R^{r\times r}$ is diagonal. The factorization is computationally equivalent to Algorithm 3.3.

Algorithm 3.3 LDL^T factorization of $A \in R^{n\times n}$ of rank $r < n$.

for $j = 1, \ldots, r$ **do**

  $(*)$ $d_j = a_{jj} - \sum_{k=1}^{j-1} d_k l_{jk}^2$

  for $i = j+1, \ldots, n$ **do**

    $(**)$ $l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} d_k l_{ik} l_{jk})/d_j$

  end for

end for

Algorithm 3.3 can be regarded as the incomplete LDL^T factorization after processing r rows/columns.

Theorem 3.10 *We consider the incomplete LDL^T factorization of a symmetric matrix $A \in R^{n\times n}$ after processing r rows/columns ($r < n$), and define the backward error ΔA by*

$$A + \Delta A = \hat{L}\hat{D}\hat{L}^T + \hat{A}^{(r+1)},$$

where

$$\hat{A}^{(r+1)} = \begin{array}{c} \\ r \\ n-r \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & \hat{S}_r(A) \end{bmatrix} \end{array},$$

with $\hat{S}_r(A)$ the computed Schur complement. Then

$$|\Delta A| \leq \epsilon_{r+1}(|\hat{L}||\hat{D}||\hat{L}^T| + |\hat{A}^{(r+1)}|). \tag{3.26}$$

Proof Denote the (i,j) entry of $\hat{A}^{(r+1)}$ by $\hat{a}_{ij}^{(r+1)}$. To simplify the notation, we define $\hat{l}_{jj} := 1$ for $j = 1, \ldots, r$. By Lemma 3.2,

$$|a_{ij} - \sum_{k=1}^{j} \hat{d}_k \hat{l}_{jk} \hat{l}_{ik}| \leq \epsilon_j \sum_{k=1}^{j} |\hat{d}_k \hat{l}_{ik} \hat{l}_{jk}| \tag{3.27}$$

for $j = 1, \ldots, r$ and $i = j, \ldots, n$.

$$|a_{ij} - \hat{a}_{ij}^{(r+1)} - \sum_{k=1}^{r} \hat{d}_k \hat{l}_{ik} \hat{l}_{jk}| \leq \epsilon_{r+1}(|\hat{a}_{ij}^{(r+1)}| + \sum_{k=1}^{r} |\hat{d}_k \hat{l}_{ik} \hat{l}_{jk}|) \tag{3.28}$$

for $j = r+1, \ldots, n$ and $i = j, \ldots, n$. The result is obtained by collecting terms in (3.27)–(3.28) into one matrix representation. □

Recall that if A is triadic then the triadic structure in preserved in L, but the last row in L can be full[8]. Therefore, the bounding coefficient ϵ_{r+1} in Theorem 3.10 cannot be reduced with the triadic structure. Instead, we bound $\|C\|_S$, where

$$|\Delta A| = C \circ (|\hat{L}||\hat{D}||\hat{L}^T| + |\hat{A}^{(r+1)}|). \tag{3.29}$$

By (3.27)–(3.28), the elements in (i,j) block entry of C satisfies $c_{ij} = c_{ji} = \epsilon_{\min\{j,r+1\}}$ for $1 \leq j \leq i \leq n$. Applying (3.4) to bound $\|C(1\!:\!r, 1\!:\!r)\|_S$,

$$\begin{aligned}
\|C\|_S &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \\
&= \sum_{i=1}^{r}\sum_{i=1}^{r} c_{ij} + 2\sum_{i=r+1}^{n}\sum_{j=1}^{r} c_{ij} + \sum_{i=r+1}^{n}\sum_{j=r+1}^{n} c_{ij} \\
&\leq \frac{1}{6}(2r^3 + 3r^2 + r)u + 2(n-r)\sum_{j=1}^{r} \epsilon_j + (n-r)^2\epsilon_{r+1} + O(u^2) \\
&\leq (n(n-r)(r+1) + \frac{1}{3}r^3 + \frac{1}{2}r^2 + \frac{1}{6}r)u + O(u^2).
\end{aligned} \tag{3.30}$$

[8]See the example with the circular shift down matrix (2.1) in Section 2.1.

Lemma 3.7 is developed to tighten the bound on $\|C\|_S$ with triadic structure and to show the improvement of stability.

Lemma 3.7 *For any triadic matrix $T \in R^{n\times r}$, TT^T has at most $6r$ nonzero off-diagonal elements/terms and $\min\{n, 3r\}$ diagonal elements/terms. The bounds, $6r$ and $\max\{n, 3r\}$, are attained with $L = (Z^3 + Z + I)(1\!:\!n, 1\!:\!r)$ for $n - r \geq 3$, where $Z \in R^{n\times n}$ is the shift-down matrix.*

Proof The proof is analogous to that of Lemma 3.3 and omitted here. □

Suppose $A \in R^{n\times n}$ is symmetric triadic of rank $r < n$ and has an LDL^T factorization. By Lemma 3.7, there are at most $9r$ terms in $\hat{L}\hat{D}\hat{L}^T$. Therefore,

$$\|C\|_S = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} \leq \epsilon_{2\cdot 9r} = 18ru + O(u^2). \tag{3.31}$$

Comparing (3.31) with (3.30), we see the improvement of componentwise backward error because of the triadic structure. Note that the analysis is independent of the order of evaluation in $(*)$ and $(**)$ in Algorithm 3.3.

3.4.2 LBL^T Factorization

Now we investigate the LBL^T factorization of a symmetric matrix $A \in R^{n\times n}$ of rank $r < n$. Assume that the necessary interchanges are done so that $A(1\!:\!r, 1\!:\!r)$ has rank r, as they would be with the Bunch-Parlett, fast Bunch-Parlett, or bounded Bunch-Kaufman pivoting. The factorization is denoted by $A = LBL^T$, where

$$B = \begin{bmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_{m-1} \end{bmatrix} \in R^{r\times r}$$

and

$$L = \begin{bmatrix} L_{11} & & & \\ L_{21} & L_{22} & & \\ \vdots & & \ddots & \\ L_{m-1,1} & L_{m-1,2} & \cdots & L_{m-1,m-1} \\ L_{m1} & L_{m2} & \cdots & L_{m,m-1} \end{bmatrix} \in R^{n\times r}.$$

Each B_i is either 1×1 or 2×2, with $L_{ii} = 1$ or $L_{ii} = I_2$, respectively. This factorization is guaranteed if $A(1\!:\!r, 1\!:\!r)$ satisfies the condition in Theorem 3.9. The factorization is computationally equivalent to Algorithm 3.4.

Algorithm 3.4 LBL^T factorization of $A \in R^{n\times n}$ of rank $r < n$.

for $j = 1, \ldots, m-1$ **do**

$(*)$ $B_j = A_{jj} - \sum_{k=1}^{j-1} L_{jk} B_k L_{jk}^T$

for $i = j+1, \ldots, m$ **do**

$(**)$ $L_{ij} = (A_{ij} - \sum_{k=1}^{j-1} L_{ik} B_k L_{jk}^T) B_j^{-1}$

end for

end for

Algorithm 3.4 can be regarded as the incomplete LBL^T factorization after processing r rows/columns. Theorem 3.11 gives a bound on its componentwise backward error. In rank estimation we are concerned with a symmetric matrix $A \in R^{n\times n}$ of rank $r < n$.

Theorem 3.11 *We consider the incomplete LBL^T factorization of a symmetric matrix $A \in R^{n\times n}$ after processing r rows/columns ($r < n$), assume condition (3.6) holds for all linear systems involving 2×2 pivots, and define the backward error ΔA by*

$$A + \Delta A = \hat{L}\hat{B}\hat{L}^T + \hat{A}^{(r+1)}, \tag{3.32}$$

where

$$\hat{A}^{(r+1)} = \begin{array}{c} \\ r \\ n-r \end{array} \begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & \hat{S}_r(A) \end{bmatrix} \end{array},$$

with $\hat{S}_r(A)$ the computed Schur complement. Then

$$|\Delta A| \leq \max\{\epsilon_c, \epsilon_{4r+1}\}(|A| + |\hat{L}||\hat{B}||\hat{L}^T| + |\hat{A}^{(r+1)}|). \tag{3.33}$$

Proof To simplify the notation, we let $\hat{L}_{ii} := 1$ or I_2 depending on whether B_i is 1×1 or 2×2 for $i = 1, \ldots, m-1$. By Lemma 3.4 with the assumption that condition (3.6) holds,

$$|A_{ij} - \sum_{k=1}^{j} \hat{L}_{ik}\hat{B}_k\hat{L}_{jk}^T| \leq \max\{\epsilon_c, \epsilon_{4j-3}\}(|A_{ij}| + \sum_{k=1}^{j} |\hat{L}_{ik}||\hat{B}_k||\hat{L}_{jk}^T|), \tag{3.34}$$

for $j = 1, \ldots, m-1$ and $i = j, \ldots, m$. Note that though A_{mm} can be larger than 2×2, the error analysis in Lemma 3.4 for $\hat{S}_r(A)$ is still valid. Therefore,

$$|A_{mm} - \hat{S}_r(A) + \sum_{k=1}^{m-1} \hat{L}_{mk}\hat{B}_k\hat{L}_{mk}^T| \leq \epsilon_{4m-3}(|A_{mm}| + |\hat{S}_r(A)| + \sum_{k=1}^{m-1} |\hat{L}_{mk}||\hat{B}_k||\hat{L}_{mk}^T|). \tag{3.35}$$

The result is obtained by collecting terms in (3.34)–(3.35) into one matrix representation. □

To show the improvement of stability because of triadic structure, we define C by

$$|\Delta A| = C \circ (|A| + |\hat{L}||\hat{B}||\hat{L}^T| + |\hat{A}^{(r+1)}|). \tag{3.36}$$

By (3.34)–(3.35), the elements in (i, j) block entry of C are bounded by $\max\{\epsilon_c, \epsilon_{4j-3}\}$ for $1 \leq j \leq i \leq m$. Applying (3.11) to bound $\|C(1\!:\!r, 1\!:\!r)\|_S$,

$$\begin{aligned} \|C\|_S &= \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} = \sum_{i=1}^{r}\sum_{j=1}^{r} c_{ij} + 2\sum_{i=r+1}^{n}\sum_{j=1}^{r} c_{ij} + \sum_{i=r+1}^{n}\sum_{j=r+1}^{n} c_{ij} \\ &\leq cn^2u + \frac{1}{3}(4r^3 - 3r^2 + 2r)u + 2(n-r)\sum_{j=1}^{r} \epsilon_{4j-3} + (n-r)^2\epsilon_{4r+1} + O(u^2) \\ &= (cn^2 + (4nr + n - 3r)(n-r) + \frac{4}{3}r^3 - r^2 + \frac{2}{3}r)u + O(u^2). \end{aligned} \tag{3.37}$$

By Lemma 3.7, if A is triadic of rank r, there are at most $9r$ block terms in the LBL^T factorization. Each has at most 4 elements. Therefore,

$$\|C\|_S \leq 4(9r\epsilon_c + \epsilon_{9(4r+1)}) \leq 36(cr + 4r + 1)u + O(u^2). \tag{3.38}$$

Comparing (3.38) with (3.37), we see the improvement of componentwise backward error because of the triadic structure. Note that the analysis is independent of the order of evaluation in $(*)$ and $(**)$ in Algorithm 3.4.

3.4.3 Normwise Analysis

In this subsection we bound $\|A - \hat{L}\hat{B}\hat{L}^T\|_F$ for analyzing the stability of the Bunch-Parlett, fast Bunch-Parlett, and bounded Bunch-Kaufman pivoting strategies applied to rank estimation for symmetric indefinite matrices. We also bound $\|A - \hat{L}\hat{D}\hat{L}^T\|_2$ and $\|A - \hat{L}\hat{D}\hat{L}^T\|_\infty$ for positive semidefinite matrices and diagonal dominance matrices, respectively.

Theorem 3.12 *With the Bunch-Parlett, fast Bunch-Parlett, or bounded Bunch-Kaufman pivoting on a symmetric indefinite matrix A of rank r,*

$$\|A - \hat{L}\hat{B}\hat{L}^T\|_F \leq \max\{c, 4r+1\}(\tau(A) + 1)((\|W\|_F+1)^2 + 1)u\|A\|_F + O(u^2),$$

where $W = A(1:r, 1:r)^{-1}A(1:r, r+1:n)$, $\tau(A) = \||L||B||L^T|\|_F/\|A\|_F$, and c is from condition (3.6).

Proof Since (3.6) holds, so does (3.33). The growth factor and $\tau(A)$ are well-controlled, so that $\Delta A = O(u)$, where ΔA is defined in (3.32). Here and later in (3.40), we use the property $|\hat{L}||\hat{B}||\hat{L}^T| = |L||B||L^T| + O(u)$ for derivation. Note that $\hat{L}\hat{B}\hat{L}^T$ is the partial LBL^T factorization of $A + \Delta A$ with $\hat{A}^{(r+1)}$ the Schur

complement. By Theorem 2.7, we obtain the perturbation bound

$$\|\hat{A}^{(r+1)}\|_F \leq (\|W\|_F + 1)^2 \|\Delta A\|_F + O(u^2), \tag{3.39}$$

where $W = A(1:r, 1:r)^{-1} A(1:r, r+1:n)$ has a bound given in Lemma 2.3. By (3.32), (3.39) and $\Delta A = O(u)$, $\hat{L}\hat{B}\hat{L}^T = A + O(u)$. Therefore,

$$\begin{aligned} \|\Delta A\|_F &\leq \max\{\epsilon_c, \epsilon_{4r+1}\}(\||\hat{L}||\hat{B}||\hat{L}^T|\|_F + \||A|\|_F + \||\hat{A}^{(r+1)}|\|_F) \\ &= \max\{c, 4r+1\}(\tau(A)+1)u\|A\|_F + O(u^2). \end{aligned} \tag{3.40}$$

Substituting (3.40) into (3.39), we obtain

$$\|\hat{A}^{(r+1)}\|_F \leq \max\{c, 4r+1\}(\tau(A)+1)(\|W\|_F + 1)^2 u\|A\|_F + O(u^2). \tag{3.41}$$

The result is concluded from (3.32), (3.40) and (3.41). □

Now we consider the Bunch-Parlett, fast Bunch-Parlett, or bounded Bunch-Kaufman pivoting strategy incorporated into the LBL^T factorization. By Theorem 3.12, the bound on $\|A - \hat{L}\hat{B}\hat{L}^T\|_F / \|A\|_F$ is governed by $\|W\|_F$ and $\tau(A)$. For any general singular symmetric $A \in R^{n\times n}$,

$$\|W\|_{2,F} \leq \sqrt{\frac{\gamma}{\gamma+2}(n-r)((1+\gamma)^{2r} - 1)}, \tag{3.42}$$

where $\gamma = \max\{\frac{1}{\alpha}, \frac{1}{1-\alpha}\}$ is the element bound of L (see Lemma 2.3 and (2.17)). With the suggested pivoting argument $\alpha = \frac{1+\sqrt{17}}{8} \approx 0.640$, $\gamma = \frac{1+\sqrt{17}}{4} \approx 2.562$. Applying the analysis for (3.23) to bound $\tau(A)$, we obtain

$$\tau(A) \leq 36n(r+1)\rho_{r+1} \tag{3.43}$$

for symmetric $A \in R^{n\times n}$ of rank $r < n$, where ρ_{r+1} is the growth factor.

If all the nonzero eigenvalues are positive, then the matrix is semidefinite, and the LBL^T factorization with the Bunch-Parlett or fast Bunch-Parlett pivoting

strategy is equivalent to the LDL^T factorization with complete pivoting. With an argument similar to that used in obtaining (3.21), the bound on $\tau(A)$ is reduced to $\tau(A) \leq r$, where we use the 2-norm instead of the Frobenius norm. Following the proof of Theorem 3.12 but using (3.26) instead of (3.33), we find that

$$\|A - \hat{L}\hat{D}\hat{L}^T\|_2 \leq r(r+1)((\|W\|_2+1)^2+1)u\|A\|_2 + O(u^2).$$

A comparable bound for Cholesky factorization of a positive semidefinite matrix was given in [38] by Higham. Note that the bound on $\|W\|_2$ is also reduced because $\gamma = 1$.

A similar analysis for diagonal dominant matrices gives

$$\|A - \hat{L}\hat{D}\hat{L}^T\|_\infty \leq (2n-1)(r+1)((\|W\|_\infty+1)^2+1)u\|A\|_\infty + O(u^2).$$

Recall that $W = L_{11}^{-T}L_{21}^T$, where $L_{11} = L(1\!:\!r, 1\!:\!r)$ and $L_{21} = L(r\!+\!1\!:\!n, 1\!:\!r)$. Diagonal dominance guarantees that L_{11}^{-T} has all elements bounded by 1. Therefore, $\|W\|_\infty \leq \||L_{11}^{-T}||L_{21}^T|e\|_\infty \leq \||L_{11}^{-T}|e\|_\infty \leq r$, which implies high stability.

3.4.4 Experiments

For rank estimation, the important practical issue is when to stop the factorization. In (3.41), the bound on $\|\hat{A}^{(r+1)}\|_F/\|A\|_F$ is governed by $\|W\|_F$ and $\tau(A)$. However, both bounds (3.42) and (3.43) are pessimistic. To investigate the typical ranges of $\|W\|_F$ and $\tau(A)$ in practice, we used the random matrices described as follows.

Each indefinite matrix was constructed as $Q\Lambda Q^T \in R^{n\times n}$, where Q is a random orthogonal matrix generated by the method of G. W. Stewart [60] (different for each matrix) and $\Lambda = \mathrm{diag}(\lambda_i)$ of rank r. The following three test sets were

used in our experiments:

$$|\lambda_1| = |\lambda_2| = \cdots = |\lambda_{r-1}| = 1,\ \lambda_r = \sigma,$$

$$|\lambda_1| = |\lambda_2| = \cdots = |\lambda_{r-1}| = \sigma,\ \lambda_r = 1,$$

$$|\lambda_i| = \beta^i,\ i = 1, \ldots, r-1,\ \lambda_r = 1,$$

where $0 < \sigma \leq 1$, and $\beta^{r-1} = \sigma$ for $r > 1$. We assign the sign of λ_i randomly for $i = 1, \ldots, r-1$, and let t denote the number of negative eigenvalues. For each test set, we experimented with all combinations of $n = 10, 20, \ldots, 100$, $r = 2, 3, \ldots, n$, $t = 1, 2, \ldots, r-1$, and $\sigma = 1, 10^{-3}, \ldots, 10^{-12}$, for a total of $94,875$ indefinite matrices for each set.

Table 3.2: Experimental maximum growth factor ρ_{r+1}, $\|W\|_F$, $\tau(A)$, ξ_r, η_r for assessment of stability of rank estimation.

Algorithm	ρ_{r+1}	$\|W\|_F$	$\tau(A)$	ξ_r	η_r
Bunch-Parlett	15.143	39.313	38.263	90.911	90.514
Fast Bunch-Parlett	18.506	37.934	41.919	172.412	172.301
Bounded Bunch-Kaufman	28.691	103.467	53.724	67720	67719.9

In addition to $\|W\|_F$ and $\tau(A)$, we also measured the growth factor ρ_{r+1}, and the relative backward error $\xi_r := \|A - \hat{L}\hat{B}\hat{L}^T\|_F/(u\|A\|_F)$. and the relative Schur residual $\eta_r := \|\|_F/(u\|A\|_F)$. Their maximum values for the Bunch-Parlett, fast Bunch-Parlett, and bounded Bunch-Kaufman pivoting strategies are displayed in Table 3.2. All these numbers are modest, except that the Bunch-Kaufman pivoting strategy introduced relatively large backward errors. The assessment showed that the rank estimation by LBL^T factorization with Bunch-Parlett and fast Bunch-Parlett pivoting strategies is potentially stable in practice.

Note that the bounds (3.42) and (3.43) depend on r more than n, as does (3.41). Experimenting with several different stopping criteria, we suggest

$$\|\hat{A}_{k+1}\|_F \leq (k+1)^{3/2} u \|A\|_F, \tag{3.44}$$

or the less expensive

$$\|\hat{B}_i\|_F \leq (k+1)^{3/2} u \|B_1\|_F, \tag{3.45}$$

where $\hat{B}_i$ is the computed ith block pivot. With the Bunch-Parlett pivoting strategy, $\|A\|_M \approx \|B_1\|_M$ and $\|\hat{A}_{k+1}\|_M \approx \|B_i\|_M$, where $\text{diag}(B_1, B_2, \ldots, B_{i-1}) \in R^{k\times k}$. Therefore, (3.44) and (3.45) are related.

One potential problem is that continuing the factorization on $\hat{S}_r(A)$ could be unstable for rank estimation. However, the element growth is well-controlled by pivoting. The dimensions of Schur complements are reduced, whereas the upper bounds in (3.44) and (3.45) are increased during factorization. These properties safeguard the stability of rank estimation.

Our experiments were on a laptop with machine epsilon $2^{-52} \approx 2.22 \times 10^{-16}$, with implementation based on `newmat10` library[9]. Using Bunch-Parlett or fast Bunch-Parlett pivoting strategy with stopping criterion (3.44), the estimated ranks were all correct. Using the Bunch-Parlett and fast Bunch-Parlett pivoting strategy with stopping criterion (3.45), there were 26 (0.009%) and 53 (0.019%) errors, respectively. Using bounded Bunch-Kaufman pivoting strategy with stopping criteria (3.44) and (3.45), there were 1,067 (0.375%) and 15 (0.005%) errors, respectively.

We also noted that while using the fast Bunch-Parlett pivoting strategy or bounded Bunch-Kaufman pivoting strategy with stopping criterion (3.45), increasing $(k+1)^{3/2}$ in (3.45) (e.g., to $n^{3/2}$) can sometimes slightly improve the

[9] `newmat` is a library of matrix operations written in C++ by Robert Davies.

accuracy of the estimated ranks, but it usually seriously affected the stability in the further experiment with $\sigma = 10^{-13}$ for all three pivoting strategies.

The further experiment with $\sigma = 10^{-13}$ and stopping criterion (3.45) exhibited the minor instability since the conditioning of the nonsingular part of a matrix affects the stability of rank estimation. Using the Bunch-Parlett, fast Bunch-Parlett, and bounded Bunch-Kaufman pivoting strategies, there were 503 (0.884%), 488 (0.857%), and 939 (1.651%) errors, respectively.

Forcing all the nonzero eigenvalues to be positive in the three test sets, we also experimented with rank estimation of positive semidefinite matrices by LDL^T factorization. For these positive semidefinite matrices, the Bunch-Parlett and fast Bunch-Parlett pivoting strategies are equivalent to complete pivoting[10]. With stopping criteria (3.44) and (3.45), all the estimated ranks were accurate for $\sigma = 1, 10^{-3}, \ldots, 10^{-12}$. Minor instability (less than 5% errors) occurred with $\sigma = 10^{-13}$ and stopping criterion (3.45).

The stopping criteria suggested by Higham [38] for rank estimation of positive semidefinite matrices by Cholesky factorization are in the same form as (3.44) and (3.45), but replacing $(k+1)^{3/2}$ by n. Using his stopping criteria with complete pivoting all the estimated ranks of the semidefinite matrices were accurate for $\sigma = 1, 10^{-3}, \ldots, 10^{-12}$ and also $\sigma = 10^{-13}$. However, his stopping criteria resulted in less accuracy than (3.44) and (3.45) for indefinite matrices.

[10] Applying the Bunch-Parlett or fast Bunch-Parlett pivoting strategy to a symmetric semidefinite matrix requires traversing only the diagonal and one column for each Schur complement. Applying fast Bunch-Kaufman pivoting strategy to a symmetric semidefinite matrix requires traversing at most two columns for each Schur complement. Therefore they are as inexpensive as partial pivoting.

In our experiments with $\sigma = 1, 10^{-3}, \ldots, 10^{-12}$ (i.e., when the rank is not ambiguous), both stopping criteria (3.44) and (3.45) work well while using the Bunch-Parlett or fast Bunch-Parlett pivoting strategies. However, they may not be the best for all matrices. *A priori* information about the matrix, such as the size of $\|W\|$, the growth factor and distribution of nonzero eigenvalues, may help adjust the stopping criterion.

3.5 Concluding Remarks

Table 3.3 lists the highest order terms of the bounds on $\|C\|_S$ for symmetric matrices and symmetric triadic matrices, with references to relevant equations[11]. For LBL^T factorization, the constant c is from (3.6). For singular matrices, r denotes the rank. Note the improvement of bounds on backward errors when triadic structure is present.

Table 3.3: Bounds on $\|C\|_S$ for LDL^T and LBL^T factorizations of $A \in R^{n\times n}$.

Bounds on $\|C\|_S$		*Def.*	*General*		*Triadic*	
LDL^T	Nonsingular	(3.2)	$\frac{1}{3}n^3u$	(3.4)	$9nu$	(3.5)
	Solving $Ax=b$	(3.15)	$\frac{11}{6}n^3u$	(3.16)	$7n^2u$	(3.17)
	Singular	(3.29)	$(nr(n-r)+\frac{1}{3}r^3)u$	(3.30)	$18ru$	(3.31)
LBL^T	Nonsingular	(3.10)	$\frac{4}{3}n^3u$	(3.11)	$4(7c+36)nu$	(3.12)
	Solving $Ax=b$	(3.18)	$\frac{17}{6}n^3u$	(3.19)	$28n^2u$	(3.20)
	Singular	(3.36)	$(4nr(n-r)+\frac{4}{3}r^3)u$	(3.37)	$36(cr+4r+1)u$	(3.38)

We have studied the componentwise backward error analysis and normwise

[11]It is possible to reduce the bounds (3.17) and (3.20) from $O(n^2u)$ to $O(nu)$, but it is not of our main interest.

analysis for LBL^T factorization and its applications to solving linear systems and rank estimation. Our concluding remarks are listed below.

1. LDL^T factorization and its use in solving linear systems are backward stable if the growth factor is modest. Both positive definiteness and diagonal dominance guarantee the stability, because the growth factors are bounded by 1 and 2, respectively. A modest growth factor does not guarantee a stable LBL^T factorization. Nevertheless, LBL^T factorization and its application to solve symmetric linear systems are backward stable if conditions (3.6) and (3.22) hold. All the pivoting strategies in the literature [5, 10, 12, 13, 14] satisfy both conditions.

2. In [39] and [40], Higham proved the stability of the Bunch-Kaufman pivoting strategy [12] and Bunch's pivoting method [10], respectively. His componentwise backward error analysis is based on the LBL^T factorization in outer product form. In this chapter, we presented a new proof of the componentwise backward stability using an inner product formulation. We also gave in Theorem 3.9 a sufficient condition such that an LBL^T factorization is guaranteed to run to completion with inertia preserved.

3. We also analyzed the rank estimation of symmetric indefinite matrices using LBL^T factorization with Bunch-Parlett, fast Bunch-Parlett or bounded Bunch-Kaufman pivoting. In our experiments, both stopping criteria (3.44) and (3.45) work well with the Bunch-Parlett and fast Bunch-Parlett pivoting strategies for $\sigma = 1, 10^{-3} \ldots, 10^{-12}$ (i.e., when rank is not ambiguous). We recommend (3.45) for its low cost, and the fast Bunch-Parlett pivoting strategy for its efficiency without losing much accuracy.

4. Due to the sparsity, the stability of LDL^T and LBL^T factorizations is improved for symmetric triadic matrices, as shown by the growth factor bounds (see Table 3.3) and backward error bounds (see Table 3.3).

Chapter 4

Newton-Type Optimization

Newton-like methods solve nonlinear programming problems that have twice continuously differentiable objective function and constraint functions. At each iteration, a search direction p is computed by solving a linear system $Hp = -g$. For unconstrained nonlinear optimization, H is the Hessian matrix and g is the gradient of the objective function.

Recall that Newton's method often yields a quadratic rate of convergence but can fail to converge, particularly when H is not positive definite. In that case the computed search direction may not even be a descent direction. *Modified Newton methods* add a perturbation E to H, so that $H + E$ is positive definite, where E is symmetric positive semidefinite.

In this chapter we review the Newton's method for unconstrained optimization and its use in the interior point methods for nonlinear programming with inequality and/or equality constraints. More details can be found in [47, Chapters 10,15–17].

4.1 Unconstrained Nonlinear Optimization

We begin with *convex programming*, a particular case of nonlinear programming.

Definition 4.1 *A set Ω is called convex, if for all $x, y \in \Omega$, $tx + (1-t)y \in \Omega$ for $0 < t < 1$. A function $f : \Omega \to R$ is convex on a convex set Ω, if*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y)$$

for all $x, y \in \Omega$ and $0 < t < 1$. In addition, f is strictly convex if

$$f(tx + (1-t)y) < tf(x) + (1-t)f(y)$$

for all $x, y \in \Omega$ and $0 < t < 1$. A function f is concave (or strictly concave), if $(-f)$ is convex (or strictly convex), respectively. A convex programming problem is to minimize a convex function on a convex set.

Given a twice continuously differentiable function $f : R^n \to R$, f is convex on a convex domain Ω, if and only if its Hessian matrix is positive semidefinite for $x \in \Omega$. If its Hessian matrix is positive definite for $x \in \Omega$, then f is strictly convex, but not vice versa (e.g., $f(x) = x^4$).

In this section we consider the problem of minimizing a twice continuously differentiable function $f : R^n \to R$. The quadratic approximation of $f(x)$ around the current estimate x is

$$f(x + \Delta x) \approx f(x) + \Delta x^T g(x) + \frac{1}{2}\Delta x^T H(x)\Delta x,$$

where $g(x)$ and $H(x)$ are the gradient and the Hessian matrix of $f(x)$, respectively. Setting the derivative to be 0, we solve

$$H(x)\Delta x = -g(x) \tag{4.1}$$

to find a vector Δx that minimizes the quadratic approximation. Newton's method uses this as a search direction for minimizing $f(x)$, noting that for a step length $\alpha > 0$,

$$f(x + \alpha\Delta x) = f(x) + \alpha\Delta x^T g(x) + \frac{1}{2}\alpha^2 \Delta x^T H(x + \beta\Delta x)\Delta x \tag{4.2}$$

for some $\beta \in (0, \alpha)$. If $f(x)$ is convex, then $H(x)$ is positive semidefinite. Assume $H(x)$ is positive definite and $g(x) \neq 0$. Then $\Delta x \neq 0$ and $\Delta x^T g(x) = -\Delta x^T H(x) \Delta x < 0$. Therefore,

$$f(x + \alpha \Delta x) < f(x)$$

if $\alpha > 0$ is small enough. In other words, Δx is a descent direction.

4.2 Nonlinear Programming with Inequality Constraints

Nonlinear problems with inequality constraints can be written in the general form

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & c(x) \geq 0, \end{aligned} \tag{4.3}$$

where for simplicity we assume $f : R^n \to R$, $c : R^n \to R^m$, and $f, c \in C^2$ for $x \in \Omega$. We denote $\Omega = \{x : c(x) \geq 0\}$.

Lemma 4.1 *For any concave functions $c_1, c_2, \ldots, c_m$ on R^n, the set $\{x : c_i(x) \geq 0$ for $i = 1, 2, \ldots, m\}$ is convex.*

Proof Consider the sets $\Omega_i = \{x : c_i(x) \geq 0\}$ for $i = 1, 2, \ldots, m$. For each Ω_i, given $x, y \in R^n$ such that $c_i(x) \geq 0$ and $c_i(y) \geq 0$,

$$c_i(tx + (1-t)y) \geq t c_i(x) + (1-t) c_i(y) \geq 0$$

for $0 < t < 1$. Hence Ω_i is convex. The result is obtained by taking the intersection of these convex sets Ω_i, which is still convex. □

Convex programming requires Ω to be a convex set. We call the constrained problem (4.3) convex (or strictly convex) for the particular case: f is convex (or strictly convex), and $c_1(x), c_2(x), \ldots, c_m(x)$ are concave, where $c(x) = [c_1(x), c_2(x), \ldots, c_m(x)]^T$. By Lemma 4.1, the set Ω is then convex.

The logarithmic barrier function of $f(x)$ for problem (4.3) is

$$B_\mu(x) = f(x) - \mu \sum_{i=1}^{m} \log c_i(x), \tag{4.4}$$

where $\mu > 0$. The minimizer of $B_\mu(x)$ converges to a solution to (4.3) as $\mu \to 0^+$. Setting the derivative of $B_\mu(x)$ to 0, we obtain necessary conditions for minimizing $B_\mu(x)$,

$$\begin{aligned} g(x) - \mu A(x)^T C(x)^{-1} e &= 0 \\ c(x) &> 0, \end{aligned} \tag{4.5}$$

where $g(x)$ is the gradient of $f(x)$, $A_{ij}(x) = [\frac{\partial}{\partial x_j} c_i(x)]$, $C(x) = \mathrm{diag}(c(x))$ and e is a column vector of ones. If problem (4.3) is convex, $B_\mu(x)$ is also convex for $\mu > 0$. In that case, the optimality conditions (4.5) are also sufficient.

If we set $\lambda = \mu C(x)^{-1} e$, or equivalently $\lambda_i c_i(x) = \mu$ for $i = 1, 2, \ldots, m$, where $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_m]^T$, we then obtain the nonlinear system,

$$\begin{aligned} g(x) - A(x)^T \lambda &= 0 \\ C(x)\lambda &= \mu e. \end{aligned} \tag{4.6}$$

If the primal problem (4.3) is convex, optimality conditions (4.6) with $c(x) > 0$ are also sufficient for minimizing $B_\mu(x)$ for $\mu > 0$. Therefore, the solution to system (4.6) converges to a solution to problem (4.3) as $\mu \to 0^+$. The system (4.6) defines the *central path*. We follow the path as $\mu \to 0^+$ to approach the solution.

Let $\Lambda = \text{diag}(\lambda)$, let $H(x)$ denote the Hessian matrix of $f(x)$, and let

$$H_L(x,\lambda) = H(x) - \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(x)$$

denote the Hessian matrix of the Lagrangian $f(x) - \lambda^T c(x)$. Applying Newton's method to solve the nonlinear system (4.6), the search direction is determined by solving the linear system,

$$\begin{bmatrix} H_L(x,\lambda) & -A(x)^T \\ \Lambda A(x) & C(x) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} g(x) - A(x)^T\lambda \\ C(x)\lambda - \mu e \end{bmatrix}.$$

Denote the gradient of $B_\mu(x)$ by $g_B(x) = g(x) - \mu A(x)^T C(x)^{-1} e$. Taking $A(x)^T C(x)^{-1}$ times the second equation set and adding it to the first, we obtain the linear system,

$$\begin{aligned} H^*(x,\lambda)\Delta x &= -g(x) + A(x)^T\lambda - A(x)^T C(x)^{-1}(C(x)\lambda - \mu e) \\ &= -g(x) + \mu A(x)^T C(x)^{-1} e \\ &= -g_B(x), \end{aligned} \tag{4.7}$$

where

$$H^*(x,\lambda) = H_L(x,\lambda) + A(x)^T C(x)^{-1} \Lambda A(x).$$

Assuming $H^*(x,\lambda)$ is nonsingular, we can obtain Δx by solving (4.7), and then

$$\Delta\lambda = -C(x)^{-1}(\Lambda A(x)\Delta x + C(x)\lambda - \mu e).$$

The new estimates are taken as $x + \alpha\Delta x$ and $\lambda + \alpha\Delta\lambda$, where $\alpha > 0$ denotes a step length. Taylor series applied to the barrier function $B_\mu(x)$ gives

$$B_\mu(x + \alpha\Delta x) = B_\mu(x) + \alpha\Delta x^T g_B(x) + \frac{1}{2}\alpha^2 \Delta x^T H_B(x + \beta\Delta x)\Delta x, \tag{4.8}$$

for some β satisfying $0 \leq \beta \leq \alpha$. If the primal problem (4.3) is convex, and the estimates are feasible (i.e., $c(x) \geq 0$ and $\lambda \geq 0$), then $B_\mu(x)$ is convex and $H_B(x)$ is positive semidefinite. Assume $H^*(x, \lambda)$ is positive definite and $g_B(x) \neq 0$. Then $\Delta x \neq 0$ and

$$\Delta x^T g_B(x) = -\Delta x H^*(x, \lambda) \Delta x < 0.$$

Therefore, when $\alpha > 0$ is small enough, we get

$$B_\mu(x + \alpha \Delta x) < B_\mu(x).$$

In other words, Newton's method produces a descent direction Δx when $H^*(x, \lambda)$ is positive definite. By driving μ down to 0^+ and keeping estimates strictly feasible, we obtain a solution to problem (4.3).

4.3 Nonlinear Programming with Equality and Inequality Constraints

Nonlinear problems with equality and inequality constraints can be written in the general form,

$$\begin{aligned} \min_x \quad & f(x) \\ \text{subject to} \quad & d(x) = 0 \\ & c(x) \geq 0, \end{aligned} \tag{4.9}$$

where for simplicity we assume $f : R^n \to R$, $c : R^n \to R^m$, $d : R^n \to R^k$ with $k < n$, and $f, d, c \in C^2$ for $x \in \Omega := \{x : d(x) = 0, c(x) \geq 0\}$.

If the equality constraints $d(x) = 0$ are linear, this problem can be simplified to be a nonlinear programming with only inequality constraints as follows. Let

$d(x) = Ex - b$, and let U denote a full-rank matrix whose columns are a basis for the null space of E. Choose a vector $\overline{x}$ to satisfy $E\overline{x} - b = 0$, and then $E(\overline{x}+Uv)-b = 0$ for any vector v. Define $\overline{f}(v) = f(\overline{x}+Uv)$ and $\overline{c}(v) = c(\overline{x}+Uv)$. Problem (4.9) is transformed into

$$\begin{aligned} \min_{v} \quad & \overline{f}(v) \\ \text{subject to} \quad & \overline{c}(v) \geq 0, \end{aligned} \tag{4.10}$$

where $\overline{f}, \overline{c} \in C^2$ for $v \in \{v : \overline{x} + Uv \in \Omega\}$.

Recall that convex programming requires minimization over a convex set. The constrained problem (4.9) is convex (or strictly convex) for the particular case: f is convex (or strictly convex), $d(x)$ is linear, and $c_1(x), c_2(x), \ldots, c_m(x)$ are concave, where $c(x) = [c_1(x), c_2(x), \ldots, c_m(x)]^T$. By Lemma 4.1, the constrained set Ω is convex[1].

Note that if problem (4.9) is a convex (or strictly convex) programming problem, then the reduced problem (4.10) is also a convex (or strictly convex) programming problem, respectively.

