Hyperbolic Rotations for Downdating the Cholesky Factorization with Application to Signal Processing

Ching-Tsuan Pan

Center for Communications and Signal Processing
Department of Mathematics
North Carolina State University

July 1987

CCSP-TR-87/10

Abstract

PAN, CHING-TSUAN. Hyperbolic Rotations for Downdating the Cholesky Factorization with Application to Signal Processing (Under the direction of Robert J. Plemmons.)

In many applications the rank one modification (updating or downdating) of the Cholesky factorization of a positive definite matrix A is an important computation. There are two standard downdating algorithms: the LINPACK algorithm, which is based on orthogonal Givens rotations, and the hyperbolic rotation algorithm, which is similar to Givens rotations except that it uses hyperbolic functions. The LINPACK algorithm is known to be backward stable while the hyperbolic rotation algorithm is faster.

This thesis presents a complete forward error analysis of the hyperbolic rotation algorithm and shows that the algorithm is forward (or weakly) stable. A new algorithm for downdating Cholesky factorizations is proposed and tested. This new algorithm is faster than the hyperbolic rotation algorithm and is expected to be as stable as the LINPACK method.

Applications of downdating and updating schemes to recursive least squares filtering are also discussed.

Acknowledgments

This work owes a great deal of thanks to my advisor, Professor R.J. Plemmons, for the topic he brought to my attention, for the financial support he gained for me, and for his direction, encouragement as well as patience in my work. I would also like to thank my committee members: Professors S.T. Alexander, R.E. Funderlic, C.D. Meyer, Jr., and E.L. Stitzinger, for their willingness to discuss with me many different and interesting aspects of this project.

A special acknowledgment is due to Professor G.W. Stewart for his valuable comments and suggestions in the error analysis part of this dissertation.

On a more personal note, I wish to express my deep gratitude to my undergraduate teacher and best friend, Dr. Yuan Lay, who gave me my basic training in mathematical thinking.

This document was prepared using the word processing facilities at ECE/CCSP. I am indebted to Ingrid Kremer; without her skill and patience the preparation of this manuscript would have been much more painful.

This research was supported in part by the U.S. Air Force, Grant No. AFOSR-83-0225-C.

Table of Contents

Chapter 0 Introduction 1
0.1 Updating and downdating least squares problems 1
0.2 Downdating by the hyperbolic method 8
0.3 Downdating by the LINPACK method 14
0.4 Applications to signal processing 19
0.5 Outline of this thesis 23
Chapter 1 Preliminary Results 26
1.1 The vector **a** 26
1.2 The basic equation $AR=D$ 32
1.3 More explicit expressions 35
1.4 A sharp bound for $||H||$ 39
Chapter 2 Error Analysis 49
2.1 Conditioning 50
2.2 Error analysis of the hyperbolic method 53
2.3 Stability of the hyperbolic method 60
Chapter 3 New Fast Downdating Algorithms 64
3.1 Reorganization of Algorithm AR 64
3.2 The new downdating algorithms 67
Chapter 4 Stability Testing Results 74
4.1 Generating the original data 74
4.2 The "theoretical" answer 76
4.3 The test results 77
Chapter 5 Concluding Remarks 79
References 81

CHAPTER 0: INTRODUCTION

In the first three sections of this chapter we introduce the central theme of this thesis, *downdating the Cholesky factorization*, and describe two existing algorithms, the LINPACK method and the hyperbolic method, used commonly now in downdating. The applications, especially to recursive least squares filtering, are discussed in section 0.4. The final section of this chapter gives an outline and indicates what are the new results in this thesis.

0.1 Updating and downdating least squares problems

The linear least squares method is one of the oldest topics in applied mathematics. It has numerous applications in modern engineering and science. In the past decade several algorithms [see, for example, Lawson and Hanson (1974)] were proposed to solve a succession of least squares problems after a rank-one modification at each step. Such a procedure is also called updating or downdating least squares problems. These algorithms especially find great applicability to fast recursive filters in linear predictions and estimations. It is becoming increasingly important to understand the numerical and mathematical properties of these fast updating algorithms.

The current section continues with a review of general least squares problems, including updating and downdating methods for recursive computations, and finally leads to the definition of the *downdating problem*. Here and

in what follows, all least squares problems are linear.

We only consider the real numbers, denoted as $\mathbb{R}$. Thus $\mathbb{R}^n$ is the column vector space over $\mathbb{R}$ with dimension n, and $\mathbb{R}^{m\times n}$ is the set of all matrices with size of m rows and n columns. The vectors are denoted by boldface lowercase. Throughout this thesis $||\cdot||$ denotes the Euclidean norm of a vector in $\mathbb{R}^n$, i.e., $||\mathbf{x}|| = \sqrt{\mathbf{x}^T\mathbf{x}}$ for $\mathbf{x}\in\mathbb{R}^n$. The symbol $A>0$ with $A\in\mathbb{R}^{n\times n}$ is the abbreviation of A being a positive definite matrix, i.e., for any nonzero vector $\mathbf{x}\in\mathbb{R}^n$, $\mathbf{x}^T A\mathbf{x}>0$.

0.1-1 **Least squares problems.** Let $X\in\mathbb{R}^{m\times n}$, $m>n$, and $\mathbf{s}\in\mathbb{R}^m$. What we mean by the least squares problems is that for an *overdetermined* linear system

$$X\mathbf{w} = \mathbf{s}\ , \tag{0.1-2}$$

one seeks a vector $\mathbf{w}_{LS}$ which minimizes the Euclidean norm of the residual vector $\mathbf{r}\equiv X\mathbf{w}-\mathbf{s}$,

$$\min||\mathbf{r}|| = \min_{\mathbf{w}}\ ||X\mathbf{w}-\mathbf{s}||\ . \tag{0.1-3}$$

In this thesis the matrix X is always assumed to have *full column rank* (i.e., $rank(X)=n$); therefore, there exists a unique vector $\mathbf{w}_{LS}$ which minimizes the residual $||X\mathbf{w}-\mathbf{s}||$ to its minimum value: $\rho_{LS} = ||X\mathbf{w}_{LS} - \mathbf{s}||$ [Golub and Van Loan (1983)]. We assume $\rho_{LS}>0$ in this thesis. One of the popular ways to find $\mathbf{w}_{LS}$ and ρ_{LS} is the following.

0.1-4 **The QR method.** Since orthogonal transformations will not change the Euclidean norm, then

$$||\mathbf{r}|| = ||Q_1 X\mathbf{w} - Q_1\mathbf{s}||$$

is true for any orthogonal transformation $Q_1 \in \mathbb{R}^{m\times m}$. It is well known that Q_1X can be made to be an upper triangular matrix $R_1 \in \mathbb{R}^{n\times n}$:

$$Q_1 X = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \begin{matrix} \}n \\ \}m-n \end{matrix} .$$

Let

$$Q_1\mathbf{s} = \begin{pmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} \begin{matrix} \}n \\ \}m-n \end{matrix} ;$$

then we immediately know that $||\mathbf{r}||$ reaches its minimum if and only if

$$R_1\mathbf{w}_{LS} = \mathbf{s}_1$$

and

$$\rho_{LS} = \min||\mathbf{r}|| = ||\mathbf{s}_2|| .$$

When $m>n$ and system (0.1-2) is not consistent one may apply another orthogonal transformation Q_2 to $Q_1\mathbf{r}$ such that

$$Q_2 \begin{bmatrix} \mathbf{s}_1 \\ \mathbf{s}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{s}_1 \\ \rho \\ 0 \end{bmatrix} \quad , \ ||\mathbf{s}_2|| = \rho$$

and let $Q = Q_2Q_1$; then the argument matrix $[X|\mathbf{s}]$ becomes upper triangular:

$$Q[X|\mathbf{s}] = \left[\begin{array}{c|c} R_1 & \mathbf{s}_1 \\ \mathbf{0}^T & \rho \\ 0 & 0 \end{array}\right] \equiv \begin{bmatrix} R \\ 0 \end{bmatrix} . \qquad (0.1\text{-}5)$$

As we know, R is the Cholesky factor of the matrix $[X|\mathbf{s}]^T[X|\mathbf{s}]$ and is unique up to the signs of the rows of R. Therefore, solving the least squares problem (0.1-1) can be converted to the problem of finding a Cholesky factorization of the positive definite matrix $[X|\mathbf{s}]^T[X|\mathbf{s}]$. Notice that when the linear system (0.1-2) is consistent; then $\rho_{LS}=0$ and the matrix $[X|\mathbf{s}]^T[X|\mathbf{s}]$ is no longer positive definite.

0.1-6 **Notation.** For the reason above and others we will use $LS(X|\mathbf{s})$ as an abbreviation throughout this thesis to denote least squares problem (0.1-1) with $m>n$ and $\rho_{LS}>0$.

We also introduce the following closely related notation.

0.1-7 **Notation.** Let $A\in\mathbb{R}^{m\times n}$ $(m\geq n)$ have full column rank; then $A^TA\in\mathbb{R}^{n\times n}$ is a positive definite matrix. Hence there exists a unique Cholesky factor of A^TA , R, with positive diagonal entries, such that $A^TA = R^TR$ and $R\in\mathbb{R}^{n\times n}$ is upper triangular. [See, for example, Horn and Johnson (1985), 406-407]. We define, in this thesis, a function Q such that

$$\mathbf{Q}(A) = R \quad . \tag{0.1-8}$$

Actually Q is a mapping from $\mathbb{R}^{m\times n}$ to $\mathbb{R}^{n\times n}$ $(m\geq n)$, and R is the upper triangular part of the well-known QR factorization of A. Using (0.1-5), one can summarize the procedure to solve $LS(X|\mathbf{s})$ problem as follows.

0.1-9 **Procedure.**

(1) Find $\mathbf{Q}(X|\mathbf{s})$ (for example, by either Householder reflections or Givens rotations).

(2) Let $\mathbf{Q}(X|\mathbf{s}) = \left[\begin{array}{c|c} R_1 & \mathbf{s}_1 \\ \hline \mathbf{0}^T & \rho \end{array}\right]$; solve $R_1\mathbf{w} = \mathbf{s}_1$.

Then $\mathbf{w}_{LS} = R_1^{-1}\mathbf{s}_1$ minimizes $||X\mathbf{w} - \mathbf{s}||$ and $\rho_{LS} = \rho$.

It is known that solving a $LS(X|\mathbf{s})$ problem requires $0(mn^2)$ multiplications. For the reasons stated in Stewart (1973), we will report only the number of multiplications (after dropping the lower order terms) required by an algorithm.

In applications, for example to signal processing, very often it is required to recalculate the $\mathbf{w}$ when the observations (i.e., the equations) are successively added to and deleted from the linear system (0.1-2). The addition of a row vector $\mathbf{z}^T$ to X is called *updating*, whereas the deletion of a row vector $\mathbf{z}^T$ from X is called *downdating*. In these situations we seek to solve the modified least squares problem

$$\min_{\mathbf{w}} = ||\bar{X}\mathbf{w} - \bar{\mathbf{s}}||$$

for the modified least squares vector $\bar{\mathbf{w}}_{LS}$, where $\bar{X}$ denotes the result of adding and/or deleting a row of X, and $\bar{\mathbf{s}}$ denotes the corresponding modification to $\mathbf{s}$. Clearly, we should be able to compute $\bar{\mathbf{w}}_{LS}$ by modifying the factor R in (0.1-5) to $\bar{R}$ without performing a complete recalculation. In particular, we will consider only algorithms involving $0(n^2)$ rather than $0(mn^2)$

multiplications.

We proceed with more detail about the updating and downdating techniques.

0.1-10 **Updating least squares problems.** Let $z \in \mathbb{R}^n$ be a new observation data set and, correspondingly, the $\eta \in \mathbb{R}$ be the new component augmented to s. One wants to solve the $LS\left(\left[\begin{array}{c|c} X & s \\ z^T & \eta \end{array}\right]\right)$ problem given $Q(X|s) = \left[\begin{array}{c|c} R & s_1 \\ 0^T & \rho \end{array}\right]$ without referring back to the matrix $\left[\begin{array}{c|c} X & s \\ z^T & \eta \end{array}\right]$.

It is easy to see that we actually want to find

$$Q\left(\left[\begin{array}{c|c} R & s_1 \\ 0^T & \rho \\ z^T & \eta \end{array}\right]\right) .$$

To solve this problem, we can use Givens rotations to zero out the last row (z^T, η) without any fill-in in the lower triangular part of its submatrix $Q(X|s)$. So solving this problem is trivial, and the algorithm is known to be stable [Wilkinson (1965)].

0.1-11 **Downdating least squares problems.** Let $X = \left[\begin{array}{c} X' \\ z^T \end{array}\right]$ and $s = \left[\begin{array}{c} s' \\ \eta \end{array}\right]$. One wants to solve the $LS(X'|s')$ problem given $Q(X|s) = \left[\begin{array}{c|c} R & s_1 \\ 0^T & \rho \end{array}\right]$ without referring back to the matrix $[X'|s']$; *this is the main underlying problem of this*

thesis. Here we assume that $LS(X'|\mathbf{s}')$ still satisfies the conditions in (0.1-1).

Notice that problem (0.1-11) is equivalent to the problem of finding the Cholesky factorization of the matrix

$$\left[\begin{array}{c|c} R & \mathbf{s}_1 \\ \mathbf{0}^T & \rho \end{array}\right]^T \left[\begin{array}{c|c} R & \mathbf{s}_1 \\ \mathbf{0}^T & \rho \end{array}\right] - \begin{pmatrix} \mathbf{z} \\ \eta \end{pmatrix} (\mathbf{z}^T, \eta) \ .$$

For this reason, henceforth this thesis will deal only with downdating the Cholesky factor problem defined as follows.

0.1-12 **The downdating problem**. Accoding to Stewart (1979), the problem of finding the Cholesky factorization of

$$R^T R - \mathbf{z}\mathbf{z}^T \ , \tag{0.1-13}$$

is called the downdating problem. In (0.1-13), $R \in \mathbb{R}^{n \times n}$ is an upper triangular matrix with positive diagonal entries and $\mathbf{z} \in \mathbb{R}^n$ is the vector being removed.

It is clear that the downdating problem can be solved if and only if $R^T R - \mathbf{z}\mathbf{z}^T$ is positive definite; in such a case there exists a new Cholesky factor, or a so-called downdated Cholesky factor D, so that

$$D^T D = R^T R - \mathbf{z}\mathbf{z}^T \ .$$

There are mainly two algorithms available for solving the downdating problem. The following two sections will deal with them separately.

0.1-14 **Remark**. In this thesis we are not concerned with the updating or downdating of the LDL^T type of factorization, where L is a lower triangular

matrix with diagonal of 1's and D is a diagonal matrix [see Fletcher and Powell (1974) and Gill, Golub, Murry and Saunders (1974)]. In fact, by using the idea of Gentleman (1973), in the LDL^T type of updating technique one updates $D^{1/2}$ and L^T separately rather than updating $R = D^{1/2}L^T$ to avoid computing square roots and to reduce the number of multiplications as well.

0.2 Downdating by the hyperbolic method

In the area of scientific computation the hyperbolic method [Golub, (1969) and Chambers, (1971)] was used earlier than the LINPACK method (See section 0.3) in downdating the Cholesky factor. But the concept of hyperbolic rotations has been used in many different branches of mathematics, and even in physics, for quite a long time. For example, in relativity theory the famous Lorentz transformation is essentially a hyperbolic rotation [Einstein, Lorentz, Weyl and Minkowski (1923)]. In numerical analysis, such transformations are also used for many different purposes: for example, in eigenvalue computations [Brebner and Grad (1982), Bunse-Gerstner (1981), and Elsner (1979)], in solving positive definite systems [Delosme and Ipsen (1986)], and in signal processing [Kailath, Vieira and Morf (1978)].

0.2-1 **Pseudo-orthogonal matrix**. A pseudo-orthogonal matrix $P \in \mathbb{R}^{n \times n}$ with respect to S is a matrix which satisfies the equation

$$P^T S P = S \quad , \tag{0.2-2}$$

where $S = diag(\pm 1)$. If $S = I$, then P is an orthogonal matrix. Notice that the

set of all pseudo-orthogonal matrices in $\mathbb{R}^{n\times n}$ with respect to a fixed S forms a group under matrix multiplication.

Now let us see how pseudo-orthogonal matrices can be used to solve the downdating problem. Let $S = \begin{bmatrix} I_n & \\ & -1 \end{bmatrix}$. Now (0.1-13) can be rewritten as

$$[R^T \ \mathbf{z}]S\begin{bmatrix} R \\ \mathbf{z}^T \end{bmatrix} = R^TR - \mathbf{z}\mathbf{z}^T \ . \tag{0.2-3}$$

If we choose a proper pseudo-orthogonal matrix P such that

$$P\begin{bmatrix} R \\ \mathbf{z}^T \end{bmatrix} = \begin{bmatrix} D \\ \mathbf{0}^T \end{bmatrix} \ , \tag{0.2-4}$$

where D is upper triangular, then (0.2-3) becomes

$$R^TR - \mathbf{z}\mathbf{z}^T = [R^T \ \mathbf{z}]P^TSP\begin{bmatrix} R \\ \mathbf{z}^T \end{bmatrix} = D^TD \ ,$$

and D is the required downdating Cholesky factor. Now we are concerned with actually constructing the P in (0.2-4).

0.2-5 **Hyperbolic rotations**. Suppose a matrix $H \in \mathbb{R}^{n\times n}$ agrees with the identity matrix everywhere except in some 2-by-2 principal submatrix, where H has the form

$$H(k,l) = \begin{bmatrix} 1 & \vdots & & \vdots & \\ \cdots & \cosh\theta & \cdots & -\sinh\theta & \cdots \ k \\ & \vdots & & \vdots & \\ \cdots & -\sinh\theta & \cdots & \cosh\theta & \cdots \ l \\ & \vdots & & \vdots & 1 \end{bmatrix} \tag{0.2-6}$$

for some $\theta \in \mathbb{R}$. Using the identity $\cosh^2\theta - \sinh^2\theta = 1$ it is easy to check that

$H(k,l)$ is indeed a pseudo-orthogonal matrix with respect to $S = diag(1,\ldots 1,-1,1,\ldots,1)$, where (-1) occurs in the l-th position. We call $H(k,l)$ a hyperbolic rotation.

