SOLVING THE INDEFINITE LEAST SQUARES PROBLEM

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

December 2002

Harikrishna Patel
Department of Mathematics

Contents

List of Tables

List of Figures

Abstract

This thesis is motivated by a problem that is a generalization of the linear least squares (LS) problem $\min_x (b - Ax)^T(b - Ax)$. The *indefinite* least squares (ILS) problem is to solve

$$\min_x (b - Ax)^T J(b - Ax), \qquad J = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}.$$

This problem reduces to the standard LS problem if $pq = 0$, while for $pq > 0$ the problem is to minimize a genuinely indefinite quadratic form. Applications of the ILS problem include the solution of total least squares problems and the area of optimization known as H^∞ smoothing.

Chandrasekaran, Gu and Sayed recently proposed the QR-Cholesky method, a backward stable method for solving the ILS problem that employs the QR and Cholesky factorizations. In this thesis, we describe and analyze the HQR method, a new method based on the hyperbolic QR factorization that has a lower operation count than the QR-Cholesky method. It is analogous to Golub's QR factorization method for solving the standard LS problem.

We explore J-orthogonal matrices and introduce the exchange operator of Stewart and Stewart which forms a bijective link between J-orthogonal and orthogonal matrices. This has surprisingly good implications on the stability of applying J-orthogonal transformations.

We look at hyperbolic rotations and show how crucial it is that they are formed and applied in a certain way to ensure that the final output is stable. We

examine conditions for applying products of J-orthogonal transformations stably, since applying one J-orthogonal transformation stably does not necessarily imply the same for products.

We show how hyperbolic rotations can be used to stably compute the hyperbolic QR factorization of a matrix, how the factorization is linked with the Cholesky downdating problem, and give a detailed error analysis.

We also give perturbation theory for the ILS problem and identify a condition number.

A detailed error analysis of the hyperbolic QR (HQR) method is given to prove that it is stable. Numerical results are given to back up this claim, and iterative refinement is implemented as a way of improving the accuracy of the results.

We also deal with the *equality constrained* indefinite least squares (ILSE) problem, an extension of the ILS problem, in which the variables are constrained by an underdetermined system of linear equations:

$$\min_x (b - Ax)^T J(b - Ax), \qquad J = \begin{bmatrix} -I_q & 0 \\ 0 & I_p \end{bmatrix}, \qquad Bx = d.$$

We analyze two new algorithms for solving this problem, which are generalizations of the QR-Cholesky and HQR methods for solving the ILS problem. We also introduce a new matrix factorization called the generalized hyperbolic QR factorization and use it to solve the ILSE problem in the second, HQR-based, algorithm. We give perturbation theory and error analyses for both methods and prove their stability. We also give numerical results to back up our analysis.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Copyright

Publications

- Material in Chapters 2, 3 and 4 is partly based on the technical report

  "Solving the Indefinite Least Squares Problem by Hyperbolic QR Factorization" (with Adam W. Bojanczyk and Nicholas J. Higham).

  Numerical Analysis Report No. 397, Manchester Centre for Computational Mathematics, Manchester, England, January 2002. 19 pp.

  `http://www.maths.man.ac.uk/~nareports/narep397.pdf`.

  This work is to appear in the SIAM Journal on Matrix Analysis and Applications, 24(4), 914-931, 2003.

- The material in Chapter 5 is based on the technical report

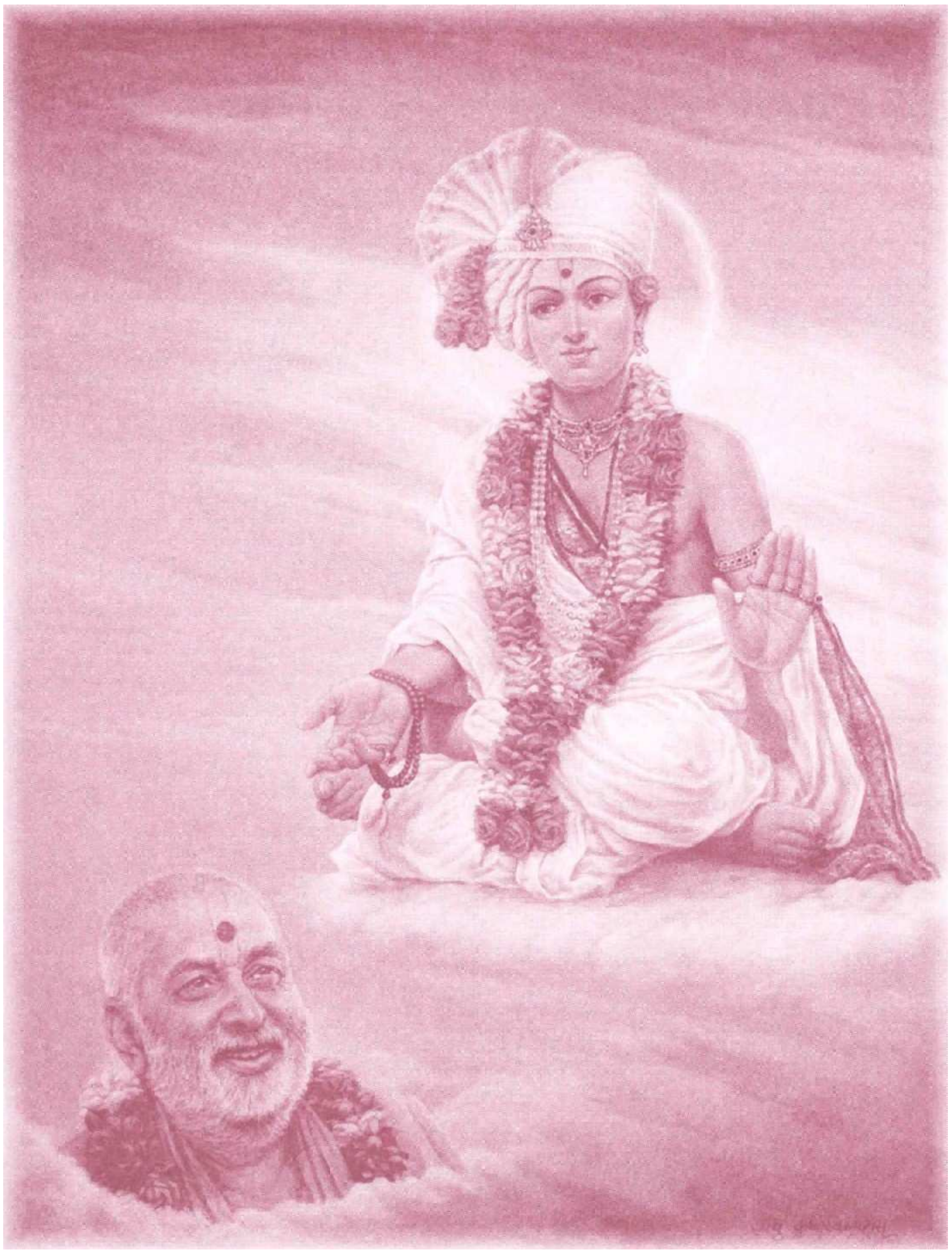
  "The Constrained Indefinite Least Squares Problem: Theory and Algorithms" (with Adam W. Bojanczyk and Nicholas J. Higham).

  Numerical Analysis Report No. 413, Manchester Centre for Computational Mathematics, Manchester, England, October 2002. 11 pp.

  `http://www.maths.man.ac.uk/~nareports/narep413.pdf`.

  This work is to appear in BIT.

॥ श्री स्वामिनारायणो विजयते ॥
Shree Swãminãrãyano Vijayaté
Victory to Shree Swãminãrãyan



Dedication

To Bhagwãn Swãminãrãyan and His fifth and present successor,
His Divine Holiness Shree Pramukh Swãmi Mahãrãj.

Acknowledgements

- First and foremost, I would like to express my most deepest gratitude to Bhagwãn Swãminãrãyan, the Supreme Lord God Almighty, for giving me the intellect and the ability to understand, think and reason, without which I would not have been able to learn even the alphabet. I also thank Him for the many hundreds of times during the whole seven years I have been here at the University of Manchester when I have experienced His presence in everything I have done and felt his support and guidance at all the times I needed it. I will forever be indebted to Him for all that He has done for me.

- Secondly, I would like to thank my dear beloved Guru, the fifth and present successor of Bhagwãn Swãminãrãyan, His Divine Holiness Shree Pramukh Swãmi Mahãrãj, whose inspiration, advice and blessings were paramount in my decision to do a Ph.D.

- Thirdly, I thank my parents for the hard effort they put into bringing me up from childhood, and the constant encouragement they gave me while studying.

- Fourthly, many thanks to Professor Nicholas J. Higham for his excellent and extremely patient supervision during my Ph.D.

- Fifthly, thanks also to all my relatives, friends, lecturers, visiting staff and colleagues who have helped to make my time here at the University of Manchester very enjoyable. In particular, thanks to Doctors Françoise Tisseur, Bobby Cheng, Philip Davies and Hyun–Min Kim for many helpful suggestions. Special thanks also to Professor Adam W. Bojanczyk from the School of Electrical and Computer Engineering, Cornell University, Ithaca, NY, USA, who visited the University of Manchester during the latter part of 2001, and collaborated with Professor Nicholas J. Higham and myself to produce the two technical reports listed in the Publications section on page 13.

- Sixthly, the support of an Engineering and Physical Sciences Research Council Ph.D. Studentship is gratefully acknowledged.
- Finally, I am very grateful to the publishers Swãminãrãyan Aksharpith for granting me permission to reproduce the image on the dedication page. This image is of a watercolour painting of Bhagwãn Swãminãrãyan and His Divine Holiness Shree Pramukh Swãmi Mahãrãj, whose artist is Sãdhu Krishnaswarupdãs and was originally printed on the dedication page of the book "Handbook to the Vachanãmrutam (Spiritual Teachings of Bhagwãn Swãminãrãyan)" by Sãdhu Mukundcharandãs, Swãminãrãyan Aksharpith, Shãhibãg, Amdãvãd – 380 004, Gujarat, India, March 1999. xxiv + 258 pp. ISBN: 81–7526–135–8.

Chapter 1

Introduction

1.1 Overview

This thesis is motivated by a problem that is a generalization of the linear least squares (LS) problem

$$\min_x (b - Ax)^T (b - Ax).$$

The *indefinite* least squares (ILS) problem is to solve

$$\min_x (b - Ax)^T J (b - Ax), \qquad J = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}. \tag{1.1.1}$$

This problem reduces to the standard LS problem if $pq = 0$, while for $pq > 0$ the problem is to minimize a genuinely indefinite quadratic form. The application of the ILS problem to the solution of total least squares problems [48] and to the area of optimization known as H^∞ smoothing [38], [64] is discussed by Chandrasekaran, Gu and Sayed [14], who recently proposed the QR-Cholesky method, a backward stable method for solving the ILS problem that employs the QR and Cholesky factorizations.

In this thesis, we describe and analyze the HQR method, a new method for solving the ILS problem, that is based on the hyperbolic QR factorization and has a lower operation count than the QR-Cholesky method. The HQR method is analogous to Golub's QR factorization method [32] for solving the standard LS problem.

Firstly, we breifly go over applications of the ILS problem and the necessary theoretical background in Chapter 1. In Chapter 2, we start by exploring J-orthogonal matrices and their properties. We introduce the exchange operator of Stewart and Stewart [74] which forms a bijective link between J-orthogonal and orthogonal matrices. This has surprisingly good implications on the stability of applying J-orthogonal transformations.

We look at several representations of hyperbolic rotations, the J-orthogonal counterpart of orthogonal Givens rotations, and show how crucial it is that the rotation is formed and applied in a certain way to ensure that the final output is stable.

We look at conditions for applying products of J-orthogonal transformations stably, since applying one J-orthogonal transformation stably does not necessarily imply the same for products.

We show how hyperbolic rotations can be used to stably compute the hyperbolic QR factorization of a matrix, how the factorization is linked with the Cholesky downdating problem, and give a detailed error analysis for computing the hyperbolic QR factorization.

In Chapter 3, we give an outline of the ILS problem and look at its solution via its normal equations and augmented system. We look at perturbation theory and describe the QR-Cholesky method in detail.

Chapter 4 is dedicated entirely to the HQR method of solving the ILS problem. A detailed error analysis is given to prove that this method is stable. Numerical results are given to back up this claim, and iterative refinement is implemented to see if it leads to an improvement in the results.

In Chapter 5, we deal with the *equality constrained* indefinite least squares

(ILSE) problem, an extension of the ILS problem, in which the variables are constrained by an underdetermined system of linear equations:

$$\min_x (b - Ax)^T J(b - Ax), \qquad J = \begin{bmatrix} -I_q & 0 \\ 0 & I_p \end{bmatrix}, \qquad Bx = d.$$

We analyze two new algorithms for solving this problem, which are generalizations of the QR-Cholesky and HQR methods for solving the ILS problem. We also introduce a new matrix factorization called the generalized hyperbolic QR factorization and use it to solve the ILSE problem in the second, HQR-based, algorithm. We give perturbation theory and error analyses for both methods and prove their stability. We give numerical results to back up our claim.

1.2 Applications

Several problems can be reduced to (1.1.1). These include the *total least squares* problem [48] and problems in the area of optimization known as *robust* or H^∞ *smoothing* [38], [64]. These are both outlined by Chandrasekaran, Gu and Sayed [14]. We give a brief description of these problems here.

Let $A \in \mathbb{R}^{m \times n}$ be a given matrix with $m \geq n$, let $b \in \mathbb{R}^m$ be a given vector and let $x \in \mathbb{R}^n$ be an unknown vector of parameters. Then A and b are linearly related by the equation

$$b = Ax + v, \tag{1.2.1}$$

where $v \in \mathbb{R}^m$ explains the mismatch between Ax and b. We may also refer to b as the observation vector. We assume that $v \neq 0$.

1.2.1 The Total Least Squares Problem

The solution of the total least squares problem takes into account data errors in both A and b. In other words, given the pair (A, b) and assuming that both data quantities are noisy or contaminated, the total least squares problem seeks a matrix $\widehat{A}$ and a vector $\widehat{x}$ that solve the following minimization problem:

$$\min_{\widehat{A},\widehat{x}} \|[\,\widehat{A}-A \quad \widehat{A}\widehat{x}-b\,]\|_F^2 \iff \min_{\widehat{A},\widehat{b}\in\mathcal{R}(\widehat{A})} \|[\,A \quad b\,]-[\,\widehat{A} \quad \widehat{b}\,]\|_F^2, \tag{1.2.2}$$

where $\widehat{b} = \widehat{A}\widehat{x}$. The optimal solution $\widehat{A}$ is regarded as an approximation to A. This is in turn used to determine an $\widehat{x}$ that guarantees $\widehat{b} \in \mathcal{R}(\widehat{A})$. The solution of the total least squares problem is known to be given by the following construction [48, p. 36].

Assume that $m > n$. Let $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ be the singular values of A with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Also, let $\{\overline{\sigma}_1, \overline{\sigma}_2, \ldots, \overline{\sigma}_{n+1}\}$ denote the singular values of the augmented matrix $[A\ b]$. If $\sigma_n > \overline{\sigma}_{n+1}$ then there is a unique solution to the total least squares problem (1.2.2) [14],[48]:

$$\widehat{x} = \left(A^TA - \overline{\sigma}_{n+1}^2 I_n\right)^{-1} A^Tb. \tag{1.2.3}$$

Hence

$$\left(A^TA - \overline{\sigma}_{n+1}^2 I_n\right)\widehat{x} = A^Tb,$$

which is equivalent to

$$A^T\left(A\widehat{x} - b\right) - \overline{\sigma}_{n+1}^2 \widehat{x} = 0. \tag{1.2.4}$$

The unique vector $\widehat{x}$ in (1.2.3) also solves the minimization problem

$$\min_x \left(\|b - Ax\|_2^2 - \overline{\sigma}_{n+1}^2 \|x\|_2^2\right), \tag{1.2.5}$$

since (1.2.4) are its normal equations. We can rewrite (1.2.5) in the form

$$\min_x \left(\begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \overline{\sigma}_{n+1}I_n \end{bmatrix} x\right)^T \begin{bmatrix} I_m & 0 \\ 0 & -I_n \end{bmatrix} \left(\begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \overline{\sigma}_{n+1}I_n \end{bmatrix} x\right).$$

This is now in the same form as the ILS problem (1.1.1).

A similar cost function also arises in the solution of a LS problem with bounded errors-in-variables [13].

1.2.2 Robust or H^∞ Smoothing

Robust or H^∞ smoothing is the name given to methods used for min-max estimation. We review the formulation of a basic H^∞ smoothing method below.

Consider once again the model (1.2.1). This time, we assume that an arbitrary vector $\widehat{x}$ has been picked as an estimate for the unknown vector x. Then we can always find a vector $\widehat{v}$ such that (1.2.1) is satisfied no matter what the given (A, b) are. In other words, we have

$$b = A\widehat{x} + \widehat{v}.$$

We have an induced error norm for our choice of $\widehat{x}$, $\|x - \widehat{x}\|_2$, and a norm for the *noise*, $\|\widehat{v}\|_2$. Note that because we have chosen $\widehat{x}$ arbitrarily, these norms can be arbitrarily large or small. In other words, our estimate for $\widehat{x}$ may be good or bad.

Now, what we want is a procedure that guarantees a certain level of performance. In other words, we want to be able to produce an estimate, $\widehat{x}$, such that if v in (1.2.1) is small, the resulting error, $x - \widehat{x}$, is also small. This idea can be formalized into what is called a robust estimation problem:

We seek an $\widehat{x}$ such that

$$\max_x \frac{\|x - \widehat{x}\|_2^2}{\|b - Ax\|_2^2} \le \gamma^2,$$

since $v = b - Ax$, for some specified value of γ. We can rewrite this as

$$\max_x (\gamma^{-2}\|x - \widehat{x}\|_2^2 - \|b - Ax\|_2^2) \le 0$$

or

$$\min_x (\|Ax - b\|_2^2 - \gamma^{-2}\|x - \widehat{x}\|_2^2) \ge 0. \tag{1.2.6}$$

We can now see that another way of writing (1.2.6) is

$$\min_x \left(\begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \gamma^{-1} I_n \end{bmatrix} z \right)^T \begin{bmatrix} I_m & 0 \\ 0 & -I_n \end{bmatrix} \left(\begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \gamma^{-1} I_n \end{bmatrix} z \right),$$

where $c = b - A\widehat{x}$ and $z = x - \widehat{x}$. We once again end up with a problem which is special case of (1.1.1).

Table 1.3.1: Standard terminology used throughout the thesis. In all cases, $A = [a_{ij}]_{i,j=1:m} \in \mathbb{R}^{m\times m}$.

Terminology	Definition
Symmetric	$A = A^T$, where A^T is the transpose of A
Positive Definite	$x^T Ax > 0$ for all $x \in \mathbb{R}^m$
Orthogonal	$A^T A = I_m$, where I_m is the $m \times m$ identity matrix
Upper (Lower) Triangular	$a_{ij} = 0$ for $i > j$ $(i < j)$
Diagonal	$a_{ij} = 0$ for $i \neq j$

1.3 Basic Definitions and Terminology

Throughout this thesis, we will adopt the standard terminology given in Table 1.3.1 for real matrices.

We remark here that the definition of a diagonal matrix in Table 1.3.1 can be extended to rectangular matrices. The same definition still applies. Also, a diagonal matrix A with diagonal elements $\alpha_1, \alpha_2, \ldots, \alpha_m$ ordered from the top-left element can be written concisely as $A = \operatorname{diag}(\alpha_1, \alpha_2, \ldots, \alpha_m)$.

The same remark applies to the definition of upper (lower) triangular matrices when they are rectangular. Once again, the same definition still applies, however, the matrices are now referred to as being *upper (lower) trapezoidal* due to the pattern of zero and nonzero elements of the matrix.

Two fundamental subspaces associated with any matrix $A = [a_1\ a_2\ \cdots\ a_n] \in \mathbb{R}^{m\times n}$ that we will be making use of are

$\mathcal{R}(A) = \operatorname{span}(a_1, a_2, \ldots, a_n)$	the *range* or *column space* of A
$\mathcal{N}(A) = \{x : Ax = 0\}$	the *null space* of A

Note that

$$\operatorname{rank}(A) = \dim(\mathcal{R}(A)) + \dim(\mathcal{N}(A)).$$

1.4 Norms

Norms provide us with a convenient scalar measure of the size of a vector or a matrix. They are a particularly useful tool in perturbation theory and rounding error analysis where they enable results to be presented in a concise form. We first look at vector norms.

1.4.1 Vector Norms

A vector norm is a function $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}$ that satisfies the following conditions:

1. $\|x\| > 0$ for all nonzero $x \in \mathbb{R}^n$.

2. $\|\alpha x\| = |\alpha|\,\|x\|$, where $\alpha \in \mathbb{R}$, $x \in \mathbb{R}^n$.

3. The triangle inequality holds: $\|x + y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{R}^n$.

The three most useful norms in error analysis and in numerical computation are

$$\begin{aligned} \|x\|_1 &= \sum_{i=1}^{n} |x_i|, \\ \|x\|_2 &= \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2} = (x^T x)^{1/2}, \\ \|x\|_\infty &= \max_{1 \leq i \leq n} |x_i|. \end{aligned}$$

These are all special cases of the class of vector norms known as the Hölder p-norms defined by

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}, \qquad p \geq 1.$$

The 2-norm, also commonly known as the *Euclidean norm*, has the useful property that it is invariant under orthogonal transformations Q, for if $Q^T Q = I$, then

$$\|Qx\|_2^2 = x^T Q^T Q x = x^T x = \|x\|_2^2 .$$

All vector p-norms are equivalent, which means that there exist constants α, β such that

$$\alpha \left\|x\right\|_p \leq \left\|x\right\|_q \leq \beta \left\|x\right\|_p$$

for $p, q \geq 1$. This is particularly useful in perturbation theory and error analysis where it is often necessary to switch between norms. Therefore inequalities that bound one norm in terms of another are required. For the 1-, 2- and ∞-norms, we have the inequalities

$$\begin{aligned}
\left\|x\right\|_2 &\leq \left\|x\right\|_1 \leq \sqrt{n} \left\|x\right\|_2 , \\
\left\|x\right\|_\infty &\leq \left\|x\right\|_2 \leq \sqrt{n} \left\|x\right\|_\infty , \\
\left\|x\right\|_\infty &\leq \left\|x\right\|_1 \leq n \left\|x\right\|_\infty .
\end{aligned}$$

This follows from a more general result for $p > q$ by Gastinel [28, pp. 27–28] and Goldberg and Straus [31, Lemma 1.1], both in 1983, that

$$\left\|x\right\|_p \leq \left\|x\right\|_q \leq n^{\left(\frac{1}{q}-\frac{1}{p}\right)} \left\|x\right\|_p .$$

1.4.2 Matrix Norms

A matrix norm is a function $\|\cdot\| : \mathbb{R}^{m\times n} \rightarrow \mathbb{R}$ that satisfies the following conditions:

1. $\|A\| > 0$ for all nonzero $A \in \mathbb{R}^{m\times n}$.
2. $\|\alpha A\| = |\alpha| \, \|A\|$, where $\alpha \in \mathbb{R}$, $A \in \mathbb{R}^{m\times n}$.
3. The triangle inequality holds: $\|A + B\| \leq \|A\| + \|B\|$ for all $A, B \in \mathbb{R}^{m\times n}$.

The most commonly used matrix norms are the Frobenius norm,

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 \right)^{1/2} = (\text{trace}(A^T A))^{1/2},$$

and the 1-, 2- and ∞-norms

$$\begin{aligned}\|A\|_1 &= \max_{1\le j\le n}\sum_{i=1}^{m}|a_{ij}|,\\ \|A\|_2 &= (\rho(A^TA))^{1/2} = \sigma_{\max}(A),\\ \|A\|_\infty &= \max_{1\le i\le m}\sum_{j=1}^{n}|a_{ij}|,\end{aligned}$$

where the spectral radius

$$\rho(B) = \max\{|\lambda| : \det(B-\lambda I)=0\}$$

and $\sigma_{\max}(B)$ denotes the largest singular value of B. Once again, the 1-, 2- and ∞-norms all belong to the class of subordinate matrix norms defined by

$$\|A\| = \max_{x\neq 0}\frac{\|Ax\|}{\|x\|} = \max_{\|x\|=1}\|Ax\|\,.$$

Both the Frobenius norm and the matrix 2-norm are unitarily invariant norms. This means that for any orthogonal (or unitary, in the complex case) matrices U and V we have $\|UAV\|_2 = \|A\|_2$ and $\|UAV\|_F = \|A\|_F$. Also for unitarily invariant norms, the useful property holds that $\|A^T\| = \|A\|$.

The unitary invariance of the Frobenius and 2-norms has implications in error analysis, for it means that multiplication by orthogonal matrices does not magnify errors. For example, if $A \in \mathbb{R}^{m\times n}$ is contaminated by errors E, and Q is orthogonal, then

$$Q(A+E)Q^T = QAQ^T + F,$$

and $\|F\|_F = \left\|QEQ^T\right\|_F = \|E\|_F$, and similarly, $\|F\|_2 = \|E\|_2$.

A norm is *consistent* if $\|AB\| \le \|A\|\,\|B\|$ whenever the product AB is defined. The Frobenius norm and all subordinate matrix norms are consistent.

Just as with vector norms, Table 1.4.1 shows that the Frobenius norm and the subordinate matrix 1-, 2- and ∞-norms are all equivalent. The table gives constants α_{pq} such that $\|A\|_p \le \alpha_{pq}\|A\|_q$, $A \in \mathbb{R}^{m\times n}$.

Table 1.4.1: Constants α_{pq} such that $\|A\|_p \le \alpha_{pq} \|A\|_q$, $A \in \mathbb{R}^{m\times n}$.

$p \backslash q$	1	2	∞	F
1	1	$\sqrt{m}$	m	$\sqrt{m}$
2	$\sqrt{n}$	1	$\sqrt{m}$	1
∞	n	$\sqrt{n}$	1	$\sqrt{n}$
F	$\sqrt{n}$	$\sqrt{\text{rank}(A)}$	$\sqrt{m}$	1

The following result will be needed in later chapters. For a proof see Higham [42, Lemma 6.6(c)].

Lemma 1.4.1 *Let $A, B \in \mathbb{R}^{m\times n}$. If $|A| \le |B|$ then $\|A\|_2 \le \sqrt{\text{rank}(B)}\, \|B\|_2$.* $\square$

1.5 The Kronecker Product and Vec Operator

A notion that is useful in the study of matrix equations is that of the *Kronecker product*, *direct product* or *tensor product* of matrices [45, Chapter 4]. We shall be using Kronecker products to help us obtain perturbation bounds for solving the ILS and ILSE problems.

The *Kronecker product*, $A\otimes B$, of a matrix $A \in \mathbb{R}^{m\times n}$ with a matrix $B \in \mathbb{R}^{p\times q}$ is the block matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp\times nq}.$$

Note that $A \otimes B \ne B \otimes A$ in general.

It is sometimes more convenient to think of matrices as vectors by ordering their entries in a conventional way. We adopt the common convention of stacking the columns from left to right. With each matrix $A = [a_1\ a_2\ \cdots\ a_n] \in \mathbb{R}^{m\times n}$, where $a_i \in \mathbb{R}^m$, $i = 1\colon n$, we associate the vector

$$\mathrm{vec}(A) := [\, a_1^T \quad a_2^T \quad \cdots \quad a_n^T \,]^T \in \mathbb{R}^{mn}.$$

Clearly, the vec operator gives us the identity

$$\|A\|_F = \|\mathrm{vec}(A)\|_2 \tag{1.5.1}$$

for any matrix A.

A historical study of Kronecker products and the vec operator has been done by Henderson [40].

The Kronecker product can be used to give a convenient representation for many linear matrix transformations and linear matrix equations with the aide of the vec operator. A key observation is the following.

Lemma 1.5.1 *Let the matrices $A \in \mathbb{R}^{m\times n}$, $X \in \mathbb{R}^{n\times p}$ and $B \in \mathbb{R}^{p\times q}$. Then*

$$\mathrm{vec}(AXB) = (B^T \otimes A)\mathrm{vec}(X).$$

Proof. Clearly

$$\begin{aligned}(AXB)(:,k) &= A(XB)(:,k)\\ &= AXB(:,k)\\ &= A\left[\sum_{i=1}^{p} B(i,k)X(:,i)\right]\\ &= [\, B(1,k)A \quad B(2,k)A \quad \cdots \quad B(p,k)A \,]\,\mathrm{vec}(X)\\ &= (B(:,k)^T \otimes A)\mathrm{vec}(X).\end{aligned}$$

Thus,

$$\mathrm{vec}(AXB) = \begin{bmatrix} B(:,1)^T \otimes A \\ B(:,2)^T \otimes A \\ \vdots \\ B(:,q)^T \otimes A \end{bmatrix} \mathrm{vec}(X),$$

but $[(B(:,1)^T \otimes A)^T \ (B(:,2)^T \otimes A)^T \ \cdots \ (B(:,q)^T \otimes A)^T]^T = B^T \otimes A$ since the transpose of a column of B is a row of B^T and thus proving the result. $\square$

Since A and A^T are matrices with the same elements, it is obvious that we must be able to obtain $\mathrm{vec}(A^T)$ by applying a permutation to $\mathrm{vec}(A)$. The next result links the vectors $\mathrm{vec}(A)$ and $\mathrm{vec}(A^T)$.

Lemma 1.5.2 *Let m, n be given positive integers. Then there is a unique matrix $\Pi(m,n) \in \mathbb{R}^{mn\times mn}$ such that*

$$\mathrm{vec}(A^T) = \Pi(m,n)\mathrm{vec}(A)$$

for all $A \in \mathbb{R}^{m\times n}$. The matrix $\Pi(m,n)$ depends only on the dimensions m and n and is given by

$$\Pi(m,n) = \sum_{i=1}^{m}\sum_{j=1}^{n} E_{ij} \otimes E_{ij}^T = [\, E_{ij}^T \,]_{i=1:m,j=i:n}\,,$$

where each $E_{ij} \in \mathbb{R}^{m\times n}$ differs from the $m \times n$ zero matrix only in the element $(i,j) = 1$. Moreover, $\Pi(m,n)$ is a permutation matrix and $\Pi(m,n) = \Pi(n,m)^{-1} = \Pi(n,m)^T$.

Proof. Note that $E_{ij}^T A E_{ij}^T = A(i,j)E_{ij}^T$, $i = 1{:}\,m$, $j = i{:}\,n$ for any matrix $A \in \mathbb{R}^{m\times n}$. Thus

$$A^T = \sum_{i=1}^{m}\sum_{j=1}^{n} A(i,j)E_{ij}^T = \sum_{i=1}^{m}\sum_{j=1}^{n} E_{ij}^T A E_{ij}^T.$$

Now using Lemma 1.5.1,

$$\mathrm{vec}(A^T) = \sum_{i=1}^{m}\sum_{j=1}^{n} \mathrm{vec}\left(E_{ij}^T A E_{ij}^T\right) = \sum_{i=1}^{m}\sum_{j=1}^{n} (E_{ij} \otimes E_{ij}^T)\mathrm{vec}(A).$$

which proves the first part of the lemma.

Since $(A^T)^T = A$ and $A^T \in \mathbb{R}^{n\times m}$, we have $\mathrm{vec}(A) = \Pi(n,m)\mathrm{vec}(A^T) = \Pi(n,m)\Pi(m,n)\mathrm{vec}(A)$, so $\Pi(n,m) = \Pi(m,n)^{-1}$. Finally,

$$\Pi(n,m) = \sum_{i=1}^{n}\sum_{j=1}^{m} E_{ji}^T \otimes E_{ji} = \sum_{i=1}^{m}\sum_{j=1}^{n} \left(E_{ij} \otimes E_{ij}^T\right)^T = \Pi(m,n)^T.$$

In other words, $\Pi(m,n)$ is a matrix of zeroes and ones for which $\Pi(m,n)^{-1} = \Pi(m,n)^T$, hence it is a permutation matrix. $\square$

1.6 The Moore-Penrose Pseudo-inverse

For any matrix $A \in \mathbb{R}^{m\times n}$, it can be shown that there exists a unique matrix X that satisfies the four *Penrose conditions*:

$$AXA = A, \tag{1.6.1a}$$

$$XAX = X, \tag{1.6.1b}$$

$$(AX)^T = AX, \tag{1.6.1c}$$

$$(XA)^T = XA. \tag{1.6.1d}$$

The matrix X is a generalized inverse of A. It is customary to denote X by A^+ and refer to it as the *pseudo-inverse* of A. A neat algebraic proof of the uniqueness of the pseudo-inverse is given by Kalman [50, p. 112], and a constructive proof is given by Stewart and Sun [73, p. 103]. The conditions in (1.6.1) were given by Roger Penrose in 1955, although the pseudo-inverse itself was originally discovered by E. H. Moore in 1920.

Important properties of A^+ include

- $(A^+)^+ = A$,
- $(A^+)^T = (A^T)^+$,
- $(\alpha A)^+ = \alpha^{-1}A^+$, where α is a scalar,
- $(A^TA)^+ = A^+(A^+)^T$,
- $(UAV^T)^+ = VA^+U^T$ for U, V orthogonal.

More detailed information on generalized inverses is given by Ben-Israel and Greville [6].

1.7 Stability

In solving the ILS problem, there is one very important aspect of a numerical algorithm that we have to take into account—this is its *stability.* In what follows, we will define stability, so that we can analyze the stability of the algorithms in this document. The classical references on stability analysis are "Rounding Errors in Algebraic Processes" [80] and "The Algebraic Eigenvalue Problem" [81], both by Wilkinson. However, we will use the more recent and more up-to-date reference by Higham [42] in most of what we do.

Suppose that we are computing the vector $y = f(x)$, where x is also a vector and f is a vector function. We denote the computed value of y by $\widehat{y}$.

- If Δx is such that $\widehat{y} = f(x + \Delta x)$ exactly, we say that the smallest such $|\Delta x|$, possibly divided by $|x|$, is called the *backward error.* If we can find a suitably small bound for this, then we say that the computation of y is *backward stable.* In other words, the solution we get is the correct solution to *almost* the correct problem.

- The absolute or relative error of $\widehat{y}$ is called the *forward error.* If we can find a suitably small bound for this, then we say that the computation of y is *forward stable.* In other words, the solution we get is *almost* the correct solution to the correct problem.

- Sometimes, we may not be able to find a relatively small Δx satisfying $\widehat{y} = f(x + \Delta x)$, but we may be able to find Δx and Δy such that $\widehat{y} + \Delta y = f(x + \Delta x)$, with relatively small Δx and Δy. This type of result is called a *mixed forward-backward error* result. If we can find a suitably small bound for both Δx and Δy, then we say that the computation of y is *mixed forward-backward stable.* In other words, the solution we get is *almost* the correct solution to *almost* the correct problem.

In the work that follows, we are interested in algorithms that are either backward, forward or mixed forward-backward stable.

1.7.1 Model of Arithmetic

We use the standard model for floating point arithmetic:

$$fl(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta_1) = \frac{x \operatorname{op} y}{1+\delta_2}, \quad |\delta_1|, |\delta_2| \le u, \quad \operatorname{op} = +, -, *, /,$$
$$fl(\sqrt{x}) = \sqrt{x}(1+\delta), \quad |\delta| \le u,$$

where u is the unit roundoff of the computer.

1.7.2 Basic Stability Results

The following particularly elegant lemma of Higham [42, Lemma 3.1] helps us to simplify the error analysis considerably.

Lemma 1.7.1 *If $|\delta_i| \le u$ and $\rho_i = \pm 1$ for $i = 1\colon n$, and $nu < 1$, then*

$$\prod_{i=1}^{n} (1+\delta_i)^{\rho_i} = 1 + \theta_n,$$

where

$$|\theta_n| \le \frac{nu}{1-nu} =: \gamma_n. \qquad \square$$

We define

$$\widetilde{\gamma}_k = \frac{cku}{1-cku}, \tag{1.7.1}$$

where c denotes a small integer constant whose exact value is unimportant. This saves us from having to compute the n in γ_n precisely. We will also write $\widetilde{\theta}_k$ to denote a quantity satisfying $|\widetilde{\theta}_k| \le \widetilde{\gamma}_k$. We may resort to these modifications to make the analysis simpler.

In many complicated analyses based on Lemma 1.7.1 it will be necessary to manipulate the $1+\theta_k$ and γ_k terms. The next lemma of Higham [42, Lemma 3.3] provides the necessary rules.

Lemma 1.7.2 *For any positive integer k let θ_k denote a quantity bounded according to $|\theta_k| \le \gamma_k = ku/(1-ku)$. The following relations hold:*

$$\begin{aligned}(1+\theta_k)(1+\theta_j) &= 1+\theta_{k+j}\\ \frac{(1+\theta_k)}{(1+\theta_j)} &= \begin{cases} 1+\theta_{k+j}, & j \le k,\\ 1+\theta_{k+2j}, & j > k,\end{cases}\\ \gamma_k\gamma_j &\le \gamma_{\min(k,j)}, \qquad \max(j,k)u \le 1/2,\\ i\gamma_k &\le \gamma_{ik},\\ \gamma_k + u &\le \gamma_{k+1},\\ \gamma_k+\gamma_j+\gamma_k\gamma_j &\le \gamma_{k+j}. \qquad \square\end{aligned}$$

Computed quantities will be denoted with a hat or using the $fl(\,\cdot\,)$ notation.