The logarithmic barrier function for problem (4.9) is

$$B_\mu(x) = f(x) - \mu \sum_{i=1}^{m} \log c_i(x), \tag{4.11}$$

where $c(x) = [c_1(x), c_2(x), \ldots, c_m(x)]^T$ and $\mu > 0$. The minimizers of $B_\mu(x)$ subject to constraints $d(x) = 0$ converge to a solution to (4.9) as $\mu \to 0^+$. Its Lagrangian is

$$L_\mu(x, y) = f(x) - \mu \sum_{i=1}^{m} \log c_i(x) - y^T d(x),$$

[1]Note that the intersection of the two convex sets $\{x : c(x) \geq 0\}$ and $\{x : Ex - b = 0\}$ is still convex.

where $y = [y_1, y_2, \ldots, y_k]^T$ are Lagrange multipliers. Setting the derivative of $L_\mu(x, y)$ to 0, we obtain the necessary conditions for minimizing $B_\mu(x)$ subject to constraints $d(x) = 0$,

$$\begin{aligned} g(x) - \mu A(x)^T C(x)^{-1} e - E(x)^T y &= 0 \\ d(x) &= 0 \\ c(x) &> 0, \end{aligned} \tag{4.12}$$

where $g(x)$ is the gradient of $f(x)$, $A_{ij}(x) = [\frac{\partial}{\partial x_j} c_i(x)]$, $C(x) = \text{diag}(c(x))$, $d(x) = [d_1(x), d_2(x), \ldots, d_k(x)]^T$, $E_{ij}(x) = [\frac{\partial}{\partial x_j} d_i(x)]$ and e is a column vector of ones. Setting $\lambda = \mu C(x)^{-1} e$, or equivalently $\lambda_i c_i(x) = \mu$ for $i = 1, 2, \ldots, m$, where $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_m]^T$, then we obtain the nonlinear system,

$$\begin{aligned} g(x) - A(x)^T \lambda - E(x)^T y &= 0 \\ d(x) &= 0 \\ C(x)\lambda &= \mu e. \end{aligned} \tag{4.13}$$

If the primal problem (4.9) is convex, optimality conditions (4.13) are also sufficient for minimizing $B_\mu(x)$. Therefore, the solution to system (4.13) converges to a solution to problem (4.9) as $\mu \to 0^+$. The system (4.13) defines the *central path*. We follow the path as $\mu \to 0^+$ to approach the solution.

The Jacobian of (4.13) is

$$\begin{bmatrix} H_L(x, y, \lambda) & -E(x)^T & -A(x)^T \\ E(x) & 0 & 0 \\ \Lambda A(x) & 0 & C(x) \end{bmatrix},$$

where $\Lambda = \text{diag}(\lambda)$, and

$$H_L(x, y, \lambda) = H(x) - \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(x) - \sum_{i=1}^{k} y_i \nabla^2 d_i(x) \tag{4.14}$$

is the Hessian matrix of the Lagrangian $f(x) - \lambda^T c(x) - y^T d(x)$, and $H(x)$ is the Hessian matrix of $f(x)$. Applying Newton's method to solve nonlinear system (4.13), the search direction is determined by solving the linear system,

$$\begin{bmatrix} H_L(x,y,\lambda) & -E(x)^T & -A(x)^T \\ E(x) & 0 & 0 \\ \Lambda A(x) & 0 & C(x) \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = - \begin{bmatrix} g(x) - A(x)^T\lambda - E(x)^T y \\ d(x) \\ C(x)\lambda - \mu e \end{bmatrix}.$$

Assuming $c(x) > 0$, we take $A(x)^T C(x)^{-1}$ times the third equation set, add it to the first, and then obtain the reduced *KKT (Karush-Kuhn-Tucker) system*,

$$\begin{bmatrix} H^*(x,y,\lambda) & E(x)^T \\ E(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ -\Delta y \end{bmatrix} = - \begin{bmatrix} g_B(x) - E(x)^T y \\ d(x) \end{bmatrix}, \tag{4.15}$$

where $g_B(x) = g(x) - \mu A(x)^T C(x)^{-1} e$ is the gradient of $B_\mu(x)$, and

$$H^*(x,y,\lambda) = H_L(x,y,\lambda) + A(x)^T C(x)^{-1} \Lambda A(x). \tag{4.16}$$

Assuming $H^*(x,y,\lambda)$ is nonsingular and $E(x)$ has full row rank, we take $-E(x)H^*(x,y,\lambda)^{-1}$ times the first equation set, add it to the second, and then obtain[2]

$$E(x)H^*(x,y,\lambda)^{-1}E(x)^T \Delta y = E(x)H^*(x,y,\lambda)^{-1}(g_B(x) - E(x)^T y) - d(x).$$

Therefore,

$$\Delta x = H^*(x,y,\lambda)^{-1}(E(x)^T(y + \Delta y) - g_B(x)), \tag{4.17}$$

[2]In practice, computing the inverse of a matrix is discouraged. Therefore, we factor $H^*(x,y,\lambda) = LL^T$, solve $LZ = E(x)^T$ for Z. The right-hand side involving $H^*(x,y,\lambda)^{-1}$ can also be calculated via L and Z. Then we can solve the symmetric linear system with matrix $Z^T Z$ for Δy. When a modified LBL^T or LTL^T factorization from Sections 5.3 or 5.4 is used, the linear system has the matrix in the form $Z^T B^{-1} Z$ or $Z^T T^{-1} Z$. In this case we can use the same trick to avoid computing the inverse. Note that for T we can take the advantage of the triadic properties described in Chapter 2. See [36] for alternatives for solving KKT systems.

and then

$$\Delta\lambda = -C(x)^{-1}(\Lambda A(x)\Delta x + C(x)\lambda - \mu e).$$

The new estimates are taken as $x + \alpha\Delta x$, $y + \alpha\Delta y$ and $\lambda + \alpha\Delta\lambda$, where $\alpha > 0$ denotes a step length.

If the equality constraints $d(x) = 0$ are linear, denoted by $d(x) \equiv Ex - b = 0$, then $E(x) \equiv E$, a constant matrix. We also assume the equality constraints are regularized (i.e., E has full row rank). Suppose we begin with feasible estimates satisfying $d(x) \equiv Ex - b = 0$. By the second equation set of (4.15), $E\Delta x = 0$, which implies that the new estimate $x + \alpha\Delta x$ still satisfies the linear constraints. We may adjust the step length $\alpha > 0$ to keep new estimates strictly feasible. Taking Δx^T times the first equation set of system (4.15), we obtain

$$\begin{aligned} -\Delta x^T g_B(x) &= \Delta x^T H^*(x, y, \lambda)\Delta x - \Delta x^T E^T(y + \Delta y) \\ &= \Delta x^T H^*(x, y, \lambda)\Delta x. \end{aligned} \tag{4.18}$$

If the primal problem (4.9) is convex and estimates are feasible, then $B_\mu(x)$ is convex, and $H_L(x, y, \lambda)$ in (4.14) and $H^*(x, y, \lambda)$ in (4.16) are positive semidefinite. Assume $H^*(x, y, \lambda)$ is positive definite and $\Delta x \neq 0$, $\Delta x^T g_B(x) < 0$ because of (4.18). Using Taylor series (4.8) for $B_\mu(x)$, $B_\mu(x + \alpha\Delta x) - B_\mu(x) < 0$ when $\alpha > 0$ is small enough. In other words, Newton's method gives a descent search direction Δx when $H^*(x, y, \lambda)$ is positive definite.

4.4 Modified Newton Methods

We have shown that if the matrix $H(x)$ in (4.1) or $H^*(x, \lambda)$ in (4.7) or $H^*(x, y, \lambda)$ in (4.17) is positive definite, then the computed Δx obtained from Newton's method is a descent search direction for $f(x)$ or for the barrier function $B_\mu(x)$.

This fact, coupled with the fast local convergence rate of Newton's method, gives a very effective algorithm. For all linear systems (4.1), (4.7), and (4.17), we assumed that the right-hand side is nonzero. Otherwise we compute a direction of *negative curvature* instead of a descent direction [25, 45, 46].

For unconstrained nonlinear optimization, if $f(x)$ is not strictly convex, then its Hessian matrix $H(x)$ may not be positive definite, and the computed search direction may not be a descent direction. In such a case, we can add a perturbation to $H(x)$ so that the resulting $\hat{H}(x)$ is positive definite. The solution from $\hat{H}(x)\Delta x = -g(x)$ satisfies

$$\Delta x^T g(x) = -\Delta x \hat{H}(x) \Delta x < 0$$

in (4.2). Hence the resulting search direction Δx is a descent direction.

Similarly, if H^* is not positive definite in (4.7) or (4.17), we can replace it by a nearby positive definite matrix in order to obtain a descent direction Δx for the barrier function (4.4) or (4.11).

Algorithms to perturb an indefinite Hessian to make it positive definite are called *modified Newton methods*. Modified Newton methods in the literature are implemented via Cholesky factorization [28, Chapter 4][54, 55], LBL^T factorization [16, 45] or the Lanczos method [46]. We develop alternatives in the next chapter.

Chapter 5

Modified Cholesky Algorithms

Parts of this chapter are drawn from material in [24].

Modified Cholesky algorithms are widely used in nonlinear optimization to compute Newton-like directions. Given a symmetric possibly indefinite $n \times n$ matrix A approximating the Hessian of a function to be minimized, the goal is to find a positive definite matrix $\hat{A} = A + E$, where E is small. The search direction Δx is then computed by solving the linear system $(A+E)\Delta x = -g(x)$ where $g(x)$ is the gradient of the function to be minimized. The proposed four objectives to be achieved when computing E are listed below [16, 54, 55].

Objective 1. If A is sufficiently positive definite, $E = 0$.

Objective 2. If A is not positive-definite, $\|E\|$ is not much larger than $\inf\{\|\Delta A\| : A + \Delta A \text{ is positive definite}\}$ for some reasonable norm.

Objective 3. The matrix $A + E$ is reasonably well-conditioned.

Objective 4. The cost of the algorithm is only a small multiple of n^2 higher than that of the standard Cholesky factorization, which takes $\frac{1}{3}n^3 + O(n^2)$ flops ($\frac{1}{6}n^3 + O(n^2)$ multiplications and $\frac{1}{6}n^3 + O(n^2)$ additions).

Objective 1 ensures that the fast convergence of Newton-like methods on convex programming problems is retained by the modified Cholesky algorithms. Objective 2 keeps the search direction close to Newton's direction, while Objective 3 implies numerical stability when computing the search direction. Objective 4 makes the work in computing the modification small relative to the work in factoring a dense matrix.

Given a diagonal matrix $A = \text{diag}(d_1, d_2, \ldots, d_n)$, we can make $A + E = \text{diag}(\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_n)$ positive definite by choosing $\hat{d}_k := \max\{|d_k|, \delta\}$ for $k = 1, \ldots, n$, where $\delta > 0$ is a preset small tolerance. A modification algorithm like this is called a Type-I algorithm. Alternatively, we can make $\hat{d}_k := \max\{d_k, \delta\}$ for $k = 1, \ldots, n$. We call modified Cholesky algorithms of this kind Type-II algorithms. In both types of algorithms, δ must be kept small to satisfy Objective 2, but large enough to satisfy Objective 3.

Early approaches were of Type-I [28, Chapter 4][45], whereas more recently the Type-II algorithms have prevailed [16, 54, 55].

There are three useful factorizations of a symmetric matrix A as $PAP^T = LXL^T$, where P is a permutation matrix for pivoting, and L is unit lower triangular.

1. If X is diagonal, it is called the LDL^T factorization[1].

2. If X is block diagonal with block order 1 or 2, it is called the LBL^T factorization [5, 12, 14].

3. If X is a tridiagonal matrix, and the off-diagonal elements in the first column are all zero, it is called the LTL^T factorization [1, 50].

[1] If D is nonnegative, it is the Cholesky factorization in the LDL^T form.

The existing modified Cholesky algorithms use either the LDL^T factorization [28, Chapter 4][54, 55] or the LBL^T factorization [16, 45]. We present new modified LDL^T factorizations and an approach via the LTL^T factorization.

In all we review five modified Cholesky algorithms in the literature and give five new ones, each of which depends on a modification tolerance parameter $\delta > 0$. Satisfaction of Objectives 1–3 is measured by bounds, discussed in detail as the algorithms are introduced, and referenced in Table 5.1, where the new algorithms are in boldface.

Table 5.1: Satisfaction of the four objectives for Modified Cholesky algorithms.

Algorithm	*Type*	δ	*Obj. 1*	*Obj. 2*	*Obj. 3*	*Obj. 4*
GMW81	I	ϵ_M	(5.4)	(5.3)	(5.25)	$O(n^2)$
GMW-I	I	ϵ_M	(5.32)	(5.22)	(5.25)	$O(n^2)$
GMW-II	II	$\bar{\tau}\eta$	(5.32)	(5.24)	(5.25)	$O(n^2)$
SE90	II	$\tau\eta$	(5.32)	(5.12) (5.13)	(5.33)	$O(n^2)$
SE99	II	$\bar{\tau}\eta$	(5.32)	(5.12) (5.19)	(5.33)	$O(n^2)$
SE-I	I	$\bar{\tau}\eta$	(5.32)	(5.29) (5.19)	(5.33)	$O(n^2)$
MS79	I	ϵ_M	(5.34)	(5.35)	(5.41)	$\leq O(n^3)$
CH98	II	$\sqrt{u}\|A\|_\infty$	(5.34)	(5.36)	(5.42)	$\leq O(n^3)$
LTL^T**-MS79**	I	ϵ_M	(5.45)	(5.46)	(5.48)	$O(n^2)$
LTL^T**-CH98**	II	$\bar{\tau}\eta$	(5.45)	(5.47)	(5.49)	$O(n^2)$

Table 5.2 lists some notation used in this chapter. We use $\mathrm{diag}(a_1, \ldots, a_n)$ to denote the diagonal matrix formed by $a_1, \ldots, a_n$, and $\mathrm{Diag}(A)$ to denote the diagonal matrix formed by the diagonal of matrix A.

Table 5.2: Notation.

Symbol	*Description*
ϵ_M	machine epsilon
u	unit roundoff, $\epsilon_M/2$
A	an $n \times n$ symmetric matrix
ξ	maximum magnitude of off-diagonal elements of A
η	maximum magnitude of diagonal elements of A
$\lambda_i(A)$	ith smallest eigenvalue of A
$\lambda_{\min}(A)$	smallest eigenvalue of A
$\lambda_{\max}(A)$	largest eigenvalue of A
τ	$\sqrt[3]{\epsilon_M}$
$\bar{\tau}$	$\sqrt[3]{\epsilon_M^2}$

The organization of this chapter is as follows. Section 5.1 presents the modified LDL^T factorizations in the literature and Section 5.2 presents our variants inspired by the existing algorithms. Section 5.3 describes the modified LBL^T factorizations in the literature. Section 5.4 gives our new LTL^T algorithms. Section 5.5 summarizes the results of our computational tests. Concluding remarks are given in Section 5.6.

5.1 Modified LDL^T Algorithms

If we are given a LDL^T factorization of a symmetric matrix A, a naïve way to modify A to be positive definite is by making nonpositive elements in the diagonal matrix D positive. However, this method fails to meet Objective 2. For example,

in the following LDL^T factorization

$$A = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \frac{1}{\epsilon} & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 0 \\ 0 & -\frac{1}{\epsilon} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{\epsilon} \\ 0 & 1 \end{bmatrix} = LDL^T,$$

the modification is unbounded when $\epsilon \to 0^+$. Another 3×3 example is given in [28, Chapter 4]. If a given symmetric matrix A is not positive semidefinite, its LDL^T factorization may not even exist (e.g., $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$).

A *modified LDL^T algorithm* for a positive definite $\hat{A} = A + E$ typically has $E = \text{diag}(\delta_1, \delta_2, \ldots, \delta_n)$ diagonal, and computes $\delta_k \geq 0$ at the kth step, for $k = 1, \ldots, n$ during the factorization in inner product form. Denote the Schur complement at the kth step by $A_k = \begin{bmatrix} a_k & c_k^T \\ c_k & \bar{A}_k \end{bmatrix}$ for $k = 1, \ldots, n$, where $a_k \in R$ and c_k is a column vector of $n-k$ elements. Initially $A_1 := A$. The factorization can be computed by setting[2]

$$L(k+1:n, k:k) := \frac{c_k}{a_k + \delta_k},\ D(k,k) := a_k + \delta_k \text{ and } A_{k+1} := \bar{A}_k - \frac{c_k c_k^T}{a_k + \delta_k} \quad (5.1)$$

for $k = 1, \ldots, n-1$. The challenge is to determine δ_k to satisfy the four objectives. All the algorithms in Sections 5.1 and 5.2 follow this model. We may optionally incorporate a diagonal pivoting strategy. In other words, at the kth step, we symmetrically interchange rows and columns to ensure that $|a_k| \geq |A_k(j,j)|$ (pivoting on the diagonal element of maximum magnitude) or $a_k \geq A_k(j,j)$ (pivoting on the element of maximum value) for $j = 1, \ldots, n-k$. The resulting modified LDL^T factorization is in the form

$$P(A + E)P^T = LDL^T = \bar{L}\bar{L}^T, \quad (5.2)$$

[2]Schnabel and Eskow [54, 55] formulated their algorithm in the LL^T form, whereas we present the model in the LDL^T form.

where P is the permutation matrix for pivoting.

Gill and Murray introduced a stable algorithm in 1974 [27]. It was subsequently refined by Gill, Murray, and Wright in 1981 [28, Chapter 4]. We call it the GMW81 algorithm hereafter. Schnabel and Eskow introduced another modified LDL^T algorithm in 1990 [54]. It was subsequently revised in 1999 [55]. We call them the SE90 and SE99 algorithms, respectively.

5.1.1 The GMW81 Algorithm

Consider the general model (5.1). The GMW81 algorithm determines δ_k by setting

$$a_k + \delta_k = \max\{\delta, |a_k|, \frac{\|c_k\|_\infty^2}{\beta^2}\}$$

for $k = 1, \ldots, n$, where $\beta > 0$ and the small tolerance $\delta > 0$ are preset. We set $\delta := \epsilon_M$ (machine epsilon) as is common in the literature [16, 55].

The rationale behind the GMW81 algorithm is that β becomes a bound on the magnitude of the off-diagonal elements in the lower triangular matrix $\bar{L}$ of the Cholesky factorization in (5.2). The challenge is to choose β such that $\|E\|_2$ is well-controlled and Objective 1 is satisfied. The correction E in 2-norm is bounded by

$$\|E\|_2 \leq (\frac{\xi}{\beta} + (n-1)\beta)^2 + 2(\eta + (n-1)\beta^2) + \delta =: f(\beta), \tag{5.3}$$

where η and ξ are the maximum magnitudes of the diagonal and off-diagonal elements of A, respectively. Note that since E is diagonal, its 1-norm, 2-norm and ∞-norm are the same.

The overall extra cost of the GMW81 algorithm relative to the standard Cholesky factorization is $O(n^2)$, so Objective 4 is satisfied. Now we consider

Objective 2. The minimum of (5.3) is

$$\min_{\beta} f(\beta) = 2\xi(\sqrt{n^2-1}+n-1)+2\eta+\delta \leq 4n\xi+2\eta+\delta,$$

which is attained with $\beta^2 = \frac{\xi}{\sqrt{n^2-1}}$ for $n > 1$.

A diagonal pivoting strategy is used in the GMW81 algorithm. The pivot is chosen as the maximum magnitude diagonal element[3].

To satisfy Objective 1, we let $\beta^2 \geq \eta$, so that $E = 0$ if A is sufficiently positive definite [27]. More precisely, $E = 0$ if $\beta^2 \geq \eta$ and

$$\lambda_{\min}(A) \geq \delta. \tag{5.4}$$

Therefore, β is chosen by

$$\beta^2 := \max\{\eta, \frac{\xi}{\sqrt{n^2-1}}, \epsilon_M\} \tag{5.5}$$

for $n > 1$. Substituting it into (5.3), we obtain $\|E\|_2 = O(n^2)$.

5.1.2 The SE90 Algorithm

The SE90 algorithm was inspired by a lemma related to the Gershgorin circle theorem [42, page 344]. We begin with the Gershgorin circle theorem and then the lemma.

Theorem 5.1 (Gershgorin) *Given $A \in C^{n\times n}$, define the i-th Gershgorin radius and circle by*

$$R_i(A) := \sum_{j=1, j\neq i}^{n} |a_{ij}| \text{ and } C_i(A) := \{z : |z - a_{ii}| \leq R_i(A)\}$$

for $1 \leq i \leq n$. Then the eigenvalues of A are contained in the union of the Gershgorin circles $\bigcup_{i=1}^{n} C_i(A)$.

[3]Alternatively, we could pivot on the maximum diagonal element, but pivoting on the maximum magnitude usually gives a smaller $\|E\|_2$ in our experiments.

By Theorem 5.1, the first naïve method to perturb a given symmetric matrix $A \in R^{n\times n}$ to be positive semidefinite is to set $\delta_k := \max\{0, -a_{kk} + R_k(A)\}$ for $k = 1, \ldots, n$. The modification δ_k can be reduced by the following lemma.

Lemma 5.1 *Given a symmetric matrix* $A = \begin{bmatrix} a & c^T \\ c & \bar{A} \end{bmatrix} \in R^{n\times n}$, *suppose we add a perturbation* $\delta \geq \{0, -a + \|c\|_1\}$ *to* a, *so that* $a + \delta \geq \|c\|_1$. *The resulting Schur complement*[4] *is* $\hat{A} := \bar{A} - \frac{cc^T}{a+\delta}$. *Then* $C_i(\hat{A}) \subseteq C_{i+1}(A)$ *for* $i = 1, \ldots, n-1$.

Proof This proof is a condensed version of that in [54]. Let $\bar{a}_{ij}$ and $\hat{a}_{ij}$ denote the (i, j) entries of $\bar{A}$ and $\hat{A}$ respectively for $1 \leq i, j < n$. Also denote $c = [(c)_1, (c)_2, \ldots, (c)_{n-1}]^T$. For $1 \leq i < n$,

$$R_{i+1}(A) - R_i(\hat{A}) = (R_{i+1}(A) - R_i(\bar{A})) + (R_i(\bar{A}) - R_i(\hat{A})).$$

The difference between $R_{i+1}(A)$ and $R_i(\bar{A})$ is $|(c)_i|$. In addition, the ith column of $\bar{A} - \hat{A}$ is $\frac{(c)_i c}{a+\delta}$, whose 1-norm minus $\frac{(c)_i^2}{a+\delta}$ is the upper bound for $|(R_i(\bar{A}) - R_i(\hat{A}))|$. Therefore,

$$\begin{aligned} R_{i+1}(A) - R_i(\hat{A}) &\geq |(c)_i| - \frac{|(c)_i|(\|c\|_1 - |(c)_i|)}{a+\delta} \\ &= |(c)_i|(1 - \frac{\|c\|_1}{a+\delta}) + \frac{(c)_i^2}{a+\delta} \\ &\geq \frac{(c)_i^2}{a+\delta} = \bar{a}_{ii} - \hat{a}_{ii} \geq 0. \end{aligned}$$

This means that the Gershgorin circles contract, and the contraction of each circle is no less than the perturbation of the circle center. Therefore, $C_i(\hat{A}) \subseteq C_{i+1}(A)$ for $i = 1, \ldots, n-1$. □

Following the general model (5.1), the second naïve method to make $A + E$ positive semidefinite arises naturally by setting $\delta_k := \max\{0, -a_k + \|c_k\|_1\}$ for

[4]Note that the ith row/column of $\bar{A}$ corresponds to the $(i+1)$st row/column of A.

$k = 1, \ldots, n$. Note that $a_k - \|c_k\|_1$ is the lower endpoint of the Gershgorin circle $C_1(A_k)$. Repeatedly applying Lemma 5.1, we obtain $\delta_k \leq \max\{0, -a_{kk} + R_k(A)\}$ for $k = 1, \ldots, n$. Taking the maximum of these values and zero, we define

$$\bar{G} := \max\{0, \max\{-a_{kk} + R_k(A) : k = 1, \ldots, n\}\},$$

Then $\|E\|_2 \leq \bar{G} \leq \eta + (n-1)\xi$, where η and ξ are the maximum magnitudes of the diagonal and off-diagonal elements of A, respectively. However, this naïve method may fail to satisfy Objective 1.

To satisfy Objective 1, the SE90 algorithm consists of two phases. The *2-phase strategy* was also presented in [25]. Phase 1 performs steps of the standard Cholesky factorization (i.e., without perturbation, $\delta_k := 0$), as long as all diagonal elements of the next Schur complement are sufficiently positive. The pseudo-code is given in Algorithm 5.1.

Algorithm 5.1 Phase 1 of a 2-Phase Strategy.

{Given a symmetric $A \in R^{n \times n}$ and a small tolerance $\delta > 0$.}

$A_1 := A$, $k := 1$

Pivot on the maximum diagonal element of A_1.

{Denote $A_k = \begin{bmatrix} a_k & c_k^T \\ c_k & \bar{A}_k \end{bmatrix}$, then $\text{Diag}(\bar{A}_k) \leq a_k I_{n-k}$ after pivoting.}

if $a_1 \geq \delta$ **then**

  while $\text{Diag}(\bar{A}_k - \frac{c_k c_k^T}{a_k}) \geq \delta I_{n-k}$ and $k < n$ **do**

    $A_{k+1} := \bar{A}_k - \frac{c_k c_k^T}{a_k}$

    $k := k + 1$

    Pivot on maximum diagonal of A_k.

  end while

end if

The SE90 algorithm uses the tolerance $\delta := \tau\eta$, where η is the maximum magnitude of the diagonal elements of A, and $\tau = \sqrt[3]{\epsilon_M}$. Therefore, in Phase 1,

$$\text{Diag}(A_k) \geq \tau\eta I_{n-k+1} \tag{5.6}$$

for $k = 1, \ldots, \min\{n, K+1\}$, where K is the number of steps in Phase 1. If A is sufficiently positive definite, then $K = n$ and the factorization completes without using Phase 2. Otherwise, Phase 1 ends when setting $\delta_{K+1} := 0$ results in A_{K+2} having a diagonal element less than δ. It is not hard to see that

$$\hat{\eta} \leq \eta \text{ and } \hat{\xi} \leq \xi + \eta, \tag{5.7}$$

where $\hat{\eta}$ and $\hat{\xi}$ (and η and ξ) are the maximum magnitudes of the diagonal and off-diagonal elements of A_{K+1} (and A), respectively [54].

In Phase 2, δ_k is determined by

$$\delta_k := \max\{\delta_{k-1}, -a_k + \max\{\|c_k\|_1, \tau\eta\}\} \leq G + \tau\eta, \tag{5.8}$$

for $k = K+1, \ldots, n-2$, where G is the maximum of zero and the negative of the lowest Gershgorin endpoint of A_{K+1}. For the case $K = 0$, we set $\delta_0 := 0$. The rationale for $\delta_k \geq \delta_{k-1}$ is because increasing δ_k up to δ_{k-1} does not increase $\|E\|_2$ at this point and may possibly reduce the subsequent δ_i for $k < i \leq n$. This *nondecreasing strategy* can be applied to virtually all modified Cholesky algorithms with modifications confined to the diagonal.

In experiments, Schnabel and Eskow [54] obtained a smaller value of $\|E\|_2$ when using special treatment for the final 2×2 Schur complement A_{n-1}, setting

$$\delta_{n-1} = \delta_n := \max\{\delta_{n-2}, -\lambda_1(A_{n-1}) + \max\{\frac{\tau(\lambda_2(A_{n-1}) - \lambda_1(A_{n-1}))}{1-\tau}, \tau\eta\}\} \tag{5.9}$$

$$\leq G + \frac{2\tau}{1-\tau}(G + \eta), \tag{5.10}$$

where $\lambda_1(A_{n-2})$ and $\lambda_2(A_{n-2})$ are the smaller and larger eigenvalues of A_{n-2}, respectively. The last inequality holds because

$$-\lambda_1(A_{n-1}) \leq G \text{ and } \lambda_2(A_{n-1}) - \lambda_1(A_{n-1}) \leq 2(G+\eta).$$

In (5.9), δ_{n-1} and δ_n are chosen to obtain the bound

$$\kappa_2(A_{n-1} + \delta_n I_2) \leq \frac{1 + (\tau/(1-\tau))}{\tau/(1-\tau)} = \frac{1}{\tau}, \tag{5.11}$$

where $I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Finally, by (5.8) and (5.10),

$$\|E\|_2 \leq G + \frac{2\tau}{1-\tau}(G+\eta). \tag{5.12}$$

If $K = 0$, then $G \leq \eta + (n-1)\xi$. By (5.7), if $K > 0$, then

$$G \leq (n - K - 1)(\xi + \eta). \tag{5.13}$$

In either case, $\|E\|_2 = O(n)$. Recall that with the GMW81 algorithm, $\|E\|_2 = O(n^2)$.

Diagonal pivoting is also used in the SE90 algorithm, as well as the later SE99 algorithm. The analysis above does not rely on the pivoting, but pivoting reduces $\|E\|_2$ empirically. In Phase 1, the pivot is chosen as the largest diagonal entry as shown in Algorithm 5.1.

In Phase 2, one may choose the pivot with the largest lower endpoint of the Gershgorin circle in the current Schur complement. This provides the least modification at the current step. In other words, after diagonally interchanging rows and columns, $G_1(A_k) \geq G_i(A_k)$ for $k = K{+}1, \ldots, n{-}2$ and $i = 1, \ldots, n{-}k{+}1$, where $G_i(A_k) = a_{ii} - R_i(A_k)$ is the lower endpoint of the ith Gershgorin circle

$C_i(A_k)$. However, computing all $G_i(A_k)$ in Phase 2 takes $\frac{(n-K)^3}{3}$ additions and fails to satisfy Objective 4. The proof of Lemma 5.1 shows

$$\hat{a}_{ii} - R_i(\hat{A}) \geq \bar{a}_{ii} - R_{i+1}(A) + |(c)_i|(1 - \frac{\|c\|_1}{a+\delta})$$

for $i = 1, \ldots, n-1$. Therefore,

$$G_i(A_{k+1}) \geq G_{i+1}(A_k) + |(c_k)_i|(1 - \frac{\|c_k\|_1}{a_k+\delta_k})$$

for $k = 1, \ldots, n-1$ and $i = 1, \ldots, n-k$. Using this fact, we recursively compute the lower bounds of these Gershgorin intervals by

$$\hat{G}_i(A_{k+1}) := \hat{G}_{i+1}(A_k) + |(c_k)_i|(1 - \frac{\|c_k\|_1}{a_k+\delta_k})$$

for $k = 1, \ldots, n-1$ and $i = 1, \ldots, n-k$. The base cases are $\hat{G}_i(A_1) := G_i(A)$ for $i = 1, \ldots, n$. Computing these estimated lower endpoints $\hat{G}_i(A_{k+1})$ for pivoting takes $2(n-K)^2$ additions and $\frac{1}{2}(n-K)^2$ multiplications. Hence Objective 4 is satisfied.

5.1.3 The SE99 Algorithm

Although the SE90 algorithm has a better *a priori* bound on $\|E\|_2$ than the GMW81 algorithm, there are matrices for which SE90 gives an inordinately large $\|E\|_2$. These matrices are generally close to being positive definite. The SE99 algorithm [55], a modification of the SE90 algorithm, was developed to remedy the excessive modifications in these worst cases. In the SE99 algorithm, condition (5.6) is *relaxed* into the following two conditions that possibly increase the number of Phase 1 pivots. First,

$$\text{Diag}(\bar{A}_k - \frac{c_k c_k^T}{a_k}) \geq -\mu\eta I_{n-k}$$

for some $0 < \mu \leq 1$. Second,

$$\text{Diag}(A_k) \geq -\mu a_k I_{n-k+1}.$$

Schnabel and Eskow suggested $\mu = 0.1$ for their SE99 algorithm [55]. The pseudo-code of the *relaxed 2-phase strategy* is given in Algorithm 5.2.

Algorithm 5.2 Relaxed Phase 1 of a 2-Phase Strategy.

{Given a symmetric $A \in R^{n\times n}$, $\delta > 0$ and $0 < \mu \leq 1$.}

$\eta := \max_{1\leq i\leq n} |A_{ii}|$

if $\text{Diag}(A) \geq -\mu\eta I_n$ **then**

$A_1 := A$, $k := 1$

Pivot on the maximum diagonal element of A_1.

{Denote $A_k = \begin{bmatrix} a_k & c_k^T \\ c_k & \bar{A}_k \end{bmatrix}$, then $\text{Diag}(\bar{A}_k) \leq a_k I_{n-k}$ after pivoting.}

while $a_k \geq \delta$ and $\text{Diag}(A_k) \geq -\mu a_k I_{n-k+1}$ and $\text{Diag}(\bar{A}_k - \frac{c_k c_k^T}{a_k}) \geq -\mu\eta I_{n-k}$ and $k < n$ **do**

$A_{k+1} := \bar{A}_k - \frac{c_k c_k^T}{a_k}$

$k := k + 1$

Pivot on maximum diagonal of A_k.

end while

end if

In the SE99 algorithm, $\delta := \bar{\tau}\eta$, where $\bar{\tau} = \sqrt[3]{\epsilon_M^2}$, is smaller than $\tau = \sqrt[3]{\epsilon_M}$ in the SE90 algorithm, potentially keeping $\|E\|$ smaller. In Phase 1, there is no perturbation, so $\delta_k = 0$ for $k = 1, 2, \ldots, K$, with K the number of steps in Phase 1. The modification in Phase 2 turns out to be

$$\delta_k := \max\{\delta_{k-1}, -a_k + \max\{\|c_k\|_1, \bar{\tau}\eta\}\} \leq G + \bar{\tau}\eta, \tag{5.14}$$

where G is the negative of the lowest Gershgorin endpoint of A_{K+1}. Recall that we set $\delta_0 := 0$ and δ_k is nondecreasing, so that δ_k is nonnegative.

Since small negative numbers are allowed on the diagonal in Phase 1, two changes have to be made. First, we need to check whether $a_k \geq \delta$ at each step, as shown in Algorithm 5.2, whereas it is not required in Algorithm 5.1. Second, it is possible that the SE99 algorithm moves into Phase 2 at the last step (i.e., the number of steps in Phase 1 is $K = n - 1$). In such a case,

$$\delta_n := \max\{0, -a_n + \max\{\frac{-\tau}{1-\tau}a_n, \bar{\tau}\eta\}\} \leq G + \frac{\tau}{1-\tau}G + \bar{\tau}\eta. \tag{5.15}$$

Similar to (5.9) in the SE90 algorithm, the special treatment in the SE99 algorithm for the final 2×2 Schur complement in Phase 2 is

$$\begin{aligned} \delta_{n-1} = \delta_n &:= \max\{\delta_{n-2}, -\lambda_1(A_{n-1}) + \max\{\frac{\tau(\lambda_2(A_{n-1}) - \lambda_1(A_{n-1}))}{1-\tau}, \bar{\tau}\eta\}\} \\ &\leq G + \frac{2\tau}{1-\tau}(G+\eta). \end{aligned} \tag{5.16}$$

By (5.14), (5.15) and (5.16), we obtain

$$\|E\|_2 \leq G + \frac{2\tau}{1-\tau}(G+\eta). \tag{5.17}$$

Although (5.17) for the SE99 algorithm looks the same as (5.12) for the SE90 algorithm, the bound on G in (5.17) is different for $0 < K < n$. Due to relaxing, the bounds (5.7) on $\hat{\eta}$ and $\hat{\xi}$ are replaced by

$$\hat{\eta} \leq \eta \text{ and } \hat{\xi} \leq \xi + (1+\mu)\eta, \tag{5.18}$$

where $\hat{\eta}$ and $\hat{\xi}$ (and η and ξ) are the maximum magnitudes of the diagonal and off-diagonal elements of A_{K+1} (and A), respectively. Therefore, if $0 < K < n$,

$$G \leq (n-K-1)(\xi + (1+\mu)\eta) + \mu\eta. \tag{5.19}$$

Recall that K is the number of steps in Phase 1, and the SE99 algorithm potentially has more steps staying in Phase 1 than the SE90 algorithm.

The pivoting strategy used in the SE99 algorithm is the same as that in the SE90 algorithm. Note that the bound on $\|E\|_2$ in (5.17) for the SE99 algorithm is independent of the pivoting strategy applied, and so is (5.12) for the SE90 algorithm.

5.2 New Modified LDL^T Algorithms

This section presents three variants of the LDL^T algorithms: GMW-I, GMW-II and SE-I, and illustrates their performance. Experiments used a laptop with an Intel Celeron 2.8GHz CPU using IEEE standard arithmetic with machine epsilon $\epsilon_M = 2^{-52} \approx 2.22 \times 10^{-16}$. We measure the size of E by the ratios

$$r_2 = \frac{\|E\|_2}{|\lambda_{\min}(A)|} \text{ and } r_F = \frac{\|E\|_F}{(\sum_{\lambda_i(A)<0} \lambda_i(A)^2)^{1/2}}. \tag{5.20}$$

Note that assuming $\lambda_{\min}(A) < 0$, the denominators are the least modifications to make the matrix positive semidefinite in their corresponding norms.

The random matrices in our experiments are of the form $Q\Lambda Q^T$, where $Q \in R^{n\times n}$ is a random orthogonal matrix computed by the method of G. W. Stewart [60], and $\Lambda \in R^{n\times n}$ is diagonal with uniformly distributed random eigenvalues in $[-1, 10000]$, $[-1, 1]$ or $[-10000, -1]$. For the matrices with eigenvalues in $[-1, 10000]$, we impose the condition that there is at least one negative eigenvalue.

5.2.1 The GMW-I Algorithm

The GMW81 algorithm, a Type-I algorithm, satisfies $\|E\|_2 = O(n^2)$, whereas the SE90 and SE99 algorithms further guarantee $\|E\|_2 = O(n)$, as shown in

(5.12) and (5.17), respectively. Schnabel and Eskow [54] pointed out that the 2-phase strategy can drop the bound on $\|E\|_2$ of the GMW81 algorithm to be $O(n)$. In our experiments, we note that incorporating the 2-phase strategy into the GMW81 algorithm introduces difficulties similar to those for SE90, and again relaxing provides the rescue.

We denote by GMW-I the algorithm that uses Relaxed Phase 1, with the GMW81 algorithm for Phase 2. Denote the number of steps in Phase 1 by K. Then $\delta_k = 0$ for $k = 1, 2, \ldots, K$. Instead of (5.3), the bound on $\|E\|_2$ is

$$\|E\|_2 \leq (\frac{\hat{\xi}}{\beta} + (n-K-1)\beta)^2 + 2(\hat{\eta} + (n-K-1)\beta^2) + \delta, \tag{5.21}$$

where $\hat{\eta}$ and $\hat{\xi}$ are the maximum magnitudes of the diagonal and off-diagonal elements of A_{K+1}, respectively. Now we do not need $\beta^2 \geq \hat{\eta}$ to satisfy Objective 1, so β is chosen as the minimizer of (5.21),

$$\beta^2 = \max\{\frac{\hat{\xi}}{\sqrt{(n-K)^2 - 1}}, \epsilon_M\}$$

for $n - K > 1$. Substituting it into (5.21) and invoking (5.18), we obtain

$$\|E\|_2 \leq 4(n-K)\hat{\xi} + 2\hat{\eta} + \delta \leq 4(n-K)(\xi + (1+\mu)\eta) + 2\eta + \delta = O(n), \tag{5.22}$$

where we ignore the extreme case $\beta^2 = \epsilon_M$.

When the 2-phase strategy or relaxed 2-phase strategy is incorporated, we still use $\delta := \epsilon_M$ as used in the GMW81 algorithm. We use $\mu = 0.75$ in the relaxed 2-phase strategy since it is an empirically good value for the GMW algorithms. Recall that $\mu = 0.1$ for the SE99 algorithm. Pivoting successfully reduces $\|E\|_2$ in the original GMW81 algorithm. When the 2-phase or relaxed 2-phase strategy is incorporated, we pivot on the maximum element instead of the maximum magnitude element in Phase 2, because on average the resulting $\kappa_2(A + E)$ is smaller in our experiments. We call our variant the GMW-I algorithm.