Like Givens rotations, hyperbolic rotations can also be used to annihilate a selected entry in a vector. For example if one wants to zero the l-th entry in a vector $\mathbf{x} \in \mathbb{R}^n$ by a hyperbolic rotation in the (k,l)-plane, then one might construct an $H(k,l)$ by letting

$$\cosh\theta = \frac{x_k}{\sqrt{x_k^2 - x_l^2}}$$

$$\sinh\theta = \frac{x_l}{\sqrt{x_k^2 - x_l^2}} \qquad (0.2\text{-}7)$$

in (0.2-6).

0.2-8 **Notation**. To be more specific, such a hyperbolic rotation [defined by (0.2-6) and (0.2-7)], applied to zero the l-th entry x_l by using k-th entry x_k of a vector $\mathbf{x}$, is denoted by

$$H(k,l) \equiv H(k,l;\ x_k,\ x_l)\ , \qquad (0.2\text{-}9)$$

throughout this thesis.

Observe that to insure the existence of such a hyperbolic rotation, it is essential that $|x_k| > |x_l|$ in (0.2-9).

0.2-10 **Remark**. In (0.2-9) if $k > l$, i.e., if one wants to use x_k to annihilate x_l in a vector

$$\mathbf{x} = \begin{pmatrix} \vdots \\ x_l \\ \vdots \\ x_k \\ \vdots \end{pmatrix},$$

then

$$H(k,l;\ x_k,\ x_l) = \begin{bmatrix} 1 & \vdots & & \vdots & \\ \cdots & \cosh\theta & \cdots & -\sinh\theta & \cdots \\ & \vdots & & \vdots & \\ \cdots & -\sinh\theta & \cdots & \cosh\theta & \cdots \\ & \vdots & & \vdots & 1 \end{bmatrix} \begin{matrix} \\ l \\ \\ k \\ \\ \end{matrix}$$

and (0.2-7) remains unchanged. Therefore, in the notation $H(k,l;\ x_k,\ x_l)$, it is not necessary that $k<l$, as long as $|x_k|>|x_l|$, both (0.2-6) and (0.2-7) remain unchanged. But in the case of Givens rotations this is not true [see Remark (0.3-6)].

0.2-11 **Remark**. Notice also that we can use trigonometric functions instead of hyperbolic functions to express the hyperbolic rotations. For example,

$$H(k,l;\ x_k,\ x_l) = \begin{bmatrix} 1 & \vdots & & \vdots & \\ \cdots & \dfrac{1}{\cos\theta} & \cdots & -\dfrac{\sin\theta}{\cos\theta} & \cdots \\ & \vdots & & \vdots & \\ \cdots & -\dfrac{\sin\theta}{\cos\theta} & \cdots & \dfrac{1}{\cos\theta} & \cdots \\ & \vdots & & \vdots & 1 \end{bmatrix} \tag{0.2-12}$$

and the left hand side of (0.2-7) may change accordingly.

The following algorithm uses the hyperbolic method to solve the down-dating problem.

0.2-13 **Algorithm** *HY*. Given an upper triangular matrix $R \equiv (r_{ij}) \in \mathbb{R}^{n \times n}$ and a vector $\mathbf{z} \equiv \mathbf{z}^{(0)} \in \mathbb{R}^n$ being removed, by using hyperbolic rotations, the following algorithm computes the downdated Cholesky factor $D \equiv (d_{ij}) \in \mathbb{R}^{n \times n}$ such that $D^T D = R^T R - \mathbf{z}\mathbf{z}^T$.

For $i = 1, n$

$$ch = \frac{r_{ii}}{\sqrt{r_{ii}^2 - (z_i^{(i-1)})^2}} \quad (\text{if } r_{ii}^2 - z_i^{(i-1)} \leq 0, \text{ stop})$$

$$sh = \frac{z_i^{(i-1)}}{\sqrt{r_{ii}^2 - (z_i^{(i-1)})^2}}$$

$$d_{ii} = \sqrt{r_{ii}^2 - (z_i^{(i-1)})^2}$$

For $j = (i+1), n$

$$d_{ij} = ch \cdot r_{ii} - sh \cdot z_j^{(i-1)}$$

$$z_j^{(i)} = -sh \cdot r_{ij} + ch \cdot z_j^{(i-1)}$$

This algorithm requires $2n^2$ multiplications and n square roots.

0.2-14 **Algorithm** *HY* **in matrix form.** For $i = 1,2\ldots,n$ one determines the hyperbolic rotations $P_i \in \mathbb{R}^{(n+1)\times(n+1)}$ such that

$$P_n \cdots P_2 P_1 \begin{bmatrix} R \\ \mathbf{z}^T \end{bmatrix} = \begin{bmatrix} D \\ \mathbf{0}^T \end{bmatrix} . \tag{0.2-15}$$

More specifically, let $\mathbf{z} = \mathbf{z}^{(0)}$ and $R = R^{(0)}$; then the i^{th}-step is

$$P_i \begin{bmatrix} R^{(i-1)} \\ (\mathbf{z}^{(i-1)})^T \end{bmatrix} = \begin{bmatrix} R^{(i)} \\ (\mathbf{z}^{(i)})^T \end{bmatrix} \tag{0.2-16}$$

and

$$R^{(i)} = \begin{bmatrix} D_1 \\ \vdots \\ D_i \\ R_{i+1} \\ \vdots \\ R_n \end{bmatrix}, \tag{0.2-17}$$

where

$$R = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} \quad \text{and} \quad D = \begin{bmatrix} D_1 \\ \vdots \\ D_n \end{bmatrix}.$$

are the row partitions. According to (0.2-8)

$$P_i = H(i, n+1;\ r_{ii},\ z_i^{(i-1)}) \tag{0.2-18}$$

and

$$\cosh \theta_i = \frac{r_{ii}}{\sqrt{r_{ii}^2 - (z_i^{(i-1)})^2}}$$

$$\sinh \theta_i = \frac{z_i^{(i-1)}}{\sqrt{r_{ii}^2 - (z_i^{(i-1)})^2}} \tag{0.2-19}$$

In Chapter 1 we show that $\frac{|z_i^{(i-1)}|}{|r_{ii}|} < 1$ when $R^T R - \mathbf{z}\mathbf{z}^T$ is positive definite [see Remark (1.1-18)].

At the end of this section, we give another version of the hyperbolic algorithm which was originally published by Chambers (1971). According to the stability test in Chapter 4, this version is slightly more stable than Algorithm *HY*.

0.2-20 **Algorithm** *HC*. Given an upper triangular matrix $R \equiv (r_{ij}) \in \mathbb{R}^{n \times n}$ and a vector $\mathbf{z} \equiv \mathbf{z}^{(0)} \in R^n$ being removed, the following algorithm computes the downdated Cholesky factor $D \equiv (d_{ij}) \in \mathbb{R}^{n \times n}$ such that $D^T D = R^T R - \mathbf{z}\mathbf{z}^T$.

For $i = 1, n$

$$s = \frac{z_i}{r_{ii}}$$

$$c = \sqrt{1 - s^2} \quad (\text{if } 1 - s^2 \leq 0, \text{ stop})$$

$$d_{ii} = c r_{ii}$$

For $j = (i+1), n$

$$d_{ij} = (r_{ij} - s z_j^{(i-1)})/c$$

$$z_j^{(i)} = -s d_{ij} + c z_j^{(i-1)}$$

This algorithm requires $2n^2$ multiplications and n square roots.

Notice that according to Remark (0.2-11) this algorithm is different from Algorithm HY only in that the transformation of $z_j^{(i-1)}$ to $z_j^{(i)}$ is done by recursively using d_{ij}.

0.3 Downdating by the LINPACK method

We call this method LINPACK because it was written (by Stewart) as one of the subroutines called "CHDD" in LINPACK [Dougarra, Bunch, Moler and Stewart (1978)]. Originally this method was in Saunders (1972) and Gill, Golub, Murray and Saunders (1974). Since Givens rotations are used in the

LINPACK method, the following review is for notation purposes only.

0.3-1 **Givens Rotations.** A matrix $G \in \mathbb{R}^{n \times n}$ is an orthogonal matrix which is formed by inbedding a (k,l)-plane rotation into the identity matrix; then G has the form

$$G(k,l) = \begin{bmatrix} 1 & \vdots & & \vdots & \\ \cdots & \cos\theta & \cdots & \sin\theta & \cdots \\ & \vdots & & \vdots & \\ \cdots & -\sin\theta & \cdots & \cos\theta & \cdots \\ & \vdots & & \vdots & 1 \end{bmatrix} \begin{matrix} \\ k \\ \\ l \\ \\ \end{matrix} . \tag{0.3-2}$$

If one uses G to annihilate the l-th entry of a vector $\mathbf{x}$ by a usual plane rotation in the (k,l)-plane, the $G(k,l)$ can be formed by setting

$$\cos\theta = \frac{x_k}{\sqrt{x_k^2 + x_l^2}}$$

$$\sin\theta = \frac{x_l}{\sqrt{x_k^2 + x_l^2}} . \tag{0.3-3}$$

0.3-4 **Notation.** Similarly to (0.2-8), we use

$$G(k, l; x_k, x_l) . \tag{0.3-5}$$

to denote the Givens rotation which uses the k-th entry x_k to zero the l-th entry x_l of a vector $\mathbf{x}$.

0.3-6 **Remark.** In contrast with the hyperbolic rotations [see Remark (0.2-10)], when $k > l$ in (0.3-5), (0.3-2) should be changed to

$$G(k,l;\ x_k,\ x_l) = \begin{bmatrix} 1 & \vdots & & \vdots & \\ \cdots & \cos\theta & \cdots & -\sin\theta & \cdots \\ & \vdots & & \vdots & \\ \cdots & \sin\theta & \cdots & \cos\theta & \cdots \\ & \vdots & & \vdots & 1 \end{bmatrix} \begin{matrix} \\ l \\ \\ k \\ \\ \end{matrix}, \qquad (0.3\text{-}7)$$

but the restriction $|x_k|>|x_l|$ there is removed now. In the following LINPACK algorithm, we need this type of Givens rotation.

To introduce the LINPACK method, suppose one wishes to use orthogonal transformation to downdate a Cholesky factor, i.e., to determine Q with $Q^TQ=I$ such that

$$Q\begin{bmatrix} R \\ \mathbf{0}^T \end{bmatrix} = \begin{bmatrix} D \\ \mathbf{z}^T \end{bmatrix}.$$

Thus it is clear that $R^TR = D^TD+\mathbf{z}\mathbf{z}^T$ and, consequently, the question is how to chose Q such that the last row of $\begin{bmatrix} R \\ \mathbf{0}^T \end{bmatrix}$ becomes $\mathbf{z}^T$ and the rest of the rows still keep an upper triangular form. This means that Q should satisfy the equation

$$(0,\ldots,0,1)Q\begin{bmatrix} R \\ \mathbf{0}^T \end{bmatrix} = \mathbf{z}^T\ ;$$

i.e., the last row of Q, say $(\mathbf{a}^T,\ \alpha) = (a_1,a_2,\ldots,a_n,\alpha)$, should satisfy

$$\mathbf{a}^TR = \mathbf{z}^T$$

$$\text{and} \qquad \alpha = \pm\sqrt{1-||\mathbf{a}||^2}\ .$$

Now $(0,\ldots,0,1)Q = (a_1,a_2,\ldots,a_n,\alpha)$; therefore,

$$Q\begin{bmatrix} a_1 \\ \vdots \\ a_n \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} . \tag{0.3-8}$$

If one chooses Givens rotations to annihilate the a_i's from a_n to a_1, the resulting matrix D should be upper triangular. We thus have the following algorithm.

0.3-9 **Algorithm** *OR*. Given an upper triangular matrix $R \equiv (r_{ij}) \in \mathbb{R}^{n \times n}$ and a vector $\mathbf{z} \equiv \mathbf{z}^{(0)} \in \mathbb{R}^n$ being removed, by using the method described above, the following algorithm computes the downdated Cholesky factor $D \equiv (d_{ij}) \in \mathbb{R}^{n \times n}$ such that $D^T D = R^T R - \mathbf{z}\mathbf{z}^T$.

(1) Solve $R^T \mathbf{a} = \mathbf{z}$.

(2) Compute $\alpha = \sqrt{1 - ||\mathbf{a}||^2}$ (if $1 - ||\mathbf{a}||^2 \leq 0$, stop) .

(3) For $i = n, \ldots, 1$ construct

$$Q_i = G(n+1,\ i;\ \beta_i,\ a_i) ,\quad \beta_n = \alpha.$$

(4) Apply $Q_1, \ldots, Q_n$ to $\begin{bmatrix} R \\ \mathbf{0}^T \end{bmatrix}$, i.e.,

$$Q_1 \cdots Q_n \begin{bmatrix} R \\ \mathbf{0}^T \end{bmatrix} = \begin{bmatrix} D \\ \mathbf{z}^T \end{bmatrix} .$$

This algorithm requires $\frac{5}{2}n^2$ multiplecations and $n+1$ square roots.

0.3-10 **Algorithm** *OR* **in matrix form.** In

$$Q_1 Q_2 \cdots Q_n \begin{bmatrix} R \\ \mathbf{0}^T \end{bmatrix} = \begin{bmatrix} D \\ \mathbf{z}^T \end{bmatrix} \quad , \tag{0.3-11}$$

$$Q_i = G(n+1,\ i;\ \beta_i,\ a_i) \quad , \beta_n = \alpha \quad \left(= \sqrt{1 - ||\mathbf{a}||^2} \right) \ .$$

Hence,

$$\cos\theta_i = \frac{\beta_i}{\beta_{i-1}} \ , \ \sin\theta_i = \frac{a_i}{\beta_{i-1}}$$

and

(0.3-12)

$$\beta_i = \sqrt{1 - a_1^2 - \cdots - a_i^2} \ , \ \beta_0 = 1 \ .$$

For every step i lpile { R Bar sup (i-1) above (ZZ pap sup (i-1)) sup T } } right] maxs left [pile { lpile { R Bar sup (i) above (ZZ pap sup (i)) sup T }} right] raum raum ,

where $\tilde{\mathbf{z}}^{(0)} = \mathbf{0}$ and $\bar{R}^{(0)} = R$, and

$$\bar{R}^{(i)} = \begin{bmatrix} R_1 \\ \vdots \\ R_{i-1} \\ D_i \\ \vdots \\ D_n \end{bmatrix} \quad , \tag{0.3-14}$$

where

$$R = \begin{bmatrix} R_1 \\ \vdots \\ R_n \end{bmatrix} \quad \textit{and} \quad D = \begin{bmatrix} D_1 \\ \vdots \\ D_n \end{bmatrix}$$

are the row partitions.

0.4 Applications to signal processing

The modern evolution of computing machinery, such as VLSI architectures, has now advanced to the stage that it is possible to design processors for solving recursive least squares problems in real-time applied to some adaptive control devices. This thesis was initially motivated by the recent paper of Rader and Steinhardt (1986), in which a so-called sliding windowed recursive least squares (SW-RLS) method was applied to the fixed-order adaptive filtering problem. In the implementation of RLS algorithms, several types of windows, which constitute the effective correlation matrix, may be chosen [Cioffi and Kailath (1985)]. The choice of any particular window depends on the application. For example, in some nonstationary signal processing with slowly changing statistics, the SW-RLS method will be a good choice, as is explained in detail by Rader and Steinhardt (1986). In the sliding window method, in order to form the moving rectangular correlation matrix one must delete old observations as well as insert new ones, thereby recursively updating the filter coefficients.

The following material explains how the SW-RLS method works in fixed-order adaptive filtering.