We remark here that when we state an inequality between two matrices or two vectors of the same size, then the result holds componentwise. In other words, if we have $A, B \in \mathbb{R}^{m\times n}$, then we interpret $A \le B$ as $a_{ij} \le b_{ij}$ for each $i = 1\colon m$, $j = 1\colon n$.

Also, for any matrix $A \in \mathbb{R}^{m\times n}$, $|A|$ will be the matrix with entries $|a_{ij}|$, $i = 1\colon m$, $j = 1\colon n$, and similarly for vectors.

We will also use the following results of Higham [42, Sections 3.1 and 3.5].

Lemma 1.7.3 *Let $x, y \in \mathbb{R}^n$. The computed inner product x^Ty satisfies the backward error result*

$$fl(x^Ty) = (x+\Delta x)^T y = x^T(y+\Delta y), \qquad |\Delta x| \le \gamma_n|x|, \qquad |\Delta y| \le \gamma_n|y|,$$

and the forward error bound

$$\left|x^Ty - fl(x^Ty)\right| \le \gamma_n|x|^T|y|.$$

Proof. It is straightforward to see that

$$fl(x^Ty) = (\ldots((x_1y_1(1+\delta_1) + x_2y_2(1+\delta_2))(1+\delta_{n+1})$$

$$
\begin{aligned}
&+ x_3 y_3 (1+\delta_3)) (1+\delta_{n+2}) \\
&+ \cdots + x_n y_n (1+\delta_n)) (1+\delta_{2n-1}), \qquad (1.7.2)
\end{aligned}
$$

where $|\delta_i| \le u$ for all i. Using Lemma 1.7.1, we then have

$$fl(x^T y) = x_1 y_1 (1+\theta_n) + x_2 y_2 (1+\theta'_n) + x_3 y_3 (1+\theta_{n-1}) + \cdots + x_n y_n (1+\theta_2).$$

This is what we get for one order of evaluation of $x^T y$. We can generalize this equation to say that the sum of the $x_i y_i$, $i = 1\colon n$, can be accumulated in any order, and hence a "worst case scenario" for computing $fl(x^T y)$ would be

$$fl(x^T y) = x_1 y_1 (1+\theta_n) + x_2 y_2 (1+\theta'_n) + \cdots + x_n y_n (1+\theta''_n).$$

We can say that this computed inner product is the exact one for the perturbed set of data points $x_1, x_2, \ldots, x_n, y_1(1+\theta_n), y_2(1+\theta'_n), \ldots, y_n(1+\theta''_n)$, or, alternatively, $x_1(1+\theta_n), x_2(1+\theta'_n), \ldots, x_n(1+\theta''_n), y_1, y_2, \ldots, y_n$. This gives us the backward error result that we require. The forward error bound follows immediately. $\square$

Lemma 1.7.4 *Let $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$. We form the product $y = Ax$. We then have the backward error result*

$$\widehat{y} = (A + \Delta A)\, x, \qquad |\Delta A| \le \gamma_n |A|,$$

and the forward error bound

$$|y - \widehat{y}| \le \gamma_n |A|\, |x|.$$

Proof. The vector y can be formed as m inner products, $y_i = a_i^T x$, $i = 1\colon m$, where a_i^T is the ith row of A. Using Lemma 1.7.3 we have

$$\widehat{y}_i = (a_i + \Delta a_i)^T x, \qquad |\Delta a_i| \le \gamma_n |a_i|,$$

which is the ith equation in the backward error result that we require. Once again, the forward error bound follows immediately. $\square$

One more result that we will be using provides us with a convenient way to bound the effect of perturbations of a matrix product using norms. For a proof see Higham [42, Lemma 3.6].

Lemma 1.7.5 *If $X_j + \Delta X_j \in \mathbb{R}^{m\times m}$ satisfies $\|\Delta X_j\| \le \delta_j \|X_j\|$ for all j for a consistent norm, then*

$$\left\| \prod_{j=0}^{m}(X_j + \Delta X_j) - \prod_{j=0}^{m} X_j \right\| \le \left(\prod_{j=0}^{m}(1+\delta_j) - 1 \right) \prod_{j=0}^{m} \|X_j\| . \qquad \square$$

1.7.3 Conditioning

The relationship between the forward and backward error for a problem is governed by the conditioning of the problem, that is, the sensitivity of the solution to perturbations in the data. In other words, the (*relative*) *condition number* of a problem is the relative change in the output for a given relative change in the input.

When backward error, forward error, and the condition number are defined in a consistent fashion we have the useful rule of thumb that

$$\text{forward error} \lesssim \text{condition number} \times \text{backward error},$$

with approximate equality possible.

1.8 Computational Complexity

Computational complexity is the name given to the number of floating point operations needed to run an algorithm.

We define a *floating point operation* to be one of the basic arithmetic operations, $+$, $-$, $*$ or $/$, or the square-root operation, $\surd$. Say, for example, an algorithm to solve a particular problem requires p floating point operations to solve, where p is a function depending on the dimensions of the problem. For tiny problems,

Table 1.8.1: The number of flops needed to compute basic vector and matrix products.

Matrix Operation	Dimensions	Flops
Vector Inner Product — $x^T y$	$x, y \in \mathbb{R}^n$	$2n$
Vector Outer Product — xy^T	$x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$	mn
Matrix-Vector Multiplication — Ax	$A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$	$2mn$
Matrix-Matrix Multiplication — AB	$A \in \mathbb{R}^{m\times p}$, $B \in \mathbb{R}^{p\times n}$	$2mnp$
Matrix-Matrix Multiplication — $A^T A$	$A \in \mathbb{R}^{m\times n}$	mn^2

the nature of p as a function is not important. In practice, however, we would not want p to be any function other than a polynomial, and usually not of order greater than 3, as this would make the algorithm too costly to run in terms of time for most real life problems. We will abbreviate floating point operations to *flops*.

We summarize the number of flops needed to carry out simple vector and matrix operations in Table 1.8.1. We omit the simple proofs. We give only the terms of the flops of highest order since these are the ones that are dominant for large problems.

1.9 Design of Numerical Experiments

With all work that we will carry out in this thesis, we will give stability results to backup our claim of an algorithm being stable. However, it is helpful to see if these results meet our expectations by running the algorithm on some test problems designed in such a way that they exploit all the potential weaknesses and delicate computations that are performed in running the algorithm. The final output of the algorithm can then be tested by one of three methods: either by computing a more accurate solution by an algorithm known to be stable but expensive, using extended precision arithmetic on the original algorithm, or testing the values of the output. The last method means that if one of the output values is meant to be an orthogonal matrix M, say, then $M^TM - I$ or its norm is computed, where I is the identity matrix, to see how close it is to the zero matrix. If there is more than one output from the algorithm in question and they are known to be theoretically related, then this relation is tested by appropriate means.

In all the numerical experiments that we will perform in this thesis, we will make use of MATLAB (Copyright 1984-2001 The MathWorks, Inc.), Version 6.1, Release 12.1. All experiments will be carried out on a UNIX machine with two 996 MHz Intel Pentium III processors and unit roundoff $u = 2^{-53} \approx 1.1 \times 10^{-16}$.

In some of the numerical experiments, the stability of the corresponding algorithm may be tested using the direct search routines `adsmax`, `mdsmax` and `nmsmax` from The Matrix Computation Toolbox of Higham [41] (documented in [44]).

Whenever more accurate results of an algorithm are required, we will use MATLAB's variable precision arithmetic function `vpa` to compute the results to 100 digit arithmetic using MATLAB's Symbolic Math Toolbox, and then round the results to double precision to obtain the result correct to machine precision.

1.10 Iterative Refinement

Iterative refinement is a technique for improving a computed solution $\widehat{x}$ to a linear system $Ax = b$, $A \in \mathbb{R}^{m\times m}$. There are three steps for computing the new improved solution:

Algorithm 1.10.1 *This algorithm carries out iterative refinement on the solution $\widehat{x}$ for a linear system $Ax = b$, $A \in \mathbb{R}^{m\times m}$, to obtain an improved solution x_{new}.*

Compute $r = b - A\widehat{x}$ (in precision $\overline{u}$).

Solve $Ad = r$ (in precision u).

Update $x_{\text{new}} = \widehat{x} + d$ (in precision u).

For traditional iterative refinement, $\overline{u} = u^2$. This process can be repeated if necessary with $\widehat{x}$ replaced by x_{new}. If there were no rounding errors in the computation of r, d and x_{new} then we can see that x_{new} would be the exact solution to the system. Thus, upon finding an approximate solution $\widehat{x}$, we could implement iterative refinement and obtain an improved solution to our problem. Algorithm 1.10.1 can be repeated until we obtain a solution that is as accurate as we want, or until iterative refinement does not improve the solution any more.

For more details on iterative refinement see Higham [42, Chapter 12], and in particular, see [42, Theorem 12.3] to see how iterative refinement can improve backward stability.

In Chapter 4 we will see how to implement iterative refinement to improve the solution we obtain for the ILS problem.

1.11 QR and Related Factorizations

In this section, we take a look at the QR factorization and other related factorizations which we will be using throughout the thesis. Of the factorizations mentioned here, the QR factorization will be studied in more detail than the others since we will be introducing more factorizations related to it in later chapters.

1.11.1 The QR Factorization

A QR factorization of $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ is a factorization

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = [\,Q_1 \quad Q_2\,] \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R, \tag{1.11.1}$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R \in \mathbb{R}^{n\times n}$ is upper triangular. Depending on the context, either the full factorization $A = Q[R^T\ 0]^T$ or the "economy size" version $A = Q_1R$ can be called the QR factorization. Furthermore, if $\operatorname{rank}(A) = n$ then the upper triangular matrix R is nonsingular.

We note that a similar factorization to that presented here does exist for the case $m < n$, but we will omit the details since we will not be using it.

The usual method used to compute the QR factorization is by applying Householder matrices to annihilate the subdiagonal elements of A a column at a time from left to right. The ith Householder matrix, P_i, used to annihilate the subdiagonal elements of the ith column of A is

$$P_i = \begin{bmatrix} I_{i-1} & 0 \\ 0 & \widetilde{P}_i \end{bmatrix}, \tag{1.11.2a}$$

$$\widetilde{P}_i = I_{m-i+1} - \beta_i v_i v_i^T, \qquad \beta_i = \frac{2}{v_i^T v_i}, \qquad v_i = x_i - \sigma_i e_1, \qquad \sigma_i = \pm \|x_i\|_2\,, \tag{1.11.2b}$$

where $\widetilde{P}_i \in \mathbb{R}^{(m-i+1)\times(m-i+1)}$, $v_i \in \mathbb{R}^{m-i+1}$ and the vector x_i in (1.11.2b) is the first column of the matrix that we postmultiply $\widetilde{P}_i$ by to perform the ith step of the QR factorization, i.e. $x_i = (P_{i-1}\dots P_1A)(i\colon m, i)$. Note that, when $i = 1$, the matrix (1.11.2a) just becomes $P_1 = \widetilde{P}_1$. Note also that the P_i and $\widetilde{P}_i$, $i = 1\colon n$, are both symmetric and orthogonal, and hence involutary.

The basic concepts and properties of Householder transformations, which made their debut in the book by Turnbull and Aitken [78, Chapter 8, Section 2] in 1932, are generally well known. In the following lemma, we give a complete breakdown of computing the QR factorization using Householder transforms, as first propounded by Householder [46] in 1958.

Lemma 1.11.1 *Let $A \in \mathbb{R}^{m\times n}$, where $m \geq n$. The QR factorization of A in* (1.11.1) *using the Householder method of reducing A to upper triangular form takes $2n^2(m - n/3)$ flops to compute R. If the explicit computation of Q or Q_1*

is required, then it takes a further $4n(m^2 - mn + n^2/3)$ *flops to compute* Q *or* $2n^2(m - n/3)$ *flops to compute* Q_1.

Proof. We first look at reducing A to $[R^T\ 0]^T$. Using the Householder method, we have

$$Q^T = P_n P_{n-1} \dots P_2 P_1,$$

where each Householder matrix, $P_i \in \mathbb{R}^{m\times m}$, is given by (1.11.2).

We remark here that v_i in (1.11.2b) only differs from x_1 in its first element. Thus, in computing v_i, we only need to compute $v_i(1) = x_i(1) - \sigma_i$. Most textbooks recommend using this formula with the sign chosen to avoid cancellation in the computation of $v_i(1)$:

$$\sigma_i = -\text{sign}\,(x_i(1))\,\|x\|_2\,, \qquad v_i(1) = x_i(1) - \sigma_i. \tag{1.11.3}$$

This approach is also used by the QR factorization routines in LINPACK [24] and LAPACK [4]. However, this does not mean that the other sign is unstable. In fact, as shown by Parlett [59], [60, Section 6.3.1] and Danloy [22], if we rearrange $v_i(1)$ so that

$$\sigma_i = \text{sign}\,(x_i(1))\,\|x\|_2\,, \tag{1.11.4a}$$

$$\begin{aligned} v_i(1) &= x_i(1) - \sigma_i \\ &= \frac{x_i(1)^2 - \|x\|_2^2}{x_i(1) + \sigma_i} \\ &= \frac{x_i(2\colon m-i+1)^T x_i(2\colon m-i+1)}{x_i(1) + \sigma_i}, \end{aligned} \tag{1.11.4b}$$

then the other sign is perfectly satisfactory. For both choices of sign it is easy to show that

$$\beta_i = \frac{2}{v_i^T v_i} = \frac{1}{\sigma_i v_i(1)}.$$

Suppose now that we have reached the kth stage of the QR factorization of A, so that we are in the process of premultiplying the current transformed A

Table 1.11.1: The number of flops required to compute the kth stage for reducing A to $[R^T\ 0]^T$ via the QR factorization.

Step of kth stage	Flops
Compute v_k using either (1.11.3) or (1.11.4)	$2(m-k)$
$w_k := \widetilde{A}_k^T v_k \in \mathbb{R}^{n-k}$	$2(m-k)(n-k)$
Compute β_k	2
$u_k := \beta_k v_k \in \mathbb{R}^{m-k+1}$	$m-k$
$M_k := u_k w_k^T \in \mathbb{R}^{(m-k+1)\times(n-k)}$	$(m-k)(n-k)$
Compute $\widetilde{A}_k - M_k$	$(m-k)(n-k)$
Total	$4(m-k)(n-k)$

by P_k. We denote by $A_k \in \mathbb{R}^{(m-k+1)\times(n-k+1)}$ the matrix that we are going to premultiply by $\widetilde{P}_k$, i.e. $A_k = (P_{k-1}\dots P_1 A)(i\colon m, i\colon n)$. Also, let us denote by $\widetilde{A}_k \in \mathbb{R}^{(m-k+1)\times(n-k)}$ the submatrix made up of all but the first column of A_k, i.e. $A_k = [x_k\ \widetilde{A}_k]$. When we premultiply $\widetilde{A}_k$ by $\widetilde{P}_k$, note that

$$\begin{aligned}\widetilde{P}_k\widetilde{A}_k &= \left(I_{m-k+1} - \beta_k v_k v_k^T\right)\widetilde{A}_k \\ &= \widetilde{A}_k - \beta_k v_k\left(v_k^T\widetilde{A}_k\right), \qquad (1.11.5)\end{aligned}$$

so we avoid performing a matrix-matrix multiply, which is $O(n)$ times more expensive than forming (1.11.5) since it only consists of a matrix-vector multiplication. Note that $\widetilde{P}_k x_k = \|x_k\|_2 e_1$.

We list the individual steps of the kth stage of reducing A to $[R^T\ 0]^T$ and show how many flops it takes to work out each step to order of highest magnitude in Table 1.11.1.

Thus, the total number of flops, $fl_{QR}(m,n)$, needed to reduce A to the upper trapezoidal matrix $[R^T\ 0]^T$ using the Householder QR factorization is

$$\begin{aligned} fl_{QR}\,(m,n) &= 4\sum_{k=1}^{n}(m-k)(n-k) \\ &= 4\sum_{k=1}^{n}(mn-(m+n)k+k^2) \\ &= 4mn-(m+n)n^2/2+n^3/3 \\ &= 2n^2(m-n/3), \end{aligned}$$

giving us the first result.

We note here that when constructing the Householder transformation $\widetilde{P}_k$, we only need to compute v_k, β_k and the product $u_k = \beta_k v_k$, since when we apply the transformation to a vector or a matrix, we do not need to explicitly form it as we saw in (1.11.5). Thus we can see from Table 1.11.1 that

$$\text{“Constructing” } \widetilde{P}_k \text{ takes } 3(m-k) \text{ flops.} \tag{1.11.6}$$

We also note that once we have computed v_k, β_k and u_k, suppose we wanted to apply the Householder transform $\widetilde{P}_k$ to a matrix $B \in \mathbb{R}^{m-k+1\times j}$ as in (1.11.5). Using Table 1.11.1 again, it is easy to see that

$$\text{Computing } \widetilde{P}_k B \text{ takes } 4j(m-k) \text{ flops.} \tag{1.11.7}$$

In certain circumstances, we may only need to compute the first s rows of the product $\widetilde{P}_k B$. We could then rewrite (1.11.5) (with $\widetilde{A}_k$ replaced by B) as

$$\begin{aligned} (\widetilde{P}_k B)(1\colon s,:) &= \left(\left(I_{m-k+1}-\beta_k v_k v_k^T\right)B\right)(1\colon s,:) \\ &= B(1\colon s,:)-\beta_k v_k(1\colon s)(v_k^T B). \end{aligned} \tag{1.11.8}$$

Hence, we can see that computing (1.11.8) takes $2j(m-k+s)$ flops.

We now show how to explicitly form Q. There are two ways in which we could form the product of the Householder matrices to compute Q. One way is to form $Q = P_1\,(P_2\,(\ldots(P_{n-1}P_n)\ldots))$, and the other way is $Q = ((\ldots(P_1P_2)\ldots)\,P_{n-1})\,P_n$.

From equation (1.11.2a), we can see that as i increases from 1 to n, P_i becomes more and more like the identity matrix, I_m, and it is only $P_i(i\colon m, i\colon m) = \widetilde{P}_i$ that is full. Therefore, it would make more sense to use the first approach of forming Q and thus, when forming the product $P_{i-1}P_i$, we are only effectively performing a matrix multiplication of two matrices of order $m - i + 2$. In other words, we use the following algorithm for explicitly computing Q [34, Section 5.2.1]:

Algorithm 1.11.2

```
$Q\colon= P_n$
for $i = n-1\colon -1\colon 1$
    $Q(\alpha,\alpha)\colon= P_i(\alpha,\alpha)\,Q(\alpha,\alpha), \qquad \alpha = i\colon m.$
end
```

We can also exploit the structure of a Householder matrix when we postmultiply it by another matrix as in (1.11.5).

Let us assume that we are at the kth pass of the loop in Algorithm 1.11.2. We list the individual steps of each pass in Table 1.11.2 and show how many flops it takes to work out each step to order of highest magnitude.

Thus, the total number of flops, $fl_Q\,(m,n)$, to explicitly compute Q in the QR factorization of $A \in \mathbb{R}^{m\times n}$ is

$$\begin{aligned} fl_Q\,(m,n) &= 4\sum_{k=1}^{n-1}(m-n+k)^2 \\ &= 4\sum_{k=1}^{n-1}((m-n)^2 + 2(m-n)k + k^2) \\ &= 4(n(m-n)^2 + 2(m-n)n^2/2 + n^3/3) \\ &= 4n(m^2 - mn + n^2/3), \end{aligned}$$

and hence giving us the second result.

Finally, if we only want to compute Q_1, i.e. only the first n columns of Q, then note that, when multiplying two $m \times m$ matrices, A, B say, of which we

Table 1.11.2: The number of flops required to compute the kth pass of the loop in Algorithm 1.11.2 for explicitly computing Q in the QR factorization of A.

Step of kth pass	Flops
$a_{n-k} := Q\,(n-k\colon m, n-k\colon m)^T\, v_{n-k} \in \mathbb{R}^{m-n+k+1}$	$2(m-n+k)^2$
$T_{n-k} := u_{n-k}a_{n-k}^T \in \mathbb{R}^{(m-n+k+1)\times(m-n+k+1)}$, where we have already computed $u_{n-k} = \beta_{n-k}v_{n-k}$	$(m-n+k)^2$
$Q\,(n-k\colon m, n-k\colon m) - T_{n-k}$	$(m-n+k)^2$
Total	$4(m-n+k)^2$

only want to know the first $n < m$ columns of the product, then it suffices to only compute the product $AB(:, 1\colon n)$. Also note that, since the Householder transforms P_i, $i = 1\colon n$, have the structure (1.11.2a), we are again, in effect, only postmultiplying $P_i(i\colon m, i\colon m)$ by $(P_{i+1} \ldots P_n)(i\colon m, i\colon n)$. Thus, the analogue of Algorithm 1.11.2 when computing Q_1 is

Algorithm 1.11.3

```
$Q_1 := P_n(:, 1\colon n)$
for $i = n-1\colon -1\colon 1$
    $Q_1(\alpha_1, \alpha_2) := P_i(\alpha_1, \alpha_1)\, Q_1(\alpha_1, \alpha_2)$,    $\alpha_1 = i\colon m$,    $\alpha_2 = i\colon n$.
end
```

Once again, we assume that we are at the kth pass of the loop in Algorithm 1.11.3. Table 1.11.3 lists the individual steps of each pass with their flop count.

Thus, the total number of flops, $fl_{Q_1}(m, n)$, to explicitly compute Q_1 is

$$fl_{Q_1}(m, n) = 4\sum_{k=1}^{n-1} k(m-n+k)$$

Table 1.11.3: The number of flops required to compute the kth pass of the loop in Algorithm 1.11.3 for explicitly computing Q_1 in the QR factorization of A.

Step of kth pass	Flops
$a_{n-k} := Q\left(n-k{:}\,m, n-k{:}\,n\right)^T v_{n-k} \in \mathbb{R}^{k+1}$	$2k(m-n+k)$
$T_{n-k} := u_{n-k}a_{n-k}^T \in \mathbb{R}^{(m-n+k+1)\times(k+1)}$	$k(m-n+k)$
$Q_1\left(n-k{:}\,m, n-k{:}\,n\right) - T_{n-k}$	$k(m-n+k)$
Total	$4k(m-n+k)$

$$
\begin{aligned}
&= 4\sum_{k=1}^{n-1}((m-n)k+k^2) \\
&= 4((m-n)n^2/2+n^3/3) \\
&= 2n^2(m-n/3),
\end{aligned}
$$

as required. $\square$

1.11.2 The QL Factorization

The QL factorization is similar in concept to that of the QR factorization. A QL factorization of $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ is a factorization

$$
A = Q\begin{bmatrix} 0 \\ L \end{bmatrix} = [\, Q_1 \quad Q_2 \,]\begin{bmatrix} 0 \\ L \end{bmatrix} = Q_2 L,
$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $L \in \mathbb{R}^{n\times n}$ is lower triangular. Note again that this factorization does exist for the case $m < n$. Also note that, trivially, this factorization has the same computational cost as the QR factorization.

1.11.3 The Generalized QR Factorization

The generalized QR factorization was introduced by Hammarling [36] in 1987 and Paige [54] in 1990 and further analyzed by Anderson, Bai and Dongarra [5] in 1992. Cox and Higham have used it in the solution of the LSE problem [20]. See also Cox [19] and Higham [42, Section 20.9] for more details. The following result can be found in [20, Theorem 2.1], [42, Theorem 20.9].

Lemma 1.11.4 (Generalized QR factorization) *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{s\times n}$ with $m + s \geq n \geq s$ and assume that*

$$\operatorname{rank}(B) = s, \qquad \mathcal{N}(A) \cap \mathcal{N}(B) = \{0\}. \tag{1.11.9}$$

There are orthogonal matrices $Q \in \mathbb{R}^{n\times n}$ and $U \in \mathbb{R}^{m\times m}$ such that

$$U^T AQ = \begin{array}{c} \\ {\scriptstyle m-n+s} \\ {\scriptstyle n-s} \end{array}\!\!\begin{array}{c} \begin{array}{cc} {\scriptstyle s} & {\scriptstyle n-s} \end{array} \\ \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \end{array}, \qquad BQ = \begin{array}{c} \\ {\scriptstyle s} \end{array}\!\begin{array}{c} \begin{array}{cc} {\scriptstyle s} & {\scriptstyle n-s} \end{array} \\ [\,S \quad 0\,] \end{array},$$

where L_{22} and S are lower triangular. More precisely, we have

$$U^T AQ = \begin{cases} \begin{array}{c} \\ {\scriptstyle m-n} \\ {\scriptstyle n} \end{array}\!\begin{array}{c} {\scriptstyle n} \\ \begin{bmatrix} 0 \\ L \end{bmatrix} \end{array}, & m \geq n, \\ \begin{array}{c} \\ {\scriptstyle m} \end{array}\!\begin{array}{c} \begin{array}{cc} {\scriptstyle n-m} & {\scriptstyle m} \end{array} \\ [\,X \quad L\,] \end{array}, & m < n, \end{cases}$$

where L is lower triangular and X is a generic matrix. The assumptions in (1.11.9) *are equivalent to S and L_{22} being nonsingular.* □

1.11.4 Stability Results Relating to the QR Factorization

In this section, we will state the results that we will be making use of throughout this document that relate to stability analysis.

The first result, from Higham [42, Theorem 19.2], gives us the error in computing the product of a Householder transformation with a vector.

Lemma 1.11.5 *Let $a \in \mathbb{R}^m$ an consider the computation of $y = \widehat{P}a$, where $\widehat{P}$ is the computed Householder transformation P constructed as described in* (1.11.2) *and applied as described in* (1.11.5). *The computed $\widehat{y}$ satisfies*

$$\widehat{y} = (P + \Delta P)b, \qquad \|\Delta P\|_F \le \widetilde{\gamma}_m. \qquad \square$$

The next result, from Higham [42, Lemma 19.3], gives us the error in applying a product of Householder transformations to a matrix $A \in \mathbb{R}^{m\times n}$.

Lemma 1.11.6 *Consider the sequence of transformations*

$$A_{k+1} = P_k A_k, \qquad k = 1{:}\, r,$$

where $A_1 = A \in \mathbb{R}^{m\times n}$ and $P_k = I - v_k v_k^T \in \mathbb{R}^{m\times m}$ is a Householder matrix. Assume that the transformations are performed using computed Householder vectors $\widehat{v}_k \approx v_k$ that satisfy $\widehat{v} = v + \Delta v$, where $|\Delta v| \le \widetilde{\gamma}_m |v|$ ($v \in \mathbb{R}^m$, $\|v\|_2 = \sqrt{2}$). The computed matrix $\widehat{A}_{r+1}$ satisfies

$$\widehat{A}_{r+1} = Q^T (A + \Delta A),$$

where $Q^T = P_r P_{r-1} \dots P_1$ and

$$\|\Delta a_j\|_2 \le r\widetilde{\gamma}_m \|a_j\|_2, \qquad j = 1{:}\, n.$$

In the special case $n = 1$, so that $A \equiv a$, we have $\widehat{a}^{r+1} = (Q + \Delta Q)^T a$ with $\|\Delta Q\|_F \le r\widetilde{\gamma}_m$. $\square$

The error in computing the upper trapezoidal R-factor in the QR factorization of a matrix is given by the following result from Higham [42, Theorem 19.4].

Lemma 1.11.7 *Let $\widehat{R} \in \mathbb{R}^{m\times n}$ be the computed upper trapezoidal R-factor of $A = [a_1\ a_2\ \cdots\ a_n] \in \mathbb{R}^{m\times n}$, ($m \ge n$) obtained via the Householder QR factorization. Then there exists an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ such that*

$$A + \Delta A = Q\widehat{R},$$

where

$$\|\Delta a_j\|_2 \le \widetilde{\gamma}_{mn} \|a_j\|_2, \qquad j = 1\colon n.$$

The matrix Q is given explicitly as $Q = (P_n P_{n-1} \dots P_1)^T$, where P_k is the Householder matrix that corresponds to annihilating the subdiagonal elements in the kth column of A. □

The following result by Cox and Higham [20, Lemma 3.3] will be used later in the thesis.

Lemma 1.11.8 *Let $s = b - Ax + e$, for any conformable matrix A and vectors s, b, x, e, where $\|e\|_2 \le \epsilon \|b - Ax\|_2$. Then $s = b + \Delta b - (A + \Delta A)x$, where $\|\Delta b\|_2 \le \epsilon \|b\|_2$ and $\|\Delta A\|_2 \le \epsilon \|A\|_2$.* □

1.12 Miscellaneous Matrix Factorizations

In this section, we state some additional matrix factorizations that we will need to refer to later on in the thesis.

1.12.1 The Cholesky Factorization

A Cholesky factorization of a symmetric positive definite matrix $A \in \mathbb{R}^{m\times m}$ is a factorization

$$A = R^T R,$$

where R is upper triangular. For an algorithm for computing R, see Higham [42, Theorem 10.2]. It takes $m^3/3$ flops to compute R.

1.12.2 Forward and Backward Substitution

Suppose we wish to solve the triangular system of equations

$$Ax = b$$

using forward substitution (see [34, Section 3.1.1]) when $A \in \mathbb{R}^{m\times m}$ is lower triangular or using backward substitution (see [34, Section 3.1.2]) when $A \in \mathbb{R}^{m\times m}$ is upper triangular, and $x, b \in \mathbb{R}^m$. We then require precisely m^2 flops to compute x.

The error in computing a solution of a triangular system of equations, either by backward or forward substitution, is given by the following result by Higham [42, Lemma 8.5].

Lemma 1.12.1 *Let the triangular system $Tx = b$, where $T \in \mathbb{R}^{m\times m}$ is nonsingular, be solves by substitution, with any ordering. Then the computed solution $\widehat{x}$ satisfies*

$$(T + \varDelta T)\widehat{x} = b, \qquad |\varDelta T| \le \gamma_m |T|. \qquad \square$$

1.12.3 The LU Factorization, Gaussian Elimination and the Block LU Factorization

A LU factorization of a matrix $A \in \mathbb{R}^{m\times m}$ is a factorization

$$A = LU \tag{1.12.1}$$

where L is unit lower triangular and U is upper triangular:

$$L = \begin{bmatrix} 1 & & & \\ l_{2,1} & 1 & & \\ \vdots & & \ddots & \\ l_{m,1} & \cdots & l_{m,m-1} & 1 \end{bmatrix}, \qquad U = \begin{bmatrix} u_{1,1} & u_{1,2} & \cdots & u_{1,m} \\ & u_{2,2} & & \vdots \\ & & \ddots & u_{m-1,m} \\ & & & u_{m,m} \end{bmatrix}.$$

For certain applications, we may need to rescale the factors L and U. We would then refer to this factorization, quite simply, as the rescaled LU factorization.

A popular method to solve the linear system $Ax = b$, where $A \in \mathbb{R}^{m\times m}$ is nonsingular, is to reduce A to upper triangular form by applying elementary row operations to it, to obtain U, and the vector b, to obtain b', and solving the new system $Ux = b'$. This method of solving the system is called Gaussian elimination and is, in effect, the same as forming the factor U in (1.12.1) and leaving L in factored form to apply to b. The whole process of Gaussian elimination to solve $Ax = b$ takes $2n^3/3$ flops. For more on the LU factorization, including conditions for existence of and an algorithm on how to compute the factorization, see Higham [42, Chapter 9].

An extension to the LU factorization is the block LU factorization $A = LU \in \mathbb{R}^{m\times m}$, where L and U are block triangular and L has identity matrices on the diagonal. The blocks can be of different dimensions. For more on the block LU factorization see Higham [42, Chapter 13].

1.12.4 The CS Decomposition

The CS decomposition is defined as follows. Let an orthogonal matrix $A \in \mathbb{R}^{m\times m}$ be partitioned so that

$$A = \begin{array}{c c} & \begin{array}{cc} p & q \end{array} \\ \begin{array}{c} p \\ q \end{array} & \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \end{array}, \qquad p + q = m, \tag{1.12.2}$$

and assume that $q \geq p$. Then there are orthogonal matrices $U_1, V_1 \in \mathbb{R}^{p\times p}$ and $U_2, V_2 \in \mathbb{R}^{q\times q}$ such that

$$\begin{bmatrix} U_1^T & 0 \\ 0 & U_2^T \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix} = \begin{array}{c c} \begin{array}{ccc} p & p & q-p \end{array} & \\ \begin{bmatrix} C & S & 0 \\ -S & C & 0 \\ 0 & 0 & I_{q-p} \end{bmatrix} & \begin{array}{c} p \\ p \\ q-p \end{array} \end{array},$$

where $C = \operatorname{diag}(c_i)$, $S = \operatorname{diag}(s_i)$, C is nonsingular and $C^2 + S^2 = I$. For more details, see, e.g., Stewart [72, p. 5] or Paige and Wei [55].

1.12.5 The Hyperbolic CS Decomposition

The hyperbolic CS decomposition is defined as follows. Let $A \in \mathbb{R}^{m\times m}$ be partitioned as in (1.12.2) and let it satisfy $AJA^T = J$, where the signature matrix

$$J = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}, \qquad p + q = m.$$

We also assume that $q \geq p$. Then there are orthogonal matrices $U_1, V_1 \in \mathbb{R}^{p\times p}$ and $U_2, V_2 \in \mathbb{R}^{q\times q}$ such that

$$\begin{bmatrix} U_1^T & 0 \\ 0 & U_2^T \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix} = \begin{array}{c@{}c} \begin{array}{ccc} \scriptstyle p & \scriptstyle p & \scriptstyle q-p \end{array} & \\ \left[\begin{array}{ccc} C & -S & 0 \\ -S & C & 0 \\ 0 & 0 & I_{q-p} \end{array}\right] & \begin{array}{l} \scriptstyle p \\ \scriptstyle p \\ \scriptstyle q-p \end{array} \end{array},$$

where $C = \operatorname{diag}(c_i)$, $S = \operatorname{diag}(s_i)$ and $C^2 - S^2 = I$. For a proof of its existence, see Higham [43, Theorem 3.2].

Chapter 2

J-Orthogonal Transformations

J-orthogonal transformations play a major role in the solution of the ILS problem. In this chapter, we look at J-orthogonal matrices and their properties, and show a link between J-orthogonal and orthogonal matrices. We then introduce hyperbolic rotations. There are many ways to form and apply the rotations, and each representation leads to varying accuracy in the output. We analyze stability of forming and applying single rotations and applying products of rotations. We also show a way to stably compute the hyperbolic QR factorization using them.

2.1 J-Orthogonal Matrices

2.1.1 Introduction and Properties

A matrix $Q \in \mathbb{R}^{m\times m}$ is *J-orthogonal* if

$$QJQ^T = J, \tag{2.1.1}$$

where the signature matrix

$$J = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}, \qquad p + q = m. \tag{2.1.2}$$

Q is also referred to as *J-unitary* [15], [16], [79], *hypernormal* [10], [49], [53], [62], [63] or *pseudo-orthogonal* [1], [23], [34, p. 612]. Note that if $p = m$ then Q is orthogonal.

Lemma 2.1.1 *A J-orthogonal matrix Q is nonsingular. Moreover,* $\det(Q) = \pm 1$.

Proof. We take the determinant on both sides of (2.1.1), which gives

$$\pm 1 = \det(J) = \det(Q)\det(J)\det(Q^T) = \pm\det(Q)^2,$$

So $\det(Q) = \pm 1$ and hence Q is nonsingular. $\Box$

Lemma 2.1.2 *A J-orthogonal matrix Q satisfies*

$$Q^T JQ = QJQ^T.$$

Proof. From equation (2.1.1) we see that $Q = JQ^{-T}J$ so

$$Q^T JQ = Q^T JJQ^{-T}J = J = QJQ^T. \quad \Box$$

In the above lemmas, note that setting $p = m$, so that $J = I_m$, gives us the corresponding results for orthogonal matrices.

We define the *hyperbolic energy* of a vector x with respect to the signature matrix J to be $x^T Jx$. It is also referred to as the *J-inner product* of x [79] and the *indefinite scalar product* [9], [30]. It is clear that premultiplication by J-orthogonal matrices preserves the hyperbolic energy of a vector. In other words, for $x, y \in \mathbb{R}^m$ and a J-orthogonal matrix $Q \in \mathbb{R}^{m\times m}$, if $Qx = y$ then

$$y^T Jy = (Qx)^T J(Qx) = x^T(Q^T JQ)x = x^T Jx.$$

Note that if $Q_1 \in \mathbb{R}^{p\times p}$ and $Q_2 \in \mathbb{R}^{q\times q}$ are orthogonal matrices, then the matrix

$$Q = \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \in \mathbb{R}^{m\times m}$$

is both orthogonal and J-orthogonal, where J is as defined in (2.1.2).