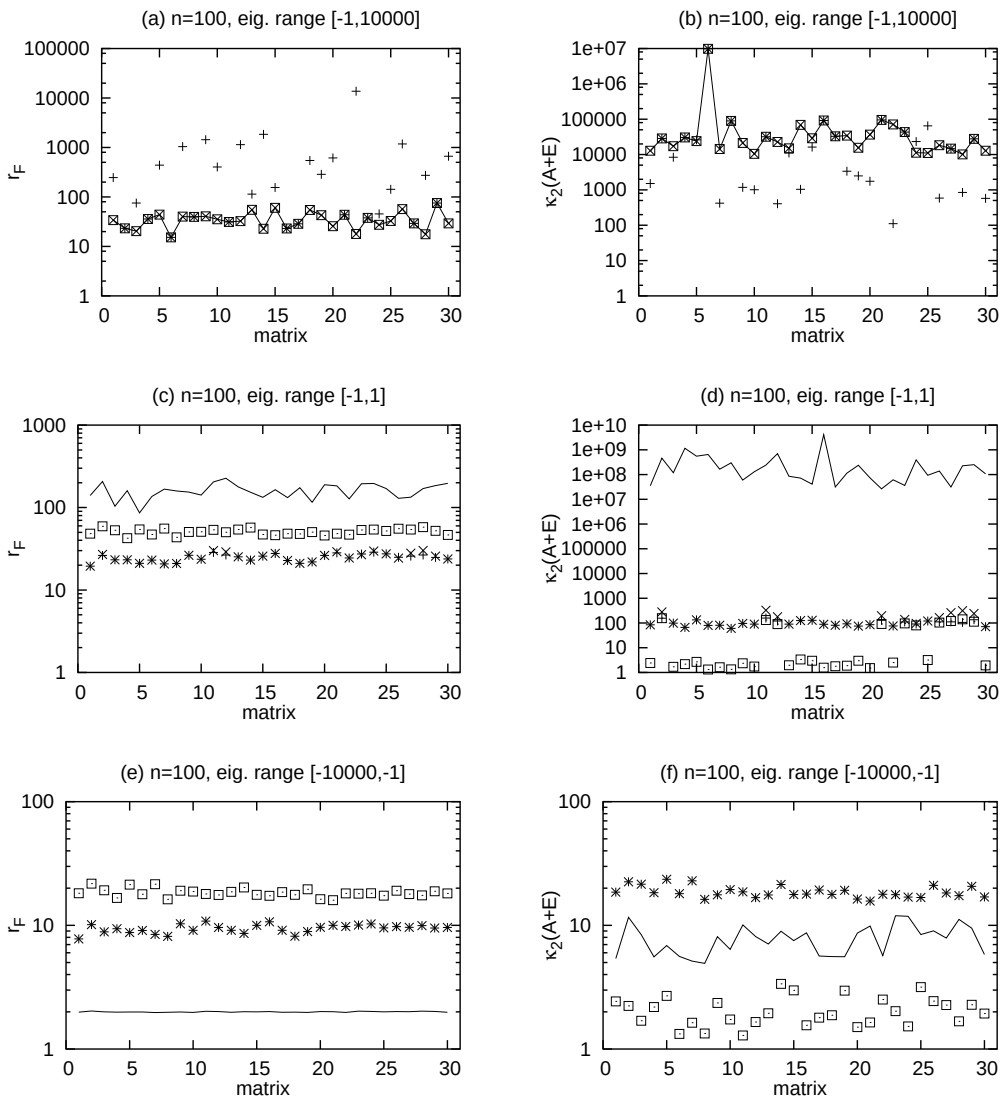

Figure 5.1: Measures of r_F and $\kappa_2(A+E)$ for the Type-I GMW algorithms for 30 random matrices with $n = 100$. Key: original GMW81 —, with 2-phase strategy $+$, with relaxed 2-phase strategy (GMW-I) $\times$, with relaxed 2-phase and nondecreasing strategy $\square$.

Figure 5.1 shows our experimental result. The GMW-I algorithm performed well, but for the random matrices with eigenvalues in $[-10000, -1]$, the $\|E\|_2$ was a few times larger than in the original GMW81 algorithm. Nevertheless, in practical optimization problems, negative definite Hessian matrices rarely occur, and indefinite Hessian matrices are usually close to being positive definite. The nondecreasing strategy was also tried. For the random matrices with eigenvalues in $[-1, 1]$ and $[-10000, -1]$, the nondecreasing strategy substantially reduced $\kappa_2(A + E)$ but roughly doubled $\|E\|_F$ (though with $\|E\|_2$ comparable). Note that the bound on $\|E\|_2$ in (5.3) is preserved with the nondecreasing strategy.

5.2.2 The GMW-II Algorithm

In this subsection we introduce our GMW-II algorithm, a Type-II variant of the GMW81 algorithm. Following the general model (5.1), we determine δ_k by

$$a_k + \delta_k := \max\{\delta, a_k + \delta_{k-1}, \frac{\|c_k\|_\infty^2}{\beta^2}\}$$

for $k = 1, \ldots, n$, where $\beta > 0$ and small tolerance $\delta > 0$ are preset, and $\delta_0 := 0$.

Three remarks are in order. First, β is still the upper bound on the magnitude of the off-diagonal elements in $\bar{L}$, where $LDL^T = \bar{L}\bar{L}^T$. Second, the original GMW81 algorithm is a Type-I method, whereas our variant is of Type II. Third, the nondecreasing strategy is applied.

The bound on $\|E\|_2$ for the GMW81 algorithm is given in (5.3). For the Type-II GMW algorithm, it is

$$\|E\|_2 \leq (\frac{\xi}{\beta} + (n-1)\beta)^2 + (\eta + (n-1)\beta^2) + \delta =: f(\beta). \tag{5.23}$$

The equality is attained with $\beta^2 = \frac{\xi}{\sqrt{n^2-n}}$ for $n > 1$. Recall that η and ξ are the maximum magnitudes of the diagonal and off-diagonal elements of A, respectively.

The minimum of (5.23) is

$$\min_{\beta} f(\beta) = 2\xi(\sqrt{n^2-n}+n-1)+\eta+\delta \leq 4n\xi+\eta+\delta.$$

The minimum is attained with $\beta^2 = \frac{\xi}{\sqrt{n^2-n}}$ for $n > 1$. Therefore, β is chosen by

$$\beta^2 := \max\{\eta, \frac{\xi}{\sqrt{n^2-n}}, \epsilon_M\}$$

for $n > 1$, where $\beta^2 \geq \eta$ is for satisfying Objective 1 with pivoting. Substituting it into (5.23), we obtain $\|E\|_2 = O(n^2)$.

The relaxed 2-phase strategy in Algorithm 5.2 is also incorporated into our GMW-II algorithm. Therefore, the bound on $\|E\|_2$ is

$$\|E\|_2 \leq (\frac{\hat{\xi}}{\beta}+(n-K-1)\beta)^2+(\hat{\eta}+(n-K-1)\beta^2)+\delta, \tag{5.24}$$

where K is the number of steps in Phase 1, and $\hat{\eta}$ and $\hat{\xi}$ are the maximum magnitudes of the diagonal and off-diagonal elements of A_{K+1}, respectively. Since $\beta^2 \geq \hat{\eta}$ is not required for satisfying Objective 1, β is determined by

$$\beta^2 := \max\left\{\frac{\hat{\xi}}{\sqrt{(n-K)^2-(n-K)}}, \epsilon_M\right\}$$

for $n-K > 1$. Substituting it into (5.24), we obtain

$$\|E\|_2 \leq 4(n-K)\hat{\xi}+\hat{\eta}+\delta \leq 4(n-K)(\xi+(1+\mu)\eta)+\eta+\delta = O(n),$$

where we ignore the extreme case $\beta^2 = \epsilon_M$. The last inequality is derived using (5.18).

The diagonal pivoting strategy can be incorporated into the Type-II GMW algorithms. We pivot on the maximum element for our GMW-II algorithm, as in the GMW-I algorithm. Note that all the *a priori* bounds on $\|E\|_2$ given above for all algorithms in the GMW class are independent of the pivoting strategy applied, if any.

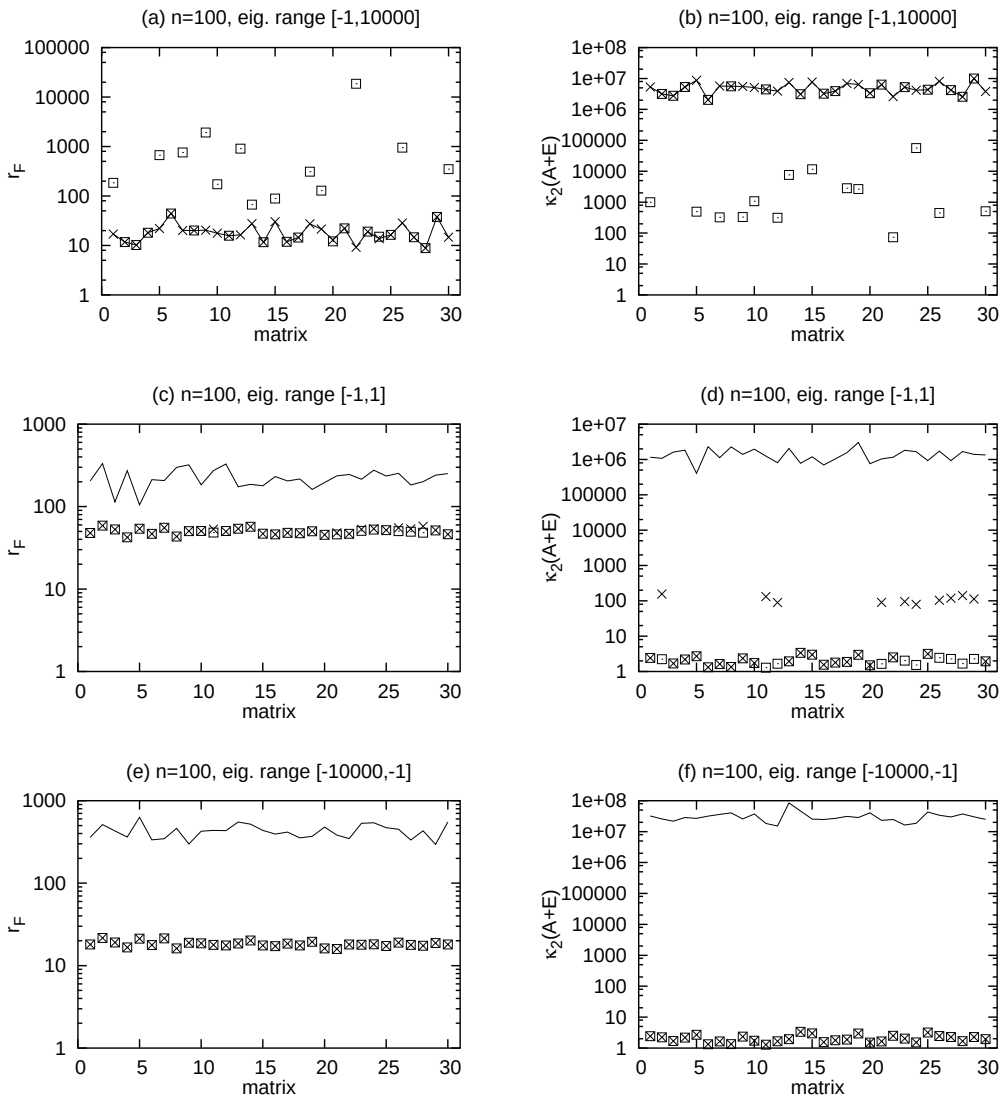

Figure 5.2: Measures of r_F and $\kappa_2(A+E)$ for the Type-II GMW algorithms for 30 random matrices with $n = 100$, nondecreasing strategy invoked. Key: original Type-II GMW —, with 2-phase strategy $\boxdot$, with relaxed 2-phase strategy (GMW-II) $\times$.

Recall that the GMW81 and GMW-I algorithms use $\delta := \epsilon_M$. For the Type-II GMW algorithms, we use $\delta := \sqrt[3]{\epsilon_M^2}\eta$ as in the SE99 algorithm. Our experimental results are shown in Figure 5.2. Similar to the SE90 algorithm and the Type-I GMW algorithms, incorporating the 2-phase strategy results in difficulties for the matrices with eigenvalues $[-1, 10000]$, and relaxing is the cure.

For all algorithms in the GMW class,

$$\text{the worst-case condition number } \kappa_2(A+E) = O(n^3(\frac{\xi+\eta}{\delta})^n). \tag{5.25}$$

The proof invokes Theorem 5.2 and Lemma 5.3 in Section 5.2.3 and uses the properties that the diagonal elements in D are bounded between δ and $\eta + (n-1)\beta^2$; the magnitude of the off-diagonal elements in $\bar{L}$ are bounded by β, where $P(A+E)P^T = LDL^T = \bar{L}\bar{L}^T$ as denoted in (5.2). Whether the 2-phase strategy or the relaxed 2-phase strategy is applied, the bound on $\kappa_2(A+E)$ remains exponential using (5.7) and (5.18), respectively. The bounds are not changed when the nondecreasing strategy is applied. All the modified Cholesky algorithms in Sections 5.1 and 5.2 are numerically stable, since they can be regarded as the Cholesky factorizations of the symmetric positive definite matrix $A+E$ [16].

5.2.3 The SE-I Algorithm

Both SE90 and SE99 algorithms are Type-II algorithms. In this subsection we present the Type-I variant corresponding to the SE99 algorithm, denoted by the SE-I algorithm, after making three changes. First, instead of (5.14), we determine δ_k by

$$\delta_k := \max\{0, -2a_k, -a_k + \max\{\|a_k\|_1, \bar{\tau}\eta\}\} \leq \max\{2G, G + \bar{\tau}\eta\} \tag{5.26}$$

for $k = K+1, \ldots, n-2$. Second, instead of (5.16), the special treatment of the last 2×2 Schur complement in Phase 2 to keep $\|E\|_2$ small is

$$\begin{aligned}\delta_{n-1} = \delta_n &:= \max\{0, -2\lambda_1(A_{n-1}), -\lambda_1(A_{n-1}) + \max\{\frac{\tau(\lambda_2(A_{n-1}) - \lambda_1(A_{n-1}))}{1-\tau}, \bar{\tau}\eta\}\} \\ &\leq \max\{2G, G + \frac{2\tau}{1-\tau}(G+\eta)\}. \end{aligned} \tag{5.27}$$

Note that $\kappa_2(A_{n-1} + \delta_n I_2) \leq \min\{\kappa_2(A_{n-1}), \frac{1}{\tau}\}$. The derivation is similar to that of (5.10). Third, if the algorithm switches to Phase 2 at the last step, then δ_n is determined by

$$\begin{aligned}\delta_n &= \max\{0, -2a_n, -a_n + \max\{\frac{-\tau}{1-\tau}a_n, \bar{\tau}\eta\}\} \\ &\leq \max\{2G, G + \frac{\tau}{1-\tau}(G+\eta)\} \end{aligned} \tag{5.28}$$

instead of (5.15).

By (5.26), (5.27) and (5.28), we obtain

$$\|E\|_2 \leq \max\{2G, G + \frac{2\tau}{1-\tau}(G+\eta)\}. \tag{5.29}$$

Comparing (5.29) with (5.17), the bound on $\|E\|_2$ for the SE-I algorithm is less than twice as that for the SE99 algorithm.

Now we formally investigate the satisfaction of Objective 1 for GMW and SE algorithms. We begin with a theorem of Ostrowski [42, page 224].

Theorem 5.2 (Ostrowski) *Suppose we are given a symmetric $M \in C^{n\times n}$ and a nonsingular $S \in C^{n\times n}$. There exists $\theta_k > 0$ such that*

$$\lambda_k(SMS^*) = \theta_k \lambda_k(M),$$

where $\lambda_1(SS^*) \leq \theta_k \leq \lambda_n(SS^*)$.

Consider the 2-phase strategy presented in Algorithm 5.1 and the relaxed 2-phase strategy presented in Algorithm 5.2 with pivoting on the maximum diagonal element. Clearly $E = 0$ if the factorization is done in Phase 1. The derivation of the condition under which the algorithm runs to completion without switching to Phase 2 is by finite induction. We denote the incomplete LDL^T factorization of a symmetric matrix $A \in R^{n\times n}$ after step k by $L_k D_k L_k^T$, where

$$D_k = \begin{array}{c} \\ k \\ n-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-k \end{array} \\ \begin{bmatrix} \bar{D}_k & 0 \\ 0 & S_k \end{bmatrix} \end{array},$$

with $\bar{D}_k$ diagonal and S_k the Schur complement. We claim that the following condition guarantees $E = 0$:

$$\lambda_{\min}(A) \geq \delta \|L_k L_k^T\|_2 \tag{5.30}$$

for $k = 1, \ldots, n-1$. At the beginning of step k, we assume all diagonal elements of the Schur complement are all larger than or equal to δ, and investigate whether this condition holds in the next Schur complement[5]. By Theorem 5.2 and (5.30),

$$\lambda_{\min}(D_k)\lambda_{\max}(L_k L_k^T) \geq \lambda_{\min}(A) \geq \delta \|L_k L_k^T\|_2 = \delta \lambda_{\max}(L_k L_k^T),$$

and therefore $\lambda_{\min}(D_k) \geq \delta$, so

$$\lambda_{\min}(S_k) \geq \lambda_{\min}(D_k) \geq \delta,$$

which implies $\mathrm{Diag}(S_k) \geq \delta I_{n-k}$. By induction, we stay in Phase 1 during the whole factorization. We conclude that if (5.30) holds, then $E = 0$.

[5]For the base case, we have $\lambda_{\min}(A) \geq \delta$ from (5.30), so $A - \delta I$ is positive definite and therefore $\mathrm{diag}(A) \geq \delta I$.

Lemma 5.3, proved using Lemma 5.2 as a tool, is developed to bound $\|LL^T\|_2$, where L is lower triangular. A bound on $\lambda_{\min}(LL^T)$ is also developed to bound the condition number of $A + E$ for algorithms in Sections 5.3 and 5.4.

Lemma 5.2 *If the positive semidefinite Hermitian matrix $M \in C^{n\times n}$ has a diagonal element equal to 1, (i.e., $m_{kk} = 1$ for some $1 \le k \le n$), then*

$$\lambda_{\min}(M) \le 1 \le \lambda_{\max}(M).$$

Proof Let $M = U\Lambda U^*$ denote the spectral decomposition of M, and $a := U^* e_k$. Since $m_{kk} = 1$,

$$1 = e_k^T M e_k = a^* U^* (U\Lambda U^*) U a = a^* \Lambda a.$$

Note that $a^* a = 1$. We conclude that the weighted average of the eigenvalues of M is 1. Therefore, $\lambda_{\min}(M) \le 1 \le \lambda_{\max}(M)$. □

Lemma 5.3 *For any lower unit triangular matrix $L \in R^{n\times n}$ with $|(L)_{ij}| \le \gamma$ for $1 \le j < i \le n$,*

$$1 \le \lambda_{\max}(LL^T) \le n + \frac{1}{2} n(n-1)\gamma^2,$$

and

$$(1+\gamma)^{2-2n} \le \lambda_{\min}(LL^T) \le 1.$$

Proof By Lemma 5.2, $\lambda_{\min}(LL^T) \le 1 \le \lambda_{\max}(LL^T)$. An upper bound on $\lambda_{\max}(LL^T)$ is $\lambda_{\max}(LL^T) \le \text{trace}(LL^T) \le n + \frac{1}{2}n(n-1)\gamma^2$. Computing the inverse of a lower triangular matrix, we obtain $(L^{-1})_{ii} = 1$ for $i = 1, \ldots, n$ and the bounds $|(L^{-1})_{ij}| \le \gamma \sum_{k=j+1}^{i} |(L^{-1})_{ik}|$ for $1 \le j < i \le n$. The solution to this recursion is

$$|(L^{-1})_{ij}| \le \gamma(1+\gamma)^{i-j-1}$$

for $1 \leq j < i \leq n$. Therefore,

$$\lambda_{\min}(LL^T)^{-1} = \|(LL^T)^{-1}\|_2 \leq \|L^{-1}\|_2^2 \leq \|L^{-1}\|_1 \|L^{-1}\|_\infty \leq (1+\gamma)^{2n-2}.$$

Cheng and Higham [16] presented this lemma with $\gamma = \frac{7+\sqrt{17}}{4} \approx 2.781$. $\square$

Now we can bound $\|L_k L_k^T\|_2$ in (5.30). Pivoting on the maximum diagonal element of each Schur complement, the magnitude of the elements in L_k are bounded by 1 for all k. By Lemma 5.3,

$$\|L_k L_k^T\|_2 \leq \frac{1}{2} n(n+1). \tag{5.31}$$

Substituting it into (5.30), we obtain the following result. For all algorithms using the 2-phase strategy or the relaxed 2-phase strategy GMW-I, GMW-II, SE90, SE99, and SE-I, if

$$\lambda_{\min}(A) \geq \frac{1}{2} n(n+1)\delta, \tag{5.32}$$

then by (5.30) and (5.31) we conclude that $E = 0$.

Our experimental results are shown in Figure 5.3. For the random matrices with eigenvalues in $[-1, 10000]$, the SE-I algorithm resulted in larger $\|E\|_2$ and $\|E\|_F$ but substantially smaller $\kappa_2(A+E)$ than those of the SE99 algorithm. For the random matrices with eigenvalues in $[-1, 1]$ and $[-10000, -1]$, the SE-I algorithm had comparable $\|E\|_2$, smaller $\|E\|_F$ but larger $\kappa_2(A+E)$ than the SE99 algorithm.

The nondecreasing strategy can be incorporated into the Type-I SE algorithm. The resulting $\|E\|_2$, $\|E\|_F$ and $\kappa_2(A+E)$ were comparable to those of the SE-I algorithm for the random matrices with eigenvalues in $[-1, 10000]$, and comparable to those of the SE99 algorithm for the random matrices with eigenvalues

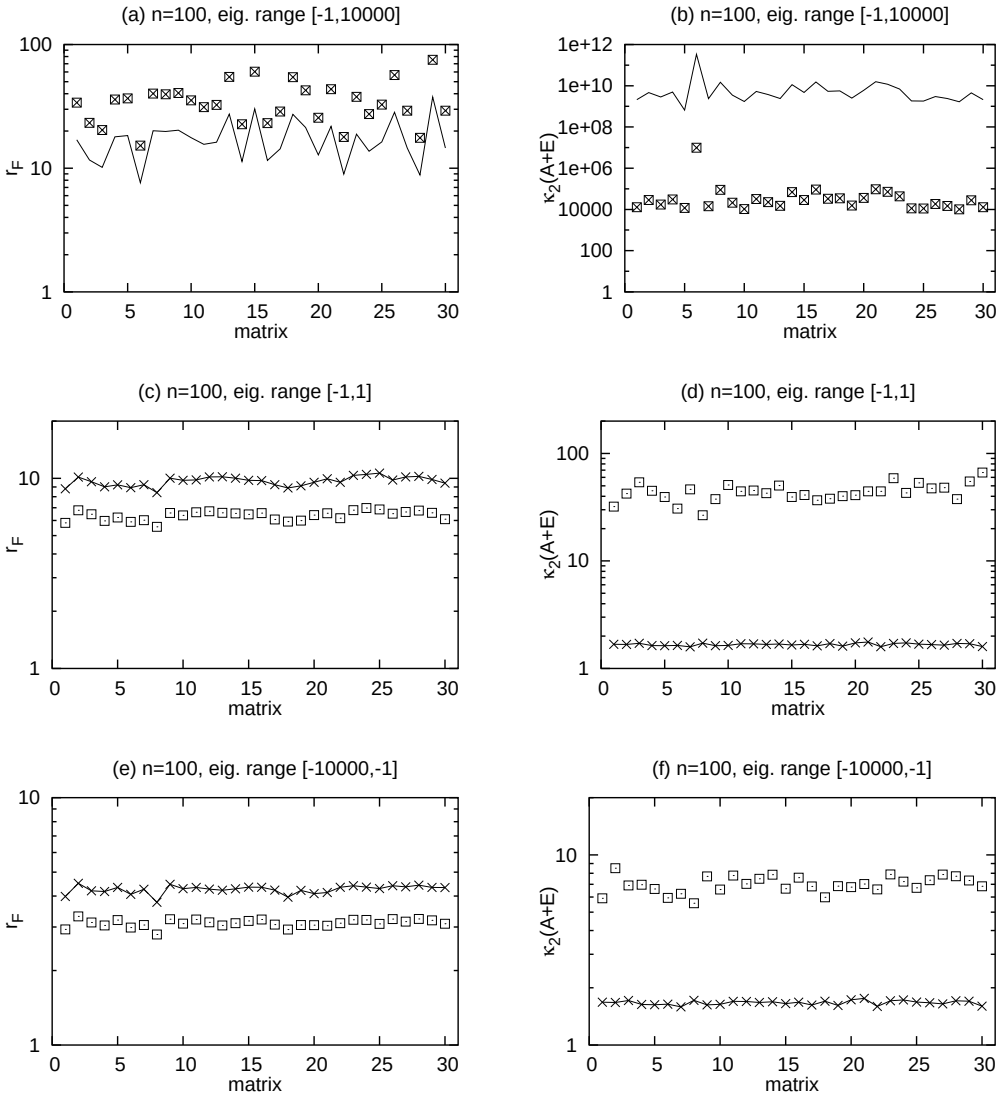

Figure 5.3: Measures of r_F and $\kappa_2(A+E)$ for the SE algorithms for 30 random matrices with $n = 100$. Key: original SE99 —, Type-I SE99 (SE-I) $\square$, Type-I SE99 with nondecreasing strategy $\times$.

in $[-1, 1]$ and $[-10000, -1]$. Incorporating the non-relaxed 2-phase strategy into the Type-I SE algorithms is possible, but it would result in difficulties similar to those of the SE90 algorithm.

For all the algorithms in the SE class,

$$\text{the worst-case condition number } \kappa_2(A+E) = O(\frac{(\xi+\eta)n^3 4^n}{\delta}). \tag{5.33}$$

The sketch of the proof is similar to that for the GMW algorithms. In practice, the condition number is bounded by about $1/\tau$ and $1/\bar{\tau}$ respectively for the SE90 and SE99 algorithms [54], and is comparable to $\kappa_2(A)$ for the SE-I algorithm.

5.3 Modified LBL^T Algorithms

Any symmetric matrix $A \in R^{n\times n}$ has an LBL^T factorization, where B is block diagonal with block order 1 or 2 [5, 12, 14]. A modified LBL^T algorithm first computes the LBL^T factorization, and then perturbs $\hat{B} = B + \Delta B$ to be positive definite, so that $P(A+E)P^T = L\hat{B}L^T$ is positive definite as well, where P is the permutation matrix for pivoting.

Moré and Sorensen suggested a modified LBL^T algorithm in 1979 [45]. Each 1×1 block in B, denoted by d, is modified to be $\hat{d} := \max\{\delta, |d|\}$, with $\delta > 0$ the preset small tolerance. For each 2×2 block D, its spectral decomposition $D = U \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} U^T$ is modified to be $\hat{D} := U \begin{bmatrix} \hat{\lambda}_1 & \\ & \hat{\lambda}_2 \end{bmatrix} U^T$, where $\hat{\lambda}_i := \max\{\delta, |\lambda_i|\}$ for $i = 1, 2$. We call this the MS79 algorithm.

Cheng and Higham proposed another modified LBL^T algorithm in 1998 [16]. Each 1×1 block d is modified to be $\hat{d} := \max\{\delta, d\}$, with $\delta > 0$ the preset small tolerance. Each 2×2 block D, with its spectral decomposition denoted by $D =$

$U \begin{bmatrix} \lambda_1 & \\ & \lambda_2 \end{bmatrix} U^T$, is modified to be $\hat{D} := U \begin{bmatrix} \hat{\lambda}_1 & \\ & \hat{\lambda}_2 \end{bmatrix} U^T$, where $\hat{\lambda}_i = \max\{\delta, \lambda_i\}$ for $i = 1, 2$. We refer to the algorithm as the CH98 algorithm hereafter.

The key distinction is that MS79 is a Type-I algorithm, whereas the CH98 algorithm is of Type II. The MS79 algorithm was given in 1979 [45], before the fast Bunch-Parlett and bounded Bunch-Kaufman pivoting strategies (rook pivoting) for the LBL^T factorization were introduced [5], but rook pivoting is also applicable to the MS79 algorithm. For the MS79 algorithm, we set $\delta := \epsilon_M$. Cheng and Higham [16] suggested $\delta := \sqrt{u}\|A\|_\infty$ for CH98 algorithm, where $u = \epsilon_M/2$ is the unit roundoff.

The MS79 algorithm predated the four objectives, first presented in 1990 [54]. The four objectives were well-investigated by Cheng and Higham for the CH98 algorithm [16], and our analysis of MS79 is similar.

For both the MS79 and CH98 algorithms, if $\lambda_{\min}(B) \geq \delta$, then $E = 0$. By Theorem 5.2, if A is positive definite, $\lambda_{\min}(B) \geq \frac{\lambda_{\min}(A)}{\lambda_{\max}(LL^T)}$. Therefore, $E = 0$ is guaranteed when

$$\lambda_{\min}(A) \geq \delta\|LL^T\|_2. \tag{5.34}$$

Consider $\|E\|_2$ for the MS79 algorithm. By Theorem 5.2,

$$\begin{aligned} \|E\|_2 &= \lambda_{\max}(E) = \lambda_{\max}(L\Delta BL^T) \leq \lambda_{\max}(LL^T)\lambda_{\max}(\Delta B) \\ &= \lambda_{\max}(LL^T)\max\{\delta - \lambda_{\min}(B), -2\lambda_{\min}(B), 0\}. \end{aligned}$$

By Theorem 5.2 again, $-\lambda_{\min}(B) \leq -\frac{\lambda_{\min}(A)}{\lambda_{\min}(LL^T)}$ and $-\lambda_{\min}(B) \geq -\frac{\lambda_{\min}(A)}{\lambda_{\max}(LL^T)}$ for $\lambda_{\min}(A) < 0$. Therefore,

$$\|E\|_2 \leq -2\lambda_{\min}(A)\kappa_2(LL^T) \text{ for } \lambda_{\min}(A) \leq -\delta\|LL^T\|_2. \tag{5.35}$$

Similarly, the bound on $\|E\|_2$ for the CH98 algorithm is

$$\|E\|_2 \leq \delta\|LL^T\|_2 - \lambda_{\min}(A)\kappa_2(LL^T) \text{ for } \lambda_{\min}(A) \leq 0. \tag{5.36}$$

Now we assess how well Objective 3 is satisfied for the MS79 algorithm. By Theorem 5.2,

$$\begin{aligned}
\lambda_{\min}(A+E) &\geq \lambda_{\min}(LL^T)\lambda_{\min}(\hat{B}) \\
&= \lambda_{\min}(LL^T)\max\{\delta, \min_{1\leq i\leq n}|\lambda_i(B)|\} &(5.37)\\
&\geq \lambda_{\min}(LL^T)\max\{\delta, \frac{\min_{1\leq i\leq n}|\lambda_i(A)|}{\lambda_{\max}(LL^T)}\}, &(5.38)
\end{aligned}$$

and

$$\begin{aligned}
\lambda_{\max}(A+E) &\leq \lambda_{\max}(LL^T)\lambda_{\max}(\hat{B}) \\
&= \lambda_{\max}(LL^T)\max\{\delta, -\lambda_{\min}(B), \lambda_{\max}(B)\} &(5.39)\\
&\leq \lambda_{\max}(LL^T)\max\{\delta, \frac{-\lambda_{\min}(A)}{\lambda_{\min}(LL^T)}, \frac{\lambda_{\max}(A)}{\lambda_{\min}(LL^T)}\}. &(5.40)
\end{aligned}$$

By (5.37) and (5.39),

$$\kappa_2(A+E) \leq \kappa_2(LL^T)\kappa_2(B).$$

By (5.38) and (5.40),

$$\kappa_2(A+E) \leq \kappa_2(LL^T)^2\kappa_2(A). \tag{5.41}$$

The bound on $\kappa_2(A+E)$ for the CH98 algorithm [16] is

$$\kappa_2(A+E) \leq \kappa_2(LL^T)\max\{1, \frac{\lambda_{\max}(A)}{\lambda_{\min}(LL^T)\delta}\}. \tag{5.42}$$

There are four pivoting algorithms for the LBL^T factorization: Bunch-Parlett (complete pivoting) [14], Bunch-Kaufman (partial pivoting) [12], fast Bunch-Parlett and bounded Bunch-Kaufman (rook pivoting) [5], denoted by **BP**, **BK**,

FBP and **BBK**, respectively. All these algorithms have a preset argument $0<\alpha<1$. The **BK** algorithm takes $O(n^2)$ time for pivoting, but the elements in L are unbounded. It is discouraged for the modified LBL^T algorithms because Objectives 1–3 may not be satisfied. For example, the following LBL^T factorization [39] is produced with the **BK** pivoting strategy for $\epsilon \neq 0$,

$$A = \begin{bmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ 0 & 1 & \\ 1/\epsilon & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \epsilon & \\ \epsilon & 0 & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1/\epsilon \\ & 1 & 0 \\ & & 1 \end{bmatrix} = LBL^T.$$

Applying MS79 or CH98 and assuming $0 < \epsilon \leq \delta$, we obtain

$$E = \begin{bmatrix} 1 & & \\ 0 & 1 & \\ 1/\epsilon & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta & -\epsilon & \\ -\epsilon & \delta & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1/\epsilon \\ & 1 & 0 \\ & & 1 \end{bmatrix} = \begin{bmatrix} \delta & -\epsilon & \delta/\epsilon \\ -\epsilon & \delta & -1 \\ \delta/\epsilon & -1 & 1+\delta \end{bmatrix}.$$

When $\epsilon \to 0^+$, $\|E\| \to \infty$, so Objective 2 is not satisfied.

From (5.34)–(5.42), it is clear that $\lambda_{\min}(LL^T)$, $\lambda_{\max}(LL^T)$ and $\kappa_2(LL^T)$ play an important role for the satisfaction of Objectives 1–3 for both MS79 and CH98. The **BP**, **BBK** and **FBP** algorithms all have a bound on the elements in L in terms of the pivoting argument α, suggested to be $\alpha = \frac{1+\sqrt{17}}{8} \approx 0.640$ to minimize the bound on the element growth of the Schur complements [5, 12, 14]. The corresponding element bound of the unit lower triangular matrix L is $\gamma = \frac{7+\sqrt{17}}{4} \approx 2.781$. Alternatively, we could choose $\alpha = 0.5$ to minimize the element bound of L, which is $\gamma = 2$ (see Table 2.1), leading to sharper bounds on $\lambda_{\min}(LL^T)$, $\lambda_{\max}(LL^T)$ and $\kappa_2(LL^T)$. The bounds in Table 5.3 are obtained using Lemma 5.3.

Although $\alpha = \frac{1}{2}$ results in smaller bounds, $\alpha = \frac{1+\sqrt{17}}{8} \approx 0.640$ is a better choice in practice, as shown in Figure 5.4.

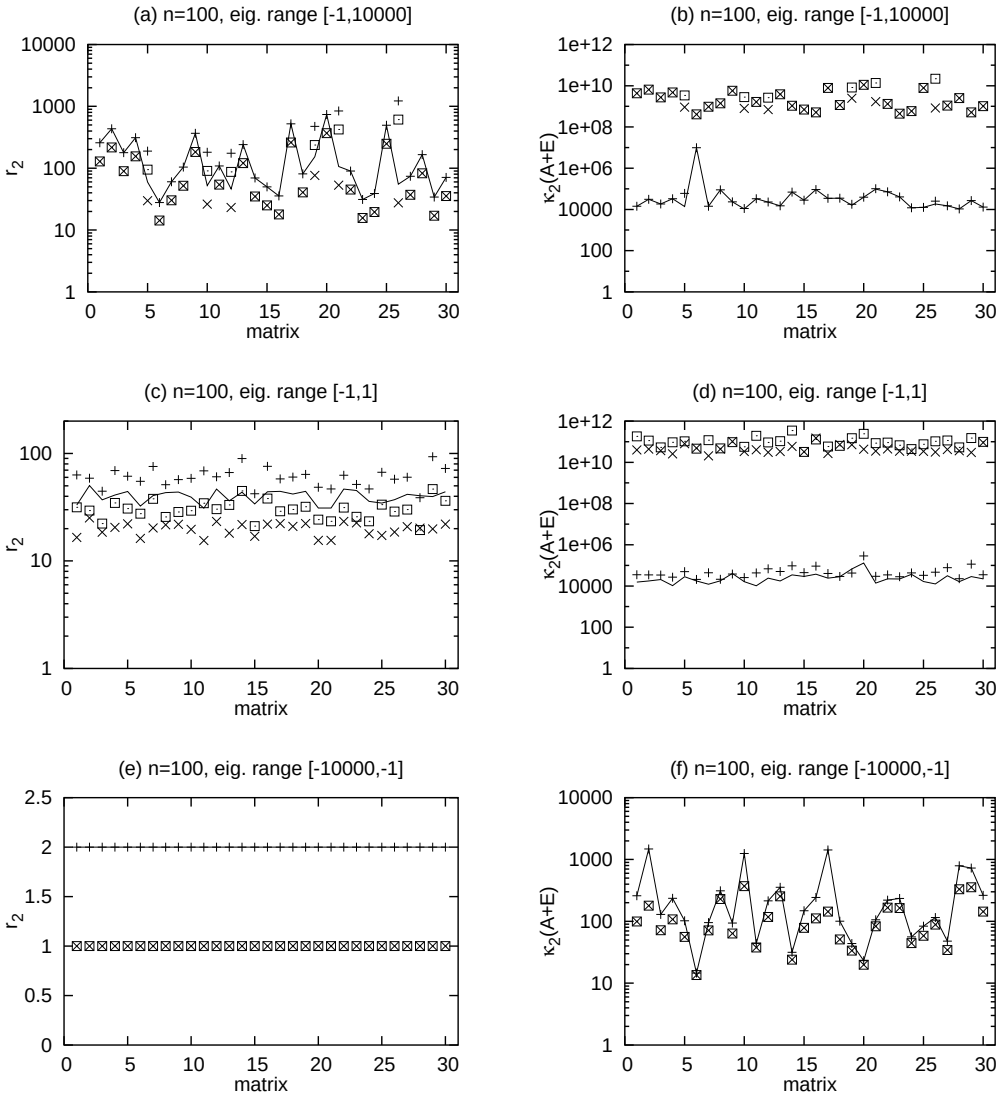

Figure 5.4: Measures of r_2 and $\kappa_2(A+E)$ for the MS79 and CH98 algorithms for 30 random matrices with $n = 100$. Key: MS79 $\alpha = 0.640$ —, MS79 $\alpha = 0.5$ $+$, CH98 $\alpha = 0.640$ $\times$, CH98 $\alpha = 0.5$ $\Box$.

Table 5.3: Bounds for the LBL^T factorization with the **BP**, **BBK** or **FBP** pivoting algorithm.

α	γ	$\lambda_{\min}(LL^T)$	$\lambda_{\max}(LL^T)$	$\kappa_2(LL^T)$
$\frac{1+\sqrt{17}}{8}$	$\frac{7+\sqrt{17}}{4}$	$\geq 3.781^{2-2n}$	$\leq 4n^2-3n$	$\leq (4n^2-3n)3.781^{2n-2}$
0.5	2	$\geq 3^{2-2n}$	$\leq 2n^2-n$	$\leq (2n^2-n)3^{2n-2}$

The **BP** pivoting strategy takes $\frac{1}{6}n^3 + O(n^2)$ comparisons and does not meet Objective 4. The number of comparisons for the **BBK** and **FBP** pivoting strategies are between those of the **BK** and **BP** algorithms (i.e., between $O(n^2)$ and $O(n^3)$). There are matrices that require traversing the whole matrix of each Schur complement with either the **BBK** or the **FBP** pivoting strategy [5]. Hence they take $\Theta(n^3)$ comparisons for pivoting in worst cases and fail to meet Objective 4.