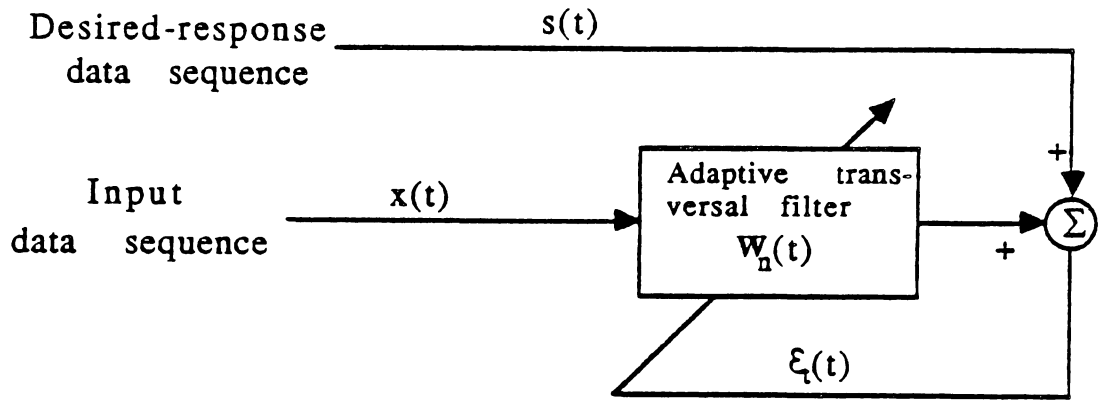

Figure 0.4-1 The adaptive filter

Figure (0.4-1) is the basic configuration of adaptive filtering. For simplicity we assume that the signal $x(t)$ is deterministic and discrete [Markhoul (1975)]. The adaptive transversal filter forms a time-varying linear combination of previous and current input data samples, $x(t),\ldots,x(t-n+1)$, to estimate the current desired response data sample $s(t)$. The estimation error is denoted by

$$\epsilon_t(t) = s(t) - \mathbf{x}^T(t)\mathbf{w}_n(t) = s(t) - (x(t),\ldots,x(t-n+1))\begin{pmatrix} w_1(t) \\ \vdots \\ w_n(t) \end{pmatrix} \quad (0.4\text{-}2)$$

where $w_1(t),\ldots,w_n(t)$ are the current filter coefficients. Actually, for a window of length m, this desired filter response $\mathbf{w}_n(t)$ minimizes

$$\xi_{m,n}(t) = \sum_{\tau=t-m+1}^{t} \Omega_t(\tau)\epsilon_t^2(\tau) \quad , \quad (0.4\text{-}3)$$

where $\Omega_t(\tau)$ is a so-called windowing function [Cioffi and Kailath (1985)]. If a sliding window is used, then m is fixed and

$$\Omega_t(\tau) = \begin{cases} 1 & 0 \le \tau \le t \\ 0 & \textit{elsewhere} \end{cases} . \tag{0.4-4}$$

One could rewrite (0.4-2) to (0.4-4) in more familiar least squares problem form:

$$\min_{\mathbf{w}_n(t)} \; ||X(t)\mathbf{w}_n(t) - \mathbf{s}_m(t)||^2 \;,$$

where

$$X(t) = \begin{bmatrix} x(t) & x(t-1) & \dots & x(t-n+1) \\ x(t-1) & x(t-2) & \dots & x(t-n) \\ \vdots & \vdots & & \vdots \\ x(t-m+2) & x(t-m+1) & \dots & x(t-m-n+3) \\ x(t-m+1) & x(t-m) & \dots & x(t-m-n+2) \end{bmatrix}$$

and

$$\mathbf{s}_m(t) = \begin{bmatrix} s(t) \\ \vdots \\ s(t-m+1) \end{bmatrix} .$$

When the discrete time t increases to $t+1$, the window $x(t)$ moves ahead in time one sample. Instead of resolving the new *LS* problem

$$\min_{\mathbf{w}_n(t+1)} \; ||X(t+1)\mathbf{w}_n(t+1) - \mathbf{s}_m(t+1)||^2$$

one can add a new row,

$$(\mathbf{x}^T(t+1),\; s(t+1) = (x(t+1),\; x(t),\dots,x(t-n+2),\; s(t+1))$$

then delete an old row,

$$(\mathbf{x}^T(t-m+1), s(t-m+1))$$

$$= (x(t-m+1),\ x(t-m),\ldots,x(t-m-n+2),\ s(t-m+1))$$

in problem

$$LS(X(t)|\mathbf{s}_m(t))$$

to update the filter coefficients $\mathbf{w}_n(t)$ to $\mathbf{w}_n(t+1)$. The reader is referred to section (0.1-6) for the symbol $LS(\cdot)$.

The procedure is that in every time step we first update the Cholesky factor

$$\mathbf{Q}(X(t)|\mathbf{s}_m(t))$$

(the reader is referred to (0.1-7) for the symbol $\mathbf{Q}$) to

$$\mathbf{Q}\begin{bmatrix} x(t+1) & s(t+1) \\ X(t) & \mathbf{s}_m(t) \end{bmatrix} = \begin{bmatrix} R & \mathbf{s} \\ & \rho \end{bmatrix},$$

then downdate $\begin{bmatrix} R & \mathbf{s} \\ & \rho \end{bmatrix}$ by either LINPACK or the hyperbolic method to

$$\mathbf{Q}(X(t+1)|\mathbf{s}_m(t+1)) = \begin{bmatrix} \bar{R} & \bar{\mathbf{s}} \\ & \bar{\rho} \end{bmatrix}.$$

Finally $\bar{\mathbf{w}}_n(t+1) = \bar{R}^{-1}\bar{\mathbf{s}}$ and $\xi_{m,n}(t+1) = \bar{\rho}^2$.

As discussed in Cioffi and Kailath (1985), a large window length m implies slower tracking of the estimates since new data added to the window and old data deleted from the window have relatively less influence in comparison to the influence of all the data remaining in the window. With this in mind, in Chapter 3 we will see that a large m implies well-conditioning in the downdating step.

Applications of downdating Cholesky factors are not limited to downdating least squares problems only. In fact, the classical paper on rank-one modification by Gill, Golub, Murray and Saunders (1974) is based on the applications to problems such as quasi-Newton methods for unconstrained optimization [Gill and Murray (1972)], large-scale linear programming [Saunders (1972)], and quadratic programming [Golub and Saunders (1970)] including least squares problems.

It is interesting to note that hyperbolic rotations have been used in signal processing for quite a while; one early paper was published by Kailath, Vieira and Morf (1978) and another by Morf and Delosme (1981).

0.5 Outline of this thesis

The central purpose of this thesis is to deal with downdating problem (0.1-12). Two algorithms are available for solving this problem, the LINPACK method and the hyperbolic method. Although the hyperbolic method is faster than the LINPACK method, the commonly-held opinion is that the hyperbolic method is not stable, due to the subtraction in the denominators of (0.2-7). This potential instability makes some people reluctant to use it. Stewart (1979) did a complete error analysis of the LINPACK method and concluded that it is backward stable. As for the hyperbolic method, the literature shows that researchers have only intuitive feelings about its potential instability, and that the few extant data examples are insufficient to confirm

or deny them.

As we mentioned in Section 0.4, increasing demands for real-time computation force us to investigate fully the numerical stability of these fast algorithms. Chapter 2 contains a complete forward error analysis of the hyperbolic method used in the downdating problem; it turns out that the method is forward stable or, so-called, weakly stable. Moreover, our data testing strongly supports the theoretical stability results. In other words, the hyperbolic rotation method used in the downdating problem has the same order of accuracy as the LINPACK method. The inaccuracy of the answer is mainly caused by ill-conditioning of the problem. Similar to the problem of subtracting two numbers, the downdating problem itself may easily be ill-conditioned. In our opinion, therefore, the subtractions in the denominators (0.2-7) are inherent in the downdating problem.

As for complexity, the hyperbolic method requires only $2n^2$ multiplications while LINPACK requires $\frac{5}{2}n^2$ multiplications. In chapter 3 we present two new algorithms for solving the downdating problem, each of which requires only $\frac{3}{2}n^2$ multiplications. Surprisingly, our data testing (Chapter 4) strongly suggested that one of these new algorithms is at least as stable as the LINPACK method. We haven't done any error analysis on the new algorithm yet, but the method appears to be promising. This new algorithm is the second important result of this thesis. Recall that the first major result is

the stability analysis of the hyperbolic method.

The vector $\mathbf{a}$, the solution of the triangular system $\mathbf{a}^T R = \mathbf{z}^T$ [see (1) in (0.3-9)], plays a central role in almost every result in this thesis. Therefore, chapter 1 is devoted to proving some preliminary results about the vector $\mathbf{a}$.

The final chapter reports some conclusions and indicates the direction of possible future research.

CHAPTER 1: PRELIMINARY RESULTS

The vector $\mathbf{a}$, the solution of the triangular system $\mathbf{a}^T R = \mathbf{z}^T$, plays a key role in downdating problem (0.1-12). In this chapter we list the preliminary results about the vector $\mathbf{a}$, which are used in the heart of this thesis, chapters 2 and 3. In the first section we introduce several basic properties of $\mathbf{a}$ and derive the formulas (in terms of $\mathbf{a}$) for hyperbolic rotations used in Algorithm HY. In section 1.3 we give the explicit expressions of Q and H in Algorithm *OR* and Algorithm *HY*. Actually we prove a more general result in section 1.2, which shows that the inherent connection between these two algorithms and the intimate relation between orthogonal transformations and pseudo-orthogonal transformations. To introduce the explicit expressions of Q and H, in section 1.2 we derive an explicit expression of A such that $A^T A = I - \mathbf{a}\,\mathbf{a}^T$, which actually leads to new downdating algorithms in Chapter 3. Section 1.4 is especially devoted to providing a sharp bound, used in chapter 2, for the product of the norms of hyperbolic rotations applied to Algorithm *HY*; again that bound is in terms of $||\mathbf{a}||$.

1.1. The vector a

From Section 0.3 it follows that the vector $\mathbf{a}$, obtained by solving the triangular system

$$\mathbf{a}^T R = \mathbf{z}^T \quad , \tag{1.1-1}$$

is important in the LINPACK method. In this section we will establish some

notation and derive several basic results about vector $\mathbf{a}$.

1.1-2 **Lemma.** R and $\mathbf{z}$ are assumed to be as in downdating problem (0.1-12). Then $R^T R - \mathbf{z}\mathbf{z}^T > 0$ if and only if $||\mathbf{a}|| < 1$.

Proof. The equation

$$R^T R - \mathbf{z}\mathbf{z}^T = R^T(I - \mathbf{a}\mathbf{a}^T)R$$

implies that $R^T R - \mathbf{z}\mathbf{z}^T > 0$ if and only if $I - \mathbf{a}\mathbf{a}^T > 0$. By direct computation it follows that the eigenvalues of $I - \mathbf{a}\mathbf{a}^T$ are $1 - ||\mathbf{a}||^2$ and 1 with multiplicity of $n-1$. The lemma follows immediately. □

Before going to the next important lemma, we must first establish some useful notation and formulas.

One can rewrite the triangular system (1.1-1) as

$$(a_1, a_2, \ldots, a_n) \begin{bmatrix} R_1 \\ R_2 \\ \vdots \\ R_n \end{bmatrix} = (z_1, z_2, \ldots, z_n) , \tag{1.1-3}$$

where R_i is the ith row of R considered as a 1-by-n matrix. If we define

$$\left(\bar{\mathbf{z}}^{(0)}\right)^T = \mathbf{z}^T = (z_1, z_2, \ldots, z_n) \tag{1.1-4}$$

and

$$\left(\bar{\mathbf{z}}^{(k)}\right)^T = \left(\bar{\mathbf{z}}^{(k-1)}\right)^T - a_k R_k , \tag{1.1-5}$$

then it is easy to verify the equation below:

$$a_k = \frac{z_k - a_1 r_{1k} - \cdots - a_{k-1} r_{k-1,k}}{r_{kk}} = \frac{\bar{z}_k^{(k-1)}}{r_{kk}} \quad . \tag{1.1-6}$$

Actually this is just the forward substitution process. On the other hand, in Algorithm *HY*, the vectors $\mathbf{z}^{(k)}$ $(k=0,\ldots,n)$ are defined by (0.2-16); more precisely

$$\mathbf{z}^{(0)} = \mathbf{z} \tag{1.1-7}$$

and

$$\left(\mathbf{z}^{(k)}\right)^T = \frac{r_{kk}}{\sqrt{r_{kk}^2 - (z_k^{(k-1)})^2}} \left(\mathbf{z}^{(k-1)}\right)^T - \frac{z_k^{(k-1)}}{\sqrt{r_{kk}^2 - (z_k^{(k-1)})^2}} R_k \tag{1.1-8}$$

$$k=(1,\ 2,\ldots,n) \quad .$$

Finally, by Lemma (1.1-2) if $R^T R - \mathbf{z}\mathbf{z}^T > 0$, one can define

$$\begin{aligned} \beta_0 &= 1 \\ \beta_k &= \sqrt{1 - a_1^2 - \cdots - a_k^2} \quad \text{,for } k=1,\ldots,n \quad . \end{aligned} \tag{1.1-9}$$

Now we are ready to prove the following basic lemma.

1.1-10 **Lemma.** Under the notation above, if

$R^T R - \mathbf{z}\mathbf{z}^T > 0$; then

(1) $$\beta_k \mathbf{z}^{(k)} = \bar{\mathbf{z}}^{(k)}, \quad k=0,1,\ldots,n \quad , \tag{1.1-11}$$

(2) $$\frac{z_k^{(k-1)}}{r_{kk}} = \frac{a_k}{\beta_{k-1}} \quad \text{and} \quad \left|\frac{a_k}{\beta_{k-1}}\right| < 1 \quad , \tag{1.1-12}$$

(3) $$\cosh\theta_k = \frac{r_{kk}}{\sqrt{r_{kk}^2 - (z_k^{(k-1)})^2}} = \frac{\beta_{k-1}}{\beta_k} \quad , \tag{1.1-13}$$

$$\sinh\theta_k = \frac{z_k^{(k-1)}}{\sqrt{r_{kk}^2 - (z_k^{(k-1)})^2}} = \frac{a_k}{\beta_k} \quad . \tag{1.1-14}$$

Proof. We first prove (1) by mathematical induction on k. According to our notation, $k=0$ is trivial.

Now assume that

$$\beta_{k-1}\mathbf{z}^{(k-1)} = \bar{\mathbf{z}}^{(k-1)} \quad . \tag{1.1-15}$$

Then (1.1-5) becomes

$$\left(\bar{\mathbf{z}}^{(k)}\right)^T = \beta_{k-1}\left(\mathbf{z}^{(k-1)}\right)^T - a_k R_k \quad , \tag{1.1-16}$$

and (1.1-6) becomes

$$\beta_{k-1}\left(z_k^{(k-1)}\right) = a_k r_{kk} \quad ;$$

consequently,

$$\frac{z_k^{(k-1)}}{r_{kk}} = \frac{a_k}{\beta_{k-1}} \quad . \tag{1.1-17}$$

If one rearranges (1.1-8) and uses (1.1-17), then it turns out that

$$\beta_k\left(\mathbf{z}^{(k)}\right)^T = \beta_{k-1}\left(\mathbf{z}^{(k-1)}\right)^T - a_k R_k \quad .$$

Comparing this with (1.1-16) yields (1). Since this is an inductive proof, every formula derived in the proof is also true for all k. Therefore, (2) is also proved by (1.1-17) and (3) follows directly. The only thing left is to prove

$$\left|\frac{a_k}{\beta_{k-1}}\right| < 1 \quad ,$$

independently of (3). Actually

$$\begin{aligned} a_k^2 &\le a_k^2 + (a_1^2 + \cdots + a_{k-1}^2)(1 - a_1^2 - \cdots - a_k^2) \\ &= (a_1^2 + \cdots + a_k^2)(1 - a_1^2 - \cdots - a_{k-1}^2); \end{aligned}$$

thus,

$$\frac{a_k}{\beta_{k-1}} \le \sqrt{a_1^2 + \cdots + a_k^2} \le \|\mathbf{a}\| < 1 \ . \qquad \square$$

1.1-18 **Remark.** According to the proof of (2) in Lemma (1.1-10), we only need to assume $R^T R - \mathbf{z}\mathbf{z}^T > 0$ to guarantee $|z_k^{(k-1)}| < |r_{kk}|$; therefore, Algorithm *HY* is valid under the same assumption [see (0.2-19)].

The following lemma shows the connection between the LINPACK method and the hyperbolic method.

1.1-19 **Lemma.** In Algorithm *OR* (0.3-10)

$$\cos\theta_k = \frac{\beta_k}{\beta_{k-1}} \ , \quad \sin\theta_k = \frac{a_k}{\beta_{k-1}} \ , \qquad (1.1\text{-}20)$$

where each β_k is defined by (1.1-9), and each $\tilde{\mathbf{z}}^{(k)}$ defined by (0.3-13) actually is the same as $\mathbf{z}^{(n-k)}$ defined by (0.2-16) in Algorithm *HY*, i.e.,

$$\tilde{\mathbf{z}}^{(k)} = \mathbf{z}^{(n-k)} \ , \quad k = 0, \ldots, n \ . \qquad (1.1\text{-}21)$$

Proof. Under the notation define by (1.1-5) and knowing that the Q_k's in Algorithm *OR* are constructed by

$$Q_1 \cdots Q_n \begin{pmatrix} \mathbf{a} \\ \beta_n \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \ ,$$

from the bottom to top, then (1.1-20) follows.

To prove (1.1-21), first recall from (0.3-13) that

$$\left(\tilde{\mathbf{z}}^{(k)}\right)^T = \frac{a_k}{\beta_{k-1}} R_k + \frac{\beta_k}{\beta_{k-1}} \left(\tilde{\mathbf{z}}^{(k-1)}\right)^T \ . \qquad (1.1\text{-}22)$$

Using Lemma (1.1-10) one can rewrite (1.1-8) as

$$\left(\mathbf{z}^{(k)}\right)^T = \frac{\beta_{k-1}}{\beta_k}\left(\mathbf{z}^{(k-1)}\right)^T - \frac{a_k}{\beta_k}R_k \quad ; \tag{1.1-23}$$

that is,

$$\left(\mathbf{z}^{(k-1)}\right)^T = \frac{a_k}{\beta_{k-1}}R_k + \frac{\beta_k}{\beta_{k-1}}\left(\mathbf{z}^{(k)}\right)^T \quad . \tag{1.1-24}$$

Compare (1.1-24) and (1.1-22) and notice $\tilde{\mathbf{z}}^{(0)} = \mathbf{z}^{(n)} = \mathbf{0}$; then (1.1-21) follows by induction. □

Let us now summarize what we have. First of all, we proved that all the transformations used in both algorithms *OR* and *HY* can be expressed via the vector $\mathbf{a}$. While the LINPACK method replaces the rows of R by rows of D from bottom to top, the hyperblic method replaces the rows of R by rows of D from top to bottom. At the same time, while Q_k transforms $\mathbf{z}^{(k)}$ to $\mathbf{z}^{(k-1)}$, $(k=n,\ldots,1)$, P_k transfers $\mathbf{z}^{(k-1)}$ to $\mathbf{z}^{(k)}$,$(k=1,\ldots,n)$. There are two basic formulas. For the LINPACK method,

$$\begin{aligned} D_k &= \frac{\beta_k}{\beta_{k-1}}R_k - \frac{a_k}{\beta_{k-1}}\left(\mathbf{z}^{(k)}\right)^T \\ \left(\mathbf{z}^{(k-1)}\right)^T &= \frac{a_k}{\beta_{k-1}}R_k + \frac{\beta_k}{\beta_{k-1}}\left(\mathbf{z}^{(k)}\right)^T \quad . \end{aligned} \tag{1.1-25}$$

For the hyperbolic method,

$$\begin{aligned} D_k &= \frac{\beta_{k-1}}{\beta_k}R_k - \frac{a_k}{\beta_k}\left(\mathbf{z}^{(k-1)}\right)^T \\ \left(\mathbf{z}^{(k)}\right)^T &= -\frac{a_k}{\beta_k}R_k + \frac{\beta_{k-1}}{\beta_k}\left(\mathbf{z}^{(k-1)}\right)^T \quad . \end{aligned} \tag{1.1-26}$$

In Chapter 3 we will give a complete list of the Cholesky factor downdating algorithms by using this kind of expression.