There are many applications in which J-orthogonal transformations are used. These range from solving many signal processing problems (see e.g. [62], [63]), downdating the Cholesky factorization [11] and algorithms for factoring structured matrices [21] to block updating and fast algorithms for Toeplitz-like matrices [74].

2.1.2 A Link Between Orthogonal and J-Orthogonal Matrices

We establish a relation between orthogonal and J-orthogonal transformations.

Let $Q \in \mathbb{R}^{m \times m}$ be an orthogonal matrix, and consider the partitioned linear system

$$Qx = \begin{array}{c c} & \begin{array}{cc} p & q \end{array} \\ \begin{array}{c} p \\ q \end{array} & \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \end{array} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = y. \tag{2.1.3}$$

If Q_{11} is nonsingular, then the first equation of (2.1.3), $Q_{11}x_1 + Q_{12}x_2 = y_1$, may be solved for x_1:

$$x_1 = Q_{11}^{-1}y_1 - Q_{11}^{-1}Q_{12}x_2.$$

If we substitute this equation into the second equation of (2.1.3), we obtain

$$\begin{aligned} y_2 &= Q_{21}(Q_{11}^{-1}y_1 - Q_{11}^{-1}Q_{12}x_2) + Q_{22}x_2 \\ &= Q_{21}Q_{11}^{-1}y_1 + (Q_{22} - Q_{21}Q_{11}^{-1}Q_{12})x_2. \end{aligned}$$

Thus, we have the system

$$\operatorname{exc}(Q) \begin{bmatrix} y_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_2 \end{bmatrix}, \tag{2.1.4}$$

where the exchange operator $\operatorname{exc}(\,\cdot\,)$ is defined by

$$\operatorname{exc}(Q) = \begin{bmatrix} Q_{11}^{-1} & -Q_{11}^{-1}Q_{12} \\ Q_{21}Q_{11}^{-1} & Q_{22} - Q_{21}Q_{11}^{-1}Q_{12} \end{bmatrix} \tag{2.1.5}$$

and is so named because it represents an exchange of x_1 and y_1, i.e. $\mathrm{exc}\,(Q)$ is applied to the mixed vector $[y_1^T\ x_2^T]^T$ to obtain $[x_1^T\ y_2^T]^T$. Note that the $(2,2)$ block of $\mathrm{exc}\,(Q)$ is the Schur complement of Q.

Conversely, let Q be a J-orthogonal matrix and consider again the system (2.1.3). We can obtain (2.1.4) using the same procedure as for the case of orthogonal Q. However, since the $(1,1)$ block of the equation $Q^TJQ = J$ leads to

$$Q_{11}^TQ_{11} = I_p + Q_{21}^TQ_{21},$$

which is the sum of a positive definite and a positive semidefinite matrix, we need no longer assume that Q_{11} is nonsingular, as this is automatically the case.

Once again, consider partitioning the matrix Q as in (2.1.3). Using (2.1.5), we have

$$\begin{aligned}\mathrm{exc}\,(\mathrm{exc}\,(Q)) &= \mathrm{exc}\left(\begin{bmatrix} Q_{11}^{-1} & -Q_{11}^{-1}Q_{12} \\ Q_{21}Q_{11}^{-1} & Q_{22}-Q_{21}Q_{11}^{-1}Q_{12}\end{bmatrix}\right)\\ &= \begin{bmatrix} Q_{11} & Q_{11}Q_{11}^{-1}Q_{12} \\ Q_{21}Q_{11}^{-1}Q_{11} & Q_{22}-Q_{21}Q_{11}^{-1}Q_{12}+Q_{21}Q_{11}^{-1}Q_{11}Q_{11}^{-1}Q_{12}\end{bmatrix}\\ &= Q,\end{aligned}$$

thus the exchange operator is involutary.

The next result of Higham [43] addresses the nonsingularity of the exchange operator and will be used in establishing a relation between orthogonal and J-orthogonal transformations. We note that the block LU factorization, which we introduced in Section 1.12.3, of $\mathrm{exc}\,(Q)$ is given by

$$\begin{aligned}\mathrm{exc}\,(Q) &= \begin{bmatrix} I_p & 0 \\ Q_{21} & Q_{22}\end{bmatrix}\begin{bmatrix} Q_{11}^{-1} & -Q_{11}^{-1}Q_{12} \\ 0 & I_q\end{bmatrix}\\ &= \begin{bmatrix} I_p & 0 \\ Q_{21} & Q_{22}\end{bmatrix}\begin{bmatrix} Q_{11} & Q_{12} \\ 0 & I_q\end{bmatrix}^{-1}.\end{aligned}\qquad (2.1.6)$$

Lemma 2.1.3 *Let $Q \in \mathbb{R}^{m\times m}$ with Q_{11} nonsingular. If Q is J-orthogonal or orthogonal then* $\mathrm{exc}\,(Q)$ *is nonsingular. More generally,* $\mathrm{exc}\,(Q)$ *is nonsingular* *if and only if* Q_{22} *is nonsingular. If* $\operatorname{exc}(Q)$ *and* $\operatorname{exc}(Q^{-1})$ *are both nonsingular then* $\operatorname{exc}(Q)^{-1} = \operatorname{exc}(Q^{-1})$.

Proof. For $Q \in \mathbb{R}^{m\times m}$ with Q_{11} nonsingular, the block LU factorization of $\operatorname{exc}(Q)$ in (2.1.6) makes it clear that $\operatorname{exc}(Q)$ is nonsingular if and only if Q_{22} is nonsingular. If Q is J-orthogonal then it is clear from its hyperbolic CS decomposition, introduced in Section 1.12.5, that Q_{11} nonsingular implies Q_{22} nonsingular; the same implication holds for orthogonal Q by the standard CS decomposition, introduced in Section 1.12.4.

The last part is obtained by rewriting (2.1.3) as

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} M_1^{-1} & -Q_{11}^{-1}Q_{12}M_2^{-1} \\ -Q_{22}^{-1}Q_{21}M_1^{-1} & M_2^{-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = Q^{-1}y,$$

where $M_1 = Q_{11} - Q_{12}Q_{22}^{-1}Q_{21}$ and $M_2 = Q_{22} - Q_{21}Q_{11}^{-1}Q_{12}$, and deriving the corresponding analogue of (2.1.4):

$$\operatorname{exc}\left(Q^{-1}\right) \begin{bmatrix} x_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_1 \\ x_2 \end{bmatrix}. \tag{2.1.7}$$

Since (2.1.4) and (2.1.7) hold for unique x_1 and y_1, we must have $\operatorname{exc}(Q)^{-1} = \operatorname{exc}(Q^{-1})$. $\square$

We now state a result of Stewart and Stewart [74] which gives a correspondence between orthogonal and J-orthogonal matrices.

Theorem 2.1.4 *Let* $Q \in \mathbb{R}^{m\times m}$ *and let* Q_{11} *be the leading principal submatrix of order* p. *If* Q *is orthogonal with* Q_{11} *nonsingular, then* $\operatorname{exc}(Q)$ *is* J*-orthogonal. Conversely, if* Q *is* J*-orthogonal, then* $\operatorname{exc}(Q)$ *is orthogonal.*

Proof. Assume first that Q is orthogonal and Q_{11} is nonsingular. Equation (2.1.3) and the orthogonality of Q gives us

$$\|x_1\|_2^2 + \|x_2\|_2^2 = \|y_1\|_2^2 + \|y_2\|_2^2 .$$

Thus,

$$\|y_1\|_2^2 - \|x_2\|_2^2 = \|x_1\|_2^2 - \|y_2\|_2^2 .$$

This means that, using (2.1.4), the vector $z = [y_1^T \; x_2^T]^T$ satisfies

$$z^T J z = z^T \operatorname{exc}(Q)^T J \operatorname{exc}(Q)\, z. \tag{2.1.8}$$

Using Lemma 2.1.3, for any vector $[y_1^T \; x_2^T]^T$ there is a unique vector $[x_1^T \; y_2^T]^T$. Thus, (2.1.8) holds identically in z. Since both J and $\operatorname{exc}(Q)^T J \operatorname{exc}(Q)$ are symmetric, by the uniqueness of quadratic forms, this implies that $\operatorname{exc}(Q)$ is J-orthogonal.

The converse follows in an entirely analogous way. $\square$

It follows that there is a one-to-one correspondence between orthogonal matrices with a nonsingular $(1, 1)$ block and J-orthogonal matrices. This has advantages when performing error analysis. If Q is J-orthogonal, then forming it and applying it to the vector $x = [x_1 \; x_2]^T$ to obtain $y = [y_1 \; y_2]^T$ can be seen as forming and applying the matrix $\operatorname{exc}(Q)$ to the mixed vector $[y_1 \; x_2]^T$ to obtain $[x_1 \; y_2]^T$. If forming and applying $\operatorname{exc}(Q)$ is backward stable, say

$$\begin{bmatrix} x_1 \\ y_2 \end{bmatrix} = \operatorname{exc}(Q) \begin{bmatrix} y_1 + \Delta y_1 \\ x_2 + \Delta x_2 \end{bmatrix}, \tag{2.1.9a}$$

then we can immediately see that

$$\begin{bmatrix} y_1 + \Delta y_1 \\ y_2 \end{bmatrix} = Q \begin{bmatrix} x_1 \\ x_2 + \Delta x_2 \end{bmatrix}, \tag{2.1.9b}$$

and thus forming and applying Q is mixed forward backward stable. Similarly, we can say the same for a matrix Q such that forming and applying $\operatorname{exc}(Q)$ is forward stable. If $P = \operatorname{exc}(Q)$ in (2.1.9), then we could prove that forming and applying P was backward stable.

For more details on the exchange operator see [43], [56], [74].

2.2 Hyperbolic Rotations

Hyperbolic rotations are the J-orthogonal counterpart of the orthogonal Givens rotations. Hyperbolic rotations were first studied by Golub [33] in 1969, who derived them in terms of complex sines and cosines. In this section, we look at properties of hyperbolic rotations and their uses. Hyperbolic rotations are also referred to as *hyperbolic Givens rotations* [49], [53].

2.2.1 Introduction, Properties and Applications

A 2×2 *hyperbolic rotation* has the form

$$H = \begin{bmatrix} c & -s \\ -s & c \end{bmatrix}, \qquad c^2 - s^2 = 1, \tag{2.2.1}$$

where $|c| = \cosh\theta$ and $s = \sinh\theta$ for some θ. Clearly, H is symmetric, unlike the orthogonal *circular* Givens rotations. Also, it is easy to see that the matrix H in (2.2.1) is J-orthogonal for $J = \operatorname{diag}(1, -1)$ and for $J = \operatorname{diag}(-1, 1)$.

We want to choose H so that $Hx = y$, where $x = [x_1\ x_2]^T$ and $y = [y_1\ 0]^T$. Since $|c| > |s|$, we are forced to have $|x_1| > |x_2|$ in order for us to obtain a real solution. When this is the case, we have

$$c = \frac{x_1}{\sqrt{x_1^2 - x_2^2}}, \qquad s = \frac{x_2}{\sqrt{x_1^2 - x_2^2}}. \tag{2.2.2}$$

Figure 2.2.1 shows how a hyperbolic rotation differs from a Givens rotation

$$G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

where $c = \cos\theta$, $s = \sin\theta$ for some θ. The application of a Givens rotation to a vector $x = [x_1\ x_2]^T \in \mathbb{R}^2$ rotates it through θ radians clockwise in two-dimensional *Euclidean space*, i.e., along the circle with equation

$$[\,x_1 \quad x_2\,] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = k_1,$$

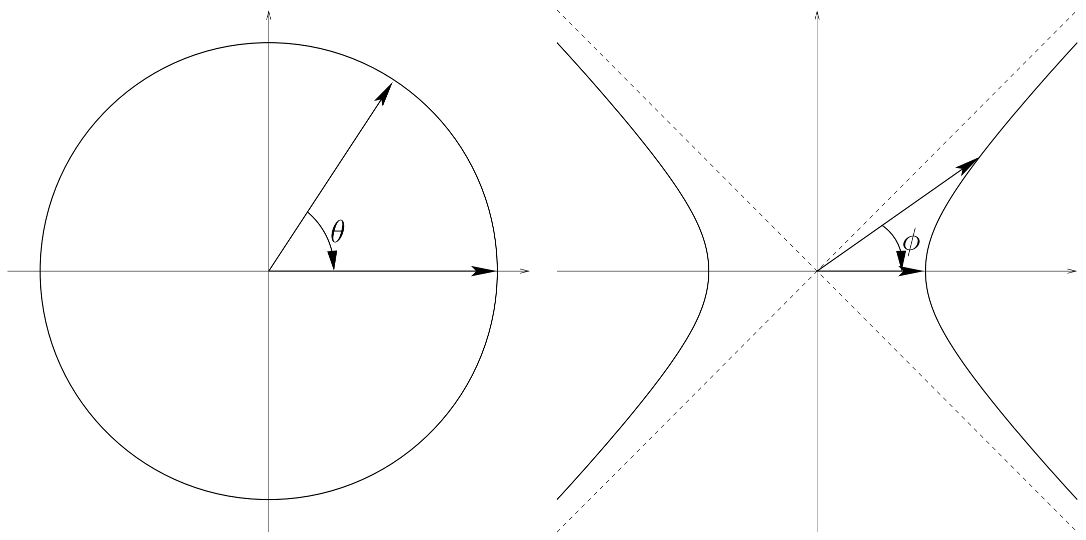


Figure 2.2.1: The difference between hyperbolic and Givens rotations. The left-hand diagram illustrates how a Givens rotation transforms a vector and the right-hand diagram illustrates how a hyperbolic rotation transforms a vector.

for some constant k_1. By contrast, a hyperbolic rotation applied to the same vector rotates it clockwise through $\phi = \tan^{-1}(\tanh\theta)$ in what is called two-dimensional *Minkowski space* [39, Section 5.3.1], i.e. along the hyperbola with equation

$$[\,x_1 \quad x_2\,]\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} = k_2,$$

for some constant k_2. The condition $|x_1| > |x_2|$ is made clearer in the second diagram of Figure 2.2.1, where it is easy to see that if $|x_1| \le |x_2|$, then it is impossible to *hyperbolically* rotate the vector $[x_1\ x_2]^T$ to lie along the x_1-axis since the hyperbola does not take real values in this region. It is also easy to see that $y_1 = \sqrt{x_1^2 - x_2^2}$ and, since H is J-orthogonal, both x and y have the same hyperbolic energy, that is $x^TJx = y^TJy$. Note that $\sqrt{x_1^2 - x_2^2}$ can be arbitrarily small, and hence the elements of H in (2.2.1) can be arbitrarily large. This will be a crucial issue in our error analysis later.

Since H is J-orthogonal, we can easily verify using (2.1.1) that the inverse of the hyperbolic rotation in (2.2.1)

$$H^{-1} = \begin{bmatrix} c & s \\ s & c \end{bmatrix}, \tag{2.2.3}$$

where c and s are given in (2.2.2). Using (2.2.1) and (2.2.3), we can also verify that the 2-norm condition number of H

$$\kappa_2(H) = \|H\|_2 \|H^{-1}\|_2 = (c+s)^2. \tag{2.2.4}$$

Note that the inverse of H is unbounded in norm and the condition number of H is also unbounded. This observation will be useful later on when we look at the error analysis of forming and applying products of hyperbolic rotations.

Hyperbolic rotations can be applied to larger matrices as well. When applied to a matrix $A \in \mathbb{R}^{m \times n}$ to annihilate an element in the jth row using an element in the ith row, say, we use an $m \times m$ hyperbolic rotation, which differs from the $m \times m$ identity matrix only in the submatrix

$$\text{HR1}: \qquad H([i,j],[i,j]) = \frac{1}{\sqrt{x_1^2 - x_2^2}} \begin{bmatrix} x_1 & -x_2 \\ -x_2 & x_1 \end{bmatrix}. \tag{2.2.5}$$

We write this particular H as $H_{i,j}$. Note that in (2.2.5) we have combined (2.2.1) and (2.2.2) by eliminating c and s. Also, if $J = \text{diag}(I_p, -I_q)$, then either $1 \leq i \leq p$ and $p > j \geq m$ or $1 \leq j \leq p$ and $p > i \geq m$ and if $J = \text{diag}(-I_q, I_p)$, then either $1 \leq j \leq p$ and $p > i \geq m$ or $1 \leq i \leq p$ and $p > j \geq m$.

There are numerous applications in which hyperbolic rotations are used. These include downdating LS problems (see e.g. [1], [47], [57]), stabilizing the generalized Schur algorithm (see e.g. [15], [16], [76]) and other similar algorithms to compute low-rank approximants of matrices [79], accurately computing singular values of long products of matrices [68], various signal processing applications [53], solving the basic deconvolution problem [52], fast algorithms for QR factorizations of structured matrices [18] and solving a linear system of equations, $Ax = b$, with A symmetric positive definite [23].

2.2.2 Representations of Hyperbolic Rotations

A hyperbolic rotation, H, can be represented in various forms. The reason for considering more than one representation is because, unlike for orthogonal rotations, how hyperbolic rotations are applied to a vector is crucial to the stability of the computation [8], [70]. We have seen HR1, the representation in (2.2.5). We look at a few more representations below, and discuss their stability in Section 2.2.3.

If we replace the $\sqrt{x_1^2 - x_2^2}$ term in (2.2.5) by $\sqrt{(x_1+x_2)(x_1-x_2)}$, then we obtain

$$\text{HR2}: \qquad H = \frac{1}{\sqrt{(x_1+x_2)(x_1-x_2)}} \begin{bmatrix} x_1 & -x_2 \\ -x_2 & x_1 \end{bmatrix}. \tag{2.2.6}$$

Substituting $t = x_2/x_1$ into (2.2.5) and (2.2.6) above gives us a further two representations of H. These are

$$\text{HR3}: \qquad H = \frac{1}{\sqrt{1-t^2}} \begin{bmatrix} 1 & -t \\ -t & 1 \end{bmatrix} \tag{2.2.7}$$

and

$$\text{HR4}: \qquad H = \frac{1}{\sqrt{(1+t)(1-t)}} \begin{bmatrix} 1 & -t \\ -t & 1 \end{bmatrix}. \tag{2.2.8}$$

There is another representation of a hyperbolic rotation that looks quite different from the ones discussed above. It is

$$\text{HR5}: \quad H = QDQ^T = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} d/2 & 0 \\ 0 & 1/(2d) \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \quad d = \sqrt{\frac{x_1+x_2}{x_1-x_2}}. \tag{2.2.9}$$

This is the mixed forward-backward stable *OD procedure* of Chandrasekaran and Sayed [15], [16], [17]. For a proof of its mixed forward-backward stability, see Lemma 2.2.6. Note that taking a factor of $1/2$ out of D and instead multiplying Q by $1/\sqrt{2}$ gives us the singular value decomposition of H. Hence, d and $1/d$ in (2.2.9) are in fact the singular values of H. Also, the $m \times m$ generalizations of Q

and D in (2.2.9) are just the $m \times m$ identity matrices with

$$Q([i,j],[i,j]) = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}, \qquad D([i,j],[i,j]) = \begin{bmatrix} d/2 & 0 \\ 0 & 1/(2d) \end{bmatrix}. \tag{2.2.10}$$

One further representation of a hyperbolic rotation is given by the exchange operator, which was defined in Section 2.1.2. Consider the computation of $b = Ha$, $a = [a_1\ a_2]^T$, $b = [b_1\ b_2]^T$, where H is as defined in (2.2.1):

$$\begin{aligned} b_1 &= ca_1 - sa_2, \qquad (2.2.11)\\ b_2 &= -sa_1 + ca_2. \end{aligned}$$

The first equation gives

$$a_1 = \frac{b_1}{c} + \frac{s}{c}a_2, \tag{2.2.12}$$

which allows the second to be rewritten as

$$\begin{aligned} b_2 &= -\frac{s}{c}b_1 + \left(-\frac{s^2}{c} + c\right)a_2 \\ &= -\frac{s}{c}b_1 + \frac{a_2}{c}. \qquad (2.2.13) \end{aligned}$$

As noted by Park and Eldén [58], this way of forming the product $b = Ha$ corresponds to the use of the rescaled LU factorization (see Section 1.12.3)

$$\text{HR6}: \qquad H = \begin{bmatrix} 1 & 0 \\ -s/c & 1/c \end{bmatrix} \begin{bmatrix} c & -s \\ 0 & 1 \end{bmatrix}, \tag{2.2.14}$$

where c and s are given in (2.2.2). That this way of forming b is advantageous for stability was proved by Bojanczyk, Brent, van Dooren and de Hoog [8] in the context of downdating a Cholesky factorization.

For later use we note that (2.2.12) and (2.2.13) can be expressed together in the form

$$\begin{bmatrix} a_1 \\ b_2 \end{bmatrix} = G \begin{bmatrix} b_1 \\ a_2 \end{bmatrix}, \tag{2.2.15}$$

where

$$G = \operatorname{exc}(H) = \begin{bmatrix} 1/c & s/c \\ -s/c & 1/c \end{bmatrix} \equiv \begin{bmatrix} \widetilde{c} & \widetilde{s} \\ -\widetilde{s} & \widetilde{c} \end{bmatrix}, \qquad \widetilde{c}^2 + \widetilde{s}^2 = 1.$$

The matrix G is a Givens rotation. Hence HR6 (2.2.14) can be interpreted as forming the first row of the product Ha by a hyperbolic rotation and the second row by a Givens rotation.

2.2.3 Error Analysis

When we premultiply a matrix A by a hyperbolic rotation H we change the entries in just two rows of A. If

$$A\left([i,j]\,,:\right) = \begin{bmatrix} a_{i1} & a_{i2} & \cdots & a_{in} \\ a_{j1} & a_{j2} & \cdots & a_{jn} \end{bmatrix}$$

and we premultiply A by the hyperbolic rotation HR1 in (2.2.5), say, then we have

$$(HA)\left([i,j]\,,:\right) = \begin{bmatrix} b_{i1} & b_{i2} & \cdots & b_{in} \\ b_{j1} & b_{j2} & \cdots & b_{jn} \end{bmatrix},$$

where

$$b_{ik} = \frac{x_1 a_{ik} - x_2 a_{jk}}{\sqrt{x_1^2 - x_2^2}}, \qquad b_{jk} = \frac{-x_2 a_{ik} + x_1 a_{jk}}{\sqrt{x_1^2 - x_2^2}}, \qquad k = 1\colon n.$$

All other elements of A are left unaffected.

We first perform an error analysis of HR1–HR4 in (2.2.5)–(2.2.8), since these are all formed in similar ways. The main difference in these representations is the factor by which all the elements of the hyperbolic rotation are divided. In HR1 this factor is $\sqrt{x_1^2 - x_2^2}$, in HR2 it is $\sqrt{(x_1 + x_2)(x_1 - x_2)}$, in HR3 it is $\sqrt{1 - t^2}$ and in HR4 it is $\sqrt{(1 + t)(1 - t)}$, where $t = x_2/x_1$. The accuracy with which we can compute these factors is the key to deciding which of HR1–HR4 we should use. This is obvious from the following result, where we calculate bounds for the errors in forming the hyperbolic rotations.

Lemma 2.2.1 *Let a hyperbolic rotation, H, be computed using the representations HR1–HR4* (2.2.5)–(2.2.8)*. The errors, $\Delta H := fl(H) - H$, in forming the* *hyperbolic rotation using these four representations are bounded by*

$$\begin{aligned}
\text{HR1}: &\quad \|\Delta H\|_2 \le \sqrt{2}\left(\frac{1+\gamma_2}{\sqrt{1-\gamma_2\frac{x_1^2+x_2^2}{x_1^2-x_2^2}}}-1\right)\|H\|_2\,,\\
\text{HR2}: &\quad \|\Delta H\|_2 \le \sqrt{2}\gamma_5\,\|H\|_2\,,\\
\text{HR3}: &\quad \|\Delta H\|_2 \le \sqrt{2}\left(\frac{1+\gamma_2}{\sqrt{1-\gamma_4\frac{x_1^2+x_2^2}{x_1^2-x_2^2}}}-1\right)\|H\|_2\,,\\
\text{HR4}: &\quad \|\Delta H\|_2 \le \sqrt{2}\left(\frac{1+\gamma_2}{\sqrt{1-\gamma_5\frac{x_1^2+x_2^2}{x_1^2-x_2^2}}}-1\right)\|H\|_2\,.
\end{aligned}$$

Proof. We will make use of the results in Section 1.7 without quoting the exact references. Note that each $|\delta_i| \le u$.

We look at the errors in forming the elements of HR1.

$$\begin{aligned}
fl\left(\frac{x_1}{\sqrt{x_1^2-x_2^2}}\right) &= \frac{x_1\,(1+\delta_1)}{\sqrt{(x_1^2\,(1+\delta_2)-x_2^2\,(1+\delta_3))\,(1+\delta_4)}\,(1+\delta_5)}\\
&= \frac{x_1}{\sqrt{x_1^2-x_2^2}}\left(\frac{(1+\theta_2)\sqrt{x_1^2-x_2^2}}{\sqrt{x_1^2\,(1+\theta_2')-x_2^2\,(1+\theta_2'')}}\right)\\
&= \frac{x_1}{\sqrt{x_1^2-x_2^2}}(1+\epsilon_1),
\end{aligned}$$

where

$$\begin{aligned}
|\epsilon_1| &= \left|\frac{(1+\theta_2)\sqrt{x_1^2-x_2^2}-\sqrt{x_1^2\,(1+\theta_2')-x_2^2\,(1+\theta_2'')}}{\sqrt{x_1^2\,(1+\theta_2')-x_2^2\,(1+\theta_2'')}}\right|\\
&\le \frac{(1+\gamma_2)\sqrt{x_1^2-x_2^2}-\sqrt{x_1^2(1-\gamma_2)-x_2^2(1+\gamma_2)}}{\sqrt{x_1^2(1-\gamma_2)-x_2^2(1+\gamma_2)}}\\
&= \frac{(1+\gamma_2)}{\sqrt{1-\gamma_2\frac{x_1^2+x_2^2}{x_1^2-x_2^2}}}-1,
\end{aligned}$$

and similarly

$$fl\left(\frac{-x_2}{\sqrt{x_1^2-x_2^2}}\right) = \frac{-x_2}{\sqrt{x_1^2-x_2^2}}(1+\epsilon_2),$$

where

$$|\epsilon_2| \leq \frac{(1+\gamma_2)}{\sqrt{1-\gamma_2\frac{x_1^2+x_2^2}{x_1^2-x_2^2}}} - 1.$$

Hence

$$|\Delta H| \leq \left(\frac{(1+\gamma_2)}{\sqrt{1-\gamma_2\frac{x_1^2+x_2^2}{x_1^2-x_2^2}}} - 1 \right) |H|$$

and the normwise bound follows using Lemma 1.4.1 since rank$(H) = 2$.

We now look at the errors in forming the elements of HR2. We have

$$fl\left(\frac{x_1}{\sqrt{(x_1+x_2)(x_1-x_2)}}\right) = \frac{x_1}{\sqrt{(x_1+x_2)(x_1-x_2)}}\left(1+\theta_5\right),$$

where $|\theta_5| \leq \gamma_5$, and similarly

$$fl\left(\frac{-x_2}{\sqrt{(x_1+x_2)(x_1-x_2)}}\right) = \frac{-x_2}{\sqrt{(x_1+x_2)(x_1-x_2)}}\left(1+\theta_5'\right),$$

where $|\theta_5'| \leq \gamma_5$, hence $|\Delta H| \leq \gamma_5 |H|$. The normwise bound follows once again follows using Lemma 1.4.1.

The results for HR3 and HR4 are proved in an entirely analogous way to that of HR1. $\square$

Thus we can see that HR2 is guaranteed to be computed most accurately all the time, though when $|x_1| \gg |x_2|$ we would expect all the above representations to be accurate. We can also show numerical evidence that this is the case. MATLAB was used to carry out experiments to find out how accurately HR1–HR4 were computed. We computed $\|\Delta H\|_2 / \|H\|_2$ for each representation, where H was the representation HR1 computed to 100 significant digits, and $\Delta H = fl(H) - H$. HR1, HR3 and HR4 were less accurately computed as the relative size of x_2 approached that of x_1. However, HR2 was still, at that stage, computing the hyperbolic rotation very accurately. The direct search routines of

Table 2.2.1: The relative errors, $\|\Delta H\|_2 / \|H\|_2$, in forming H with representations HR1–HR4 in (2.2.5)–(2.2.8) using the vector $x = [x_1\ x_2]^T$. The value $\alpha := (|x_1| - |x_2|)/\sqrt{x_1^2 + x_2^2}$ is the relative difference in magnitude of x_1 and x_2.

α	HR1	HR2	HR3	HR4
1e-02	1.6e-15	0	3.0e-16	7.5e-16
1e-04	2.3e-13	0	7.1e-14	2.5e-14
1e-06	2.7e-12	0	2.0e-11	1.2e-11
1e-08	6.5e-10	7.6e-17	1.5e-09	1.8e-09
1e-10	1.4e-08	0	1.7e-08	1.7e-08
1e-12	7.8e-06	0	1.9e-05	1.9e-05
1e-14	2.5e-04	7.8e-17	1.3e-03	1.3e-03
1e-16	1.1e-01	1.3e-16	1.3e-01	1.3e-01

Higham [41] were also used to try and find examples in which the errors were large. The results are summarized in Table 2.2.1.

In view of Lemma 2.2.1 and the results in Table 2.2.1 it would seem reasonable to redefine c and s since the representation in equation (2.2.2) was for HR1, and we have just shown that HR2 is the most stable hyperbolic rotation to form out of HR1–HR4. Thus, we redefine c and s as

$$c = \frac{x_1}{\sqrt{(x_1 + x_2)(x_1 - x_2)}}, \qquad s = \frac{x_2}{\sqrt{(x_1 + x_2)(x_1 - x_2)}} \tag{2.2.16}$$

and hence

$$\widehat{c} = c\,(1 + \theta_5)\,, \qquad \widehat{s} = s\,(1 + \theta_5')\,. \tag{2.2.17}$$

We describe the construction of H in the following algorithm.

Algorithm 2.2.2

```
    function $[c, s] = \mathrm{Hrotate}(x_1, x_2)$
```

```
% Compute $c$ and $s$ defining hyperbolic rotation $H$ as described
% in (2.2.16) such that $Hx$ has zero second element.
if $|x_1| > |x_2|$
    $d = \sqrt{(x_1 + x_2)(x_1 - x_2)}$
    $[c\ s] = [x_1\ x_2]/d$
else
    No real rotation exists—abort.
end
```

Cost: 5 flops.

Out of HR1–HR4, we will only analyze the error in applying HR2 to a vector since HR1, HR3 and HR4 are not always guaranteed to be accurate.

We express the application of HR2 as follows.

Algorithm 2.2.3

```
function $B = \mathrm{Happly}(c, s, A)$
% Apply hyperbolic rotation HR2 defined by $c$ and $s$
% to the $2 \times n$ matrix $A$ to obtain $B$.
$B(1,:) = cA(1,:) - sA(2,:)$
$B(2,:) = cA(2,:) - sA(1,:)$
```

Cost: 6 flops.

Applying HR2 in this way leads to the following result.

Lemma 2.2.4 *Let a hyperbolic rotation, H, be formed using the representation HR2 in* (2.2.6) *as described in Algorithm 2.2.2 and applied to a vector $a = [a_1\ a_2]^T$ to obtain $b = [b_1\ b_2]^T$ as described in Algorithm 2.2.3. The error in computing the product $b = Ha$ can be expressed in three ways. As a forward error bound:*

$$\widehat{b} = b + \Delta b, \qquad |\Delta b| \le \gamma_7 |H||a|,$$

as a backward error bound on a:

$$\widehat{b} = H(a + \varDelta a), \qquad |\varDelta a| \le \gamma_7 |a|,$$

and as a backward error bound on H:

$$\widehat{b} = (H + \varDelta H)a, \qquad |\varDelta H| \le \gamma_7 |H|.$$

Proof. When we apply HR2 (2.2.6) to the vector a, using (2.2.17), we get

$$\begin{aligned}
\widehat{b_1} &= fl\,(ca_1 - sa_2) \\
&= (c\,(1+\theta_5)\,a_1\,(1+\delta_6) - s\,(1+\theta_5')\,a_2\,(1+\delta_7))\,(1+\delta_8) \\
&= (ca_1 - sa_2 + ca_1(\theta_5 + \delta_6 + \delta_8) - sa_2(\theta_5' + \delta_7 + \delta_8)) \\
&= (ca_1\,(1+\theta_7) - sa_2\,(1+\theta_7')) \\
&= [\,c \quad -s\,] \begin{bmatrix} 1+\theta_7 & 0 \\ 0 & 1+\theta_7' \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix},
\end{aligned}$$

and similarly,

$$\widehat{b_2} = [\,-s \quad c\,] \begin{bmatrix} 1+\theta_7'' & 0 \\ 0 & 1+\theta_7''' \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}.$$

Combining these results and using $|\theta_7|, |\theta_7'|, |\theta_7''|, |\theta_7'''| \le \gamma_7$, we obtain

$$\begin{aligned}
\widehat{b} = \begin{bmatrix} \widehat{b_1} \\ \widehat{b_2} \end{bmatrix} &= H(I + \varDelta_1)a, \qquad |\varDelta_1| \le \gamma_7 I \qquad (2.2.18) \\
&= b + \varDelta b, \qquad |\varDelta b| \le |H||\varDelta_1||a| \le \gamma_7 |H||a|,
\end{aligned}$$

which is a forward error in computing b. Using (2.2.18), a backward error bound on a for computing b is given by

$$\widehat{b} = H(a + \varDelta a), \qquad |\varDelta a| \le |\varDelta_1||a| \le \gamma_7 |a|$$

and a backward error bound on H is given by

$$\widehat{b} = (H + \varDelta H)a, \qquad |\varDelta H| \le |H||\varDelta_1| \le \gamma_7 |H|. \qquad \square$$

We now analyze HR5. We express the application of HR5 as follows. Note that for this representation, we form and apply HR5 to a vector directly and do not explicitly form the elements c and s of the hyperbolic rotation H as described in Algorithm 2.2.2.

Algorithm 2.2.5

```
function $B = \text{Happly}(x_1, x_2, A)$
% Apply hyperbolic rotation HR5 to $2 \times n$ matrix $A$ to obtain $B$.
$d = \sqrt{\frac{x_1+x_2}{x_1-x_2}}$
$B = [\begin{smallmatrix} 1 & -1 \\ 1 & 1 \end{smallmatrix}]A$
$B = [\begin{smallmatrix} d/2 & 0 \\ 0 & 1/(2d) \end{smallmatrix}]B$
$B = [\begin{smallmatrix} 1 & 1 \\ -1 & 1 \end{smallmatrix}]B$
```

Cost: 14 flops.

We now look at the error analysis for applying HR5 (2.2.9) as described in Algorithm 2.2.5. Due to the way it is factorized, we can perform a slightly different analysis to prove mixed forward-backward stability. We state and prove this result below.

Lemma 2.2.6 *Let a hyperbolic rotation, $H = QDQ^T$, be constructed according to (2.2.9). Let $a = [a_1 \ a_2]^T \in \mathbb{R}^2$ and consider the computation of $b = [b_1 \ b_2]^T = Ha$. The computed $\widehat{b}$ satisfies*

$$\widehat{b} + \Delta b = H(a + \Delta a), \qquad \|\Delta b\|_2 \le \gamma_1 \|\widehat{b}\|_2, \quad \|\Delta a\|_2 \le 4\gamma_6 \|a\|_2 .$$

Proof. Let $b' = Q^T a$, $b'' = Db'$ and $b = Qb''$. We then have

$$\begin{aligned}
\widehat{b'} &= fl(Q^T a) \\
&= \begin{bmatrix} 1+\delta_1 & 0 \\ 0 & 1+\delta_2 \end{bmatrix} \begin{bmatrix} a_1 - a_2 \\ a_1 + a_2 \end{bmatrix}, \qquad |\delta_1|, |\delta_2| \le u, \\
&= (I + \Delta_1)Q^T a, \qquad \Delta_1 = \text{diag}(\delta_1, \delta_2), \quad |\Delta_1| \le \gamma_1 I.
\end{aligned}$$

Now,

$$\widehat{d} = fl\left(\sqrt{\frac{x_1+x_2}{x_1-x_2}}\right)$$

$$\begin{aligned} &= \sqrt{\frac{(x_1+x_2)(1+\delta_3)}{(x_1-x_2)(1+\delta_4)}(1+\delta_5)(1+\delta_6)} \\ &= d(1+\theta_4), \end{aligned}$$

and so $\widehat{d/2} = d(1+\theta_4)/2$ and $\widehat{1/(2d)} = (1+\theta_5)/(2d)$ since dividing by 2 can be considered to be done exactly on a computer. Thus, when we apply D to $\widehat{b}'$, we get

$$\begin{aligned} \widehat{b}'' &= fl(D\widehat{b}') \\ &= fl(D((I+\varDelta_1)Q^T a)) \\ &= D(I+\varDelta_2)(I+\varDelta_1)Q^T a, \qquad |\varDelta_2| \le \gamma_5 I, \\ &= D(I+\varDelta_3)Q^T a, \qquad |\varDelta_3| \le \gamma_6 I. \end{aligned}$$

Applying Q to $\widehat{b}''$ gives us

$$\begin{aligned} \widehat{b} &= fl(Q\widehat{b}'') \\ &= fl(Q(D(I+\varDelta_3)Q^T a)) \\ &= (I+\varDelta_4)QD(I+\varDelta_3)Q^T a, \qquad |\varDelta_4| \le \gamma_1 I. \end{aligned}$$

This gives us

$$\begin{aligned} (I+\varDelta_5)\widehat{b} &= QD(I+\varDelta_3)Q^T a, \qquad |\varDelta_5| \le \gamma_1 I, \\ &= QDQ^TQ(I+\varDelta_3)Q^T a, \qquad \text{since } Q^TQ = I, \\ &= QDQ^T(I+Q\varDelta_3 Q^T)a \\ &= QDQ^T(I+\varDelta_6)a, \end{aligned}$$

where

$$|\varDelta_6| = \left|Q\varDelta_3 Q^T\right| \le |Q|\,|\varDelta_3|\,\left|Q^T\right| \le \gamma_6\,|Q|^2 = \gamma_6 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}^2 = 2\gamma_6 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = 2\gamma_6\,|Q|\,.$$

Hence we have

$$\widehat{b} + \varDelta b = H(a+\varDelta a), \qquad |\varDelta b| \le \gamma_1 |\widehat{b}|, \quad |\varDelta a| \le 2\gamma_6\,|Q|\,|a|$$

and the normwise bounds follow on using $\||Q|\|_2 = 2$. $\square$

Lastly, we analyze HR6. We express the application of HR6 as follows.