Here and throughout the remainder of this chapter, we assume the pivoting strategy applied to the MS79 and the CH98 algorithms is **BBK**, unless otherwise noted. Three remarks are in order. First, both MS79 and CH98 satisfy Objectives 1–3. Second, the bound on $\|E\|_2$ for the MS79 algorithm is about twice that for the CH98 algorithm, whereas $A+E$ is generally better conditioned for MS79 than for CH98. Third, both algorithms fail to satisfy Objective 4 in the worst case.

5.4 A New Approach via Modified LTL^T Factorization

Aasen [1], Parlett and Reid [50] introduced the LTL^T factorization and its application to solving symmetric linear systems. We denote the factorization by

$PAP^T = LTL^T$, where T is symmetric tridiagonal, and L is unit lower triangular with the magnitude of its elements bounded by 1 and the off-diagonal elements in the first column all zero.

The required computation of the LTL^T factorization in the formulation by Aasen [1] is about the same as that of the Cholesky factorization, whereas the formulation by Parlett and Reid [50] doubles the cost. In both formulations the required storage is the same as that required by the Cholesky factorization. For solving linear systems, its numerical stability is empirically comparable to that of the LBL^T factorization [5].

Our new approach arises from the fact that A is positive definite if and only if T is positive definite. A modified LTL^T algorithm makes $\hat{T} = T + \Delta T$ symmetric positive definite, and the resulting factorization $\hat{A} = P(A + E)P^T = L\hat{T}L^T$ is also symmetric positive definite.

We can apply the modified LDL^T algorithms in Sections 5.1 and 5.2 and the modified LBL^T algorithms in Section 5.3 to the matrix T. The resulting modified LTL^T factorization roughly satisfies Objective 1, assuming the modified Cholesky algorithm applied to T satisfies Objective 1. Our method was inspired by the merits of the triadic structure discussed in Chapter 2.

By Theorem 2.3, the triadic structure is preserved in the LDL^T or LBL^T factorizations. It implies that the modified LDL^T or LBL^T algorithms in Sections 5.1–5.3 applied to a symmetric triadic matrix are very efficient. Recall that both MS79 and CH98 algorithms have difficulties in satisfying Objective 4. The potential excessive cost can be reduced to be $O(n^2)$ by instead applying the MS79 or CH98 algorithm to the symmetric tridiagonal matrix T of the LTL^T factorization. We call the resulting algorithms LTL^T-MS79 and LTL^T-CH98, re-

spectively. For the LTL^T-MS79 algorithm, we use $\delta := \epsilon_M$. For the LTL^T-CH98 algorithm, we use $\delta := \sqrt[3]{\epsilon_M^2}\eta$, as used in the SE99 algorithm.

Table 5.4 compares the costs of these LBL^T pivoting strategies for symmetric and symmetric tridiagonal matrices. We use the **BBK** pivoting strategy for both MS79 and CH98, because it is the cheapest pivoting strategy that guarantees a bounded L. Even so, Objective 4 is not satisfied in worst cases. We use the **BP** pivoting strategy for both LTL^T-CH98 and LTL^T-MS79 algorithms. By Theorem 2.3, Objective 4 is satisfied, even though **BP** is the most expensive pivoting strategy.

Table 5.4: Comparison costs of various pivoting strategies for the LBL^T factorization.

Symmetric Matrix	*General*		*Tridiagonal*	
case	*worst*	*best*	*worst*	*best*
BP	$O(n^3)$		$O(n^2)$	
FBP	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n)$
BBK	$O(n^3)$	$O(n^2)$	$O(n^2)$	$O(n)$

Given a symmetric matrix $A \in R^{n\times n}$, we denote its LTL^T factorization by $PAP^T = LTL^T$, and the LBL^T factorization of T by $\widetilde{P}T\widetilde{P}^T = \widetilde{L}\widetilde{B}\widetilde{L}^T$. The resulting sandwiched factorization is $PAP^T = L\widetilde{P}^T\widetilde{L}\widetilde{B}\widetilde{L}^T\widetilde{P}L^T$. Adding a perturbation $\Delta\widetilde{B}$ to $\widetilde{B}$ to make it positive definite, the modified factorization of T is $\widetilde{P}(T+\Delta T)\widetilde{P}^T = \widetilde{L}(\widetilde{B}+\Delta\widetilde{B})\widetilde{L}^T$. The modified LTL^T factorization is

$$P(A+E)P^T = L\widetilde{P}^T\widetilde{L}(\widetilde{B}+\Delta\widetilde{B})\widetilde{L}^T\widetilde{P}L^T. \tag{5.43}$$

The matrix L is unit lower triangular with the magnitude of all elements bounded by 1 and all the off-diagonal elements in the first column zero. By Lemma 5.3,

the LTL^T factorization satisfies

$$\begin{aligned} \lambda_{\max}(LL^T) &\leq \tfrac{1}{2}n(n-1) \\ \lambda_{\min}(LL^T) &\geq 2^{4-2n} \end{aligned} \tag{5.44}$$

for $n > 1$. Lemma 5.4 gives the bounds on $\lambda_{\max}(\widetilde{L}\widetilde{L}^T)$ and $\lambda_{\min}(\widetilde{L}\widetilde{L}^T)$, where $\widetilde{L}$ is triadic and unit lower triangular.

Lemma 5.4 *Let* $F_\gamma(k) = \sum_{i=1}^{\lceil k/2 \rceil} \binom{k-i}{i-1} \gamma^{k-i}$ *and* $\Phi_\gamma = \frac{1+\sqrt{1+4/\gamma}}{2}\gamma$ *for* $k \in N$ *and* $\gamma > 0$. *For any triadic and unit lower triangular* $\widetilde{L} \in R^{n\times n}$ *with the magnitude of the off-diagonal elements bounded by* γ,

1. $\lambda_{\max}(\widetilde{L}\widetilde{L}^T) \leq n + (2n-3)\gamma^2$ *for* $n > 1$.
2. $\lambda_{\min}(\widetilde{L}\widetilde{L}^T) \geq (\frac{\Phi_\gamma - 1}{\Phi_\gamma^n - 1})^2$.

Proof First, for $n > 1$,

$$\lambda_{\max}(\widetilde{L}\widetilde{L}^T) \leq \mathrm{trace}(\widetilde{L}\widetilde{L}^T) = \|\widetilde{L}\|_F^2 \leq n + (2n-3)\gamma^2.$$

By Lemma 2.4 and (2.18),

$$\begin{aligned} \lambda_{\min}(\widetilde{L}\widetilde{L}^T)^{-1} &= \|(\widetilde{L}\widetilde{L}^T)^{-1}\|_2 = \|\widetilde{L}^{-1}\|_2^2 \leq \|\widetilde{L}^{-1}\|_1 \|\widetilde{L}^{-1}\|_\infty \\ &\leq (\sum_{k=1}^{n} F_\gamma(k))^2 \leq (\sum_{k=1}^{n} \Phi_\gamma^{k-1})^2 = (\frac{\Phi_\gamma^n - 1}{\Phi_\gamma - 1})^2. \ \square \end{aligned}$$

Now we can assess the satisfaction of Objectives 1–3 for our LTL^T-MS79 and LTL^T-CH98 algorithms. To ensure a bounded L of the LBL^T factorization, we can use **BP**, **FBP** or **BBK**, but not **BK**. By (5.34), $\lambda_{\min}(T) \geq \delta\|\widetilde{L}\widetilde{L}^T\|_2$ implies $E = 0$. By Theorem 5.2, if A is positive definite, $\lambda_{\min}(A) \geq \lambda_{\min}(T)\lambda_{\min}(LL^T)$. We conclude that $E = 0$ if

$$\lambda_{\min}(A) \geq \delta\|\widetilde{L}\widetilde{L}^T\|_2 \lambda_{\min}(LL^T). \tag{5.45}$$

For the LTL^T-MS79 algorithm, by Theorem 5.2 and (5.35),

$$\begin{aligned} \|E\|_2 &= \lambda_{\max}(E) = \lambda_{\max}(L\Delta T L^T) \\ &\leq \lambda_{\max}(LL^T)\lambda_{\max}(\Delta T) = \|LL^T\|_2\|\Delta T\|_2 \\ &\leq -2\lambda_{\min}(A)\kappa_2(LL^T)\kappa_2(\widetilde{L}\widetilde{L}^T) \end{aligned} \tag{5.46}$$

for $\lambda_{\min}(A) \leq -\delta\|LL^T\|_2\|\widetilde{L}\widetilde{L}^T\|_2$. For the LTL^T-CH98 algorithm, by Theorem 5.2 and (5.36),

$$\|E\|_2 \leq \delta\|LL^T\|_2\|\widetilde{L}\widetilde{L}^T\|_2 - \lambda_{\min}(A)\kappa_2(LL^T)\kappa_2(\widetilde{L}\widetilde{L}^T) \tag{5.47}$$

for $\lambda_{\min}(A) \leq 0$. For the LTL^T-MS79 algorithm, by Theorem 5.2 and (5.41),

$$\begin{aligned} \kappa_2(A+E) &\leq \kappa_2(LL^T)\kappa_2(T+\Delta T) \\ &\leq \kappa_2(LL^T)\kappa_2(\widetilde{L}\widetilde{L}^T)^2\kappa_2(T) \\ &\leq \kappa_2(LL^T)^2\kappa_2(\widetilde{L}\widetilde{L}^T)^2\kappa_2(A). \end{aligned} \tag{5.48}$$

For the LTL^T-CH98 algorithm, by Theorem 5.2 and (5.42),

$$\begin{aligned} \kappa_2(A+E) &\leq \kappa_2(LL^T)\kappa_2(\widetilde{L}\widetilde{L}^T)\max\{1, \frac{\lambda_{\max}(T)}{\lambda_{\min}(\widetilde{L}\widetilde{L}^T)\delta}\} \\ &\leq \kappa_2(LL^T)\kappa_2(\widetilde{L}\widetilde{L}^T)\max\{1, \frac{\lambda_{\max}(A)}{\lambda_{\min}(LL^T)\lambda_{\min}(\widetilde{L}\widetilde{L}^T)\delta}\}. \end{aligned} \tag{5.49}$$

Note that the pivoting argument used in $\widetilde{P}T\widetilde{P}^T = \widetilde{L}\widetilde{B}\widetilde{L}^T$ is $\alpha = \frac{\sqrt{5}-1}{2} \approx 0.618$ for symmetric triadic matrices (see Theorem 2.6). The corresponding element bound of L is $\gamma = \frac{\sqrt{5}+3}{2} \approx 2.618$. One may choose $\alpha = 0.5$ to obtain the minimum element bound of L, which is $\gamma = 2$ (see Table 2.1), but it could result in an excessive $\|E\|_2$ for random matrices with eigenvalues $[-1, 10000]$ as shown in Figure 5.5.

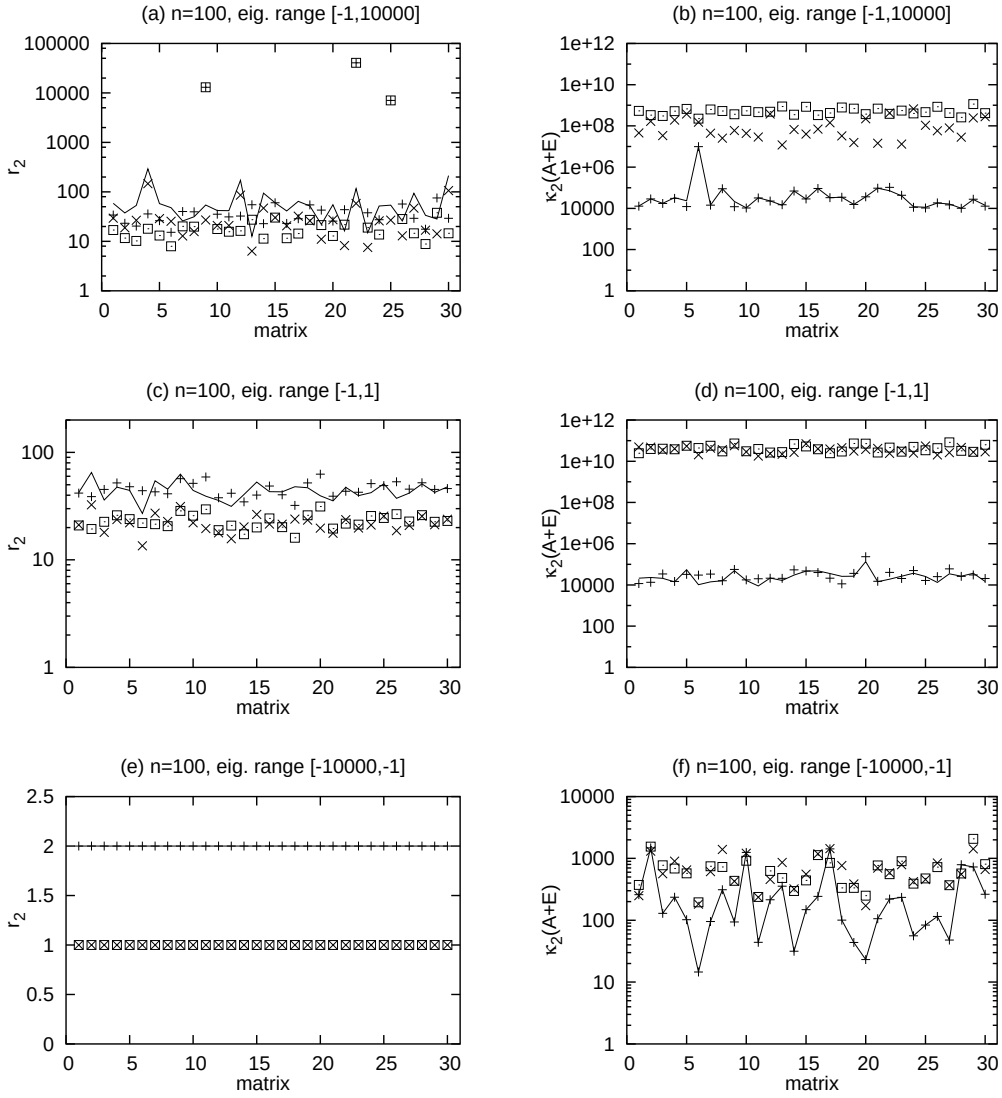

Figure 5.5: Measures of r_2 and $\kappa_2(A+E)$ for the LTL^T-MS79 and LTL^T-CH98 algorithms for 30 random matrices with $n = 100$. Key: LTL^T-MS79 $\alpha = 0.618$ —, LTL^T-MS79 $\alpha = 0.5$ $+$, LTL^T-CH98 $\alpha = 0.618$ $\times$, LTL^T-CH98 $\alpha = 0.5$ $\Box$.

The bounds on $\|LL^T\|_2$ and $\lambda_{\min}(LL^T)$ are given in (5.44). The bounds on $\|\widetilde{L}\widetilde{L}^T\|_2$ and $\lambda_{\min}(\widetilde{L}\widetilde{L}^T)$ are in Lemma 5.4 with $\gamma = \frac{\sqrt{5}+3}{2} \approx 2.618$. We conclude that

$$\begin{aligned} \|LL^T\|_2\|\widetilde{L}\widetilde{L}^T\|_2 &\leq 7.5n^3 - 17.5n^2 + 10.5n \\ \lambda_{\min}(LL^T)\lambda_{\min}(\widetilde{L}\widetilde{L}^T) &\geq \tfrac{91}{4^n(3.4^n-1)^2} \end{aligned} \tag{5.50}$$

for $n > 1$.

Comparing (5.50) with Table 5.3 with $\alpha = \frac{1+\sqrt{17}}{8}$, $\|LL^T\|_2$ and $\lambda_{\min}(LL^T)$ for the MS79 and CH98 algorithms have sharper bounds than $\|LL^T\|_2\|\widetilde{L}\widetilde{L}^T\|_2$ and $\lambda_{\min}(LL^T)\lambda_{\min}(\widetilde{L}\widetilde{L}^T)$ for the LTL^T-MS79 and LTL^T-CH98 algorithms, respectively. Comparing (5.35) and (5.36) with (5.46) and (5.47), the MS79 and CH98 algorithms have sharper bounds on $\|E\|_2$ than the LTL^T-MS79 and LTL^T-CH98 algorithms, respectively. Comparing (5.41) and (5.42) with (5.48) and (5.49), the MS79 and CH98 algorithms have sharper bounds on $\kappa_2(A+E)$ than the LTL^T-MS79 and LTL^T-CH98 algorithms, respectively. In our experiments, however, our LTL^T-MS79 and LTL^T-CH98 algorithms usually performed as well as (and sometimes better than) the MS79 and CH98 algorithms, respectively.

In our experiments on the random matrices with eigenvalues in $[-1, 1]$ and $[-10000, -1]$, $\|E\|_2$ produced by the LTL^T-MS79 and LTL^T-CH98 algorithms were comparable to those by the MS79 and CH98 algorithms, respectively. For the random matrices with eigenvalues in $[-1, 10000]$, our LTL^T-MS79 and LTL^T-CH98 algorithms slightly outperformed the MS79 and CH98 algorithms by keeping $\|E\|_2$ smaller on average, respectively. Figures 5.6 and 5.7 show the result of the MS79 and the LTL^T-MS79 algorithms and that of the CH98 and the LTL^T-CH98 algorithms, respectively.

The 2-phase strategy can also be incorporated into the LTL^T-MS79 and LTL^T-CH98 algorithms. However, this results in the potential problem of large

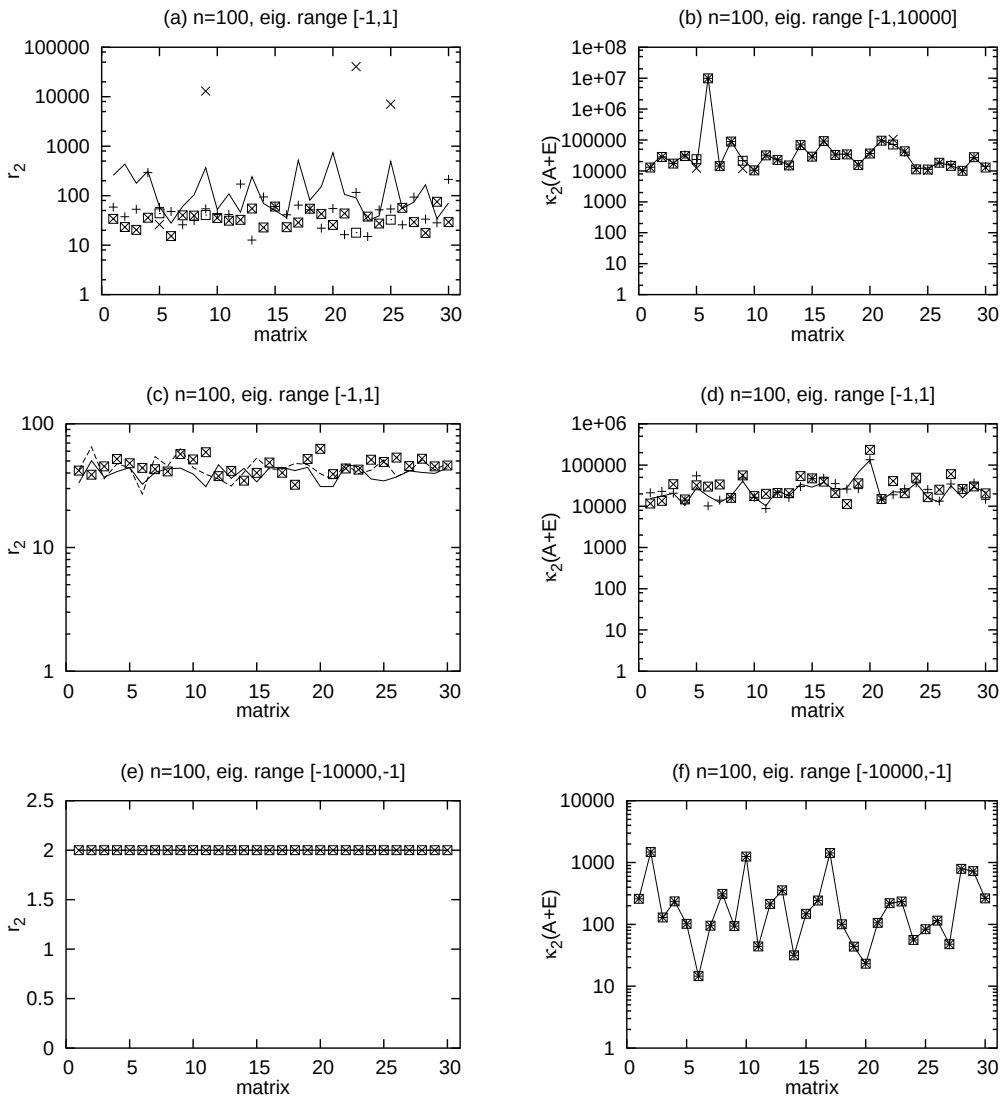

Figure 5.6: Measures of r_2 and $\kappa_2(A+E)$ for the MS79, LTL^T-MS79, 2-phase LTL^T-MS79, relaxed 2-phase LTL^T-MS79 algorithms for 30 random matrices with $n = 100$. Key: MS79 —, LTL^T-MS79 $+$, 2-phase LTL^T-MS79 $\times$, relaxed 2-phase LTL^T-MS79 $\square$.

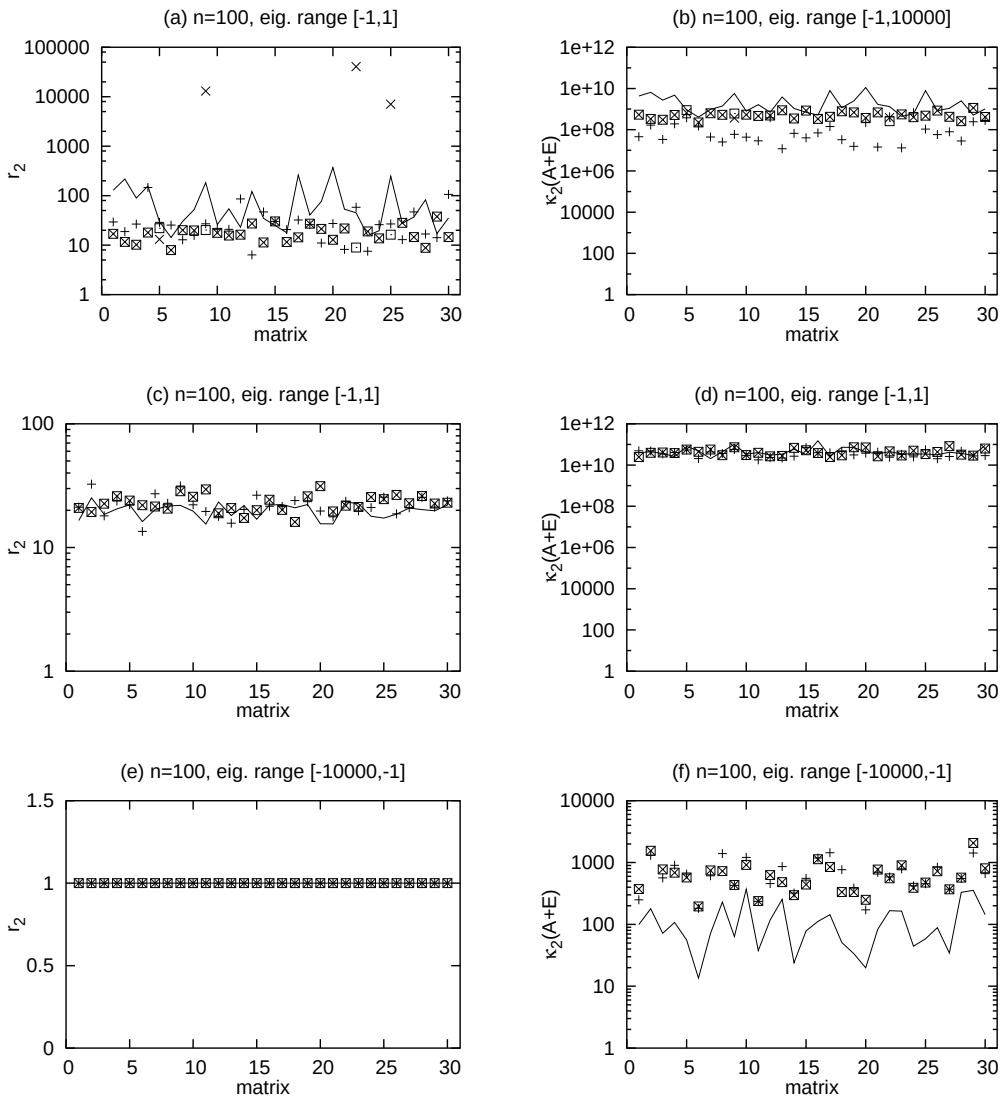

Figure 5.7: Measures of r_2 and $\kappa_2(A+E)$ for the CH98, LTL^T-CH98, 2-phase LTL^T-CH98, relaxed 2-phase LTL^T-CH98 algorithms for 30 random matrices with $n = 100$. Key: CH98 —, LTL^T-CH98 $+$, 2-phase LTL^T-CH98 $\times$, relaxed 2-phase LTL^T-CH98 $\square$.

$\|E\|$, similar to that of the SE90 algorithm. The problem was roughly resolved by relaxing in our experiments, as shown in Figures 5.6 and 5.7. Unfortunately, the problem was not extinguished with the relaxed 2-phase strategy. See the discussion in Subsection 5.5.2. Therefore, we do not advise incorporating the 2-phase or the relaxed 2-phase strategy into the LTL^T-MS79 or the LTL^T-CH98 algorithm.

5.5 Additional Numerical Experiments

Our previous experiments provided good values for the parameters in our methods. Now we present more extensive comparisons among the methods.

We ran three tests in our experiments. The first test contains the random matrices similar to those in [16, 54, 55]. The second test was on the first matrix in [54] for which the SE90 algorithm had difficulties. The third test was on the 33 matrices used in [55]. Our experiments were on a laptop with a Intel Celeron 2.8GHz CPU using IEEE standard arithmetic with machine epsilon $\epsilon_M = 2^{-52} \approx 2.22 \times 10^{-16}$.

5.5.1 Random Matrices

To investigate the behaviors of the factorization algorithms, we experimented on the random matrices with eigenvalues in $[-1, 10000]$, $[-1, 1]$, and $[-10000, -1]$ for dimensions $n = 25, 50, 100$. The random matrices were generated as described in Section 5.2. We compare the performances of the four Type-I algorithms, GMW-I, SE-I, MS79 and LTL^T-MS79, and the four Type-II algorithms, GMW-II, SE99, CH98 and LTL^T-CH98.

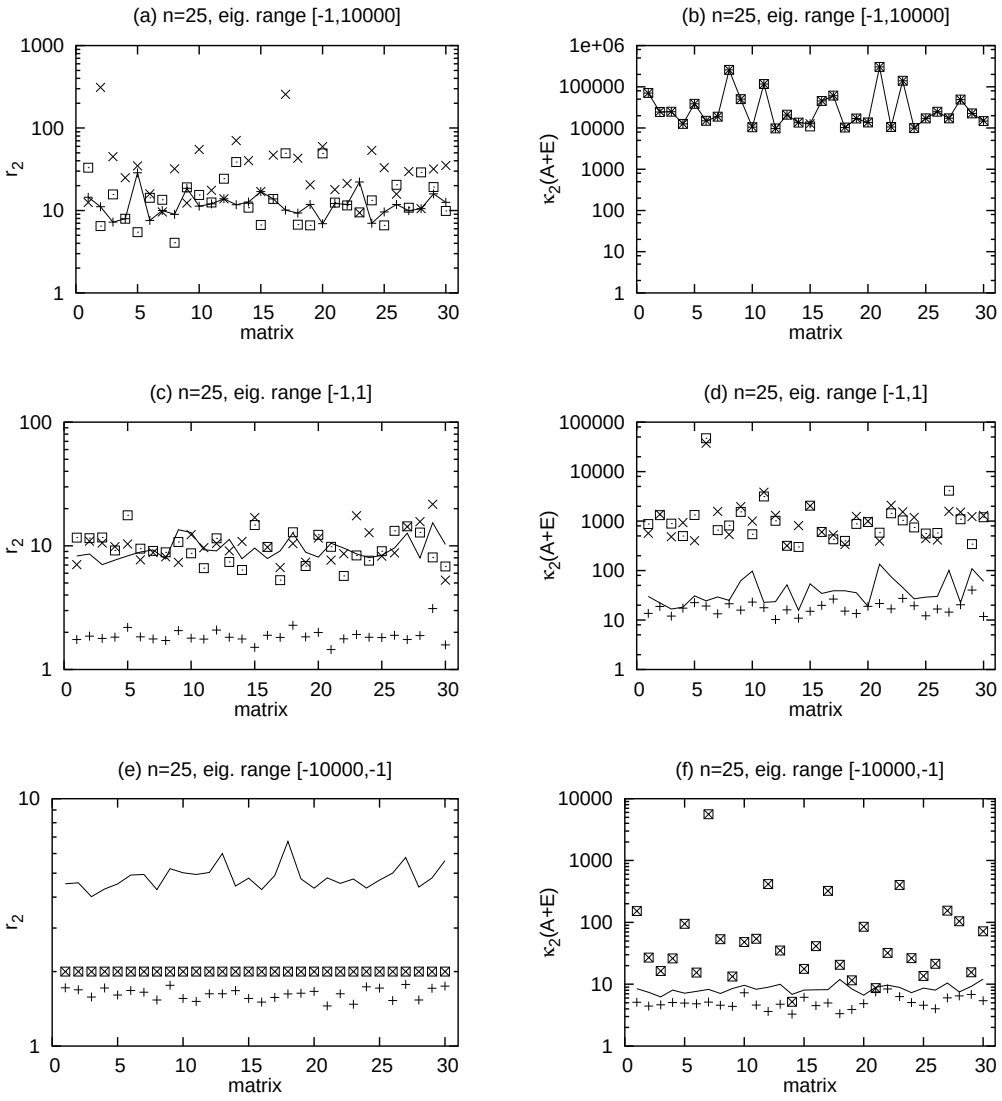

Figure 5.8: Measures of r_2 and $\kappa_2(A+E)$ for GMW-I, SE-I, MS79, and LTL^T-MS79 algorithms for 30 random matrices with $n = 25$. Key: GMW-I —, SE-I $+$, CH98 $\times$, LTL^T-CH98 $\square$.

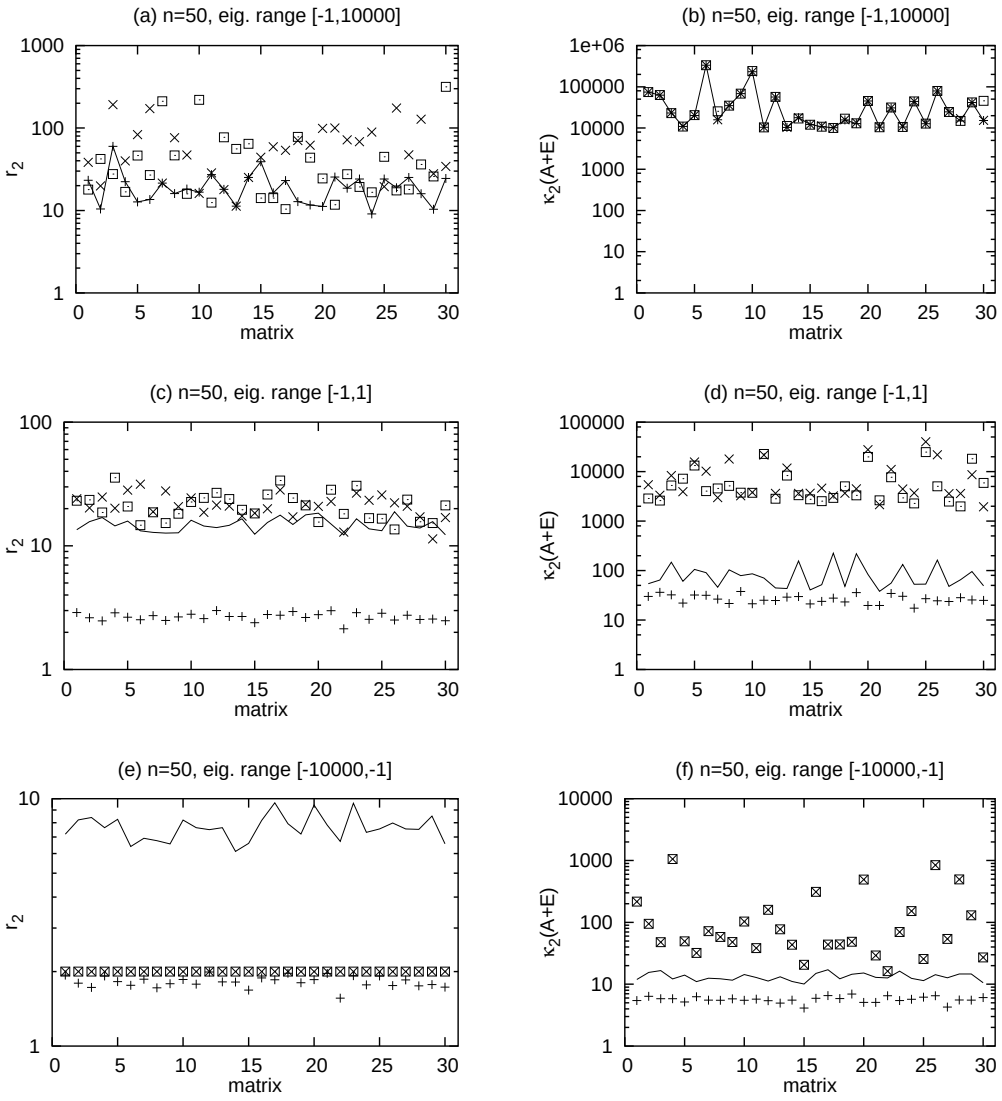

Figure 5.9: Measures of r_2 and $\kappa_2(A+E)$ for GMW-I, SE-I, MS79, and LTL^T-MS79 algorithms for 30 random matrices with $n = 50$. Key: GMW-I —, SE-I $+$, CH98 $\times$, LTL^T-CH98 $\square$.

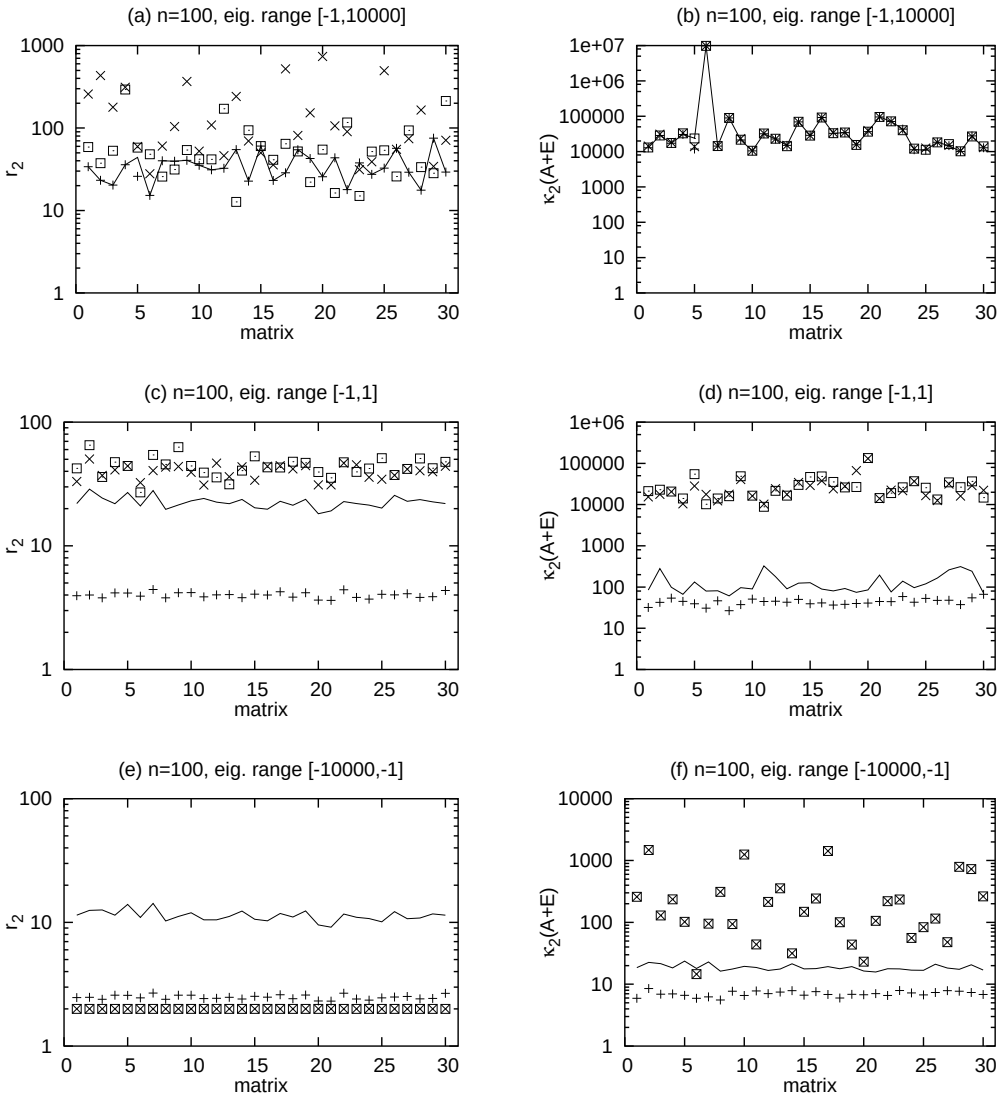

Figure 5.10: Measures of r_2 and $\kappa_2(A+E)$ for GMW-I, SE-I, MS79, and LTL^T-MS79 algorithms for 30 random matrices with $n = 100$. Key: GMW-I —, SE-I $+$, CH98 $\times$, LTL^T-CH98 $\square$.