Secondly, from Lemma (1.1-10) (1), we derived a very important connection between downdating and solving the system $\mathbf{a}^T R = \mathbf{z}^T$; that is,

$$\left(\beta_k \mathbf{z}^{(k)}\right)^T = \left(\beta_{k-1}\mathbf{z}^{(k-1)}\right)^T - a_k R_k \ . \tag{1.1-27}$$

This will actually lead us to a new algorithm which incorporates the two different processes, downdating the Cholesky factor and solving the system $\mathbf{a}^T R = \mathbf{z}^T$.

1.2 The Basic Equation $AR = D$

In this section an explicit expression for the Cholesky factor, A, of $I - \mathbf{a}\mathbf{a}^T = A^T A$ will be given in terms of vector $\mathbf{a}$; then theoretically the downdated Cholesky factor D can be obtained by multiplying A by the old Cholesky factor R.

1.2-1 **Lemma.** Let the vector $\mathbf{a} \in \mathbb{R}^n$ and $\|\mathbf{a}\| < 1$. Then the Cholesky factor A of $I - \mathbf{a}\mathbf{a}^T$ is given by the following expression:

$$A = \begin{bmatrix} \frac{\beta_1}{\beta_0} & -\frac{a_1 a_2}{\beta_0 \beta_1} & -\frac{a_1 a_3}{\beta_0 \beta_1} & \cdots & -\frac{a_1 a_n}{\beta_0 \beta_1} \\ & \frac{\beta_2}{\beta_1} & -\frac{a_2 a_3}{\beta_1 \beta_2} & \cdots & -\frac{a_2 a_n}{\beta_1 \beta_2} \\ & & \ddots & \cdots & \vdots \\ & & & & \vdots \\ & & & \frac{\beta_{n-1}}{\beta_{n-2}} & -\frac{a_{n-1} a_n}{\beta_{n-1}\beta_{n-2}} \\ & & & & \frac{\beta_n}{\beta_{n-1}} \end{bmatrix} \tag{1.2-2}$$

with $\beta_0 = 1$, $\beta_k = \sqrt{1 - a_1^2 - \cdots - a_k^2}$ for $k = 1, \ldots, n$.

Proof. The proof proceeds by direct verification. □

1.2-3 **Theorem.** Let $R \in \mathbb{R}^{n \times n}$ be upper triangular and $\mathbf{z} \in \mathbb{R}^n$. Assume $R^T R - \mathbf{z}\mathbf{z}^T > 0$, $D^T D = R^T R - \mathbf{z}\mathbf{z}^T$, and D is the downdated Cholesky factor. Then

$$D = AR \quad , \tag{1.2-4}$$

where $A^T A = I - \mathbf{a}\mathbf{a}^T$ and $\mathbf{a}^T R = \mathbf{z}^T$.

Proof: The theorem follows from Lemma (1.1-1) and the equation

$$\begin{aligned} R^T R - \mathbf{z}\mathbf{z}^T &= R^T (I - \mathbf{a}\mathbf{a}^T) R \\ &= R^T A^T A R \quad , \\ &= D^T D \end{aligned}$$

if we notice the uniqueness of the Cholesky factor of $R^T R - \mathbf{z}\mathbf{z}^T$. □

This actually is one way to downdate the Cholesky factor. Right now it only has theoretical interest. The number of multiplications required to form AR only is $\frac{n^3}{6}$. But because of its theoretical importance, we still list the algorithm below. Later in Chapter 3 we will see how it can be reduced to an algorithm which requires only $\frac{3}{2}n^2$ multiplications (even less than the number of multiplications needed for applying the hyperbolic algorithm).

1.2-5 **Algorithm** AR. Given an upper triangular matrix $R \equiv (r_{ij}) \in \mathbb{R}^{n \times n}$ and a vector $\mathbf{z} \equiv \mathbf{z}^{(0)} \in \mathbb{R}^n$ being removed, by using the basic equation $D = AR$, the following algorithm computes the downdated Cholesky factor $D \equiv (d_{ij}) \in \mathbb{R}^{n \times n}$ such that $D^T D = R^T R - \mathbf{z}\mathbf{z}^T$

(1) Solve $R^T \mathbf{a} = \mathbf{z}$.

(2) $\beta_0 = 1$

For $i = 1,n$

$$\beta_i = \sqrt{\beta_{i-1}^2 - a_i^2} .$$

(3) Form the matrix A (1.2-2):

For $i, j = 1, n$

$$\text{if } i = j, \qquad A_{ij} = \frac{\beta_i}{\beta_{i-1}}$$

$$\text{if } i < j, \qquad A_{ij} = \frac{a_i a_j}{\beta_{i-1}\beta_i}$$

$$\text{else,} \qquad A_{ij} = 0$$

(4) $D = AR$.

This algorithm requires $\frac{n^3}{6} + 0(n^2)$ multiplications and n square roots.

It is also easy to check directly that the expression for A^{-1} below is true.

1.2-6 **Corollary**. Under the assumption of Lemma (1.2-1) A^{-1} exists and is given by

$$A^{-1} = \begin{bmatrix} \frac{\beta_0}{\beta_1} & \frac{a_1a_2}{\beta_1\beta_2} & \frac{a_1a_3}{\beta_2\beta_3} & \cdots & \frac{a_1a_n}{\beta_{n-1}\beta_n} \\ & \frac{\beta_1}{\beta_2} & \frac{a_2a_3}{\beta_2\beta_3} & \cdots & \frac{a_2a_n}{\beta_{n-1}\beta_n} \\ & & \ddots & \cdots & \vdots \\ & & & \frac{\beta_{n-2}}{\beta_{n-1}} & \frac{a_{n-1}a_n}{\beta_{n-1}\beta_n} \\ & & & & \frac{\beta_{n-1}}{\beta_n} \end{bmatrix} . \tag{1.2-7}$$

1.3 More Explicit Expressions

To understand the inherent connection between Algorithm *OR* and Algorithm *HY*, we will derive the explicit expressions of the matrices Q and H such that

$$Q\begin{bmatrix} R \\ \mathbf{0}^T \end{bmatrix} = \begin{bmatrix} D \\ \mathbf{z}^T \end{bmatrix} \tag{1.3-1}$$

and

$$H\begin{bmatrix} R \\ \mathbf{z}^T \end{bmatrix} = \begin{bmatrix} D \\ \mathbf{0}^T \end{bmatrix} . \tag{1.3-2}$$

In fact, a general theorem connecting orthogonal and pseudo-orthogonal matrices is given as follows.

1.3-3 **Theorem.** Let

$$Q = \begin{bmatrix} A & B \\ C^T & D \end{bmatrix} \begin{matrix} \}p \\ \}n-p \end{matrix} \tag{1.3-4}$$

be an arbitrary orthogonal transformation, i.e., $QQ^T = Q^TQ = I$, where $Q \in \mathbb{R}^{n\times n}$ and $A \in \mathbb{R}^{p\times p}$ are both invertible. If

$$Q\begin{pmatrix}x_1\\x_2\end{pmatrix} = \begin{pmatrix}y_1\}p\\y_2\}n-p\end{pmatrix}, \tag{1.3-5}$$

then there exists a corresponding pseudo-orthogonal matrix, H, associated with

$$S = diag(\underbrace{1,\ldots,1}_{p},\underbrace{-1,\ldots,-1}_{n-p}) \tag{1.3-6}$$

such that

$$H\begin{pmatrix}x_1\\y_2\end{pmatrix} = \begin{pmatrix}y_1\\x_2\end{pmatrix}, \tag{1.3-7}$$

where

$$H = \begin{bmatrix}A^{-T} & BD^{-1}\\ -D^{-1}C^T & D^{-1}\end{bmatrix}. \tag{1.3-8}$$

Proof. From (1.3-4) and (1.3-5) one has

$$Ax_1 + Bx_2 = y_1 \tag{1.3-9}$$

and

$$C^Tx_1 + Dx_2 = y_2 . \tag{1.3-10}$$

Rewriting (1.3-10) as

$$-D^{-1}C^Tx_1 + D^{-1}y_2 = x_2 \tag{1.3-11}$$

and substituting (1.3-11) into (1.3-9), one has

$$Ax_1 + B(-D^{-1}C^Tx_1 + D^{-1}y_2) = y_1 ,$$

which is $(A - BD^{-1}C^T)\mathbf{x}_1 + BD^{-1}\mathbf{y}_2 = \mathbf{y}_1$.

To show

$$A - BD^{-1}C^T = A^{-T} \quad , \tag{1.3-12}$$

one needs equations

$$A^TB + CD = 0 \quad \text{and} \quad A^TA + CC^T = I_p \tag{1.3-13}$$

obtained from $Q^TQ = I$.

Rewrite (1.3-13) as $-BD^{-1} = A^{-T}C$; then

$$\begin{aligned} A - BD^{-1}C^T &= A + A^{-T}CC^T \\ &= A + A^{-1}(I_p - A^TA) \\ &= A + A^{-T} - A \\ &= A^{-T} \end{aligned}$$

and (1.3-7) and (1.3-8) are proved.

The last step is to verify that H indeed is a pseudo-orthogonal matrix associated with S (1.3-6). Actually

$$H^TSH = \begin{bmatrix} A^{-1} & CD^{-T} \\ D^{-T}B^T & -D^{-T} \end{bmatrix} \begin{bmatrix} A^{-T} & BD^{-1} \\ -D^{-1}C^T & D^{-1} \end{bmatrix} = S$$

is easy to check by using $QQ^T = I$ and $Q^TQ = I$. For example, $A^{-1}BD^{-1} + CD^{-T}D^{-1} = 0$ and $D^{-T}B^TA^{-T} + D^{-T}D^{-1}C^T = 0$ can be verified by equation

$$AC + BD^T = 0 \quad . \tag{1.3-14}$$

By using (1.3-14) and $AA^T + BB^T = I_p$, one derives

$$A^{-1}A^{-T} - CD^{-T}D^{-1}C = I_p \quad .$$

Finally

$$D^{-T}B^{T}BD^{-1} - D^{-T}D^{-1} = -I_{n-p}$$

is derived by equation $B^{T}B + D^{T}D = I_{n-p}$. □

1.3-15 **Remark.** According to the proof, for given Q, $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, and $\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, the pseudo-orthogonal matrix H is uniquely determined provided that A and D are invertible.

Now we are concerned with the explicit expressions of Q and H in (1.3-1) and (1.3-2).

1.3-16 **Theorem.** The matrices Q and H in (1.3-1) and (1.3-2) are uniquely determined (up to the sign of δ below) by R and $\mathbf{z}$. Moreover,

$$Q = \begin{bmatrix} A & -\frac{A\mathbf{a}}{\delta} \\ \mathbf{a}^{T} & \delta \end{bmatrix} \tag{1.3-17}$$

and

$$H = \begin{bmatrix} A^{-T} & -\frac{A\mathbf{a}}{\delta^2} \\ -\frac{\mathbf{a}^{T}}{\delta} & \frac{1}{\delta} \end{bmatrix}, \tag{1.3-18}$$

where A is given by (1.2-2), A^{-1} is given by (1.2-6), and $\mathbf{a}$ is the solution of $\mathbf{a}^{T}R = \mathbf{z}^{T}$ with $\delta = \pm\sqrt{1-||\mathbf{a}||^2}$.

Proof. One partitions Q conformally with the other matrices in (1.3-1); for example, if

$$Q = \begin{bmatrix} A' & \mathbf{b} \\ \mathbf{c}^{T} & d \end{bmatrix} \begin{matrix} \}n \\ \}1 \end{matrix};$$

then $A'R = D$. Compared with (1.2-3), one has

$$A' = A \quad .$$

From (1.3-1) again one should have

$$\mathbf{c}^T R = \mathbf{z}^T$$

and

$$\mathbf{c} = \mathbf{a} \quad .$$

Since Q is orthogonal, $d = \pm\sqrt{1-||\mathbf{a}||^2} = \delta$.

Again using $QQ^T = I$, one has $A\mathbf{a} + \mathbf{b}\delta = 0$, and consequently

$$\mathbf{b} = -\frac{A\mathbf{a}}{\delta} \quad .$$

Once we proved that (1.3-17) is uniquely determined by (1.3-1), formula (1.3-18) becomes a special case of formular (1.3-8) of Theorem (1.3-3). According to Remark (1.3-15), H is unique, too. □

1.4 A sharp bound for $||H||$

In order to perform an error analysis on Algorithm HY in Chapter 2, we need an upper bound for the spectral norm of the product of hyperbolic rotations $||P_n \cdots P_2 P_1||_2$ [see (0.2-15) and (0.2-18)]. It is easy to check that

$$||P_k||_2 = |c_k| + |s_k| \quad ,$$

where

$$c_k = \cosh\theta_k \quad , \quad s_k = \sinh\theta_k$$

[for detail, see the proof of Lemma (2.2-1)]. By using (1.1-13) and (1.1-14) one

obtains

$$\|P_n \cdots P_2 P_1\|_2 \le \prod_{k=1}^{n} \|P_k\|_2$$

$$= \prod_{k=1}^{n} c_k \left(1 + \frac{|s_k|}{c_k}\right)$$

$$= \frac{1}{\sqrt{1-\|\mathbf{a}\|^2}} \prod_{k=1}^{n} \left(1 + \frac{|a_k|}{\beta_{k-1}}\right) .$$

Therefore, to estimate the upper bound of $\|P_n \cdots P_2 P_1\|_2$, one needs to find the upper bound of the following product:

$$\prod_{k=1}^{n} \left(1 + \frac{|a_k|}{\beta_{k-1}}\right) = (1 + |a_1|)\left(1 + \frac{|a_2|}{\sqrt{1-a_1^2}}\right) \cdot \qquad (1.4\text{-}1)$$

$$\cdot \left(1 + \frac{|a_3|}{\sqrt{1-a_1^2 - a_2^2}}\right) \cdots \left(1 + \frac{|a_n|}{\sqrt{1-a_1^2-\cdots-a_{n-1}^2}}\right) \cdot \cdots$$

A rather surprising result that (1.4-1) has a maximum value. This value M_n can be expressed in terms of the norm of **a** as follows:

$$\max \prod_{k=1}^{n} \left(1 + \frac{|a_k|}{\beta_{k-1}}\right) = \left(1 + \sqrt{1-(1-\|\mathbf{a}\|^2)^{1/n}}\right)^n \equiv M_n \ , \qquad (1.4\text{-}2)$$

Now we want to prove (1.4-2) by using a so-called vector majorization method [see, for example, Hardy, Littlewood and Polya (1934) or Marshall and olkin (1979)]. Precisely, we want to prove the following more general theorem.

1.4-3 **Theorem.** Suppose that $\mathbf{a} = (a_1,\ldots,a_n)^T \in \mathbb{R}^n$ and $||\mathbf{a}|| \leq \theta < 1$ (for a given θ). Then

$$\max\left\{\prod_{i=1}^{n}\left(1 + \frac{a_i}{\sqrt{1-a_1^2-\cdots-a_{i-1}^2}}\right)\right\} = \left(1 + \sqrt{(1-\theta^2)^{1/n}}\right)^n ,$$

and the maximum is reached when $a_1 = \dfrac{a_i}{\sqrt{1-a_1^2-\cdots-a_{i-1}^2}}$ for $i = 2,\ldots,n$.

In order to prove theorem (1.4-3), first we need a short introduction to the vector majorization method. we follow Schur by using doubly stochastic matrices to define vector majorization, a partial ordering of $\mathbb{R}^n$.

1.4-4 **Definition.** A matrix $P \equiv (p_{ij}) \in \mathbb{R}^{n\times n}$ is doubly stochastic if $p_{ij} \geq 0$ for $i,j = 1,\ldots,n$, and if every row sum as well as every column sum is 1, i.e.,

$$\sum_{i=1}^{n} p_{ij} = 1 \quad for \quad j=1,\ldots,n$$

and

$$\sum_{j=1}^{n} p_{ij} = 1 \quad for \quad i=1,\ldots,n \quad .$$

1.4-5 **Definition.** For $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, $\mathbf{x}$ is majorized by $\mathbf{y}$, denoted as $\mathbf{x} \prec \mathbf{y}$, if there exists a doubly stochastic matrix P such that

$$\mathbf{x} = P\mathbf{y} \quad . \tag{1.4-6}$$

1.4-7 **Example.** In $\mathbb{R}^3$,

$$\left(\frac{1}{3},\ \frac{1}{3},\ \frac{1}{3}\right)^T \prec (1,0,0)^T$$

since

$$\begin{pmatrix}\frac{1}{3}\\ \frac{1}{3}\\ \frac{1}{3}\end{pmatrix} = \begin{bmatrix}\frac{1}{3} & \frac{1}{3} & \frac{1}{3}\\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3}\\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3}\end{bmatrix} \begin{pmatrix}1\\0\\0\end{pmatrix} .$$

Generally, in $\mathbb{R}^n$ one has

$$(\bar{x},\ldots,\bar{x}) \prec (x_1,\ldots,x_n) \tag{1.4-8}$$

with

$$\bar{x} = \frac{x_1+x_2+\cdots+x_n}{n}$$

since

$$\begin{bmatrix}\bar{x}\\ \vdots\\ \bar{x}\end{bmatrix} = \begin{bmatrix}\frac{1}{n} & \cdots & \frac{1}{n}\\ \vdots & & \vdots\\ \frac{1}{n} & \cdots & \frac{1}{n}\end{bmatrix} \begin{bmatrix}x_1\\ \vdots\\ x_n\end{bmatrix} . \tag{1.4-9}$$

1.4-10 **Example.** Any permutation matrix (obtained by exchanging any rows or columns of the identity matrix) is doubly stochastic.