Algorithm 2.2.7

```
function $B = \text{Happly}(c, s, B)$
% Apply hyperbolic rotation defined by $c$ and $s$ to $2 \times n$ matrix $B$.
for $j = 1{:}\,n$
    $B(1,j) = cB(1,j) - sB(2,j)$
    $B(2,j) = -(s/c)B(1,j) + B(2,j)/c$
end
```

Cost: 7 flops.

Note once again that we are using (2.2.16) to compute c and s.

Since HR6 in (2.2.14) was obtained using the exchange operator, we can analyze it in terms of applying an orthogonal transformation, which is backward stable, and obtain a very satisfactory mixed forward-backward stability result as we show in the following lemma.

Lemma 2.2.8 *Let a hyperbolic rotation, H, be constructed according to HR6 (2.2.14). The error in applying H to a vector $a = [a_1\ a_2]^T$ to obtain $b = [b_1\ b_2]^T$ satisfies*

$$\widehat{b} + \Delta b = H(a + \Delta a),$$

where

$$\Delta b = \begin{bmatrix} \Delta b_1 \\ 0 \end{bmatrix}, \qquad \Delta a = \begin{bmatrix} 0 \\ \Delta a_2 \end{bmatrix}, \qquad \max(|\Delta b_1|, |\Delta a_2|) \le \gamma_{20} \max(|\widehat{b}_1|, |a_2|).$$

Proof. Using c and s from (2.2.16) and their computed values from (2.2.17), the computed quantities from (2.2.11) and (2.2.13) satisfy

$$\widehat{b}_1 = (c\,(1+\theta_5)\,a_1\,(1+\delta_1) - s\,(1+\theta_5')\,a_2\,(1+\delta_2))\,(1+\delta_3)\,,$$

that is,

$$a_1 = \frac{\widehat{b}_1}{c}(1+\theta_7) + \frac{s}{c}a_2(1+\theta_7'),$$

and

$$\widehat{b}_2 = -\frac{s}{c}\widehat{b}_1(1+\theta_{10}) + \frac{a_2}{c}(1+\theta_7'').$$

Hence the analogue of (2.2.15) for the computed quantities is

$$\begin{aligned}\begin{bmatrix} a_1 \\ \widehat{b}_2 \end{bmatrix} &= \begin{bmatrix} g_{11}(1+\theta_7) & g_{12}(1+\theta_7') \\ g_{21}(1+\theta_{10}) & g_{22}(1+\theta_7'') \end{bmatrix} \begin{bmatrix} \widehat{b}_1 \\ a_2 \end{bmatrix} \\ &= (G+\Delta G)\begin{bmatrix} \widehat{b}_1 \\ a_2 \end{bmatrix}, \qquad |\Delta G| \le \gamma_{10}|G|,\end{aligned}$$

where G is orthogonal. This result can be rewritten as

$$\begin{bmatrix} a_1 \\ \widehat{b}_2 \end{bmatrix} = G\begin{bmatrix} \widehat{b}_1 + \Delta b_1 \\ a_2 + \Delta a_2 \end{bmatrix}, \tag{2.2.19}$$

where

$$\begin{bmatrix} \Delta b_1 \\ \Delta a_2 \end{bmatrix} = G^T \Delta G \begin{bmatrix} \widehat{b}_1 \\ a_2 \end{bmatrix},$$

so that

$$\max(|\Delta b_1|, |\Delta a_2|) \le \gamma_{10}(1+2|\widetilde{c}||\widetilde{s}|)\max(|\widehat{b}_1|, |a_2|) \le \gamma_{20}\max(|\widehat{b}_1|, |a_2|),$$

where $\widetilde{c} = 1/c$, $\widetilde{s} = s/c$. Finally, applying the exchange operator of Section 2.1.2 to G, we recover the hyperbolic rotation H and can rewrite (2.2.19) as

$$\begin{bmatrix} \widehat{b}_1 + \Delta b_1 \\ \widehat{b}_2 \end{bmatrix} = H\begin{bmatrix} a_1 \\ a_2 + \Delta a_2 \end{bmatrix},$$

hence proving the result. $\square$

This is a mixed forward-backward error result, since one element of each of the input and output vectors is perturbed.

We have shown the representations of three hyperbolic rotations to be stable. Given in Table 2.2.2 is a summary of the errors in applying these three representations to vectors. The results were obtained in much the same way as for the

Table 2.2.2: The relative errors, $\|fl(Ha) - Ha\|_2 / \|Ha\|_2$, in forming and applying the hyperbolic rotation H with representations HR2, HR5 and HR6 in (2.2.6), (2.2.9) and (2.2.14) to the vector $a = [a_1\ a_2]^T$. The values $\alpha_1 := (|x_1| - |x_2|)/\sqrt{x_1^2 + x_2^2}$ and $\alpha_2 := (|a_1| - |a_2|)/\sqrt{a_1^2 + a_2^2}$ are the relative differences in magnitude of the elements in vectors x and a respectively.

α_1	α_2	HR2	HR5	HR6	$\|H\|_2$
1e-01	1e-01	5.2e-16	2.0e-17	3.1e-16	3.8e+00
1e-01	1e-16	1.4e-15	1.1e-15	3.0e-16	3.8e+00
1e-06	1e-12	6.0e-11	7.9e-12	4.3e-11	1.2e+03
1e-12	1e-12	8.9e-06	5.6e-06	1.3e-06	1.2e+06
1e-16	1e-01	3.5e-16	2.5e-16	6.3e-16	1.2e+08
1e-16	1e-16	1.1e+00	1.1e+00	1.9e-01	1.2e+08

results in Table 2.2.1. MATLAB was used to carry out experiments to find out how accurately HR2, HR5 and HR6 were formed and applied to a random vector $a = [a_1\ a_2]^T$. We computed $\|fl(Ha) - Ha\|_2 / \|Ha\|_2$ for each representation, where Ha was HR1 formed and then applied to a and answers produced to 100 significant digits.

Just as we saw for HR2 in Table 2.2.1, HR6 was also formed very accurately, and in all three cases, the relative difference in x_1 and x_2 did not affect the accuracy with which the hyperbolic rotations were formed. All three representations performed about as well as each other in the tests, with the errors in computing the application of the three hyperbolic rotations not differing by more than 10^2.

However, there were a couple of things that were noticed. Firstly, the hyperbolic rotations were applied to the vector $a = [a_1\ a_2]^T$, and as the relative difference between a_1 and a_2 decreased, the errors in applying the hyperbolic rotations to a increased in all the three representations. This meant the errors in applying the rotations were proportional to its norm. This agrees with the error

Table 2.2.3: More results like the ones in Table 2.2.2 for α_1 = 1e-6.

α_2	HR2	HR5	HR6
1e-2	6.0e-15	7.9e-16	5.8e-15
1e-4	6.0e-13	7.9e-14	1.7e-13
1e-6	4.2e-11	5.6e-12	1.3e-11
1e-8	6.0e-11	7.9e-12	5.4e-11
1e-10	6.0e-11	7.9e-12	4.9e-11
1e-12	6.0e-11	7.9e-12	4.3e-11
1e-14	6.0e-11	7.9e-12	7.2e-12
1e-16	1.2e-10	1.1e-10	1.2e-10

bound we got for applying HR2 to a vector in Lemma 2.2.4, but not with the corresponding results for HR5 and HR6 in Lemmas 2.2.6 and 2.2.8 respectively.

Secondly, in Table 2.2.3, there is one example of when the error did not increase significantly as the relative differences in a_1 and a_2 decreased after a certain stage. This phenomenon occurred for larger relative differences in a_1 and a_2 as the relative difference in x_1 and x_2 decreased. We have been unable to find any explanation for this phenomenon.

2.2.4 Products of Hyperbolic Rotations

Now that we have three stable representations of hyperbolic rotations, we can analyze products of them.

We first need to establish the notion of disjoint hyperbolic rotations. As we have seen earlier, when we premultiply a vector x by a hyperbolic rotation $H_{i,j}$ we change only the ith and jth rows of x. Thus, H_{i_1,j_1} and H_{i_2,j_2} commute, i.e.

$$H_{i_1,j_1}H_{i_2,j_2} = H_{i_2,j_2}H_{i_1,j_1},$$

for distinct integers i_1, i_2, j_1, j_2. We say that H_{i_1,j_1} and H_{i_2,j_2} are *disjoint*, or *nonconflicting*, if this is the case.

Lemma 2.2.9 *Let $W = H_{i_k,j_k} H_{i_{k-1},j_{k-1}} \cdots H_{i_1,j_1} \in \mathbb{R}^{m\times m}$ be a product of hyperbolic rotations such that the integers i_l, j_l, $l = 1\colon k$, $2k \le m$, are all distinct. Let $a \in \mathbb{R}^m$ and consider the computation of $b = Wa$.*

- *If each H_{i_l,j_l}, $l = 1\colon k$, is constructed according to HR2 in (2.2.6), then the computed $\widehat{b}$ satisfies*

$$\widehat{b} = W(a + \Delta a), \qquad \|\Delta W\|_2 \le \gamma_5 \|W\|_2 .$$

- *If each $H_{i_l,j_l} = Q_l D_l Q_l^T$, $l = 1\colon k$, is constructed according to HR5 in (2.2.9) so that Q_l and D_l satisfy (2.2.10), then the computed $\widehat{b}$ satisfies*

$$\widehat{b} + \Delta b = W(a + \Delta a), \qquad \|\Delta b\|_2 \le \gamma_1 \|\widehat{b}\|_2, \quad \|\Delta a\|_2 \le 4\gamma_6 \|a\|_2 .$$

- *If each H_{i_l,j_l}, $l = 1\colon k$, is constructed according to HR6 in (2.2.14), then the computed $\widehat{b}$ satisfies*

$$\widehat{b} + \Delta b = W(a + \Delta a),$$

  with

$$\max(\|\Delta b_i\|_2, \|\Delta a_j\|_2) \le \sqrt{k}\gamma_{20} \max(\|b_i\|_2, \|a_j\|_2),$$

  where $b_i \colon = b([i_1, i_2, \ldots, i_k])$ and $a_i \colon = a([j_1, j_2, \ldots, j_k])$.

Moreover, in each of the three cases above, Δb and Δa are zero vectors except for

$$\Delta b_i = \Delta b([i_1, i_2, \ldots, i_k]) = [\Delta b_{i_1}\ \Delta b_{i_2}\ \cdots\ \Delta b_{i_k}]^T,$$
$$\Delta a_j = \Delta a([j_1, j_2, \ldots, j_k]) = [\Delta a_{j_1}\ \Delta a_{j_2}\ \cdots\ \Delta a_{j_k}]^T.$$

Proof. We prove only the second result, since the first result is a special case of it and the third result can be proved analogously.

By the disjointness of the rotations, the application of H_{i_l,j_l}, $l = 1\colon k$ to $H_{i_{l-1},j_{l-1}} \dots H_{i_1,j_1} a$ only alters $a([i_l, j_l])$. In other words, only elements that have not yet been altered by any of $H_{i_{l-1},j_{l-1}}, \dots, H_{i_1,j_1}$ are altered. Also, only $a([i_1, \dots, i_l, j_1, \dots, j_l])$ is altered by the product $H_{i_l,j_l} \dots H_{i_1,j_1}$. Hence, by Lemma 2.2.6,

$$\widehat{b}_l + \Delta b_l = H_l(a_l + \Delta a_l), \qquad \|\Delta b_l\|_2 \le \gamma_1 \|\widehat{b}_l\|_2, \quad \|\Delta a_l\|_2 \le 4\gamma_6 \|a_l\|_2 ,$$

where $b_l = b([i_l, j_l])$, $a_l = a([i_l, j_l])$ and $H_l = (H_{i_k,j_k} H_{i_{k-1},j_{k-1}} \dots H_{i_1,j_1})([i_l, j_l], [i_l, j_l])$, $l = 1\colon k$. The result then follows trivially. $\square$

Given an error bound for a product of disjoint orthogonal transformations, V_i, it is relatively easy to obtain a useful error bound for a product of several such products of orthogonal transformations, $b = V_k \dots V_2 V_1 a$, as first shown by Wilkinson in the 1960s. The situation is quite different in the case of hyperbolic transformations, $b = W_k \dots W_2 W_1 a$, say. It is possible to mimic the analysis for orthogonal transformations and write, for example,

$$\widehat{b} = (W_k + \Delta W_k) \dots (W_2 + \Delta W_2)(W_1 + \Delta W_1) a, \qquad \|\Delta W_j\|_2 \le \|W_j\|_2 , \qquad j = 1\colon k,$$

where the W_j are products of disjoint hyperbolic rotations constructed according to HR2 in (2.2.6). Then, using Lemma 2.2.9 and Lemma 1.7.5, the error

$$\begin{aligned} \frac{\|\widehat{b} - b\|_2}{\|b\|_2} &= \frac{\|(W_k + \Delta W_k) \dots (W_2 + \Delta W_2)(W_1 + \Delta W_1) a - W_k \dots W_2 W_1 a\|_2}{\|W_k \dots W_2 W_1 a\|_2} \\ &\le \left(\prod_{j=0}^{k} (1 + \gamma_5) - 1 \right) \frac{\|W_k\|_2 \dots \|W_2\|_2 \|W_1\|_2 \|a\|_2}{\|W_k \dots W_2 W_1 x\|_2}, \end{aligned}$$

which is unbounded since the W_j are unbounded in norm. Hence, forming the product in this way does not lead to a satisfactory forward or backward error bound.

We now consider applying groups of disjoint hyperbolic rotations of the form HR5 in (2.2.9) one by one to a vector a. We first analyze the error in applying three groups of disjoint rotations to a. In other words, let $b = W_3 W_2 W_1 a$ and define $b' = W_1 a$ and $b'' = W_2 b'$ so that $b = W_3 b''$. Using Lemma 2.2.9, we have

$$\widehat{b'} + \varDelta_F b' = W_1(a + \varDelta_B a),$$

where

$$\|\varDelta_F b'\|_2 \leq \gamma_1 \|\widehat{b'}\|_2, \qquad \|\varDelta_B a\|_2 \leq 4\gamma_6 \|a\|_2.$$

Now applying W_2 to $\widehat{b'} + \varDelta_F b'$ we get

$$\begin{aligned}\widehat{b''} + \varDelta_F b'' &= W_2(\widehat{b'} + \varDelta_F b' + \varDelta_B b') \\ &= W_2 W_1 (a + \varDelta_B a + W_1^{-1} \varDelta_B b'),\end{aligned}$$

where

$$\begin{aligned}\|\varDelta_F b''\|_2 &\leq \gamma_1 \|\widehat{b''}\|_2, \\ \|\varDelta_B a + W_1^{-1} \varDelta_B b'\|_2 &\leq 4\gamma_6 \|a\|_2 + \|W_1^{-1}\|_2 4\gamma_6 \|W_1\|_2 \|a\|_2 \\ &= 4\gamma_6 \|a\|_2 (1 + \kappa_2(W_1)).\end{aligned}$$

Finally, applying W_3 to $\widehat{b''} + \varDelta_F b''$ we get

$$\begin{aligned}\widehat{b} + \varDelta_F b &= W_3(\widehat{b''} + \varDelta_F b'' + \varDelta_B b'') \\ &= W_3 W_2 W_1 (a + \varDelta_B a + W_1^{-1} \varDelta_B b' + W_1^{-1} W_2^{-1} \varDelta_B b''),\end{aligned}$$

where

$$\begin{aligned}\|\varDelta_F b\|_2 &\leq \gamma_1 \|\widehat{b}\|_2, \\ \|\varDelta_B a + W_1^{-1} \varDelta_B b' + W_1^{-1} W_2^{-1} \varDelta_B b''\|_2 &\leq 4\gamma_6 \|a\|_2 (1 + \kappa_2(W_1)) \\ &\quad + \|W_1^{-1}\|_2 \|W_2^{-1}\|_2 4\gamma_6 \|W_2\|_2 \|W_1\|_2 \|a\|_2 \\ &= 4\gamma_6 \|a\|_2 (1 + \kappa_2(W_1) + \kappa_2(W_1)\kappa_2(W_2)).\end{aligned}$$

We can generalize this result to the case when we are applying k groups of disjoint rotations to a. If is straightforward to see that we would get

$$\widehat{b} + \varDelta_F b = W_k \ldots W_2 W_1 (a + \varDelta_B a),$$

where

$$\begin{aligned}\|\Delta_F b\|_2 &\le \gamma_1 \|\widehat{b}\|_2,\\ \|\Delta_B a\|_2 &\le 4\gamma_6 \|a\|_2 \sum_{j=0}^{k-1} \prod_{i=1}^{j} \kappa_2(W_i).\end{aligned}$$

Thus, we see that applying HR5 also does not give us satisfactory bounds since the backward error will always be bounded by sums of products of the condition numbers of disjoint groups of hyperbolic rotations which are themselves, by equation (2.2.4), unbounded.

It is obvious that by performing our error analysis in this way we would encounter a similar problem when applying products of disjoint groups of hyperbolic rotations. However, there are other methods of forming products of hyperbolic rotations, and other J-orthogonal transformations, without worrying about their disjointness. Later, in Section 2.3, we will need to form products of J-orthogonal transformations in a special way to compute the hyperbolic QR factorization of a matrix. A better approach to analyze the error of applying a product of J-orthogonal transformations is to exploit the equivalence between orthogonal and hyperbolic transformations using the exchange operator which we introduced in Section 2.1.2. Note that (2.2.15) is a special case of (2.1.4).

As we mentioned in Section 2.1.2, the advantage of (2.1.4) is that because the transformation matrix is orthogonal error terms can be moved around in the equation without changing their norm. The disadvantage is that it is hard to analyze more than one transformation. For example, let $C = PA$ where P is J-orthogonal. Then $C = PQB$ and corresponding to (2.1.4) we have

$$\begin{bmatrix} A_1 \\ C_2 \end{bmatrix} = \operatorname{exc}(PQ) \begin{bmatrix} C_1 \\ A_2 \end{bmatrix}. \tag{2.2.20}$$

Despite the elegance of this relation, $\operatorname{exc}(PQ)$ is a complicated function of P and Q. In practice the equations $A = QB$ and $C = PA$ must be modified to include rounding error terms, and these terms appear to preclude a suitably perturbed version of (2.2.20) with satisfactory bounds on the perturbations.

The gist of this analysis is that it is unclear how to obtain useful error bounds for the product of two or more *arbitrary* J-orthogonal transformations. Fortunately, the transformations that we will be dealing with in Section 2.3 are far from arbitrary, and we show below that by exploiting their structure we can make useful progress.

We analyze a product of two J-orthogonal transformations, where J is given in (2.1.2), that satisfy one key assumption: that the two transformations are "non-overlapping" in components $1\colon p$. Non-overlapping means that for $i = 1\colon p$ at least one of the two transformations agrees with the identity matrix in row i and column i. Without loss of generality, we consider a transformation Q_1 agreeing with the identity matrix in rows and columns $1\colon t$ and a transformation Q_2 agreeing with the identity matrix in rows and columns $t{+}1\colon p$, where $1 \le t < p$. Let

$$Q_1 \begin{bmatrix} R \\ S \\ X \end{bmatrix} \begin{array}{l} {\scriptstyle t} \\ {\scriptstyle p-t} \\ {\scriptstyle q} \end{array} =: \begin{bmatrix} R \\ S_1 \\ X_1 \end{bmatrix} \begin{array}{l} {\scriptstyle t} \\ {\scriptstyle p-t} \\ {\scriptstyle q} \end{array}, \qquad Q_2 \begin{bmatrix} R \\ S_1 \\ X_1 \end{bmatrix} =: \begin{bmatrix} R_1 \\ S_1 \\ X_2 \end{bmatrix}, \tag{2.2.21}$$

or, overall,

$$Q_2 Q_1 \begin{bmatrix} R \\ S \\ X \end{bmatrix} = \begin{bmatrix} R_1 \\ S_1 \\ X_2 \end{bmatrix}.$$

We now have the following lemma which analyzes the errors in forming the product of two hyperbolic rotations in this way.

Lemma 2.2.10 *Let $Q_1, Q_2 \in \mathbb{R}^{m\times m}$ be the two J-orthogonal transformations of* (2.2.21) *satisfying the non-overlapping assumption described above. Suppose that* *the error in applying Q_1 to the vector $[R^T\ S^T\ X^T]^T$ is given by*

$$Q_1 \begin{bmatrix} R \\ S \\ X + \Delta X \end{bmatrix} = \begin{bmatrix} R \\ S_1 + \Delta S_1 \\ X_1 \end{bmatrix} \tag{2.2.22a}$$

and the error in applying Q_2 to this new vector $[R^T\ (S_1 + \Delta S_1)^T\ X_1^T]^T$ is given by

$$Q_2 \begin{bmatrix} R \\ S_1 + \Delta S_1 \\ X_1 + \Delta X_1 \end{bmatrix} = \begin{bmatrix} R_1 + \Delta R_1 \\ S_1 + \Delta S_1 \\ X_2 \end{bmatrix}, \tag{2.2.22b}$$

where

$$\max(\|\Delta S_1\|_2, \|\Delta X\|_2) \leq \mu \max(\|S_1\|_2, \|X\|_2), \tag{2.2.23a}$$

$$\max(\|\Delta R_1\|_2, \|\Delta X_1\|_2) \leq \mu \max(\|R_1\|_2, \|X_1\|_2). \tag{2.2.23b}$$

We then have

$$Q_2 Q_1 \begin{bmatrix} R \\ S \\ X + \overline{\Delta X} \end{bmatrix} = \begin{bmatrix} R_1 + \overline{\Delta R_1} \\ S_1 + \overline{\Delta S_1} \\ X_2 \end{bmatrix},$$

where

$$\max(\|\overline{\Delta R_1}\|_2, \|\overline{\Delta S_1}\|_2, \|\overline{\Delta X}\|_2) \leq 3\mu \max(\|R_1\|_2, \|X\|_2, \|S_1\|_2) + O(\mu^2).$$

Proof. We know from (2.1.3) and (2.1.4) that the operations (2.2.21) can be rewritten in terms of orthogonal transformations P_1 ad P_2 as follows, where we now express the relations in terms of the affected components only:

$$P_1 \begin{bmatrix} S_1 \\ X \end{bmatrix} = \begin{bmatrix} S \\ X_1 \end{bmatrix}, \tag{2.2.24a}$$

$$P_2 \begin{bmatrix} R_1 \\ X_1 \end{bmatrix} = \begin{bmatrix} R \\ X_2 \end{bmatrix}. \tag{2.2.24b}$$

These two relations can be rewritten as

$$\begin{bmatrix} R_1 \\ S_1 \\ X \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & P_1^T \end{bmatrix} \begin{bmatrix} R_1 \\ S \\ X_1 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & P_1^T \end{bmatrix} \widetilde{P}_2^T \begin{bmatrix} R \\ S \\ X_2 \end{bmatrix} \equiv P \begin{bmatrix} R \\ S \\ X_2 \end{bmatrix}, \tag{2.2.25}$$

where $\widetilde{P}_2([1\colon t, p+1\colon m],[1\colon t, p+1\colon m]) = P_2$ and elsewhere $\widetilde{P}_2$ agrees with the identity matrix, and P is orthogonal. This relation shows that $\operatorname{exc}(Q_2Q_1) = P$ is of a relatively simple form given the no-overlap assumption.

Now we incorporate errors into the analysis. Consider the perturbed versions of (2.2.24),

$$P_1 \begin{bmatrix} S_1 + \mathit{\Delta} S_1 \\ X + \mathit{\Delta} X \end{bmatrix} = \begin{bmatrix} S \\ X_1 \end{bmatrix}, \tag{2.2.26a}$$

$$P_2 \begin{bmatrix} R_1 + \mathit{\Delta} R_1 \\ X_1 + \mathit{\Delta} X_1 \end{bmatrix} = \begin{bmatrix} R \\ X_2 \end{bmatrix}, \tag{2.2.26b}$$

where the errors are the same as in (2.2.22) and satisfy the bounds (2.2.23).

We now obtain an analogue of (2.2.25) for the perturbed quantities. We have

$$\begin{aligned}\begin{bmatrix} R_1 + \mathit{\Delta} R_1 \\ S_1 + \mathit{\Delta} S_1 \\ X + \mathit{\Delta} X \end{bmatrix} &= \begin{bmatrix} I & 0 \\ 0 & P_1^T \end{bmatrix} \begin{bmatrix} R_1 + \mathit{\Delta} R_1 \\ S \\ X_1 \end{bmatrix} \\ &= \begin{bmatrix} I & 0 \\ 0 & P_1^T \end{bmatrix} \left(\widetilde{P}_2^T \begin{bmatrix} R \\ S \\ X_2 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \mathit{\Delta} X_1 \end{bmatrix} \right).\end{aligned}$$

This may be rewritten as

$$\begin{bmatrix} R_1 \\ S_1 \\ X \end{bmatrix} + \mathit{\Delta} = P \begin{bmatrix} R \\ S \\ X_2 \end{bmatrix}, \tag{2.2.27}$$

where, using $\|X_1\|_2 \le 2\max(\|S_1\|_2, \|X\|_2) + O(\mu)$,

$$\mathit{\Delta} = \begin{bmatrix} \overline{\mathit{\Delta} R_1} \\ \overline{\mathit{\Delta} S_1} \\ \overline{\mathit{\Delta} X} \end{bmatrix}, \qquad \max_i \|\mathit{\Delta}_i\|_2 \le 3\mu \max(\|R_1\|_2, \|X\|_2, \|S_1\|_2) + O(\mu^2). \qquad \square$$

The key fact is that the error bound for the two transformations combined is commensurate with that for the individual transformations. Because P is orthogonal the relation (2.2.27) can, if desired, be rewritten so that the 3×1

block matrix on the right is perturbed instead of the one on the left, as in the assumptions (2.2.26).

Lemma 2.2.10 dealt with applying two non-overlapping J-orthogonal transformations whose errors could be written as those of the hyperbolic rotation HR6 in (2.2.14). However, we also want to be able to apply two non-overlapping J-orthogonal transformations whose errors could be written as those of the hyperbolic rotation HR5 in (2.2.9). We obtain the following analysis.

Let

$$Q_1\left(\begin{bmatrix} R \\ S \\ X \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta S \\ \Delta X \end{bmatrix}\right) = \begin{bmatrix} R \\ S_1 \\ X_1 \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta S_1 \\ \Delta X_1 \end{bmatrix}\begin{array}{l} {\scriptstyle t} \\ {\scriptstyle p-t} \\ {\scriptstyle q} \end{array}, \tag{2.2.28a}$$

and

$$Q_2\left(\begin{bmatrix} R \\ S_1 \\ X_1 \end{bmatrix} + \begin{bmatrix} \Delta R \\ 0 \\ \Delta' X_1 \end{bmatrix}\right) = \begin{bmatrix} R_1 \\ S_1 \\ X_2 \end{bmatrix} + \begin{bmatrix} \Delta R_1 \\ 0 \\ \Delta X_2 \end{bmatrix}, \tag{2.2.28b}$$

where, using Lemma 2.2.6,

$$\left\| \begin{bmatrix} \Delta S \\ \Delta X \end{bmatrix} \right\|_2 \le 4\gamma_6 \left\| \begin{bmatrix} S \\ X \end{bmatrix} \right\|_2, \tag{2.2.29a}$$

$$\left\| \begin{bmatrix} \Delta S_1 \\ \Delta X_1 \end{bmatrix} \right\|_2 \le \gamma_1 \left\| \begin{bmatrix} S_1 \\ X_1 \end{bmatrix} \right\|_2, \tag{2.2.29b}$$

$$\left\| \begin{bmatrix} \Delta R \\ \Delta' X \end{bmatrix} \right\|_2 \le 4\gamma_6 \left\| \begin{bmatrix} R \\ X_1 \end{bmatrix} \right\|_2, \tag{2.2.29c}$$

$$\left\| \begin{bmatrix} \Delta R_1 \\ \Delta X_2 \end{bmatrix} \right\|_2 \le \gamma_1 \left\| \begin{bmatrix} R_1 \\ X_2 \end{bmatrix} \right\|_2. \tag{2.2.29d}$$

Let

$$\begin{aligned} a_1 &= [R^T\ S^T\ X^T]^T, & \Delta a_1 &= [0\ \Delta S^T\ \Delta X^T]^T, \\ a_2 &= [R^T\ S_1^T\ X_1^T]^T, & \Delta' a_2 &= [0\ \Delta S_1^T\ \Delta X_1^T]^T, & \Delta a_2 &= [\Delta R^T\ 0\ \Delta' X_1^T]^T, \\ a_3 &= [R_1^T\ S_1^T\ X_2^T]^T, & \Delta' a_3 &= [\Delta R_1^T\ 0\ \Delta X_2^T]^T, \end{aligned}$$

so that (2.2.28) becomes

$$Q_1(a_1 + \Delta a_1) = a_2 + \Delta' a_2, \tag{2.2.30a}$$

and

$$Q_2(a_2 + \mathit{\Delta} a_2) = a_3 + \mathit{\Delta}' a_3. \tag{2.2.30b}$$

Using (2.2.30a), we have

$$a_2 = Q_1(a_1 + \mathit{\Delta} a_1) - \mathit{\Delta}' a_2,$$

which implies that (2.2.30b) becomes

$$Q_2(Q_1(a_1 + \mathit{\Delta} a_1) - \mathit{\Delta}' a_2 + \mathit{\Delta} a_2) = a_3 + \mathit{\Delta}' a_3.$$

In other words,

$$Q_2 Q_1(a_1 + \mathit{\Delta} a_1) = a_3 + \mathit{\Delta}' a_3 + Q_2(\mathit{\Delta}' a_2 - \mathit{\Delta} a_2) =: a_3 + \mathit{\Delta}'' a_3. \tag{2.2.31}$$

Using (2.2.29) and (2.2.28), we can see that

$$\|\mathit{\Delta} a_1\|_2 \le 4\gamma_6 \|a_1\|_2$$

and

$$\begin{aligned}
\mathit{\Delta}'' a_3 &= \mathit{\Delta}' a_3 + Q_2(\mathit{\Delta}' a_2 - \mathit{\Delta} a_2) \\
&= \begin{bmatrix} \mathit{\Delta} R_1 \\ 0 \\ \mathit{\Delta} X_2 \end{bmatrix} + Q_2 \left(\begin{bmatrix} 0 \\ \mathit{\Delta} S_1 \\ \mathit{\Delta} X_1 \end{bmatrix} - \begin{bmatrix} \mathit{\Delta} R \\ 0 \\ \mathit{\Delta}' X_1 \end{bmatrix} \right) \\
&= \begin{bmatrix} \mathit{\Delta} R_1 \\ \mathit{\Delta} S_1 \\ \mathit{\Delta} X_2 \end{bmatrix} + Q_2 \begin{bmatrix} -\mathit{\Delta} R \\ 0 \\ \mathit{\Delta} X_1 - \mathit{\Delta}' X_1 \end{bmatrix}
\end{aligned}$$

since Q_2 does not operate on $\mathit{\Delta} S_1$. Thus,

$$\begin{aligned}
\|\mathit{\Delta}'' a_3\|_2 &= \left\| \begin{bmatrix} \mathit{\Delta} R_1 \\ \mathit{\Delta} S_1 \\ \mathit{\Delta} X_2 \end{bmatrix} + Q_2 \begin{bmatrix} -\mathit{\Delta} R \\ 0 \\ \mathit{\Delta} X_1 - \mathit{\Delta}' X_1 \end{bmatrix} \right\|_2 \\
&\le \left\| \begin{bmatrix} \mathit{\Delta} R_1 \\ \mathit{\Delta} S_1 \\ \mathit{\Delta} X_2 \end{bmatrix} \right\|_2 + \|Q_2\|_2 \left\| \begin{bmatrix} -\mathit{\Delta} R \\ 0 \\ \mathit{\Delta} X_1 - \mathit{\Delta}' X_1 \end{bmatrix} \right\|_2 .
\end{aligned}$$

We can rewrite this result so that Δa_1 depends on Q_1 and $\Delta'' a_3$ does not depend on Q_2, but the gist of the analysis is that since Q_2 (or Q_1) can be unbounded in norm, we conclude that we cannot guarantee the stability of applying more than one non-conflicting J-orthogonal transformation with errors like that of HR5.

We will compare numerical results for applying non-overlapping products of hyperbolic rotations with representations HR2, HR5 and HR6 in Section 2.3, where we will be using them to compute the hyperbolic QR factorization of a matrix.

2.3 The Hyperbolic QR Factorization

The concept of J-orthogonality gives rise to numerous matrix factorizations. The hyperbolic QR factorization is one of these. As we shall see later, this factorization helps us to compute a solution to the ILS problem. We introduce the hyperbolic QR factorization below.

2.3.1 Introduction

Assume that the matrix $A \in \mathbb{R}^{m\times n}$ is such that A^TJA is positive definite, where J is as defined in (2.1.2). Then the hyperbolic QR factorization of A, where $m \geq n$, is the product

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = [\, Q_1 \quad Q_2 \,] \begin{bmatrix} R \\ 0 \end{bmatrix} = Q_1 R \tag{2.3.1}$$

of a J-orthogonal matrix $Q \in \mathbb{R}^{m\times m}$ and a matrix $[R^T \;\; 0]^T$, where $R \in \mathbb{R}^{n\times n}$ is upper triangular. Throughout this chapter, unless otherwise stated, we will assume that $J = \mathrm{diag}(I_p, -I_q)$. Before we prove existence of the hyperbolic QR factorization, we look at one preliminary result.

Lemma 2.3.1 *If A^TJA is positive definite, then $p \geq n$.*

Proof. We give a proof by contradiction. Assume that $p < n$ and let

$$A = \begin{array}{c c} & n \\ p & \left[\begin{array}{c} A_1 \\ \\ A_2 \end{array}\right] \\ q & \end{array}.$$

It is obvious that $\operatorname{rank}(A_1^T A_1) \leq p$. This implies that $A^T JA = A_1^T A_1 - A_2^T A_2$ has at most p nonnegative eigenvalues, since subtracting $A_2^T A_2$ from $A_1^T A_1$ reduces its eigenvalues. However, since $p < n$, $A_1^T A_1 - A_2^T A_2$ cannot be positive definite. Hence $p \geq n$. $\square$

Thus, Lemma 2.3.1 implies that the hyperbolic QR factorization exists only if $p \geq n$.

Theorem 2.3.2 *The hyperbolic QR factorization of a matrix $A \in \mathbb{R}^{m\times n}$ exists for $J = \operatorname{diag}(I_p, -I_q)$, $p + q = m$ and $p \geq n$ if $A^T JA$ is positive definite.*

Proof. Since $A^T JA \in \mathbb{R}^{n\times n}$ is positive definite, we know by the Cholesky factorization (see Section 1.12.1) that there exists a nonsingular matrix $R \in \mathbb{R}^{n\times n}$ such that

$$A^T JA = R^T R.$$

Let $Q_1 = AR^{-1} \in \mathbb{R}^{m\times n}$. Then $A = Q_1 R$ and

$$Q_1^T JQ_1 = R^{-T} A^T JAR^{-1} = R^{-T}(R^T R)R^{-1} = I_n.$$

Note that since we require $Q = [Q_1\ Q_2]$ to be J-orthogonal, $Q_2 \in \mathbb{R}^{m\times(m-n)}$ must be found that satisfies

$$Q_1^T JQ_2 = 0, \tag{2.3.2a}$$

$$Q_2^T JQ_2 = \widetilde{J}, \tag{2.3.2b}$$

where $\widetilde{J} = J(n + 1\colon m, n + 1\colon m)$. Since $Q_1^T J \in \mathbb{R}^{n\times m}$, we can find $m - n$ linearly independent vectors that lie in its null space. Thus, (2.3.2a) implies that we can take for the columns of Q_2 these vectors, which we can transform by $Q_2 \leftarrow Q_2 W$ so that $Q_2^T J Q_2 = J_1$, where $J_1 = \mathrm{diag}(\{1, 0, -1\})$.