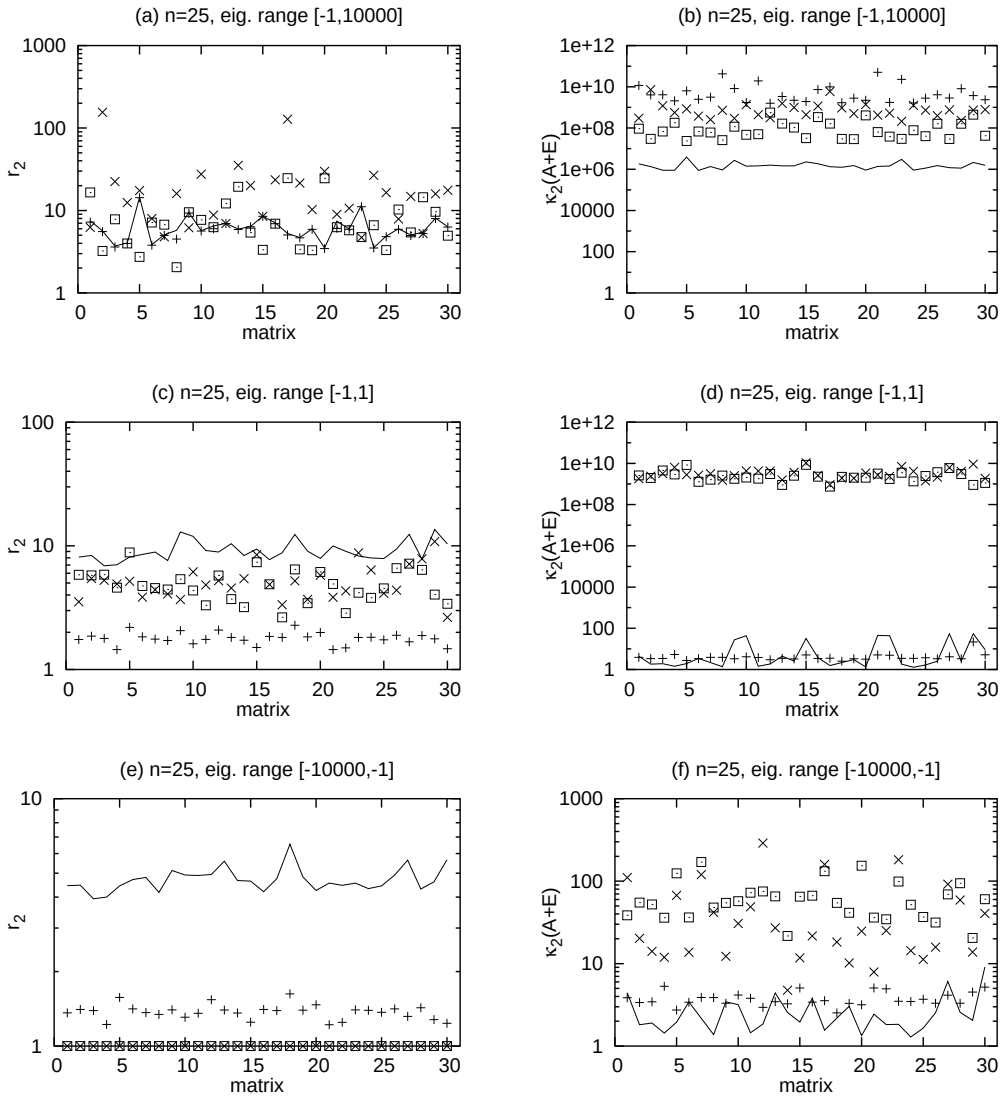

Figure 5.11: Measures of r_2 and $\kappa_2(A+E)$ for GMW-II, SE99, CH98, and LTL^T-CH98 algorithms for 30 random matrices with $n = 25$. Key: GMW-II —, SE99 + , CH98 $\times$, LTL^T-CH98 $\square$.

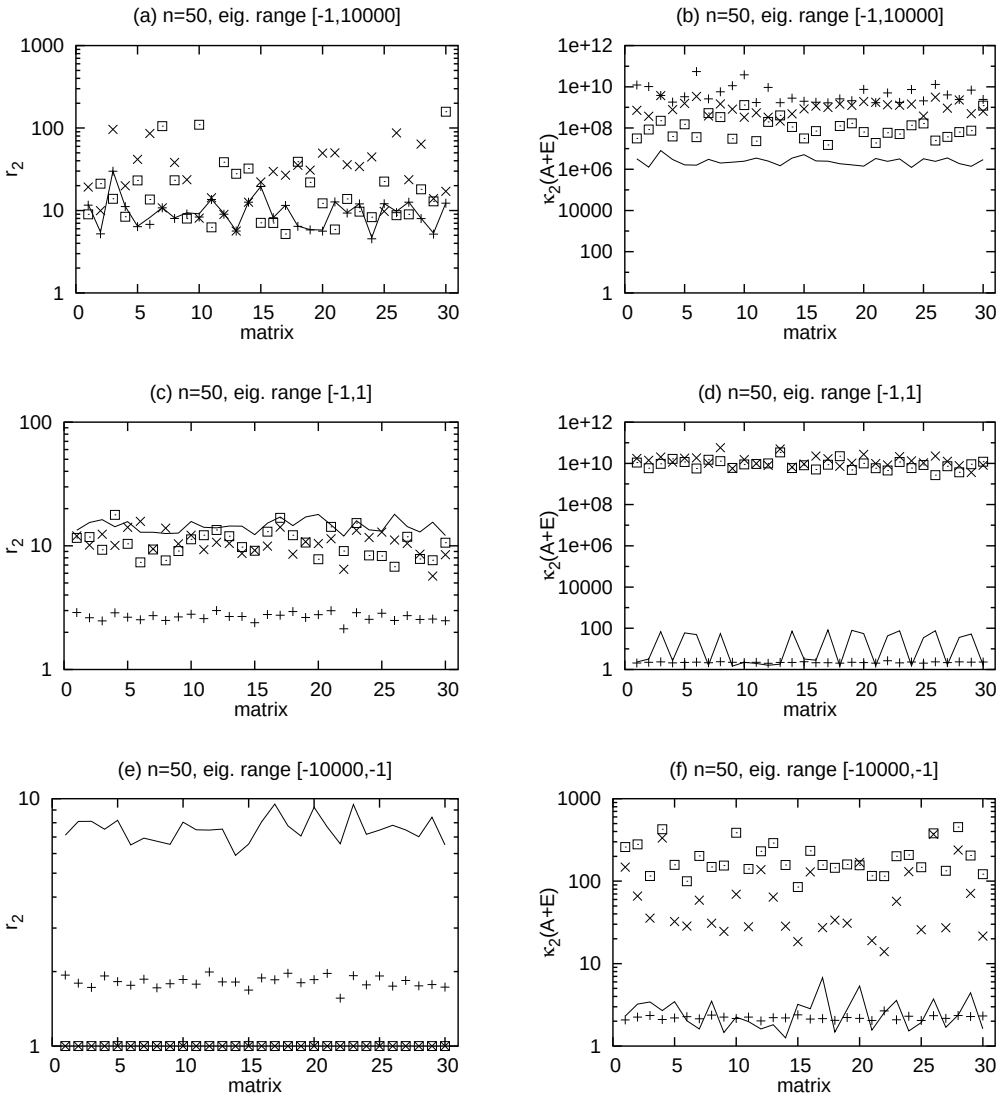

Figure 5.12: Measures of r_2 and $\kappa_2(A+E)$ for GMW-II, SE99, CH98, and LTL^T-CH98 algorithms for 30 random matrices with $n = 50$. Key: GMW-II —, SE99 $+$, CH98 $\times$, LTL^T-CH98 $\square$.

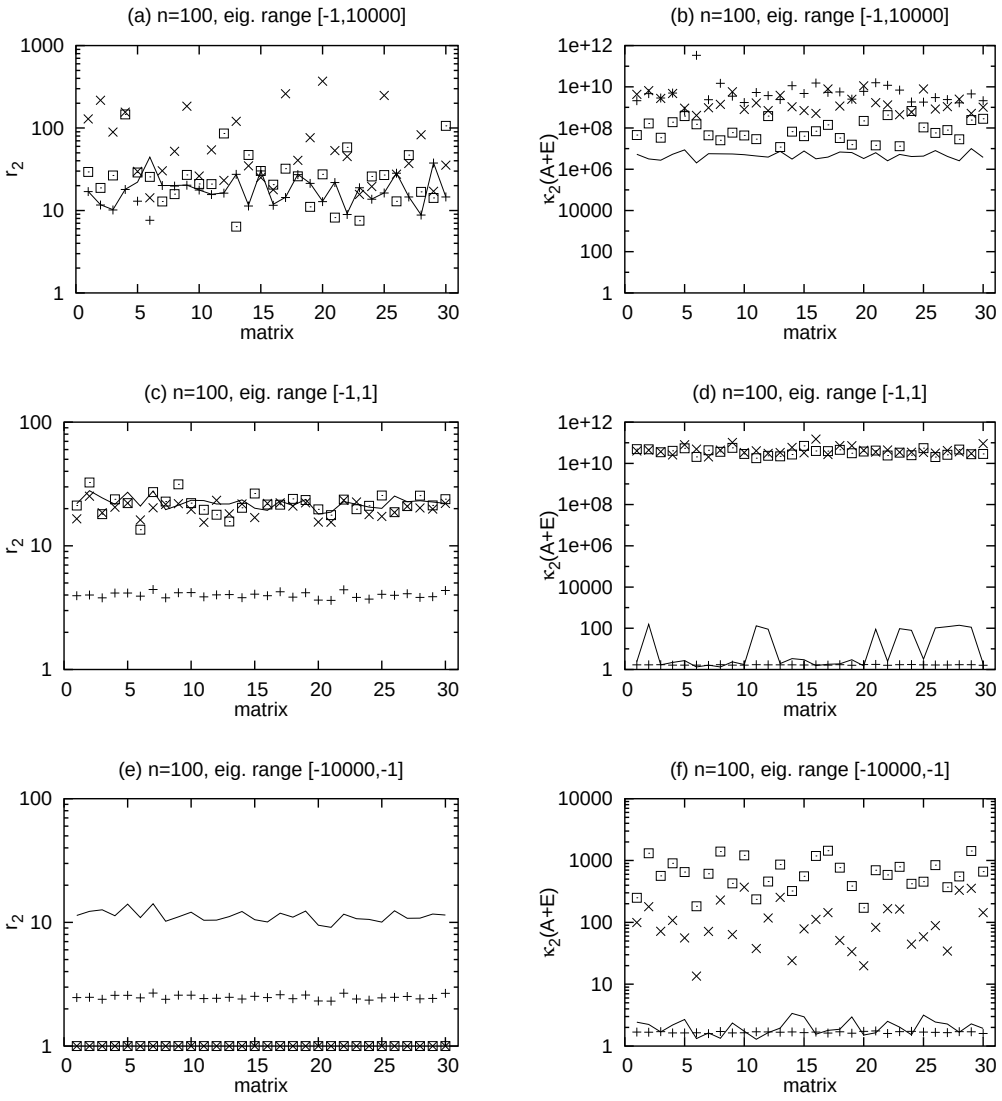

Figure 5.13: Measures of r_2 and $\kappa_2(A+E)$ for GMW-II, SE99, CH98, and LTL^T-CH98 algorithms for 30 random matrices with $n = 100$. Key: GMW-II —, SE99 $+$, CH98 $\times$, LTL^T-CH98 $\square$.

Figures 5.8–5.10 show the results of the Type-I algorithms, whereas Figures 5.11–5.13 show results of the Type-II algorithms. The experiments were on random matrices with sizes $n = 25, 50, 100$. We measure $\|E\|_2$ by $r_2 = \frac{\|E\|_2}{|\lambda_{\min}(A)|}$ as defined in (5.20).

Consider the Type-I algorithms. MS79 and LTL^T-MS79 algorithms generally produced comparable $\|E\|_2$ and condition numbers, but for matrices with eigenvalues in $[-1, 10000]$, LTL^T-MS79 achieved a smaller $\|E\|_2$ than MS79 in several cases. For matrices with eigenvalues in $[-1, 1]$, SE-I outperformed the other Type-I algorithms by not only producing smaller $\|E\|_2$ but also smaller $\kappa_2(A + E)$. For matrices with eigenvalues in $[-10000, -1]$, the GMW-II algorithm produced larger $\|E\|_2$ than the others.

Now compare the Type-II algorithms. In the experiments on the matrices with eigenvalues in $[-1, 10000]$, the GMW-II and SE99 algorithms produced $\|E\|_2$ smaller than the others on average. The LTL^T-CH98 algorithm outperformed the CH98 algorithm by usually achieving a smaller $\|E\|_2$. For the random matrices with eigenvalues in $[-1, 1]$, the SE99 algorithm remains the best. For the random matrices with eigenvalues in $[-10000, -1]$, the CH98 and LTL^T-CH98 algorithms achieved the minimal $\|E\|_2$.

5.5.2 The Benchmark Matrix

Schnabel and Eskow [54] identified the matrix,

$$A = \begin{bmatrix} 1890.3 & -1705.6 & -315.8 & 3000.3 \\ -1705.6 & 1538.3 & 284.9 & -2706.6 \\ -315.8 & 284.9 & 52.5 & -501.2 \\ 3000.3 & -2706.6 & -501.2 & 4760.8 \end{bmatrix}, \tag{5.51}$$

that the SE90 algorithm has difficulties with. It became one of the benchmark matrices for the modified Cholesky algorithms [16, 55]. This matrix has eigenvalues $\{-0.378, -0.343, -0.248, 8.24 \times 10^3\}$.

Table 5.5: Measures of $\|E\|$ and $\kappa_2(A+E)$ for the benchmark matrix (5.51).

Algorithm	r_2	r_F	$\kappa_2(A+E)$
GMW81	2.733	2.674	4.50×10^4
GMW-I	3.014	2.739	4.51×10^4
GMW-II	2.564	2.489	1.64×10^5
SE90	2.78×10^3	3.70×10^3	8.858
SE99	1.759	1.779	1.04×10^{10}
SE-I	3.346	3.289	3.61×10^4
MS79	3.317	2.689	3.33×10^4
CH98	1.659	1.345	9.88×10^7
LTL^T**-MS79**	3.317	2.689	3.33×10^4
LTL^T**-CH98**	1.658	1.344	6.74×10^{10}
LTL^T**-MS79**, 2-phase	3.317	2.689	3.33×10^4
LTL^T**-CH98**, 2-phase	1.658	1.344	8.59×10^{10}
LTL^T**-MS79**, relaxed 2-phase	2.15×10^4	2.03×10^4	3.68×10^4
LTL^T**-CH98**, relaxed 2-phase	2.15×10^4	2.03×10^4	7.47×10^{10}

The measures of $\|E\|_2$ and $\|E\|_F$ in terms of r_2 and r_F, and the condition numbers $\kappa_2(A+E)$ are listed in Table 5.5 for various modified Cholesky algorithms, where the new methods are in boldface. This illustrates the instability of incorporating the relaxed 2-phase strategy into the LTL^T-CH98 and LTL^T-MS79 algorithms, where the relaxation factor was $\mu = 0.1$. In this case the instability can be resolved by dropping the relaxation factor down to $\mu = 10^{-4}$. However, the instability was not extinguished for the matrices A15_1, A15_2, and A15_3 in Subsection 5.5.3, after trying several different relaxation factors.

5.5.3 The 33 Matrices

There were 33 matrices generated by Gay, Overton, and Wright and used by Schnabel and Eskow [55] for the performance evaluation of the modified Cholesky algorithms. These matrices were from the optimization problems where the GMW81 algorithm outperformed the SE90 algorithm.

Table 5.6 summarizes $r_2 = \frac{\|E\|_2}{|\lambda_{\min}(A)|}$ and $\zeta = \lfloor \log_{10}(\kappa_2(A+E)) \rfloor$ for the existing algorithms in the literature, whereas Tables 5.7 gives the result of the new algorithms. Matrix B13_1 is positive definite but extremely ill-conditioned, so that we measure E by $\|E\|_2$ instead of r_2. We see that SE90 did not perform well on several matrices, and the r_2 for the CH98 algorithm is somewhat large on a few matrices (e.g., A6_7). The other methods produced a reasonable E in all cases. For these 33 matrices, Type-I algorithms generally resulted in better conditioning of $A+E$, whereas Type-II algorithms generally produced smaller $\|E\|$, except for the SE90 and CH98 algorithms.

We also noted that incorporating the special treatments from the SE99 algorithm for the last 1×1 and 2×2 Schur complements (see (5.15) and (5.16)) into the GMW-II algorithm can often produce slightly smaller $\|E\|_2$ for matrices close to being positive definite. Similarly, the special treatments for the SE-I algorithm in (5.27) and (5.28) can help the GMW-I algorithm reduce $\|E\|_2$. The detailed discussion is omitted for simplicity.

5.6 Concluding Remarks

The modified Cholesky algorithms in this chapter are categorized in Table 5.8, where the new methods are in boldface. Our conclusions are listed below.

Table 5.6: $r_2 = \frac{\|E\|_2}{|\lambda_{\min}(A)|}$ and $\zeta = \lfloor \log_{10}(\kappa_2(A+E)) \rfloor$ of the existing methods.

Method	GMW81		SE90		SE99		MS79		CH98	
	r_2	ζ	r_2	ζ	r_2	ζ	r_2	ζ	r_2	ζ
A6_1	1.365	5	3.6e+2	1	1.079	9	2.188	5	1.094	8
A6_2	4.844	3	1.175	5	1.180	7	2.304	3	1.152	7
A6_3	4.847	4	1.200	5	1.208	6	2.328	3	1.164	7
A6_4	2.501	5	1.275	5	1.270	8	2.541	4	1.271	8
A6_5	2.347	5	6.503	3	1.448	9	4.512	5	2.257	8
A6_6	1.693	8	2.947	5	1.201	10	2.757	8	1.384	8
A6_7	1.953	12	4.6e+4	5	1.334	10	2.033	12	3.6e+2	8
A6_8	1.953	8	6.611	5	1.138	10	2.033	8	1.030	8
A6_9	1.958	8	47.221	5	1.125	10	2.031	9	1.131	8
A6_10	5.887	8	5.4e+6	0	1.076	11	6.636	8	3.675	8
A6_11	2.334	8	7.3e+6	0	1.648	7	6.049	8	3.570	8
A6_12	4.847	4	1.200	5	1.208	6	2.328	3	1.164	7
A6_13	2.180	2	1.322	5	1.322	6	3.115	2	1.558	8
A6_14	4.847	4	1.200	5	1.208	6	2.328	3	1.164	7
A6_15	5.188	1	1.090	5	1.090	5	2.146	1	1.073	7
A6_16	2.180	2	1.322	5	1.322	6	3.115	2	1.558	8
A6_17	1.527	2	1.246	5	1.246	6	2.752	2	1.376	8
A13_1	2.253	10	8.9e+3	5	1.183	10	3.847	9	57.944	8
A13_2	2.599	8	1.5e+4	5	1.317	10	2.805	8	4.716	8
A15_1	2.421	9	2.5e+7	5	1.895	11	4.165	10	5.954	8
A15_2	2.375	9	3.9e+5	3	1.449	10	2.834	10	9.948	8
A15_3	1.957	6	2.183	5	1.503	10	3.991	7	2.021	8
B6_1	4.901	3	52.418	0	1.773	8	3.024	2	1.512	8
B6_2	4.495	2	45.866	0	2.315	7	4.200	3	2.100	8
B7_1	1.666	2	3.450	2	1.067	2	2.263	2	1.131	8
B7_2	1.932	2	11.005	0	1.309	7	3.320	2	1.660	7
B7_3	1.967	2	6.998	0	1.227	6	2.669	2	1.334	7
B7_4	1.929	2	5.325	1	1.189	6	2.619	2	1.310	7
B8_1	4.164	12	8.7e+2	5	1.279	10	4.164	12	9.705	8
B13_1 (abs.)	0	9	27.15	5	0	9	0	9	0.215	7
B13_2	1.762	7	7.846	5	1.291	10	3.887	7	1.949	8
B26_1	9.833	1	2.234	3	2.364	7	28.293	2	14.146	8
B55_1	3.504	1	1.714	5	1.714	6	95.603	3	47.802	9

Table 5.7: $r_2 = \frac{\|E\|_2}{|\lambda_{\min}(A)|}$ and $\zeta = \lfloor \log_{10}(\kappa_2(A+E)) \rfloor$ of the new methods.

Method	GMW-I		GMW-II		SE-I		LTL^T–MS79		LTL^T–CH98	
	r_2	ζ	r_2	ζ	r_2	ζ	r_2	ζ	r_2	ζ
A6_1	1.989	4	2.111	5	2.181	4	2.153	5	1.077	11
A6_2	4.265	3	3.881	2	2.360	3	2.306	3	1.153	10
A6_3	5.160	2	2.528	11	2.416	2	2.323	3	1.162	10
A6_4	2.574	3	1.290	10	2.541	3	2.756	4	1.378	10
A6_5	3.120	4	1.647	10	2.895	4	3.111	4	1.556	10
A6_6	2.363	7	1.418	5	2.403	6	2.754	8	1.377	10
A6_7	2.189	11	1.331	10	2.109	10	2.032	11	1.352	10
A6_8	2.189	7	1.051	10	2.277	7	2.032	8	1.016	10
A6_9	2.179	8	1.064	10	2.249	8	2.031	9	1.016	10
A6_10	4.883	7	4.399	6	2.031	7	2.438	8	1.219	11
A6_11	2.550	7	2.311	7	1.737	7	3.604	8	1.802	11
A6_12	5.160	2	2.528	11	2.416	2	2.323	3	1.162	10
A6_13	3.289	1	2.971	1	2.643	1	2.911	2	1.455	11
A6_14	5.160	2	2.528	11	2.416	2	2.323	3	1.162	10
A6_15	5.338	1	2.666	11	2.181	1	3.195	1	1.597	10
A6_16	3.289	1	2.971	1	2.643	1	2.911	2	1.455	11
A6_17	2.713	1	2.461	1	2.492	1	2.519	2	1.259	11
A13_1	2.288	10	1.198	10	2.257	9	2.258	9	1.184	10
A13_2	2.767	8	1.406	10	2.627	8	2.642	8	1.324	10
A15_1	5.718	9	5.372	8	3.815	8	4.886	10	2.444	11
A15_2	2.925	8	2.728	8	2.887	8	2.834	10	1.432	10
A15_3	3.953	6	3.789	6	3.006	6	2.689	7	1.344	11
B6_1	2.817	2	2.512	2	3.545	2	2.224	2	1.112	11
B6_2	3.367	2	3.061	2	4.630	2	2.398	2	1.199	11
B7_1	2.062	2	1.663	2	2.005	2	2.019	2	1.010	11
B7_2	2.721	2	1.449	11	2.618	1	7.217	2	3.609	11
B7_3	2.610	2	1.377	11	2.453	1	6.795	2	3.397	11
B7_4	2.538	2	1.337	11	2.378	1	2.683	2	1.342	10
B8_1	4.164	12	2.087	11	2.548	10	4.022	12	2.017	11
B13_1	0	9	0	9	0	9	0	9	0	9
B13_2	5.273	7	4.859	5	2.581	6	2.405	7	1.203	10
B26_1	6.639	1	3.721	2	5.827	1	17.386	2	8.693	11
B55_1	3.504	1	1.752	10	3.428	1	11.289	1	5.645	10

Table 5.8: Categories of various modified Cholesky algorithms.

Category	*Type I*	*Type II*
LDL^T	GMW81, **GMW-I**, **SE-I**	**GMW-II**, SE90, SE99
LBL^T	MS79	CH98
LTL^T	LTL^T**-MS79**	LTL^T**-CH98**

1. The rationale for the algorithms in the GMW class is to bound the off-diagonal elements in $\bar{L}$. The rationale for the algorithms in the SE class is to control the Gershgorin circles in the Schur complements.

2. The nondecreasing strategy can be incorporated into virtually all algorithms which confine the modification to the diagonal. The rationale is that it does not increase $\|E\|_2$ at each stage, and it may keep the subsequent modifications smaller. It is especially favored by the Type-II algorithms, since it can also empirically improve the conditioning of $A + E$.

3. The 2-phase and relaxed 2-phase strategies are incorporated into the SE90 and SE99 algorithms respectively for satisfying Objective 1, whereas they are not required for the GMW81 algorithm.

4. The GMW81 algorithm and its Type-II variant have $\|E\| = O(n^2)$. The 2-phase strategy can drop the bound to be $\|E\| = O(n)$. However, it may result in excessive $\|E\|_2$ for matrices close to being positive definite. The problem can be solved by relaxing. The situation is similar to that of the SE90 and SE99 algorithms. The relaxed 2-phase strategy usually improves the modified LDL^T algorithms.

5. For all algorithms in the GMW class and in the SE class, pivoting is not

required for the theoretical bounds on $\|E\|_2$ and $\kappa_2(A+E)$. In practice, pivoting reduces $\|E\|_2$.

6. Our GMW-II algorithm outperforms GMW81 and GMW-I algorithms by generally keeping $\|E\|_2$ smaller for the random matrices with eigenvalues in $[-1, 10000]$, whereas the GMW81 algorithm outperforms our GMW-I and GMW-II algorithms for the random matrices with eigenvalues in $[-10000, -1]$.

7. In our experiments, the SE99 algorithm and our GMW-II algorithm are the best modified LDL^T algorithms for reasonable $\|E\|$ with matrices with eigenvalues in $[-1, 10000]$, whereas the SE-I algorithm generally produces $\|E\|$ smaller than those for the SE90 and SE99 algorithms for matrices with eigenvalues in $[-10000, -1]$ and $[-1, 1]$.

8. In the experiments in the next chapter, we noted that increasing the relaxation factor μ from 0.75 to 1.0 can significantly improve the performance of the GMW-II algorithm for not only random problems with unspecified entries 65% or more but also the protein problems (see Section 6.4). With these changes, however, Objective 2 was not satisfied as well for the 33 matrices in Section 5.5.3.

For the modified LBL^T factorizations and our new approach via the LTL^T factorization, the concluding remarks are as follows.

1. In worst cases, the MS79 and CH98 algorithms take $\Theta(n^3)$ time more than the standard Cholesky factorization and therefore do not satisfy Objective 4, whereas our LTL^T-MS79 and LTL^T-CH98 algorithms guarantee the $O(n^2)$ modification expense.

2. In experiments on random matrices with eigenvalues $[-1, 10000]$, the LTL^T-MS79 and LTL^T-CH98 algorithms usually produce an $\|E\|_2$ smaller than the MS79 and CH98 algorithms, respectively. Our new approach outperforms the modified LBL^T algorithms in the literature, not only by guaranteeing the $O(n^2)$ modification cost, but also by usually producing a smaller $\|E\|_2$ for matrices close to being positive definite.

3. It is possible to incorporate the 2-phase strategy or the relaxed 2-phase strategy into the LTL^T-MS79 and LTL^T-CH98 algorithms, but the resulting algorithms may produce unreasonably large $\|E\|$, as shown in Figure 5.7 and discussed in Subsection 5.5.2, respectively.

4. The modification arguments δ listed in Table 5.1 aimed at the satisfaction the four objectives. In practice, especially for Type-II algorithms, they could be too small and affect the conditioning, from which difficulty may arise. In the experiments in the next chapter, difficulty was apparent for the CH98 and LTL^T-CH98 algorithms. To amend the problem, we increased the modification tolerance parameter δ to be $\tau\eta$ (used by SE90 algorithm) for both the CH98 and the LTL^T-CH98 algorithms.

Chapter 6

Euclidean Distance Matrix Completion Problems

In this chapter we illustrate the use of the modified Newton methods on a challenging optimization problem, the *Euclidean distance matrix problem*. The *Euclidean distance* between two points p_1, p_2 (column vectors) is defined by

$$\|p_1 - p_2\| = \sqrt{(p_1 - p_2)^T(p_1 - p_2)}.$$

The set of all $n \times n$ real symmetric matrices is denoted by $\mathcal{S}_n$. A matrix $D = [d_{ij}] \in \mathcal{S}_n$ is called a *Euclidean distance matrix* (EDM)[1], if there are points $p_1, p_2, \ldots, p_n$ such that $d_{ij} = \|p_i - p_j\|^2$ for $i, j = 1, 2, \ldots, n$. Apparently a EDM has zero diagonal and all off-diagonal elements nonnegative.

A matrix $A = [a_{ij}]$ is called *symmetric partial* if there are unspecified entries, and a_{ij} is specified and equal to a_{ji} whenever a_{ji} is specified. Let

$$\mathcal{C}(A) = \{D = [d_{ij}] \in \mathcal{S}_n : d_{ij} = a_{ij} \text{ for all specified entries } a_{ij} \text{ in } A.\}.$$

A matrix D is called a *EDM completion* of A if and only if $D \in \mathcal{C}(A)$ is a EDM. The Euclidean distance matrix completion problem (EDMCP) is to find a EDM D that completes a given symmetric partial matrix A.

[1]Some authors define a EDM $D = [d_{ij}]$ by $d_{ij} = \|p_i - p_j\|$, so our D is their $D \circ D$, where $\circ$ denotes *Hadamard* (elementwise) product.

Theoretical properties of EDMs have been well studied (e.g., [2, 6, 33, 56]), and numerical optimization is widely used to tackle the EDMCPs (e.g., [3, 44, 61, 69]). One prominent application of the EDMCP is protein structure prediction. The interatomic distance information comes from the structural interpretation of nuclear magnetic resonance data. We transform the EDMCP into a global optimization problem [44, 69], and use modified Newton methods to generate descent directions. To reach the global minimum, we develop a dimensional relaxation method. This approach is not new. To our knowledge, it was first suggested by Crippen [18], and later used by Havel [35], and Purisima and Scheraga [51].

This chapter is organized as follows. Section 6.1 introduces the basic properties of EDMs. Section 6.2 transforms the EDMCP into three different optimization problems. Section 6.3 presents our dimensional relaxation method to tackle the EDMCP via global optimization. Experimental results are given in Section 6.4 and a conclusion in Section 6.5.

6.1 Distance Geometry

Section 6.1.1 gives the preliminaries and Section 6.1.2 presents the linear transformations for the EDMs.

6.1.1 Preliminaries

We call $D \in R^{n\times n}$ a *predistance matrix* if it is symmetric and has zero diagonal. Clearly every EDM is a predistance matrix but not vice versa, even if all entries are nonnegative (e.g., distances might violate the triangle inequality). It is well-known [3, 33, 56, 68] that a predistance matrix $D \in \mathcal{S}_n$ is a EDM if and only if

D is negative semidefinite in the subspace

$$M := \{x \in R^n : x^T e = 0\}.$$

Here and throughout this chapter, e is the column vector of ones. We also denote the column vector and the diagonal matrix formed from the diagonal elements in D by $\mathrm{diag}(D)$ and $\mathrm{Diag}(D)$, respectively.

Let $V \in R^{n\times(n-1)}$ be a matrix whose columns form an orthonormal basis of M and let

$$\bar{V} := \left[\begin{array}{cc} V & \frac{e}{\sqrt{n}} \end{array}\right] \in R^{n\times n}.$$

Then $\bar{V}$ is orthogonal (i.e., $\bar{V}\bar{V}^T = \bar{V}^T\bar{V} = I$). The orthogonal projection onto M, denoted by J, is

$$J := VV^T = I - \frac{ee^T}{n} \in \mathcal{S}_n.$$

Note that the Householder reflection is $I - \frac{2ee^T}{n}$. Although the choice of V is not unique, J is unique and $J^2 = J$.

Now define the linear operator

$$\mathcal{T}(D) := -\frac{1}{2}JDJ. \tag{6.1}$$

The inner product of $A, B \in \mathcal{S}_n$ is given by the trace product

$$\langle A, B\rangle := \mathrm{trace}(AB).$$

Then $\langle \mathcal{T}(A), B\rangle = \langle A, \mathcal{T}(B)\rangle$ for all $A, B \in \mathcal{S}_n$; therefore, $\mathcal{T}$ is self-adjoint. Also, $-2\mathcal{T}$ is idempotent, because $-2\mathcal{T}(D) = -2\mathcal{T}(-2\mathcal{T}(D))$ for all $D \in \mathcal{S}_n$.

Note that $D \in \mathcal{S}_n$ is a EDM if and only if D is a predistance matrix (i.e., zero

diagonal) and $\mathcal{T}(D)$ is positive semidefinite. Let $B := \mathcal{T}(D)$. A matrix

$$P = \begin{bmatrix} p_1^T \\ p_2^T \\ \vdots \\ p_n^T \end{bmatrix}$$

is called a *realization* of D if $PP^T = B$. Three properties are listed below:

- Since $(V^T e = 0) \implies (Be = 0) \implies (P^T e = 0)$, the centroid of the points $p_1, p_2, \ldots, p_n$ is at the origin.
- The choice of P is not unique. Given a realization P of D, PQ is also a realization for any orthogonal Q.
- Denote the rank of B by r (i.e., $\text{rank}(B) = r$). There exists a realization $P \in R^{n \times r}$ of D. In such a case, there is no proper hyperplane in R^r that contains the points $p_1, p_2, \ldots, p_n$, since $\text{rank}(P) = r$. Here r is called the *embedding dimension*, the least dimension to realize the given EDM D.

6.1.2 Linear Transformations

We use $X \succeq 0$ to indicate that the matrix X is positive semidefinite. Denote the sets of $n \times n$ symmetric positive semidefinite matrices by

$$\mathcal{S}_n^+ := \{S \in \mathcal{S}_n : S \succeq 0\}.$$

Define the *centered* and *hollow* subspaces of the set of $n \times n$ symmetric matrices $\mathcal{S}_n$ by

$$\mathcal{B}_n := \{B \in \mathcal{S}_n : Be = 0\} \text{ and } \mathcal{D}_n := \{D \in \mathcal{S}_n : \text{diag}(D) = 0\},$$

respectively. Let

$$\mathcal{B}_n^+ := \{B \in \mathcal{B}_n : B \succeq 0\}$$

and

$$\mathcal{D}_n^- := \{D \in \mathcal{D}_n : x^T D x \leq 0 \text{ for } x^T e = 0\}. \tag{6.2}$$

All these sets are convex. Now define the linear operator,

$$\mathcal{K}(B) := \text{diag}(B)\, e^T + e\, \text{diag}(B)^T - 2B. \tag{6.3}$$

Then its adjoint is

$$\mathcal{K}^*(A) = 2(\text{Diag}(Ae) - A),$$

since $\langle \mathcal{K}(B), A\rangle = \langle B, \mathcal{K}^*(A)\rangle$ for all $A, B \in \mathcal{S}_n$.

Lemma 6.1 *The linear operators $\mathcal{T}$ and $\mathcal{K}$ satisfy*

$$\mathcal{T}(\mathcal{D}_n^-) = \mathcal{B}_n^+ \text{ and } \mathcal{K}(\mathcal{B}_n^+) = \mathcal{D}_n^-.$$

Moreover, $\mathcal{T}$ on $\mathcal{D}_n^-$ is the inverse function of $\mathcal{K}$ on $\mathcal{B}_n^-$.

Proof From (6.1) and (6.3) we can derive that $\mathcal{T}$ on $\mathcal{D}_n$ is the inverse function of $\mathcal{K}$ on $\mathcal{B}_n$. For any $D \in \mathcal{D}_n^-$, D is negative semidefinite on $\{x \in R^n : x^T e = 0\}$, which is the same as $\{Jy : y \in R^n\}$. This implies that $\mathcal{T}(D) = -\frac{1}{2} J D J^T$ is positive semidefinite and therefore in $\mathcal{B}_n^+$. For any $x \in R^n$ with $x^T e = 0$, $x^T \mathcal{K}(B) x = -2xBx^T$. Therefore, if $B \in \mathcal{B}_n$ is positive semidefinite, then $\mathcal{K}(B) \in \mathcal{D}_n^-$. □

We now define the two composite linear operators

$$\begin{aligned} \mathcal{T}_V(D) &:= V^T \mathcal{T}(D) V = -\frac{1}{2} V^T D V, \\ \mathcal{K}_V(X) &:= \mathcal{K}(V X V^T). \end{aligned}$$

The adjoint of $\mathcal{K}$ is $\mathcal{K}^*$; therefore, $\langle \mathcal{K}_V(X), Y\rangle = \langle X, \mathcal{K}_V^*(Y)\rangle$ for $X \in \mathcal{S}_{n-1}$ and $Y \in \mathcal{S}_n$, where

$$\mathcal{K}_V^*(Y) = V^T \mathcal{K}^*(Y) V,$$

the adjoint of $\mathcal{K}_V(X)$.

Lemma 6.2 *The linear operators $\mathcal{T}_V$ and $\mathcal{K}_V$ satisfy*

$$\mathcal{T}_V(\mathcal{D}_n^-) = \mathcal{S}_{n-1}^+ \text{ and } \mathcal{K}_V(\mathcal{S}_{n-1}^+) = \mathcal{D}_n^-.$$

Moreover, $\mathcal{T}_V$ on $\mathcal{D}_n^-$ is the inverse function of $\mathcal{K}_V$ on $\mathcal{S}_{n-1}^+$.

Proof This is straightforward from Lemma 6.1 and the definition of V. □

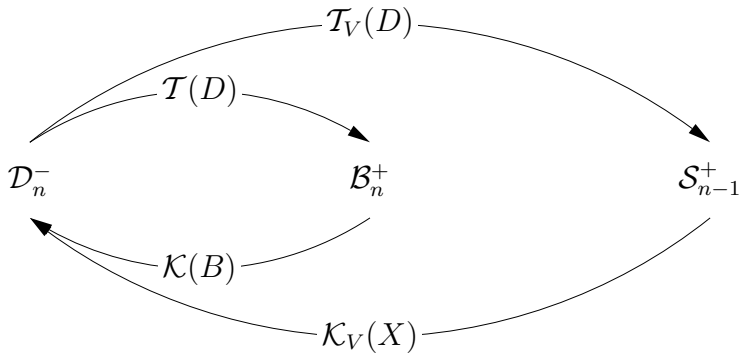

Figure 6.1: Relationships between the linear transformations.

By Lemma 6.1 and Lemma 6.2, we conclude that the following four conditions are equivalent for a given predistance matrix $D \in \mathcal{S}_n$.

1. D is a EDM.
2. $D \in \mathcal{D}_n^-$.
3. $\mathcal{T}(D) \in \mathcal{B}_n^+$.
4. $\mathcal{T}_V(D) \in \mathcal{S}_{n-1}^+$.

The inverse functions of $\mathcal{T}$ and $\mathcal{T}_V$ on $\mathcal{D}_n^-$ are $\mathcal{K}$ on $\mathcal{B}_n^+$ and $\mathcal{K}_V$ on $\mathcal{S}_{n-1}^+$, respectively. The relationships are presented in Figure 6.1. Using these transformations we can convert the set of EDMs to another set, giving us several approaches to the EDMCP.

6.2 Solving EDMCP via Numerical Optimization

In the literature, attempts have been made to solve the EDMCP via numerical optimization. Using the transformations in Figure 6.1, we may choose $\mathcal{D}_n^-$, $\mathcal{B}_n^+$ or $\mathcal{S}_{n-1}^+$ as the domain and the range for the objective function. We present three optimization programs in Sections 6.2.1–6.2.3. In all cases a given symmetric partial matrix A has a distance matrix completion if and only if the global minimum is zero.

6.2.1 Trosset's Formulation

Trosset [61] used the formulation

$$\begin{aligned} \min_{B,D} \quad & \|B - \mathcal{T}(D)\|_F^2 \\ \text{subject to} \quad & D \in \mathcal{C}(A), B \in \mathcal{B}_n^+, \end{aligned} \tag{6.4}$$

where A is a given $n \times n$ symmetric partial matrix. If the embedding dimension r is known, we may impose the constraint $\text{rank}(B) = r$.

Given a symmetric matrix $B \in R^{n \times n}$, define

$$F_r(B) = \sum_{i=1}^{n-r} \lambda_i(B)^2 + \sum_{i=n-r+1}^{n} (\lambda_i(B) - \max\{\lambda_i(B), 0\})^2,$$

where $\lambda_i(B)$ is the ith smallest eigenvalue of B. Then a predistance matrix D is a EDM that can be embedded in r-dimensional space if and only if $F_r(\mathcal{T}(D)) = 0$. Hence program (6.4) can be transformed into

$$\begin{aligned} \min_{D} \quad & F_r(\mathcal{T}(D)) \\ \text{subject to} \quad & D \in \mathcal{C}(A). \end{aligned} \tag{6.5}$$

The program size of Trosset's formulation (6.5) is proportional to the number of unspecified entries. It is favored for problems with only a few unspecified entries. On the other hand, his method is discouraged for problems with a large number of unspecified entries (e.g., $O(n^2)$). See [61] for details.