Therefore, $(x_1,x_2)^T \prec (x_2,x_1)^T$ and $(x_2,x_1)^T \prec (x_1,x_2)^T$ are both true.

Notice that in $\mathbb{R}$, $a \prec b$ if and only if $a=b$; therefore, in this section we are concerned with $\mathbb{R}^n$, $n \geq 2$.

1.4-11 **Definition.** A function $\phi : I^n \to \mathbb{R}$ is called *Schur-convex (Schur-concave)* on I^n where I is an open interval of $\mathbb{R}$ and I^n is the corresponding open n-

dimension box in $\mathbb{R}^n$, if

$$\phi(x) \leq \phi(y) \quad (\phi(x) \geq \phi(y), \text{ correspondingly})$$

whenever $\mathbf{x} \prec \mathbf{y}$ for any $\mathbf{x}, \mathbf{y} \in I^n$.

Observe that Schur-convex and Schur-concave functions are always symmetric functions on some symmetric set since the permutation matrices are all doubly stochastic [Example (1.4-10)].

Finally, we introduce the following theorem without proof.

1.4-12 **Theorem.** [Schur (1923), Ostrowski (1952)]. Let $I \subseteq \mathbb{R}$ be an open interval and $\phi : I^n \to \mathbb{R}$ be a continuously differentiable function. Then ϕ is Schur-convex (Schur-concave) if and only if

(1) ϕ is symmetric on I^n;

(2) $(x_1 - x_2)\left(\dfrac{\partial \phi}{\partial x_1} - \dfrac{\partial \phi}{\partial x_2}\right) \geq 0$ (≤ 0, correspondingly) for all $\mathbf{x} \in I^n$.

Now we can proceed with our proof of Theorem (1.4-3) by several limmas.

1.4-13 **Lemma.** The function

$$F(\mathbf{t}) = \sum_{i=1}^{n} t_i^2 - \sum_{i \neq j} t_i^2 t_j^2 + \cdots + (-1)^{n-1} t_1^2 t_2^2 \cdots t_n^2$$

is a Schur-convex function on $(-1,1)^n$.

Proof. According to Theorem (1.4-12), one only needs to check that

$$(t_1 - t_2)\left(\frac{\partial F}{\partial t_1} - \frac{\partial F}{\partial t_2}\right) \geq 0 \ , \tag{1.4-14}$$

where $\mathbf{t} \in I^n$ and $I = (-1,1)$, since the symmetry of $F(\mathbf{t})$ on $(-1,1)^n$ is obvious.

First it is claimed that

$$\frac{\partial F}{\partial t_1} - \frac{\partial F}{\partial t_2} = 2(t_1 - t_2)(1+t_1t_2)(1-t_3^2)\cdots(1-t_n^2) \tag{1.4-15}$$

for $n \geq 2$.

For $n=2$, (1.4-15) is easy to verify. To proceed with induction on n one needs an equality,

$$F(t_1,\ldots,t_n) = F(t_1,\ldots,t_{n-1}) + t_n^2(1-t_1^2)\cdots(1-t_{n-1}^2) \ , \tag{1.4-16}$$

which is obvious.

Thus,

$$\begin{aligned}
&\frac{\partial F}{\partial t_1} - \frac{\partial F}{\partial t_2} \\
&= \frac{\partial F(t_1,\ldots,t_{n-1})}{\partial t_1} - \frac{\partial F(t_1,\ldots,t_{n-1})}{\partial t_2} + \\
&+ 2t_2(1-t_1^2)(1-t_3^2)\cdots(1-t_{n-1}^2)t_n^2 - 2t_1(1-t_2^2)\cdots(1-t_{n-1}^2)t_n^2 \\
&= 2(t_1-t_2)(1+t_1t_2)(1-t_3^2)\cdots(1-t_n^2) \ ,
\end{aligned}$$

and (1.4-15) is proved. Therefore, (1.4-14) is true and the lemma is proved.

□

1.4-17 **Lemma.** The function

$$Q(\mathbf{t}) = \prod_{i=1}^{n}(1+t_i)$$

is Schur-concave on $(-1,\infty)^n$.

Proof. Again $Q(\mathbf{t})$ is symmetric about $t_1,\ldots,t_n$; therefore, one only needs to check condition (2) in Theorem (1.4-12). By computation one knows

$$\frac{\partial Q}{\partial t_1} - \frac{\partial Q}{\partial t_2} = (1+t_3)\cdots(1+t_n)(t_2-t_1);$$

thus,

$$(t_1-t_2)\left[\frac{\partial Q}{\partial t_1} - \frac{\partial Q}{\partial t_2}\right] = -(t_1-t_2)^2(1+t_3)\cdots(1+t_n) \le 0 \quad \text{for } \textit{all} \quad t_i \in (-1,\infty). \ \square$$

Before introducing the next Lemma we suggest another symbol, which will be frequently used in the following text. The function $\bar{Q}$ is defined by $\bar{Q} = \bar{Q}(t) = Q(t,t,\ldots,t)$, where Q is a vector variable function.

1.4-18 **Lemma.** For the following nonlinear programming problem:

$$\max \prod_{i=1}^{n}(1+t_i) \ ,$$

subject to

$$0 \le F(\mathbf{t}) = \sum_i t_i^2 - \sum_{i\neq j} t_i^2 t_j^2 + \cdots + (-1)^{n-1} t_1^2 t_2^2 \cdots t_n^2 \le \theta^2 < 1 \qquad (1.4\text{-}19)$$

and

$$0 \le t_i < 1 \ ;$$

the optimal value $(1+\tilde{t}\,)^n$ with $\tilde{t} = \sqrt{1-(1-\theta^2)^{1/n}}$ is reached when $\tilde{\mathbf{t}} = (\tilde{t},\ldots,\tilde{t}\,)$.

Proof. Let $t \in \mathbb{R}^n$ be a vector satisfying (1.4-19) and

$$\bar{\mathbf{t}} = (\bar{t},\ldots,\bar{t}) \text{ with } \bar{t} = \left(\frac{t_1 + \cdots + t_n}{n}\right) \text{ and } \mathbf{t} = (t_1,\ldots,t_n) \ .$$

By Lemma (1.4-17), $\prod_{i=1}^{n}(1+t_i)$ is Schur-concave on $(-1,+\infty)^n$; therefore

$$\prod_{i=1}^{n}(1+t_i) \leq (1+\bar{t})^n \ . \tag{1.4-20}$$

Notice that vector $\bar{\mathbf{t}}$ need not satisfy (1.4-19); even $0 \leq \bar{t} < 1$ is guaranteed by $\bar{t} = \left(\frac{t_1+\cdots+t_n}{n}\right)$ and $0 \leq t_i < 1$. But Lemma (1.4-14) implies that

$$F(\bar{\mathbf{t}}) \leq F(\mathbf{t}) \ .$$

On the other hand,

$$\begin{aligned} F(\bar{\mathbf{t}}) &= C_n^1(\bar{t})^2 - C_n^2(\bar{t})^4 + \cdots + (-1)^{n-1}C_n^n(\bar{t})^{2n} \\ &= 1 - (1-(\bar{t})^2)^n \ . \end{aligned}$$

We define a function

$$\bar{F}(x) = 1 - (1-x^2)^n \tag{1.4-21}$$

and let $\tilde{t}$ be such that $\bar{F}(\tilde{t}) = \theta^2$, i.e. $\tilde{t}$ is the solution of $\bar{F}(x) = \theta^2$; actually

$$\tilde{t} = \sqrt{1 - (1-\theta^2)^{1/n}} \ , \tag{1.4-22}$$

if we restrict $0 \leq \tilde{t} < 1$.

Summarizing from the beginning of this proof, we have

$$\bar{F}(\bar{t}) = F(\bar{\mathbf{t}}) \leq F(\mathbf{t}) \leq \theta^2 = \bar{F}(\tilde{t}) \tag{1.4-23}$$

for any $\mathbf{t}$ satisfying (1.4-19).

By directly checking the derivative, we know that $\bar{F}(x)$ is a monotonic increasing function on [0,1], and so is the inverse of $\bar{F}(x)$. Thus, from (1.4-23) one has

$$\bar{t} \leq \tilde{t} \quad .$$

Finally we proved

$$\sum_{i=1}^{n}(1+t_i) \leq (1+\bar{t})^n \leq (1+\tilde{t})^n \tag{1.4-24}$$

with an arbitrary $\mathbf{t}$ satisfying (1.4-19). From (1.4-22) we know $\tilde{\mathbf{t}} = (\tilde{t},\ldots,\tilde{t})$ satisfies (1.4-19) also; therefore this lemma is proved. □

Finally Theorem (1.4-3) can be proved.

Proof of Theorem (1.4-3). In Lemma (1.4-18) let

$$t_i = \frac{a_i}{\sqrt{1-a_1^2-\cdots-a_{i-1}^2}} \quad , \quad i=2,\ldots,n \tag{1.4-25}$$

and

$$t_1 = a_1.$$

Notice that as the maximum value in (1.4-2) is of interest, we need only to consider the case, when all $a_i \geq 0$.

But it is easy to check that $a_i \geq 0$ and $||\mathbf{a}|| < 1$ are equivalent to $0 \leq t_i < 1$ for $i=1,\ldots,n$, if we observe that

$$\begin{aligned} a_i^2 &\leq a_i^2 + (a_1^2+\cdots+a_{i-1}^2)(1 - a_1^2-\cdots-a_i^2) \\ &= (a_1^2+\cdots+a_i^2)(1 - a_1^2-\cdots-a_{i-1}^2) \quad . \end{aligned}$$

We also need to check that $0 \le ||\mathbf{a}|| \le \theta$ is equivalent to $0 \le F(\mathbf{t}) \le \theta^2$. By using (1.4-25), it comes from direct computing that

$$a_1^2 + \cdots + a_n^2 = t_1^2 + t_2^2(1-t_1^2) + \cdots + t_n^2(1-t_1^2)(1-t_2^2)\cdots(1-t_{n-1}^2) = F(\mathbf{t}) \ .$$

Therefore, the two nonlinear programming problems (1.4-2) and (1.4-19) are equivalent and Theorem (1.4-3) is proved. □

An alternative proof can be found in Pan and Sigmon (1987).

The following Table (1.4-26) shows how the bound M_n in (1.4-2) is a significantly tighter bound than 2^n unless $||\mathbf{a}||$ is very near to 1.

Table 1.4-26 **The approximate value of** M_n

n \ $\|\mathbf{a}\|$	$1\text{-}10^{-1}$	$1\text{-}10^{-2}$	$1\text{-}10^{-4}$	$1\text{-}10^{-6}$	$1\text{-}10^{-8}$	1
10	27	91	280	482	649	1024
20	144	1138	1_{10^4}	4_{10^4}	9_{10^4}	1_{10^6}
30	524	8103	2_{10^5}	1_{10^6}	5_{10^6}	1_{10^9}
40	1553	4266	2_{10^6}	2_{10^7}	1_{10^8}	$1_{10^{12}}$
60	1_{10^4}	7_{10^4}	1_{10^8}	4_{10^9}	$5_{10^{10}}$	$1_{10^{18}}$
80	5_{10^4}	7_{10^6}	4_{10^9}	$3_{10^{11}}$	$6_{10^{12}}$	$1_{10^{24}}$
100	2_{10^5}	6_{10^7}	$8_{10^{10}}$	$1_{10^{13}}$	$5_{10^{11}}$	$1_{10^{30}}$

CHAPTER 2: ERROR ANALYSIS OF HYPERBOLIC ROTATIONS

In this chapter we perform a complete forward error analysis on the hyperbolic rotations algorithm for the downdating problem (Algorithm *HY*). In section 2.1 we discuss the perturbation theory for downdating problem (0.1-12). The results in section 2.1 are largely due to Stewart (1979); we use them in section 2.3 to conclude that the hyperbolic method is forward stable. The heart of this chapter is section 2.2, in which the forward error of hyperbolic rotations for the downdating problem is bounded in terms of the condition number of the problem. This bound can be employed in many other algorithms in which hyperbolic rotations are used. In section 2.3, we also compare the forward error bound of the LINPACK method with that of the hyperbolic method, and find that they are of comparable size in terms of the condition number of the problem. We then conclude that the hyperbolic method is forward stable since the LINPACK method has been shown to be backward stable by Stewart (1979).

We adopt the symbol $fl(e)$ to denote the result of a floating point operation performed on an arithmetic expression e with a specified order of evaluation [Wilkinson (1965)]. For a matrix $A \in \mathbb{R}^{m \times n}$, $||\cdot||_F$ denotes the Frobenius norm defined by

$$||A||_F^2 = \sum_{i,j} a_{ij}^2 \quad .$$

Here $||\cdot||_2$ denotes the spectral norm defined by

$$||A||_2^2 = \sigma_1^2 \ ,$$

where σ_1^2 is the largest eigenvalue of $A^T A$, i.e., σ_1 is the largest singular value of A.

2.1 Conditioning

As discussed in detail by Stewart (1979), downdating problem (0.1-12) has at least the potential for being ill-conditioned. Obviously when the matrix $R^T R - \mathbf{z}\mathbf{z}^T$ is not positive definite, its Cholesky factorization does not even exist. As we know from Lemma (1.1-2), in this case vector $\mathbf{a}$ has as its norm $||\mathbf{a}|| \geq 1$. One might guess that when $||\mathbf{a}|| < 1$ and tends to be very close to 1, then the associated downdating problem (0.1-12) must be ill-conditioned. Stewart (1979) proved this is indeed the situation.

Throughout this chapter, let the singular values of the matrix R be $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$, and let the singular values of the matrix D be $\delta_1 \geq \delta_2 \geq \cdots \geq \delta_n > 0$, such that $D^T D = R^T R - \mathbf{z}\mathbf{z}^T$. From the interlacing property of eigenvalues of Hermitian matrices, it is known that $\sigma_1 \geq \delta_1 \geq \sigma_2 \geq \cdots \geq \delta_n \geq \sigma_n$ [Horn and Johnson (1985)].

More quantitatively, we begin with a lemma which is a revised version of a similar result in Stewart (1979). Our proof is also different from his.

2.1-1 **Lemma.** With $||\mathbf{a}|| < 1$, where $\mathbf{a}$ is defined by $\mathbf{a}^T R = \mathbf{z}^T$,

$$\frac{\sigma_n}{\delta_n} \ , \ \frac{\sigma_1}{\delta_1} \leq \frac{1}{\sqrt{1-||\mathbf{a}||^2}} \leq \frac{\sigma_1}{\delta_n} \ . \tag{2.1-2}$$

Proof. The equation $AR = D$, where $A^T A = I - \mathbf{a}\mathbf{a}^T$ is the Cholesky factorization of $I - \mathbf{a}\mathbf{a}^T$, implies that $||A^{-1}||_2 \leq ||R||_2 \cdot ||D^{-1}||_2$, $||R||_2 \leq ||A^{-1}||_2 \cdot ||D||_2$, and $||D^{-1}||_2 \leq ||R^{-1}||_2 \cdot ||A^{-1}||_2$. From $||A^{-1}||_2 = \frac{1}{\sqrt{1-||\mathbf{a}||^2}}$, the three inequalities in this lemma immediately follow. □

Stewart (1979) did a perturbation analysis on the downdating problem and concluded that ill-conditioning is associated with small singular values in D. From Lemma (2.1-1) one has

$$\sigma_n \leq \frac{\delta_n}{\sqrt{1-||\mathbf{a}||^2}} \leq \sigma_1 \ , \tag{2.1-3}$$

which means that the smallest singular value of D is always of the same order of magnitude as the number $\sqrt{1-||\mathbf{a}||^2}$, provided that the original upper triangular matrix R does not have wide-spread singular values, i.e., the condition number of R, $\kappa_2(R)$, is not large. Therefore, the ill-conditioning in the downdating problem is associated with $||\mathbf{a}||$ being close to 1, provided that R is a well-conditioned matrix. This fact will be seen in the following forward error analysis on the hyperbolic rotation algorithm.

In the following we quote the main perturbation analysis results of Stewart (1979) for later use.

Let

$$\widetilde{D}^T\widetilde{D} = (R + E)^T(R + E) - \mathbf{z}\mathbf{z}^T \qquad (2.1\text{-}4)$$
$$= D^TD + R^TE + E^TR + E^TE \ ,$$

where E is a small perturbation of R. Suppose the singular values of $\widetilde{D}$ are $\widetilde{\delta}_1 \geq \cdots \geq \widetilde{\delta}_n$. From the classical perturbation theory for eigenvalues of symmetric matrices, Stewart (1979) showed that $||\widetilde{D} - D||_2$ can be *as large as*

$$\frac{2\sigma_1||E||_2 + ||E||_2^2}{\delta_n} \ . \qquad (2.1\text{-}5)$$

If one eliminates the term of $||E||_2^2$ and makes use of (2.1-2), then the conclusion above becomes:

$||\widetilde{D} - D||_2$ can be *as large as*

$$\frac{2\sigma_1||E||_2}{\delta_n} \leq \frac{2\sigma_1||E||_2}{\sigma_n}\ \frac{1}{\sqrt{1-||\mathbf{a}||^2}} = \frac{2\kappa_2(R)}{\sqrt{1-||\mathbf{a}||^2}}\ ||E||_2 \ . \qquad (2.1\text{-}6)$$

From (2.1-6) one can say that

$$\frac{\kappa_2(R)}{\sqrt{1-||\mathbf{a}||^2}} \qquad (2.1\text{-}7)$$

is an important factor of the condition number of downdating problem (0.1-12), when the perturbation from D to $\widetilde{D}$ results from a small perturbation of R only in (2.1-4). A similar result is obtained from the perturbation

$$\widetilde{D}^T\widetilde{D} = R^TR - (\mathbf{z} + \mathbf{f})(\mathbf{z} + \mathbf{f})^T \qquad (2.1\text{-}8)$$
$$= D^TD - \mathbf{z}\mathbf{f}^T - \mathbf{z}\mathbf{f}^T - \mathbf{f}\mathbf{f}^T \ ;$$

then similarly

$$\frac{2||\mathbf{z}||||\mathbf{f}||}{\delta_n} \leq \frac{2||R||_2 \cdot ||\mathbf{a}||||\mathbf{f}||}{\sigma_n} \cdot \frac{1}{\sqrt{1-||\mathbf{a}||^2}} \qquad (2.1\text{-}9)$$

$$= \frac{2\kappa_2(R)}{\sqrt{1-||\mathbf{a}||^2}} \cdot ||\mathbf{f}|| \ ,$$

if one uses $||\mathbf{z}|| \leq ||R||_2 \cdot ||\mathbf{a}|| \leq ||R||_2$.