Since Q is nonsingular and J is symmetric, we can appeal to Sylvester's law of inertia (see e.g. [34, Theorem 8.1.17]) to show that

$$Q^T J Q = \mathrm{diag}(I_n, J_1) \tag{2.3.3}$$

and J have the same inertia (i.e. the same number of positive, negative and zero eigenvalues). Now $J_1 = \mathrm{diag}(\{1, 0, -1\})$, but J has no zero diagonal elements, and so J_1 also has no zero diagonal elements. Then (2.3.3) implies that J_1 must have $p - n$ ones and q minus ones, just like $\widetilde{J}$.

We have $A = [Q_1 \; Q_2][R^T \; 0]^T$, and so permuting the columns of Q_2 does not affect A. We do this until we can make Q_2 satisfy (2.3.2b). $\square$

Theorem 2.3.2 is a special case of a more general result by Bunse-Gerstner [12] in which A is a complex matrix, $Q \in \mathbb{C}^{m \times m}$ satisfies $QJQ^* = \widehat{J}$, where $\widehat{J}$ is a signature matrix with the same inertia as J, and all leading principle minors of $A^* J A$ are nonsingular.

We note here that just as we use Givens rotations and Householder transforms to compute the R-factor in the QR factorization of a matrix, we will be looking to do something similar here when computing the R-factor in the hyperbolic QR factorization of the matrix A.

We will not be looking at how to use hyperbolic Householder matrices to reduce A to R since, as shown by Tisseur [77, Theorem 4.3], the condition number of the product of a Householder transform with a hyperbolic rotation is equal to or smaller than that of the single hyperbolic Householder transform when we are reducing a vector to a multiple of a unit vector.

2.3.2 The Cholesky Downdating Problem

In this section, we look at the Cholesky downdating problem, and see how it is linked to computing the hyperbolic QR factorization.

Suppose we are given an upper triangular matrix $R_1 \in \mathbb{R}^{n\times n}$ and a vector $x \in \mathbb{R}^n$ such that $R_1^T R_1 - xx^T$ is positive definite. We can then find an upper triangular matrix R that is unique up to the signs of its diagonal elements [34], [42] such that

$$R^T R = R_1^T R_1 - xx^T = \begin{bmatrix} R_1 \\ x^T \end{bmatrix}^T J \begin{bmatrix} R_1 \\ x^T \end{bmatrix},$$

where $J = \operatorname{diag}(I_n, -1)$. This is the Cholesky downdating problem. We have a matrix R_1 which is the computed Cholesky factor of a matrix $A_1^T A_1$, say, where $A_1 \in \mathbb{R}^{p\times n}$, $p \geq n$, and has full rank. We want to find an economical way to perform a rank one downdate of A_1 and recompute the new Cholesky factor, R, which doesn't involve going back to the matrix A_1 and computing R from scratch.

It is obvious that this problem can be generalized so that we are subtracting q vector outer-products, $x_i x_i^T$, $i = 1{:}\,q$, from $R_1^T R_1$.

Let $A_2 := [x_1 \ x_2 \ \cdots \ x_q]^T \in \mathbb{R}^{q\times n}$ and let

$$A := \begin{array}{c} \\ p \\ q \end{array}\!\!\begin{array}{c} n \\ \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \end{array}.$$

Thus, to perform the downdate, we need to compute the matrix R that satisfies

$$\begin{aligned} R^T R &= \begin{bmatrix} R \\ 0 \end{bmatrix}^T J_1 \begin{bmatrix} R \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} R_1 \\ A_2 \end{bmatrix}^T J_1 \begin{bmatrix} R_1 \\ A_2 \end{bmatrix} \\ &= R_1^T R_1 - A_2^T A_2 \\ &= A_1^T A_1 - A_2^T A_2 \\ &= A^T J A, \end{aligned}$$

where $J_1 = \mathrm{diag}(I_n, -I_q)$, $J = \mathrm{diag}(I_p, -I_q)$.

To perform the downdate we need to annihilate A_2 from $[R_1^T \; A_2^T]^T$. During the course of this chapter we will see that this process of annihilating A_2 is precisely what we do to compute the hyperbolic QR factorization of the matrix A.

For more details on the Cholesky downdating problem, see, for example, [8] and [69].

2.3.3 The Hyperbolic QR Factorization using Hyperbolic Rotations

There are various ways in which we can compute R in equation (2.3.1) using hyperbolic rotations. It is obvious that if

$$A = \begin{array}{c c} & n \\ p & \left[\begin{array}{c} A_1 \\ A_2 \end{array}\right. \\ q & \end{array}\!\!\!\left.\begin{array}{c} \\ \\ \end{array}\right], \qquad J = \mathrm{diag}(I_p, -I_q), \qquad p \geq n,$$

then one way of reducing A_1 to upper triangular form is to use orthogonal transformations. We can then use J-orthogonal transformations to annihilate A_2. We concentrate on using hyperbolic rotations in this section. When analyzing the error in applying products of hyperbolic rotations in Section 2.2.4 we concluded that we should use transformations that are *non-overlapping*. We describe one way to do this below.

We now undertake a constructive approach to showing that the hyperbolic QR factorization exists. Clearly, A_1 can be reduced to upper triangular form by computing its QR factorization

$$A_1 = Q_1 \begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

where $Q_1 \in \mathbb{R}^{p \times p}$, $R_1 \in \mathbb{R}^{n \times n}$. Thus we have

$$A = \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} Q_1 & 0 \\ 0 & I_q \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \\ A_2 \end{bmatrix} \begin{array}{l} n \\ p-n \\ q \end{array},$$

which implies that

$$\begin{bmatrix} Q_1^T & 0 \\ 0 & I_q \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} R_1 \\ 0 \\ A_2 \end{bmatrix},$$

since Q_1 is orthogonal. We define

$$Q^{(0)^T} = \begin{bmatrix} Q_1^T & 0 \\ 0 & I_q \end{bmatrix}, \qquad R^{(0)} = R_1, \qquad A^{(0)} = \begin{array}{c} p \\ q \end{array}\begin{bmatrix} A_1^{(0)} \\ A_2^{(0)} \end{bmatrix} = \begin{bmatrix} R_1 \\ 0 \\ A_2 \end{bmatrix}. \tag{2.3.4}$$

Note that, trivially, $Q^{(0)^T}$ is both orthogonal and J-orthogonal.

We now need to annihilate $A_2^{(0)}$. We assume that we have annihilated the first $j-1$ columns of $A_2^{(0)}$ to obtain $A^{(j-1)} = [A_1^{(j-1)^T} \; A_2^{(j-1)^T}]^T$ and describe how to annihilate the jth column of $A_2^{(j-1)}$. Note that

$$\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = Q^{(j-1)} \begin{bmatrix} A_1^{(j-1)} \\ A_2^{(j-1)} \end{bmatrix} = Q^{(j-1)} \begin{bmatrix} R^{(j-1)} \\ 0 \\ A_2^{(j-1)} \end{bmatrix} \begin{array}{l} n \\ p-n \\ q \end{array} \tag{2.3.5}$$

for some J-orthogonal matrix $Q^{(j-1)}$ and upper triangular matrix $R^{(j-1)}$.

Clearly, since $J(p+1\colon m, p+1\colon m) = I_{m-p}$, we can apply a Householder transform, P_j, to $A^{(j-1)}$ to reduce its first column of $A_2^{(j-1)}$ to a single element in the position $A_2^{(j-1)}(1,j)$. Also note that, trivially, since P_j is equal to the identity matrix, I_m, everywhere except in the submatrix $P_j(p+1\colon m, p+1\colon m)$, P_j is J-orthogonal.

Let

$$A^{(j-1)'} = \begin{bmatrix} A_1^{(j-1)'} \\ A_2^{(j-1)'} \end{bmatrix} := P_j \begin{bmatrix} A_1^{(j-1)} \\ A_2^{(j-1)} \end{bmatrix} = P_j A^{(j-1)}.$$

Clearly, there is now only one nonzero element in the jth column of $A_2^{(j-1)'}$. We can annihilate this element by applying a hyperbolic rotation, H_j, to $A^{(j-1)'}$ with the aide of the element $A^{(j-1)'}(j,j)$. So

$$A^{(j)} = \begin{bmatrix} A_1^{(j)} \\ A_2^{(j)} \end{bmatrix} := H_j A^{(j-1)'} = H_j P_j A^{(j-1)}.$$

Since the matrix P_j is involutary and the matrix H_j is J-orthogonal and symmetric, we have

$$A^{(j-1)} = P_j^{-1} H_j^{-1} A^{(j)} = P_j J H_j J A^{(j)}.$$

Thus, using (2.3.5), we also have

$$\begin{aligned}\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} &= Q^{(j-1)} P_j J H_j J \begin{bmatrix} A_1^{(j)} \\ A_2^{(j)} \end{bmatrix} \\ &=: Q^{(j)} \begin{bmatrix} A_1^{(j)} \\ A_2^{(j)} \end{bmatrix} \\ &=: Q^{(j)} \begin{array}{c} \begin{bmatrix} R^{(j)} \\ 0 \\ A_2^{(j)} \end{bmatrix} \end{array} \begin{array}{l} {\scriptstyle n} \\ {\scriptstyle p-n} \\ {\scriptstyle q} \end{array}.\end{aligned}$$

Note that since $Q^{(j-1)}$, P_j and H_j are all J-orthogonal, $Q^{(j)}$ is also J-orthogonal.

We repeat this process for $j = 1\colon n$ to get

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $Q = Q^{(n)}$ and $R = R^{(n)}$.

What now remains to be shown is that the required hyperbolic rotation exists at the jth step. We know that $A^{(j-1)'} = P_j A^{(j-1)}$ satisfies

$$\|A^{(j-1)}(p+1\colon m, j)\|_2 = \|A^{(j-1)'}(p+1\colon m, j)\|_2 = A^{(j-1)'}(p+1, j).$$

To be able to apply the hyperbolic rotation H_j to $A^{(j-1)'}$ we need to show that

$$|A^{(j-1)'}(j,j)| > |A^{(j-1)'}(p+1,j)|.$$

Note that $A^{(j-1)'}(j,j) = A^{(j-1)}(j,j)$, again, due to the construction of P_j. We define $C \in \mathbb{R}^{p\times q}$ by

$$C^T A_1^{(j-1)'} = A_2^{(j-1)'}. \tag{2.3.6}$$

Since

$$\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} = Q^{(j-1)'} \begin{bmatrix} A_1^{(j-1)'} \\ A_2^{(j-1)'} \end{bmatrix}, \qquad A_1^{(j-1)'} = \begin{bmatrix} R^{(j-1)'} \\ 0 \end{bmatrix},$$

for some J-orthogonal matrix $Q^{(j-1)'}$ and upper triangular matrix $R^{(j-1)'}$, we have that

$$\begin{aligned} A^TJA &= \begin{bmatrix} A_1 \\ A_2 \end{bmatrix}^T J \begin{bmatrix} A_1 \\ A_2 \end{bmatrix} \\ &= \begin{bmatrix} A_1^{(j-1)'} \\ A_2^{(j-1)'} \end{bmatrix}^T Q^{(j-1)'^T} J Q^{(j-1)'} \begin{bmatrix} A_1^{(j-1)'} \\ A_2^{(j-1)'} \end{bmatrix} \\ &= \begin{bmatrix} A_1^{(j-1)'} \\ A_2^{(j-1)'} \end{bmatrix}^T J \begin{bmatrix} A_1^{(j-1)'} \\ A_2^{(j-1)'} \end{bmatrix} \\ &= A_1^{(j-1)'^T} A_1^{(j-1)'} - A_2^{(j-1)'^T} A_2^{(j-1)'} \\ &= A_1^{(j-1)'^T} (I_p - CC^T) A_1^{(j-1)'} \\ &= \begin{bmatrix} R^{(j-1)'} \\ 0 \end{bmatrix}^T (I_p - CC^T) \begin{bmatrix} R^{(j-1)'} \\ 0 \end{bmatrix} \\ &= R^{(j-1)'^T} (I_p - CC^T)(1{:}\,n, 1{:}\,n) R^{(j-1)'} \end{aligned}$$

is positive definite. This implies that $I_p - CC^T$ is also positive definite, and this implies that we must have $|c_{ij}| < 1$ for all i and j.

Since $R_1^{(j-1)'}$ is upper triangular we can solve for the first j columns of C in the system (2.3.6). We get $C\,(:, 1{:}\,j-1) = 0$ since $A_2^{(j-1)'}(:, 1{:}\,j-1) = 0$, $C\,(2{:}\,p, j) = 0$ since $A_2^{(j-1)'}(2{:}\,m-p, j) = 0$, and

$$1 > |c_{1j}| = \left| \frac{A^{(j-1)'}(p+1, j)}{A^{(j-1)'}(j, j)} \right|, \qquad (2.3.7)$$

which is what we needed to show to ensure the existence of the required hyperbolic rotations to eliminate the sole remaining element in the jth column of $A_2^{(j-1)}$.

This gives us the following algorithm:

Algorithm 2.3.3 *This algorithm computes the hyperbolic QR factorization* (2.3.1) *of a matrix $A \in \mathbb{R}^{m\times n}$ using non-overlapping transformations by interleaving Householder transforms and hyperbolic rotations.*

Compute the Householder QR factorization $A(1{:}\,p, :) = Q_1[R^T\ 0]^T$

```
        ($Q_1 \in \mathbb{R}^{p\times p}$, $R \in \mathbb{R}^{n\times n}$), overwriting $A(1\colon p, :)$ with $[R^T\ 0]^T$
    $Q = \mathrm{diag}(Q_1, I_q)$
    for $j = 1\colon \min(m-1, n)$
        Construct a Householder transformation $P_j$ such that
            $P_j\, A(p+1\colon m, j) = \sigma_j e_1$.
        $A(p+1\colon m, j\colon n) = P_j\, A(p+1\colon m, j\colon n)$
        $Q(:, p+1\colon m) = Q(:, p+1\colon m)P_j$
        % Eliminate sole remaining subdiagonal element in column $j$ by a
        % hyperbolic rotation.
        $[c, s] = \mathrm{Hrotate}(A(j,j), A(p+1,j))$
        $A([j\ p+1], j\colon n) = \mathrm{Happly}(c, s, A([j\ p+1], j\colon n))$
        $Q(:, [j\ p+1]) = Q(:, [j\ p+1])\left[\begin{smallmatrix} c & s \\ s & c\end{smallmatrix}\right]$
    end
    $R = A(1\colon n, :)$
```

Cost: We give a breakdown of the costs involved in each step of Algorithm 2.3.3 for computing R in Table 2.3.1 and for computing Q in Table 2.3.2. We note that computing the R-factor in the hyperbolic QR factorization of A takes just as long as computing the R-factor in the QR factorization of A since the forming an applying of the hyperbolic rotations only contributes to the lower order terms in the operation count. Also, it is clear from Table 2.3.2 that if we only require the first n columns of the Q-factor in the hyperbolic QR factorization of A, then using Lemma 2.3.1, we can see that we do not need to multiply the P_j, $i = 1\colon n$, to form Q since the P_j only alters its last $m-p$ columns. Also, we only need to compute the first n columns of Q_1, so using Lemma 1.11.1, explicitly forming the first n columns of Q only costs $2n^2(p - n/3)$ flops, a great saving if $m \gg p$ and more so if $p \approx n$. This is summarized in Table 2.3.3.

Table 2.3.1: The number of flops required to compute the upper triangular factor $R \in \mathbb{R}^{n\times n}$ of the hyperbolic QR factorization (2.3.1) of a matrix $A \in \mathbb{R}^{m\times n}$ using non-overlapping transformations by interleaving Householder transforms and hyperbolic rotations as described in Algorithm 2.3.3.

Step of Algorithm 2.3.3	Flops	Reference
Compute the R-factor in the Householder QR factorization $A(1{:}p,:) = Q_1[R^T\ 0]^T$	$2n^2(p-n/3)$	Lemma 1.11.1
Construct the Householder transformation P_j (n times)	$3n(m-p)$	(1.11.6)
$A(p+1{:}m, j{:}n) = P_j\, A(p+1{:}m, j{:}n)$ (for $j = 1{:}n$)	$2n^2(m-p)$	(1.11.7)
$[c,s] = \text{Hrotate}(A(j,j), A(p+1,j))$ (n times)	$5n$	Algorithm 2.2.2
$A(\alpha, j{:}n) = \text{Happly}(c,s,A(\alpha,j{:}n))$, $\alpha = [j\ p+1]$ (for $j = 1{:}n$)	$21n^2/2$	Algorithm 2.2.7
Total	$2n^2(m-n/3)$	

Table 2.3.2: The number of flops required to explicitly compute the J-orthogonal factor $Q \in \mathbb{R}^{m\times m}$ in the hyperbolic QR factorization (2.3.1) of a matrix $A \in \mathbb{R}^{m\times n}$ using non-overlapping transformations by interleaving Householder transforms and hyperbolic rotations as described in Algorithm 2.3.3.

Step of Algorithm 2.3.3	Flops	Reference
Compute the Q-factor in the Householder QR factorization $A(1{:}\,p,:) = Q_1[R^T\ 0]^T$	$4n(p^2 - pn + n^2/3)$	Lemma 1.11.1
$Q(:,p+1{:}\,m) = Q(:,p+1{:}\,m)P_j$ (for $j = 1{:}\,n$)	$4mn(m-p)$	(1.11.7)
$Q(:,[j\ p+1]) = Q(:,[j\ p+1])[\begin{smallmatrix} c & s \\ s & c \end{smallmatrix}]$ (for $j = 1{:}\,n$)	$8mn$	Table 1.8.1
Total	$4n(m^2-mp+p^2-pn+n^2/3)$	

Table 2.3.3: The total number of flops required to compute each factor of the hyperbolic QR factorization (2.3.1) of a matrix $A \in \mathbb{R}^{m\times n}$ using non-overlapping transformations by interleaving Householder transforms and hyperbolic rotations as described in Algorithm 2.3.3.

	R	Q	Q_1
	$2n^2(m-n/3)$	$4n(m^2-mp+p^2-pn+n^2/3)$	$2n^2(m-n/3)$
$m \approx p \approx n$	$4n^3/3$	$4n^3/3$	$4n^3/3$
$m \gg p \approx n$	$2n^2(m-n/3)$	$4n(m^2-mn+n^2/3)$	$4n^3/3$
$m \approx p \gg n$	$2n^2(m-n/3)$	$4n(p^2-pn+n^2/3)$	$2n^2(m-n/3)$
$m \gg p \gg n$	$2mn^2$	$4m^2n$	$2mn^2$

2.3.4 Error Analysis

We now give a rounding error analysis of Algorithm 2.3.3. First, we note that from (2.3.1) we have $R^TR = A_1^TA_1 - A_2^TA_2$. Hence if A_1 is upper trapezoidal then the hyperbolic QR factor R is the result of (block) downdating a Cholesky factorization. Various algorithms, both hyperbolic and non-hyperbolic, are known for downdating Cholesky factorizations, and error analysis is available; see, for example, [8], [11], [26], [27], [70], [74]. While we could invoke some of the earlier results in that part of the analysis that does not involve the right hand side, b, we have chosen to give an independent development, aiming to make clear how the various errors combine and provide building blocks that should be of use in future analyses. In particular, we emphasize the high-level features of the analysis and thereby provide new insight into what is required of a sequence of hyperbolic transformations in order for satisfactory error bounds to be obtainable.

As before, we partition

$$A = \begin{array}{c c} & n \\ p & \left[\begin{array}{c} A_1 \\ \\ A_2 \end{array}\right. \\ q & \end{array}\!\!\!\left.\vphantom{\begin{array}{c} A_1 \\ \\ A_2 \end{array}}\right].$$

The first stage of Algorithm 2.3.3 computes the Householder QR factorization $A_1 = Q_1\widetilde{U}_1$, where $\widetilde{U}_1 = [\widetilde{R}_1^T\ 0]^T \in \mathbb{R}^{p\times n}$ is upper trapezoidal. Using Lemma 1.11.7, we can see that the computed $\widetilde{U}_1$ is the exact factor of $A_1 + \mathit{\Delta}_1$, with $\|\mathit{\Delta}_1\|_F \le \widetilde{\gamma}_{pn}\|A_1\|_F$. To simplify the notation, we will assume for the moment that A_1 is already in upper trapezoidal form and will introduce the error $\mathit{\Delta}_1$ at the end.

Consider the jth column of A,

$$A(:,j) = \begin{bmatrix} A_1(:,j) \\ \\ A_2(:,j) \end{bmatrix} = \begin{array}{c} p \\ \\ q \end{array}\begin{bmatrix} a_{1,j}^{(0)} \\ \\ a_{2,j}^{(0)} \end{bmatrix} = a_j^{(0)}.$$

It undergoes n Householder transformations in the last q components, intertwined with n hyperbolic rotations in the planes $(1, p+1), \dots, (n, p+1)$; the jth pair of these transformations introduce the required zeros in this column. Note that the final $n-j$ pairs of transformations leave the first j columns unchanged since we have already annihilated those columns.

Consider the first Householder transformation, P_1, and the subsequent hyperbolic rotation, H_1. From (1.11.2a), we know that P_1 agrees with the identity in rows and columns $1\colon p$ and so, using Lemma 1.11.5, we can see that it satisfies (2.2.26a) and (2.2.23) with $\varDelta S_1 = 0$ and $\mu = \widetilde{\gamma}_q$. On the other hand, H_1 agrees with the identity in rows and columns $2\colon p$, satisfies the bound of Lemma 2.2.8 and is non-overlapping with P_1 for all values of t in the range $1\colon p$ in (2.2.21). Note that since H_1 does not alter rows $2\colon p$ of the vector we apply it to, we lose no accuracy in applying H_1 to the vector. We choose $t=1$. Applying Lemma 2.2.10 to these two transformations gives us

$$H_1P_1\left(\begin{bmatrix} a_{1,j}^{(0)} \\ a_{2,j}^{(0)} \end{bmatrix} + \begin{bmatrix} 0 \\ \varDelta a_{2,j}^{(0)} \end{bmatrix}\right) = \begin{bmatrix} a_{1,j}^{(1)} \\ a_{2,j}^{(1)} \end{bmatrix} + \begin{bmatrix} \varDelta a_{1,j}^{(1)'} \\ 0 \end{bmatrix},$$

where

$$\max(\|\varDelta a_{2,j}^{(0)}\|_2, |\varDelta a_{1,j}^{(1)'}(1)|) \le \widetilde{\gamma}_q \max(\|a_{2,j}^{(0)}\|_2, |a_{1,j}^{(1)}(1)|).$$

In fact, for any pair H_kP_k of the Householder transform P_k and hyperbolic rotation H_k applied to the vector $[\alpha_1^T \ \alpha_2^T]^T$ to obtain $[\beta_1^T \ \beta_2^T]^T$, Lemma 2.2.10 gives us

$$H_kP_k\left(\begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \varDelta\alpha_2 \end{bmatrix}\right) = \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} + \begin{bmatrix} \varDelta\beta_1 \\ 0 \end{bmatrix},$$

where

$$\max(\|\varDelta\alpha_2\|_2, |\varDelta\beta_1(k)|) \le \widetilde{\gamma}_q \max(\|\alpha_2\|_2, |\beta_1(k)|)$$

by choosing $t=k$ in (2.2.21), since P_k does not alter α_1 and H_k does not alter $\alpha_1([1\colon k-1, k+1\colon p])$.

Now, the product H_kP_k agrees with the identity in rows and columns $1\colon k-1$ and $k+1\colon p$, and so all H_kP_k, $k = 1\colon n$, are non-overlapping with each other. Hence we can inductively apply Lemma 2.2.10 to obtain

$$H_nP_n\dots H_1P_1\left(\begin{bmatrix} a_{1,j}^{(0)} \\ a_{2,j}^{(0)} \end{bmatrix} + \begin{bmatrix} 0 \\ \varDelta' a_{2,j}^{(0)} \end{bmatrix}\right) = \begin{bmatrix} a_{1,j}^{(j)} \\ a_{2,j}^{(j)} \end{bmatrix} + \begin{bmatrix} \varDelta a_{1,j}^{(j)'} \\ 0 \end{bmatrix},$$

where

$$\max(\|\Delta' a_{2,j}^{(0)}\|_2, \|\Delta a_{1,j}^{(j)'}(1\colon j)\|_2) \le \widetilde{\gamma}_{qj} \max(\|a_{2,j}^{(0)}\|_2, \|a_{1,j}^{(j)}(1\colon j)\|_2),$$

since only j pairs of Householder transforms and hyperbolic rotations are applied to $a_j^{(0)}$, and $a_{1,j}^{(l)}(j{+}1\colon n) = 0$, $l = 1\colon n$. Also, when applying $H_k P_k$, we set $t = k-1$.

The overall finding relating the jth columns of A and the final upper triangular factor $[R^T\ 0]^T$ is that

$$\begin{array}{c} n \\ p-n \\ q \end{array}\begin{bmatrix} \widehat{r}_j \\ 0 \\ a_j^{(2)} \end{bmatrix} + h_j = G \begin{bmatrix} a_j^{(1)} \\ 0 \end{bmatrix}\begin{array}{c} p \\ q \end{array}, \qquad \|h_j\|_2 \le \widetilde{\gamma}_{qj} \max(\|\widehat{r}_j(1\colon j)\|_2, \|a_j^{(2)}\|_2), \quad (2.3.8)$$

where $\widehat{r}_j = \widehat{R}(:,j)$, for some exactly orthogonal G that is *independent of* j. Importantly, $h_j(n{+}1\colon p) = 0$, because after the initial hyperbolic QR factorization rows $n+1\colon p$ of A rest untouched. Putting these equations together for $j = 1\colon n$ and incorporating the error from the initial QR factorization of A_1 gives

$$\begin{array}{c} n \\ p-n \\ q \end{array}\begin{bmatrix} \widehat{R}_1 + \Delta_3 \\ 0 \\ A_2 + \Delta_2 \end{bmatrix} = G \begin{bmatrix} A_1 + \Delta_1 \\ 0 \end{bmatrix}\begin{array}{c} p \\ q \end{array}, \qquad (2.3.9)$$

where, using Lemma 1.11.7 and (2.3.8),

$$\|\Delta_1\|_F \le \widetilde{\gamma}_{pn} \|A_1\|_F, \qquad (2.3.10a)$$

$$\|\Delta_i\|_F \le \widetilde{\gamma}_{qn} \max(\|\widehat{R}\|_F, \|A_2\|_F) \le \widetilde{\gamma}_{qn} \|A_1\|_F, \quad i = 2\colon 3. \qquad (2.3.10b)$$

Certainly, $\max_i \|\Delta_i\|_F \le \widetilde{\gamma}_{mn} \|A\|_F$.

Note that in view of the equivalence (2.1.3) and (2.1.4), as long as G has a nonsingular (1,1) block this result is equivalent to

$$\begin{array}{c} p \\ q \end{array}\begin{bmatrix} A_1 + \Delta_1 \\ A_2 + \Delta_2 \end{bmatrix} = Q \begin{bmatrix} \widehat{R} + \Delta_3 \\ 0 \end{bmatrix}\begin{array}{c} n \\ m-n \end{array}, \qquad (2.3.11)$$

for a J-orthogonal Q. Both (2.3.9) and (2.3.11) are mixed backward-forward error results, because both the original data A and the upper triangular factor

R are perturbed. We can obtain a genuine backward error result with the aid of the following lemma [71, pp. 302–304].

Lemma 2.3.4 *Let $m = p + q$ and $n \geq p$. Given a full rank matrix $A \in \mathbb{R}^{p\times n}$ and $E \in \mathbb{R}^{q\times n}$ there exists an orthogonal $Q \in \mathbb{R}^{m\times m}$ such that*

$$Q\begin{bmatrix} A \\ E \end{bmatrix} = \begin{bmatrix} A+F \\ 0 \end{bmatrix}, \tag{2.3.12}$$

where, for small $\|E\|_2$,

$$\|F\|_2 \leq \frac{\|E\|_2^2}{2\sigma_{\min}(A)} + O(\|E\|_2^4).$$

Proof. Let $A = UH$ be a polar decomposition, where $U \in \mathbb{R}^{p\times n}$ has orthonormal columns and $H \in \mathbb{R}^{n\times n}$ is symmetric positive definite. Then

$$\begin{aligned} A^TA + E^TE &= H^2 + E^TE \\ &= H(I + \widetilde{E}^T\widetilde{E})H, \qquad \widetilde{E} = EH^{-1} \\ &= H(I+G)\cdot(I+G)H, \end{aligned} \tag{2.3.13}$$

where the symmetric positive definite matrix G satisfies $(I+G)^2 = I + \widetilde{E}^T\widetilde{E}$. The latter equation is $2G = \widetilde{E}^T\widetilde{E} - G^2$, which implies

$$G = \frac{\widetilde{E}^T\widetilde{E}}{2} + O(\|\widetilde{E}\|_2^4).$$

Now (2.3.13) implies that for some orthogonal $Z \in \mathbb{R}^{m\times m}$

$$Z\begin{bmatrix} A \\ E \end{bmatrix} = \mathrm{diag}(U, I)\begin{bmatrix} (I+G)H \\ 0 \end{bmatrix} = \begin{bmatrix} UH \\ 0 \end{bmatrix} + \begin{bmatrix} UGH \\ 0 \end{bmatrix}.$$

So (2.3.12) holds with $F = UGH$ and hence

$$\begin{aligned} \|F\|_2 \leq \|GH\|_2 &= \frac{1}{2}\|\widetilde{E}^T\widetilde{E}H\|_2 + O(\|\widetilde{E}\|_2^4) = \frac{1}{2}\|H^{-1}E^TE\|_2 + O(\|E\|_2^4) \\ &\leq \frac{\|E\|_2^2}{2\sigma_{\min}(A)} + O(\|E\|_2^4). \qquad \square \end{aligned}$$

Rewriting (2.3.9) as

$$\begin{array}{c} n \\ p-n \\ q \end{array}\begin{bmatrix} \widehat{R} \\ 0 \\ A_2+\varDelta_2 \end{bmatrix} = G\begin{bmatrix} A_1+\varDelta_1+\widetilde{\varDelta}_1 \\ \widetilde{\varDelta}_2 \end{bmatrix}\begin{array}{c} p \\ q \end{array}, \qquad \widetilde{\varDelta} = -G^T\begin{bmatrix} \varDelta_3 \\ 0 \end{bmatrix}\begin{array}{c} n \\ m-n \end{array}$$

and applying Lemma 2.3.4 to the right-hand side leads to the conclusion that

$$\begin{bmatrix} \widehat{R} \\ 0 \\ A_2+\varDelta_2 \end{bmatrix} = \widetilde{G}\begin{bmatrix} A_1+\overline{\varDelta}_1 \\ 0 \end{bmatrix},$$

where $\widetilde{G}$ is orthogonal and

$$\begin{aligned} \|\varDelta_2\|_F &\le \widetilde{\gamma}_{mn}\,\|A\|_F\,, \\ \|\overline{\varDelta}_1\|_F &\le \frac{\widetilde{\gamma}_{mn}^2\,\|A_1\|_F^2}{2\sigma_{\min}(A_1+\varDelta_1+\widetilde{\varDelta}_1)} + O(u^4) \\ &\le \frac{\sqrt{n}}{2}(\kappa_2(A_1)\widetilde{\gamma}_{mn})\widetilde{\gamma}_{mn}\,\|A\|_F + O(u^3). \end{aligned}$$

We conclude that backward stability of the factorization is guaranteed if $\kappa_2(A_1)u$ is of order 1. Thus the factorization is only conditionally backward stable, although the condition is quite weak.

2.3.5 Numerical Results

We generated test problems to see how accurately we could compute the hyperbolic QR factorization. Details of how the test problems were generated can be found in Section 4.3.

In Table 2.3.4, we give some results where we compare two methods for computing the hyperbolic QR factorization of a matrix $A \in \mathbb{R}^{m\times n}$. The results show that if Q is large in norm then using Algorithm 2.3.3 to compute it does not guarantee it to be numerically J-orthogonal (i.e. $\|Q^TJQ\|_2 \approx u\|Q\|_2^2$), however, it is always almost numerically J-orthogonal when the hyperbolic QR factorization is computed by annihilating $A(p+1\colon m,:)$ using only hyperbolic rotations.

This is surprising since we could not prove stability of applying arbitrary products of hyperbolic rotations in Section 2.2.4, yet it seems as though we do not need to worry about how to apply it. We have no explanation for these results other than it may be possible to prove stability of applying arbitrary products of hyperbolic rotations in a way that we have not yet looked at. In the tests, we computed Q explicitly before applying it, as oppose to leaving it in factored form and applying the individual factors turn by turn. The accuracy of the computed value of Q is further put into doubt by examining the values of $\|A - Q[R^T\ 0]^T\|_2/(\|A\|_2 + \|Q\|_2\,\|R\|_2)$ for both methods, since it also increases with the norm of Q. The upper triangular factor R is shown to always be computed accurately.

Table 2.3.5 compares the computation of the hyperbolic QR factorization using hyperbolic rotations with representations HR2 in (2.2.6), HR5 in (2.2.9) and HR6 in (2.2.14). The results were somewhat surprising considering that we could not prove stability for representations HR2 and HR5. As we can see in the table, for problems with large $\|Q\|_2$, HR5 was the best overall representation since its values of $\|A - Q[R^T\ 0]^T\|_2/(\|A\|_2 + \|Q\|_2\,\|R\|_2)$ were the smallest. HR2 fared much better than anticipated, since the results it produced were comparable to those for HR6.

For the results in both Table 2.3.4 and Table 2.3.5, details of how Q and R were computed are given in Section 4.3.

2.3.6 A More General Hyperbolic QR Factorization

A more general hyperbolic QR factorization requiring milder assumptions on it than those for the factorization (2.3.1) was proposed by Singer and Singer [65] in 2000. We present their main result below.

Lemma 2.3.5 *Let $A \in \mathbb{C}^{m\times n}$, $m \geq n$, and $J = \operatorname{diag}(j_{11}, j_{22}, \ldots, j_{mm})$, $J_{ii} \in$*

Table 2.3.4: Errors in computing the hyperbolic QR factorization $A = Q[R^T\ 0]^T = Q_1R$ of a matrix $A \in \mathbb{R}^{m\times n}$ for a given $\|Q\|_2$ and $\|R\|_2$. The method of non-overlapping Householder transforms and hyperbolic rotations with representation HR6 of Algorithm 2.3.3 is compared with a method where the whole of $A(p+1\colon m, :)$ is annihilated using only hyperbolic rotations to obtain $A = Q_{\mathrm{hr}}[R_{\mathrm{hr}}^T\ 0]^T$.

$\|Q\|_2$	$\|R\|_2$	$\frac{\|A-Q_1R\|_2}{\|A\|_2+\|Q\|_2\|R\|_2}$	$\frac{\|Q^TJQ-J\|_2}{\|Q\|_2^2}$	$\frac{\|A^TJA-R^TR\|_2}{\|A\|_2^2}$	$\frac{\|A-Q_{\mathrm{hr}}[R_{\mathrm{hr}}^T\ 0]^T\|_2}{\|A\|_2+\|Q_{\mathrm{hr}}\|_2\|R_{\mathrm{hr}}\|_2}$	$\frac{\|Q_{\mathrm{hr}}^TJQ_{\mathrm{hr}}-J\|_2}{\|Q_{\mathrm{hr}}\|_2^2}$	$\frac{\|A^TJA-R_{\mathrm{hr}}^TR_{\mathrm{hr}}\|_2}{\|A\|_2^2}$
1.0e+00	1.0e+00	2.0e-16	5.3e-16	4.8e-16	2.0e-16	5.3e-16	4.8e-16
1.0e+00	1.0e+04	7.4e-17	7.9e-16	9.6e-17	7.4e-17	7.9e-16	9.6e-17
1.0e+00	1.0e+08	1.5e-16	5.2e-16	2.2e-16	1.5e-16	5.2e-16	2.2e-16
9.4e+01	1.0e+04	5.0e-15	8.3e-16	9.6e-17	9.9e-16	1.9e-16	4.4e-17
9.4e+01	1.0e+10	7.8e-15	2.1e-15	2.1e-16	9.4e-16	5.2e-16	1.2e-16
9.4e+03	1.0e+02	8.4e-11	1.1e-13	1.5e-16	8.0e-12	5.7e-15	1.3e-16
9.4e+03	1.0e+06	1.1e-10	1.1e-13	1.6e-16	2.9e-11	6.7e-15	6.9e-17
9.4e+05	1.0e+00	4.6e-06	2.0e-11	2.4e-16	4.0e-07	5.4e-15	7.6e-16
9.3e+05	1.0e+04	2.9e-07	1.6e-11	1.6e-16	3.5e-07	7.0e-15	1.4e-16
6.6e+07	1.2e+00	1.5e-02	2.2e-09	1.8e-16	9.4e-04	1.1e-14	1.6e-16

Table 2.3.5: Errors in computing the hyperbolic QR factorization of a matrix $A \in \mathbb{R}^{m\times n}$ for a given $\|Q\|_2$ and $\|R\|_2$. The method of non-overlapping Householder transforms and hyperbolic rotations of Section 2.3 is tested using hyperbolic rotations with representations HR2, HR5 and HR6.