6.2.2 Semidefinite Programming

Alfakih, Khandani and Wolkowicz [3] tackled the EDMCP via semidefinite programming. Their program formulation is

$$\begin{aligned} \min_{X} \quad & \|H \circ (A - \mathcal{K}_V(X))\|_F^2 \\ \text{subject to} \quad & X \in \mathcal{S}_{n-1}^+, \end{aligned} \tag{6.6}$$

where $\circ$ denotes the Hadamard (elementwise) product, and $H = [h_{ij}]$ is a weight matrix such that $h_{ij} > 0$ if a_{ij} is specified, and otherwise $h_{ij} = 0$. We can set $h_{ij} := 1$ for a_{ij} specified to minimize the Frobenius norm of the partial error matrix. On the other hand, if relative distance errors are the concern, we may set $h_{ij} := 1/a_{ij}$. The weights of the distances may also depend on their relative uncertainty. This weight function also appears in the global optimization formulation (6.8) in Section 6.2.3.

This program formulation is convex. Hence any local minimum will also be a global minimum. On the other hand, the working domain in (6.6) is a symmetric

positive semidefinite matrix $X \in R^{(n-1)\times(n-1)}$, so the number of variables is $O(n^2)$.

In this program formulation, the embedding dimension does not have to be known in advance. However, a minimizer X to (6.6) can have a high rank. In this case, the techniques in [4] can be applied to reduce the rank of X without leaving the minimum [3].

6.2.3 Global Optimization

Instead of solving the optimization problem (6.5) or (6.6), we consider the program formulation

$$\begin{aligned} \min_{B} \quad & \|H \circ (A - \mathcal{K}(B))\|_F^2 \\ \text{subject to} \quad & B \in \mathcal{B}_n^+, Be = 0, \text{rank}(B) = r, \end{aligned} \tag{6.7}$$

where $H = [h_{ij}]$ is a weight matrix. Note that we impose the embedding dimension r in the constraints.

Let $P \in R^{n\times r}$ be a realization of the resulting EDM D so that $B = PP^T$ and $Pe = 0$. Problem (6.7) has the same minimum as

$$\begin{aligned} \min_{P} \quad & \|H \circ (A - \mathcal{K}(PP^T))\|_F^2 \\ \text{subject to} \quad & P \in R^{n\times r}. \end{aligned} \tag{6.8}$$

A careful reader may notice that the equality constraints $Pe = 0$ are not present in (6.8). In Section 6.2.4 we will show that removing the equality constraints does not change the global minimum. Denote the objective function by

$$\begin{aligned} f(P) &:= \|H \circ (A - \mathcal{K}(PP^T))\|_F^2 \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(a_{ij} - d_{ij})^2 \end{aligned} \tag{6.9}$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(a_{ij} - \|p_i - p_j\|^2)^2.$$

The last term is frequently used as the objective function in the literature (e.g., [37, 44, 69]).

Compared with (6.5) and (6.6), the program (6.8) has the key advantage of the relatively small number of variables, since $P \in R^{n\times r}$. For real problems such as protein structure prediction, r is a small constant (e.g., $r = 3$), and therefore the problem size is $O(n)$. On the other hand, since the program (6.8) is not convex, modified Newton methods can converge to a local minimum, whereas a global minimizer is required to declare a solution of the EDMCP.

We now derive the gradient and the Hessian matrix of $f(P)$ defined by (6.9). Note that $f(P)$ is the objective function of the global optimization problem (6.8). Denote $B = [b_{ij}]$, $P = [p_{ij}]$, $\mathcal{K}(B) = [d_{ij}]$. By (6.3) and $B = PP^T$,

$$\begin{aligned} d_{ij} &= b_{ii} + b_{jj} - 2b_{ij} \\ &= \sum_{k=1}^{r} p_{ik}^2 + \sum_{k=1}^{r} p_{jk}^2 - 2\sum_{k=1}^{r} p_{ik}p_{jk} \end{aligned}$$

for $i, j = 1, 2, \ldots, n$. Therefore,

$$\frac{\partial d_{ij}}{\partial p_{st}} = \begin{cases} 2(p_{st} - p_{jt}) & \text{if } s = i \neq j, \\ 2(p_{st} - p_{it}) & \text{if } s = j \neq i, \\ 0 & \text{otherwise.} \end{cases}$$

and

$$\frac{\partial d_{ij}}{\partial p_{st}\partial p_{uv}} = \begin{cases} 2 & \text{if } ((s = u = i \neq j) \vee (s = u = j \neq i)) \wedge (t = v), \\ -2 & \text{if } ((s = i \neq u = j) \vee (s = j \neq u = i)) \wedge (t = v), \\ 0 & \text{otherwise.} \end{cases}$$

By (6.9),

$$\frac{\partial f}{\partial p_{st}} = 2\sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}^2(d_{ij} - a_{ij})\frac{\partial d_{ij}}{\partial p_{st}}$$

$$
\begin{aligned}
&= 4\sum_{j=1}^{n} h_{sj}^2(d_{sj}-a_{sj})(p_{st}-p_{jt}) + 4\sum_{i=1}^{n} h_{is}^2(d_{is}-a_{is})(p_{st}-p_{it}) \\
&= 8\sum_{i=1}^{n} h_{is}^2(d_{is}-a_{is})(p_{st}-p_{it}).
\end{aligned}
$$

If $u = s$,

$$
\begin{aligned}
\frac{\partial f}{\partial p_{st}\partial p_{uv}} &= 8\sum_{i=1}^{n} h_{is}^2(2(p_{sv}-p_{iv})(p_{st}-p_{it}) + (d_{is}-a_{is})\frac{\partial}{\partial p_{uv}}(p_{st}-p_{it})) \\
&= \begin{cases} 16\sum_{i=1}^{n} h_{is}^2(p_{sv}-p_{iv})(p_{st}-p_{it}) & \text{if } v \neq t, \\ 8\sum_{i=1}^{n} h_{is}^2(2(p_{st}-p_{it})^2 + (d_{is}-a_{is})) & \text{if } v = t. \end{cases}
\end{aligned}
$$

If $u \neq s$,

$$
\begin{aligned}
\frac{\partial f}{\partial p_{st}\partial p_{uv}} &= 8h_{us}^2(2(p_{uv}-p_{sv})(p_{st}-p_{ut}) + (d_{us}-a_{us})\frac{\partial}{\partial p_{uv}}(p_{st}-p_{ut})) \\
&= \begin{cases} 16h_{us}^2(p_{uv}-p_{sv})(p_{st}-p_{ut}) & \text{if } v \neq t, \\ -8h_{us}^2(2(p_{st}-p_{ut})^2 + (d_{us}-a_{us})) & \text{if } v = t. \end{cases}
\end{aligned}
$$

As a result, we may obtain the gradient and the Hessian matrix of $f(P)$. Therefore, problem (6.8) can be solved using Newton's method for minimization.

6.2.4 Equality Constraints

Our program formulation (6.8) has linear equality constraints $Pe = 0$. In this section we show that the linear constraints are dispensable.

Lemma 6.3 *Suppose we are given $P \in R^{n\times r}$ with each row representing a point in the Euclidean space. Any rigid transformation of P, denoted by P_1, satisfies*

$$\mathcal{K}(P_1P_1^T) = \mathcal{K}(PP^T),$$

where the linear transformation $\mathcal{K}$ is defined in (6.3).

Proof We denote $P_1 = PQ + eq^T$, where the orthogonal matrix $Q \in R^{r\times r}$ represents the rotation/reflection and the column vector $q \in R^r$ corresponds to the translation. Then

$$\begin{aligned} & \mathcal{K}(P_1P_1^T) - \mathcal{K}(PP^T) \\ = \; & \mathcal{K}(P_1P_1^T - PP^T) \\ = \; & \mathcal{K}(PQqe^T + eq^TQ^TP^T + eq^Tqe^T) \\ = \; & \mathcal{K}(PQqe^T + eq^TQ^TP^T) + q^Tq\mathcal{K}(ee^T) \\ = \; & 0. \end{aligned}$$

The last equality is because $\mathcal{K}(pe^T + ep^T) = 0$ for all vectors p. □

By Lemma 6.3, it is not required to have the centroid of the points in P at the origin. Removing the linear equality constraints $Pe = 0$ in program (6.8) results in an unconstrained optimization problem. However, the equality constraints reduce the flexibility of P and sometimes help the convergence to a global minimum in our experiments (e.g., $n = 10$ in Tables 6.2 and 6.3).

6.2.5 Inequality Constraints

For unspecified entries in a partial distance matrix, we can bound their ranges by triangle inequalities, used, for example, in [35, 61].

Since the (i, j) entry of D is $d_{ij} = \|p_i - p_j\|^2$ for $i, j = 1, \ldots, n$, the triangle inequalities gives

$$|\sqrt{d_{ik}} - \sqrt{d_{jk}}| \le \sqrt{d_{ij}} \le \sqrt{d_{ik}} + \sqrt{d_{jk}}$$

for $i, j, k = 1, \ldots, n$. Therefore,

$$\max_k |\sqrt{d_{ik}} - \sqrt{d_{jk}}| \le \sqrt{d_{ij}} \le \min_k (\sqrt{d_{ik}} + \sqrt{d_{jk}})$$

for $i, j = 1, \ldots, n$. Since $D \in \mathcal{C}(A)$,

$$\max_{a_{ik},\, a_{jk} \text{ specified.}} (\sqrt{a_{ik}} - \sqrt{a_{jk}})^2 \leq d_{ij} \leq \min_{a_{ik},\, a_{jk} \text{ specified.}} (\sqrt{a_{ik}} + \sqrt{a_{jk}})^2 \quad (6.10)$$

for each pair of i, j such that a_{ij} is not specified[2].

Note that in Trosset's formulation (6.5) the inequalities are linear, whereas in the global optimization formulation (6.8) the inequalities are nonlinear.

6.3 Improving the Convergence by Careful Initialization and Dimensional Relaxation

Several methods have been proposed to tackle the EDMCP via the global optimization formulation. For example,

1. Crippen [18] suggested a dimensional relaxation scheme, where he called it *energy embedding*.

2. Glunt, Hayden, and Raydan [31] proposed the spectral gradient method and the data box algorithm.

3. Havel [35] used simulated annealing optimization.

4. Hendrickson [37] presented a divide-and-conquer algorithm.

5. Moré and Wu [44] used the smoothing and continuation method via Gaussian transformation.

6. Zou, Bird, and Schnabel [69] developed a stochastic/perturbation algorithm.

[2]If no pair of a_{ik}, a_{jk} is specified, then further investigation is required to obtain a bound.

The authors in [37, 44, 69] minimize (6.9), while the authors in [18, 31, 35] minimize[3]

$$\sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(\sqrt{a_{ij}} - \|p_i - p_j\|)^2. \tag{6.11}$$

In this section we present our initialization methods and dimensional relaxation strategy for global optimization.

6.3.1 EDM Initialization

Consider the global optimization formulation (6.8). A good initial estimate of the configuration can speed up the convergence and increase the chance of reaching the global minimum. We first compute an initial estimated predistance matrix, denoted by F, and extend it to a EDM. We have explored two methods:

Method 1. The replacement of the negative eigenvalues of $\mathcal{T}(F)$ by zero.

Method 2. The computation of the nearest distance matrix to F.

Given an $n \times n$ partial distance matrix A, the initial predistance matrix $F \in \mathcal{C}(A)$ is computed as follows. The triangle inequalities (6.10) give bounds on the unspecified entries in A. For each bounded unspecified entry, we take the median of its bounds. For each unbounded entry, we fill it by the median of all specified entries. For protein problems, since the unspecified entries are usually the largest distances, we fill the unbounded entries by the maximum known distance value.

Now we describe our first method to modify F to determine a EDM. We first compute $C := \mathcal{T}(F) \in R^{n\times n}$, and its spectral decomposition $C = Q\Lambda Q^T$. We determine Λ_+, whose jth diagonal element is $\max\{\lambda_j, 0\}$. Recall that the

[3]Some authors (e.g., [18]) called (6.11) the an *energy function* in the sense that a_{ij} is the desired energetically minimal value of the distance between the ith and jth atoms.

predistance matrix F is a EDM if and only if $C \in R^{n\times n}$ is positive semidefinite. Using the property that $\mathcal{T}(D)$ and $\mathcal{K}(B)$ are inverse functions of each other, we compute the EDM $\hat{F} := K(Q\Lambda_+ Q^T)$.

The other approach is to find the *nearest* distance matrix to F. In other words, the objective is to find the solution to

$$\begin{aligned} \min_{D} \quad & \|D - F\|_F^2 \\ \text{such that} \quad & D \text{ is a EDM.} \end{aligned} \tag{6.12}$$

We use an alternating projection algorithm presented in [30]. The Frobenius norm of a symmetric matrix S can be computed by the trace product $\|S\|_F = \sqrt{\langle S, S\rangle}$. Since a matrix is a EDM if and only if it has zero diagonal and is negative semidefinite for x satisfying $x^T e = 0$, the set of EDMs is the intersection of the two convex sets

$$\mathcal{D}_1 = \{S : S \in \mathcal{S}_n, \text{Diag}(S) = 0\}$$

and

$$\mathcal{D}_2 = \{S : S \in \mathcal{S}_n, x^T S x \leq 0 \text{ for } x^T e = 0\}.$$

The projections onto $\mathcal{D}_1$ and $\mathcal{D}_2$ are denoted by $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively. In other words, given a symmetric matrix S, $\mathcal{P}_1(S)$ and $\mathcal{P}_2(S)$ are the matrices in $\mathcal{D}_1$ and $\mathcal{D}_2$ nearest to S, respectively. The alternating projection algorithm computes

$$\mathcal{P}_1(F), \mathcal{P}_2(\mathcal{P}_1(F)), \mathcal{P}_1(\mathcal{P}_2(\mathcal{P}_1(F))), \ldots$$

until it converges to the minimizer of (6.12). Clearly $\mathcal{P}_1(S) = S - \text{Diag}(S)$. See [30] for the formula for $\mathcal{P}_2$, a modified alternating projection algorithm, and discussions on convergence.

By either method, we obtain an estimated distance matrix, denoted by $\hat{F}$. The rank of $\mathcal{T}(\hat{F})$, denoted by $\hat{r}$, is the embedding dimension of $\hat{F}$. For real

problems the embedding dimension r is typically small (e.g., protein problems have $r = 3$), whereas the $\hat{r}$ is usually high. In this case we can use the spectral decomposition of $\mathcal{T}(\hat{F})$, keep the r largest positive eigenvalues, replace the others by zero, and obtain a EDM, denoted by $\hat{D}$, with the embedding dimension r.

We have experimented on proteins 1BPI, 1CBNa, 1MBC, and 2GDM, considering only the $C\alpha$ atoms, one per amino acid. We dropped $5\%, 10\%, \ldots, 95\%$ of the distances, retaining the shortest distances. See Section 6.4.2 for more information about protein structure and data preparation. Tables 6.1 gives the result of 1MBC. The results of 1BPI, 1CBNa, and 2GDM delivered similar information.

To assess the two methods, we measured the relative errors $\Delta F := \hat{F} - F$ and $\Delta D := \hat{D} - D$ in the Frobenius norm, listed in the columns $\frac{\|\Delta F\|_F}{\|F\|_F}$ and $\frac{\|\Delta D\|_F}{\|D\|_F}$ in Table 6.1, respectively. The maximum relative error of distance, denoted by ϵ, is also reported. They are all measured in percent. The result is summarized as follows.

- Method 1 generally gives higher rank of $\mathcal{T}(\hat{D})$ than Method 2.
- Compared with Method 2, Method 1 generally drops $\frac{\|\Delta F\|_F}{\|F\|_F}$ by a factor of 6 or more. This is as expected, since Method 1 minimizes (6.12) to obtain the nearest EDM.
- On the other hand, the two methods generated comparable relative errors of the estimated EDM $\hat{D}$ with the embedding dimension r.
- Both methods generate distance matrices $\hat{D}$ with maximum relative distance errors usually less than 100%.

Using a local minimization procedure with our initialization methods, we can usually reach the global minimum for EDMCPs with about 60% unspecified entries or less. See Section 6.4 for more information.

Table 6.1: Percent errors in EDM initialization, protein 1MBC (153 Cα atoms).

Unspec.	*Method 1:* $Q\Lambda_+Q$				*Method 2: nearest* EDM			
Rate	$\hat{r}$	$\frac{\|\Delta F\|_F}{\|F\|_F}$	$\frac{\|\Delta D\|_F}{\|D\|_F}$	ϵ	$\hat{r}$	$\frac{\|\Delta F\|_F}{\|F\|_F}$	$\frac{\|\Delta D\|_F}{\|D\|_F}$	ϵ
5%	42	3.059	2.035	70.304	9	0.220	2.540	71.707
10%	58	4.790	4.887	82.497	10	0.299	5.656	83.148
15%	70	5.941	8.132	90.535	11	0.331	9.060	89.852
20%	73	6.619	11.828	82.618	12	0.348	12.864	82.332
25%	78	7.123	15.411	79.997	15	0.357	16.506	78.318
30%	77	7.396	18.579	78.777	16	0.351	19.713	78.663
35%	78	7.579	21.221	78.311	17	0.343	22.377	74.436
40%	78	8.456	23.909	78.744	19	0.364	25.187	79.419
45%	79	11.086	26.948	83.655	22	0.480	28.571	85.247
50%	81	16.088	31.722	87.072	24	0.730	33.821	86.233
55%	81	21.254	37.852	89.356	26	0.968	40.269	84.859
60%	81	28.314	45.773	97.294	29	1.265	48.426	96.013
65%	82	33.733	55.099	96.094	28	1.457	57.794	96.770
70%	81	42.043	63.805	95.967	30	1.647	66.194	96.794
75%	82	40.009	73.051	98.041	32	1.527	74.941	98.435
80%	82	48.165	82.098	98.076	38	1.795	83.664	98.121
85%	84	45.438	89.703	98.692	42	1.678	90.719	98.984
90%	87	33.894	94.845	99.495	52	1.249	95.405	99.182
95%	100	6.626	98.670	99.736	85	0.264	98.778	99.716

6.3.2 Dimensional Relaxation

The higher the dimension, the more free variables we have. From this point of view we have a better chance to reach the global minimum in a high-dimensional space. A global minimizer in the high-dimensional space may help us find a solution in a low-dimensional space.

The high-level description of the dimensional relaxation procedure is given in Algorithm 6.1, where α indicates the dimensional increase and β denotes the dimensional decrease. When a global minimizer is found for $\alpha = \beta$, we declare a solution in the desired dimension.

Algorithm 6.1 Dimensional relaxation.

If a global minimizer can be found in the r-dimensional space, then declare a solution and return.

for all $\alpha := 1, 2, \ldots$ **do**

  Apply a minimization procedure in the $(r+\alpha)$-dimensional space.

  if the minimizer is global **then**

    for all $\beta := 1, 2, \ldots, \alpha$ **do**

      Apply a minimization procedure in the $(r+\alpha-\beta)$-dimensional space using the information from the higher dimensional spaces.

      If the new minimizer is not global, then break the inner for loop.

      If the new minimizer is global with $\beta = \alpha$, then declare a solution and return.

    end for

  end if

end for

Figure 6.2 shows the procedure of up to 2 dimensional relaxation, with each node labeled $\alpha - \beta$, where α and β are the dimensional increase and decrease, respectively.

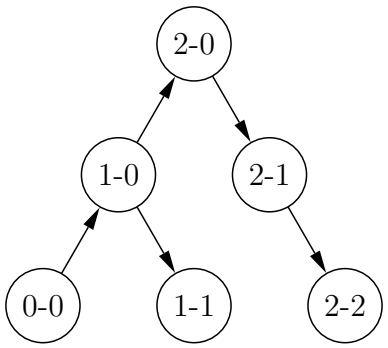

Figure 6.2: Procedure of dimensional relaxation.

Optimization in a high dimensional space is straightforward. The challenge is to convert these coordinates back to the desired dimensional space. Crippen [18] projected along the direction of the eigenvector corresponding to the smallest eigenvalue of the weighted *inertial tensor matrix* of the interatomic separation vectors. Havel [35] used four-dimensional relaxation and simulated annealing to find a rigid transformation to approximately project back to the three-dimensional space. Purisima and Scheraga [51] used the Cayley-Menger determinants to reduce the dimensionality of structure. We have explored two methods to reduce dimensions:

Method 1. An effective process involving a number of random unitary matrices.

Method 2. Rigid transformation by multiplying the orthogonal matrix from the spectral decomposition of the inertial tensor matrix.

Assume we have a global minimizer $P \in R^{n \times r'}$ in dimension $r' > r$, where r is the least embedding dimension. The task is to drop the dimension r' of the

minimizer down to r by an iterative process. More precisely, we compute another global minimizer $P_2 \in R^{n\times(r'-1)}$ using the information from P, and repeat the process until the dimension is r. We denote the column in P (i.e., the dimension) eliminated to obtain P_2 by $q = [q_1, \ldots, q_n]^T$. Then the change in the EDM is

$$\begin{aligned}
\Delta D &= \mathcal{K}(PP^T) - \mathcal{K}(P_2P_2^T) \\
&= \mathcal{K}(PP^T - P_2P_2^T) = \mathcal{K}(qq^T) \\
&= \operatorname{diag}(qq^T)e^T + e\operatorname{diag}(qq^T)^T - 2qq^T.
\end{aligned}$$

In other words, the (i, j) entry of ΔD is

$$q_i^2 + q_j^2 - 2q_iq_j = (q_i - q_j)^2 \geq 0.$$

Let $\bar{q} := \sum_{i=1}^n q_i/n$ and $\bar{q}_i := q_i - \bar{q}$. Then the sum of the elements in ΔD is

$$\begin{aligned}
\|\Delta D\|_S &= \sum_{i=1}^n\sum_{j=1}^n (q_i - q_j)^2 = \sum_{i=1}^n\sum_{j=1}^n (\bar{q}_i - \bar{q}_j)^2 \\
&= \sum_{i=1}^n\sum_{j=1}^n (\bar{q}_i^2 + \bar{q}_j^2) - 2\sum_{i=1}^n\sum_{j=1}^n \bar{q}_i\bar{q}_j \\
&= 2n\sum_{i=1}^n \bar{q}_i^2 - 2\sum_{i=1}^n \bar{q}_i \sum_{j=1}^n \bar{q}_j = 2n\sum_{i=1}^n \bar{q}_i^2 \\
&= 2n^2\,\mathrm{STD}(q)^2,
\end{aligned}$$

where $\mathrm{STD}(q)$ is the standard deviation of the elements in q. Therefore, minimization of the sum of the element changes in the EDM is equivalent to minimization of the standard deviation of the elements in a dimension (i.e., a column of P).

Lemma 6.3 states that any rigid transformation does not change the value of the objective function (6.8). Without loss of generality, we assume that the dimension to eliminate is the first column of P. Since rigid movement does not change the value of the objective function, we minimize the standard deviation

of the elements in the first column:

$$\min_{Q} \quad \mathrm{STD}(PQe_1) \tag{6.13}$$
$$\text{subject to} \quad QQ^T = I.$$

With the minimizer Q of (6.13), we drop the first column of PQ to obtain coordinates in a space of dimension one less.

The purpose is to find a good initial estimate for minimization in $(r'-1)$-dimensional space. Hence a global minimizer for (6.13) is not required. We generate a large number of random orthogonal matrices computed by the method of G. W. Stewart [60], and pick the Q that achieves the smallest value of the objective function in (6.13). An alternative is given in Algorithm 6.2, where $\mathrm{minSTD}(P)$ is the minimum of $\mathrm{STD}(Pe_j)$ for $j = 1, 2, \ldots, r'$.

Algorithm 6.2 An effective process to reduce dimension.

repeat

Generate a random orthogonal matrix Q.

if $\mathrm{minSTD}(PQ) < \mathrm{minSTD}(P)$ **then**

$P := PQ$

end if

until P is unchanged for K iterations.

{We usually set $K := 10^6$ in our experiments.}

Return P by dropping the column with minimum standard deviation.

Using this scheme with relaxation up to 2 (i.e., in up to five-dimensional space), we successfully solved the protein problems with unspecified entries up to 80% in essentially every case. See Section 6.4.2 for more information.

The inertial tensor matrix in [18] can help us find the minimizer of (6.13) as follows. Denote the dimension reduced by $q = [q_1, \ldots, q_n]^T$, which is PQe_1 in

(6.13). Let $v := Qe_1$, and therefore $q = Pv$. Then

$$\begin{aligned}
\|\Delta D\|_S &= \sum_{i=1}^{n}\sum_{j=1}^{n}(q_i - q_j)^2 = 2n\sum_{i=1}^{n} q_i^2 - 2(\sum_{i=1}^{n} q_i)^2 \\
&= 2nq^Tq - 2(q^Te)^2 \\
&= 2nv^TP^TPv - 2v^TP^Tee^TPv \\
&= 2v^T(nP^TP - P^Tee^TP)v \\
&= 2v^T(n\sum_{i=1}^{n} p_ip_i^T - \sum_{i=1}^{n} p_i\sum_{i=1}^{n} p_i^T)v \\
&= v^T(\sum_{i=1}^{n}\sum_{j=1}^{n} p_ip_i^T - p_ip_j^T - p_jp_i^T + p_jp_j^T)v \\
&= v^T(\sum_{i=1}^{n}\sum_{j=1}^{n}(p_i - p_j)(p_i - p_j)^T)v =: v^TTv,
\end{aligned}$$

where $T := \sum_{i=1}^{n}\sum_{j=1}^{n}(p_i - p_j)(p_i - p_j)^T$ is the inertial tensor matrix of the interatomic separation vectors.

Since $v = Qe_1$ is a unit vector, the minimum of $\|\Delta D\|_S$ is the smallest eigenvalue of T and the corresponding eigenvector is the minimizer v of $\|\Delta D\|_S$. We can choose any orthogonal matrix Q whose first column is the minimizer v. In practice we use Q from the spectral decomposition of $T = Q\Lambda Q^T$. This orthogonal matrix Q is particularly favored if we want to reduce multiple dimensions at a time.

Note that the minimizer of $\|\Delta D\|_S$ does not minimize the change of the objective function (6.9) or (6.11). However, it usually results in a relatively small change of the objective function. In addition, we can take the weights $H = [h_{ij}]$ in (6.9) and (6.11) into account and use the *weighted* inertial tensor matrix $\sum_{i=1}^{n}\sum_{j=1}^{n} h_{ij}(p_i - p_j)(p_i - p_j)^T$, whose eigenvector corresponding to the smallest eigenvalue minimizes the sum of the changes of in the weighted distance matrix $\|H \circ \Delta D\|_S$.

6.4 Experimental Results

We have incorporated all the modified Cholesky algorithms in Chapter 5 into the interior point methods for nonlinear optimization implemented in the OPT++ library[4], and programmed our method of dimensional relaxation in the global optimization formulation.

Our experiments used a PC with an Intel 2.93mHz CPU. Sections 6.4.1 and Section 6.4.2 give the experimental results on random matrices and protein problems, respectively.

6.4.1 Random Problems

This section presents the results of our experiments on the random partial matrices. For each matrix, we generated random points uniformly distributed in $[-1, 1]^r$ for P, computed the distance matrix $D = \mathcal{K}(PP^T)$, and randomly dropped some entries of D to form the symmetric partial matrices A. We have three sets of random matrices as follows.

1. In the first set 50% of the entries are unspecified, with the fixed embedding dimension $r = 3$ and varying number of points $n = 5, 10, \ldots, 100$.

2. The second set contains matrices with $n = 50$, fixed embedding dimension $r = 3$, and various rates of unspecified entries $x = 1\%, 5\%, 10\%, \ldots, 95\%$.

3. In the last set, the number of points is $n = 20$, the rate of unspecified entries is 25%, and the embedding dimension varies $r = 1, 2, \ldots, 19$.

[4]OPT++ is a library of nonlinear optimization algorithms written in C++ by Patty Hough, Juan Meza, and Pam Williams, Sandia National Laboratory, USA.

For each set, we experimented with the following three program formulations.

1. Unconstrained programming formulation (see Sections 4.1 and 6.2.3).

2. Program formulation with equality constraints $Pe = 0$, where $P \in R^{n \times r}$ (see Sections 4.2 and 6.2.4). The number of inequalities is the embedding dimension r.

3. Program formulation with inequality constraints (see Sections 4.3 and 6.2.5). We chose the 10 narrowest intervals by triangle inequality (6.10), resulting in 10 pairs of inequalities.

The performance depends on the modified Newton methods. We experimented with the modified Cholesky algorithms in the literature GMW81 [28, Chapter 4], SE90 [54], SE99 [55], MS79 [45], and CH98 [16], and our new ones GMW-I, GMW-II, SE-I, LTL^T-MS79, LTL^T-CH98 [24] (also see Chapter 5). In this set of random problems, we used initialization Method 1 in Section 6.3.1, and did *not* use dimensional relaxation schemes. The results are displayed in Tables 6.2–6.10, where the new methods are in boldface.

For each test, we report the number of iterations. The cases that failed due to failure in a line search are marked with an 'F'. If it did not converge within 200 iterations or reached the maximum number of function evaluations, it is marked with an 'M'. Those that converged to a local minimum are marked with a '*'.

For the CH98 and LTL^T-CH98 algorithms we increased the modification argument δ (displayed in Table 5.1) to be $\tau\eta$ (as used in the SE90 algorithm). Without this change, the algorithms failed in several cases, probably due to ill-conditioning.

Table 6.2: Number of iterations for modified Cholesky algorithms, applied to the unconstrained formulation, various numbers of points n, embedding dimension $r = 3$, and rate of unspecified entries 50%.

n	GMW			SE			LBL^T		LTL^T	
	81	**I**	**II**	90	99	**I**	MS79	CH98	**MS79**	**CH98**
5	5	5	5	8	5	5	5	5	5	5
10	16*	15*	16	54	11*	13	16*	14*	17*	17*
15	19*	19*	24	21	17	19*	17*	18	16	15
20	11	7	7	15	7	7	10	15	9	11
25	15	7	10	15	8	7	10	12	10	13
30	13	7	7	15	6	6	9	12	9	11
35	14	7	15	12	7	7	10	12	10	12
40	10	7	7	6	6	6	11	13	10	10
45	12	7	6	13	7	6	12	13	8	13
50	1F	7	9	10	6	5	10	14	11	11
55	13	7	8	9	10	7	10	17	10	16
60	2F	6	6	6	8	6	10	13	8	14
65	12	6	6	7	8	6	9	14	10	15
70	15	7	7	9	5	8	9	18	9	15
75	11	5	5	11	6	5	9	13	8	10
80	12	5	5	6	7	5	12	15	8	11
90	12	5	5	12	5	6	11	14	8	13
95	12	6	5	6	6	7	8	20	7	12
100	12	6	6	6	5	7	10	13	9	14

Table 6.3: Number of iterations for modified Cholesky algorithms, applied to the program formulation with equality constraints, various numbers of points n, embedding dimension $r = 3$, and rate of unspecified entries 50%.

n	GMW			SE			LBL^T		LTL^T	
	81	**I**	**II**	90	99	**I**	MS79	CH98	**MS79**	**CH98**
5	5	5	5	8	5	5	5	5	5	5
10	10	10	17	37	12	10	14	16*	13	17
15	16*	20*	26	25	15	13	17*	18*	21*	17
20	6	7	8	15	8	6	10	12	10	11
25	9	9	13	16	8	7	10	12	10	13
30	7	6	7	17	6	6	9	12	10	13
35	7	7	15	15	6	7	11	14	9	11
40	6	6	10	7	6	6	11	13	10	12
45	7	7	6	17	7	6	12	15	10	13
50	9	7	11	12	7	6	12	12	10	12
55	8	7	6	14	8	6	9	17	10	10
60	7	6	6	6	7	6	10	12	9	13
65	1F	1F	1F	1F	1F	1F	1F	9F	1F	10F
70	1F	1F	1F	1F	1F	1F	1F	12F	1F	7F
75	5	5	5	9	6	6	8	12	8	10
80	6	6	6	15	6	6	8	15	9	13
90	6	6	6	14	7	6	9	13	8	11
95	9	7	6	6	8	6	9	19	6	11
100	7	6	7	6	7	8	10	13	9	12

Table 6.4: Number of iterations for modified Cholesky algorithms, applied to the program formulation with inequality constraints, various numbers of points n, embedding dimension $r = 3$, and rate of unspecified entries 50%.

n	GMW			SE			LBL^T		LTL^T	
	81	**I**	**II**	90	99	**I**	MS79	CH98	**MS79**	**CH98**
5	6	6	6	5	61M	7	7	6	7	6
10	36*	18*	27	68	29	21	24*	99M	20	51F
15	25	23	43*	24*	23	16	19*	88M	17*	19
20	48F	14	16	16	17F	17	16	28F	15	7F
25	69	13	13	18	14	13	15	18	17	15
30	5F	15	14	20	15	14	17	20	17	16
35	77	18	17	182M	6F	16	21	102M	19	154M
40	5F	14	14	19	14	14	15	18	15	18
45	75	17	17	18	17	17	16	23	16	19
50	32	18	18	23	18	22	19	25	19	20
55	27	18	18	23	18	18	18	22	18	20
60	65M	20	20	27	22	20	21	25	19	22
65	52	20	21	23	24	21	20	28	20	23
70	30	20	20	23	24	20	21	26	21	22
75	24	20	20	26	21	17	21	23	20	21
80	54	18	18	22	19	18	18	23	18	19
90	30	24	24	27	24	24	23	30	22	24
95	30	23	23	25	27	23	22	28	22	25
100	105	32	32	41	38	32	33	39	31*	34

Table 6.5: Number of iterations for modified Cholesky algorithms, applied to the unconstrained formulation, number of points $n = 50$, embedding dimension $r = 3$, and various rates of unspecified entries x.

x	GMW			SE			LBL^T		LTL^T	
	81	**I**	**II**	90	99	**I**	MS79	CH98	**MS79**	**CH98**
1%	3	3	3	11	4	3	3	3	3	3
5%	4	4	3	4	3	3	4	5	3	8
10%	6	4	4	14	4	4	4	9	4	6
15%	5	4	4	14	5	4	4	9	4	7
20%	7	4	4	13	6	4	7	9	4	8
25%	9	5	5	5	7	4	5	11	5	9
30%	10	5	4	14	6	6	6	10	5	8
35%	9	6	6	15	7	5	7	10	6	8
40%	12	6	7	8	5	5	8	14	7	11
45%	10	6	6	12	6	5	9	8	10	12
50%	1F	7	9	10	6	5	10	14	11	11
55%	16	7	10	7	6	6	12	15	10	12
60%	18	7	13	10	8	6	13	17	13	14
65%	3F	9	66	7	7	7	15	16	12	14
70%	3F	16*	121	13	13	12*	34	33	32	36
75%	5F	18*	160*	38*	31*	23*	30*	27*	36*	41*
80%	1F	29*	189*	31*	23*	58*	28*	25*	22*	23*
85%	1F	26*	165*	115*	15F	30*	29*	38*	37*	43*
90%	1F	101*	200M	200M	70*	200M	1F	73M	38*	195*
95%	1F	17	200M	200M	12	11	1F	69M	1F	69M

Table 6.6: Number of iterations for modified Cholesky algorithms, applied to the program formulation with equality constraints, number of points $n = 50$, embedding dimension $r = 3$, and various rates of unspecified entries x.

x	GMW			SE			LBL^T		LTL^T	
	81	**I**	**II**	90	99	**I**	MS79	CH98	**MS79**	**CH98**
1%	3	3	3	12	3	3	3	3	3	3
5%	4	3	3	5	3	3	4	7	3	5
10%	4	4	4	14	4	4	4	6	4	7
15%	4	4	4	15	4	4	4	8	4	5
20%	5	4	4	16	4	4	7	8	4	7
25%	6	5	5	8	5	4	5	8	5	8
30%	5	5	5	15	4	4	6	11	5	8
35%	6	6	6	13	5	5	7	9	6	10
40%	7	6	6	9	5	5	8	11	7	10
45%	5	6	6	8	6	5	9	11	8	12
50%	9	7	11	12	7	6	12	12	10	12
55%	8	7	13	10	6	6	13	12	12	10
60%	10	8	16	9	6	7	12	15	14	10
65%	11	9	70	16	7	8	14	15	12	12
70%	13*	16*	122	23	12	15*	25*	26*	43	19*
75%	33*	16*	158*	32*	34*	27*	28*	28*	20*	24*
80%	17*	26*	200M	33*	19*	32*	24*	22*	21*	28*
85%	46*	26*	194*	118*	36*	35*	32*	52*	41*	29*
90%	1F	101*	200M	200M	61*	200M	2F	71M	136*	58*
95%	1F	17	200M	200M	12	11	1F	69M	1F	69M

Table 6.7: Number of iterations for modified Cholesky algorithms, applied to the program formulation with inequality constraints, number of points $n = 50$, embedding dimension $r = 3$, and various rates of unspecified entries x.

x	GMW			SE			LBL^T		LTL^T	
	81	**I**	**II**	90	99	**I**	MS79	CH98	**MS79**	**CH98**
1%	13	13	13	18	13	13	13	14	13	13
5%	14	14	14	18	14	14	14	15	14	16
10%	16	14	14	16	14	14	14	15	14	15
15%	15	14	14	17	14	14	15	16	14	16
20%	17	16	16	16	17	16	16	17	16	17
25%	19	17	17	18	18	17	17	18	17	18
30%	18	17	17	20	18	17	17	20	17	18
35%	22	17	17	21	18	17	17	19	17	18
40%	27	18	18	24	20	18	18	21	18	21
45%	25	18	18	24	18	18	18	23	17	20
50%	32	18	18	23	18	22	19	25	19	20
55%	16F	15	15	19	16	15	17	22	18	18
60%	76M	18	19	19	20	18	19	26	22	24
65%	8F	20	200M	23	24	21	21	26	20	25
70%	2F	25*	82F	176F	23*	23*	28	33*	33	114F
75%	36F	18*	129F	37F	20*	58*	34*	133F	46*	200M
80%	2F	82*	51F	31F	44*	80*	29*	200M	60*	192M
85%	24F	50*	114F	21F	27F	38*	109*	100M	90*	200M
90%	11F	128*	85F	198F	104*	110*	11F	200M	72	96F
95%	6F	20	86F	200M	74M	20	9F	9F	9F	9F

Table 6.8: Number of iterations for modified Cholesky algorithms, applied to the unconstrained formulation, number of points $n = 20$, various embedding dimensions r, and rate of unspecified entries 50%.

r	GMW			SE			LBL^T		LTL^T	
	81	**I**	**II**	90	99	**I**	MS79	CH98	**MS79**	**CH98**
1	3	3	3	3	3	3	3	3	3	3
2	4	4	4	10	6	4	5	6	4	6
3	9	7	6	18	5	5	7	8	7	8
4	13	7	7	12	7	7	9	11	9	9
5	17	8	11	40	12	10	11	13	10	10
6	16	10	11	12	10	9	12	15	12	14
7	1F	25	25	200M	29	12*	26	33	25	27
8	17	15	17	157	14	15	17	18	17	16
9	37*	47	37*	200M	32*	42*	29*	30*	34*	50
10	27*	27*	27*	200M	48*	59	47*	50*	46*	49*
11	18	20	16	200M	18	18	18	20	20	21
12	10	11	11	18	11	11	11	11	10	11
13	8	8	8	16	8	8	8	8	8	8
14	8	9	9	200M	9	9	8	9	8	9
15	8	8	8	200M	8	8	8	8	8	8
16	7	7	7	200M	7	7	7	7	7	7
17	7	8	8	200M	8	8	8	8	8	8
18	8	8	8	200M	8	8	8	8	8	8
19	9	9	8	200M	8	4F	9	9	9	9

Table 6.9: Number of iterations for modified Cholesky algorithms, applied to the program formulation with equality constraints, number of points $n = 20$, various embedding dimensions r, and rate of unspecified entries 50%.

r	GMW			SE			LBL^T		LTL^T	
	81	**I**	**II**	90	99	**I**	MS79	CH98	**MS79**	**CH98**
1	3	3	3	3	3	3	3	3	3	3
2	4	4	4	13	4	4	5	7	4	7
3	6	5	6	17	5	5	7	8	7	7
4	7	7	6	13	6	6	8	9	8	8
5	9	8	10	21	11	10	11	13	11	12
6	10	9	11	12	11	9	11	12	10	11
7	28	30	26	200M	26	24	25	33	24	27
8	15	14	18	200M	14	15	17	19	16	19
9	29*	47	33*	200M	35*	33*	28*	37*	48	36*
10	25*	25*	25*	200M	25*	25*	25*	25*	25*	25*
11	24	23	22	200M	18	19	26	21	22	21
12	11	11	11	26	11	11	11	11	11	12
13	11	11	11	17	11	11	11	11	11	11
14	11	11	11	200M	11	11	11	11	11	11
15	13	13	13	200M	13	13	13	13	13	13
16	11	11	11	200M	11	11	11	11	11	11
17	13	13	13	200M	13	13	13	13	13	13
18	13	13	13	200M	13	13	13	13	13	13
19	17	17	17	200M	17	17	17	17	17	17

Table 6.10: Number of iterations for modified Cholesky algorithms, applied to the program formulation with inequality constraints, number of points $n = 20$, various embedding dimensions r, and rate of unspecified entries 50%.

r	GMW			SE			LBL^T		LTL^T	
	81	**I**	**II**	90	99	**I**	MS79	CH98	**MS79**	**CH98**
1	19	19	19	19	19	19	19	19	19	19
2	17	16	16	25	16	16	16	20	16	18
3	14	13	13	18	14	13	13	19	14	15
4	11F	11	11	11	12	11	13	18	12	14
5	4F	13	12	32	12	14	12	26	10	16
6	43F	11	11	86	14	13	14	24	15	15
7	18F	35	22*	200M	34	31	37	45	21	23
8	40	14	14	138	14	14	16	28*	17	19
9	9F	41*	42*	200M	40*	38*	32*	52*	52	40*
10	32	39	39	200M	39	39	39	40	39	40
11	18	19	19	200M	20	19	20	21	20	19
12	14	14	14	104	14	14	14	14	14	14
13	14	16	16	15	16	16	16	15	16	15
14	15	13	13	200M	13	13	13	13	13	13
15	19	18	18	200M	18	18	18	17	16	17
16	17	16	16	200	16	16	14	15	16	15
17	16	15	15	200M	15	15	14	15	15	15
18	22	16	16	200M	13F	13F	16	16	16	16
19	19	19	19	104	19	19	19	18	19	18

The concluding remarks from this set of experiments are as follows.