2.1-10 **Remark**. Neither (2.1-6) nor (2.1-9) is a rigorous perturbation bound for $||\widetilde{D} - D||_2$ since in the original derivation of (4.1-5) by Stewart (1979), the directions of the inequalities used are not consistent (that is why the words " *as large as* " are used instead of an inequality). But from an intuitive viewpoint, such bounds are reasonable; from a practical viewpoint, they are good enough for us to estimate quantitatively the relation between accuracy and conditioning of downdating problem (0.1-12) in drawing some conclusions about the stability of the algorithm used.

Now we turn to the main topic of this chapter, the forward error analysis of the hyperbolic method.

2.2 Error analysis of the hyperbolic method

In this section a forward error bound for Algorithm *HY* is provided, and a comparison between Algorithm *HY* and Algorithm *OR* is also discussed. We claim Algorithm *HY* is forward stable [see, for example, Cybenko (1980)], after a comparison with the backward stable Algorithm *OR*. In order to make

the proof more readable, we proceed with the main results through several lemmas.

2.2-1 **Lemma**. For any pair of floating point numbers x and y, where $|x| > |y|$, let $H(1,2;\ x,y)$ be the exact hyperbolic rotation defined by (0.2-9) and $\bar{H}(1,2;\ x,y)$ be the computed $H(1,2;\ x,y)$. Then $||\bar{H} - H||_2 \leq 3.003\mu\sqrt{\frac{1+t}{1-t}}$, where $t = \left|\frac{y}{x}\right|$.

Proof. The proof closely follows the proof in Wilkinson (1965), where a similar result was proved for Givens rotations (p.131).

Let

$$c = \cosh\theta = \frac{x}{\sqrt{x^2 - y^2}} ,$$

$$s = \sinh\theta = \frac{y}{\sqrt{x^2 - y^2}} ,$$

$$\bar{c} = fl\left(\frac{x}{\sqrt{x^2 - y^2}}\right) ,$$

and

$$\bar{s} = fl\left(\frac{y}{\sqrt{x^2 - y^2}}\right) .$$

Then we have

$$\bar{c} = c(1+\epsilon) \ , \ \bar{s} = s(1+\epsilon)$$

and

$$|\epsilon| \leq 3.003\mu \ .$$

Here

$$||\bar{H} - H||_2 = |\epsilon| \cdot ||\Delta||_2 \tag{2.2-2}$$

where $||\Delta||_2$ is the spectral radius, $\rho(\Delta)$, of $\begin{bmatrix} c & -s \\ -s & c \end{bmatrix} = \Delta$.

Solving the characteristic equation

$$\det(\lambda I - \Delta) = 0 \ ,$$

which is

$$(\lambda - c + s)(\lambda - c - s) = 0 \ ,$$

one has

$$\lambda_1 = c - s$$
$$\lambda_2 = c + s \ .$$

But

$$\rho(\Delta) = \max\left\{|\lambda_1|, \ |\lambda_2|\right\} = \max\left\{|c - s|, \ |c + s|\right\} = |c| + |s| \ ;$$

Therefore

$$||\bar{H} - H||_2 \leq 3.003 \ \mu \frac{|x| + |y|}{\sqrt{x^2 - y^2}} \ .$$

Since $|x| > |y|$, the lemma is proved. □

Observe also that, the bound on $||\bar{H} - H||_2$ is sharp, since $||H||_2 = |c| + |s|$ is exactly true.

2.2-3 **Lemma**. Under the similar assumption in Lemma (2.2-1), for any vector $\mathbf{a} \in \mathbb{R}^2$ ($|a_1| > |a_2|$, $t = \left|\frac{a_2}{a_1}\right|$), the following inequality holds:

$$||fl(\bar{H}\mathbf{a}) - H\mathbf{a}|| \leq 6\mu \left(\frac{1+t}{1-t}\right)^{1/2} ||\mathbf{a}|| \ .$$

Proof. Let

$$\bar{\mathbf{b}} = fl(\bar{H}\mathbf{a}) \quad \text{and} \quad \mathbf{b} = H\mathbf{a} \ ;$$

then

$$\begin{pmatrix}\bar{b}_1\\ \bar{b}_2\end{pmatrix} = fl\left(\begin{bmatrix}\bar{c} & -\bar{s}\\ -\bar{s} & \bar{c}\end{bmatrix}\begin{pmatrix}a_1\\ a_2\end{pmatrix}\right) = \begin{bmatrix}\bar{c} & -\bar{s}\\ -\bar{s} & \bar{c}\end{bmatrix}\begin{pmatrix}a_1\\ a_2\end{pmatrix} + \begin{pmatrix}(\bar{c}a_1\delta_1 - \bar{s}a_2\delta_2)\\ (-\bar{s}a_1\delta_3 + \bar{c}a_2\delta_4)\end{pmatrix} \quad (2.2\text{-}4)$$

with

$$|\delta_i| \leq \mu \quad , \ i = 1,2,3,4 \ .$$

From (2.2-2),

$$\begin{pmatrix}\bar{b}_1\\ \bar{b}_2\end{pmatrix} - \begin{pmatrix}b_1\\ b_2\end{pmatrix} = (\bar{H} - H)\begin{pmatrix}a_1\\ a_2\end{pmatrix} + \begin{pmatrix}(\bar{c}a_1\delta_1 - \bar{s}a_2\delta_2)\\ (-\bar{s}a_1\delta_3 + \bar{c}a_2\delta_4)\end{pmatrix}$$

and $||fl(\bar{H}\mathbf{a}) - H\mathbf{a}|| \leq ||\bar{H} - H||_2 \cdot ||\mathbf{a}|| +$

$$\left((\bar{c}a_1\delta_1)^2 + (\bar{s}a_1\delta_3)^2\right)^{1/2} + \left((\bar{s}a_2\delta_2)^2 + (\bar{c}a_2\delta_4)^2\right)^{1/2} = ||\bar{H} - H||_2 \cdot ||\mathbf{a}|| + p \ .$$

Analogously to the similar result in Wilkinson (1965),

$$p \leq 2.0002\,\mu \left(\bar{c}^2 + \bar{s}^2\right)^{1/2} \left(|a_1| + |a_2|\right)$$

$$\leq 2.0002\,\mu \left(1 + |\epsilon|\right) \left(c^2 + s^2\right)^{1/2} \left(|a_1| + |a_2|\right)$$

$$\leq 2.0002\,\mu \left(1 + 3.003\mu\right) \sqrt{2} \left(c^2 + s^2\right)^{1/2} ||\mathbf{a}||$$

$$< 6\mu \left(c^2 + s^2\right)^{1/2} ||\mathbf{a}|| \ ;$$

since $\sqrt{\frac{1+t^2}{1-t^2}} < \sqrt{\frac{1+2t+t^2}{1-t^2}} = \sqrt{\frac{1+t}{1-t}}$,

the lemma follows. □

Observe also that all the inequalities used are attainable and that the constant 6 is generous. In order to obtain the constant 6 one needs only $\mu \leq 2^{-15} < 10^{-6}$. This is guaranteed by most modern computing machinery.

2.2-5 **Lemma.** Consider a vector $\mathbf{r}^{(0)} \in \mathbb{R}^{n+1}$ whose components are in floating point form, and a set of n pairs of floating point numbers x_p, y_p with $|x_p| > |y_p|$, $p = 1,\ldots,n$. Then for each pair there is an exact hyperbolic rotation matrix $H_p = H(p, n+1; x_p, y_p)$ such that by defining

$$\mathbf{r}^{(p)} = H_p \mathbf{r}^{(p-1)} \quad , \quad p = 1,\ldots,n \quad ,$$

the computed $\mathbf{r}^{(n)}$, $\bar{\mathbf{r}}^{(n)}$, satisfies

$$||\bar{\mathbf{r}}^{(n)} - H_n \cdots H_1 \mathbf{r}^{(0)}|| \leq 6n\mu(1+6\mu)^{n-1}||\mathbf{r}^{(0)}|| \prod_{p=1}^{n} \left(\frac{1+t_p}{1-t_p}\right)^{1/2} , \quad (2.2\text{-}6)$$

with $t_p = \left|\frac{x_p}{x_p}\right|$.

Proof. It follows from Lemma (2.2-3) that

$$fl(\bar{H}_1 \mathbf{r}^{(0)}) = H_1 \mathbf{r}^{(0)} + \mathbf{f}_1 \ ;$$

then

$$||\mathbf{f}_1|| \leq 6\mu \left(\frac{1+t_1}{1-t_1}\right)^{1/2} \left([\mathbf{r}_1^{(0)}]^2 + [\mathbf{r}_{n+1}^{(0)}]^2\right)^{1/2} \leq 6\mu \left(\frac{1+t_1}{1-t_1}\right)^{1/2} ||\mathbf{r}^{(0)}|| \ .$$

Letting

$$\bar{r}^{(1)} = fl(\bar{H}_1 r^{(0)}),$$

then

$$\bar{r}^{(2)} = fl(\bar{H}_2 \bar{r}^{(1)}) = H_2 \bar{r}^{(1)} + f_2 ,$$

and it follows again from Lemma (2.2-3) that

$$||f_2|| \le 6\mu \left(\frac{1 + t_2}{1 - t_2}\right)^{1/2} ||\bar{r}^{(1)}|| \le 6\mu \left(\frac{1 + t_2}{1 - t_2}\right)^{1/2} (1 + 6\mu) \cdot$$

$$\cdot \left(\frac{1 + t_1}{1 - t_1}\right)^{1/2} ||r^{(0)}|| = 6\mu(1 + 6\mu) \left(\frac{(1 + t_1)(1 + t_2)}{(1 - t_1)(1 - t_2)}\right)^{1/2} ||r^{(0)}|| .$$

Continuing in the same manner, it follows that

$$||\bar{r}^{(0)} - H_n \cdots H r^{(0)}||$$

$$\le ||H_p||_2 \cdots ||H_2||_2 \ ||f_1|| + ||H_p||_2 \cdots ||H_3||_2 ||f_2|| + \cdots + ||f_n||$$

$$\le \prod_{p=1}^{n} \left(\frac{1 + t_p}{1 - t_p}\right)^{1/2} \cdot \left(6\mu \sum_{i=1}^{n} (1 + 6\mu)^{i-1} ||r^{(0)}|| \right)$$

$$\le \prod_{p=1}^{n} \left(\frac{1 + t_p}{1 - t_p}\right)^{1/2} \cdot n \cdot 6\mu(1 + 6\mu)^{n-1} ||r^{(0)}|| . \quad \square$$

Now it is easy to show the following main result.

2.2-7 **Theorem.** Using Algorithm *HY* (0.2-13) to solve downdating problem (0.1-12), one reduces

$$A_0 = \begin{bmatrix} R \\ z^T \end{bmatrix}$$

to

$$A_n = \begin{bmatrix} D \\ \mathbf{0}^T \end{bmatrix} ,$$

where $\bar{A}_n = \begin{bmatrix} \bar{D} \\ \mathbf{0}^T \end{bmatrix}$, the computed value of A_n, satisfies

$$||\bar{A}_n - \hat{H}_n \cdots \hat{H}_1 A_0||_F \leq 6n\mu(1 + 6\mu)^{n-1}||A_0||_F \prod_{p=1}^{n} \left(\frac{1 + \bar{t}_p}{1 - \bar{t}_p} \right)^{1/2} . \qquad (2.2\text{-}8)$$

Here each $\hat{H}_p = H(p, n+1; r_{pp}, \bar{z}_p^{(p-1)})$ is an exact hyperbolic rotation and $\bar{t}_p = \left| \frac{\bar{z}_p^{(p-1)}}{r_{pp}} \right|$, where $\bar{z}_p^{(p-1)}$ denotes the $(n+1,p)$ element of the computed A_p, $\bar{A}_p$, defined by

$$\bar{A}_p = fl(\hat{H}_p \bar{A}_{p-1}), \quad \bar{A}_0 = A_0 , \quad p = 1, \ldots, n.$$

Proof. If we rewrite

$$\bar{A}_p = fl(\hat{H}_p \bar{A}_{p-1}) = \hat{H}_p \bar{A}_{p-1} + F_p , \quad p = 2, \ldots, n$$

and

$$\bar{A}_1 = fl(\hat{H}_1 A_0) = \hat{H}_1 A_0 + F_1 ,$$

the proof follows by applying Lemma 2 to the columns of $A_0 = \begin{bmatrix} R \\ \mathbf{z}^T \end{bmatrix}$ and by using the relation between the Frobenius norm of a matrix and the 2-norm of its column vectors. □

This completes the forward error analysis of Algorithm *HY*. Since the computed $\bar{t}_p$'s appear in the bound (2.2-8), it may be referred to as an *a posteriori* bound, following Wilkinson (1965).

For the purpose of comparison with Algorithm *OR*, we simplify the bound (2.2-8) by

(1) eliminating the terms of $0(\mu^2)$,

(2) using the theoretical $z_p^{(p-1)}$, hence the t_p, in place of the computed correspondents.

After the above simplifications, we can make use of the important sharp bound (1.4-2) for the otherwise messy product in (2.2-8); therefore,

$$||\bar{A}_n - \hat{H}A_0||_F \lesssim 6n\mu||A_0||_F \frac{M_n}{\sqrt{1 - ||\mathbf{a}||^2}} , \qquad (2.2\text{-}9)$$

where

$$M_n = \left(1 + \sqrt{1 - (1 - ||\mathbf{a}||^2)^{1/n}}\right)^n .$$

From Section 1.4 we know M_n is of reasonable size when $||\mathbf{a}||$ is not very close to 1 and n is small. But even if $||\mathbf{a}||$ is very close to 1, M_n is still far less than 2^n [see Table (1.4-26)]. Using this observation, in the next section, we compare Algorithms *HY* and *OR* and conclude that Algorithm *HY* is forward stable.

2.3 Stability of The Hyperbolic Method

In order to compare Algorithm *HY* with Algorithm *OR*, the main results of Stewart (1979) are reviewed below with minor changes in some constants.

Stewart (1979) shows that for the LINPACK method (Algorithm *OR*) there exists an exact orthogonal matrix $\hat{Q}$ such that

$$\hat{Q}\begin{bmatrix} R \\ \mathbf{0}^T \end{bmatrix} = \begin{bmatrix} \bar{D} & + G \\ \mathbf{z}^T & + \mathbf{s}^T \end{bmatrix} ,$$

where

$$||G||_F \underset{\sim}{\leq} 6n\mu||R||_F , \tag{2.3-1}$$

$$||\mathbf{s}|| \underset{\sim}{\leq} \left(\frac{15n + 7}{2}\right) \mu||R||_F , \tag{2.3-2}$$

and $\bar{D}$ is the computed result of downdating problem (0.1-12). This is essentially a backward error analysis of Algorithm *OR*, since the error represented in G is much less important than the error represented in $\mathbf{s}$. This backward error analysis actually shows that for given R and $\mathbf{z}$, Algorithm *OR* yields a computed downdated Cholesky factor $\bar{D}$, which is very near to the exact Cholesky factor of a slightly perturbed downdating problem $R^TR - (\mathbf{z}+\mathbf{s})(\mathbf{z}+\mathbf{s})^T$; i.e.,

$$\tilde{D}^T\tilde{D} = (\bar{D} + G)^T(\bar{D} + G) = R^TR - (\mathbf{z} + \mathbf{s})(\mathbf{z} + \mathbf{s})^T . \tag{2.3-3}$$

As Stewart (1979) indicates, if the problem is ill-conditioned, even a small perturbation in $\mathbf{s}$ can cause severe inaccuracy in $\bar{D}$.

In order to see how ill-conditioning affects the accuracy, we use the results (2.1-7) and (2.1-8) of the perturbation analysis; then, (2.3-3) and (2.3-2) imply that $||\tilde{D} - D||_2$ can be *as large as*

$$\frac{2\kappa_2(R)}{\sqrt{1 - ||a||^2}}\ ||s|| \leq (15n + 7)\mu||R||_F\ \frac{\kappa_2(R)}{\sqrt{1 - ||a||^2}}\ . \qquad (2.3\text{-}4)$$

If we consider the G term, then $||\bar{D} - D||_F$ can be *as large as*

$$\left\{\sqrt{n}\,(15n + 7)\left(\frac{\kappa_2(R)}{\sqrt{1 - ||a||^2}}\right) + 6n\right\}\mu||R||_F\ . \qquad (2.3\text{-}5)$$

On the other hand, since the pseudo-orthogonal matrix $\hat{H}$ in the left-hand side of (2.2-9) is exact, letting $\begin{pmatrix}\bar{D}\\ 0^T\end{pmatrix} - \hat{H}\begin{pmatrix}R\\ z^T\end{pmatrix} = \begin{pmatrix}-F\\ -g^T\end{pmatrix}$, we have

$$(\bar{D} + F)^T(\bar{D} + F) + gg^T = R^TR - zz^T\ ,$$

and omitting gg^T, we have

$$||\bar{D} - D||_F \leq 6n\mu\left(||R||_F + ||z||\right)\frac{M_n}{\sqrt{1 - ||a||^2}} \qquad (2.3\text{-}6)$$

$$\leq (6n + \sqrt{n})\,\mu||R||_F\frac{M_n}{\sqrt{1 - ||a||^2}}\ .$$

Comparing bound (2.3-5) with bound (2.3-6), where M_n is given by (1.4-2), since they are comparable in magnitude and since algorithm *OR* is backward stable, it follows that Algorithm *HY* is forward (or weakly) stable, in the sense of Cybenko (1980) and Bunch (1987). The approach taken here is consistent with that of Cybenko (1980), in which he found that the forward error bound for the Levinson-Durbin algorithm for positive definite Toeplitz systems is comparable in size with the error for the backward stable Cholesky Algorithm for the same problems.