$\|Q\|_2$	$\|R\|_2$	Representation	$\frac{\|A-QR\|_2}{\|A\|_2+\|Q\|_2\|R\|_2}$	$\frac{\|Q^TJQ-J\|_2}{\|Q\|_2^2}$	$\frac{\|A^TJA-R^TR\|_2}{\|A\|_2^2}$
1.0e+00	1.0e+00	HR2	2.0e-16	5.3e-16	4.8e-16
		HR5	2.0e-16	5.3e-16	4.8e-16
		HR6	2.0e-16	5.3e-16	4.8e-16
1.0e+00	1.0e+04	HR2	7.4e-17	7.9e-16	9.6e-17
		HR5	7.4e-17	7.9e-16	9.6e-17
		HR6	7.4e-17	7.9e-16	9.6e-17
1.0e+00	1.0e+08	HR2	1.5e-16	5.2e-16	2.2e-16
		HR5	1.5e-16	5.2e-16	2.2e-16
		HR6	1.5e-16	5.2e-16	2.2e-16
9.4e+01	1.0e+04	HR2	1.2e-15	1.4e-15	7.6e-17
		HR5	1.7e-16	1.4e-15	7.6e-17
		HR6	5.0e-15	8.3e-16	9.6e-17
9.4e+03	1.0e+06	HR2	1.9e-10	1.4e-13	1.3e-16
		HR5	9.1e-15	9.4e-14	9.1e-17
		HR6	1.1e-10	1.1e-13	1.6e-16
9.3e+05	1.0e+04	HR2	2.1e-07	1.4e-11	1.7e-16
		HR5	2.0e-12	9.6e-12	1.7e-16
		HR6	2.9e-07	1.6e-11	1.6e-16
6.6e+07	1.2e+00	HR2	6.8e-03	2.0e-09	1.7e-16
		HR5	6.4e-10	7.0e-10	1.3e-16
		HR6	1.5e-02	2.2e-09	1.8e-16

$\{-1,1\}$ *be given. If the matrix* $B = A^*JA$ *is nonsingular, then* A *can be factorized as*

$$A = P_1QRP_2^*, \qquad Q^*J_1Q = J_1, \qquad J_1 = P_1^*JP_1, \qquad R = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} \in \mathbb{R}^{m\times n},$$

where P_1 *and* P_2 *are permutation matrices,* R_1 *is block upper triangular of order* n *with diagonal blocks of order 1 or 2 and* Q *is* J_1*-unitary.* $\square$

By comparing the algorithm by Singer and Singer with Algorithm 2.3.3, it can easily be seen that the two algorithms are very different. We list these differences below.

- The main difference is that the matrix R_1 in Theorem 2.3.5 is *block* upper triangular. Hence we cannot always be guaranteed of being able to compute the hyperbolic QR factorizationas described in Section 2.3.
- The permutation matrices P_1 and P_2 imply that there are some row and column changes that take place during the course of factorizing the matrix A. Also, due to these permutations, the diagonal elements of the signature matrix J are permuted as well. We could view this factorization as being the hyperbolic QR factorization of the matrix A with *complete pivoting.*
- In the algorithm described by Singer and Singer [65], Givens rotations have been used instead of Householder transforms. This is because of the number of times that pivoting may be needed to be carried out during the course of the algorithm. Since the hyperbolic QR factorization algorithm in Section 2.3 does not require any pivoting, we can make use of Householder transforms. Note that Singer and Singer's hyperbolic QR factorization only requires A^TJA to be nonsingular, whereas the hyperbolic QR factorization in Section 2.3 requires A^TJA to be positive definite.

- Finally, in their algorithm, Singer and Singer [65] annihilate the subdiagonal of a column of A at a time rather than firstly performing a QR factorization of $A(1\colon p, :)$ as in Algorithm 2.3.3.

Chapter 3

The Indefinite Least Squares Problem

3.1 Introduction

A variant of the *least squares* (LS) problem,

$$\text{LS} \ : \qquad \min_x \left(b - Ax\right)^T \left(b - Ax\right), \tag{3.1.1}$$

is the *indefinite least squares* (ILS) problem,

$$\text{ILS} \ : \qquad \min_x \left(b - Ax\right)^T J \left(b - Ax\right), \tag{3.1.2}$$

where $A \in \mathbb{R}^{m\times n}$, $m \geq n$, and the signature matrix

$$J = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix} \qquad p + q = m. \tag{3.1.3}$$

For $p = 0$ or $q = 0$ we have the standard LS problem and the quadratic form that we are trying to minimize is definite, while for $pq > 0$ the problem is to minimize a genuinely indefinite quadratic form.

We usually associate LS with fitting polynomials (or any other function that depends linearly on its parameters) to a set of data points to obtain a best fit.

However, to obtain the ILS problem in normal data fitting would require us to attach weights of 1 or -1 to all the data points. This means that in fitting the polynomial through the data, we want to be as close as possible to some of the points but, at the same time, be as far away as possible from other points. If a data point was not needed, then it would make more sense to attach the weight 0 to it, so it is not clear how to interpret (3.1.2) in the case of normal data fitting. We will look at other applications of the ILS problem in Section 1.2.

Differentiating (3.1.2) with respect to x gives us $-2A^TJ(b-Ax)$, hence the first order conditions for optimality, also known as the normal equations, for the ILS problem are

$$A^TJ(b-Ax)=0. \tag{3.1.4}$$

The Hessian matrix for (3.1.2) is $2A^TJA$, so in order to get a unique solution to (3.1.2) a necessary and sufficient condition is that

$$A^TJA \text{ is positive definite.} \tag{3.1.5}$$

We will assume from now on that (3.1.5) holds.

Note that if (3.1.5) holds then $p \geq n$ by Lemma 2.3.1, and this implies that A_1 (and hence A) has full rank. For a genuinely ILS problem we therefore need $m > n$.

Defining r to be the residual vector

$$r := b - Ax,$$

we can rewrite the normal equations (3.1.4) as the augmented system

$$\begin{bmatrix} I & A \\ A^TJ & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix},$$

which can be rewritten with a symmetric coefficient matrix as

$$\begin{bmatrix} J & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} s \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{3.1.6}$$

where $s = Jr$.

3.2 Perturbation Theory

3.2.1 Normwise Perturbation Bound

In this section we derive normwise perturbation bounds for the solution x and a residual r of the ILS problem. Our approach is based on that used by Cox and Higham [20] to obtain perturbation bounds for the equality constrained least squares problem. We let $\widetilde{x}$ be the solution of the perturbed ILS problem

$$\min_x (b + \Delta b - (A + \Delta A)x)^T J (b + \Delta b - (A + \Delta A)x) \tag{3.2.1}$$

and define

$$\widetilde{r} = b + \Delta b - (A + \Delta A)\widetilde{x}, \qquad r = b - Ax$$

to be the residuals of the perturbed and unperturbed problems, respectively. We assume that $A + \Delta A$ satisfies the uniqueness condition (3.1.5), which will always be the case for ΔA sufficiently small in norm. The perturbations to the data will be measured by the smallest ϵ for which

$$\|\Delta A\|_F \le \epsilon \|\mathbf{A}\|_F, \quad \|\Delta b\|_2 \le \epsilon \|\mathbf{b}\|_2, \tag{3.2.2}$$

where $\mathbf{A}$ and $\mathbf{b}$ are a matrix and vector of tolerances.

The perturbed augmented system corresponding to (3.1.6) is

$$\begin{bmatrix} J & A + \Delta A \\ (A + \Delta A)^T & 0 \end{bmatrix} \begin{bmatrix} \widetilde{s} \\ \widetilde{x} \end{bmatrix} = \begin{bmatrix} b + \Delta b \\ 0 \end{bmatrix}, \tag{3.2.3}$$

where $\widetilde{s} = J\widetilde{r}$. Writing

$$\widetilde{s} = s + \Delta s, \quad \widetilde{x} = x + \Delta x$$

and subtracting (3.1.6) from (3.2.3) we obtain

$$\begin{bmatrix} J & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \Delta s \\ \Delta x \end{bmatrix} = \begin{bmatrix} \Delta b - \Delta A \widetilde{x} \\ -\Delta A^T \widetilde{s} \end{bmatrix}. \tag{3.2.4}$$

It is straightforward to verify that the inverse of the matrix on the left-hand side of (3.2.4) is

$$\begin{bmatrix} J - JAM^{-1}A^TJ & JAM^{-1} \\ M^{-1}A^TJ & -M^{-1} \end{bmatrix}, \qquad \text{where } M = A^TJA.$$

Premultiplying by the inverse and expanding the right-hand side, we obtain

$$\mathit{\Delta} s = (J - JAM^{-1}A^TJ)(\mathit{\Delta} b - \mathit{\Delta} A\widetilde{x}) - (JAM^{-1})\mathit{\Delta} A^T\widetilde{s}, \tag{3.2.5a}$$

$$\mathit{\Delta} x = M^{-1}A^TJ(\mathit{\Delta} b - \mathit{\Delta} A\widetilde{x}) + M^{-1}\mathit{\Delta} A^T\widetilde{s}. \tag{3.2.5b}$$

If we put $J = I$ then we recover perturbation expressions for the standard LS problem.

Since the perturbations $\mathit{\Delta} s$ and $\mathit{\Delta} x$ are of order ϵ, we can substitute $s = Jr$ and x for their perturbed counterparts to obtain first order expressions. Then, taking norms, we deduce

$$\begin{aligned} \|\mathit{\Delta} r\|_2 &\le \epsilon\Big[\|I - JAM^{-1}A^T\|_2(\|\mathbf{b}\|_2 + \|\mathbf{A}\|_F\,\|x\|_2) + \|AM^{-1}\|_2\,\|\mathbf{A}\|_F\,\|r\|_2\Big], \\ &\quad + O(\epsilon^2) \\ \|\mathit{\Delta} x\|_2 &\le \epsilon\Big[\|M^{-1}A^T\|_2(\|\mathbf{b}\|_2 + \|\mathbf{A}\|_F\,\|x\|_2) + \|M^{-1}\|_2\,\|\mathbf{A}\|_F\,\|r\|_2\Big] \\ &\quad + O(\epsilon^2). \end{aligned} \tag{3.2.6}$$

Hence, provided $x \neq 0$,

$$\begin{aligned} \frac{\|\mathit{\Delta} x\|_2}{\|x\|_2} \le \epsilon\Bigg[&\|M^{-1}A^T\|_2\,\|\mathbf{A}\|_F\left(\frac{\|\mathbf{b}\|_2}{\|\mathbf{A}\|_F\,\|x\|_2} + 1\right) \\ &+ \|M^{-1}\|_2\,\|A\|_F^2\,\frac{\|\mathbf{A}\|_F}{\|A\|_F}\frac{\|r\|_2}{\|A\|_F\,\|x\|_2}\Bigg] + O(\epsilon^2). \end{aligned} \tag{3.2.7}$$

This bound shows that the sensitivity of the ILS problem is bounded in terms of $\|M^{-1}A^T\|_2\,\|\mathbf{A}\|_F$ when the residual is zero or small and $\|M^{-1}\|_2\,\|A\|_F^2$ otherwise; note that for $\mathbf{A} = A$ the former quantity is no larger than the latter and is potentially much smaller.

Now we examine whether (3.2.7) is attainable for some $\mathit{\Delta} A$ and $\mathit{\Delta} b$. The three terms in brackets in (3.2.6) are

$$E_1 = \|M^{-1}A^T\|_2 \,\|\mathbf{b}\|_2\,, \tag{3.2.8a}$$

$$E_2 = \|M^{-1}A^T\|_2 \,\|\mathbf{A}\|_F \,\|x\|_2\,, \tag{3.2.8b}$$

$$E_3 = \|M^{-1}\|_2 \,\|\mathbf{A}\|_F \,\|r\|_2\,, \tag{3.2.8c}$$

and from (3.2.5b) we can see that they result from the perturbations $\mathit{\Delta} b$, $\mathit{\Delta} A$ and $\mathit{\Delta} A^T$, respectively. It follows that the bound (3.2.6) can fail to be achieved for some $\mathit{\Delta} b$ and $\mathit{\Delta} A$ only if $E_1 < E_2 \approx E_3$ and there is substantial cancellation in the expression $-M^{-1}A^TJ\mathit{\Delta} Ax + M^{-1}\mathit{\Delta} A^TJr$ for all $\mathit{\Delta} A$. We can show in various special cases that these circumstances cannot arise (for example when r is small, or when $|r^TJr| \approx \|r\|_2^2$), but we have been unable to establish attainability of the bound (3.2.7) in general.

A natural definition of the condition number of the ILS problem is

$$\kappa_{\mathrm{ILS}}(A,b) = \lim_{\epsilon\to 0}\sup\left\{\frac{\|x-\widetilde{x}\|_2}{\|x\|_2} : (3.2.2)\text{–}(3.2.3) \text{ hold}\right\}. \tag{3.2.9}$$

Without a guarantee of sharpness, the bound (3.2.7) does not provide an estimate of $\kappa_{\mathrm{ILS}}(A,b)$ to within a readily identifiable constant factor. Therefore we take a different approach in which we combine the two $\mathit{\Delta} A$ terms in (3.2.5b) before taking norms. To do this, we use the vec operator together with the Kronecker product which we introduced in Section 1.5. Applying the vec operator to (3.2.5b) and using the relation $\mathrm{vec}(AXB) = (B^T \otimes A)\mathrm{vec}(X)$,Lemma 1.5.1 we obtain

$$\mathit{\Delta} x = M^{-1}A^TJ\mathit{\Delta} b - (x^T \otimes M^{-1}A^TJ)\mathrm{vec}(\mathit{\Delta} A) + (r^TJ \otimes M^{-1})\mathrm{vec}(\mathit{\Delta} A^T) + O(\epsilon^2).$$

Using Lemma 1.5.2 gives

$$\mathit{\Delta} x = M^{-1}A^TJ\mathit{\Delta} b - [(x^T \otimes M^{-1}A^TJ) - (r^TJ \otimes M^{-1})\mathit{\Pi}]\mathrm{vec}(\mathit{\Delta} A) + O(\epsilon^2).$$

Now we take 2-norms. Using (3.2.2) and (1.5.1), we deduce that

$$\frac{\|\mathit{\Delta} x\|_2}{\|x\|_2} \le \psi\epsilon + O(\epsilon^2), \tag{3.2.10}$$

where

$$\psi = (\|M^{-1}A^T\|_2 \, \|\mathbf{b}\|_2 + \|(x^T \otimes M^{-1}A^T J) - (r^T J \otimes M^{-1})\Pi\|_2 \, \|\mathbf{A}\|_F) / \, \|x\|_2$$

and we have

$$\kappa_{\mathrm{ILS}}(A, b) \leq \psi \leq 2\kappa_{\mathrm{ILS}}(A, b).$$

3.2.2 Numerical Comparison

We generated test problems to compare the first order terms of the bounds (3.2.7) and (3.2.10). Details of how the test problems were generated can be found in Section 4.3. In the tests, the terms were found to always be within a small factor of each other. We believe that (3.2.7) is nearly attainable and, because this bound is much easier to work with than (3.2.10), we will use it when we investigate the stability of solving the ILS problem.

We summarize our findings in Table 3.2.1. We have compared the values of (3.2.7) and (3.2.10) and given the corresponding values of the terms E_1, E_2 and E_3 in (3.2.8). We can clearly see in the table that all values of the bound in (3.2.7) are very similar to the values of the bound in (3.2.10), even when $E_1 < E_2 \approx E_3$. For further details of how the values of Q, R and b were computed, and how the exact value of x was found for the particular problem, see Section 4.3.

3.2.3 Componentwise Perturbation Bound

To end this section, we note that we can also use (3.2.5) to obtain componentwise perturbation bounds for the ILS problem. For the solution, we obtain

$$|\Delta x| \leq \epsilon |M^{-1}A^T|(\mathbf{b} + \mathbf{A}|x|) + |M^{-1}|\mathbf{A}^T|r| + O(\epsilon^2), \tag{3.2.11}$$

where inequalities and the absolute value are interpreted componentwise and ϵ has been redefined as the smallest value for which $|\Delta A| \leq \epsilon\mathbf{A}$, $|\Delta b| \leq \epsilon\mathbf{b}$, where

Table 3.2.1: Comparison of the two perturbation bounds in (3.2.7) and (3.2.10) for various values of $\|Q\|_2$ and $\|R\|_2$, where Q and R are the factors in the hyperbolic QR factorization (2.3.1) of A in the ILS problem. Also given are the corresponding values of the terms E_1, E_2 and E_3 in (3.2.8) with $\|\mathbf{A}\|_F = \|A\|_F$ and $\|\mathbf{b}\|_2 = \|b\|_2$.

$\|Q\|_2$	$\|R\|_2$	r	(3.2.7)	E_1	E_2	E_3	(3.2.10)
1.0e+00	1.0e+00	4.9e-17	4.3e-16	1.5e-16	4.4e-16	2.1e-32	4.3e-16
1.0e+00	1.0e+00	1.5e+00	9.8e-16	1.5e-16	2.4e-16	3.6e-16	7.6e-16
1.0e+00	1.0e+08	0	1.4e-08	4.4e-09	1.5e-08	0	1.4e-08
1.0e+00	1.0e+08	1.1e-07	1.3e-07	4.4e-09	3.3e-02	3.6e-01	1.2e-07
9.4e+01	1.0e+04	2.2e-14	2.8e-09	8.9e-10	3.0e-09	5.6e-19	2.8e-09
9.4e+01	1.0e+04	2.1e-04	5.9e-09	8.9e-10	2.8e-03	4.9e-03	5.5e-09
9.4e+03	1.0e+06	5.5e-10	2.5e-03	8.0e-04	2.7e-03	1.2e-06	2.5e-03
9.4e+03	1.0e+06	2.3e-06	5.6e-03	8.0e-04	2.3e+09	4.4e+09	5.1e-03
6.6e+07	1.2e+00	1.9e-02	7.1e-01	2.2e-01	7.5e-01	2.2e-02	7.0e-01
6.6e+07	1.2e+00	6.9e-01	1.1e+00	2.2e-01	2.0e+14	2.1e+14	9.7e-01

A and **b** are now assumed to have nonnegative entries. The bound in (3.2.11) is useful when we know that a particular value of x is going to be perturbed more or less than the others due to the nature of A, b or M, whereas, a normwise bound only gives us a perturbation bound for x as a whole.

3.3 Existing Methods for Solving the Indefinite Least Squares Problem

3.3.1 Solving the Normal Equations

The simplest way to solve the ILS problem is to solve for x in the normal equations (3.1.4). This, however, is not the method of choice due to its bad stability properties. To illustrate this, we consider the following example. Suppose

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{bmatrix}, \qquad J = \begin{bmatrix} I_3 & 0 \\ 0 & -1 \end{bmatrix},$$

then

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

The problem is clear. The matrix A^TJA is of rank 3, but if $\epsilon < \sqrt{u}$, where u is the unit roundoff, then A^TJA, and hence A, will be interpreted as having rank one. In other words, we are in effect ignoring all but the first row of data in the matrix A. This makes A^TJA singular, and thus violating the condition (3.1.5) which is necessary for us to obtain a unique solution to the ILS problem. Also, the usual method for solving the normal equations of the ILS problem, and, for that matter, any other type of LS problem, is to employ the Cholesky factorization, and this is likely to breakdown as it is singular [42, Section 20.4].

We note in passing that (3.1.4) gives $x = M^{-1}A^TJb$, where $M = A^TJA$, and the matrix $X = M^{-1}A^TJ$ is a pseudo-inverse of A but not the Moore-Penrose pseudo-inverse ($XA = I$, but AX is not symmetric).

To solve the ILS problem by solving its normal equations, the following procedure can be used:

Algorithm 3.3.1 *This algorithm solves the ILS problem by solving its normal equations.*

```
Form $C := A^TJA = A_1^TA_1 - A_2^TA_2$.
Form $c := A^TJb$.
Compute the Cholesky factorization $C = R^TR$.
Solve $R^Ty = c$.
Solve $Rx = y$.
```

Cost: Table 3.3.1 shows cost of the individual steps for forming and solving the ILS problem by solving its normal equations as described in Algorithm 3.3.1. Note that the Cholesky factor R in Algorithm 3.3.1 is precisely the upper triangular factor in the hyperbolic QR factorization of A (see Section 2.3) since $A^TJA = R^TR$.

3.3.2 Solving the Augmented System

Another way to solve the ILS problem is to solve the augmented system in (3.1.6). The following algorithm shows how to do this.

Algorithm 3.3.2 *This algorithm solves the ILS problem by solving the augmented system.*

```
Solve $\left[\begin{smallmatrix} J & A \\ A^T & 0 \end{smallmatrix}\right] z = [b^T\ 0]^T$ by Gaussian elimination with partial pivoting
$x := z(m+1\colon m+n)$.
```

Table 3.3.1: The number of flops required to solve the ILS problem by solving its normal equations as described in Algorithm 3.3.1.

Step of Algorithm 3.3.1	Flops
Form $C\colon = A^TJA = A_1^TA_1 - A_2^TA_2$	mn^2
Form $c\colon = A^TJb$.	$2mn$
Compute the Cholesky factorization $C = R^TR$.	$n^3/3$
Solve $R^Ty = c$.	n^2
Solve $Rx = y$.	n^2
Total	$n^2(m + n/3)$

Table 3.3.2: The number of flops required to solve the ILS problem by solving its augmented system as described in Algorithm 3.3.2.

Step of Algorithm 3.3.2	Flops
Solve $[\begin{smallmatrix} J & A \\ A^T & 0 \end{smallmatrix}]z = [b^T\ 0]^T$.	$2(m+n)^3/3$
Total	$2(m+n)^3/3$

Cost: Table 3.3.2 shows the cost of the individual steps for solving the ILS problem by solving its augmented system as described in Algorithm 3.3.2. Clearly, this method would be unsuitable for most problems due to its cost, except in the case where the dimensions of the problem were tiny.

3.3.3 The QR-Cholesky Method

There already exists a backward stable method to solve the ILS problem, which we outline below. This method is explained in detail by Chandrasekaran, Gu and Sayed [14]. We will refer to this method as the QR-Cholesky method from now on.

We first compute the QR factorization of the matrix A in (3.1.2) to get

$$A = \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} R = QR, \tag{3.3.1}$$

where $R \in \mathbb{R}^{n\times n}$ is upper triangular, $Q_1 \in \mathbb{R}^{p\times n}$, $Q_2 \in \mathbb{R}^{q\times n}$, and Q is orthogonal, i.e.,

$$Q^T Q = Q_1^T Q_1 + Q_2^T Q_2 = I_n. \tag{3.3.2}$$

It follows from (3.1.5) that R is nonsingular.

Let us denote the unique solution to (3.1.4) by x_s. Substituting (3.3.1) into (3.1.4), and simplifying, we obtain

$$\left(Q^T J Q\right) R x_s = \left(Q_1^T Q_1 - Q_2^T Q_2\right) R x_s = Q^T J b. \tag{3.3.3}$$

We also note here that substituting (3.3.1) into (3.1.5), we get that

$$R^T \left(Q_1^T Q_1 - Q_2^T Q_2\right) R$$

is symmetric positive definite, from which we can deduce that

$$T := Q_1^T Q_1 - Q_2^T Q_2 \tag{3.3.4}$$

is also symmetric positive definite.

Note that using (3.3.2), we have

$$T = 2Q_1^T Q_1 - I_n \tag{3.3.5a}$$

and

$$T = I_n - 2Q_2^T Q_2. \tag{3.3.5b}$$

So if we want to compute T, then we have three ways in which it can be done. It is obvious that evaluation by (3.3.4) can be a lot more expensive to compute than by using (3.3.5a) and (3.3.5b), especially when either of p and q is large.

To solve (3.3.3), we explicitly compute the matrix T and then compute its Cholesky factorization $T = LL^T$, where $L \in \mathbb{R}^{n\times n}$ is lower triangular. We then compute the right-hand side of (3.3.3),

$$d := Q^T Jb.$$

Equation (3.3.3) now reduces to solving

$$LL^T Rx_s = d.$$

This can be done by solving $Ld' = d$ by forward substitution, $L^T d'' = d'$ by backward substitution, and $Rx_s = d''$ again by backward substitution.

It is shown in [14] that this method, with (3.3.4) used to compute T, produces a computed solution $\widehat{x}$ that solves the problem

$$\min_x (b + \Delta b - (A + \Delta A)x)^T J(b + \Delta b - (A + \Delta A)x),$$

where

$$\|\Delta A\|_F \leq c_{m,n} u \|A\|_F, \qquad \|\Delta b\|_2 \leq c_{m,n} u \|b\|_2,$$

with $c_{m,n}$ a constant depending on the problem dimensions and u the unit roundoff; in other words, the QR-Cholesky method is backward stable. Error analysis for this method with one of the equations in (3.3.5) used to compute T has not been given, however, it is clear that if $p < q$ then (3.3.5a) would be cheaper to compute and if $p < q$ then (3.3.5a) would be the cheaper choice. In either case, and in the case $p = q$, both equations in (3.3.5) are cheaper to compute than (3.3.4), except when $p = 0$ or $q = 0$.

Below, we have given the algorithm for solving the ILS problem using the QR-Cholesky method, as shown by Chandrasekaran, Gu and Sayed in [14].

Algorithm 3.3.3 *Solution of the ILS problem using the QR-Cholesky method, as shown by Chandrasekaran, Gu and Sayed in [14], using* (3.3.4) *to compute* T*.*

Table 3.3.3: The number of flops required to solve the ILS problem by the QR-Cholesky method as described in Algorithm 3.3.3.

Step of Algorithm 3.3.3	Flops
Compute the QR factorization $A = QR$	$2n^2(m - n/3)$
Explicitly compute Q	$2n^2(m - n/3)$
Form $T = Q_1^T Q_1 - Q_2^T Q_2$	mn^2
Cholesky factorize $T = LL^T$	$n^3/3$
Compute $d = Q^T(Jb)$	$2mn$
Solve $Ld' = d$ by forward substitution	n^2
Solve $L^T d'' = d'$ by backward substitution	n^2
Solve $Rx_s = d''$ by backward substitution	n^2
Total	$n^2(5m - n)$

```
Compute the "economy size" Householder QR factorization $A = QR$
    ($Q \in \mathbb{R}^{m\times n}$, $R \in \mathbb{R}^{n\times n}$)
Explicitly compute $Q$ in the Householder QR factorization of $A$
Form $T = Q_1^T Q_1 - Q_2^T Q_2$ ($Q = [Q_1^T\ Q_2^T]^T$, $Q_1 \in \mathbb{R}^{p\times n}$, $Q_2 \in \mathbb{R}^{q\times n}$)
Compute the Cholesky factorization $T = LL^T$ ($L \in \mathbb{R}^{n\times n}$ lower triangular)
Compute $d = Q^T(Jb)$
Solve $Ld' = d$ by forward substitution
Solve $L^T d'' = d'$ by backward substitution
Solve $Rx_s = d''$ by backward substitution
```

Cost: We give a breakdown of the costs involved in each step of Algorithm 3.3.3 for computing the solution of the ILS problem in Table 3.3.3.

Note that if we used (3.3.5) to compute T, then the cost for computing the solution of the ILS problem would reduce to $n^2(4m + \min(p, q) - n)$ flops, giving us a significant reduction when $\min(p, q) \ll m$.

Chapter 4

The Hyperbolic QR Factorization Method

A well known method for solving the standard LS problem (3.1.1) is to compute the QR factorization (1.11.1) of $A \in \mathbb{R}^{m\times n}$ and solve the system of equations $Rx = c\,(1{:}\,n)$, where $R \in \mathbb{R}^{n\times n}$ is upper triangular, and $c = Qb$. We show below that something similar can also be done for the indefinite case.

4.1 The Hyperbolic QR Factorization Method

We know that since (3.1.5) holds for the matrix $A \in \mathbb{R}^{m\times n}$, we can compute its hyperbolic QR factorization (2.3.1). Thus

$$\begin{aligned} Q^{-1}(b - Ax) &= JQ^TJ(b - Ax) \\ &= d - \begin{array}{c} \\ n \\ m-n \end{array}\!\!\begin{array}{c} n \\ \begin{bmatrix} R \\ 0 \end{bmatrix} \end{array} x, \qquad b = Qd \\ &= \begin{bmatrix} d_1 - Rx \\ d_2 \end{bmatrix}, \qquad d = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}\!\!\begin{array}{c} n \\ m-n \end{array}. \end{aligned}$$

Using this relation, the ILS problem (3.1.2) can be written as

$$\begin{aligned}(b-Qx)^T J(b-Ax) &= \begin{bmatrix} d_1 - Rx \\ d_2 \end{bmatrix}^T Q^T J Q \begin{bmatrix} d_1 - Rx \\ d_2 \end{bmatrix}, \\ &= \begin{bmatrix} d_1 - Rx \\ d_2 \end{bmatrix}^T J \begin{bmatrix} d_1 - Rx \\ d_2 \end{bmatrix}, \\ &\qquad \text{using (2.1.1) and Lemma 2.1.2} \\ &= \|d_1 - Rx\|_2^2 + d_2 J(n+1\colon m, n+1\colon m) d_2 \qquad (4.1.1)\end{aligned}$$

using Lemma 2.3.1. Since the second term in (4.1.1) does not depend on x, this reduces the solution of the ILS problem to solving the system of linear equations

$$Rx = d_1.$$

We will refer to this method of solving the ILS problem as the HQR method. This, of course, is the analogue of Golub's method for solving the LS problem [32].

The complete algorithm for solving the ILS problem is summarized as follows.

Algorithm 4.1.1 *This algorithm solves the ILS problem* (3.1.2) *using the Householder QR factorization and interleaving non-overlapping Householder transforms and hyperbolic rotations.*

```
Compute the Householder QR factorization $A(1\colon p,:) = Q_1[R^T\ 0]^T$
    ($Q_1 \in \mathbb{R}^{p\times p}$, $R \in \mathbb{R}^{n\times n}$), overwriting $A(1\colon p,:)$ with $[R^T\ 0]^T$ and
    $b(1\colon p)$ with $Q_1^T b(1\colon p)$.
for $j = 1\colon \min(m-1, n)$
    Construct a Householder transformation $P_j$ such that
        $P_j\, A(p+1\colon m, j) = \sigma_j e_1$.
    $A(p+1\colon m, j\colon n) = P_j\, A(p+1\colon m, j\colon n)$
    $b(p+1\colon m) = P_j\, b(p+1\colon m)$
    % Eliminate sole remaining subdiagonal element in column $j$ by a
```

```
        % hyperbolic rotation.
        $[c, s] = \text{Hrotate}(A(j, j), A(p+1, j))$
        $A([j\ p+1], j\!:n) = \text{Happly}(c, s, A([j\ p+1], j\!:n))$
        $b([j\ p+1]) = \text{Happly}(c, s, b([j\ p+1]))$
    end
    $R = A(1\!:n, :)$
    Solve $Rx = b(1\!:n)$ by backward substitution.
```

Cost: We give a breakdown of the costs involved in each step of Algorithm 4.1.1 for solving the ILS problem in Table 4.1.1. Note that the J-orthogonal matrix Q is applied to the vector b as it is formed in the same way as it is applied to A to obtain $[R^T\ 0]^T$. Thus we do not explicitly need to compute Q, giving us a considerable saving on the number of flops needed to compute the hyperbolic QR factorization of A.

The operation count of Algorithm 4.1.1 is the same as that for the solution of the standard LS problem by Householder QR factorization. This is essentially because the hyperbolic rotations contribute only to the lower order terms in the operations count as can be seen in Table 4.1.1. Table 4.1.2 compares the cost of Algorithm 4.1.1 with the cost of forming and solving the normal equations (3.1.4) as described in Algorithm 3.3.1, the cost of solving the augmented system (3.1.6) as described in Algorithm 3.3.2, and the cost of the QR-Cholesky method as described in Algorithm 3.3.3. Clearly, Algorithm 4.1.1 is significantly less expensive than the QR-Cholesky method, especially when $m \gg n$.

4.2 Rounding Error Analysis

We now give a rounding error analysis of Algorithm 4.1.1. In Section 2.3.4, we saw that the computation of R in the hyperbolic QR factorization of A is stable

Table 4.1.1: The number of flops required to solve the ILS problem by the HQR method as described in Algorithm 4.1.1.

Step of Algorithm 4.1.1	Flops
Compute the QR factorization $A(1{:}\,p,:) = Q_1[R^T\ 0]^T$	$2n^2(p - n/3)$
Compute $Q_1^T b(1{:}\,p)$	$2n(2m - n)$
Construct P_j (n times)	$3n(m - p)$
$A(p+1{:}\,m, j{:}\,n) = P_j\, A(p+1{:}\,m, j{:}\,n)$ (n times)	$2n^2(m - p)$
$b(p+1{:}\,m) = P_j\, b(p+1{:}\,m)$ (n times)	$2n(m - p)$
$[c, s] = \text{Hrotate}(A(j,j), A(p+1,j))$ (n times)	$5n$
$A([j\ p+1], j{:}\,n) = \text{Happly}(c, s, A([j\ p+1], j{:}\,n))$ (n times)	$21n^2/2$
$b([j\ p+1]) = \text{Happly}(c, s, b([j\ p+1]))$ (n times)	$21n/2$
Solve $Rx = b(1{:}\,n)$ by backward substitution.	n^2
Total	$2n^2(m - n/3)$

Table 4.1.2: A comparison of the number of flops required to solve the ILS problem by solving the normal equations (Algorithm 3.3.1), by solving the augmented system (Algorithm 3.3.2), by using the HQR method (Algorithm 4.1.1) and by using the QR-Cholesky method (Algorithm 3.3.3).

	Normal Equations	Augmented System	Hyperbolic QR method	QR-Cholesky method
	$n^2(m + n/3)$	$(m + n)^3/3$	$2n^2(m - n/3)$	$n^2(5m - n)$
$m \approx n$	$4n^3/3$	$8n^3/3$	$4n^3/3$	$4n^3$
$m \gg n$	mn^2	$m^3/3$	$2mn^2$	$5mn^2$

when we are using non-overlapping J-orthogonal transformations.

In solving the ILS problem we also transform the right-hand side, $b = [b_1^T \; b_2^T]^T$ to $d = [d_1^T \; d_2^T]^T$. The above analysis gives

$$\begin{array}{c} p \\ q \end{array}\begin{bmatrix} \widehat{d}_1 + \delta_3 \\ b_2 + \delta_2 \end{bmatrix} = G \begin{bmatrix} b_1 + \delta_1 \\ \widehat{d}_2 \end{bmatrix}\begin{array}{c} p \\ q \end{array}, \tag{4.2.1}$$

where $\delta_3(n+1\colon p) = 0$ and

$$\|\delta_1\|_2 \le \widetilde{\gamma}_p \|b_1\|_2, \qquad \|\delta_i\|_2 \le \widetilde{\gamma}_q \max(\|\widehat{d}_1(1\colon n)\|_2, \|b_2\|_2), \quad i = 2\colon 3.$$

The ensuing analysis is simpler if $\widehat{d}_1$ is not perturbed, so we rewrite this relation as

$$\begin{bmatrix} \widehat{d}_1 \\ b_2 + \overline{\delta}_2 \end{bmatrix} = G \begin{bmatrix} b_1 + \overline{\delta}_1 \\ \widehat{d}_2 + \overline{\delta}_3 \end{bmatrix}, \tag{4.2.2}$$

where $\overline{\delta}_2 = \delta_2$ and

$$\max_{i=1:3} \|\overline{\delta}_i\|_2 \le \widetilde{\gamma}_m \max(\|\widehat{d}_1(1\colon n)\|_2, \|b\|_2). \tag{4.2.3}$$

In Algorithm 4.1.1 the final step is to solve the triangular system $Rx = d_1$, where $R = R_1(1\colon n, :)$. The computed solution $\widehat{x}$ satisfies $(\widehat{R} + \Delta R)\widehat{x} = \widehat{d}_1(1\colon n)$, $|\Delta R| \le \gamma_n |\widehat{R}|$ [42, Theorem 8.5], that is, the rounding errors in the substitution correspond to a further small perturbation of $\widehat{R}$.