1. Using the unconstrained program formulation, the required numbers of iterations were all modest for 50% unspecified entries and various $n = 5, 10, \ldots, 100$. Using the constrained program formulations, the required number of iterations moderately increased while n increased.

2. For about 65% unspecified entries or less, it usually converged to a global minimum using a local minimization procedure. For 70% unspecified entries or more, it was easily trapped by local minima.

3. The equality constraints sometimes reduced the number of iterations for convergence (e.g., $n = 10$ in Tables 6.2 and 6.3).

4. The inequality constraints usually slowed down the convergence, especially for the cases of low embedding dimension.

5. These ten modification algorithms usually performed well, except that the SE90 algorithm had difficulty for optimization in high dimensional spaces. The SE-I algorithm usually performed the best.

6. We also noted that increasing the relaxation factor μ from 0.75 to 1.0 for GMW-II algorithm can significantly improve the performance for rate of unspecified entries 65% or more. This improvement is not reflected in Tables 6.2–6.10.

6.4.2 Protein Problems

We begin by a short introduction to amino acids and protein structure, largely taken from [17, Chapter 1][63, 64].

Proteins are *amino acid* chains; each amino acid is a chemical group that contains both amine and carboxylic acid functional groups as shown in Figure 6.3, where the central carbon atom is the α carbon (Cα), the left NH_2 group is the *amino group*, the right COOH group is the *carboxyl group*, and $\mathcal{R}$ is a chemical group called the *chain residue*, specifying the type of amino acid. There are 20 different chain residues, having different chemical properties.

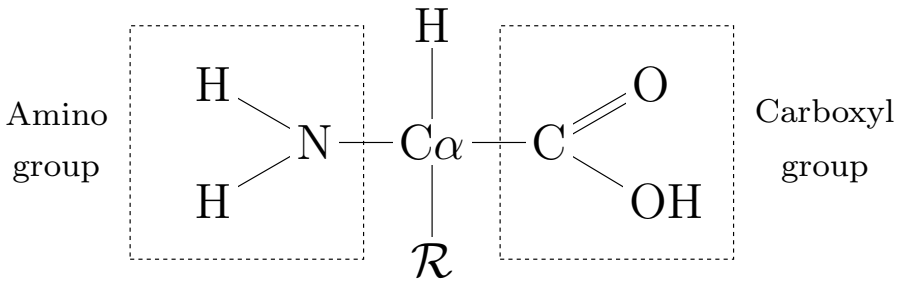

Figure 6.3: General form of an amino acid.

Two amino acids can be connected via a peptide bound, where the carboxyl group of the first amino acid is bonded to the amino group of the second. Amino acids can be linked together in varying sequences to form a huge variety of proteins, just as the letters of the alphabet can be combined in different ways to form an endless variety of words.

The smallest protein contains 40–50 amino acids[5]. A large multi-functional or structural protein can have several thousand amino acids; however, the estimated average protein number of amino acids in a protein is around 300 [64]. The amino acids fold into unique three-dimensional protein structures.

In protein structure prediction we aim at determining the three-dimensional structure of proteins. The distance information comes from the structural in-

[5]Below about 40 amino acids the term *peptide* is frequently used.

terpretation of nuclear magnetic resonance (NMR) data. The longer distances will be missing. More precisely, the distance cutoff for the so-called nuclear Overhauser enhancement (NOE) distance constraints is about 5 Å. See [67] for information about data preparation.

We experimented on structure prediction of proteins 1BPI, 1CBNa, 1MBC, and 2GDM, considering only the Cα atoms, one per amino acid ($n = 58, 46, 153$ and 153, respectively). We computed the distance matrix, denoted by D, using the data in the Protein Data Bank [8]. To simulate the real protein structure prediction problem, we dropped from 5% to 95% of the distances to get the partial matrix, retaining the shortest distances. We then tried to reconstruct the distance matrix by minimizing $f(P)$ defined in (6.9). We used Method 1 in Section 6.3.1 for EDM initialization and projection Method 2 in Section 6.3.2 for dimensional reduction.

The results of the experiments on protein 1CBNa are given in Tables 6.11–6.20. The experiments on other proteins 1BPI, 1MBC and 2GDM delivered similar relative performance. In these tables the first column lists the rates of unspecified entries. The second column indicates the dimensional relaxations in the form $\alpha-\beta$, with α the dimensional increase and β the dimensional decrease. If $\alpha = \beta = 0$, then the algorithm is a standard local minimization procedure. When $\alpha = \beta > 0$, we go back to the least embedding dimension $r = 3$. The contents of the evaluation columns are discussed in Section 6.3.1. We also list the numbers of evaluations of $f(P)$ and its gradient. An (F) indicates the failure to reach a global minimum, and a '*' before the rate means that although a global minimizer is found, it is not the protein conformation due to insufficient distance information.

Table 6.11: Protein 1CBNa, 46 Cα atoms (GMW81 algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation			# of $f$eval	# of $g$eval	Time (sec.)
		$f(P)$	$\frac{\|\Delta D\|_F}{\|D\|_F}$	ϵ			
5%	0-0	3.1e-18	2.3e-13	4.3e-12	38	11	0.34
10%	0-0	1.3e+06	0.237	0.750	21	2	0.05
	1-0	3.5e+05	0.200	0.466	21	2	0.09
	2-0	8.6e-14	9.1e-11	8.7e-10	41	40	4.62
(F)	2-1	2.2e+07	0.668	1.788	34	4	0.21
15%	0-0	1.7e+06	0.289	0.729	29	3	0.08
	1-0	4.7e+05	0.253	0.577	21	2	0.08
(F)	2-0	3.2e+05	0.243	0.576	21	2	0.15
20%	0-0	2.1e+06	0.354	0.843	21	2	0.05
	1-0	5.4e+05	0.287	0.575	29	3	0.15
(F)	2-0	3.4e+05	0.284	0.442	29	3	0.26
25%	0-0	2.1e+06	0.351	1.110	28	3	0.09
	1-0	8.1e+05	0.321	0.611	29	3	0.15
(F)	2-0	4.1e+05	0.322	0.593	21	2	0.15
30%	0-0	2.4e+06	0.437	0.730	21	2	0.05
	1-0	8.9e+05	0.367	0.672	37	4	0.23
(F)	2-0	4.7e+05	0.379	0.578	29	3	0.27
35%	0-0	1.2e+06	0.396	1.178	58	7	0.21
	1-0	9.4e+05	0.437	0.695	21	2	0.09
(F)	2-0	5.2e+05	0.417	0.890	21	2	0.14
40%	0-0	1.4e+06	0.476	1.364	60	7	0.22
	1-0	9.1e+05	0.458	0.685	37	4	0.23
(F)	2-0	5.1e+05	0.458	0.791	21	2	0.14
45%	0-0	1.7e+06	0.549	0.906	37	4	0.11
	1-0	9.7e+05	0.508	0.719	21	2	0.10
(F)	2-0	5.6e+05	0.489	0.651	21	2	0.14
50%	0-0	1.3e+06	0.615	0.898	46	5	0.16
	1-0	7.7e+05	0.557	0.709	39	4	0.22
(F)	2-0	5.1e+05	0.546	0.674	30	3	0.27
55%	0-0	1.2e+06	0.618	1.213	28	3	0.09
	1-0	7.2e+05	0.596	0.847	29	3	0.15
(F)	2-0	4.4e+05	0.577	0.609	21	2	0.15
60%	0-0	8.1e+05	0.660	1.266	28	3	0.08
	1-0	5.7e+05	0.647	0.995	29	3	0.15
(F)	2-0	3.9e+05	0.630	0.834	21	2	0.15
65%	0-0	6.2e+05	0.750	1.244	44	5	0.15
	1-0	4.8e+05	0.712	0.791	21	2	0.08
(F)	2-0	3.1e+05	0.687	0.718	21	2	0.15

Table 6.12: Protein 1CBNa, 46 Cα atoms (GMW-I algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation			# of $f$eval	# of $g$eval	Time (sec.)
		$f(P)$	$\frac{\|\Delta D\|_F}{\|D\|_F}$	ϵ			
5%	0-0	1.2e-21	4.9e-15	1.1e-13	11	8	0.23
10%	0-0	6.7e-21	1.1e-14	2.7e-13	9	8	0.23
15%	0-0	1.1e-15	4.4e-12	2.2e-11	13	9	0.26
20%	0-0	1.8e-19	7.9e-14	1.0e-12	12	10	0.29
25%	0-0	3.6e-17	1.8e-12	1.4e-11	10	10	0.29
30%	0-0	3.3e-23	1.2e-15	2.8e-14	22	13	0.40
35%	0-0	3.0e-23	2.0e-15	2.6e-14	16	12	0.37
40%	0-0	9.3e-23	2.5e-15	9.9e-15	12	12	0.36
45%	0-0	5.0e-23	3.0e-15	4.8e-14	23	14	0.42
50%	0-0	8.1e-18	1.4e-12	1.7e-11	30	20	0.61
55%	0-0	5.3e-24	8.6e-16	1.2e-14	36	21	0.64
60%	0-0	3.0e-23	1.5e-15	2.5e-14	33	22	0.68
65%	0-0	2.6e-19	4.0e-13	4.1e-12	57	30	0.93
70%	0-0	3.3e+04	0.452	0.737	40	26	0.80
	1-0	1.3e+03	0.551	0.771	54	37	2.32
	2-0	2.1e-09	2.5e-06	3.3e-06	215	144	16.20
	2-1	4.0e-10	1.0e-06	1.3e-06	165	117	7.46
	2-2	6.4e-16	3.9e-11	1.9e-10	2	2	0.04
75%	0-0	7.5e+03	0.079	0.575	51	39	1.20
	1-0	630.741	0.239	0.583	120	76	4.86
	2-0	4.3e-11	7.2e-07	1.1e-06	272	188	21.18
	2-1	6.9e-11	6.5e-07	9.7e-07	2	2	0.08
	2-2	2.0e-17	1.5e-11	3.2e-11	2	2	0.05
80%	0-0	1.0e+04	0.668	0.935	69	45	1.39
	1-0	153.201	0.390	0.810	103	71	4.53
	2-0	4.9e-04	0.022	0.054	321	201	22.66
	2-1	2.8e-09	6.2e-05	1.3e-04	313	201	12.86
	2-2	9.3e-15	3.6e-10	9.1e-10	39	30	0.93
85%	0-0	1.4e+03	0.775	0.932	57	36	1.11
	1-0	0.417	0.748	0.912	294	201	12.90
	2-0	1.3e-04	0.707	0.895	303	201	22.65
	2-1	4.1e-04	0.691	0.838	306	201	12.87
(F)	2-2	1.6e+03	0.725	0.905	58	37	1.14
*90%	0-0	2.7e-15	0.874	0.966	61	33	1.02
*95%	0-0	3.0e-24	0.957	0.993	30	23	0.70

Table 6.13: Protein 1CBNa, 46 Cα atoms (GMW-II algorithm, $\mu = 1.0$).

Unspec. Rate	Dimen. Relax.	Evaluation			# of $f$eval	# of $g$eval	Time (sec.)
		$f(P)$	$\frac{\|\Delta D\|_F}{\|D\|_F}$	ϵ			
5%	0-0	5.7e-16	3.2e-12	7.8e-11	18	9	0.27
10%	0-0	2.3e-14	2.6e-11	1.9e-10	11	8	0.23
15%	0-0	4.6e-23	8.3e-16	1.6e-14	15	10	0.31
20%	0-0	9.5e-23	1.5e-15	2.0e-14	19	10	0.29
25%	0-0	6.3e-24	2.9e-16	8.0e-15	31	14	0.42
30%	0-0	1.7e-13	8.5e-11	5.8e-10	22	11	0.33
35%	0-0	2.6e-14	3.5e-11	3.8e-10	41	15	0.45
40%	0-0	4.3e-24	3.4e-16	7.7e-15	34	16	0.49
45%	0-0	4.5e-24	5.0e-16	9.6e-15	38	17	0.52
50%	0-0	3.2e-16	1.0e-11	1.5e-10	37	18	0.56
55%	0-0	4.8e-24	6.6e-16	9.9e-15	55	26	0.81
60%	0-0	4.3e-24	6.2e-16	9.2e-15	56	25	0.77
65%	0-0	8.4e-22	1.2e-14	1.5e-13	107	43	1.35
70%	0-0	2.0e+04	0.289	0.798	105	38	1.20
	1-0	1.3e+03	0.551	0.771	102	47	3.00
	2-0	9.5e-11	5.0e-07	6.4e-07	254	159	17.86
	2-1	2.1e-11	2.2e-07	2.8e-07	14	10	0.59
	2-2	1.2e-18	1.7e-12	7.7e-12	2	2	0.04
75%	0-0	4.5e+03	0.156	0.341	122	46	1.46
	1-0	2.8e-11	4.6e-07	5.4e-07	289	181	11.62
	1-1	1.2e+04	0.152	0.715	81	36	1.13
	2-0	1.7e-10	1.3e-06	1.6e-06	332	201	22.65
	2-1	7.4e-11	7.7e-07	1.0e-06	14	10	0.59
	2-2	4.6e-17	1.6e-11	3.2e-11	2	2	0.05
80%	0-0	7.7e+03	0.726	0.898	95	37	1.15
	1-0	237.545	0.722	0.890	168	67	4.31
	2-0	1.6e-03	0.069	0.138	337	201	22.67
	2-1	9.7e-07	1.1e-03	1.8e-03	350	201	12.90
	2-2	3.3e-24	6.2e-15	2.9e-14	4	4	0.10
85%	0-0	795.591	0.743	0.948	152	69	2.17
	1-0	0.882	0.723	0.902	346	201	12.91
	2-0	2.5e-04	0.586	0.787	367	201	22.68
	2-1	1.4e-04	0.567	0.811	326	201	12.88
(F)	2-2	512.941	0.486	0.964	77	30	0.94
*90%	0-0	2.8e-20	0.902	0.956	167	70	2.20
*95%	0-0	1.2e-17	0.894	0.985	287	142	4.47

Table 6.14: Protein 1CBNa, 46 Cα atoms (SE90 algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation $f(P)$	Evaluation $\frac{\|\Delta D\|_F}{\|D\|_F}$	Evaluation ϵ	# of $f$eval	# of $g$eval	Time (sec.)
5%	0-0	4.0e-13	8.7e-11	1.3e-09	22	22	0.65
10%	0-0	6.8e-13	1.3e-10	3.7e-09	26	25	0.75
15%	0-0	9.6e-13	2.1e-10	6.0e-09	39	37	1.13
20%	0-0	1.2e-12	3.0e-10	9.0e-09	36	35	1.05
25%	0-0	5.1e-13	2.5e-10	3.9e-09	30	30	0.91
30%	0-0	6.9e-13	3.1e-10	4.1e-09	27	27	0.81
35%	0-0	1.0e-12	4.1e-10	6.7e-09	25	24	0.71
40%	0-0	6.9e-13	3.8e-10	2.5e-09	24	23	0.68
45%	0-0	1.8e-12	5.7e-10	7.8e-09	25	23	0.69
50%	0-0	1.7e-12	6.8e-10	9.4e-09	41	35	1.06
55%	0-0	4.6e-12	1.3e-09	1.4e-08	55	40	1.22
60%	0-0	2.4e-12	1.0e-09	8.9e-09	41	31	0.94
65%	0-0	7.0e-12	4.0e-09	1.9e-08	44	37	1.13
70%	0-0	3.3e+04	0.431	0.804	39	34	1.03
	1-0	1.3e+03	0.551	0.771	97	93	5.83
(F)	2-0	0.221	0.025	0.029	215	201	22.30
75%	0-0	3.5e-12	5.2e-09	1.8e-08	60	52	1.58
80%	0-0	5.9e+03	0.646	0.959	66	58	1.76
	1-0	102.948	0.611	0.853	124	90	5.68
	2-0	8.4e-03	0.080	0.257	252	201	22.29
(F)	2-1	0.011	0.062	0.251	201	201	12.61
85%	0-0	2.0e+03	0.811	0.969	69	40	1.22
	1-0	0.056	0.701	0.920	225	201	12.68
(F)	2-0	1.842	0.854	0.912	207	201	22.24
90%	0-0	7.5e-04	0.937	0.980	209	201	6.15
	1-0	3.3e-11	0.931	0.979	150	150	9.37
	1-1	0.800	0.923	0.980	202	201	6.13
	2-0	6.6e-05	0.929	0.950	201	201	22.13
	2-1	9.6e-05	0.928	0.960	201	201	12.55
(F)	2-2	1.199	0.921	0.985	201	201	6.17
*95%	0-0	8.0e-12	0.968	0.991	95	89	2.68

Table 6.15: Protein 1CBNa, 46 Cα atoms (SE99 algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation $f(P)$	Evaluation $\frac{\|\Delta D\|_F}{\|D\|_F}$	Evaluation ϵ	# of $f$eval	# of $g$eval	Time (sec.)
5%	0-0	6.6e-24	2.5e-16	8.7e-15	9	8	0.23
10%	0-0	2.3e-23	5.8e-16	8.0e-15	11	9	0.26
15%	0-0	3.6e-15	1.1e-11	1.4e-10	12	9	0.25
20%	0-0	1.8e+05	0.153	0.439	23	4	0.11
	1-0	4.9e-14	1.5e-10	6.2e-10	48	46	2.90
	1-1	1.2e-13	7.9e-11	8.9e-10	17	16	0.48
25%	0-0	4.6e-22	9.6e-15	1.2e-13	10	9	0.26
30%	0-0	8.4e-19	2.0e-13	2.3e-12	11	10	0.29
35%	0-0	6.5e-14	1.2e-10	7.8e-10	16	11	0.32
40%	0-0	9.5e+04	0.269	0.570	24	5	0.14
	1-0	1.1e-11	1.6e-08	2.4e-08	61	54	3.41
	1-1	4.2e-24	3.4e-16	9.6e-15	17	10	0.30
45%	0-0	3.6e-15	2.7e-11	1.5e-10	16	13	0.39
50%	0-0	2.7e-13	2.7e-10	3.7e-09	40	19	0.58
55%	0-0	4.1e+04	0.125	0.627	50	16	0.49
	1-0	4.4e-12	2.4e-08	3.2e-08	142	105	6.70
	1-1	2.9e-13	5.2e-10	2.1e-09	17	11	0.32
60%	0-0	3.2e+04	0.155	0.828	43	14	0.42
	1-0	1.2e-11	6.4e-08	8.4e-08	160	114	7.27
	1-1	1.5e-22	9.8e-15	1.2e-13	23	14	0.42
65%	0-0	9.9e+04	0.619	1.080	33	7	0.20
	1-0	1.6e-11	1.0e-07	1.2e-07	172	126	8.03
	1-1	2.3e+04	0.104	0.808	65	24	0.75
	2-0	6.8e-12	7.4e-08	8.6e-08	209	138	15.50
	2-1	7.3e-12	7.0e-08	8.4e-08	2	2	0.08
(F)	2-2	2.3e+04	0.104	0.808	65	24	0.74
70%	0-0	3.3e+04	0.431	0.804	29	24	0.72
	1-0	1.3e+03	0.551	0.771	79	47	2.94
	2-0	2.2e-11	2.5e-07	3.4e-07	245	169	19.00
	2-1	1.9e-12	6.4e-09	1.1e-08	20	11	0.66
	2-2	3.8e-19	3.9e-13	4.7e-13	2	2	0.04
75%	0-0	4.6e-15	1.1e-10	3.2e-10	61	35	1.09
80%	0-0	5.9e+03	0.646	0.959	100	48	1.51
	1-0	103.603	0.602	0.870	118	69	4.37
	2-0	1.4e-04	0.017	0.068	312	201	22.56
	2-1	1.6e-10	1.1e-05	1.6e-05	336	183	11.74
	2-2	1.6e-12	5.9e-09	1.1e-08	28	5	0.16

Table 6.16: Protein 1CBNa, 46 Cα atoms (SE-I algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation $f(P)$	Evaluation $\frac{\|\Delta D\|_F}{\|D\|_F}$	Evaluation ϵ	# of $f$eval	# of $g$eval	Time (sec.)
5%	0-0	7.2e-15	8.6e-12	9.7e-11	8	7	0.20
10%	0-0	6.0e-24	2.5e-16	7.7e-15	9	8	0.23
15%	0-0	6.4e-24	2.6e-16	8.0e-15	10	9	0.26
20%	0-0	1.1e-17	4.8e-13	7.7e-12	9	9	0.26
25%	0-0	6.5e-14	5.8e-11	1.0e-09	12	9	0.26
30%	0-0	8.2e-19	4.6e-13	5.4e-12	15	11	0.32
35%	0-0	2.6e-19	1.1e-13	4.9e-13	10	10	0.28
40%	0-0	1.5e-19	2.2e-13	1.7e-12	11	11	0.32
45%	0-0	4.2e-24	5.2e-16	8.7e-15	14	13	0.39
50%	0-0	7.8e-18	1.3e-12	1.4e-11	27	19	0.58
55%	0-0	4.6e-16	1.2e-11	1.2e-10	32	21	0.64
60%	0-0	3.5e-21	3.4e-14	2.0e-13	30	20	0.60
65%	0-0	4.7e-17	5.5e-12	5.7e-11	59	29	0.89
70%	0-0	3.3e+04	0.431	0.804	26	22	0.67
	1-0	1.3e+03	0.551	0.771	58	43	2.70
	2-0	2.7e-11	2.8e-07	3.7e-07	233	162	18.21
	2-1	3.1e-11	2.6e-07	3.6e-07	2	2	0.09
	2-2	8.4e-22	3.9e-14	2.7e-13	28	21	0.62
75%	0-0	7.5e+03	0.079	0.575	79	44	1.38
	1-0	630.741	0.239	0.583	119	74	4.71
	2-0	3.2e-10	1.9e-06	2.6e-06	280	192	21.60
	2-1	6.0e-11	7.0e-07	8.7e-07	24	14	0.85
	2-2	3.0e-17	1.4e-11	2.8e-11	2	2	0.04
80%	0-0	5.5e+03	0.635	0.955	94	46	1.44
	1-0	34.643	0.622	0.913	152	95	6.06
	2-0	1.1e-04	9.5e-03	0.020	307	201	22.62
	2-1	4.7e-09	8.6e-05	1.9e-04	334	201	12.89
	2-2	1.7e-18	4.8e-12	2.1e-11	3	3	0.07
85%	0-0	2.6e+03	0.789	0.965	51	34	1.04
	1-0	1.7e-03	0.610	0.900	294	201	12.81
	1-1	848.359	0.648	0.905	17	14	0.42
	2-0	9.0e-04	0.665	0.818	299	201	22.59
	2-1	6.6e-03	0.681	0.936	322	201	12.86
(F)	2-2	615.609	0.532	0.943	45	28	0.86
*90%	0-0	1.1e-24	0.918	0.964	32	23	0.70
*95%	0-0	2.9e-14	0.967	0.995	19	14	0.42

Table 6.17: Protein 1CBNa, 46 Cα atoms (MS79 algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation $f(P)$	Evaluation $\frac{\|\Delta D\|_F}{\|D\|_F}$	Evaluation ϵ	# of $f$eval	# of $g$eval	Time (sec.)
5%	0-0	1.3e-13	3.8e-11	9.1e-10	20	9	0.38
10%	0-0	4.5e-16	2.6e-12	5.1e-11	38	12	0.53
15%	0-0	1.0e-23	3.3e-16	8.0e-15	43	14	0.61
20%	0-0	3.4e-18	5.7e-13	1.1e-11	45	14	0.61
25%	0-0	2.2e-23	5.4e-16	1.2e-14	43	14	0.62
30%	0-0	2.4e-21	1.6e-14	3.0e-13	41	14	0.62
35%	0-0	5.3e-14	6.5e-11	1.3e-09	63	18	0.80
40%	0-0	1.6e-18	5.5e-13	9.5e-12	51	16	0.71
45%	0-0	4.1e-11	2.3e-09	1.5e-08	53	16	0.71
50%	0-0	1.4e-22	1.9e-15	3.7e-14	90	24	1.09
55%	0-0	2.9e-19	3.1e-13	4.5e-12	84	30	1.35
60%	0-0	5.3e-22	6.5e-15	7.2e-14	88	25	1.12
65%	0-0	3.7e-10	2.3e-08	1.2e-07	104	30	1.36
70%	0-0	1.6e-11	4.8e-09	1.8e-08	129	45	2.05
75%	0-0	7.5e+03	0.079	0.575	141	50	2.28
	1-0	411.631	0.294	0.554	157	67	6.66
	2-0	3.2e-10	1.8e-06	2.3e-06	352	198	37.03
	2-1	5.4e-10	2.1e-06	2.7e-06	305	162	16.36
(F)	2-2	7.5e+03	0.079	0.575	98	35	1.58
80%	0-0	5.7e+03	0.661	0.931	109	32	1.46
	1-0	206.403	0.509	0.785	286	136	13.53
	2-0	1.1e-03	0.056	0.092	403	201	37.44
(F)	2-1	7.788	0.531	0.814	164	74	7.34
85%	0-0	1.4e+03	0.742	0.948	149	40	1.77
	1-0	1.3e-03	0.526	0.748	359	201	20.19
	1-1	573.179	0.557	0.932	99	29	1.32
	2-0	3.3e-05	0.677	0.852	376	201	37.55
	2-1	4.0e-03	0.581	0.894	332	201	20.20
(F)	2-2	1.0e+03	0.751	0.977	64	30	1.35
*90%	0-0	2.3e-15	0.649	0.923	125	36	1.64
*95%	0-0	4.8e-14	0.660	2.570	129	34	1.57

Table 6.18: Protein 1CBNa, 46 Cα atoms (CH98 algorithm).

Unspec.	Dimen.	Evaluation			# of	# of	Time
Rate	Relax.	$f(P)$	$\frac{\|\Delta D\|_F}{\|D\|_F}$	ϵ	$f$eval	$g$eval	(sec.)
5%	0-0	8.9e+05	0.161	0.702	21	2	0.06
	1-0	4.8e-07	2.2e-07	1.9e-06	144	137	13.78
	1-1	1.6e-14	1.2e-11	2.5e-10	2	2	0.06
10%	0-0	1.3e+06	0.216	0.743	39	4	0.16
	1-0	3.5e+05	0.200	0.466	21	2	0.12
	2-0	5.7e-07	2.6e-07	3.0e-06	158	157	29.41
	2-1	2.1e-07	1.0e-07	2.2e-06	3	3	0.21
	2-2	3.2e-16	1.9e-12	5.5e-11	2	2	0.05
15%	0-0	1.6e+06	0.289	0.697	39	4	0.17
	1-0	4.7e+05	0.253	0.577	21	2	0.12
(F)	2-0	3.2e+05	0.243	0.576	21	2	0.20
20%	0-0	2.1e+06	0.354	0.843	21	2	0.06
	1-0	6.1e+05	0.308	0.567	21	2	0.12
(F)	2-0	3.4e+05	0.292	0.442	21	2	0.21
25%	0-0	2.3e+06	0.391	0.814	21	2	0.07
	1-0	8.5e+05	0.349	0.718	21	2	0.12
(F)	2-0	4.1e+05	0.322	0.593	21	2	0.20
30%	0-0	2.4e+06	0.437	0.730	21	2	0.07
	1-0	1.0e+06	0.392	0.637	21	2	0.12
(F)	2-0	4.8e+05	0.368	0.547	21	2	0.21
35%	0-0	2.3e+06	0.488	0.839	21	2	0.06
	1-0	9.2e+05	0.434	0.713	31	3	0.22
(F)	2-0	4.9e+05	0.417	0.875	30	3	0.41
40%	0-0	2.0e+06	0.521	0.823	21	2	0.06
	1-0	8.6e+05	0.438	0.793	29	3	0.22
(F)	2-0	5.1e+05	0.458	0.791	21	2	0.22
45%	0-0	1.9e+06	0.555	0.774	21	2	0.06
	1-0	9.7e+05	0.508	0.719	21	2	0.12
(F)	2-0	5.1e+05	0.479	0.719	30	3	0.41
50%	0-0	1.6e+06	0.603	0.928	21	2	0.07
	1-0	9.0e+05	0.565	0.689	21	2	0.11
(F)	2-0	5.3e+05	0.545	0.644	21	2	0.20
55%	0-0	1.3e+06	0.637	0.843	21	2	0.07
	1-0	7.9e+05	0.597	0.763	21	2	0.11
(F)	2-0	4.4e+05	0.577	0.609	21	2	0.21
60%	0-0	1.0e+06	0.694	0.872	21	2	0.06
	1-0	6.0e+05	0.650	0.869	21	2	0.12
(F)	2-0	3.9e+05	0.630	0.834	21	2	0.20

Table 6.19: Protein 1CBNa, 46 Cα atoms (LTL^T-MS79 algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation $f(P)$	Evaluation $\frac{\|\Delta D\|_F}{\|D\|_F}$	Evaluation ϵ	# of $f$eval	# of $g$eval	Time (sec.)
5%	0-0	8.1e-20	3.5e-14	7.8e-13	17	9	0.21
10%	0-0	8.7e-24	2.9e-16	1.2e-14	29	12	0.29
15%	0-0	1.0e-18	2.0e-13	2.5e-12	23	11	0.27
20%	0-0	8.0e-24	3.2e-16	8.0e-15	32	13	0.31
25%	0-0	2.6e-21	1.4e-14	2.7e-13	31	12	0.29
30%	0-0	8.6e-14	8.0e-11	1.9e-09	30	12	0.30
35%	0-0	2.9e-17	2.3e-12	2.3e-11	48	16	0.39
40%	0-0	2.0e-20	2.0e-14	2.8e-13	55	18	0.45
45%	0-0	2.5e-20	7.7e-14	1.1e-12	48	16	0.39
50%	0-0	9.4e-18	1.7e-12	2.6e-11	77	27	0.67
55%	0-0	8.0e-18	1.6e-12	2.4e-11	97	30	0.76
60%	0-0	9.3e-11	6.7e-09	4.7e-08	77	26	0.65
65%	0-0	8.1e-13	1.0e-09	5.3e-09	57	23	0.57
70%	0-0	1.6e-20	1.1e-13	2.9e-13	119	46	1.15
75%	0-0	8.3e+03	0.532	0.852	74	27	0.68
	1-0	411.631	0.294	0.554	142	66	3.31
	2-0	2.1e-09	4.9e-06	6.6e-06	308	176	14.90
	2-1	5.3e-10	1.8e-06	2.3e-06	210	117	5.83
(F)	2-2	1.2e+04	0.190	0.713	99	37	0.93
80%	0-0	5.9e+03	0.569	0.890	103	36	0.91
	1-0	35.327	0.683	0.905	227	117	5.86
	2-0	4.0e-04	0.025	0.037	358	201	17.03
(F)	2-1	608.693	0.405	0.712	166	92	4.59
85%	0-0	1.9e+03	0.825	0.919	107	35	0.88
	1-0	0.012	0.579	0.889	358	201	10.07
	2-0	1.7e-04	0.681	0.879	351	201	16.98
	2-1	7.5e-04	0.636	0.865	347	201	10.05
(F)	2-2	923.705	0.697	0.884	103	37	0.92
*90%	0-0	1.2e-21	0.824	0.951	99	29	0.73
*95%	0-0	3.4e-16	0.859	2.477	125	30	0.78

Table 6.20: Protein 1CBNa, 46 Cα atoms (LTL^T-CH98 algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation $f(P)$	Evaluation $\frac{\|\Delta D\|_F}{\|D\|_F}$	Evaluation ϵ	# of $f$eval	# of $g$eval	Time (sec.)
5%	0-0	6.3e-24	2.4e-16	9.6e-15	38	11	0.27
10%	0-0	1.4e-14	1.5e-11	2.1e-10	64	14	0.36
15%	0-0	6.1e-24	2.5e-16	8.0e-15	63	15	0.37
20%	0-0	3.8e-20	4.3e-14	5.3e-13	68	15	0.38
25%	0-0	2.3e-19	7.9e-14	1.5e-12	60	15	0.38
30%	0-0	2.5e-18	3.6e-13	4.9e-12	77	17	0.44
35%	0-0	2.5e-20	5.8e-14	5.0e-13	92	20	0.52
40%	0-0	2.0e+06	0.521	0.823	21	2	0.04
	1-0	6.8e-07	3.8e-06	4.7e-06	282	198	9.85
	1-1	2.4e-13	2.9e-10	1.3e-09	2	2	0.03
45%	0-0	1.9e+06	0.555	0.774	21	2	0.04
	1-0	9.7e+05	0.508	0.719	21	2	0.07
	2-0	5.3e-07	3.3e-06	4.5e-06	289	161	13.64
	2-1	3.0e-07	2.7e-06	4.2e-06	4	4	0.17
	2-2	1.9e-13	2.4e-10	2.1e-09	2	2	0.04
50%	0-0	1.6e+06	0.603	0.928	21	2	0.05
	1-0	9.0e+05	0.565	0.689	21	2	0.06
(F)	2-0	5.3e+05	0.545	0.644	21	2	0.12
55%	0-0	1.3e+06	0.637	0.843	21	2	0.05
	1-0	7.9e+05	0.597	0.763	21	2	0.07
(F)	2-0	4.4e+05	0.577	0.609	21	2	0.11
60%	0-0	1.0e+06	0.694	0.872	21	2	0.04
	1-0	6.0e+05	0.650	0.869	21	2	0.07
(F)	2-0	3.9e+05	0.630	0.834	21	2	0.11
65%	0-0	8.0e+05	0.761	0.928	21	2	0.04
	1-0	4.8e+05	0.712	0.791	21	2	0.07
	2-0	5.3e-07	1.7e-05	2.0e-05	325	193	16.36
	2-1	6.8e-08	8.0e-07	1.5e-06	3	3	0.12
	2-2	5.0e-16	2.3e-11	9.0e-11	2	2	0.04
70%	0-0	5.1e+05	0.789	0.873	29	3	0.07
	1-0	4.1e+05	0.764	0.821	21	2	0.07
(F)	2-0	1.9e+05	0.714	0.730	49	5	0.39
75%	0-0	3.2e+05	0.853	0.954	39	4	0.10
	1-0	2.8e+05	0.827	0.907	30	3	0.13
(F)	2-0	1.9e+05	0.801	0.797	21	2	0.10
80%	0-0	2.3e+05	0.908	0.947	30	3	0.07
	1-0	2.0e+05	0.889	0.931	21	2	0.07
(F)	2-0	1.6e+05	0.867	0.832	21	2	0.11

Due to the limit of page length, we list the rates of unspecified entries up to 65%, 60%, 60%, and 80%, for GMW81, SE99, CH98, and LTL^T-CH98 algorithms in Tables 6.11, 6.15, 6.18, and 6.20, respectively. All those unlisted did not get the correct protein conformations.

The GMW81 algorithm was less than useful. The GMW-II algorithm did not function well, either. However, increasing the relaxation factor μ from 0.75 to 1.0 made GMW-II a useful algorithm. The SE99 algorithm requires dimensional relaxation in a few cases with the rates of unspecified entries less than 50%. The difficulty of SE90 algorithm was present for experiments on proteins 1MBC and 2GDM.

Both the CH98 and LTL^T-CH98 algorithms showed difficulties. Increasing modification tolerance parameter δ to be $\tau\eta$ significantly improved the performance in the experiments on random problems in Section 6.4.1. However, the improvement by this change was very minor for protein problems (see Tables 6.18 and 6.20).

The MS79 and our GMW-I, SE-I, and LTL^T-MS79 algorithms usually worked well. Note that these algorithms are all of Type I. The only Type-I algorithm that had difficulty was GMW81, which is the only modified LDL^T algorithm that cannot achieve $O(n)$ modification (see Sections 5.1 and 5.2).

We also give the results of the SE-I algorithm applied to the 1BPI, 1MBC, and 2GDM protein problems ($n = 58, 153, 153$) in Tables 6.21–6.23, where we used Method 2 in Section 6.3.1 for EDM initialization and projection Method 1 in Section 6.3.2 for dimensional reduction.

Three remarks for algorithms GMW-I, GMW-II ($\mu = 1.0$), SE-I, MS79 and LTL^T-MS79 are given as follows.