In the last chapter our data testing also shows that the LINPACK method and the hyperbolic method share comparable accuracy with little favor to the LINPACK method, as we predicted.

CHAPTER 3: A NEW AND FAST DOWNDATING ALGORITHM

In this chapter a new fast downdating algorithm is introduced, which requires only $\frac{3}{2}n^2$ multiplications and is expected to be just as stable as the LINPACK downdating algorithm. Actually there are two new algorithms requiring the same numbers of operations; we call them Algorithm A and Algorithm B, respectively. We derive Algorithm B in two different ways in order to see its relation to previous algorithms. In section 3.2 a complete list of all possible downdating algorithms of (1.1-26) type is given [recall from remark (0.1-14) that we are not concerned with the downdating algorithms related to LDL^T decompositions]. This list gives us an overview of all previous algorithms and exposes an as yet empty position which should be occupied by Algorithm A.

3.1 Reorganization of Algorithm *AR*

By exploiting the special structure of the matrix A given in (1.2-2), we can reduce the computations involved in Algorithm *AR* (1.2-5) significantly. The following decomposition of matrix A is the key step in understanding the structure of A.

3.1-1 **Lemma.** Matrix A (1.1-2) can be decomposed as

$$A = D_1(D_2\bar{A} + D_3) \ , \tag{3.1-2}$$

where

$$D_1 = diag\left(\frac{1}{\beta_0\beta_1},\ \frac{1}{\beta_1\beta_2},\ldots,\frac{1}{\beta_{n-1}\beta_n}\right)\ , \tag{3.1-3}$$

$$D_2 = diag(-a_1,\ -a_2,\ldots,-a_n)\ , \tag{3.1-4}$$

$$D_3 = diag(\beta_0^2,\ \beta_1^2,\ldots,\beta_{n-1}^2)\ , \tag{3.1-5}$$

and

$$\bar{A} = \begin{bmatrix} a_1 & a_2 & a_3 & \cdots & a_n \\ & a_2 & a_3 & \cdots & a_n \\ & & a_3 & \cdots & a_n \\ & & & \ddots & \vdots \\ & & & & a_n \end{bmatrix} \tag{3.1-6}$$

Proof. The proof proceeds by direct verification. □

Now the number of multiplications involved in computing the matrix product AR will depend mainly on the computation of $\bar{A}R$, since all the D_i's are diagonal and matrix multiplication with one diagonal matrix requires only $O\left(n^2\right)$ scalar multiplications.

Observe that now the matrix $\bar{A}$ has a very special structure. If we write $\bar{A}$ as

$$\bar{A} = \begin{bmatrix} \bar{A}_1 \\ \bar{A}_2 \\ \vdots \\ \bar{A}_n \end{bmatrix}\ .$$

then $\bar{A}_1 = \mathbf{a}^T$ and $\mathbf{a}^T R = \mathbf{z}^T$, i.e. the first row of $\bar{A}R$ is $\mathbf{z}^T$. The second row of $\bar{A}R$ will be $\mathbf{z}^T - a_1 R_1$, where R_1 is the first row of R, and so on. Therefore, we have

$$\bar{A}R = \begin{bmatrix} \mathbf{z}^T \\ \mathbf{z}^T - a_1 R_1 \\ \mathbf{z}^T - a_1 R_1 - a_2 R_2 \\ \vdots \\ \mathbf{z}^T - a_1 R_1 - \cdots - a_{n-1} R_{n-1} \end{bmatrix} . \qquad (3.1\text{-}7)$$

Clearly, $\bar{A}R$ can be computed recursively and the computations can be vectorized on a vector machine like the Cray X-MP. Even with serial machines only $\frac{n^2}{2}$ multiplications are needed in computing $\bar{A}R$.

The total number of multiplications involved in computing

$$AR = (D_1 D_2)\bar{A}R + (D_1 D_3)R$$

is $\frac{3}{2}n^2 + 0(n)$. But we need vector $\mathbf{a}$ as well as all the β_i's: that costs $\frac{n^2}{2}$ additional multiplications and n square roots. Therefore, the computational complexity is now comparable to that of the hyperbolic algorithm (0.2-24). Actually this algorithm is very close to Method C5 in Gill, Golub, Murray and Saunders (1974).

But if one recognizes that (3.1-7) is essentially the process of solving the system $\mathbf{a}^T R = \mathbf{z}^T$, then computing $\bar{A}R$ and solving $\mathbf{a}^T R = \mathbf{z}^T$ can be merged nicely to reduce that total number of multiplications further to $\frac{3}{2}n^2$, which is $\frac{n^2}{2}$ fewer than the hyperbolic algorithm.

Thus, we finally reformulate Algorithm AR as the following.

3.1-8 **Algorithm** B.

Given an upper triangular matrix, $R \equiv (r_{ij}) \in \mathbb{R}^{n \times n}$ and a vector $\mathbf{z} \equiv \bar{\mathbf{z}}^{(0)} \in \mathbb{R}^n$ being removed, the following algorithm computes the downdated Cholesky factor $D \equiv (d_{ij}) \in \mathbb{R}^{n \times n}$ such that $D^T D = R^T R - \mathbf{z}\mathbf{z}^T$.

$$\alpha_0 = 1$$

For $i = 1, n$

$$a_i = \frac{\bar{z}_i^{(i-1)}}{r_{ii}}$$

$$\alpha_i = \alpha_{i-1} - a_i^2$$

$$\beta_i = \sqrt{\alpha_i}$$

$$d_{ii} = \frac{\beta_i}{\beta_{i-1}} r_{ii}$$

For $j = i+1, n$

$$\bar{z}_j^{(i)} = \bar{z}_j^{(i-1)} - a_i r_{ij}$$

$$d_{ij} = \frac{\beta_{i-1}}{\beta_i} r_{ij} - \frac{a_i}{\beta_{i-1}\beta_i} \bar{z}_j^{(i-1)}$$

This algorithm requires $\frac{3}{2}n^2$ multiplications and n square roots.

3.2 The new downdating algorithm

First we give another derivation of Algorithm B which reveals its intimate relationship with the hyperbolic algorithm. Recall that at the end of

Chapter 1 we summarized the hyperbolic method in vector form (1.1-26):

$$D_k = \frac{\beta_{k-1}}{\beta_k} R_k - \frac{a_k}{\beta_k} (\mathbf{z}^{(k-1)})^T$$

$$(\mathbf{z}^{(k)})^T = -\frac{a_k}{\beta_k} R_k + \frac{\beta_{k-1}}{\beta_k} (\mathbf{z}^{(k-1)})^T$$

and notice that we proved in lemma (1.1-10) that

$$\beta_k (\mathbf{z}^{(k)})^T = (\bar{\mathbf{z}}^{(k)})^T \quad .$$

Then we immediately know that (1.1-26) can be changed to

$$D_k = \frac{\beta_{k-1}}{\beta_k} R_k - \frac{a_k}{\beta_k \beta_{k-1}} (\bar{\mathbf{z}}^{(k-1)})^T \qquad (3.2\text{-}1)$$

$$(\bar{\mathbf{z}}^{(k)})^T = -a_k R_k + (\bar{\mathbf{z}}^{(k-1)})^T \quad .$$

But this is exactly Algorithm B. What really happens is that the transition of $\mathbf{z}^{(k-1)}$ to $\mathbf{z}^{(k)}$ in downdating is replaced by the transition of $\bar{\mathbf{z}}^{(k-1)}$ to $\bar{\mathbf{z}}^{(k)}$, which actually is the process of solving the system $\mathbf{a}^T R = \mathbf{z}^T$.

Now we are curious about how many different versions of the downdating algorithms of (1.1-26) type may exist [for solving problem (0.1-12)]. The following theorem tends to answer this question, assuming that vector $\mathbf{a}$ and therefore the β_i's are given.

3.2-2 **Theorem.** Considering downdating problem (0.1-12), if one counts Algorithm *HY* (0.2-14) and Algorithm *HC* (0.2-20) differently, then there are only six different algorithms of (1.1-26) type. That means there are three downdating algorithms other than the LINPACK, hyperbolic, and Chambers'

version of hyperbolic algorithms. But only two of the six can be incorporated with the process of solving $\mathbf{a}^T R = \mathbf{z}^T$. The number of multiplications involved in these two algorithms can be further reduced to $\frac{3}{2}n^2$.

Proof. Since we assume vector **a** is known, by Lemmas (1.1-10) and (1.1-19), one can write out all the possible transition formulas of R_i's, D_i's, and $\mathbf{z}^{(i)}$'s, by using a_i's and β_i's as coefficients as follows:

(1) $D_i = T(R_i, \mathbf{z}^{(i)})$

i.e. $D_i = \frac{\beta_i}{\beta_{i-1}} R_i - \frac{a_i}{\beta_{i-1}} (\mathbf{z}^{(i)})^T$

(2) $D_i = H(R_i, \mathbf{z}^{(i-1)})$

i.e. $D_i = \frac{\beta_{i-1}}{\beta_i} R_i - \frac{a_i}{\beta_i} (\mathbf{z}^{(i-1)})^T$

(3) $R_i = H(D_i, \mathbf{z}^{(i)})$

i.e. $R_i = \frac{\beta_{i-1}}{\beta_i} D_i + \frac{a_i}{\beta_i} (\mathbf{z}^{(i)})^T$

(4) $R_i = T(D_i, \mathbf{z}^{(i-1)})$

i.e. $R_i = \frac{\beta_i}{\beta_{i-1}} D_i + \frac{a_i}{\beta_{i-1}} (\mathbf{z}^{(i-1)})^T$

(5) $\mathbf{z}^{(i-1)} = T(R_i, \mathbf{z}^{(i)})$

i.e. $(\mathbf{z}^{(i-1)})^T = \frac{a_i}{\beta_{i-1}} R_i + \frac{\beta_i}{\beta_{i-1}} (\mathbf{z}^{(i)})^T$

(6) $\mathbf{z}^{(i)} = H(R_i, \mathbf{z}^{(i-1)})$

i.e. $(\mathbf{z}^{(i)})^T = -\frac{a_i}{\beta_i} R_i + \frac{\beta_{i-1}}{\beta_i} (\mathbf{z}^{(i-1)})^T$

(7) $\mathbf{z}^{(i-1)} = H(D_i, \mathbf{z}^{(i)})$

i.e. $(\mathbf{z}^{(i-1)})^T = \frac{a_i}{\beta_i} D_i + \frac{\beta_{i-1}}{\beta_i} (\mathbf{z}^{(i)})^T$

(8) $\mathbf{z}^{(i)} = T(D_i, \mathbf{z}^{(i-1)})$

i.e. $(\mathbf{z}^{(i)})^T = -\frac{a_i}{\beta_{i-1}} D_i + \frac{\beta_i}{\beta_{i-1}} (\mathbf{z}^{(i-1)})^T$

(3.2-3)

In (3.2-3) the function T (not the transpose T) means trigonometric function coefficients are used in the expression while the function H means hyperbolic function coefficients are used. The formulas from (1) to (4) are all the possible transitions between R_i and D_i, and the formulas from (5) to (8) are all the

possible transitions between $\mathbf{z}^{(i-1)}$ and $\mathbf{z}^{(i)}$. The derivation of these 8 formulas is trivial: (1) and (5) together form the LINPACK algorithm, while (2) and (6) together form the hyperbolic algorithm; then the remaining formulas can be derived from these four. Thus, the downdating algorithm should be formed by choosing (1) or (2) combined with (5), (6), (7) or (8). But (1) cannot be combined with (8), (2) cannot be combined with (7); therefore there are only 6 downdating algorithms. They are shown in Table 3.2-4.

Table 3.2-4 The classification of downdating algorithms

$R_i \to D_i$	$\mathbf{z}^i \to \mathbf{z}^{(i-1)}$ $\mathbf{z}^{(i-1)} - \mathbf{z}^{(i)}$	Algorithms	Lowest possible # of mult.	Test results
(1)	(5)	Algorithm *OR* (LINPACK type)	$5/2\ n^2$	stable
	(6)	Algorithm *A* (hyperbolic type)	$3/2\ n^2$	stable
	(7)	Algorithm (1,7) (LINPACK type)	x	
(2)	(5)	Algorithm (2,5) (LINPACK type)	x	
	(6)	Algorithm *HY* (hyperbolic type), B	$3/2\ n^2$	less stable
	(8)	Algorithm *HC* (hyperbolic type, Chambers)	$2n^2$	less stable

If one notices that of formulas (5) to (8), only formula (6) can be incorporated with solving the system $\mathbf{a}^T R = \mathbf{z}^T$; then the theorem is proved. □

From Table (3.2-4) the transition of $\mathbf{z}^{(k)}$ has two different types, one from $\mathbf{z} \to \mathbf{0}$, one from $\mathbf{0} \to \mathbf{z}$. We call the $(\mathbf{z} \to \mathbf{0})$-type the hyperbolic type and the

$(\mathbf{0} \rightarrow \mathbf{z})$-type the LINPACK type; therefore (5) and (7) are LINPACK types while (6) and (8) are hyperbolic types. For the LINPACK types there is no way of knowing the β_i's in the order from β_n to β_1 without knowing all components of vector $\mathbf{a}$. Therefore, the $\frac{5}{2}n^2$ multiplications required by Algorithm *OR* cannot be reduced. On the other hand, the "*H*" expressions, using hyperbolic rotations, are less stable than the "*T*" expressions, using orthogonal rotations. So there is no reason to use algorithm (1,7) since it requires the same number of multiplications as the LINPACK does [indicated by lower case "x" in table (3.2-4)]. For the same reason, algorithm (2,5) is also dropped. There are three versions of algorithm (2,6), requiring $\frac{5}{2}n^2$, $2n^2$, and $\frac{3}{2}n^2$ multiplications, respectively. The $\frac{3}{2}n^2$ multiplications version is exactly Algorithm *B* (3.1-8):

$$D_k = \frac{\beta_{k-1}}{\beta_k} R_k - \frac{a_k}{\beta_k \beta_{k-1}} (\bar{\mathbf{z}}^{(k-1)})^T$$

$$(\bar{\mathbf{z}})^T = (\bar{\mathbf{z}}^{(k-1)})^T - a_k R_k \quad .$$

There is no reason to use an algorithm which requires more operations while the stability is expected to be the same, so Algorithm *B* should be substituted for Algorithm *HY* anyway. The same situation exists for algorithm (1,6). The $\frac{3}{2}n^2$ multiplications version of algorithm (1.6) is

$$(\bar{\mathbf{z}}^{(i)})^T = (\bar{\mathbf{z}}^{(i-1)})^T - a_i R_i \quad , \tag{3.2-5}$$

$$D_i = \frac{\beta_i}{\beta_{i-1}} R_i - \frac{a_i}{\beta_{i-1}\beta_i} (\bar{z}^{(i)})^T \quad . \tag{3.2-6}$$

This is our second new downdating algorithm. In fact this is the new downdating algorithm we recommend.

3.2-7 **Algorithm** *A*. Given an upper triangular matrix, $R \equiv (r_{ij}) \in \mathbb{R}^{n \times n}$ and a vector $z \equiv \bar{z}^{(0)} \in \mathbb{R}^n$ being removed, the following algorithm computes the downdated Cholesky factor $D \equiv (d_{ij}) \in \mathbb{R}^{n \times n}$ such that $D^T D = R^T R - z z^T$.

$\alpha_0 = 1$

For $i = 1, n$

$$a_i = \frac{\bar{z}_i^{(i-1)}}{r_{ii}}$$

$$\alpha_i = \alpha_{i-1} - a_i^2$$

$$\beta_i = \sqrt{\alpha_i}$$

$$d_{ii} = \frac{\beta_i}{\beta_{i-1}} r_{ii}$$

For $j = i+1, n$

$$\bar{z}_j^{(i)} = \bar{z}_j^{(i-1)} - a_i r_{ij}$$

$$d_{ij} = \frac{\beta_i}{\beta_{i-1}} r_{ij} - \frac{a_i}{\beta_{i-1}\beta_i} \bar{z}_j^{(i)}.$$

This algorithm requires $\frac{3}{2} n^2$ multiplications and n square roots.

Comparing Algorithm *A* with Algorithm *B*, since equation (1) is a part of the LINPACK method while (2) is a part of the hyperbolic rotation method,

we can expect that Algorithm A will perform slightly better than Algorithm B when the conditioning gets ill. In fact, Algorithm A is a rearrangement of the LINPACK method. First of all, equation (3.2-5) is exactly the process of solving the system $\mathbf{a}^T R = \mathbf{z}^T$ which is the first step in the LINPACK method. Second, the computation of β_is is included in steps (2) and (3) in the LINPACK method (0.3-9). Finally, LINPACK step (4) in (0.3-9) is equivalent to equation (3.2-6) of Algorithm A. Observe that in the LINPACK method at first $\mathbf{z}$ is transformed to $\mathbf{0}$ in the forward substitution process, then $\mathbf{0}$ is transformed back to $\mathbf{z}$ by the orthogonal transformation process. But in Algorithm A we have only one process. The computation of the transformation from $\mathbf{0}$ back to $\mathbf{z}$ will contribute extra error to the computed D_is in the LINPACK method. Therefore we predict that Algorithm A will perform slightly better than the LINPACK method. This observation is also supported by our data testing so far (see Chapter 4).