We now consider the forward error of the computed solution $\widehat{x}$. First, let z_1 be the solution of the perturbed ILS problem with data

$$A + \Delta A := \begin{bmatrix} A_1 + \Delta_1 \\ A_2 + \Delta_2 \end{bmatrix}, \qquad b + \Delta b := \begin{bmatrix} b_1 + \overline{\delta}_1 \\ b_2 + \overline{\delta}_2 \end{bmatrix},$$

for which we know from (2.3.9) that the exact upper triangular R-factor is $\widehat{R} + \widetilde{\Delta}_3$, where $\widetilde{\Delta}_3 = \Delta_3(1\colon n, :)$. Then, in view of (4.2.2),

$$(\widehat{R} + \widetilde{\Delta}_3) z_1 = \widehat{d}_1(1\colon n).$$

Write

$$x - \widehat{x} = (x - z_1) + (z_1 - \widehat{x}).$$

Using the bounds on ΔA and Δb in (2.3.10) and (4.2.3), we have, from (3.2.7),

$$\begin{aligned}\frac{\|x - z_1\|_2}{\|x\|_2} \le \widetilde{\gamma}_{mn}\Bigg[\|M^{-1}A^T\|_2\,\|A\|_F\left(\frac{\max(1,\theta)\,\|b\|_2}{\|A\|_F\,\|x\|_2} + 1\right)&\\ + \|M^{-1}\|_2\,\|A\|_F^2\,\frac{\|r\|_2}{\|A\|_F\,\|x\|_2}\Bigg] + O(u^2),&\qquad (4.2.4)\end{aligned}$$

where

$$\theta = \frac{\|d_1(1\colon n)\|_2}{\|b\|_2}. \qquad (4.2.5)$$

The quantity θ measures the growth in the leading n components of the right hand side as a result of the transformations that reduce A to triangular form. We now show that even though θ can be large, it is innocuous. Suppose that $\theta \gg 1$. Note first that, since $\|d_1\|_2^2 + \|b_2\|_2 = \|b_1\|_2 + \|d_2\|_2$, we have $\|d_2\|_2 \approx \|d_1\|_2 \gg \|b_1\|_2$. Note also that $b_1(n+1\colon p)$ is not subjected to hyperbolic rotations and hence $\|d_1(n+1\colon p)\|_2 \le \|b\|_2$. Hence, from (4.1.1),

$$\|r\|_2^2 \ge |(b - Ax)^T J(b - Ax)| = |\ \|d(n+1\colon p)\|_2^2 - \|d_2\|_2^2\ | \approx \|d_2\|_2^2 \approx \|d_1\|_2^2\,.$$

Therefore $\theta \lesssim \|r\|_2 / \|b\|_2$ and it follows that the first term in (4.2.4) is no larger than the second, showing that a large θ does not worsen the bound. Therefore (4.2.4) is essentially the same as (3.2.7) with $\epsilon = \widetilde{\gamma}_{mn}$.

From standard perturbation theory for square linear systems, the term $\|z_1 - \widehat{x}\|_2 / \|x\|_2$ is bounded by

$$\begin{aligned}\phi &= \kappa_2(R)\left(\gamma_n + \widetilde{\gamma}_{qn}\frac{\max(\|R\|_F\,,\|A_2\|_F)}{\|R\|_2}\right)\\ &= \|R^{-1}\|_2(\gamma_n\,\|R\|_2 + \widetilde{\gamma}_{qn}\max(\|R\|_F\,,\|A_2\|_F))\\ &\le \widetilde{\gamma}_{qn}\|R^{-1}\|_2\,\|A\|_F\,. \qquad (4.2.6)\end{aligned}$$

Now from the exact arithmetic analogue of (2.3.9) we have

$$\begin{bmatrix} R \\ A_2 \end{bmatrix} = G\begin{bmatrix} A_1 \\ 0 \end{bmatrix},$$

where G is orthogonal. Postmultiplying by $R^{-1}R^{-T}$ and transposing gives

$$[\,R^{-1} \quad R^{-1}R^{-T}A_2^T\,] = [\,R^{-1}R^{-T}A_1^T \quad 0\,]\,G^T.$$

Recalling that $M = A^T JA = R^T R$, it follows that

$$\begin{aligned}\|R^{-1}\|_2 &\leq \|[\, R^{-1}R^{-T}A_1^T \quad 0\,]\|_2 \\ &\leq \|R^{-1}R^{-T}\,[\, A_1^T \quad A_2^T\,]\,\|_2 \\ &= \|M^{-1}A^T\|_2.\end{aligned}$$

Hence

$$\phi \leq \widetilde{\gamma}_{qn}\|M^{-1}A^T\|_2\,\|A\|_F\,,$$

which is smaller than the first term in (4.2.4). Our overall conclusion is that $\|x - \widehat{x}\|_2 \,/\, \|x\|_2$ has an upper bound no larger than (3.2.7) with $\epsilon = \widetilde{\gamma}_{mn}$.

Recall that a method for solving the ILS problem is forward stable if it produces a computed solution with forward error similar to that for a backward stable method. If we make the reasonable assumption that the perturbation bound (3.2.7) is approximately attainable, then our rounding error analysis has shown that the hyperbolic QR factorization method for solving the ILS problem is forward stable.

It is unclear whether the hyperbolic QR factorization method is mixed forward-backward stable, or even backward stable. It is an open problem to determine a computable formula for the backward error of an arbitrary approximate solution to the ILS problem, and without such a formula it is difficult to test numerically for backward instability.

4.3 Numerical Experiments

We have used MATLAB to carry out experiments to compare the forward errors $\|x - \widehat{x}\|_2 \,/\, \|x\|_2$ from Algorithm 4.1.1, the normal equations method as described in Algorithm 3.3.1, solving the augmented system as described in Algorithm 3.3.2, and the QR-Cholesky method as described in Algorithm 3.3.3. We approximated the exact solution by forming and solving the normal equations in 100 digit arithmetic using MATLAB's Symbolic Math Toolbox. We report results with $m = 16$, $n = 8$ and $p = 10$.

We generate test problems for which Q and R in the hyperbolic QR factorization of A can be chosen to have particular norms. The matrix R was generated via the QR factorization of a random matrix with preassigned singular value distributions (generated via MATLAB's `gallery('randsvd',...)`). Random J-orthogonal matrices Q of specified norm were generated using the method of Higham [43]. Once we compute A, we can randomly generate the solution x from the $N(0, 1)$ distribution. By either setting $b = Ax$ or choosing a random b of size $\|Ax\|_2$, we found that we could control whether θ, defined in (4.2.5), was small or large, particularly for ill-conditioned problems. For well conditioned problems, we also found that choosing $b = Ax$ made the residual $r = b - Ax_{\text{actual}}$ tiny, and choosing a random b made it very large, where x_{actual} is the actual solution of the ILS problem. For ill-conditioned problems, the difference in the sizes of r was not as extreme, though still noticable. Note that the chosen x is not necessarily the *actual* solution of the ILS problem, since choosing $b = Ax$ nullifies $r^T Jr$, which is what we are trying to minimize, but another x' may make $r^T Jr$ negative. This method for generating test problems was also employed in all the tests performed in Section 2.3.5 for testing the stability of the hyperbolic QR factorization, in Section 3.2.2 for testing the values of the two perturbation bounds (3.2.7) and (3.2.10), and in all the tests performed in this chapter.

We show the results in Table 4.3.1. The first three columns show the values for $\|Q\|_2$, $\|R\|_2$ and r for the test problem. The next four columns show the relative forward errors for the four methods of interest. The eighth column tabulates the first order part of the perturbation bound (3.2.7) (which we numerically showed in Section 3.2.2 to always be of similar size to the *tighter* bound (3.2.10) derived using Kronecker products), with $\epsilon = u$, $\mathbf{A} = A$ and $\mathbf{b} = b$; thus (3.2.7) is a first order bound for the forward of a backward stable method. Finally, the ninth column tabulates the value of θ defined in (4.2.5).

Two main conclusions can be drawn from the results shown. First, as expected, the normal equations method is not forward stable, particularly for large $\|R\|_2$. Second, Algorithm 4.1.1 behaves in a forward stable way in these tests, and is just as accurate as the backward stable QR-Cholesky method, even when θ and $\|Q\|_2$ are large. The latter behaviour adequately summarizes more extensive experiments that we have carried out. Finally, we can see from the results that the augmented system method also performs stably.

4.4 Iterative Refinement

The accuracy and stability of an approximate solution of the ILS problem can be improved by applying iterative refinement (see Section 1.10) to the augmented system in (3.1.6) just as with the standard LS problem.

To apply iterative refinement we need to be able to solve systems of the form

$$\begin{bmatrix} J & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} s \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}.$$

We show how this can be done for the hyperbolic QR factorization method. The equations are

$$\begin{aligned} Js + Ay &= f, \\ A^T s &= g, \end{aligned}$$

which can be rewritten using the hyperbolic QR factorization as

$$Js + Q \begin{bmatrix} R \\ 0 \end{bmatrix} y = f, \tag{4.4.1a}$$

$$[\, R^T \quad 0 \,]\, Q^T s = g. \tag{4.4.1b}$$

Table 4.3.1: Errors $\|x - \widehat{x}\|_2 \,/\, \|x\|_2$, where x is the exact solution of the ILS problem, for the four methods for solving the ILS problem with their corresponding values of the perturbation bound (3.2.7) and θ (4.2.5).

$\|Q\|_2$	$\|R\|_2$	$\dfrac{\|r\|_2}{\|A\|_2\,\|x\|_2}$	QR-Cholesky method	Hyperbolic QR method	Normal Equations	Augmented System	Bound (3.2.7)	θ (4.2.5)
1.0e+00	1.0e+00	4.9e-17	1.6e-16	3.6e-16	2.8e-16	9.9e-17	4.3e-16	1.0e-00
1.0e+00	1.0e+00	1.5e+00	1.6e-16	3.1e-16	1.6e-16	1.5e-16	9.8e-16	5.6e-01
1.0e+00	1.0e+08	0	1.6e-09	1.7e-09	8.8e-02	9.2e-10	1.4e-08	1.0e+00
1.0e+00	1.0e+08	1.1e-07	8.9e-09	8.9e-09	6.9e-02	1.0e-09	1.3e-07	5.6e-01
9.4e+03	1.0e+06	5.5e-10	1.7e-05	8.4e-05	1.2e-01	6.4e-05	2.5e-03	2.5e-04
9.4e+03	1.0e+06	2.3e-06	9.3e-06	7.7e-04	8.8e-02	9.0e-05	5.6e-03	1.2e+03
6.6e+07	1.2e+00	1.9e-02	1.2e-01	1.3e-01	7.5e-05	1.5e-02	7.1e-01	4.7e-08
6.6e+07	1.2e+00	6.9e-01	2.4e-01	2.4e-01	5.6e-05	8.7e-02	1.1e+00	8.3e+06

Let

$$h = R^{-T}g, \qquad d = \begin{array}{c} n \\ m-n \end{array}\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = JQ^TJf. \tag{4.4.2}$$

Substituting (4.4.2) into (4.4.1) gives us

$$JQ^Ts + \begin{bmatrix} R \\ 0 \end{bmatrix} y = JQ^TJf = d, \tag{4.4.3a}$$

$$Q(:,1\colon n)^Ts = h. \tag{4.4.3b}$$

Since $J(1\colon n, 1\colon n) = I_n$, we can write the first n rows of (4.4.3a) using (4.4.3b) as

$$h + Ry = d_1,$$

and hence

$$y = R^{-1}(d_1 - h). \tag{4.4.4a}$$

Combining (4.4.3b) with the last $m-n$ rows of (4.4.3a) now gives us

$$JQ^Ts = \begin{bmatrix} h \\ d_2 \end{bmatrix},$$

and so

$$s = JQ\begin{bmatrix} h \\ d_2 \end{bmatrix}. \tag{4.4.4b}$$

Note that in our case, $y = x$, $f = b$ and $g = 0$, so these equations reduce to

$$x = R^{-1}d_1, \qquad s = JQ\begin{bmatrix} 0 \\ d_2 \end{bmatrix}, \tag{4.4.5}$$

where $d = JQ^TJb$.

Thus, using the hyperbolic QR factorization, the augmented system in (3.1.6) reduces to solving (4.4.5).

As we saw in Algorithm 1.10.1, we can now use (4.4.4) to improve the solution we have obtained using the hyperbolic QR factorization method by first computing the residual of (3.1.6) to increased accuracy.

4.4.1 Numerical Results

We present some numerical results in Tables 4.4.1 and 4.4.2. The results presented in these tables were generated using the same method as those in Section 4.3.

The first three columns of Table 4.4.1 tabulate the values of $\|Q\|_2$, $\|R\|_2$ and the residual r of the problem that we have generated. The fourth column tabulates the error in the solution of the ILS problem that we obtained by the HQR method. The fifth column tabulates the error in the solution of the ILS problem obtained by solving the augmented system (3.1.6). Column six tabulates the error in the solution after iteratively refining the solution by the HQR method. Column seven gives the number of iterations that were required to obtain this new solution. As we mentioned in Section 1.10, the first step of iterative refinement in Algorithm 1.10.1 has been computed to 32 significant digits, which is approximately the same as computing it to precision $u^2 = (2^{-53})^2 \approx 1.2 \times 10^{-32}$. The eighth and ninth columns are just as the previous two, but the residual in the iterative refinement have been computed to precision u. The final column tabulates the perturbation bound (3.2.7) for the problem.

Table 4.4.2 tabulates the results of iterative refinement just as in Table 4.4.1, but instead of using the augmented system in the iterative refinement steps, we have used (4.4.4) to reduce the number of flops in computing the solution. In all these results, we have always applied the matrix Q in factored form.

The two conclusions that we draw from these results are:

- Using (4.4.4) in iterative refinement always seemed to give results comparable to those obtained by solving the augmented system using Gaussian elimination with partial pivoting.

- Computing the residuals to single precision did not produce results as accurate as those in which the residual was computed to double precision, even though, in some cases, a lot more refinement steps were computed in single precision than in double precision.

Table 4.4.1: Comparison of errors in solving the ILS problem by the HQR method and iterative refinement by solving the augmented system using Gaussian elimination.

HQR	—	Solution by the HQR method.
AUG	—	Solution by solving the augmented system (3.1.6).
IR2	—	Iterative refinement of the solution by the HQR method with the residual computed to double precision.
its2	—	Number of times iterative refinement in double precision was carried out.
IR	—	Iterative refinement of the solution by the HQR method with the residual computed to single precision.
its	—	Number of times iterative refinement in single precision was carried out.
Bound	—	The perturbation bound for the ILS problem in (3.2.7).

$\|Q\|_2$	$\|R\|_2$	r	HQR	AUG	IR2	its2	IR	its	Bound
1.0e+00	1.0e+00	4.9e-17	3.6e-16	9.9e-17	0	1	2.0e-17	1	4.3e-16
1.0e+00	1.0e+00	1.5e+00	3.1e-16	1.5e-16	0	1	8.1e-17	1	9.8e-16
1.0e+00	1.0e+02	3.1e-17	2.6e-15	9.7e-16	0	1	2.7e-16	13	1.7e-14
1.0e+00	1.0e+08	1.1e-07	8.9e-09	1.0e-09	3.8e-17	1	9.0e-13	2	1.3e-07
9.4e+01	1.0e+00	3.0e-14	3.0e-13	2.7e-14	0	1	1.7e-14	6	1.1e-12
9.4e+03	1.0e+00	7.3e-10	1.3e-09	1.8e-10	0	1	3.9e-10	3	1.1e-08
9.4e+05	1.0e+02	3.0e-06	2.4e-04	7.1e-05	7.6e-13	2	4.9e-05	16	3.3e-03
9.4e+05	1.0e+02	1.7e-02	9.0e-04	1.0e-04	5.9e-12	2	2.8e-06	2	6.4e-03
6.6e+07	1.2e+00	6.9e-01	2.4e-01	8.7e-02	2.6e-03	2	3.4e-02	4	1.1e+00

Table 4.4.2: Comparison of errors in solving the ILS problem by the HQR method and iterative refinement by using (4.4.4).

HQR	—	Solution by the HQR method.
HAUG	—	Solution by solving (4.4.5).
IR2	—	Iterative refinement of the solution by the HQR method with the residual computed to double precision.
its2	—	Number of times iterative refinement in double precision was carried out.
IR	—	Iterative refinement of the solution by the HQR method with the residual computed to single precision.
its	—	Number of times iterative refinement in single precision was carried out.
Bound	—	The perturbation bound for the ILS problem in (3.2.7).

$\|Q\|_2$	$\|R\|_2$	r	HQR	HAUG	IR2	its2	IR	its	Bound
1.0e+00	1.0e+00	4.9e-17	3.6e-16	3.6e-16	0	1	2.0e-17	1	4.3e-16
1.0e+00	1.0e+00	1.5e+00	3.1e-16	3.1e-16	0	1	8.1e-17	1	9.8e-16
1.0e+00	1.0e+02	3.1e-17	2.6e-15	2.6e-15	0	1	2.7e-16	13	1.7e-14
1.0e+00	1.0e+08	1.1e-07	8.9e-09	8.9e-09	4.2e-17	1	7.2e-13	2	1.3e-07
9.4e+01	1.0e+00	3.0e-14	3.0e-13	3.0e-13	0	1	1.7e-14	6	1.1e-12
9.4e+03	1.0e+00	7.3e-10	1.3e-09	1.3e-09	0	1	4.7e-10	3	1.1e-08
9.4e+05	1.0e+02	3.0e-06	2.4e-04	2.4e-04	5.3e-08	2	5.5e-05	3	3.3e-03
9.4e+05	1.0e+02	1.7e-02	9.0e-04	9.0e-04	3.4e-10	2	3.7e-06	2	6.4e-03
6.6e+07	1.2e+00	6.9e-01	2.4e-01	2.4e-01	2.4e-02	2	3.7e-02	2	1.1e+00

Chapter 5

The Indefinite Least Squares Problem with Equality Constraints

5.1 Introduction

We are concerned here with the extension of the ILS problem to include equality constraints:

$$\text{ILSE}: \qquad \min_x (b - Ax)^T J(b - Ax) \qquad \text{subject to } Bx = d, \tag{5.1.1}$$

where $B \in \mathbb{R}^{s\times n}$ and $d \in \mathbb{R}^s$. This work is analogous to the work on the LS problem subject to equality constraints,

$$\text{LSE}: \qquad \min_x (b - Ax)^T (b - Ax) \qquad \text{subject to } Bx = d, \tag{5.1.2}$$

which was discussed by Cox and Higham [20].

In what follows, we will derive a method that is analogous to the well known null space method for solving the LSE problem (5.1.2).

Throughout this chapter, we will assume that

$$J = \begin{bmatrix} -I_q & 0 \\ 0 & I_p \end{bmatrix}, \qquad p + q = m. \tag{5.1.3}$$

This change from the J we used in Chapters 2, 3 and 4 will help us to prove the existence of the generalized hyperbolic QR factorization in Theorem 5.1.1 below. Note that we are not in any way changing the ILSE problem by defining J like this, since we can also permute the rows of $[A \mid b]$ when moving from the J in (2.1.2) to the one in (5.1.3).

We will also assume that

$$\operatorname{rank}(B) = s, \qquad x^T(A^TJA)x > 0 \text{ for all nonzero } x \in \mathcal{N}(B). \tag{5.1.4}$$

The first condition ensures that there is a solution to the constraint equations. The second condition, which says that A^TJA is positive definite on the null space of B, then ensures that there is a unique solution to the ILSE problem. The uniqueness can be shown by manipulating the normal equations

$$A^TJ(b - Ax) = B^T\mu, \qquad Bx = d, \tag{5.1.5}$$

where μ is a vector of Lagrange multipliers, and it is also established by our analysis in terms of the generalized hyperbolic QR factorization (see Section 5.4).

We note for later use that A^TJA has rank at most p, and so since $\mathcal{N}(B)$ has dimension $n - s$, the second condition in (5.1.4) implies that

$$p \geq n - s. \tag{5.1.6}$$

This can be proved in an analogous way as Lemma 2.3.1.

Central to this work is a new factorization that we call the generalized hyperbolic QR factorization. The following result defines the factorization and establishes its existence.

Theorem 5.1.1 (generalized hyperbolic QR factorization) *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{s\times n}$, let J be given by* (5.1.3)*, and assume that* (5.1.4) *holds. Then there exist an orthogonal $Q \in \mathbb{R}^{n\times n}$ and a J-orthogonal $H \in \mathbb{R}^{m\times m}$ such that*

$$A = H \begin{array}{c} \\ {\scriptstyle m-(n-s)} \\ {\scriptstyle n-s} \end{array}\!\! \begin{array}{cc} {\scriptstyle s} & {\scriptstyle n-s} \\ \left[\begin{matrix} L_{11} & 0 \\ L_{21} & L_{22} \end{matrix}\right. & \left.\right] \end{array} Q^T, \qquad BQ = \begin{array}{cc} {\scriptstyle s} & {\scriptstyle n-s} \\ [\,K & 0\,] \end{array} {\scriptstyle s}\ , \tag{5.1.7}$$

where L_{22} and K are lower triangular and nonsingular.

Proof. Let E_k denote the $k \times k$ exchange matrix, that is, the identity matrix I_k with its columns in reverse order: $E_k = I_k(:, k{:}\,{-1}{:}\,1)$. Let

$$Q_B^T B^T = \begin{bmatrix} K^T \\ 0 \end{bmatrix}$$

be a QR factorization of B^T. The lower triangular matrix K is nonsingular since B has full rank. Write

$$E_m A Q_B = \begin{array}{cc} {\scriptstyle s} & {\scriptstyle n-s} \\ [\,\widetilde{A}_1 & \widetilde{A}_2\,] \end{array}$$

and set

$$J_E = \begin{bmatrix} I_p & 0 \\ 0 & -I_q \end{bmatrix}.$$

Note that $E_m = E_m^T$ and $E_m J_E E_m = J$. Now, from the second assumption in (5.1.4) it follows that $\widetilde{A}_2^T J_E \widetilde{A}_2$ is positive definite. Hence, using Theorem 2.3.2, there exists a hyperbolic QR factorization of $\widetilde{A}_2$

$$\widetilde{A}_2 = W \begin{bmatrix} R \\ 0 \end{bmatrix},$$

where R is a nonsingular upper triangular matrix and W is J_E-orthogonal. Moreover we have

$$E_m J W^T J \widetilde{A}_2 E_{n-s} = \begin{bmatrix} 0 \\ L \end{bmatrix},$$

where $L = E_{n-s} R E_{n-s}$ is lower triangular. Define

$$Q = Q_B \begin{bmatrix} I_s & 0 \\ 0 & E_{n-s} \end{bmatrix}, \quad H = E_m W E_m, \quad L_{22} = L.$$

Then (5.1.7) holds. $\square$

As we will show below, the generalized hyperbolic QR factorization plays a similar role for the ILSE problem as the generalized QR factorization plays for the standard LS problem with equality constraints.

The outline of this chapter is as follows. In Section 5.2 we give perturbation theory for the ILSE problem, obtaining a bound that is cheaply estimable but possibly non-sharp, and another bound based on Kronecker products that is sharp but more expensive to compute or estimate. In Section 5.3 we show how the ILSE problem can be solved using a generalized QR factorization and Cholesky factorization, in what is an extension of the method of Chandrasekaran, Gu and Sayed [14] for the ILS problem which we saw in Section 3.3.3. We prove that this method is mixed forward-backward stable. A method based on the generalized hyperbolic QR factorization is presented in Section 5.4 and the method is proved to be forward stable under a mild assumption. Numerical experiments are described in Section 5.5 that corroborate the error analysis.

5.2 Perturbation Theory

We begin by examining the sensitivity of the ILSE problem to perturbations in the data. We will consider perturbations ΔA, Δb, ΔB and Δd to the data A, b, B and d measured normwise by the smallest ϵ for which

$$\|\Delta A\|_F \le \epsilon \|A\|_F, \quad \|\Delta b\|_2 \le \epsilon \|b\|_2, \tag{5.2.1a}$$

$$\|\Delta B\|_F \le \epsilon \|B\|_F, \quad \|\Delta d\|_2 \le \epsilon \|d\|_2. \tag{5.2.1b}$$

We assume that the conditions (5.1.4) continue to hold for the perturbed problem.

The solution to the ILSE problem (5.1.1) satisfies the augmented system

$$\begin{bmatrix} 0 & 0 & B \\ 0 & J & A \\ B^T & A^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ s \\ x \end{bmatrix} = \begin{bmatrix} d \\ b \\ 0 \end{bmatrix}, \tag{5.2.2}$$

where $s = J(b - Ax) = Jr$, which is a rewritten version of the normal equations (5.1.5) with $\lambda = -\mu$. The augmented system for the perturbed problem is

$$\begin{bmatrix} 0 & 0 & B + \Delta B \\ 0 & J & A + \Delta A \\ B^T + \Delta B^T & A^T + \Delta A^T & 0 \end{bmatrix} \begin{bmatrix} \lambda + \Delta\lambda \\ s + \Delta s \\ x + \Delta x \end{bmatrix} = \begin{bmatrix} d + \Delta d \\ b + \Delta b \\ 0 \end{bmatrix},$$

which leads to the relationship for the perturbations

$$\begin{bmatrix} 0 & 0 & B \\ 0 & J & A \\ B^T & A^T & 0 \end{bmatrix} \begin{bmatrix} \Delta\lambda \\ \Delta s \\ \Delta x \end{bmatrix} = \begin{bmatrix} \Delta d - \Delta Bx \\ \Delta b - \Delta Ax \\ -\Delta B^T\lambda - \Delta A^T s \end{bmatrix} + O(\epsilon^2). \tag{5.2.3}$$

It is straightforward to verify that the inverse of the matrix on the left-hand side of (5.2.3) is

$$\begin{bmatrix} N^{-1} & -N^{-1}BM^{-1}A^TJ & N^{-1}BM^{-1} \\ -JAM^{-1}B^TN^{-1} & J + JAP^TM^{-1}A^TJ & -JAM^{-1}P \\ M^{-1}B^TN^{-1} & -P^TM^{-1}A^TJ & M^{-1}P \end{bmatrix},$$

where $M = A^TJA$, $N = BM^{-1}B^T$ and $P = B^TN^{-1}BM^{-1} - I$. Premultiplying by the inverse we obtain

$$\begin{aligned} \Delta x = {} & M^{-1}B^TN^{-1}(\Delta d - \Delta Bx) - P^TM^{-1}A^TJ(\Delta b - \Delta Ax) \\ & - M^{-1}P(\Delta B^T\lambda + \Delta A^Ts) + O(\epsilon^2). \end{aligned} \tag{5.2.4}$$

Solving for λ in the third equation of (5.2.2) we find that

$$\lambda = -(BB^T)^{-1}BA^TJr = -(B^+)^TA^TJr = -(AB^+)^TJr, \tag{5.2.5}$$

where B^+ is the pseudo-inverse of B. (See Section 1.6 for more on pseudo-inverses.)

Taking norms in (5.2.4) and using (5.2.5), we deduce that

$$\begin{aligned}\|\Delta x\|_2 \le \epsilon\Big[&\|M^{-1}B^TN^{-1}\|_2(\|d\|_2 + \|B\|_F\,\|x\|_2)\\ &+ \|P^TM^{-1}A^T\|_2(\|b\|_2 + \|A\|_F\,\|x\|_2)\\ &+ \|M^{-1}P\|_2(\|B\|_F\,\|AB^+\|_2\,\|r\|_2 + \|A\|_F\,\|r\|_2\end{aligned}$$

Hence, provided $x \neq 0$,

$$\begin{aligned}\frac{\|\Delta x\|_2}{\|x\|_2} \le \epsilon\Big[&\|M^{-1}B^TN^{-1}\|_2\,\|B\|_F\left(\frac{\|d\|_2}{\|B\|_F\,\|x\|_2} + 1\right)\\ &+ \|P^TM^{-1}A^T\|_2\,\|A\|_F\left(\frac{\|b\|_2}{\|A\|_F\,\|x\|_2} + 1\right)\\ &+ \|M^{-1}P\|_2\,\|A\|_F\left(\|B\|_F\,\|AB^+\|_2 + \|A\|_F\right)\frac{\|r\|_2}{\|A\|_F\,\|x\|_2}\Big]\\ &+ O(\epsilon^2).\end{aligned} \tag{5.2.6}$$

This bound shows that the sensitivity of the ILS problem is bounded in terms of $\|M^{-1}B^TN^{-1}\|_2\,\|B\|_F$ and $\|PM^{-1}A^T\|_2\,\|A\|_F$ when the residual is zero or small and $\|M^{-1}P\|_2\,\|A\|_F\,\|B\|_F\,\|AB^+\|_2$ otherwise.

The problem with (5.2.6) is that it is not very amenable to interpretation or computation. However, it is also possible to derive perturbation bounds by incorporating the generalized hyperbolic QR factorization. We will see that this method of deriving perturbation bounds is much simpler, and we can make the bound slightly tighter.

Let A and B have the generalized hyperbolic QR factorization

$$A = H\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22}\end{bmatrix}Q^T, \qquad BQ = [\,K \quad 0\,],$$

where H is J-orthogonal and Q is orthogonal. Premultiplying (5.2.3) by the matrix $\mathrm{diag}(I, JH^TJ, Q^T)$, we obtain

$$\begin{bmatrix} 0 & 0 & BQ \\ 0 & JH^TJHJ & JH^TJAQ \\ Q^TB^T & Q^TA^TJHJ & 0 \end{bmatrix}\begin{bmatrix}\Delta\lambda \\ H^T\Delta s \\ Q^T\Delta x\end{bmatrix} = \begin{bmatrix}\Delta d - \Delta Bx \\ JH^TJ(\Delta b - \Delta Ax) \\ -Q^T(\Delta B^T\lambda + \Delta A^Ts)\end{bmatrix} + O(\epsilon^2). \tag{5.2.7}$$

Using (2.1.1) and the generalized hyperbolic QR factorization the matrix on the left can be rewritten as

$$Z = \left[\begin{array}{c|cc|cc} 0 & 0 & 0 & K & 0 \\ \hline 0 & \widetilde{J} & 0 & L_{11} & 0 \\ 0 & 0 & I_{n-s} & L_{21} & L_{22} \\ \hline K^T & L_{11}^T & L_{21}^T & 0 & 0 \\ 0 & 0 & L_{22}^T & 0 & 0 \end{array}\right],$$

where $\widetilde{J} = \operatorname{diag}(-I_q, I_{p-(n-s)})$. The inverse of Z is

$$Z^{-1} = \left[\begin{array}{c|cc|cc} K^{-T}L_{11}^T\widetilde{J}L_{11}K^{-1} & -K^{-T}L_{11}^T\widetilde{J} & 0 & K^{-T} & -K^{-T}L_{21}^TL_{22}^{-T} \\ \hline -\widetilde{J}L_{11}K^{-1} & \widetilde{J} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & L_{22}^{-T} \\ \hline K^{-1} & 0 & 0 & 0 & 0 \\ -L_{22}^{-1}L_{21}K^{-1} & 0 & L_{22}^{-1} & 0 & -L_{22}^{-1}L_{22}^{-T} \end{array}\right]. \tag{5.2.8}$$

From (5.2.7) we therefore obtain

$$\begin{aligned} Q^T\Delta x = &\begin{bmatrix} I \\ -L_{22}^{-1}L_{21} \end{bmatrix} K^{-1}(\Delta d - \Delta Bx) + \begin{bmatrix} 0 & 0 \\ 0 & L_{22}^{-1} \end{bmatrix} JH^TJ(\Delta b - \Delta Ax) \\ &+ \begin{bmatrix} 0 & 0 \\ 0 & L_{22}^{-1}L_{22}^{-T} \end{bmatrix} Q^T(\Delta B^T\lambda + \Delta A^Ts) + O(\epsilon^2). \end{aligned} \tag{5.2.9}$$

The third equation in (5.2.2) is $B^T\lambda = -A^Ts$. Using the generalized hyperbolic QR factorization this equation can be written

$$\begin{bmatrix} K^T \\ 0 \end{bmatrix} \lambda = - \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & L_{22}^T \end{bmatrix} H^Ts. \tag{5.2.10}$$

Examining the second equation in (5.2.10), we can see that since L_{22} is nonsingular, we have $H(:, h+1\colon m)^Ts = 0$, and so the first equation in (5.2.10) gives us

$$\lambda = -K^{-T}L_{11}^TH(:, 1\colon h)^Ts.$$

Hence the penultimate term in (5.2.9) is

$$\begin{bmatrix} 0 & 0 \\ 0 & L_{22}^{-1}L_{22}^{-T} \end{bmatrix} Q^T(-\Delta B^T K^{-T} L_{11}^T H(:,1\colon h)^T + \Delta A^T)s. \tag{5.2.11}$$

Taking norms in (5.2.9) we therefore obtain the perturbation bound

$$\begin{aligned}\frac{\|\Delta x\|_2}{\|x\|_2} \le \epsilon \Bigg[\|B\|_F \left\| \begin{bmatrix} I \\ -L_{22}^{-1}L_{21} \end{bmatrix} K^{-1} \right\|_2 \left(\frac{\|d\|_2}{\|B\|_F \|x\|_2} + 1 \right) \\ + \|L_{22}^{-1}H(:,h+1\colon m)\|_2 \|A\|_F \left(\frac{\|b\|_2}{\|A\|_F \|x\|_2} + 1 \right) \\ + \|L_{22}^{-1}\|_2^2 \|A\|_F (\|B\|_F \|K^{-T}L_{11}^T H(:,1\colon h)^T\|_2 + \|A\|_F) \frac{\|r\|_2}{\|A\|_F \|x\|_2} \Bigg] \\ + O(\epsilon^2),\end{aligned} \tag{5.2.12}$$

where $h = m - (n-s)$. An interesting feature of this bound is that the first $m-(n-s)$ rows of H contribute to the third term of the bound and the last $n-s$ rows contribute to the second term. This bound includes as special cases bounds for standard LS problem, the equality constrained LS problem [20], [25], and the unconstrained ILS problem which we saw in Section 3.2.1.

An advantage of expressing (5.2.12) in terms of the generalized hyperbolic QR factorization is that only triangular matrices are inverted in the formula and hence the bound can be cheaply estimated using standard condition estimation techniques [42, Chapter 15].

As for the results in Section 3.2.1 and [20], it is unclear how close (5.2.12) is to being attainable for all sets of A, b, B and d. We obtain a sharp perturbation bound by making use of the vec operator and the Kronecker product (see Section 1.5), as in Section 3.2.1 and [20]. First, we rewrite (5.2.9), using (5.2.11), as

$$Q^T\Delta x = G_1\Delta d - G_1\Delta Bx + G_2\Delta b - G_2\Delta Ax + (G_3\Delta B^T G_4 - G_3\Delta A^T)s + O(\epsilon^2),$$

where the G_i notation is used to simplify the ensuing expressions. Applying the vec operator, using Lemma 1.5.1, and recalling that $s = Jr$, we obtain

$$Q^T\Delta x = G_1\Delta d - (x^T \otimes G_1)\mathrm{vec}(\Delta B) + G_2\Delta b - (x^T \otimes G_2)\mathrm{vec}(\Delta A)$$

$$+ (r^T J G_4^T \otimes G_3)\mathrm{vec}(\Delta B^T) - (r^T J \otimes G_3)\mathrm{vec}(\Delta A^T) + O(\epsilon^2).$$

Using the relations $\mathrm{vec}(\Delta A^T) = \Pi_1 \mathrm{vec}(\Delta A)$ and $\mathrm{vec}(\Delta B^T) = \Pi_2 \mathrm{vec}(\Delta B)$, where Π_1 and Π_2 are vec-permutation matrices, as introduced in Section 1.5,

$$\begin{aligned} Q^T \Delta x = G_1 \Delta d + G_2 \Delta b - [(x^T \otimes G_1) - (r^T J G_4^T \otimes G_3)\Pi_2]\mathrm{vec}(\Delta B) \\ - [(x^T \otimes G_2) + (r^T J \otimes G_3)\Pi_1]\mathrm{vec}(\Delta A) + O(\epsilon^2), \end{aligned}$$

which is expressed in terms of the four independent perturbations Δd, Δb, ΔA and ΔB. Now we take 2-norms. Using (5.2.1) and the relation (1.5.1), we deduce that

$$\frac{\|\Delta x\|_2}{\|x\|_2} \le \psi\epsilon + O(\epsilon^2), \tag{5.2.13}$$

where

$$\begin{aligned} \psi = \|x\|_2^{-1} \Big(\|G_1\|_2 \|d\|_2 + \|G_2\|_2 \|b\|_2 + \|(x^T \otimes G_1) - (r^T J G_4^T \otimes G_3)\Pi_2\|_2 \|B\|_F \\ + \|(x^T \otimes G_2) + (r^T J \otimes G_3)\Pi_1\|_2 \|A\|_F \Big), \end{aligned}$$

and this bound is attainable to within a factor 4. This bound will be used in the experiments of Section 5.5 to test the sharpness of (5.2.12).

5.3 The Generalized QR-Cholesky Method

Our first method for solving the ILSE problem does not involve any J-orthogonal transformations. It is an extension of the QR-Cholesky method (which we saw in Section 3.3.3) to handle equality constraints.

Consider a generalized QR factorization (see Section 1.11.3) of A and B

$$U^T A Q = \begin{array}{c c} & \begin{array}{cc} s & n-s \end{array} \\ \begin{array}{c} m-(n-s) \\ n-s \end{array} & \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \end{array}, \qquad BQ = \begin{array}{cc} \begin{array}{cc} s & n-s \end{array} & \\ [\, K \quad 0 \,] & s \end{array}, \tag{5.3.1}$$

where U and Q are orthogonal and L_{22} and K are lower triangular and, in view of (5.1.4), nonsingular. (Note that this factorization differs from (5.1.7) only in the U factor, which is replaced by a J-orthogonal matrix in (5.1.7).) Let

$$\begin{array}{c c} s & \\ n-s & \end{array}\!\!\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = y := Q^T x, \qquad C = \begin{array}{cc} s & n-s \\ \end{array}\!\![\, C_1 \quad C_2 \,] := AQ. \tag{5.3.2}$$

Then the constraint $Bx = d$ is equivalent to $Ky_1 = d$, which determines y_1. Thus

$$\begin{aligned} b - Ax &= b - AQy \\ &= b - [\, C_1 \quad C_2 \,]\, y \\ &= (b - C_1 y_1) - C_2 y_2 \\ &=: f - C_2 y_2, \end{aligned}$$

and so minimizing $(b - Ax)^T J(b - Ax)$ subject to $Bx = d$ is equivalent to the unconstrained ILS problem

$$\min_{y_2} (f - C_2 y_2)^T J(f - C_2 y_2). \tag{5.3.3}$$

It is easy to see that (5.1.4) implies that $C_2^T J C_2$ is positive definite. Noting from (5.3.2) and (5.3.1) that

$$C_2 = U \begin{bmatrix} 0 \\ L_{22} \end{bmatrix} =: \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix} \begin{bmatrix} 0 \\ L_{22} \end{bmatrix} = \begin{bmatrix} U_{12} \\ U_{22} \end{bmatrix} L_{22},$$

the normal equations $C_2^T J C_2 y_2 = C_2^T J f$ for (5.3.3) can be rewritten as

$$(U_{22}^T U_{22} - U_{12}^T U_{12}) L_{22} y_2 = \begin{bmatrix} U_{12} \\ U_{22} \end{bmatrix}^T Jf.$$

The matrix in parentheses is positive definite and so this system can be solved via a Cholesky factorization. Effectively, we are solving (5.3.3) by Algorithm 3.3.3. The entire procedure is summarized as follows.

Algorithm 5.3.1 (Generalized QR-Cholesky method) *This algorithm solves the ILSE problem* (5.1.1) *with the aid of a generalized QR factorization of A and B.*

```
Compute the generalized QR factorization (5.3.1).
Solve the lower triangular system $Ky_1 = d$.
$f = b - AQ(:,1\colon s)y_1$.
Form the symmetric matrix $W = U_{22}^T U_{22} - U_{12}^T U_{12}$.
Compute the Cholesky factorization $W = R^T R$.
Solve $R^T R L_{22} y_2 = [\,U_{12}^T \quad U_{22}^T\,]\, Jf$ by one back and
    two forward substitutions.
$x = Qy$.
```

Note that the Generalized QR-Cholesky method does not require Q to be formed explicitly, and only the last $n-s$ columns of U need to be formed explicitly. The operation count of the method is a complicated function of p, q, n and s, and when $m \gg n \gg s$ it is approximately $7mn^2$ flops, where a flop denotes one of the four elementary operations.

5.3.1 Error Analysis

The Generalized QR-Cholesky method has very satisfactory numerical stability properties. It is mixed forward-backward stable, in the sense that the computed solution is very close to the solution of a slightly perturbed problem. The next result makes this statement precise. We denote by $c_i(u)$ a term of the form $g(p, q, n, s)u + O(u^2)$, where g is a polynomial.

Theorem 5.3.2 *Suppose the Generalized QR-Cholesky method is implemented with the generalized QR factorization computed using Householder transformations. The computed solution $\widehat{x} = \overline{x} + \Delta\overline{x}$, where $\overline{x}$ solves a perturbed ILSE problem with data $A + \Delta A$, $B + \Delta B$, $b + \Delta b$ and d, where*

$$\|\Delta A\|_F \le c_1(u)\,\|A\|_F\,, \qquad \|\Delta B\|_F \le c_2(u)\,\|B\|_F\,,$$

$$\|\Delta b\|_2 \le c_3(u)\,\|b\|_2\,, \qquad \|\Delta\overline{x}\|_2 \le c_4(u)\,\|\overline{x}\|_2\,.$$

Proof. Standard error analysis of Householder transformations (Lemma 1.11.6 and Lemma 1.11.7) shows that the computed $\widehat{K}$ and $\widehat{C}$ satisfy

$$(B + \Delta B_1)\widetilde{Q} = [\,\widehat{K} \quad 0\,], \qquad \|\Delta B_1\|_F \le c_1(u)\,\|B\|_F\,,$$

$$\widehat{C} = (A + \Delta A_1)\widetilde{Q}, \qquad \|\Delta A_1\|_F \le c_2(u)\,\|A\|_F\,,$$

where $\widetilde{Q}$ is orthogonal. Using Lemma 1.12.1, we can see that the computed solution of the triangular system $\widehat{K}y_1 = d$ satisfies

$$(\widehat{K} + \Delta K)\widehat{y}_1 = d, \qquad \|\Delta K\|_F \le c_3(u)\|\widehat{K}\|_F.$$

The computed coefficient vector $\widehat{f} = fl(b - \widehat{C}_1\widehat{y}_1)$ of the ILS problem (5.3.3) satisfies

$$\widehat{f} = b + \Delta b_0 - (\widehat{C}_1 + \Delta\widehat{C}_0)\widehat{y}_1,$$

$$\|\Delta b_0\|_2 \le u\,\|b\|_2\,, \qquad \|\Delta\widehat{C}_0\|_F \le c_4(u)\|\widehat{C}_1\|_F.$$

(For simplicity, we are assuming that C_1 is explicitly formed and applied to y_1; the analysis remains valid if $C_1y_1 = AQ(:, 1\colon s)y_1$ is computed without forming C_1 or $Q(:, 1\colon s)$.) We can apply to (5.3.3) the error analysis of [14] to deduce that the computed $\widehat{y}_2$ solves

$$\min_{y_2}(\widehat{f} + \Delta f - (\widehat{C}_2 + \Delta\widehat{C}_2)y_2)^T J(\widehat{f} + \Delta f - (\widehat{C}_2 + \Delta\widehat{C}_2)y_2),$$

where

$$\|\Delta\widehat{C}_2\|_F \le c_5(u)\|\widehat{C}_2\|_F, \qquad \|\Delta f\|_2 \le c_6(u)\|\widehat{f}\|_F.$$

In view of the latter inequality we can write, using Lemma 1.11.8,

$$\widehat{f} + \Delta f = b + \Delta b - (\widehat{C}_1 + \Delta\widehat{C}_1)\widehat{y}_1,$$

$$\|\Delta b\|_2 \le c_7(u)\,\|b\|_2\,, \quad \|\Delta C_1\|_2 \le c_7(u)\|\widehat{C}_1\|_F.$$

Putting all these results together, we conclude that $\overline{x} = \widetilde{Q}\widehat{y}$ is the true solution to the perturbed problem with data $A + \Delta A$, $B + \Delta B$ and $b + \Delta b$, where

$$\Delta A = \Delta A_1 + [\,\Delta\widehat{C}_1 \quad \Delta\widehat{C}_2\,]\,\widetilde{Q}^T, \quad \Delta B = \Delta B_1 + [\,\Delta K \quad 0\,]\,\widetilde{Q}^T.$$

The bounds on $\|\Delta A\|_F$ and $\|\Delta B\|_F$ follow readily. Finally,

$$\widehat{x} = fl(Q\widehat{y}) = \widetilde{Q}\widehat{y} + \Delta\overline{x}, \qquad \|\Delta\overline{x}\|_2 \le c_8(u)\,\|\widehat{y}\|_2 = c_8(u)\,\|\overline{x}\|_2\,,$$

using Lemma 1.11.6 again. $\square$

5.4 The Generalized Hyperbolic QR Method

Another method for solving the ILSE problem can be derived using the generalized hyperbolic QR factorization introduced in Theorem 5.1.1. Using (5.1.7) the constraint $Bx = d$ can be written

$$Ky_1 = [\,K \quad 0\,]\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = d, \qquad y = Q^T x.$$

This triangular system uniquely determines y_1. Then, with $c = Hb$,

$$\begin{aligned}(b - Ax)^T J(b - Ax) &= (c - HAQy)^T J(c - HAQy) \\ &= \begin{bmatrix} c_1 - L_{11}y_1 \\ c_2 - L_{21}y_1 - L_{22}y_2 \end{bmatrix}^T J \begin{bmatrix} c_1 - L_{11}y_1 \\ c_2 - L_{21}y_1 - L_{22}y_2 \end{bmatrix} \\ &= \|c_2 - L_{21}y_1 - L_{22}y_2\|_2^2 \\ &\quad + (c_1 - L_{11}y_1)^T \widetilde{J}(c_1 - L_{11}y_1), \qquad (5.4.1)\end{aligned}$$

where $\widetilde{J} = J(1\colon m-(n-s), 1\colon m-(n-s)) = \operatorname{diag}(-I_q, I_{p-(n-s)})$. Therefore y_2 is uniquely determined as the solution of the triangular system $L_{22}y_2 = c_2 - L_{21}y_1$. This use of the generalized hyperbolic QR factorization forms the basis for our next method.

Algorithm 5.4.1 (GHQR method) *This algorithm solves the ILSE problem* (5.1.1) *using a generalized hyperbolic QR factorization of A and B.*

1. Compute the hyperbolic QR factorization of B^T:

   $BQ = B\,[\,Q_1 \quad Q_2\,] = [\,K \quad 0\,].$

2. Solve the lower triangular system $Ky_1 = d$.
3. $C_2 = AQ_2$.
4. Compute the hyperbolic QL factorization $HC_2 = [\,0 \quad L_{22}^T\,]^T$, where L_{22} is lower triangular and H is J-orthogonal, using the natural modification of the method of Section 2.3.
5. $f = [\,f_1^T \quad f_2^T\,]^T = H(b - AQ_1y_1)$ (compute by applying H in factored form).
6. Solve the lower triangular system $L_{22}y_2 = f_2$.
7. $x = Qy$.

Again, the cost of the method is a complicated function of p, q, n and s. For $m \gg n \gg s$ it is approximately $4mn^2$ flops, compared with $7mn^2$ flops for the Generalized QR-Cholesky method. It is clear that the GHQR method always requires fewer flops than the Generalized QR-Cholesky method, since the latter method requires significant additional computation beyond a generalized (hyperbolic) QR factorization while the former does not.

5.4.1 Error Analysis

We re-interpret the GHQR method in order to be able to make use of existing error analysis in Sections 2.3.4 and 4.2 for the ILS problem. The method essentially recasts the ILSE problem as the unconstrained ILS problem

$$\min_{y_2}(g - C_2y_2)^T J(g - C_2y_2), \qquad C_2 = AQ_2, \quad g = b - AQ_1y_1, \tag{5.4.2}$$

where y_1 is determined by the constraints, and then solves this problem by the hyperbolic QR factorization method in Chapter 4.

The analysis for steps 1–3 of Algorithm 5.4.1 is given in the proof of Theorem 5.3.2: we recall that

$$\begin{aligned}(B+\Delta B_1)\widetilde{Q} &= [\,\widehat{K} \quad 0\,], & \|\Delta B_1\|_F &\le c_1(u)\,\|B\|_F\,,\\ \widehat{C}_2 &= (A+\Delta A_2)\widetilde{Q}_2, & \|\Delta A_2\|_F &\le c_2(u)\,\|A\|_F\,,\\ (\widehat{K}+\Delta K)\widehat{y}_1 &= d, & \|\Delta K\|_F &\le c_3(u)\|\widehat{K}\|_F,\end{aligned}$$

where $\widetilde{Q} = [\,\widetilde{Q}_1 \quad \widetilde{Q}_2\,]$ is orthogonal. In addition,

$$\widehat{g} = b + \Delta b - (A+\Delta A_1)\widetilde{Q}_1\widehat{y}_1, \qquad \|\Delta A_1\|_F \le c_4(u)\,\|A\|_F\,, \quad \|\Delta b\|_2 \le u\,\|b\|_2\,.$$

The computed data $\widehat{g}$ and $\widehat{C}_2$ is therefore exact for the perturbed ILSE problem with data $A+\Delta A$, $B+\Delta B$ and $b+\Delta b$ with

$$\Delta A = [\,\Delta A_1\widetilde{Q}_1 \quad \Delta A_2\widetilde{Q}_2\,]\,\widetilde{Q}^T, \qquad \Delta B = \Delta B_1 + [\,\Delta K \quad 0\,]\,\widetilde{Q}^T.$$

Clearly, these are small normwise relative perturbations.

The analysis in Section 4.2 shows that, under an assumption on the attainability of a perturbation bound, (5.4.2) is solved in a forward stable manner (that is, the forward error is bounded in the same way as for a backward stable method), It is not too hard to see that a normwise relative perturbation of g and C_2 in (5.4.2) corresponds to a normwise relative perturbation of the same size of A, B and b in the original ILSE problem. Therefore forward errors introduced by a forward stable method applied to (5.4.2) are no larger than those that can be produced by small normwise relative perturbations in the ILSE problem.

Combining these two parts of analysis leads to the next result.

Theorem 5.4.2 *Under the assumption in Section 3.2.1 that a certain perturbation bound for the ILS problem is approximately attainable, the computed solution $\widehat{x}$ from the GHQR method is forward stable; therefore the forward error $\Delta x = \widehat{x} - x$ satisfies* (5.2.12) *with $\epsilon = c(u)$.* $\square$

5.5 Numerical Experiments

The aim of this section is to compare the predictions of the error analysis with the errors observed in practice. We do not have any way to test the mixed forward-backward stability of the Generalized QR-Cholesky method, so we concentrate on testing forward stability.

As well as the Generalized QR-Cholesky method (Algorithm 5.3.1) and the GHQR method (Algorithm 5.4.1), we also test the "augmented system method" that forms the augmented system (5.2.2) and solves it by LU factorization with partial pivoting. This method requires $2(s+m+n)^3/3$ flops, which is much more than the other two methods, but its ease of coding makes it of possible interest for small dimensions.

We generate test problems for which different terms of the perturbation bound (5.2.12) dominate. This is achieved by choosing the matrices in the generalized hyperbolic QR factorization (5.1.7) and thereby defining A and B. Crucial here are the conditioning of the triangular matrices L_{22} and K and the norm of H. The triangular matrices are generated via the QR factorizations of random matrices with pre-assigned singular value distributions (generated via MATLAB's `gallery('randsvd',...)`). Random J-orthogonal matrices H of specified norm are generated using the method of Higham [43]. The orthogonal factor Q in (5.1.7) is also generated randomly (via MATLAB's `gallery('qmult',...)`). Given that we know the generalized hyperbolic QR factorization we are able to control the size of the residual $r = b - Ax$ by choosing c_1 (and hence b) in (5.4.1) appropriately.

We show results for five different problems with $p = 8$, $q = 6$, $n = 6$ and $s = 4$ in Table 5.5.1. The first three columns show the relative errors $\|x - \widehat{x}\|_2 \,/\, \|x\|_2$ for the three methods of interest; for the true solution x we take a solution computed at high precision using MATLAB's Symbolic Math Toolbox. The fourth column tabulates the first order part of the perturbation bound (5.2.12), with $\epsilon = u$, and the next three columns show the sizes of the three first order terms in that bound (without the ϵ factor). The final column is the first order part of the Kronecker-based perturbation bound (5.2.13), with $\epsilon = u$, which we know is sharp.

Two main features are notable in the results. First, all three methods behave in a forward stable manner, since the errors are no larger than the bound (5.2.13). Our results are therefore consistent with our error analysis of the Generalized QR-Cholesky and GHQR methods. Second, the bound (5.2.12) is of similar size to (5.2.13), being at most a factor 10 bigger, showing that it is reasonably sharp in these tests.

Finally, we emphasize that computations with J-orthogonal matrices are delicate, and must be carefully arranged in order to avoid instability. To illustrate, we repeated the tests with step 5 of Algorithm 5.4.1 carried out by explicitly forming H and then multiplying the result into $b - A_1y_1$ (the algorithm requires H to be applied in factored form and this is done using the factored or mixed form of hyperbolic rotations used in Section 2.3.3). The results for the GHQR method were quantitatively unchanged *except* for the fourth test problem: here $\|H\|_2 \approx 10^6$ and the error was 2.4e0, so the algorithm performed unstably with this modified implementation.

Table 5.5.1: Errors for the three methods for solving the ILSE problem with corresponding values of perturbation bounds (5.2.12) and (5.2.13).

GQR-Cholesky	GHQR	Augmented system	Bound (5.2.12)	Terms	Bound (5.2.13)
2.2e-8	1.9e-8	1.1e-8	8.0e-6	3.2e3, 9.6e6, 7.2e10	7.1e-7
9.5e-14	9.7e-14	3.1e-13	1.0e-12	1.9e3, 7.4e3, 1.4e-9	1.0e-12
2.3e-8	2.3e-8	2.5e-9	2.3e-7	2.1e9, 4.6e3, 3.2e2	2.3e-7
1.2e-3	1.4e-3	2.7e-4	2.2e-2	6.8e2, 1.9e14, 7.8e11	2.2e-2
2.3e-7	2.2e-7	3.6e-7	8.1e-6	2.4e5, 7.3e10, 2.9e5	8.1e-6

Chapter 6

Conclusions

This thesis dealt with solving a generalization of the linear LS problem called the ILS problem. There were many obstacles that had to be alleviated in order to find a solution to the problem that was stable and quicker to compute than the backward stable QR-Cholesky method proposed by Chandrasekaran, Gu and Sayed [14].

After going through the necessary theoretical background in Chapter 1, we dealt with J-orthogonal transformations in Chapter 2. We looked at the properties of J-orthogonal matrices and we established a relation between J-orthogonal and orthogonal matrices using the exchange operator, as was done by Stewart and Stewart [74] in 1998. We showed how we could use this relation to help us prove stability of suitable application of certain J-orthogonal transformations, for which it is otherwise hard to prove stability.

We looked at hyperbolic rotations, the J-orthogonal counterpart of Givens rotations. We saw applications of hyperbolic rotations and how others have used them in their work. We looked at a total of six representations of hyperbolic rotations, and compared the stability in forming and applying them. Our first major observation came at this stage where we showed, using error analysis *and* numerical results, that it was essential that we formed the hyperbolic rotations in a certain way to guarantee accurately forming them, and that certain representations were not always stable. Also important was the way we formed products of hyperbolic rotations. We had to make use of their structure in a special way to prove stability of applying products of hyperbolic rotations to a matrix.

There is a lot more work that can be done to investigate the stability of hyperbolic rotations. There are obviously numerous other representations of hyperbolic rotations that can be investigated to see if they are more stable to form and apply. Rather misleading is the fact that many people who have used hyperbolic rotations in their work have only mentioned the standard representation HR1 in (2.2.5), and have not delved into the stability of individual hyperbolic rotations to show which representation is best to use. One area which might be of interest is that of applying *fast* hyperbolic rotations that don't use any divisions or square-roots, just like the fast Givens rotations [2], [3], [29], [35], [37]. Another important question that needs to be considered is that of overflow and underflow when forming the hyperbolic rotation according to the representation HR2. As we saw, HR3 and HR4 were scaled to avoid overflow and underflow, but we could not guarantee their stability when forming them.

We then showed the existence of the hyperbolic QR factorization, its links with the Cholesky downdating problem, and showed how to compute the factorization using non-overlapping J-orthogonal transformations. Also observed was the fact that applying the Q-factor in the hyperbolic QR factorization of a matrix after it has been explicitly computed could be more unstable than if it was left in factored form.

One major omission in this chapter was that of hyperbolic Householder transforms. This was due to the fact that it is not straightforward to prove stability for these transforms since it involves stably computing $v^T Jv$ accurately, where v is any vector, and J is as defined in (2.1.2). This was the only obstacle, and it is not clear how it is expected to be by-passed by studying the work carried out in, for example, [9], [11], [21], [49], [61], [70] and [74].

There are many other J-orthogonal matrix factorizations that can be studied. One of these is the hyperbolic CS Decomposition which we only stated in Chapter 1 and has been investigated by Higham [43] and Stewart and van Dooren [75].

In Chapter 3, we introduced the ILS problem. We looked at its applications to the solution of total least squares problems and to H^∞ smoothing. We gave perturbation theory for the ILS problem and found two perturbation bounds. One of these was to within a factor of two of the condition number of the ILS problem but was difficult to compute. The other was much easier to compute and we numerically showed that it was always of similar size to the other, more accurate, bound. We looked at three existing methods for computing the solution to the problem. These three methods were the solution of the normal equations of the ILS problem, solving the augmented system, and the QR-Cholesky method of Chandrasekaran, Gu and Sayed [14].

Chapter 4 dealt with solving the ILS problem using the hyperbolic QR factorization. This method is entirely analogous to Golub's method for solving the LS problem which he dealt with in [32]. We showed that the cost of solving the ILS problem using this method was the same as that for solving the LS problem, and considerably cheaper than the QR-Cholesky method. Also, it is more stable than solving both the normal equations and the augmented systems, and has a forward error comparable to that of the backward stable QR-Cholesky method, even when the problem is ill-conditioned.

We tried implementing iterative refinement to the computed solution of the ILS problem which we obtained by the HQR method, but we found that it was not improving the results. One way to improve this could be to use double precision in implementing iterative refinement.

In Chapter 5, we went one step further to solve the ILSE problem. We introduced the generalized hyperbolic QR factorization, and used it to solve the ILSE problem. Also, we extended the QR-Cholesky method to solve this problem as well. Perturbation theory was developed, and numerical results for both methods were producing forward errors within the perturbation bounds.

Another extension to solving the ILS problem where $A \in \mathbb{R}^{m \times n}$ has full rank is to solve the rank deficient case. The rank deficient LS problem can be solved using the singular value decomposition, so it might be possible to solve the ILS problem using the hyperbolic singular value decomposition. Its existence is discussed by Bojanczyk, Onn and Steinhardt [10] and Zha [82]. This decomposition is of interest in its own right, because it involves J-orthogonal transformations, and can be used to compute a solution to, amongst other problems, the symmetric nonsingular eigenvalue problem as discussed by Slapničar [66]. Slapničar and Truhar [67] also look at perturbation theory for the hyperbolic singular value problem.

Björck [7] and Lawson and Hanson [51] have both studied LS problems in detail. It would be very interesting to see how the methods discussed in those references can be adopted with minor changes for the ILS case. Examples of possible further study in this field could include solving weighted ILS problems, considering iterative methods for ILS problems and methods for not having to recompute the hyperbolic QR factorization when a row is added to or deleted from the original matrix A in (3.1.2), and how the solution of the new ILS problem can be computed efficiently.

Bibliography

[1] S. T. Alexander, C.-T. Pan, and R. J. Plemmons. Analysis of a recursive least squares hyperbolic rotation algorithm for signal processing. *Linear Algebra Appl.*, 98:3–40, 1988.

[2] Andrew A. Anda and Haesun Park. Fast plane rotations with dynamic scaling. *SIAM J. Matrix Anal. Appl.*, 15(1):162–174, 1994.

[3] Andrew A. Anda and Haesun Park. Self-scaling fast rotations for stiff and equality-constrained linear least squares problems. *Linear Algebra and its Applications*, 234(1–3):137–161, February 1996.

[4] E. Anderson, Z. Bai, C. H. Bischof, S. Blackford, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, and D. C. Sorensen. *LAPACK Users' Guide.* Third edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999. xxvi+407 pp. ISBN 0-89871-447-8.

[5] E. Anderson, Z. Bai, and J. Dongarra. Generalized QR factorization and its applications. *Linear Algebra Appl.*, 162–164:243–271, 1992.

[6] Adi Ben-Israel and Thomas N. E. Greville. *Generalized Inverses: Theory and Applications.* Wiley, New York, 1974.

[7] Åke Björck. *Numerical Methods for Least Squares Problems.* Society for

Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. xvii+408 pp. ISBN 0-89871-360-9 (paperback).

[8] A. W. Bojanczyk, R. P. Brent, P. van Dooren, and F. R. de Hoog. A note on downdating the Cholesky factorization. *SIAM J. Sci. Statist. Comput.*, 8(3):210–221, May 1987.

[9] Adam Bojanczyk, Sanzheng Qiao, and Allan O. Steinhardt. Unifying unitary and hyperbolic transformations. *Linear Algebra Appl.*, 316:183–197, 2000.

[10] Adam W. Bojanczyk, Ruth Onn, and Allan O. Steinhardt. Existence of the hyperbolic singular value decomposition. *Linear Algebra Appl.*, 185:21–30, May 1993.

[11] Adam W. Bojanczyk and Allan O. Steinhardt. Stability analysis of a Householder-based algorithm for downdating the Cholesky factorization. *SIAM J. Sci. Statist. Comput.*, 12(6):1255–1265, November 1991.

[12] Angelika Bunse-Gerstner. An analysis of the HR algorithm for computing the eigenvalues of a matrix. *Linear Algebra Appl.*, 35:155–173, 1981.

[13] Shivkumar Chandrasekaran, Gene H. Golub, Ming Gu, and Ali H. Sayed. An efficient algorithm for a bounded errors-in-variables model. *SIAM J. Matrix Anal. Appl.*, 20(4):839–859, 1999.

[14] Shivkumar Chandrasekaran, Ming Gu, and Ali H. Sayed. A stable and efficient algorithm for the indefinite linear least-squares problem. *SIAM J. Matrix Anal. Appl.*, 20(2):354–362, 1998.

[15] Shivkumar Chandrasekaran and Ali H. Sayed. Stabilizing the generalized Schur algorithm. *SIAM J. Matrix Anal. Appl.*, 17(4):950–983, 1996.

[16] Shivkumar Chandrasekaran and Ali H. Sayed. A fast stable solver for nonsymmetric Toeplitz and quasi-Toeplitz systems of linear equations. *SIAM J. Matrix Anal. Appl.*, 19(1):107–139, 1998.

[17] Shivkumar Chandrasekaran and Ali H. Sayed. Stabilized Schur algorithms. In *Fast Reliable Algorithms for Matrices with Structure*, T. Kailath and A. H. Sayed, editors, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

[18] J. Chun, T. Kailath, and H. Lev-Ari. Fast parallel algorithms for QR and triangular factorization. *SIAM J. Sci. Statist. Comput.*, 8(6):899–913, November 1987.

[19] Anthony Cox. *Stability of Algorithms for Solving Weighted and Constrained Least Squares Problems.* PhD thesis, University of Manchester, Oxford Road, Manchester, England, October 1997.

[20] Anthony J. Cox and Nicholas J. Higham. Accuracy and stability of the null space method for solving the equality constrained least squares problem. *BIT*, 39(1):34–50, 1999.

[21] G. Cybenko and M. Berry. Hyperbolic Householder algorithms for factoring structured matrices. *SIAM J. Matrix Anal. Appl.*, 11(4):499–520, October 1990.

[22] B. Danloy. On the choice of signs for Householder matrices. *J. Comput. Appl. Math.*, 2(1):67–69, 1976.

[23] Jean-Mare Delosme and Ilse C. F. Ipsen. Parallel solution of symmetric positive definite systems with hyperbolic rotations. *Linear Algebra Appl.*, 77:75–111, 1986.

[24] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1979. ISBN 0-89871-172-X.

[25] Lars Eldén. Perturbation theory for the least squares problem with linear equality constraints. *SIAM J. Numer. Anal.*, 17(3):338–350, 1980.

[26] Lars Eldén and Haesun Park. Perturbation analysis for block downdating of a Cholesky decomposition. *Numer. Math.*, 68(4):457–467, October 1994.

[27] Lars Eldén and Haesun Park. Perturbation and error analyses for block downdating of a Cholesky decomposition. *BIT*, 36(2):247–263, June 1996.

[28] Noel Gastinel. *Linear Numerical Analysis*. Kershaw Publishing, London, 1983. ix+341 pp. ISBN 0-901665-16-9.

[29] W. Morven Gentleman. Least squares computations by Givens transformations without square roots. *J. Inst. Maths. Applics.*, 12:329–336, 1973.

[30] I. Gohberg, Peter Lancaster, and Leiba Rodman. *Matrices and Indefinite Scalar Products*. Birkhäuser, Boston, MA, USA, July 1983. 302 pp. ISBN 376431527X(hardback).

[31] Moshe Goldberg and E. G. Straus. Multiplicativity of l_p norms for matrices. *Linear Algebra Appl.*, 52/53:351–360, 1983.

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

[33] Gene H. Golub. Matrix decompositions and statistical calculations. In *Statistical Computation*, Roy C. Milton and John A. Nelder, editors, Academic Press, London, 1969, pages 365–397.

[34] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Third edition, Johns Hopkins University Press, Baltimore, MD, USA, 1996. xxvii+694 pp. ISBN 0-8018-5413-X (hardback), 0-8018-5414-8 (paperback).

[35] J. Götze and U. Schwiegelshohn. A square root and division free Givens rotation for solving least squares problems on systolic arrays. *SIAM J. Matrix Anal. Appl.*, 12(4):800–807, 1991.

[36] S. J. Hammarling. The numerical solution of the general Gauss–Markov linear model. In *Mathematics in Signal Processing*, T. S. Durrani, J. B. Abbiss, and J. E. Hudson, editors, Oxford University Press, 1987, pages 451–456.

[37] Sven Hammarling. A note on modifications to the Givens plane rotation. *J. Inst. Maths. Applics.*, 13:215–218, 1974.

[38] Babak Hassibi, Ali H. Sayed, and Thomas Kailath. Linear estimation in Krein spaces—Part I: Theory. *IEEE Trans. Automat. Control*, 41(1):18–33, 1996.

[39] Babak Hassibi, Ali H. Sayed, and Thomas Kailath. *Indefinite-Quadratic Estimation and Control: a unified approach to H^2 and H^∞ smoothing.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999. ISBN 0-89871-411-7 (hardback).

[40] Harold V. Henderson and S. R. Searle. The vec-permutation matrix, the vec operator and Kronecker products: A review. *Linear and Multilinear Algebra*, 9:271–288, 1981.

[41] Nicholas J. Higham. The Matrix Computation Toolbox. `http://www.ma.man.ac.uk/~higham/mctoolbox`.

[42] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms.* Second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. xxx+680 pp. ISBN 0-89871-521-0.

[43] Nicholas J. Higham. J-orthogonal matrices: Properties and generation. Numerical Analysis Report No. 408, Manchester Centre for Computational Mathematics, Manchester, England, September 2002. 15 pp. `http://www.maths.man.ac.uk/~nareports/narep408.pdf`.

[44] Nicholas J. Higham. The Matrix Computation Toolbox for MATLAB (version 1.0). Numerical Analysis Report No. 410, Manchester Centre for Computational Mathematics, Manchester, England, August 2002. 19 pp. `http://www.maths.man.ac.uk/~nareports/narep410.pdf`.

[45] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, 1991. viii+607 pp. ISBN 0-521-30587-X.

[46] Alston S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comput. Mach.*, 5:339–342, 1958.

[47] S. F. Hsieh, K. J. R. Liu, and K. Yao. Dual-state systolic architectures for adaptive filtering using up/downdating RLS. Technical Report 47, Electrical Engineering Department, University of Maryland, College Park, MD, USA, 1991.

[48] Sabine Van Huffel and Joos Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1991. xiii+300 pp. ISBN 0-89871-275-0.

[49] Dáša Janovská and Gerhard Opfer. A note on hyperbolic transformations. *Numerical Linear Algebra with Applications*, 8:127–146, 2001.

[50] R. E. Kalman. Algebraic aspects of the generalized inverse of a rectangular matrix. In *Generalized Inverses and Applications*, M. Zuhair Nashed, editor, Academic Press, New York, 1976.

[51] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995. xii+337 pp. Revised republication of work first published in 1974 by Prentice-Hall. ISBN 0-89871-356-0.

[52] Nicola Mastronardi, Philippe Lemmerling, and Sabine van Huffel. Fast structured total least squares algorithm for solving the basic deconvolution problem. *SIAM J. Matrix Anal. Appl.*, 22(2):533–553, 2000.

[53] Ruth Onn, Allan O. Steinhardt, and Adam J. Bojanczyk. The hyperbolic singular value decomposition and applications. *IEEE Trans. Signal Processing*, 39(7):1575–1588, July 1991.

[54] C. C. Paige. Some aspects of generalized QR factorizations. In *Reliable Numerical Computation*, M. G. Cox and S. J. Hammarling, editors, Oxford University Press, 1990, pages 73–91.

[55] C. C. Paige and M. Wei. History and generality of the CS decomposition. *Linear Algebra Appl.*, 208/209:303–326, 1994.

[56] C. T. Pan and R. J. Plemmons. Least squares modifications with inverse factorizations: Parallel implementations. *J. Comput. Appl. Math.*, 27:109–127, 1989.

[57] C. T. Pan and Kermit Sigmon. A sharp bound for products of hyperbolic plane rotations. *SIAM J. Matrix Anal. Appl.*, 9(4):587–593, 10 1988.

[58] Haesun Park and Lars Eldén. Stability analysis and fast algorithms for triangularization of Toeplitz matrices. *Numer. Math.*, 76:383–402, 1997.

[59] Beresford N. Parlett. Analysis of algorithms for reflections in bisectors. *SIAM Rev.*, 12(2):197–208, 1971.

[60] Beresford N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1980. xix + 348 pp. ISBN 0-13-880047-2.

[61] Sanzheng Qiao. Unifying unitary and hyperbolic rotations and reflectors. Technical Report Technical Report No. 98-05, Department of Computer Science and Systems, McMaster University, Ontario, Canada, January 1998.

[62] Charles M. Rader and Allan O. Steinhardt. Hyperbolic Householder transforms. *IEEE Trans. Acoust., Speech, Signal Processing*, 34(6):1589–1602, 1986.

[63] Charles M. Rader and Allan O. Steinhardt. Hyperbolic Householder transforms. *SIAM J. Matrix Anal. Appl.*, 9(2):269–290, 1988.

[64] Ali H. Sayed, Babak Hassibi, and Thomas Kailath. Inertia properties of indefinite quadratic forms. *IEEE Signal Processing Letters*, 3(2):57–59, 1996.

[65] Sanja Singer and Saša Singer. Rounding-error and perturbation bounds for the indefinite QR factorization. *Linear Algebra Appl.*, 309:103–119, 2000.

[66] Ivan Slapničar. Highly accurate symmetric eigenvalue decomposition and hyperbolic SVD. *Linear Algebra Appl.*, 358:387–424, 2003.

[67] Ivan Slapničar and Ninoslav Truhar. Relative perturbation theory for hyperbolic singular value problem. *Linear Algebra Appl.*, 358:367–386, 2003.

[68] David E. Stewart. A new algorithm for the SVD of a long product of matrices and the stability of products. *Electron. Trans. Numer. Anal.*, 5:29–47, 1997.

[69] G. W. Stewart. The effects of rounding error on an algorithm for downdating a cholesky factorization. *J. Inst. Maths. Applics.*, 23:203–213, 1979.

[70] G. W. Stewart. On the stability of sequential updates and downdates. *IEEE Trans. Signal Processing*, 43(11):2642–2648, 1995.

[71] G. W. Stewart. *Matrix Algorithms. Volume I: Basic Decompositions.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1998. xx + 458 pp. ISBN 0-89871-414-1.

[72] G. W. Stewart. *Matrix Algorithms. Volume II: Eigensystems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001. xix+469 pp. ISBN 0-89871-503-2.

[73] G. W. Stewart and Ji guang Sun. *Matrix Perturbation Theory.* Academic Press, London, 1990.

[74] Michael Stewart and G. W. Stewart. On hyperbolic triangularization: Stability and pivoting. *SIAM J. Matrix Anal. Appl.*, 19(4):847–860, 1998.

[75] Michael Stewart and Paul van Dooren. On the application of Σ-orthgonal transformations with elementary hyperbolic rotations. Internal report, CESAME, Northeastern University, Boston, Massachusetts, 1996. 20 pp.

[76] Michael Stewart and Paul van Dooren. Stability issues in the factorization of structured matrices. *SIAM J. Matrix Anal. Appl.*, 18(1):104–118, March 1997.

[77] Françoise Tisseur. Tridiagonal-diagonal reduction of symmetric indefinite pairs. Numerical Analysis Report No. 409, Manchester Centre for Computational Mathematics, Manchester, England, September 2002. 20 pp. `http://www.maths.man.ac.uk/~nareports/narep409.pdf`.

[78] H. W. Turnbull and A. C. Aitken. *An Introduction to the Theory of Canonical Matrices.* Blackie, London and Glasgow, 1932. xiii+200 pp. Reprinted with appendix, 1952.

[79] Alle-Jan van der Veen. A Schur method for low-rank matrix approximation. *SIAM J. Matrix Anal. Appl.*, 17(1):139–160, January 1996.

[80] J. H. Wilkinson. *Rounding Errors in Algebraic Processes.* Notes on Applied Science No. 32, Her Majesty's Stationery Office, London, 1963. vi+161 pp. Also published by Prentice-Hall, Englewood Cliffs, NJ, USA. Reprinted by Dover, New York, 1994. ISBN 0-486-67999-3.

[81] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford University Press, 1965. xviii+662 pp. ISBN 0-19-853403-5 (hardback), 0-19-853418-3 (paperback).

[82] Hongyuan Zha. A note on the existence of the hyperbolic singular value decomposition. *Linear Algebra Appl.*, 240:199–205, 1996.