Table 6.21: Protein 1BPI, 58 Cα atoms (SE-I algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation $f(P)$	Evaluation $\frac{\|\Delta D\|_F}{\|D\|_F}$	Evaluation ϵ	# of $f$eval	# of $g$eval	Time (sec.)
5%	0-0	8.7e-22	5.2e-30	8.2e-14	8	8	0.42
10%	0-0	1.3e-18	1.0e-26	2.3e-12	12	8	0.42
15%	0-0	1.1e-22	4.6e-31	2.4e-14	13	10	0.53
20%	0-0	2.9e-22	1.3e-30	2.4e-14	9	8	0.41
25%	0-0	9.2e-23	5.1e-31	2.2e-14	15	10	0.53
30%	0-0	2.3e-22	2.7e-30	2.5e-14	13	11	0.54
35%	0-0	3.1e-14	6.3e-22	2.3e-10	17	11	0.58
40%	0-0	1.1e-16	5.4e-24	2.1e-11	18	12	0.64
45%	0-0	7.3e-23	8.8e-31	2.8e-14	25	16	0.87
50%	0-0	2.6e-17	1.7e-24	1.0e-11	31	21	1.14
55%	0-0	8.2e-20	7.6e-27	6.0e-13	44	24	1.32
60%	0-0	4.5e-15	8.1e-22	2.2e-10	47	25	1.38
65%	0-0	3.8e+04	0.048	0.682	61	29	1.62
	1-0	6.2e-11	2.6e-14	2.3e-07	200	138	16.11
	1-1	9.3e-14	2.3e-20	2.3e-09	3	3	0.14
70%	0-0	2.2e+04	0.077	0.872	49	31	1.71
	1-0	6.3e-11	5.7e-14	3.4e-07	233	162	18.85
	1-1	6.3e-13	9.3e-20	4.3e-09	3	3	0.13
75%	0-0	6.7e+04	0.584	0.935	35	24	1.30
	1-0	3.8e-10	1.4e-12	1.7e-06	294	198	23.15
	1-1	7.9e-21	1.5e-27	4.2e-13	3	3	0.13
80%	0-0	2.9e+04	0.448	0.970	57	36	1.99
	1-0	1.6e+03	0.384	0.865	37	31	3.55
	2-0	9.1e-07	1.7e-05	0.042	302	198	42.94
	2-1	9.4e-11	1.9e-05	0.042	129	82	9.56
	2-2	5.7e-14	4.4e-18	1.8e-08	13	9	0.48
*85%	0-0	9.2e+03	0.748	0.969	45	32	1.75
	1-0	108.004	0.514	0.915	80	62	7.18
	2-0	3.7e-03	5.7e-03	0.256	298	198	42.88
	2-1	4.3e-08	1.1e-03	0.281	338	198	23.23
	2-2	7.5e-12	1.8e-03	0.414	6	6	0.30
90%	0-0	587.062	0.840	0.946	57	36	1.99
	1-0	1.2e-11	0.837	0.959	261	178	20.78
	1-1	30.411	0.744	0.950	160	96	5.41
	2-0	1.6e-11	0.803	0.927	245	173	37.44
	2-1	9.8e-12	0.813	0.966	127	81	9.45
(F)	2-2	42.886	0.740	1.124	142	90	5.07
*95%	0-0	2.6e-26	0.928	0.997	18	15	0.80

Table 6.22: Protein 1MBC, 153 Cα atoms (SE-I algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation $f(P)$	Evaluation $\frac{\|\Delta D\|_F}{\|D\|_F}$	Evaluation ϵ	# of $f$eval	# of $g$eval	Time (sec.)
5%	0-0	5.4e-22	7.9e-32	2.3e-14	7	7	5.45
10%	0-0	2.3e-20	7.6e-30	1.5e-13	8	8	6.30
15%	0-0	1.9e-16	7.3e-26	1.1e-11	8	8	6.29
20%	0-0	1.9e-21	7.4e-31	3.5e-14	11	9	7.14
25%	0-0	4.6e-22	9.4e-32	2.3e-14	19	11	8.93
30%	0-0	4.0e-22	9.1e-32	2.3e-14	11	10	8.03
35%	0-0	1.2e-16	1.3e-25	1.9e-11	15	10	8.12
40%	0-0	1.7e-17	2.0e-26	5.2e-12	12	10	8.00
45%	0-0	3.2e-22	1.1e-31	2.3e-14	11	10	8.01
50%	0-0	2.9e-22	1.2e-31	2.0e-14	19	16	13.17
55%	0-0	2.6e-22	1.2e-31	2.3e-14	24	18	14.92
60%	0-0	2.6e+06	0.047	2.602	34	21	17.52
	1-0	3.4e-12	4.7e-18	4.2e-09	148	106	225.00
	1-1	9.5e-16	2.9e-24	3.6e-11	3	3	1.82
65%	0-0	4.5e+06	0.196	1.200	66	33	27.99
	1-0	4.4e-11	1.4e-16	2.2e-08	92	76	159.46
	1-1	9.9e-15	4.3e-23	1.4e-10	3	3	1.83
70%	0-0	2.6e+06	0.213	0.962	105	55	47.09
	1-0	4.8e-11	3.4e-16	3.3e-08	179	127	267.79
	1-1	1.3e-19	8.4e-28	5.8e-13	3	3	1.83
75%	0-0	3.8e+06	0.413	0.975	63	33	27.87
	1-0	2.0e-11	3.7e-16	3.3e-08	216	149	317.07
	1-1	7.8e-14	7.6e-22	5.9e-10	3	3	1.82
80%	0-0	4.2e+05	0.038	1.281	96	52	44.26
	1-0	3.3e-11	1.4e-15	6.4e-08	241	164	348.08
	1-1	1.8e-17	2.8e-25	7.6e-12	3	3	1.83
85%	0-0	2.3e+05	0.152	1.164	112	66	56.43
	1-0	1.1e-10	1.6e-14	2.5e-07	268	184	389.25
	1-1	9.1e-23	7.6e-31	2.3e-14	3	3	1.82
90%	0-0	1.4e+05	0.158	0.948	145	74	63.50
	1-0	1.4e+04	0.179	0.913	126	79	167.29
	2-0	1.1e-07	1.4e-10	2.4e-05	301	198	835.47
	2-1	6.3e-10	2.0e-13	8.3e-07	38	26	53.23
	2-2	6.0e-22	4.3e-29	1.8e-13	3	3	1.82
95%	0-0	1.3e+04	0.835	0.979	127	75	64.26
	1-0	1.745	0.652	1.062	314	198	418.23
(F)	2-0	0.170	0.407	1.655	312	198	835.24

Table 6.23: Protein 2GDM, 153 Cα atoms (SE-I algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation			# of $f$eval	# of $g$eval	Time (sec.)
		$f(P)$	$\frac{\|\Delta D\|_F}{\|D\|_F}$	ϵ			
5%	0-0	1.1e-20	1.6e-30	9.3e-14	7	7	5.44
10%	0-0	4.6e-17	1.2e-26	6.1e-12	8	7	5.42
15%	0-0	1.0e-20	1.6e-30	9.3e-14	9	8	6.29
20%	0-0	1.1e-19	5.2e-29	3.7e-13	8	8	6.22
25%	0-0	1.4e-17	5.6e-27	3.5e-12	9	8	6.28
30%	0-0	3.4e-19	1.7e-28	6.7e-13	13	10	8.04
35%	0-0	6.0e-20	5.7e-29	2.6e-13	13	10	8.00
40%	0-0	2.8e-16	3.0e-25	1.1e-11	12	11	8.85
45%	0-0	1.6e-20	1.7e-29	9.1e-14	10	10	8.02
50%	0-0	2.3e-19	5.2e-28	6.7e-13	16	14	11.44
55%	0-0	6.5e-21	3.1e-30	1.1e-13	26	18	14.91
60%	0-0	6.0e-15	1.5e-23	1.6e-10	53	31	26.26
65%	0-0	5.1e+06	0.153	1.486	55	33	28.03
	1-0	1.1e-11	3.4e-17	1.2e-08	169	119	253.43
	1-1	2.2e-18	1.1e-26	1.9e-12	3	3	1.83
70%	0-0	5.4e+06	0.245	1.597	63	36	30.44
	1-0	2.3e-10	1.8e-15	7.4e-08	131	90	190.51
	1-1	7.2e-16	4.3e-24	4.0e-11	3	3	1.83
75%	0-0	4.2e+06	0.409	1.314	41	22	18.44
	1-0	3.7e-11	7.6e-16	4.8e-08	214	148	312.16
	1-1	1.2e-17	1.4e-25	3.2e-12	3	3	1.82
80%	0-0	3.8e+05	0.103	1.065	103	55	47.42
	1-0	1.1e+05	0.269	0.968	37	31	64.02
	2-0	1.6e-09	6.2e-14	4.7e-07	209	148	623.25
	2-1	7.7e-12	2.3e-16	3.3e-08	33	25	50.99
	2-2	8.8e-17	1.8e-24	2.8e-11	3	3	1.84
85%	0-0	5.8e+05	0.279	1.524	63	36	30.51
	1-0	2.9e-10	6.6e-14	4.8e-07	260	176	372.10
	1-1	6.3e-13	1.9e-20	3.2e-09	3	3	1.83
90%	0-0	1.9e+05	0.569	0.979	85	50	42.54
	1-0	1.8e+04	0.253	0.946	148	83	174.02
	2-0	5.6e-06	1.6e-08	2.6e-04	300	198	836.01
	2-1	2.3e-10	6.6e-13	2.0e-06	152	100	209.67
	2-2	7.7e-19	1.5e-25	2.1e-12	3	3	1.84
95%	0-0	1.5e+04	0.757	0.987	149	82	70.43
	1-0	2.230	0.512	1.365	334	198	417.91
(F)	2-0	0.293	0.490	1.776	303	198	834.75

1. For about 55%–60% unspecified entries or less, we can usually reach the global minimum without dimensional relaxation.

2. For unspecified entries 60%-80%, we can usually find the solution with dimensional relaxation 1 or 2.

3. Difficulties were still present for some cases with unspecified entries 85% or higher. Even if a global minimizer is found, it is not reliable because insufficient distance information may result in multiple solutions.

We also experimented on all 460 atoms of 1BPI and all 1244 atoms of 1MBC, by dropping the longest distance information $5\%, 10\%, \ldots, 95\%$, where we used Method 1 in Section 6.3.1 for EDM initialization and projection Method 1 in Section 6.3.2 for dimensional reduction. The modification algorithm to generate descent directions is still SE-I. The results are displayed in Tables 6.24 and 6.25, respectively. For rates of unspecified entries 65% or less, we reached the global minimum using a local minimization procedure in all cases. Using dimensional relaxation, we also solved all the problems with up to 95% unspecified entries.

In real problems, the measured distances may have errors. To show how sensitive the solution is to perturbation, we did perturbation experiments as follows. For each rate of unspecified entries $10\%, 20\%, \ldots, 90\%$, we added uniformly distributed perturbations to the distances, with each perturbation rate uniformly distributed in $[-p, p]$ for $p = 5\%, 10\%, \ldots, 95\%$. We then solved the EDMCP with perturbations, and measured $\frac{\|\Delta D\|_S}{\|D\|_S}$, where D is the actual EDM and ΔD is the difference between the computed EDM and D. See Figure 6.4 for the result of protein 1CBNa. The higher the rate of unspecified entries, the more sensitive to the perturbation the solution.

Table 6.24: Protein 1BPI, 460 atoms (SE-I algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation			# of $f$eval	# of $g$eval	Time
		$f(P)$	$\frac{\|\Delta D\|_F}{\|D\|_F}$	ϵ			
5%	0-0	8.2e-21	6.5e-16	3.0e-13	7	7	2m37s
10%	0-0	4.7e-15	6.7e-13	1.6e-10	18	10	3m54s
15%	0-0	7.6e-21	6.8e-16	3.0e-13	12	10	3m54s
20%	0-0	8.9e-16	3.6e-13	5.2e-11	9	9	3m30s
25%	0-0	6.5e-21	6.7e-16	3.0e-13	12	10	3m56s
30%	0-0	2.3e-15	8.4e-13	2.5e-10	14	11	4m22s
35%	0-0	6.6e-15	1.9e-12	3.4e-10	17	15	6m08s
40%	0-0	5.3e-21	6.9e-16	2.9e-13	27	17	6m57s
45%	0-0	7.3e-21	2.2e-15	3.0e-13	22	20	8m14s
50%	0-0	5.6e-21	1.4e-15	3.0e-13	30	22	9m04s
55%	0-0	8.0e-19	5.2e-14	3.1e-12	65	43	18m08s
60%	0-0	1.9e-12	1.3e-10	6.3e-09	69	40	16m47s
65%	0-0	3.1e-19	6.2e-14	2.1e-12	95	46	19m20s
70%	0-0	8.1e-21	1.5e-15	2.9e-13	105	47	19m45s
75%	0-0	1.8e-20	2.0e-14	8.1e-13	109	58	24m30s
80%	0-0	1.9e-11	1.2e-09	3.6e-08	119	66	27m52s
85%	0-0	5.0e+05	0.059	1.569	181	94	39m53s
	1-0	2.0e+05	0.119	0.893	94	60	19m40s
	2-0	3.8e-09	8.7e-13	1.6e-06	290	201	6h33m42s
	2-1	2.8e-11	1.1e-15	7.0e-08	33	16	20m01s
	2-2	1.2e-19	5.0e-27	5.8e-12	3	3	51s
90%	0-0	5.9e+05	0.512	2.556	109	68	28m43s
	1-0	5.3e+04	0.369	0.933	139	90	1h01m51s
	2-0	2.9e-06	4.2e-09	1.1e-04	296	201	6h44m24s
	2-1	1.2e-09	1.6e-12	2.4e-06	91	65	1h04m07s
	2-2	5.3e-14	2.0e-18	1.7e-08	3	3	1m08s
95%	0-0	1.2e+05	0.622	1.208	166	91	38m28s
	1-0	1.0e+04	0.498	0.961	214	116	1h55m48s
	2-0	4.8e-03	2.4e-04	0.033	317	201	6h33m20s
	2-1	2.7e-08	4.6e-09	9.4e-04	320	201	3h21m17s
	2-2	1.4e-12	2.0e-15	6.1e-07	5	5	1m42s

Table 6.25: Protein 1MBC, 1244 atoms (SE-I algorithm).

Unspec. Rate	Dimen. Relax.	Evaluation $f(P)$	Evaluation $\frac{\|\Delta D\|_F}{\|D\|_F}$	Evaluation ϵ	# of $f$eval	# of $g$eval	Time
5%	0-0	3.9e-20	8.3e-32	3.0e-12	7	7	54m07s
10%	0-0	3.8e-20	8.9e-32	3.0e-12	13	9	1h11m38s
15%	0-0	3.7e-20	9.1e-32	3.0e-12	24	11	1h29m25s
20%	0-0	4.9e-18	2.7e-29	6.0e-12	9	8	1h03m00s
25%	0-0	2.1e-15	1.4e-26	1.4e-11	8	8	1h03m20s
30%	0-0	3.0e-20	9.2e-32	3.0e-12	23	12	1h38m22s
35%	0-0	3.5e-20	1.2e-31	6.0e-12	20	13	1h47m07s
40%	0-0	2.7e-20	1.0e-31	3.0e-12	21	13	1h48m28s
45%	0-0	3.4e-20	1.3e-31	3.0e-12	20	12	1h38m38s
50%	0-0	2.5e-16	5.1e-27	1.2e-11	13	12	1h38m38s
55%	0-0	2.0e-20	1.3e-31	3.0e-12	35	31	4h28m57s
60%	0-0	1.8e-20	1.1e-31	4.5e-12	68	47	6h51m45s
65%	0-0	5.3e-15	6.2e-25	2.6e-10	101	69	10h04m42s
70%	0-0	1.4e+08	0.092	4.362	104	82	12h01m30s
	1-0	3.0e-11	1.9e-18	3.1e-09	170	132	2d02h57m55s
	1-1	3.6e-19	5.0e-29	4.5e-12	3	3	17m30s
75%	0-0	8.6e+07	0.101	2.718	110	75	10h56m57s
	1-0	4.9e-11	7.7e-18	6.0e-09	188	140	2d05h56m23s
	1-1	6.9e-18	9.1e-28	3.0e-12	3	3	17m24s
80%	0-0	7.2e+07	0.218	6.086	156	119	17h32m00s
	1-0	1.1e-10	5.6e-17	1.5e-08	190	143	2d07h29m16s
	1-1	3.1e-19	4.7e-29	3.0e-12	3	3	17m28s
85%	0-0	2.9e+07	0.225	4.005	125	93	13h34m45s
	1-0	1.3e-09	1.2e-15	8.4e-08	190	141	2d06h09m48s
	1-1	2.7e-14	1.0e-23	3.2e-10	3	3	17m25s
90%	0-0	4.7e+06	0.087	1.216	257	142	20h45m59s
	1-0	4.1e-07	4.4e-12	4.1e-06	321	201	3d05h52m01s
	1-1	1.5e-16	9.4e-26	8.2e-11	3	3	17m29s
95%	0-0	3.0e+06	0.579	2.272	342	146	21h17m38s
	1-0	3.4e+05	0.297	0.971	287	131	2d02h39m10s
	2-0	2.6e-03	7.6e-04	1.4e-03	319	201	7d18h58m26s
	2-1	1.1e-09	4.5e-07	1.1e-06	223	142	2d06h47m23s
	2-2	1.5e-16	1.0e-12	1.7e-10	3	3	17m23s

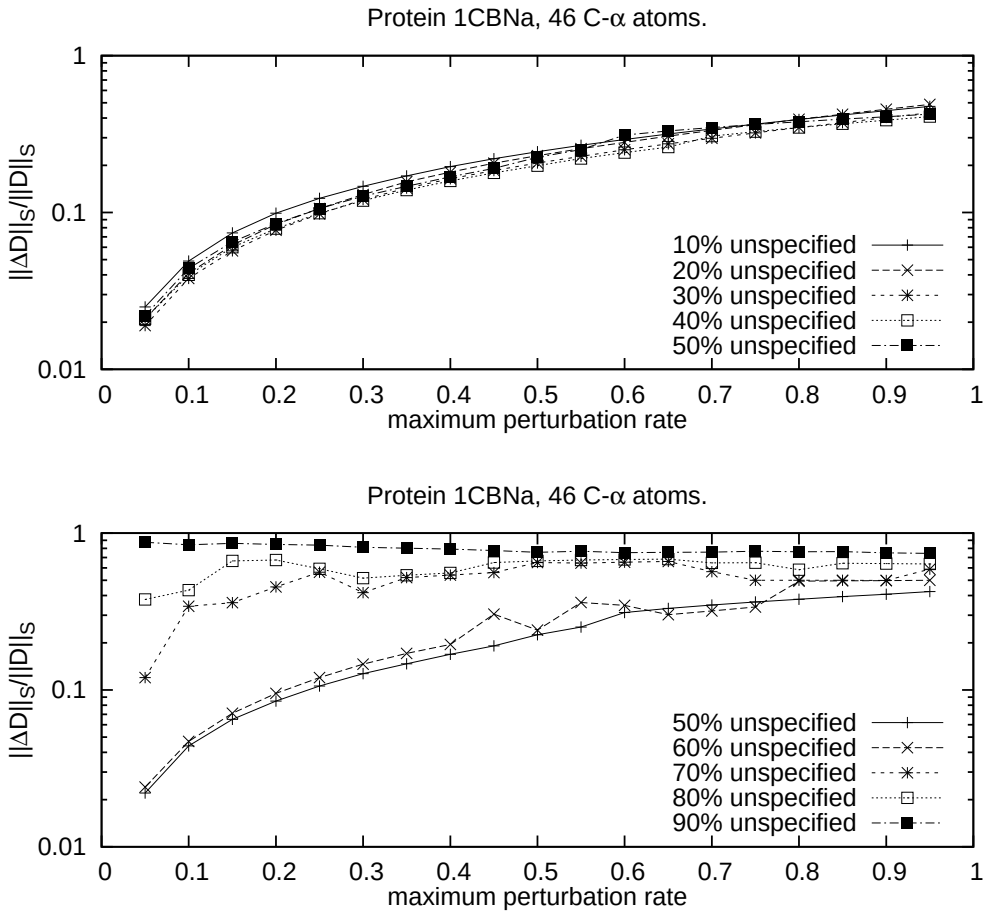

Figure 6.4: Perturbation experiment on protein 1CBNa, 46 Cα atoms.

6.5 Conclusion

We used the global optimization formulation to tackle the EDMCP and presented two methods for initialization. For EDMCPs with unspecified entries about 60% or less, our initialization methods usually reach the global minimum using a local minimization procedure.

We developed a dimensional relaxation scheme with two dimensional reduction methods to avoid local minima. In our experiments with dimensional re-

laxation 2, we can generally find the solution in a modest number of iterations for protein problems (1BPI, 1CBNa, 1MBC, 2GDM) considering only Cα atoms ($n = 58, 46, 153, 153$) with up to 80% unspecified entries. For those with unspecified entries 85% or more, the solution may not be reliable because insufficient distance information results in multiple solutions to the optimization problem (see Tables 6.11–6.23).

We also experimented on all 460 atoms of 1BPI and all 1244 atoms of 1MBC. In both experiments we successfully solved all the problems with unspecified entries up to 95% (see Tables 6.24–6.25).

Chapter 7

Summary and Future Directions

We have analyzed Cholesky-related factorizations of symmetric triadic matrices using various pivoting methods. We gave new backward error stability analysis for LBL^T factorizations using the inner product formulation.

We studied existing modified Cholesky algorithms, developed new ones, and investigated their satisfaction of the objectives for being reliable modified Newton methods. We incorporated these modification algorithms into the interior point methods for nonlinear optimization implemented in the OPT++ library[1].

We tackled the Euclidean distance matrix problem, by transforming it into a global optimization problem, using modified Cholesky algorithms to generate descent directions. We also presented the dimensional relaxation method to avoid local solutions. In our experiments on distance matrices derived from protein data (1BPI, 1CBNa, 1MBC, 2GDM) considering only Cα atoms ($n = 58, 46, 153, 153$), we have successfully solved problems with up to 80% missing data. We also experimented on proteins 1BPI and 1MBC with all atoms ($n = 460, 1244$, respectively) and solved all problems with up to 95% missing data.

[1]OPT++ is a library of nonlinear optimization algorithms by Patty Hough, Juan Meza, and Pam Williams, Sandia National Laboratory, USA.

7.1 Summary

Our results concerning symmetric matrix factorizations are listed below.

1. We proved that triadic structure is preserved in the LL^T, LDL^T, and LBL^T factorizations (see Section 2.1).

2. Growth factors of factorization algorithms play an important role in stability analysis (e.g., [41, Chapters 9–11]). We gave bounds on the growth factors of the LBL^T factorizations for symmetric triadic matrices (see Section 2.3). The bounds are much lower than those for general symmetric matrices (see Table 3.1 for a comparison).

3. We developed perturbation analysis of the LDL^T and LBL^T factorizations (see Section 2.4). This is a generalization of Higham's analysis for the LL^T factorization [38].

4. We presented the new backward error analysis of LBL^T factorizations via the inner product formulation (see Sections 3.1 and 3.2), rather than via the outer product formulation used in the literature [13, 39, 40, 58].

5. We analyzed the stability of rank estimation for symmetric indefinite matrices using the LBL^T factorization (see Section 3.4). This is a generalization of Higham's analysis on rank estimation for symmetric positive definite matrices using the Cholesky factorization [38].

6. Our numerical experiments on rank estimation showed that there is a trade-off between computational cost and reliability of the estimated ranks. To balance the two factors, we recommend using the fast Bunch-Parlett pivoting strategy with stopping criterion given in (3.45).

We cataloged the modified Cholesky algorithms in the literature and our new ones (see Table 5.8). Our results on modified Cholesky algorithms for Newton-type optimization are summarized below.

1. Inspired by the existing algorithms GMW81 [28, Chapter 4], SE90 [54], and SE99 [55] (see Section 5.1), we developed new modified LDL^T algorithms: GMW-I, GMW-II, and SE-I (see Section 5.2).

2. The two modified LBL^T algorithms, MS79 [45] and CH98 [16], in the worst cases take $\Theta(n^3)$ time more than the standard Cholesky factorization (see Section 5.3). We developed the LTL^T-MS79 and LTL^T-CH98 algorithms that guarantee a worst case cost of $O(n^2)$, resulting from the merits of triadic structure (see Section 5.4). In our experiments, for matrices close to being positive definite, LTL^T-MS79 also outperformed MS79 by usually producing a smaller $\|E\|_2$, and similarly for LTL^T-CH98 vs. CH98.

Our results on the Euclidean distance matrix completion problems (EDMCP) are as follows.

1. We studied the properties of Euclidean distance matrices, presented three different optimization program formulations to solve the EDMCP, and showed relationships between them via linear transformations. We tackled the EDMCP via the global optimization formulation.

2. We presented two methods to estimate the EDM from a predistance matrix F: replacing the negative eigenvalues of $\mathcal{T}(F)$ by zero, and computing the nearest distance matrix to F. In our experiments on the EDMCPs with about 60% unspecified entries or less, a local minimization procedure

started from our estimated configurations generally led to the global minimum.

3. We developed a dimensional relaxation method via the approach by Crippen [18]. Increasing the dimensions in an optimization program is straightforward. The challenge is to obtain a global minimizer in a low dimensional space given a global minimizer in a higher dimensional space. We have developed an effective process involving a number of random orthogonal matrices, and also explored the use of inertial tensor matrix.

4. We successfully solved the protein problems (1BPI, 1CBNa, 1MBC, 2GDM) considering only Cα atoms ($n = 58, 46, 153, 153$) with up to 80% unspecified entries. We also experimented on 1BPI and 1MBC with all atoms ($n = 460, 1244$), and successfully solved all problems with up to 95% unspecified entries.

7.2 Future Directions

Possible future directions include the following.

1. There has been some recent attention given to matrices whose graphs are trees plus a few edges [59]. It should be possible to obtain bounds for the growth factor analogous to those for triadic matrices.

2. The LBL^T factorizations of symmetric matrices have been well studied [5, 9, 10, 12, 13, 14]. Skew-symmetric matrices[2] also have factorizations

[2]A matrix $A \in R^{n \times n}$ is called skew-symmetric if $A^T = -A$.

in LBL^T form [7, 11], where L is unit lower triangular and B is skew-symmetric and block diagonal with blocks of order 1 or 2. Our stability analysis, including that for rank estimation, can be adapted for LBL^T factorizations of skew-symmetric matrices.

3. A symmetric matrix has an LTL^T factorization [1, 50], where L is unit lower triangular and T is symmetric tridiagonal. Bunch [11] pointed out that a skew-symmetric matrix can also be factorized into the LTL^T form, where T is skew-symmetric and tridiagonal. This direction may deserve investigation.

4. Parallel Cholesky factorizations have been well studied and investigated (e.g., [20, 48, 49]). Parallel LBL^T factorizations deserve attention for symmetric indefinite systems.

5. We have studied modified Cholesky factorizations [16, 45, 54, 55][28, Chapter 4] and developed new ones for computing descent directions. Cholesky-related factorizations can be also applied to compute directions of negative curvature [25, 45]. One idea is to investigate the usefulness of our factorization in computing these directions.

6. Conjugate gradient and Lanczos algorithms are widely applied to solve linear systems and compute eigenvalues from symmetric matrices[3]. These iterative methods can also be applied to compute descent directions and directions of negative curvature [46]. This approach is currently under-explored.

[3]See [32] for a review article of conjugate gradient and Lanczos algorithms.

7. For a good starting point in global optimization to solve the EDMCP, we used the nearest EDM, computed by an alternating projection algorithm [30]. The set of EDMs is convex, a key property required for the alternating projection approach. For a better estimation of the EDM, we could constrain the embedding dimension, but this ruins the convexity. Hence the alternating projection algorithms are unlikely to apply. This problem was studied by Mathar [43]. Developing an efficient algorithm to compute the nearest EDM with a constrained embedding dimension is a research problem.

8. If a EDMCP has a reducible symmetric partial matrix, then it can be split into multiple simpler EDMCPs. If a symmetric partial matrix is irreducible, then the corresponding indirected graph is connected. This property, accompanied with the triangle inequalities, may help improve the current EDM initialization.

9. In dimensional relaxation, the challenge is to project the coordinates in the higher dimensional space to the coordinates in the lower dimensional space after a rigid transformation. Using inertial tensor matrix we can minimize the sum of element changes in the distance matrix (see Section 6.3.2). However, it does not minimize the change of the objective function. The approaches for orthogonal Procrustes problem [34, 57] may be useful here.

10. Optimization in higher dimensions results in more variables and therefore is expensive. Alternative strategies, such as multistarts or changing the worst few coordinates, can be applied to our system. This may reduce the need to relax the dimensions.

11. Solving Euclidean distance matrix problem via semidefinite programming [3] usually results in a high rank minimizer, and then the embedding dimension is higher than required. Low rank semidefinite programming [15] can be used to suppress the excessive rank, but it ruins the convexity.

12. It is possible to adapt our system to solve *position calibration* problems [52, 53], where the distances between acoustic sensors (microphones) and acoustic actuators (speakers) are measured by the multiplication of the time sound takes to travel and its speed. The goal is to estimate their relative three-dimensional coordinates.

BIBLIOGRAPHY

[1] J. O. Aasen. On the reduction of a symmetric matrix to tridiagonal form. *BIT*, 11(3):233–242, 1971.

[2] A. Y. Alfakih. On the uniqueness of Euclidean distance matrix completions. *Linear Algebra Appl.*, 370:1–14, 2003.

[3] A. Y. Alfakih, A. Khandani, and H. Wolkowicz. Solving Euclidean distance matrix completion problems via semidefinite programming. *Comput. Optim. Appl.*, 12(1-3):13–30, 1999.

[4] A. Y. Alfakih and H. Wolkowicz. On the embeddability of weighted graphs in Euclidean spaces. Technical Report CORR 98-12, Computer Science Department, Univ. of Waterloo, 1998.

[5] C. Ashcraft, R. G. Grimes, and J. G. Lewis. Accurate symmetric indefinite linear equation solvers. *SIAM J. Matrix Anal. Appl.*, 20(2):513–561, 1998.

[6] M. Bakonyi and C. R. Johnson. The Euclidean distance matrix completion problem. *SIAM J. Matrix Anal. Appl.*, 16(2):646–654, 1995.

[7] P. Benner, R. Byers, H. Faßbender, V. Mehrmann, and D. Watkins. Cholesky-like factorizations of skew-symmetric matrices. *Electr. Trans. Num. Anal.*, 11:85–93, 2000.

[8] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[9] J. R. Bunch. Analysis of the diagonal pivoting method. *SIAM J. Numer. Anal.*, 8(4):656–680, Dec. 1971.

[10] J. R. Bunch. Partial pivoting strategies for symmetric matrices. *SIAM J. Numer. Anal.*, 11(3):521–528, 1974.

[11] J. R. Bunch. A note on the stable decomposition of skew-symmetric matrices. *Math. Comp.*, 38(158):475–479, 1982.

[12] J. R. Bunch and L. Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Math. Comp.*, 31:163–179, 1977.

[13] J. R. Bunch and R. F. Marcia. A pivoting strategy for symmetric tridiagonal matrices. *Numer. Linear Algebra Appl.*, 12(9):911–922, 2005.

[14] J. R. Bunch and B. N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 8(4):639–655, 1971.

[15] S. Burer and R. D.C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Math. Programming*, 103(3):427–444, July 2005.

[16] S. H. Cheng and N. J. Higham. A modified Cholesky algorithm based on a symmetric indefinite factorization. *SIAM J. Matrix Anal. Appl.*, 19(4):1097–1110, 1998.

[17] Peter Clote and Rolf Backofen. *Computational Molecular Biology: An Introduction*. John Wiley & Sons, August 2000.

[18] G. M. Crippen. Conformational analysis by energy embedding. *J. Comp. Chem.*, 3:471–476, 1982.

[19] J. W. Demmel, N. J. Higham, and R. S. Schreiber. Stability of block *LU* factorization. *Numer. Linear Algebra Appl.*, 2(2):173–190, 1995.

[20] C. T. Djamegni. Synthesis of space-time optimal systolic algorithms for the Cholesky factorization. *Discrete Mathematics and Theoretical Computer Science*, 5:109–120, 2002.

[21] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers*. SIAM, 1991.

[22] H.-r. Fang. Backward error analysis of factorization algorithms for symmetric and symmetric triadic matrices. Technical Report CS-4734, Computer Science Department, Univ. of Maryland, College Park, MD, July, 2005.

[23] H.-r. Fang and D. P. O'Leary. Stable factorizations of symmetric tridiagonal and triadic matrices. *SIAM J. Matrix Anal. Appl., to Appear*, 2006.

[24] H.-r. Fang and D. P. O'Leary. Modified Cholesky factorizations: A catalog with new approaches. Technical Report CS-4807, Computer Science Department, Univ. of Maryland, College Park, MD, July, 2006.

[25] A. Forsgren, P. E. Gill, and W. Murray. Computing modified Newton directions using a partial Cholesky factorization. *SIAM J. Sci. Comput.*, 16(1):139–150, 1995.

[26] L. V. Foster. The growth factor and efficiency of Gaussian elimination with rook pivoting. *J. Comput. Appl. Math.*, 98(1):177, 1998.

[27] P. E. Gill and W. Murray. Newton-type methods for unconstrained and linearly constrained optimization. *Math. Programming*, 28:311–350, 1974.

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

[29] P. E. Gill, M. A. Saunders, and J. R. Shinnerl. On the stability of Cholesky factorization for symmetric quasi-definite systems. *SIAM J. Matrix Anal. Appl.*, 17(1):35–46, 1996.

[30] W. Glunt, T. L. Hayden, S. Hong, and J Wells. An alternating projection algorithm for computing the nearest Euclidean distance matrix. *SIAM J. Matrix Anal. Appl.*, 11(4):589–600, 1990.

[31] W. Glunt, T. L. Hayden, and M. Raydan. Molecular conformation from distance matrices. *J. Comp. Chem.*, 14:114–120, 1993.

[32] Gene H. Golub and Dianne P. O'Leary. Some history of the conjugate gradient and lanczos algorithms: 1948-1976. *SIAM Review*, 31:50–102, 1989.

[33] J. C. Gower. Properties of Euclidean and non-Euclidean distance matrices. *Linear Algebra Appl.*, 67:81–97, 1985.

[34] B. F. Green. The orthogonal approximation of an oblique structure in factor analysis. *Psychometrika*, 17:429–440, 1952.

[35] T. F. Havel. An evaluation of computational strategies for use in the determination of protein structure from distance geometry constraints obtained by nuclear magnetic resonance. *Prog. Biophys. Mol. Biol.*, 56:43–78, 1991.

[36] J. C. Haws and C. D. Meyer. Preconditioning KKT systems. Technical Report M&CT-Tech-01-021, The Boeing Co., 2003.

[37] B. A. Hendrickson. The molecule problem: exploiting structure in global optimization. *SIAM J. Optim.*, 5(4):835–857, 1995.

[38] N. J. Higham. Analysis of the Cholesky decomposition of a semi-definite matrix. In M. G. Cox and S. J. Hammarling, editors, *Reliable Numerical Computation*, pages 161–185. Oxford University Press, New York, 1990.

[39] N. J. Higham. Stability of the diagonal pivoting method with partial pivoting. *SIAM J. Matrix Anal. Appl.*, 18(1):52–65, 1997.

[40] N. J. Higham. Stability of block LDL^T factorization of a symmetric tridiagonal matrix. *Linear Algebra Appl.*, 287:181–189, 1999.

[41] N. J. Higham. *Accuracy and Stability of Numerical Algorithms, 2nd ed.* SIAM, 2002.

[42] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[43] R. Mathar. The best Euclidian fit to a given distance matrix in prescribed dimensions. *Linear Algebra Appl.*, pages 1–6, 1985.

[44] J. Moré and Z. Wu. Global continuation for distance geometry problems. *SIAM J. Optim.*, 7(3):814–836, 1997.

[45] J. J. Moré and D. C. Sorensen. On the use of directions of negative curvature in a modified Newton method. *Math. Programming*, 16:1–20, 1979.

[46] S. G. Nash. Newton-type minimization via the Lanczos algorithm. *SIAM J. Numer. Anal.*, 21:770–788, 1984.

[47] Stephen G. Nash and Ariela Sofer. *Linear and Nonlinear Programming*. The McGraw-Hill Companies, Inc., 1996.

[48] D. P. O'Leary and G. W. Stewart. Data-flow algorithms for parallel matrix computations. *Comm. of the ACM*, 28:840–853, 1985.

[49] D. P. O'Leary and G. W. Stewart. Assignment and scheduling in parallel matrix factorization. *Linear Algebra Appl.*, 77:275–300, 1986.

[50] B. N. Parlett and J. K. Reid. On the solution of a system of linear equations whose matrix is symmetric but not definite. *BIT*, 10(3):386–397, 1970.

[51] E. O. Purisima and H. A. Scheraga. An approach to the multiple-minima problems by relaxing dimensionality. *Proc. Natn. Acad. Sci. USA*, 83:2782–2786, 1986.

[52] V. C. Raykar and R. Duraiswami. Automatic position calibration of multiple microphones. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04)*, volume 4, pages iv–69 – iv–72. 2004.

[53] V. C. Raykar, Igor V. Kozintsev, and R. Lienhart. Position calibration of microphones and loudspeakers in distributed computing platforms. *IEEE Transaction on Speech and Audio Processing*, 13(1):70–83, 2005.

[54] R. B. Schnabel and E. Eskow. A new modified Cholesky factorization. *SIAM J. Sci. Stat. Comput.*, 11:1136–1158, 1990.

[55] R. B. Schnabel and E. Eskow. A revised modified Cholesky factorization algorithm. *SIAM J. Optim.*, 9(4):1135–1148, 1999.

[56] I. J. Schoenberg. Remarks to Maurice Frechet's article: Sur la definition axiomatique d'une classe d'espaces vectoriels distancies applicables vectoriellement sur l'espace de Hilbert. *Ann. Math.*, 36:724–732, 1935.

[57] P. H. Schonemann. A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31, 1966.

[58] I. Slapničar. Componentwise analysis of direct factorization of real symmetric and Hermitian matrices. *Linear Algebra Appl.*, 272:227–275, 1998.

[59] D. Spielman and S.-H. Teng. Nearly linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In László Babai, editor, *Proc. the 36th Annual ACM Symposium on Theory of Computing (STOC'04)*, pages 81–90. ACM, 2004.

[60] G. W. Stewart. The efficient generation of random orthogonal matrices with an application to condition estimation. *SIAM J. Numer. Anal.*, 17:403–409, 1980.

[61] M. W. Trosset. Distance matrix completion by numerical optimization. *Comput. Optim. Appl.*, 17(1):11–22, 2000.

[62] R. J. Vanderbei. Symmetric quasidefinite systems. *SIAM J. Optim.*, 5(1):100–113, 1995.

[63] Wikipedia. Amino acid — wikipedia, the free encyclopedia, 2006. [Online; accessed 27-July-2006].

[64] Wikipedia. Protein structure — wikipedia, the free encyclopedia, 2006. [Online; accessed 27-July-2006].

[65] J. H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. ACM*, 8:281–330, 1961.

[66] J. H. Wilkinson. A priori error analysis of algebraic processes (cited in [41, 38]). *Proc. International Congress of Mathematicians*, 25:629–640, 1968.

[67] K. Wüthrich. NMR studies of structure and function of biological macromolecules (nobel lecture). *Angew. Chem.*, 42(29):3340–3361, 2003.

[68] G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22, 1935.

[69] Z. Zou, R. H. Bird, and R. B. Schnabel. A stochastic/perturbation global optimization algorithm for distance geometry problems. *J. Global Optim.*, 11(1):91–105, 1997.