CHAPTER 4: STABILITY TESTING RESULTS

Most of the downdating algorithms in this thesis were tested on the IBM 4381 at NCSU. The results reflected the stability analysis developed so far. It is interesting to note that the new downdating algorithm (Algorithm A) appears to be just as stable as the LINPACK method.

We had to resolve two problems before testing the algorithms. The first was how to generate the original data R and $\mathbf{z}^T$. The second was how to find the "theoretical" downdated Cholesky factor, $\hat{D}$, such that the equation

$$\hat{D}^T \hat{D} = R^T R - \mathbf{z}\mathbf{z}^T \tag{4.0-1}$$

holds almost exactly. Sections 4.1 and 4.2 address these two concerns. Finally in Section 4.3 we give the test results.

4.1 Generating the original data

The most important thing which must be controlled in testing the stability of an algorithm is the condition number of the underlying problem. For the downdating problem, it is natural to choose as the critical number the 2-norm of the vector a (see section 2.1). When $||\mathbf{a}||$ is very close to 1, the downdating problem generated will be ill-conditioned. In the test programs we used a non-negative integer h to control $||\mathbf{a}||$ in the following equation:

$$||\mathbf{a}|| = \frac{10^h - 1}{10^h} \quad . \tag{4.1-1}$$

Secondarily, the weight carried by each component of **a** should also be controlled. This was done by a predefined vector **b** [see procedure (4.1-2) below]. Finally, vector **a** should also be generated by some random procedure. The following procedure was used to generate vector **a** in the test programs.

4.1-2 **Procedure**.

(1) Generate a random matrix $T \equiv (t_{ij})$, $t_{ij} \in (0, 1)$.

(2) Define a vector **b**,

for example, $\mathbf{b} = (1,1,\ldots,1)$ or $\mathbf{b} = (n,1,\ldots,1)^T$.

(3) Create a vector **q** such that

$$\mathbf{q} = T\mathbf{b}.$$

(4) Normalize **q** to generate vector **a**,

$$a_i = q_i \frac{||\mathbf{a}||}{||\mathbf{q}||},$$

where $||\mathbf{a}||$ is defined by (4.1-1).

Notice that from (4.1-1) the vector **a** generated here has norm

$$||\mathbf{a}|| = 1 - 10^{-h}.$$

When h gets large we cannot expect the computed results to be accurate.

The upper triangular matrix R can be generated by any random number generator, and vector **z** is defined by

$$\mathbf{z}^T = R\mathbf{a}^T.$$

Since the conditioning of R is not our major concern, it is reasonable [(see (2.1-3)] to assume that $r_{ij} \in (0, 1)$.

4.2 The "theoretical" answer

A more subtle problem in testing the numerical stability of an algorithm is to find the "theoretical" answer. The true theoretical answer to a numerical problem is the answer obtained from original data without any rounding error, that is, when every step of the computation is done in exact arithmetic. But this is impossible for us if we choose to create a *random* data set. The best thing we can hope for is to get a downdated Cholesky factor $\hat{D}$ which satisfies (4.0-1) as rigorously as possible.

In the test programs we used a what we call the "half LINPACK" procedure to produce the $\hat{D}$. This means that right after Procedure (4.1-2) was performed, the generated vector $\mathbf{a}$ and the predetermined norm $||\mathbf{a}||$ were sent directly to the LINPACK algorithm (0.3-11) step 3 and step 4. Therefore, the rounding error produced by solving the triangular system and recomputing $||\mathbf{a}||$ in the LINPACK algorithm was avoided. To verify that $\hat{D}$ was near to the exact solution of the matrix equation (4.0-1), the relative error

$$\frac{||R^T R - \mathbf{z}\mathbf{z}^T - \hat{D}^T \hat{D}||_F}{||R^T R - \mathbf{z}\mathbf{z}^T||_F}$$

was checked. Our results [see the first column of Table (4.3-2)] showed that this relative error was almost always independent of h, i.e., independent of

the condition number of the underlying downdating problem. Since the accuracy of any answer obtained from an *algorithm* is dependent on the conditioning , we now have the right to believe that the $\hat{D}$ is an "algorithm free" answer, i.e., it is "theoretical", up to single precision.

4.3 The test results

In Table (4.3-2) we use the Frobenius norm to compute the relative error

$$\frac{||\hat{D} - \bar{D}||_F}{||\hat{D}||_F} \quad , \tag{4.3-1}$$

where $\hat{D}$ is computed by the "half LINPACK" method in double precision and, $\bar{D}$ is the computed result from the tested algorithm in single precision. Single precision for IBM 4381 is $\mu < 10^{-6}$ and double precision is $\mu < 10^{-16}$.

The data listed in Table (4.3-2) were obtained using the vector $\mathbf{b} = (1,\ldots,1)^T$ [see Procedure (4.1-2)]. In instances where "__" appears, the corresponding computation broke down due to attempting to take a square root from a negative number.

Table 4.3-2 Test results

$n=10$	$\frac{\|\|R^TR-\mathbf{z}\mathbf{z}^T-\hat{D}^T\hat{D}\|\|_F}{\|\|R^TR-\mathbf{z}\mathbf{z}^T\|\|_F}$	$\frac{\|\|\hat{D}-\bar{D}\|\|_F}{\|\|\hat{D}\|\|_F}$			
$\|\|a\|\|$	Half-LINPACK	Alg. *HY*	Alg.*HC*	Alg.*OR*	Alg.*A*
0.2	$0.24_{10^{-15}}$	$.49_{10^{-6}}$	$.11_{10^{-6}}$	$.11_{10^{-6}}$	$.11_{10^{-6}}$
0.5	$0.25_{10^{-15}}$	$.38_{10^{-6}}$	$.20_{10^{-6}}$	$.29_{10^{-6}}$	$.31_{10^{-6}}$
0.8	$0.28_{10^{-15}}$	$.15_{10^{-3}}$	$.13_{10^{-5}}$	$.60_{10^{-6}}$	$.58_{10^{-6}}$
$1-10^{-1}$	$0.54_{10^{-15}}$	$.53_{10^{-5}}$	$.36_{10^{-5}}$	$.17_{10^{-5}}$	$.16_{10^{-5}}$
$1-10^{-2}$	$0.64_{10^{-15}}$	$.22_{10^{-4}}$	$.15_{10^{-4}}$	$.46_{10^{-5}}$	$.45_{10^{-5}}$
$1-10^{-4}$	$0.67_{10^{-15}}$	$.21_{10^{-3}}$	$.18_{10^{-3}}$	$.54_{10^{-4}}$	$.50_{10^{-4}}$
$1-10^{-6}$	$0.44_{10^{-15}}$	$.15_{10^{-2}}$	$.77_{10^{-3}}$	$.60_{10^{-3}}$	$.57_{10^{-3}}$
$1-10^{-8}$	$0.60_{10^{-15}}$	$.12_{10^{-2}}$	$.12_{10^{-2}}$	$.82_{10^{-3}}$	$.79_{10^{-3}}$

$n=20$	$\frac{\|\|R^TR-\mathbf{z}\mathbf{z}^T-\hat{D}^T\hat{D}\|\|_F}{\|\|R^TR-\mathbf{z}\mathbf{z}^T\|\|_F}$	$\frac{\|\|\hat{D}-\bar{D}\|\|_F}{\|\|\hat{D}\|\|_F}$			
$\|\|a\|\|$	Half-LINPACK	Alg. *HY*	Alg.*HC*	Alg.*OR*	Alg.*A*
0.2	$0.32_{10^{-15}}$	$.31_{10^{-5}}$	$.47_{10^{-5}}$	$.51_{10^{-5}}$	$.51_{10^{-5}}$
0.5	$0.44_{10^{-15}}$	$.35_{10^{-4}}$	$.71_{10^{-5}}$	$.30_{10^{-4}}$	$.30_{10^{-4}}$
0.8	$0.10_{10^{-14}}$	$.29_{10^{-3}}$	$.19_{10^{-4}}$	$.25_{10^{-4}}$	$.25_{10^{-4}}$
$1-10^{-1}$	$0.14_{10^{-14}}$	$0.46_{10^{-3}}$	$.15_{10^{-3}}$	$.13_{10^{-3}}$	$.13_{10^{-3}}$
$1-10^{-2}$	$0.17_{10^{-14}}$	$0.15_{10^{-2}}$	$.48_{10^{-3}}$	$.15_{10^{-3}}$	$.15_{10^{-3}}$
$1-10^{-4}$	$0.17_{10^{-14}}$	$.63_{10^{-3}}$	—	$.26_{10^{-2}}$	$.26_{10^{-2}}$
$1-10^{-6}$	$0.21_{10^{-14}}$	$.63_{10^{-3}}$	—	—	—
$1-10^{-8}$	$0.18_{10^{-14}}$	—	—	—	—

CHAPTER 5: CONCLUDING REMARKS

In this study we have focused our attention on the downdating problem [see (0.1-12)] and the two existing algorithms: the hyperbolic method (section 0.2) and the LINPACK method (section 0.3).

In light of the pioneering work of Stewart (1979), this study shows that the hyperbolic method is weakly stable, a kind of stability which "does not guarantee that the answer comes from a slightly perturbed problem, but at least it is no more inaccurate than one that does." [Stewart (1986)].

This study also shows that two different processes, the solution of the triangular system $\mathbf{a}^T R = \mathbf{z}^T$ and the downdating of R to D such that $R^T R - \mathbf{z}\mathbf{z}^T = D^T D$, can be combined in a fashion that reduces the number of required multiplications to $\frac{3}{2}n^2$. In comparison, the LINPACK method requires $\frac{5}{2}n^2$ multiplications and the hyperbolic method requires $2n^2$ multiplications. This new algorithm (Algorithm A) accomplishes a forty-percent reduction of the number of multiplications required by the LINPACK method without sacrificing the stability of that method (see the analysis and Table (4.3-2) at the end of section 4.3). A further contrast is with the modifications of the LDL^T method, which achieve savings of computation by rescaling the diagonal elements of the Cholesky factor R to 1: some of its n^2 or $\frac{3}{2}n^2$ multiplication techniques are reported not to be stable [see Fletcher and Powell

(1974) and Miller and Wrathall (1980)]. In fact, according to the analysis and our data tests, it is reasonable to conjecture that Algorithm A is the best downdating algorithm so far for problem (0.1-12).

All the work above was done by exploiting the inherent mathematical properties of vector $\mathbf{a}$, the solution of the system $\mathbf{a}^T R = \mathbf{z}^T$. In this sense this thesis is an expansion of the last section, 'The meaning of $||\mathbf{a}||$', in the paper by Stewart (1979).

During our research, Bojanczyk, Brent, Van Dooren and de Hoog (1987) also developed an error analysis for Algorithm HY, but their results are some-what different from ours due to different approaches. It would be interesting to investigate the differences between these two independent results. Our immediate future work is to try to derive a backward stability result for Algorithm A such as that obtained for the LINPACK method by Stewart. Apparentely, the algorithms and theory of vector a developed for downdating can be similarly expanded to those for updating. We have done this already. A special paper to deal with updating and its applications is in preparation.

References

S.T. Alexander, C.-T. Pan and R.J. Plemmons, "Analysis of a recursive least squares hyperbolic rotation algorithm for signal processing," *Lin. Alg. Appl.*, to appear.

J. Bennett (1965), 'Triangular factors of modified matrices," *Numer. Math.*, 7, 217-21.

A.W. Bojanczyk, R.P. Brent, P. Van Dooren and F.R. de Hoog (1987), "A note on downdating the Cholesky factorization," *SIAM. Sci. Stat. Comp.*, 8, 201-21.

M.A. Brebner and J. Grad (1982), 'Eigenvalues of $Ax = \lambda Bx$ for real symmetric matrices A and B computed by reduction to a pseudosymmetric form and the HR process," *Lin. Alg. Appl.*, 43, 99-118.

J.R. Bunch (1987), 'The weak and strong stability of algorithms in numerical linear algebra," *Lin. Alg. Appl.*, 88/89, 49-66.

A. Bunse-Gerstner (1981), "An analysis of the HR algorithm for computing the eigenvalues of a matrix," *Lin. Alg. Appl.*, 35, 155-73.

J.M. Cioff and T. Kailath (1985), 'Windowed fast transversal filters adaptive algorithms with normalization," *IEEE Trans. on Acous., Speech, and Sig. Processing*, ASSP-33, 607-25.

J.M. Chambers (1971), 'Regression updating," *J. Amer. Stat. Assn.*, 66, 744-48.

G. Cybenko (1980), 'The numerical stability of Levinson-Durbin algorithm for Toeplitz system of equations," *SIAM J. Sci. Stat. Comp.*, 1, 303-19.

J.W. Daniel, W.B. Gragg, L. Kaufman and G.W. Stewart (1976), 'Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization," *Math. Comp.*, 30, 772-95.

J. Della Dora (1975), 'Numerical linear algorithms and group theory," *Lin. Alg. Appl.*, 10, 267-83.

J.-M. Delosme and I.C.F. Ipsen (1986), 'Parallel solution of symmetric positive definite systems with hyperbolic rotations," *Lin. Alg. Appl.*, 77, 75-112.

J.E. Dennis and R.B. Schnabel (1983), *Quasi-Newton Method for Nonlinear*

Problems, Prentice-Hall, Englewood Cliffs. N.J.

J. Dongarra, J.R. Bunch, C.B. Moler and G.W. Stewart (1978), *LINPACK Users Guide*, SIAM Publication, Philadalphia.

A. Einstein, H.A. Lorentz, H. Weyl and H. Minkowski (1923), *The Principle of Relativity*, Dover Publications, Inc.

L. Elsner (1979), "On some algebraic problems in connection with general eigenvalue algorithms," *Lin. Alg. Appl.*, 26, 123-38.

R. Fletcher and M. Powell (1974), "On the modification of LDL^T factorizations," *Math. Comp.*, 28, 1067-87.

R.E. Funderlic and R.J. Plemmons (1986), "Updating LU factorizations for computing stationary distributions," *SIAM J. Alg. Disc. Meth.*, 7, 30-42.

W.M. Gentleman (1973), "Least squares computations by Givens transformations without squares roots," *J. Inst. Math. Appl.*, 12, 329-36.

W.M. Gentleman (1975), "Error anslysis of *QR* decompositions by Givens transformations," *Lin. Alg. Appl.*, 10, 189-97.

P.E. Gill, W. Murray and M.A. Saunders (1975), "Methods for computing and modifying the *LDU* factors of a matrix," *Math. Comp.*, 29, 1051-77.

P.E. Gill, G.H. Golub, W. Murray and M.A. Saunders (1974), "Methods for modifying matrix factorizations," *Math. Comp.*, 28, 505-35.

P.E. Gill and W. Murray (1977), "Modification of matrix after a rank one change," *Proc. Conf. On the State of Art in Numerical Analysis at Univ. of York*, Academic Press, N.Y.

G.H. Golub (1969), "Matrix decompositions and statistical calculations," in R.C. Milton and J.A. Nelder(eds.), *Statistical Computation*, Academic Press, N.Y., 365-95.

G.H. Golub (1965), "Numerical methods for solving linear least squares problems," *Numer. Math.*, 7, 206-16.

C.H. Golub and M.A. Saunders (1970), *Linear least squares and quadratic programming*, North-Holland, Amsterdam.

G.H. Golub and C. Van Loan (1983), *Matrix Computation*, Johns Hopkins Press, Baltimore, M.D.

G.H. Hardy, J.E. Littlewood and G. Polya (1934), *Inequalities*, 1st ed, Cambridge Univ. Press., London and N.Y.

S. Haykin (1986), *Adaptive Filter Theory*, Prentice-Hall, Englewood Cliffs. N.J.

R.A. Horn and C.A. Johnson (1985), *Matrix Analysis*, Cambridge Univ. Press, N.Y.

T. Kailath, A. Vieira and M. Morf (1978), "Orthogonal transformation (square-root) implementations of the generalized Chandrasekhar and generalized Levinson algorithms," *Proc. 1978 IRIA Conf. on Syst. Optim. & Analysis*, edited by A. Bensoussan and J. Lions, Springer-Verlag, N.Y.,81-91.

C.L. Lawson and R.J. Hanson (1974), *Solving Least Squares Problems*, Prentice-Hall, Englewood Cliffs.

J. Makhoul (1975), "Linear prediction: a tutorial review," *Proc. IEEE*, 63, 561-80.

A.W. Marshall and I. Olkin (1979), *Inequalities: Theory of Majorization and its Applications*, Academic Press, N.Y.

W. Miller and C. Wrathall (1980), *Software for roundoff analysis of matrix algorithms*, Academic Press, N.Y.

M. Morf and J.-M. Delosme (1981), "Matrix decompositions and inversions via elementary signature-orthogonal transformations," *Proc. Int. Symposium on Mini and Micro Computers in Control and Measurement.*

C.-T. Pan and K. Sigmon (1986), "A sharp bound for products of hyperbolic plane rotations," Preprint.

C.C. Paige (1980), "Error Analysis of some techniques for updating orthogonal decompositions," *Math. Comp.*, 34, 465-71.

C. Rader and A.O. Steinhardt (1987), "Hyperbolic Householder transformations," *IEEE Trans. on Acous., Speech, and Sig. Processing*, Preprint.

M.A. Saunders (1972), "Large scale linear programming using the Cholesky factorization," *Stanford Univ. Report.* STAN-CS-72-252.

G.W. Stewart (1979), 'The effects of rounding error on an algorithm for downdating a Cholesky factorization," *J. Inst. Math. Appl.*, 23, 203-13.

G.W. Stewart (1973), *Introduction to Matrix Computations*, Academic Press, N.Y.

G.W. Stewart (1986), private communication.

J.H. Wilkinson (1965), *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford.