A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem

by

Inderjit Singh Dhillon

B.Tech. (Indian Institute of Technology, Bombay) 1989

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA, BERKELEY

Committee in charge:

Professor James W. Demmel, Chair
Professor Beresford N. Parlett
Professor Phil Colella

1997

The dissertation of Inderjit Singh Dhillon is approved:

Chair Date

Date

Date

University of California, Berkeley

1997

A New $\mathbf{O(n^2)}$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem

Abstract

A New $\mathbf{O(n^2)}$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem

by

Inderjit Singh Dhillon
Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor James W. Demmel, Chair

Computing the eigenvalues and orthogonal eigenvectors of an $n \times n$ symmetric tridiagonal matrix is an important task that arises while solving any symmetric eigenproblem. All practical software requires $O(n^3)$ time to compute all the eigenvectors and ensure their orthogonality when eigenvalues are close. In the first part of this thesis we review earlier work and show how some existing implementations of inverse iteration can fail in surprising ways.

The main contribution of this thesis is a new $O(n^2)$, easily parallelizable algorithm for solving the tridiagonal eigenproblem. Three main advances lead to our new algorithm. A tridiagonal matrix is traditionally represented by its diagonal and off-diagonal elements. Our most important advance is in recognizing that its bidiagonal factors are "better" for computational purposes. The use of bidiagonals enables us to invoke a relative criterion to judge when eigenvalues are "close". The second advance comes by using multiple bidiagonal factorizations in order to compute different eigenvectors independently of each other. Thirdly, we use carefully chosen dqds-like transformations as inner loops to compute eigenpairs that are highly accurate and "faithful" to the various bidiagonal representations. Orthogonality of the eigenvectors is a consequence of this accuracy. Only $O(n)$ work per eigenpair is needed by our new algorithm.

Conventional wisdom is that there is usually a trade-off between speed and accuracy in numerical procedures, i.e., higher accuracy can be achieved only at the expense of greater computing time. An interesting aspect of our work is that increased accuracy in the eigenvalues and eigenvectors obviates the need for explicit orthogonalization and leads to greater speed.

We present timing and accuracy results comparing a computer implementation of our new algorithm with four existing EISPACK and LAPACK software routines. Our test-bed contains a variety of tridiagonal matrices, some coming from quantum chemistry applications. The numerical results demonstrate the superiority of our new algorithm. For example, on a matrix of order 966 that occurs in the modeling of a biphenyl molecule our method is about 10 times faster than LAPACK's inverse iteration on a serial IBM RS/6000 processor and nearly 100 times faster on a 128 processor IBM SP2 parallel machine.

Professor James W. Demmel
Dissertation Committee Chair

To my parents,

for their constant love and support.

Contents

List of Figures **vi**

List of Tables **viii**

1 Setting the Scene **1**
1.1 Our Goals 2
1.2 Outline of Thesis 3
1.3 Notation 5

2 Existing Algorithms & their Drawbacks **7**
2.1 Background 7
2.2 The QR Algorithm 9
2.3 Bisection and Inverse Iteration 11
2.4 Divide and Conquer Methods 12
2.5 Other Methods 13
2.6 Comparison of Existing Methods 14
2.7 Issues in Inverse Iteration 16
2.8 Existing Implementations 20
2.8.1 EISPACK and LAPACK Inverse Iteration 22
2.9 Our Approach 32

3 Computing the eigenvector of an isolated eigenvalue **34**
3.1 Twisted Factorizations 38
3.2 The Eigenvector Connection 44
3.3 Zero Pivots 49
3.4 Avoiding Divisions 55
3.4.1 Heuristics for choosing r 56
3.5 Twisted Q Factorizations — A Digression* 57
3.5.1 "Perfect" Shifts are perfect 61
3.6 Rank Revealing Factorizations* 64

4 Computing orthogonal eigenvectors when relative gaps are large **68**
4.1 Benefits of High Accuracy 69
4.2 Tridiagonals Are Inadequate 70

4.3 Relative Perturbation Theory for Bidiagonals . . . 72
4.4 Using Products of Bidiagonals . . . 76
4.4.1 qd-like Recurrences . . . 78
4.4.2 Roundoff Error Analysis . . . 83
4.4.3 Algorithm X — orthogonality for large relative gaps . . . 93
4.5 Proof of Orthogonality . . . 94
4.5.1 A Requirement on r . . . 95
4.5.2 Outline of Argument . . . 97
4.5.3 Formal Proof . . . 99
4.5.4 Discussion of Error Bounds . . . 105
4.5.5 Orthogonality in Extended Precision Arithmetic . . . 107
4.6 Numerical Results . . . 108

5 Multiple Representations **111**
5.1 Multiple Representations . . . 112
5.2 Relatively Robust Representations (RRRs) . . . 115
5.2.1 Relative Condition Numbers . . . 116
5.2.2 Examples . . . 119
5.2.3 Factorizations of Nearly Singular Tridiagonals . . . 123
5.2.4 Other RRRs . . . 126
5.3 Orthogonality using Multiple Representations . . . 126
5.3.1 Representation Trees . . . 127
5.4 Algorithm Y — orthogonality even when relative gaps are small . . . 132

6 A Computer Implementation **134**
6.1 Forming an RRR . . . 135
6.2 Computing the Locally Small Eigenvalues . . . 136
6.3 An Enhancement using Submatrices . . . 138
6.4 Numerical Results . . . 139
6.4.1 Test Matrices . . . 140
6.4.2 Timing and Accuracy Results . . . 143
6.5 Future Enhancements to Algorithm Y . . . 152

7 Conclusions **155**
7.1 Future Work . . . 156

Bibliography **158**

A The need for accurate eigenvalues **169**

B Bidiagonals are Better **173**

C Multiple representations lead to orthogonality **177**

List of Figures

2.1 A typical implementation of Inverse Iteration to compute the jth eigenvector 20
2.2 Eigenvalue distribution in Example 21
2.3 TINVIT — EISPACK's implementation of Inverse Iteration 23
2.4 STEIN — LAPACK's implementation of Inverse Iteration 24

3.1 Twisted Triangular Factorization at $k = 3$ (the next elements to be annihilated are circled) 40
3.2 Twisted Q Factorization at $k = 3$ (the next elements to be annihilated are circled): forming N_k. 58
3.3 Twisted Triangular Factorization of a Hessenberg matrix at $k = 3$ (the next elements to be annihilated are circled) 65
3.4 Twisted Orthogonal Factorization of a Hessenberg matrix at $k = 3$ (the next elements to be annihilated are circled) 65
3.5 The above may be thought of as a twisted triangular or orthogonal factorization of a dense matrix at $k = 3$ (the next elements to be annihilated are circled) 66

4.1 Effects of roundoff — **dstqds** transformation 85
4.2 Effects of roundoff — **dqds** transformation 87
4.3 Effects of roundoff — **dtwqds** transformation 88
4.4 Effects of roundoff — LDL^T decomposition 90
4.5 **dtwqds** transformation applied to compute an eigenvector 97
4.6 An eigenvector of a tridiagonal : most of its entries are negligible 106

5.1 Representation Tree — Forming an extra RRR based at 1 129
5.2 Representation Tree — Only using the RRR based at 1 130
5.3 Representation Tree — An extra RRR based at 1 is essential 131

6.1 Eigenvalue distribution for Biphenyl 142
6.2 Eigenvalue distribution for $SiOSi_6$ 142
6.3 Eigenvalue distribution for Zeolite ZSM-5 142
6.4 Absolute and Relative Eigenvalue Gaps for Biphenyl 152
6.5 Absolute and Relative Eigenvalue Gaps for $SiOSi_6$ 152
6.6 Absolute and Relative Eigenvalue Gaps for Zeolite ZSM-5 153

C.1 The strong correlation between element growth and relative robustness . . . 181
C.2 Blow up of earlier figure : X-axis ranges from $1+10^{-8}$ to $1+3.5\times 10^{-8}$ . . 181
C.3 Relative condition numbers (in the figure to the right, X-axis ranges from $1+10^{-8}$ to $1+3.5\times 10^{-8}$) 182

List of Tables

2.1 Summary of xSTEIN's iterations to compute the second eigenvector of T. . . 31

4.1 Timing results on matrices of type 1 109
4.2 Accuracy results on matrices of type 1 109
4.3 Accuracy results on matrices of type 2 109

6.1 Timing Results 144
6.2 Timing Results 145
6.3 Maximum Residual Norms $\equiv \max_i \|T\hat{v}_i - \hat{\lambda}_i\hat{v}_i\|/n\varepsilon\|T\|$ 146
6.4 Maximum Residual Norms $\equiv \max_i \|T\hat{v}_i - \hat{\lambda}_i\hat{v}_i\|/n\varepsilon\|T\|$ 147
6.5 Maximum Dot Products $\equiv \max_{i\neq j} |\hat{v}_i^T\hat{v}_j|/n\varepsilon$ 148
6.6 Maximum Dot Products $\equiv \max_{i\neq j} |\hat{v}_i^T\hat{v}_j|/n\varepsilon$ 149

Acknowledgements

The author would like to express his extreme gratitude to Professor Beresford Parlett for making worthwhile the whole experience of working on this thesis. The author has benefited greatly from Professor Parlett's expertise, vision and clarity of thought in conducting research as well as in its presentation. Moreover, helpful discussions over many a lunch has greatly shaped this work and sparked off future ideas.

The author is grateful to Professor Demmel for introducing him to the field of numerical linear algebra. Professor Demmel's constant support, numerous suggestions and careful reading have played a great role in the development of this thesis. The author also wishes to thank Professor Kahan for much well-received advice, and Dr. George Fann for many helpful discussions.

The author is also grateful to many friends who have made life in Berkeley an enriching experience.

This research was supported in part by the Defense Advanced Research Projects Agency, Contract No. DAAL03-91-C-0047 through a subcontract with the University of Tennessee, the Department of Energy, Contract No. DOE-W-31-109-Eng-38 through a subcontract with Argonne National Laboratory, and by the Department of Energy, Grant No. DE-FG03-94ER25219, and the National Science Foundation Grant No. ASC-9313958, and NSF Infrastructure Grant No. CDA-9401156.

Part of this work was done when the author was supported by a fellowship, DOE Contract DE-AC06-76RLO 1830 through the Environmental Molecular Sciences construction project at Pacific Northwest National Laboratory (PNNL). The information presented here does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

Chapter 1

Setting the Scene

In this thesis, we propose a new algorithm for finding all or a subset of the eigenvalues and eigenvectors of a symmetric tridiagonal matrix. The main advance is in being able to compute numerically orthogonal "eigenvectors" without taking recourse to the Gram-Schmidt process or a similar technique that explicitly orthogonalizes vectors. All existing software for this problem needs to do such orthogonalization and hence takes $O(n^3)$ time in the worst case, where n is the order of the matrix. Our new algorithm is the result of several innovations which enable us to compute, in $O(n^2)$ time, eigenvectors that are highly accurate and numerically orthogonal as a consequence. We believe that the ideas behind our new algorithm can be gainfully applied to several other problems in numerical linear algebra.

As an example of the speedups possible due to our new algorithm, the parallel solution of a 966×966 dense symmetric eigenproblem, that comes from the modeling of a biphenyl molecule by the Møller-Plesset theory, is now nearly 3 times faster than an earlier implementation [39]. This speedup is a direct consequence of a 10-fold increase in speed of the tridiagonal solution, which previously accounted for 80-90% of the total time. Detailed numerical results are presented in Chapter 6.

Before we sketch an outline of our thesis, we list the features of an "ideal" algorithm.

1.1 Our Goals

At the onset of our research in 1993, we listed the desirable properties of the "ultimate" algorithm for computing the eigendecomposition of the symmetric tridiagonal matrix T. Our wish-list was for

1. **An $O(n^2)$ algorithm.** Such an algorithm would achieve the minimum output complexity in computing all the eigenvectors.

2. **An algorithm that guarantees accuracy.** Due to the limitations of finite precision, we cannot hope to compute the true eigenvalues and ensure that the computed eigenvectors are exactly orthogonal. A plausible goal is to find approximate eigenpairs $(\hat{\lambda}_i, \hat{v}_i)$, $\hat{v}_i^T \hat{v}_i = 1$, $i = 1, 2, \ldots, n$, such that

   - the residual norms are small, i.e.,

   $$\|(T - \hat{\lambda}_i I)\hat{v}_i\| \;=\; O(\varepsilon\|T\|), \tag{1.1.1}$$

   - the computed vectors are *numerically orthogonal*, i.e.,

   $$|\hat{v}_i^T \hat{v}_j| \;=\; O(\varepsilon), \quad i \neq j, \tag{1.1.2}$$

   where ε is the machine precision. For some discussion on how to relate the above goals to backward errors in T, see [87, Theorem 2.1] and [71]. However it may not be possible to achieve (1.1.1) and (1.1.2) in all cases and we will aim for bounds that grow slowly with n.

3. **An embarrassingly parallel algorithm** that allows independent computation of each eigenvalue and eigenvector making it easy to implement on a parallel computer.

4. **An adaptable algorithm** that permits computation of any k of the eigenvalues and eigenvectors at a reduced cost of $O(nk)$ operations.

We have almost succeeded in accomplishing all these lofty goals. The algorithm presented at the end of Chapter 5 is an $O(n^2)$, embarrassingly parallel and adaptable method that passes all our numerical tests. Work to further improve this algorithm is still ongoing and we believe we are very close to a provably accurate algorithm.

1.2 Outline of Thesis

The following summarizes the contents of this thesis.

1. In Chapter 2, we give some background explaining the importance of the symmetric tridiagonal eigenproblem. We then briefly describe some of the existing methods — the popular QR algorithm, bisection followed by inverse iteration and the relatively new divide and conquer approach. In Section 2.6, we compare the existing methods in terms of speed, accuracy and memory requirements discussing how they do not satisfy *all* the desirable goals of Section 1.1. Since our new algorithm is related to inverse iteration, in Section 2.7 we discuss in detail the tricky issues involved in its computer implementation. In Section 2.8.1, we show how the existing LAPACK and EISPACK inverse iteration software can fail. The expert reader may skip this chapter and move on to Chapter 3 where we start describing our new methods.

2. In Chapter 3, we describe in detail the computation of an eigenvector corresponding to an *isolated* eigenvalue that has already been computed. We begin by showing how some of the obvious ways can fail miserably due to the tridiagonal structure. Twisted factorizations, introduced in Section 3.1, are found to reveal singularity and provide an elegant mechanism for computing an eigenvector. The method suggested by these factorizations may be thought of as deterministically "picking a good starting vector" for inverse iteration thus avoiding the random choices currently used in LAPACK and solving a question posed by Wilkinson in [136, p.318]. Section 3.4 shows how to modify this new method in order to eliminate divisions. We then digress a little and in Sections 3.5 and 3.6 briefly discuss how twisted factorizations can be employed to reveal the rank of denser matrices and guarantee deflation in *perfect shift* strategies.

3. In Chapter 4, we show how to independently compute eigenvectors that turn out to be numerically orthogonal when eigenvalues differ in most of their digits. Note that eigenvalues may have tiny absolute gaps without agreeing in any digit, e.g., 10^{-50} and 10^{-51} are far apart from each other in a relative sense. We say that such eigenvalues have large relative gaps. The material of this chapter represents a major advance towards our goal of an $O(n^2)$ algorithm. Sections 4.1 and 4.2 extol the benefits of high accuracy in the computed eigenvalues, and show that for computational purposes, a bidiagonal factorization of T is "better" than the traditional way of representing T by

its diagonal and off-diagonal elements. In Section 4.3, we review the known properties of bidiagonal matrices that make them attractive for computation. Section 4.4.1 gives the *qd-like* recurrences that allow us to exploit the good properties of bidiagonals, and in Section 4.4.2 we give a detailed roundoff error analysis of their computer implementations. Section 4.5 gives a rigorous analysis that proves the numerical orthogonality of the computed eigenvectors when relative gaps are large. To conclude, we present a few numerical results in Section 4.6 to verify the above claims.

4. Chapter 5 deals with the case when relative gaps between the eigenvalues are small. In this chapter, we propose that for each such cluster of eigenvalues, we form an additional bidiagonal factorization of $T + \mu I$ where μ is close to the cluster, and then apply the techniques of Chapter 4 to compute eigenvectors that are automatically numerically orthogonal. The success of this approach depends on finding *relatively robust representations* that are defined in Section 5.2. Section 5.3.1 introduces the concept of a representation tree which is a tool that facilitates proving orthogonality of the vectors computed using different representations. We present Algorithm Y in Section 5.4 that also handles the remaining case of small relative gaps. We cannot prove the correctness of this algorithm as yet but extensive numerical experience indicates that it is accurate.

5. In Chapter 6 we give a detailed numerical comparison between Algorithm Y and four existing EISPACK and LAPACK software routines. Section 6.4.1 describes our extensive collection of test tridiagonals, some of which come from quantum chemistry applications. We find our computer implementation of Algorithm Y to be uniformly faster than existing implementations of inverse iteration and the QR algorithm. The speedups range from factors of 4 to about 3500 depending on the eigenvalue distribution. In Section 6.5, we speculate on further improvements to Algorithm Y.

6. Finally, in Chapter 7, we discuss how some of our techniques may be applicable to other problems in numerical linear algebra.

In addition to the above chapters, we have included some case studies at the end of this thesis, which examine various illuminating examples in detail. Much of the material in the case studies appears scattered in various chapters, but we have chosen to collate and expand on it at the end, where it can be read independently of the main text.

1.3 Notation

We now say a few words about our notation. The reader would benefit by occasionally reviewing this section during the course of his/her reading of this thesis. We adopt Householder's conventions and denote matrices by uppercase roman letters such as A, B, J, T and scalars by lowercase greek letters α, β, γ, η, or lowercase italic such as a_i, b_i etc. We also try to follow the Kahan/Parlett convention of denoting symmetric matrices by symmetric letters such as A, T and nonsymmetric matrices by B, J etc. In particular, T stands for a symmetric tridiagonal matrix while J denotes a nonsymmetric tridiagonal. However, we will occasionally depart from these principles, for example, the letters L and U will denote lower and upper triangular matrices respectively while D stands for a diagonal matrix by strong tradition. Overbars will be frequently used when more than one matrix of a particular type is being considered, e.g., L and $\tilde{L}$. The submatrix of T in rows i through j will be denoted by $T^{i:j}$ and its characteristic polynomial by $\chi^{i:j}$.

We denote vectors by lowercase roman letters such as u, v and z. The ith component of the vector v will be denoted by v_i or $v(i)$. The (i, j) element of matrix A will be denoted by A_{ij}. All vectors are n-dimensional and all matrices are $n \times n$ unless otherwise stated. In cases where there are at most n non-trivial entries in a matrix, we will use only one index to denote these matrix entries, for example $L(i)$ might denote the $L(i+1, i)$ entry of a unit lower bidiagonal matrix while $D(i)$ or D_i can denote the (i, i) element of a diagonal matrix. The ith column vector of the matrix V will be denoted by v_i. We note that this might lead to some ambiguity, but we will try and explicitly state our notation before such usage.

We denote the n eigenvalues of a matrix by $\lambda_1, \lambda_2, \cdots, \lambda_n$, while the n singular values are denoted by $\sigma_1, \sigma_2, \ldots, \sigma_n$. Normally we will assume that these quantities are ordered, i.e., $\lambda_1 \leq \cdots \leq \lambda_n$ while $\sigma_1 \geq \cdots \geq \sigma_n$. Note that the eigenvalues are arranged in increasing order while the singular values are in decreasing order. We have done this to abide by existing conventions. The ordering is immaterial to most of our presentation, but we will make explicit the order of arrangement whenever our exposition requires ordering. Eigenvectors and singular vectors will be denoted by v_i and u_i. The diagonal matrix of eigenvalues and singular values will be denoted by Λ and Σ respectively, while V and U will stand for matrices whose columns are eigenvectors and/or singular vectors.

Since finite precision computations are the driving force behind our work, we briefly introduce our model of arithmetic. We assume that the floating point result of a basic arithmetic operation $\circ$ satisfies

$$fl(x \circ y) = (x \circ y)(1 + \eta) = (x \circ y)/(1 + \delta)$$

where η and δ depend on x, y, $\circ$, and the arithmetic unit. The relative errors satisfy

$$|\eta| < \varepsilon, \quad |\delta| < \varepsilon$$

for a given ε that depends only on the arithmetic unit and will be called the *machine precision*. We shall choose freely the form (η or δ) that suits the analysis. We also adopt the convention of denoting the computed value of x by $\hat{x}$. In fact, we have already used some of this notation in Section 1.1.

The IEEE single and double precision formats allow for 24 and 53 bits of precision respectively. Thus the corresponding ε are $2^{-23} \approx 1.2 \times 10^{-7}$ and $2^{-52} \approx 2.2 \times 10^{-16}$ respectively. Whenever ε occurs in our analysis it will either denote machine precision or should be taken to mean "of the order of magnitude of" the machine precision, i.e, $\varepsilon = O(\text{machine precision})$.

Just as we did in the above sentence and in equations (1.1.1) and (1.1.2), we will continue to abuse the "big oh" notation. Normally the O notation, introduced by Bachmann in 1894 [68, Section 9.2], implies a limiting process. For example, when we say that an algorithm takes $O(n^2)$ time, we mean that the algorithm performs less than Kn^2 operations for some constant K as $n \to \infty$. However in our informal discussions, sometimes there will not be any limiting process or we may not always make it precise. In the former case, O will be a synonym for "of the order of magnitude of". Our usage should be clear from the context. Of course, we will be precise in the statements of our theorems — in fact, the O notation does not appear in any theorem or proof in this thesis.

We will also be sloppy in our usage of the terms "eigenvalues" and "eigenvectors". An unreduced symmetric tridiagonal matrix has exactly n distinct eigenvalues and n normalized eigenvectors that are mutually orthogonal. However, in several places we will use phrases like "the computed eigenvalues are close to the exact eigenvalues" and "the computed eigenvectors are not numerically orthogonal". In these phrases, we refer to *approximations* to the eigenvalues and eigenvectors and we are deliberately sloppy for the sake of brevity. In a similar vein, we will use "orthogonal" to occasionally mean numerically orthogonal, i.e., orthogonal to working precision.

Chapter 2

Existing Algorithms & their Drawbacks

In this chapter, we start by giving a quick background to the problem of computing an eigendecomposition of a dense symmetric matrix for the benefit of a newcomer. We then discuss and compare existing methods of solving the resulting tridiagonal problem in Sections 2.2 through 2.6. Later, in Sections 2.7 and 2.8, we show how various issues that arise in implementing inverse iteration are handled in existing LAPACK [1] and EISPACK [128] software, and present some examples where they fail to deliver correct answers. Finally, we sketch our alternate approach on handling these issues in Section 2.9.

2.1 Background

Eigenvalue computations arise in a rich variety of contexts. A quantum chemist may compute eigenvalues to reveal the electronic energy states in a large molecule, a structural engineer may need to construct a bridge whose natural frequencies of vibration lie outside the earthquake band while eigenvalues may convey information about the stability of a market to an economist. A large number of such physically meaningful problems may be posed as the abstract mathematical problem of finding all numbers λ and non-zero vectors q that satisfy the equation

$$Aq = \lambda q,$$

where A is a real, symmetric matrix of order n. λ is called an *eigenvalue* of the matrix A while q is a corresponding *eigenvector*.

All eigenvalues of A must satisfy the characteristic equation, $\det(A - \lambda I) = 0$ and since the left hand side of this equation is a polynomial in λ of degree n, A has exactly n eigenvalues. A symmetric matrix further enjoys the properties that

1. all eigenvalues are real, and

2. a complete set of n mutually orthogonal eigenvectors may be chosen.

Thus a symmetric matrix A admits the *eigendecomposition*

$$A = Q \Lambda Q^T,$$

where Λ is a diagonal matrix, $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, and Q is an *orthogonal* matrix, i.e., $Q^T Q = I$. In the case of an *unreduced* symmetric tridiagonal matrix, i.e., where all off-diagonal elements of the tridiagonal are non-zero, the eigenvalues are distinct while the eigenvectors are unique up to a scale factor and are mutually orthogonal.

Armed with this knowledge, several algorithms for computing the eigenvalues of a real, symmetric matrix have been constructed. Prior to the 1950s, explicitly forming and solving the characteristic equation seems to have been a popular choice. However, eigenvalues are extremely sensitive to small changes in the coefficients of the characteristic polynomial and the inadequacy of this representation became clear with the advent of the modern digital computer. Orthogonal matrices became, and still remain, the most important tools of the trade. A sequence of orthogonal transformations,

$$A_0 = A, \quad A_{i+1} = Q_i^T A_i Q_i,$$

where $Q_i^T Q_i = I$, is numerically stable and preserves the eigenvalues of A_i. Many algorithms for computing the eigendecomposition of A attempt to construct such a sequence so that A_i converges to a diagonal matrix. But, Galois' work in the nineteenth century implies that for $n > 4$ there can be no finite m for which A_m is diagonal as long as the Q_i are computed by algebraic expressions and taking kth roots. It seems natural to try and transform A to a tridiagonal matrix instead. In 1954, Givens proposed the reduction of A to tridiagonal form by using orthogonal plane rotations [62]. However, most current efficient algorithms work by reducing A to a tridiagonal matrix T by a sequence of $n-2$ orthogonal reflectors, now named after Householder who first introduced them in 1958, see [81]. Mathematically,

$$T = (Q_{n-3}^T \cdots Q_1^T Q_0^T) A (Q_0 Q_1 \cdots Q_{n-3}) = Z^T A Z$$

The eigendecomposition of T may now be found as

$$T = V \Lambda V^T, \qquad (2.1.1)$$

where $V^T V = I$, and *back-transformation* may be used to find the eigenvectors of A,

$$A = (ZV)\Lambda(ZV)^T = Q \Lambda Q^T.$$

The tridiagonal eigenproblem is one of the most intensively studied problems in numerical linear algebra. A variety of methods exploit the tridiagonal structure to compute (2.1.1). Extensive research has led to plenty of software, especially in the linear algebra software libraries, EISPACK [128] and the more recent LAPACK [1]. We will examine existing algorithms and related software in the next section.

We now briefly discuss the relative costs of the various components involved in solving the dense symmetric eigenproblem. Reducing A to tridiagonal form by Householder transformations costs about $\frac{4}{3}n^3$ multiplications and additions, while back-transformation to get the eigenvectors of A from the tridiagonal solution needs $2n^3$ multiplication and addition operations. The cost of solving the tridiagonal eigenproblem varies according to the method used and the numerical values in the matrix. If the distribution of eigenvalues is favorable some methods may solve such a tridiagonal problem in $O(n^2)$ time. However, all existing software takes kn^3 operations in the worst case, where k is a modest number that can vary from 4 to 12. In many cases of practical importance, the tridiagonal problem can indeed be the bottleneck in the total computation. The extent to which it is a bottleneck can be much greater than suggested by the above numbers because the other two phases, Householder reduction and back-transformation can exploit fast matrix-multiply based operations [12, 45], whereas most algorithms for the tridiagonal problem are sequential in nature and/or cannot be expressed in terms of matrix multiplication. We now discuss these existing algorithms.

2.2 The QR Algorithm

Till recently, the method of choice for the symmetric tridiagonal eigenproblem was the QR Algorithm which was independently invented by Francis [59] and Kublanovskaja [95]. The QR method is a remarkable iteration process,

$$T_0 = T,$$

$$\begin{aligned} T_i - \sigma_i I &= Q_i R_i, \\ T_{i+1} &= R_i Q_i + \sigma_i I, \quad i = 0, 1, 2, \ldots \end{aligned} \tag{2.2.2}$$

where $Q_i^T Q_i = I$, R_i is upper triangular and σ_i is a shift chosen to accelerate convergence. The off-diagonal elements of T_i are rapidly driven to zero by this process. Francis, helped by Strachey and Wilkinson, was the first to note the invariance of the tridiagonal form and incorporate shifts in the QR method. These observations make the method computationally viable. The initial success of the method sparked off an incredible amount of research into the QR method, which carries on till this day. Several shift strategies were proposed and convergence properties of the method studied. The ultimate cubic convergence of the QR algorithm with suitable shift strategies was observed by both Kahan and Wilkinson. In 1968, Wilkinson proved that the tridiagonal QL iteration (the QL method is intimately related to the QR method) always converges using his shift strategy. A simpler proof of global convergence is due to Hoffman and Parlett, see [79]. An excellent treatment of the QR method is given by Parlett in [110].

Each orthogonal matrix Q_i in (2.2.2) is a product of $n-1$ elementary rotations, known as Givens rotations [62]. The tridiagonal matrices T_i converge to diagonal form that gives the eigenvalues of T. The eigenvector matrix of T is then given by the product $Q_1 Q_2 Q_3 \cdots$. When only eigenvalues are desired, the QR transformation can be reorganized to eliminate all square roots that are required to form the Givens rotations. This was first observed by Ortega and Kaiser in 1963 [109] and a fast, stable algorithm was developed by Pal, Walker and Kahan (PWK) in 1968-69 [110]. Since a square root operation can be about 20 or more times as expensive as addition or multiplication, this yields a much faster method. In particular, the PWK algorithm finds all eigenvalues of T in approximately $9n^2$ multiply and add operations and $3n^2$ divisions, with the assumption that 2 QR iterations are needed per eigenvalue. However, when eigenvectors are desired, the product of all the Q_i must be accumulated during the algorithm. The $O(n^2)$ square root operations cannot be eliminated in this process and approximately $6n^3$ multiplications and additions are needed to find all the eigenvalues and eigenvectors of T. In the hope of cutting down this work by half, Parlett suggested the alternate strategy of computing the eigenvalues by the PWK algorithm, and then executing the QR algorithm using the previously computed eigenvalues as origin shifts to find the eigenvectors [110, p.173]. However this *perfect shift* strategy was not found to work much better than Wilkinson's shift strategy, taking an average of about

2 QR iterations per eigenvalue [69, 98].

2.3 Bisection and Inverse Iteration

In 1954, Wallace Givens proposed the method of bisection to find some or all of the eigenvalues of a real, symmetric matrix [62]. This method is based on the availability of a simple recurrence to count the number of eigenvalues less than a floating point number μ.

Let $\chi_j(\tau) = \det(\tau I - T^{1:j})$ be the characteristic polynomial of the leading principal $j \times j$ submatrix of T. The sequence $\{\chi_0, \chi_1, \ldots, \chi_n\}$, where $\chi_0 = 1$ and $\chi_1 = \mu - T_{11}$, forms a Sturm sequence of polynomials. The tridiagonal nature of T allows computation of $\chi_j(\mu)$ using the three-term recurrence

$$\chi_{j+1}(\mu) = (\mu - T_{j+1,j+1})\chi_j(\mu) - T_{j+1,j}^2 \chi_{j-1}(\mu). \tag{2.3.3}$$

The number of sign agreements in consecutive terms of the numerical sequence $\{\chi_i(\mu), i = 0, 1, \ldots, n\}$ equals the number of eigenvalues of T less than μ. Based on this recurrence, Givens devised a method that repeatedly halves the size of an interval that contains at least one eigenvalue. However, it was soon observed that recurrence (2.3.3) was prone to overflow with a limited exponent range. An alternate recurrence that computes $d_j(\mu) = \chi_j(\mu)/\chi_{j-1}(\mu)$ is now used in most software [89],

$$d_{j+1}(\mu) = (\mu - T_{j+1,j+1}) - T_{j+1,j}^2 / d_j(\mu), \quad d_1(\mu) = \mu - T_{11}.$$

The bisection algorithm permits an eigenvalue to be computed in about $2bn$ addition and bn division operations where b is the number of bits of precision in the numbers ($b = 24$ in IEEE single while $b = 53$ in IEEE double precision arithmetic). Thus all eigenvalues may be found in $O(bn^2)$ operations. Faster iterations that are superlinearly convergent can beat bisection and we give some references in Section 2.5.

Once an accurate eigenvalue approximation $\hat{\lambda}$ is known, the method of inverse iteration may be used to compute an approximate eigenvector [118, 87] :

$$v^{(0)} = b, \quad (A - \hat{\lambda} I)v^{(i+1)} = \tau^{(i)} v^{(i)}, \qquad i = 0, 1, 2, \ldots,$$

where b is the starting vector and $\tau^{(i)}$ is a scalar.

Earlier fears about loss of accuracy in solving the linear system given above due to the near singularity of $T - \hat{\lambda} I$ were allayed in [119]. Inverse iteration delivers a vector $\hat{v}$ that has a small residual, i.e. small $||(T - \hat{\lambda} I)\hat{v}||$, whenever $\hat{\lambda}$ is close to λ. However small residual norms do not guarantee orthogonality of the computed vectors when eigenvalues are close together. A commonly used "remedy" for clusters of eigenvalues is to orthogonalize each approximate eigenvector, as soon as it is computed, against previously computed eigenvectors in the cluster. Typical implementations orthogonalize using the modified Gram-Schmidt method.

The amount of work required by inverse iteration to compute all the eigenvectors of a symmetric tridiagonal matrix strongly depends upon the distribution of eigenvalues. If eigenvalues are well-separated, then $O(n^2)$ operations are sufficient. However, when eigenvalues are close, current implementations can take up to $10n^3$ operations due to the orthogonalization.

2.4 Divide and Conquer Methods

In 1981, Cuppen proposed a solution to the symmetric tridiagonal eigenproblem that was meant to be efficient for parallel computation [25, 46]. It is quite remarkable that this method can also be faster than other implementations on a serial computer!

The matrix T may be expressed as a modification of a direct sum of two smaller tridiagonal matrices. This modification may be a rank-one update [15], or may be obtained by crossing out a row and column of T [72]. The eigenproblem of T can then be solved in terms of the eigenproblems of the smaller tridiagonal matrices, and this may be done recursively. For several years after its inception, it was not known how to guarantee numerically orthogonality of the eigenvector approximations obtained by this approach. However in 1992, Gu and Eisenstat found a clever solution to this problem, and paved the way for robust software based on their algorithms [72, 73, 124] and Li's work on a faster zero-finder [102].

The main reason for the unexpected success of divide and conquer methods on serial machines is *deflation*, which occurs when an eigenpair of a submatrix of T is an acceptable eigenpair of a larger matrix. For symmetric tridiagonal matrices, this phenomenon is quite common. The greater the amount of deflation, the lesser is the work required in these methods. The amount of deflation depends on the distribution of eigenvalues and on the structure of the eigenvectors. In the worst case when no deflation occurs $O(n^3)$ operations are needed, but on matrices where eigenvalues cluster and the eigenvector matrix contains many tiny entries, substantial deflation occurs and many fewer than $O(n^3)$ operations are required [73].

In [71, 73], Gu and Eisenstat show that by using the fast multipole method of Carrier, Greengard and Rokhlin [70, 16], the complexity of solving the symmetric tridiagonal eigenproblem can be considerably lowered. All the eigenvalues and eigenvectors can be found in $O(n^2)$ operations while all the eigenvalues can be computed in $O(n \log_2 n)$ operations. In fact, the latter method also finds the eigenvectors but in an implicit factored form (without assembling the n^2 entries of all the eigenvectors) that allows multiplication of the eigenvector matrix by a vector in about n^2 operations. However, the constant factor in the above operation counts is quite high, and matrices encountered currently are not large enough for these methods to be viable. There is no software available as yet that uses the fast multipole method for the eigenproblem.

2.5 Other Methods

The oldest method for solving the symmetric eigenproblem is one due to Jacobi that dates back to 1846 [86], and was rediscovered by von Neumann and colleagues in 1946 [7]. Jacobi's method does not reduce the dense symmetric matrix A to tridiagonal form, as most other methods do, but instead works on A. It performs a sequence of plane rotations each of which annihilates an off-diagonal element (which is filled in during later steps). There are a variety of Jacobi methods that differ solely in their strategies for choosing the next element to be annihilated. All good strategies tend to diminish the off-diagonal elements, and the resulting sequence of matrices converges to the diagonal matrix of eigenvalues. Jacobi methods cost $O(n^3)$ or more operations but the constant is larger than in any of the algorithms discussed above. Despite their slowness, these methods are still valuable as they seem to be more accurate than other methods [37]. They can also be quite fast on strongly diagonally dominant matrices.

The bisection algorithm discussed in Section 2.3 is a reliable way to compute eigenvalues. However, it can be quite slow and there have been many attempts to find faster zero-finders such as the Rayleigh Quotient Iteration [110], Laguerre's method [90, 113] and the Zeroin scheme [31, 13]. These zero-finders can considerably speed up the computation of isolated eigenvalues but they seem to stumble when eigenvalues cluster.

Homotopy methods for the symmetric eigenproblem were suggested by Chu in [23, 24]. These methods start from an eigenvalue of a simpler matrix D and follow a smooth curve to find an eigenvalue of $A(t) \equiv D + t(A - D)$. D was chosen to be the diagonal of the tridiagonal in [103], but greater success was obtained by taking D to be a direct sum of submatrices of T [105, 99]. An alternate divide and conquer method that finds the eigenvalues by using Laguerre's iteration instead of homotopy methods is given in [104]. The corresponding eigenvectors in these methods are obtained by inverse iteration.

2.6 Comparison of Existing Methods

All currently implemented software for finding all the eigenvalues and eigenvectors of a symmetric tridiagonal matrix requires $O(n^3)$ work in the worst case. The fastest current implementation is the divide and conquer method of [73]. As mentioned in Section 2.4, many fewer than $O(n^3)$ operations are needed when heavy deflation occurs. In fact, for some matrices, such as a small perturbation of the identity matrix, just $O(n)$ operations are sufficient to solve the eigenproblem. This method was designed to work well on parallel computers, offering both task and data parallelism [46]. Efficient parallel implementations are not straightforward to program, and the decision to switch from task to data parallelism depends on the characteristics of the underlying machine [17]. Due to such complications, all the currently available parallel software libraries, such as ScaLAPACK [22] and PeIGS [52], use algorithms based on bisection and inverse iteration. A drawback of the current divide and conquer software in LAPACK is that it needs extra workspace of more than $2n^2$ floating point numbers, which can be prohibitively excessive for large problems. Also, the divide and conquer algorithm does not allow the computation of a subset of the eigenvalues and eigenvectors at a proportionately reduced cost.

The bisection algorithm enables any subset of k eigenvalues to be computed with $O(nk)$ operations. Each eigenvalue can be found independently and this makes it suitable for parallel computation. However, bisection is slow if all the eigenvalues are needed. A faster root-finder, such as Zeroin [31, 13], speeds up computation when an eigenvalue is isolated in an interval. Multisection maintains the simplicity of bisection and in certain situations, can speed up the performance on a parallel machine [106, 11, 127]. When the k eigenvalues are well separated, inverse iteration can find the eigenvectors independently, each in $O(n)$ time. However, to find eigenvectors of k close eigenvalues, all existing implementations resort to reorthogonalization and this costs $O(nk^2)$ operations. Orthogonalization can also lead to heavy communication in a parallel implementation. The computed eigenvectors may also not be accurate enough in some rare cases. See Section 2.8.1 for more details. Despite these drawbacks, the embarrassingly parallel nature of bisection followed by inverse iteration makes it easy and efficient to program on a parallel computer. As a result, this is the method that has been implemented in current software libraries for distributed memory computers, such as ScaLAPACK [47], PeIGS [52, 54] and in [75].

Like the recent divide and conquer methods, the QR algorithm guarantees numerical orthogonality of the computed eigenvectors. The $O(n^2)$ computation performed in the QR method to find all the eigenvalues is sequential in nature and is not easily parallelized on modern parallel machines despite the attempts in [96, 132, 93]. However, the $O(n^3)$ computation in accumulating the Givens' rotations into the eigenvector matrix is trivially and efficiently parallelized, see [3] for more details. But the higher operation count and inability to exploit fast matrix-multiply based operations make the QR algorithm much slower than divide and conquer and also, slower on average than bisection followed by inverse iteration.

Prior to beginning this work, we were hopeful of finding an algorithm that could fulfil the rather ambitious goals of Section 1.1 because

- when eigenvalues are well separated, bisection followed by inverse iteration requires $O(n^2)$ operations, and
- when eigenvalues are clustered, the divide and conquer method is very fast.

The above observations suggest a hybrid algorithm that solves clusters by the divide and conquer algorithm and computes eigenvectors of isolated eigenvalues by inverse iteration. Indeed such an approach has been taken in [60]. Another alternative is to perform bisection and inverse iteration in higher precision. This may be achieved by simulating quadrupled precision, i.e., doubling the precision of the machine's native arithmetic, in software in an attempt to obviate the need for orthogonalization [38].

We choose to take a different approach in our thesis. By revisiting the problem at a more fundamental level, we have been able to arrive at an algorithm that shares the attractive features of inverse iteration and the divide and conquer method. Since our approach can be viewed as an alternate way of doing inverse iteration, we now look in detail at the issues involved in any implementation of inverse iteration. Although many surprising aspects of current implementations are revealed by our careful examination, the reader who is pressed for time may skip on to Chapter 3 for the main results of this thesis. The material in the upcoming sections also appears in [40].

2.7 Issues in Inverse Iteration

Inverse Iteration is a method to find an eigenvector when an approximate eigenvalue $\hat{\lambda}$ is known :

$$v^{(0)} = b, \quad (A - \hat{\lambda} I)v^{(i+1)} = \tau^{(i)} v^{(i)}, \qquad i = 0, 1, 2, \ldots. \tag{2.7.4}$$

Here b is the starting vector. Usually $\|v^{(i)}\| = 1$ and $\tau^{(i)}$ is a scalar chosen to try and make $\|v^{(i+1)}\| \approx 1$. We now list the key issues that arise in a computer implementation.

I. **Choice of shift.** Given an approximate eigenvalue $\hat{\lambda}$, what shift σ should be chosen when doing the inverse iteration step :

$$(A - \sigma I)v^{(i+1)} = \tau^{(i)} v^{(i)} \ ? \tag{2.7.5}$$

Should σ always equal $\hat{\lambda}$? How close should σ be to an eigenvalue? Should the accuracy of $\hat{\lambda}$ be checked?

II. **Direction of starting vector.** How should b be chosen?

III. **Scaling of right hand side.** When the shift σ is very close to an eigenvalue, $\|(A - \sigma I)^{-1}\|$ is large and solving (2.7.5) may result in overflow. Can $\tau^{(i)}$ be chosen to prevent overflow?

IV. **Convergence Criterion.** When does an iterate $v^{(i)}$ satisfy (1.1.1)? If the criterion for acceptance is too strict, the iteration may never stop and the danger of too loose a criterion is that poorer approximations than necessary may be accepted.

V. **Orthogonality.** Will the vectors for different eigenvalues computed by (2.7.4) be numerically orthogonal? If not, what steps must be taken to ensure orthogonality?

We now examine these issues in more detail. Before we do so, it is instructive to look at the first iterate of (2.7.5) in the eigenvector basis. Suppose that b is a starting vector with $\|b\|_2 = 1$, and that σ is an approximation to the eigenvalue λ_1. Writing b in terms of the eigenvectors, $b = \sum_{i=1}^{n} \xi_i v_i$, we get, in exact arithmetic

$$\hat{v}_1 = \tau^{(1)}(A - \sigma I)^{-1} b = \tau^{(1)} \left(\frac{\xi_1}{\lambda_1 - \sigma} v_1 + \sum_{i=2}^{n} \frac{\xi_i}{\lambda_i - \sigma} v_i \right)$$

$$\Rightarrow \quad \hat{v}_1 \; = \; \frac{\xi_1 \tau^{(1)}}{\lambda_1 - \sigma} \left(v_1 + \sum_{i=2}^{n} \frac{\xi_i}{\xi_1} \frac{\lambda_1 - \sigma}{\lambda_i - \sigma} v_i \right). \tag{2.7.6}$$

I. **Choice of shift.** The above equation shows that for $\hat{v}_1$ to be a good approximation to v_1, σ must be close to λ_1. But in such a case, the linear system (2.7.5) is ill-conditioned and small changes in σ or A can lead to large changes in the solution $v^{(i+1)}$. Initially, it was feared that roundoff error would destroy these calculations in finite precision arithmetic. However Wilkinson showed that the errors made in computing $v^{(i+1)}$, although large, are almost entirely in the direction of v_1 when λ_1 is isolated. Since we are interested only in computing the direction of v_1 these errors pose no danger, see [119]. Thus to compute the eigenvector of an isolated eigenvalue, the more accurate the shift is the better is the approximate eigenvector.

It is common practice now to compute eigenvalues first, and then invoke inverse iteration with very accurate σ. Due to the fundamental limitations of finite precision arithmetic, eigenvalues of symmetric matrixes can, in general, only be computed to a guaranteed accuracy of $O(\varepsilon\|A\|)$ [110]. Even when a very accurate eigenvalue approximation is available, the following may influence the choice of the shift when more than one eigenvector is desired.

- **The pairing problem.** In [20], Chandrasekaran gives a surprising example showing how inverse iteration can fail to give small residuals in exact arithmetic if the eigenvalues and eigenvectors are not paired up properly. We reproduce the example in Section 2.8. To prevent such an occurrence, Chandrasekaran proposes perturbing the eigenvalue approximations so that each shift used for inverse iteration lies to the left of, i.e., is smaller than, its nearest eigenvalue (see Example 2.8.1 for more details).

- **The separation problem.** The solution $v^{(i+1)}$ in (2.7.5) is very sensitive to small changes in σ when there is more than one eigenvalue near σ. In [136, p.329], Wilkinson notes that

  > 'The extreme sensitivity of the computed eigenvector to very small changes in λ [σ in our notation] may be turned to practical advantage and used to obtain independent eigenvectors corresponding to coincident or pathologically close eigenvalues'.

  Wilkinson proposed that such nearby eigenvalues be 'artificially separated' by a tiny amount.

II. **Direction of starting vector.** From (2.7.6), assuming that $|\lambda_1 - \sigma| \ll |\lambda_i - \sigma|$ for $i \neq 1$, $\hat{v}_1$ is a good approximation to v_1 provided that ξ_1 is not "negligible", i.e., the starting vector must have a non-negligible component in the direction of the desired eigenvector. In [136, pp.315-321], Wilkinson investigates and rejects the choice of e_1 or e_n as a starting vector (where e_i is the ith column of the $n \times n$ identity matrix). e_k is a desirable choice for a starting vector if the kth component of v_1 is above average ($> 1/\sqrt{n}$). In the absence of an efficient procedure to find such a k, Wilkinson proposed choosing PLe as the starting vector, where $T - \sigma I = PLU$ and e is the vector of all 1's [134, 136]. A random starting vector is a popular choice since the probability that it has a negligible component in the desired direction is extremely low, see [87] for a detailed study.

III. **Scaling of right hand side.** Equation (2.7.6) implies that $\|\hat{v}_1\| = O(|\tau^{(1)}/(\lambda_1 - \sigma)|)$ where $\tau^{(1)}$ is the scale factor in the first iteration of (2.7.5). If σ is very close to an eigenvalue, $\|\hat{v}_1\|$ can be very large and overflow may occur and lead to breakdown in the eigenvector computation. To avoid such overflow, $\tau^{(1)}$ should be chosen appropriately to scale down the right hand side. This approach is taken in EISPACK and LAPACK.

IV. **Convergence Criterion.** In the iteration (2.7.4), when is $v^{(i+1)}$ an acceptable eigenvector? The residual norm is

$$\frac{\|(A - \hat{\lambda}_1 I)v^{(i+1)}\|}{\|v^{(i+1)}\|} = \frac{\|\tau^{(i)} \cdot v^{(i)}\|}{\|v^{(i+1)}\|}. \tag{2.7.7}$$

The factor $\|v^{(i+1)}\|/\|\tau^{(i)} \cdot v^{(i)}\|$ is called the norm growth. To guarantee (1.1.1), $v^{(i+1)}$ is usually accepted when the norm growth is $O(1/n\varepsilon\|T\|)$, see [136, p.324] for details. For the basic iteration of (2.7.4) this convergence criterion can always be met in a few iterations, provided the starting vector is not pathologically deficient in the desired eigenvector and $|\lambda_1 - \hat{\lambda}_1| = O(n\varepsilon\|T\|)$. As we have mentioned before, these requirements are easily met.

Since the eigenvalue approximations are generally input to inverse iteration, what should the software do if the input approximations are not accurate, i.e., bad data is input to inverse iteration? We believe that the software should raise some sort of error flag either by testing for the accuracy of the input eigenvalues, or through non-convergence of the iterates.

When λ_1 is isolated, a small residual implies orthogonality of the computed vector to other eigenvectors (see (2.7.8) below). However when λ_1 is in a cluster, goal (1.1.2) is not automatic. As we now discuss, the methods used to compute numerically orthogonal vectors can impact the choice of the convergence criterion.

V. **Orthogonality.** Standard perturbation theory [110, Section 11-7] says that if $\hat{v}$ is a unit vector, λ is the eigenvalue closest to $\hat{\lambda}$ and v is λ's eigenvector then

$$|\sin \angle(v, \hat{v})| \le \frac{\|A\hat{v} - \hat{\lambda}\hat{v}\|}{\mathrm{gap}(\hat{\lambda})} \tag{2.7.8}$$

where $\mathrm{gap}(\hat{\lambda}) = \min_{\lambda_i \neq \lambda} |\hat{\lambda} - \lambda_i|$.

In particular, the above implies that the simple iteration scheme of (2.7.4) cannot guarantee orthogonality of the computed "eigenvectors" when eigenvalues are close. To achieve numerical orthogonality, current implementations modify (2.7.4) by explicitly orthogonalizing each iterate against eigenvectors of nearby eigenvalues that have already been computed.

However, orthogonalization can fail if the vectors to be orthogonalized are close to being parallel. When this happens, two surprising difficulties arise :

- The orthogonalized vectors may not provide an orthogonal basis of the desired subspace.
- Orthogonalization may lead to cancellation and a decrease in norm of the iterate. Thus a simple convergence criterion (as suggested above in issue IV) may not be reached.

The eigenvalues found by the QR algorithm and the divide and conquer method can be in error by $O(\varepsilon\|T\|)$. As a result, approximations to small eigenvalues may not be correct in most of their digits. Thus computed eigenvalues may not be accurate "enough" leading to the above failures surprisingly often. We give examples of such occurrences in Section 2.8.1. In response to the above problems, Chandrasekaran proposes a new version of inverse iteration that is considerably different from the EISPACK and LAPACK implementations in [20]. The differences include an alternate convergence criterion. The drawback of this new version is the potential increase in the amount of computation required.

INVERSE ITERATION(A,$\hat{\lambda}$)

/* assume that $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_{j-1}$ have been computed, and $\hat{\lambda}_i, \hat{\lambda}_{i+1}, \ldots, \hat{\lambda}_j$ form a cluster */

```
Choose a starting vector b_j;
Orthogonalize b_j against v̂_i, v̂_{i+1}, ..., v̂_{j-1};
l = 0; v^(0) = b_j;
do
    l = l + 1;
    Solve (A − λ̂_j I) v^(l+1) = τ^(l) v^(l);
    Orthogonalize v^(l+1) against v̂_i, v̂_{i+1}, ..., v̂_{j-1};
while(||v^(l+1)|| / ||τ^(l) v^(l)|| is not "big" enough)
v̂_j = v^(l+1) / ||v^(l+1)||;
```

Figure 2.1: A typical implementation of Inverse Iteration to compute the jth eigenvector

2.8 Existing Implementations

Figure 2.1 gives the pseudocode for a typical implementation of inverse iteration to compute v_j, the jth eigenvector, assuming that $v_1, v_2, \ldots, v_{j-1}$ have already been computed. Note that in this pseudocode, both the starting vectors and iterates are orthogonalized against previously computed eigenvectors. Surprisingly, as the following example shows, this implementation can fail to give small residual norms *even in exact arithmetic* by incorrectly pairing up the eigenvalues and eigenvectors.

Example 2.8.1 [The Pairing Error.] (Chandrasekaran [20]) Let λ_1 be an arbitrary real number, and

$$\lambda_2 = \lambda_1 + \varepsilon, \quad \lambda_{i+1} - \lambda_i = \lambda_i - \lambda_1, \quad \lambda_n \geq 1, \quad i = 2, \ldots, n-1$$

where ε is of the order of the machine precision. Explicitly, $\lambda_i = \lambda_1 + 2^{i-2}\varepsilon$. Suppose that

$$\hat{\lambda}_i > \lambda_i, \quad i = 1, \ldots, n \qquad \text{and most importantly} \qquad \hat{\lambda}_1 - \lambda_1 > \lambda_2 - \hat{\lambda}_1.$$

Figure 2.2 illustrates the situation.

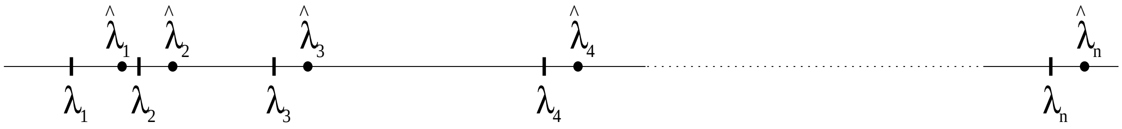

Figure 2.2: Eigenvalue distribution in Example

Assume that in Figure 2.1 each b_j is orthogonalized against $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_{i-1}$. If $b_j^T v_{j+1} \neq 0$, then in exact arithmetic the computed eigenvectors are

$$\hat{v}_i = v_{i+1}, \quad i = 1, \ldots, n-1$$

and, because $\hat{v}_n$ must be orthogonal to $\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_{n-1}$,

$$\hat{v}_n = v_1.$$

Since the eigenvalues grow exponentially, the residual norm $\|(A - \hat{\lambda}_n I)\hat{v}_n\|$ is large! This is because even though the eigenvectors have been computed correctly, each is associated with the wrong eigenvalue. □

Hence, a simple inverse iteration code based on orthogonalization may appear to fail even in exact arithmetic. To cure this problem, Chandrasekaran proposes that $\hat{\lambda}_i - O(n\varepsilon\|A\|)$ be used as the shifts for inverse iteration so that all shifts are guaranteed to lie to the left of the actual eigenvalues [20]. Neither EISPACK nor LAPACK do this 'artificial' perturbation.

The discerning reader will realize that the above problem is not the failure of the basic inverse iteration process. Iterates do converge to the closest eigenvector that is orthogonal to the eigenvectors computed earlier. The error is elusive but once seen, it may be argued that the implementation in Figure 2.1 is sloppy. An easy cure would be to associate with each computed eigenvector its Rayleigh Quotient, which is available at a modest cost. Unfortunately, because of the premium on speed, most current software does not check if its output is correct. Thus, errors can go undetected since the task of proving correctness of numerical software is often compromised by testing it on a finite sample of a multi-dimensional infinite space of inputs.

We now look in detail at two existing implementations of inverse iteration and see how they address the issues discussed in the previous section. EISPACK [128] and

LAPACK [1] are linear algebra software libraries that contain routines to solve various eigenvalue problems. EISPACK's implementation of inverse iteration is named TINVIT while LAPACK's inverse iteration subroutine is called xSTEIN[1] (STEIN is an acronym for Symmetric Tridiagonal's Eigenvectors through Inverse Iteration). xSTEIN was developed to be more accurate than TINVIT as the latter was found to deliver less than satisfactory results in several test cases. In order to achieve accuracy comparable to that of the divide and conquer and QR/QL methods, the search for a better implementation of inverse iteration led to xSTEIN [87]. However, as we will see in Section 2.8.1, xSTEIN also suffers from some of the same problems as TINVIT in addition to introducing a new serious error.

Both EISPACK and LAPACK solve the dense symmetric eigenproblem by reducing the dense matrix to tridiagonal form by Householder transformations [81], and then finding the eigenvalues and eigenvectors of the tridiagonal matrix. Both TINVIT and xSTEIN operate on a symmetric tridiagonal matrix. In the following, we will further assume that the tridiagonal is unreduced, i.e., all the off-diagonal elements are nonzero.

2.8.1 EISPACK and LAPACK Inverse Iteration

Figure 2.3 gives the pseudocode for TINVIT [128, 118] while Figure 2.4 outlines the pseudocode for xSTEIN as it appears in LAPACK release 2.0. The latter code has changed little since it was first released in 1992. It is not necessary for the reader to absorb all details of the implementations given in Figures 2.3 and 2.4 to follow the ensuing discussion. We provide the pseudocodes as references in case the reader needs to look in detail at a particular aspect of the implementations.

In each iteration, TINVIT and xSTEIN solve the scaled linear system $(T - \hat{\lambda} I)y = \tau b$ by Gaussian Elimination with partial pivoting. If eigenvalues agree in more than three digits relative to the norm, the iterates are orthogonalized against previously computed eigenvectors by the modified Gram-Schmidt method. Note that in both these routines the starting vector is not made orthogonal to previously computed eigenvectors, as is done in Figure 2.1. Both TINVIT and xSTEIN flag an error if the convergence criterion is not satisfied within five iterations. To achieve greater accuracy, xSTEIN does two extra iterations after the stopping criterion is satisfied. We now compare and contrast how these implementations handle the various issues discussed in Section 2.7.

[1] The prefix 'x' stands for the data type: real single(S) or real double(D), or complex single(C) or complex double(Z)

TINVIT(T,$\hat{\lambda}$)

/* TINVIT computes all the eigenvectors of T given the computed eigenvalues $\hat{\lambda}_1, \ldots, \hat{\lambda}_n$ */

for $j = 1, n$

  $\sigma_j = \hat{\lambda}_j$;

  if $j > 1$ **and** $\hat{\lambda}_j \leq \sigma_{j-1}$ **then**

    $\sigma_j = \sigma_{j-1} + \varepsilon \|T\|_R$; /* Perturb identical eigenvalues */

  end if

  Factor $T - \sigma_j I = PLU$; /* Gaussian Elimination with partial pivoting */

  if $U(n,n) = 0$ **then** $U(n,n) = \varepsilon \|T\|$;

  $\tau = \sqrt{n}\varepsilon \|T\|_R$; /* Compute scale factor */

  Solve $Uy = \tau e$; /* Solve. Here, e is the vector of all 1's */

  for all $k < j$ such that $|\sigma_j - \sigma_k| \leq 10^{-3} \|T\|_R$

    $y = y - (y^T \hat{v}_k)\hat{v}_k$; /* Apply Modified Gram-Schmidt */

  end for

  $b = y$; iter $= 1$;

  while($\|y\|_1 < 1$ **and** iter ≤ 5) **do**

    $\tau = n\varepsilon \|T\|_R / \|b\|_1$; /* Compute scale factor */

    Solve $PLUy = \tau b$; /* Solve with scaled right hand side */

    for all $k < j$ such that $|\sigma_j - \sigma_k| \leq 10^{-3} \|T\|_R$ **do**

      $y = y - (y^T \hat{v}_k)\hat{v}_k$; /* Apply Modified Gram-Schmidt */

    end for

    $b = y$; iter $=$ iter $+ 1$;

  end while

  if $\|y\|_1 < 1$ **then**

    $\hat{v}_j = 0$; $ierr = -j$; /* set error flag */

    print "jth eigenvector failed to converge";

  else

    $\hat{v}_j = y / \|y\|_2$;

  end if

end for

Figure 2.3: TINVIT — EISPACK's implementation of Inverse Iteration

```
xSTEIN(T, λ̂)
/* xSTEIN computes all the eigenvectors of T given the eigenvalues λ̂ */
for j = 1, n
    σ_j = λ̂_j;
    if j > 1 and λ̂_j − σ_{j−1} ≤ 10ε|λ̂_j| then
        σ_j = σ_{j−1} + 10ε|λ̂_j|; /* Perturb nearby eigenvalues */
    end if
    Factor T − σ_j I = PLU; /* Gaussian Elimination with partial pivoting */
    Initialize vector b with random vector;
    iter = 0; extra = 0; converged = false;
    do
        τ = n||T||_1 max(ε, |U(n, n)|)/||b||_1; /* Compute scale factor */
        Solve PLUy = τb; /* Solve with scaled right hand side */
        for all k < j such that |σ_j − σ_k| ≤ 10^{−3}||T||_1 do
            y = y − (y^T v̂_k)v̂_k; /* Apply Modified Gram-Schmidt */
        end for
        b = y; iter = iter + 1;
        if converged == true then extra = extra + 1; end if
        if ||y||_∞ ≥ sqrt(1/(10n)) then converged = true; end if
    while((converged == false or extra < 2) and iter ≤ 5)
    v̂_j = y/||y||_2;
    if iter == 5 and extra < 2 then
        print "jth eigenvector failed to converge";
    end if
end for
```

Figure 2.4: STEIN — LAPACK's implementation of Inverse Iteration

I. **Direction of starting vector.** TINVIT chooses the starting vector to be PLe where $T - \sigma I = PLU$, σ being an approximation to the eigenvalue, and e is the vector of all 1's. Note that this choice of starting vector reduces the first iteration to simply solving $Uy = \tau e$. On the other hand, xSTEIN chooses a random starting vector, each of whose elements comes from a uniform $(-1, 1)$ distribution. Neither choice of starting vectors is likely to be pathologically deficient in the desired eigenvector. The random starting vectors are designed to be superior to TINVIT's choice [87].

II. **Choice of shift.** Even though in exact arithmetic all the eigenvalues of an unreduced tridiagonal matrix are distinct, some of the computed eigenvalues may be identical to working accuracy. In [136, p.329], Wilkinson recommends that pathologically close eigenvalues be perturbed by a small amount in order to get an orthogonal basis of the desired subspace. Following this, TINVIT replaces the $k+1$ equal approximations $\hat{\lambda}_j = \hat{\lambda}_{j+1} = \cdots = \hat{\lambda}_{j+k}$ by

$$\hat{\lambda}_j < \hat{\lambda}_j + \varepsilon\|T\|_R < \cdots < \hat{\lambda}_j + k\varepsilon\|T\|_R,$$

where $\|T\|_R = \max_i |T_{ii}| + |T_{i,i+1}| \le \|T\|_1$.

We now give an example where this perturbation is too big. As a result, the shifts used to compute the eigenvectors are quite different from the computed eigenvalues and prevent the convergence criterion from being attained.

Example 2.8.2 [Excessive Perturbation.] Using LAPACK's test matrix generator [36], we generated a 200×200 tridiagonal matrix such that

$$\hat{\lambda}_1 \approx \cdots \approx \hat{\lambda}_{100} \approx -\varepsilon, \qquad \hat{\lambda}_{101} \approx \cdots \approx \hat{\lambda}_{199} \approx \varepsilon, \qquad \hat{\lambda}_n = 1$$

where $\varepsilon \approx 1.2 \times 10^{-7}$ (this run was in single precision). $\|T\|_R = O(1)$, and the shift used by TINVIT to compute $\hat{v}_{199}$ is

$$\sigma = -\varepsilon + 198\varepsilon\|T\|_R \approx 2.3 \times 10^{-5}.$$

Since $|\sigma - \lambda_{199}|$ can be as large as

$$|\sigma - \hat{\lambda}_{199}| + |\hat{\lambda}_{199} - \lambda_{199}| \approx 4.6 \times 10^{-5}$$

the norm growth when solving (2.8.9) is not big enough to meet the convergence criterion of (2.8.11). TINVIT flags this as an error and returns $ierr = -199$. □

Clearly, perturbing the computed eigenvalues relative to the norm can substantially decrease their accuracy. However, not perturbing them is also not acceptable in TINVIT as coincident shifts would lead to identical $L\ U$ factors, identical starting vectors and hence iterates that are parallel to the eigenvectors computed previously !

In xSTEIN, coincident eigenvalues are also perturbed. However, the perturbations made are relative. Equal approximations $\hat{\lambda}_j = \hat{\lambda}_{j+1} = \cdots = \hat{\lambda}_{j+k}$ are replaced by

$$\hat{\lambda}_j < \hat{\lambda}_{j+1} + \delta\hat{\lambda}_{j+1} < \cdots < \hat{\lambda}_{j+k} + \delta\hat{\lambda}_{j+k},$$

where $\delta\hat{\lambda}_i = \sum_{l=j}^{i-1} \delta\hat{\lambda}_l + 10\varepsilon|\hat{\lambda}_i|$. This choice does not perturb the small eigenvalues drastically, and appears to be better than TINVIT's. On the other hand, this perturbation is too small in some cases to serve its purpose of finding a linearly independent basis of the desired subspace (see Example 2.8.7 and Wilkinson's quote given on page 17). Thus it is easier to say "tweak close eigenvalues" than to find a satisfactory formula for it.

III. **Scaling of right hand side and convergence criterion.** For each $\hat{\lambda}$, the system

$$(T - \hat{\lambda}I)y = \tau b \tag{2.8.9}$$

is solved in each iteration. With TINVIT's choice of τ, the residual norm

$$\frac{\|(T - \hat{\lambda}I)y\|}{\|y\|} = \frac{\tau\|b\|}{\|y\|} = \frac{n\varepsilon\|T\|_R}{\|y\|}, \tag{2.8.10}$$

where $\|T\|_R = \max_i |T_{ii}| + |T_{i,i+1}| \leq \|T\|_1$. TINVIT accepts y as an eigenvector if

$$\|y\| \geq 1. \tag{2.8.11}$$

By (2.8.10), the criterion (2.8.11) ensures that goal (1.1.1) is satisfied, i.e., $\|(T - \hat{\lambda}I)y\|/\|y\| \leq n\varepsilon\|T\|_R$.

Suppose that in (2.8.9), $\hat{\lambda}$ is an approximation to λ_1. Then by analysis similar to (2.7.6),

$$\|y\| = O\left(\frac{\tau\|b\|}{|\lambda_1 - \hat{\lambda}|}\right) = O\left(\frac{n\varepsilon\|T\|_R}{|\lambda_1 - \hat{\lambda}|}\right).$$

Since $\|y\|$ is expected to be larger than 1 at some iteration, TINVIT requires that $\hat{\lambda}$ be such that $|\lambda_1 - \hat{\lambda}| = O(n\varepsilon\|T\|)$. If the input approximations to the eigenvalues are not accurate enough, the iterates do not "converge" and TINVIT flags an error.

When $\hat{\lambda}$ is very close to an eigenvalue, $\|y\|$ can be huge. TINVIT tries to avoid overflow by replacing a zero value of the last pivot in the PLU decomposition, u_{nn}, by $\varepsilon\|T\|$. However this measure cannot avoid failure in the following example.

Example 2.8.3 [Perturbing zero values is not enough.]

$$T = \begin{bmatrix} -\eta & 10 & 0 \\ 10 & 0 & 10 \\ 0 & 10 & \eta(1+\varepsilon) \end{bmatrix} \tag{2.8.12}$$

Here ε is the machine precision while η is the underflow threshold of the machine ($\eta \approx 10^{-308}$ in IEEE double precision arithmetic). T is nearly singular and

$$\hat{\lambda}_2 = 0 \quad \text{and partial pivoting} \quad \Rightarrow \quad u_{nn} = \eta\varepsilon.$$

Since $PLUy = \tau b$ and $\tau = n\varepsilon\|T\|_R/\|b\|$,

$$y(n) \;\approx\; \frac{\varepsilon\|T\|}{\eta\varepsilon} \;=\; \frac{10}{\eta} \quad \Rightarrow \quad \text{overflow!}$$

Note that to exhibit this failure, we required gradual underflow in IEEE arithmetic (the value $\eta\varepsilon$ should not underflow to zero). However gradual underflow is not necessary to exhibit such a failure. A similar error, where there is no gradual underflow, occurs on $(1/\varepsilon)T$ where T is as in (2.8.12). □

In xSTEIN, τ is chosen to be

$$\tau \;=\; \frac{n\|T\|_1 \max(\varepsilon, |u_{nn}|)}{\|b\|_1}, \tag{2.8.13}$$

where $T - \hat{\lambda}I = PLU$ and u_{nn} is the last diagonal element of U [85]. The significant difference between this scale factor and the one in TINVIT is the term $\max(\varepsilon, |u_{nn}|)$ instead of ε. xSTEIN accepts the iterate y as a computed eigenvector if

$$\|y\|_\infty \geq \sqrt{\frac{1}{10n}}. \tag{2.8.14}$$

The above choice of scale factor in xSTEIN introduces a serious error not present in TINVIT. Suppose $\hat{\lambda}$ approximates λ_1. When $\hat{\lambda} = \lambda_1$, it can be proved that u_{nn} must be zero in exact arithmetic. We now examine the values u_{nn} may take when $\hat{\lambda} \neq \lambda_1$. Since $T - \hat{\lambda}I = PLU$,

$$\begin{aligned} U^{-1}L^{-1} &= (T - \hat{\lambda}I)^{-1}P \\ \Rightarrow\ e_n^T U^{-1}L^{-1}e_n &= e_n^T(T - \hat{\lambda}I)^{-1}Pe_n \end{aligned}$$

Since L is unit lower triangular, $L^{-1}e_n = e_n$. Letting $Pe_n = e_k$ and $T = V\Lambda V^T$, we get

$$\begin{aligned} \frac{1}{u_{nn}} &= e_n^T V(\Lambda - \hat{\lambda}I)^{-1}V^T e_k \\ \Rightarrow\ \frac{1}{u_{nn}} &= \frac{v_{n1}v_{k1}}{\lambda_1 - \hat{\lambda}} + \sum_{i=2}^{n} \frac{v_{ni}v_{ki}}{\lambda_i - \hat{\lambda}}, \end{aligned} \tag{2.8.15}$$

where v_{ki} denotes the kth component of v_i. By examining the above equation, we see how the choice of scale factor in xSTEIN opens up a Pandora's box. Equation 2.8.15 says that for u_{nn} to be small, $|\hat{\lambda} - \lambda_1| \ll |v_{n1}v_{k1}|$.

Example 2.8.4 [A code may fail but should never lie.] Consider

$$T = \begin{bmatrix} 1 & \sqrt{\varepsilon} & 0 \\ \sqrt{\varepsilon} & 7\varepsilon/4 & \varepsilon/4 \\ 0 & \varepsilon/4 & 3\varepsilon/4 \end{bmatrix} \tag{2.8.16}$$

where ε is about machine precision ($\varepsilon \approx 2.2 \times 10^{-16}$ in IEEE double precision arithmetic). T has eigenvalues near $\varepsilon/2, \varepsilon, 1+\varepsilon$. Suppose, $\hat{\lambda}$ *is incorrectly input as* 2. Then by (2.8.13) and (2.8.15),

$$\hat{\lambda} = 2 \ \Rightarrow\ |u_{nn}| = O(1) \ \Rightarrow\ \tau = O(1)\,!$$

Clearly, this value of τ does not ensure a large norm growth when the stopping criterion (2.8.14) is satisfied in solving (2.8.9). As a result, any arbitrary vector can achieve the "convergence" criterion of (2.8.14) and be output as an approximate eigenvector. In a numerical run, the vector $[-0.6446\ \ 0.6373\ \ 0.4223]^T$ was accepted as an eigenvector by xSTEIN even though it is nowhere close to any eigenvector of T! $\square$

This example represents one of the more dangerous errors of numerical software — the software performs erroneous computation but does not flag any error at all. Failure to handle incorrect input data can have disastrous consequences [2]. On the above example, TINVIT correctly flags an error indicating that the computation did not "converge". Of course, most of the times the eigenvalues input to xSTEIN will be quite accurate and the above phenomenon will not occur.

Even if $\hat{\lambda}$ is a very good approximation to λ_1, (2.8.15) indicates that u_{nn} may not be small if v_{n1} is tiny. It is not at all uncommon for a component of an eigenvector of a tridiagonal matrix to be tiny [136, pp.317-321]. xSTEIN's choice of scale factor may lead to unnecessary overflow as shown below.

Example 2.8.5 [Undeserved overflow.] Consider the matrix given in (2.8.16). The eigenvector corresponding to the eigenvalue $\lambda_3 = 1 + \varepsilon + O(\varepsilon^2)$ is

$$v_3 = \begin{bmatrix} 1 - \varepsilon/2 + O(\varepsilon^3) \\ \sqrt{\varepsilon} + O(\varepsilon^{3/2}) \\ \varepsilon^{3/2}/4 + O(\varepsilon^{5/2}) \end{bmatrix}.$$

If $\hat{\lambda} = 1$, then $|(v_{n3}v_{k3})/(\lambda_3 - \hat{\lambda})| < \sqrt{\varepsilon}$ and by (2.8.15), $|u_{nn}| = O(\|T\|)$. In such a case, (2.8.13) implies that $\tau = O(\|T\|^2)$ and if $\|T\| > 1$ the right hand side is scaled up rather than being scaled down! As a consequence, xSTEIN overflows on the scaled matrix $\sqrt{\Omega}\, T$ where Ω is the overflow threshold of the computer ($\Omega = 2^{1023} \approx 10^{308}$ in IEEE double precision arithmetic). $\square$

Note that the above matrix does not deserve overflow. A similar overflow occurrence (in IEEE double precision arithmetic) on an 8×8 matrix, with a largest element of magnitude $2^{484} \approx 10^{145}$, was reported to us by Jeremy DuCroz [48].

The problems reported above can be cured by reverting back to the choice of scale factor in EISPACK's TINVIT.

IV. **Orthogonality.** TINVIT and xSTEIN use the modified Gram-Schmidt (MGS) procedure to orthogonalize iterates corresponding to eigenvalues whose separation is less than $10^{-3}\|T\|$. In order for the orthogonalized vectors to actually be numerically orthogonal, the vectors must not be parallel prior to the orthogonalization. In the

[2] In the summer of 1996, a core dump on the main computer aboard the Ariane 5 rocket was interpreted as flight data, causing a violent trajectory correction that led to the disintegration of the rocket

following example the vectors to be orthogonalized are almost parallel. The next two examples are reproduced in Case Study A.

Example 2.8.6 [Parallel Iterates.] Consider the matrix of (2.8.16). T has the eigenvalues

$$\lambda_1 = \varepsilon/2 + O(\varepsilon^2), \qquad \lambda_2 = \varepsilon + O(\varepsilon^2), \qquad \lambda_3 = 1 + \varepsilon + O(\varepsilon^2).$$

The eigenvalues of T as computed by MATLAB's **eig**[3] function are

$$\hat{\lambda}_1 = \varepsilon, \qquad \hat{\lambda}_2 = \varepsilon, \qquad \hat{\lambda}_3 = 1 + \varepsilon.$$

We perturb $\hat{\lambda}_2$ to $\varepsilon(1+\varepsilon)$ and input these approximations to TINVIT to demonstrate this failure (equal approximations input to TINVIT are perturbed by approximately $\varepsilon\|T\|$, see Example 2.8.2).

The first eigenvector is computed by TINVIT as

$$y_1 = (T - \hat{\lambda}_1 I)^{-1} b_1.$$

In exact arithmetic (taking $b_1 = \sum_i \xi_i v_i$),

$$\begin{aligned} y_1 &= \frac{\xi_2}{\lambda_2 - \hat{\lambda}_1}\left(v_2 + \frac{\xi_1}{\xi_2}\frac{\lambda_2 - \hat{\lambda}_1}{\lambda_1 - \hat{\lambda}_1} v_1 + \frac{\xi_3}{\xi_2}\frac{\lambda_2 - \hat{\lambda}_1}{\lambda_3 - \hat{\lambda}_1} v_3\right) \\ &= \frac{1}{O(\varepsilon^2)}\left(v_2 + O(\varepsilon) v_1 + O(\varepsilon^2) v_3\right) \end{aligned}$$

provided (ξ_1/ξ_2) and (ξ_3/ξ_2) are $O(1)$. Due to the inevitable roundoff errors in finite precision, the best we can hope to compute is

$$\hat{y}_1 = \frac{1}{O(\varepsilon^2)}\left(v_2 + O(\varepsilon) v_1 + O(\varepsilon) v_3\right).$$

This vector is normalized to remove the $1/O(\varepsilon^2)$ factor and give $\hat{v}_1$. The second eigenvector is computed as

$$y_2 = (T - \hat{\lambda}_2 I)^{-1} b_2.$$

Since $\hat{\lambda}_2 \approx \hat{\lambda}_1$, the computed value of y_2 is almost parallel to $\hat{v}_1$ (assuming that b_2 has a non-negligible component in the direction of v_2), i.e.,

$$\hat{y}_2 = \frac{1}{O(\varepsilon^2)}\left(v_2 + O(\varepsilon) v_1 + O(\varepsilon) v_3\right).$$

[3] MATLAB's **eig** function computes eigenvalues by the QR algorithm

Step	1		2		3	
	Before MGS	After MGS	Before MGS	After MGS	Before MGS	After MGS
Iterate (y)	$1.05 \cdot 10^{-8}$ -0.707 -0.707	$1.04 \cdot 10^{-8}$ -0.697 0.716	$1.05 \cdot 10^{-8}$ -0.707 -0.707	$1.05 \cdot 10^{-8}$ -0.7069 0.7073	$1.05 \cdot 10^{-8}$ -0.707 -0.707	$1.05 \cdot 10^{-8}$ -0.707108 0.707105
$\|y^T\hat{v}_1\|$	1.000	0.0014	1.000	$3.0 \cdot 10^{-4}$	1.000	$1.9 \cdot 10^{-6}$
$\|y^T\hat{v}_3\|$	$3.9 \cdot 10^{-24}$	$4.1 \cdot 10^{-11}$	$5.8 \cdot 10^{-25}$	$2.9 \cdot 10^{-12}$	$2.2 \cdot 10^{-24}$	$1.1 \cdot 10^{-13}$

Table 2.1: Summary of xSTEIN's iterations to compute the second eigenvector of T.

Since $\hat{\lambda}_1$ and $\hat{\lambda}_2$ are nearly coincident, $\hat{y}_2$ is orthogonalized against $\hat{v}_1$ by the MGS process in an attempt to reveal the second eigenvector :

$$\begin{aligned} z &= \hat{y}_2 - (\hat{y}_2^T\hat{v}_1)\hat{v}_1 \qquad (2.8.17)\\ &= \frac{1}{O(\varepsilon^2)}(O(\varepsilon)v_1 + O(\varepsilon)v_2 + O(\varepsilon)v_3)\\ \hat{v}_2 &= z/\|z\|. \end{aligned}$$

Clearly z is not orthogonal to $\hat{v}_1$. But TINVIT accepts z as having "converged" since $\|z\|$ is big enough to satisfy the convergence criterion (2.8.14) even after the severe cancellation in (2.8.17). The above observations are confirmed by a numerical run in double precision arithmetic wherein the first two eigenvectors of the matrix in (2.8.16) as output by TINVIT had a dot product of 0.0452! □

Unlike TINVIT, xSTEIN computes each eigenvector from a different random starting vector. The hope is to get greater linear independence of the iterates before orthogonalization by the MGS method [87]. However, as we now show, the TINVIT error as reported above persists.

Example 2.8.7 [Linear Dependence Persists.] Consider again the matrix T given in (2.8.16). The eigenvalues input to xSTEIN are computed by the **eig** function in MATLAB as $\hat{\lambda}_1 = \hat{\lambda}_2 = \varepsilon$ and $\hat{\lambda}_3 = 1 + \varepsilon$. As in TINVIT the first eigenvector computed as $\hat{v}_1$ is almost parallel to v_2. The iterations to compute the second eigenvector are summarized in Table 2.1.

This behavior is very similar to that of TINVIT (see Example 2.8.6). xSTEIN does two more iterations than TINVIT and alleviates the problem slightly, but a dot product of 1.9×10^{-6} between computed eigenvectors is far from satisfactory (this run was in double precision). □

xSTEIN avoids the overflow problems of TINVIT shown in example 2.8.3. It checks to see if overflow would occur, and if so, perturbs tiny entries on the diagonal of U [85]. This check is in the inner loop when solving $Uy = x$ where $x = \tau L^{-1}P^{-1}b$. Coupled with the extra iterations done after convergence, this results in xSTEIN being slower than TINVIT. On an assorted collection of test matrices of sizes 50 to 1000, we observed xSTEIN to be 2-3 times slower than TINVIT. However xSTEIN was more accurate than TINVIT in general.

2.9 Our Approach

As we observed in the previous section, the computer implementation of a seemingly straightforward task can lead to surprising failures. We want to avoid such failures and indicate our approach to the various aspects of inverse iteration discussed above.

I. **Choice of shift.** Of the various issues discussed in Section 2.7, the choice of starting vector and convergence criterion have been extensively studied [136, 118, 119, 87]. Surprisingly, the choice of shift for inverse iteration seems to have drawn little attention. We feel that the shift is probably the most important variable in inverse iteration. Examples 2.8.6 and 2.8.7 highlight the importance of shifts that are *as accurate as possible. We present our solution in Chapter 4.*

II. **Direction of starting vector and scaling of right hand side.** *This problem is solved.* It is now possible to deterministically find a starting vector that is guaranteed to have an above average component in the direction of the desired eigenvector v. Knowing the position of a large component of v also enables us to avoid the possibility of overflow in the eigenvector computation. See Chapter 3 for more details.

III. **Convergence criterion and Orthogonality.** It is easy to find a criterion that guarantees small residual norms (goal (1.1.1)). However, as we saw in earlier sections, the goal of orthogonality (1.1.2) can lead to a myriad of problems. Most of the "difficult" errors in the EISPACK and LAPACK implementations arise due to the explicit orthogonalization of iterates when eigenvalues are close. In [20], Chandrasekaran explains the theory behind some of these failures, and proposes an alternate version of inverse iteration that is provably correct. However, this version is more involved and potentially requires much more orthogonalization than in existing implementations. We want to take a different approach to orthogonality, and this is discussed in Chapters 4, 5 and 6.

Chapter 3

Computing the eigenvector of an isolated eigenvalue

Given an eigenvalue λ of a tridiagonal matrix $\tilde{J}$, a natural way to solve for $v \neq 0$ in

$$(\tilde{J} - \lambda I)v = 0 \tag{3.0.1}$$

is to set $v(1) = 1$ and to use the first equation of (3.0.1) to determine $v(2)$, and the second to determine $v(3)$, using $v(1)$ and $v(2)$. Proceeding like this, the rth equation may be used to determine $v(r+1)$ and thus v may be obtained without actually making use of the nth equation which will be satisfied automatically since the system (3.0.1) is singular.

It would be equally valid to begin with $\tilde{v}(n) = 1$ and to take the equations in reverse order to compute $\tilde{v}(n-1), \ldots, \tilde{v}(2), \tilde{v}(1)$ in turn without using the first equation in (3.0.1). When normalized in the same way v and $\tilde{v}$ will yield the same eigenvector in exact arithmetic.

The method described above is 'obvious' and was mentioned by W. Givens in 1957, see [61]. It often gives good results when realized on a computer but, at other times, delivers vectors pointing in completely wrong directions for the following reason. It is rare that an eigenvalue of a tridiagonal (or any other) matrix is representable in limited precision. Consequently the systems such as (3.0.1) that are to be solved are not singular and, in (3.0.1), the unused equation will not be satisfied automatically even if the solutions of the other equations were obtained exactly. The two methods given above result in solving

$$(\tilde{J} - \hat{\lambda} I)x^{(n)} = \delta_n e_n \tag{3.0.2}$$

and

$$(\tilde{J} - \hat{\lambda} I)x^{(1)} = \delta_1 e_1 \tag{3.0.3}$$

respectively, where $\hat{\lambda}$ is an approximation to λ and δ_1, δ_n are the residuals of the equations that are not satisfied. Note that (3.0.2) and (3.0.3) show that the natural method may be thought of as doing one step of inverse iteration as given in (2.7.4) with the starting vector equal to e_1 or e_n. We now present an example illustrating how this natural method can fail.

Example 3.0.1 [Choice of e_1 is wrong.] Consider

$$T = \begin{bmatrix} 3\varepsilon/4 & \varepsilon/4 & 0 \\ \varepsilon/4 & 7\varepsilon/4 & \sqrt{\varepsilon} \\ 0 & \sqrt{\varepsilon} & 1 \end{bmatrix} \tag{3.0.4}$$

where ε is of the order of the machine precision. An eigenvalue of T is

$$\lambda = 1 + \varepsilon + O(\varepsilon^2),$$

and its associated eigenvector is

$$v = \begin{bmatrix} \varepsilon^{3/2}/4 + O(\varepsilon^{5/2}) \\ \sqrt{\varepsilon} + O(\varepsilon^{3/2}) \\ 1 - \varepsilon/2 + O(\varepsilon^2) \end{bmatrix}. \tag{3.0.5}$$

The vector obtained *in exact arithmetic* by ignoring the first equation, with the approximation $\hat{\lambda} = 1$, is

$$x^{(1)} = \begin{bmatrix} -4/\sqrt{\varepsilon} \\ 0 \\ 1 \end{bmatrix},$$

but

$$\frac{\|(T - \hat{\lambda} I)x^{(1)}\|_2}{\|x^{(1)}\|_2} = \frac{4 - 3\varepsilon}{\sqrt{16 + \varepsilon}} \rightarrow 1 \quad \text{as} \quad \varepsilon \rightarrow 0.$$

□

In [136, pp.319–321] and[134], Wilkinson presents a similar example where omitting the last equation results in a poor approximation to an eigenvector. His example matrix is

$$W_{21}^{-} = \begin{bmatrix} 10 & 1 & & & & & 0 \\ 1 & 9 & 1 & & & & \\ & 1 & 8 & . & & & \\ & & . & . & . & & \\ & & & . & -8 & 1 & \\ & & & & 1 & -9 & 1 \\ 0 & & & & & 1 & -10 \end{bmatrix}. \tag{3.0.6}$$

Letting λ be the largest eigenvalue of W_{21}^{-}, Wilkinson shows that there is no 9-figure approximation $\hat{\lambda}$ to λ for which the *exact* normalized solution of the first 20 equations of $(W_{21}^{-} - \hat{\lambda}I)x = 0$ is anywhere close to the desired eigenvector. We now repeat his analysis.

Suppose

$$(\tilde{J} - \hat{\lambda}I)x^{(r)} = e_r,$$

where $\hat{\lambda}$ approximates λ_j. Writing e_r in terms of the eigenvectors, $e_r = \sum_{i=1}^{n} \xi_i v_i$, we get

$$\begin{aligned} x^{(r)} &= (\tilde{J} - \hat{\lambda}I)^{-1} e_r \\ &= \frac{\xi_j}{\lambda_j - \hat{\lambda}} \left(v_j + \sum_{i \neq j} \frac{\xi_i}{\xi_j} \frac{\lambda_j - \hat{\lambda}}{\lambda_i - \hat{\lambda}} v_i \right) \end{aligned} \tag{3.0.7}$$

Fundamental limitations of finite precision arithmetic dictate that $|\hat{\lambda} - \lambda_j|$ cannot be better than $O(n\varepsilon\|\tilde{J}\|)$, in general, where ε is the machine precision. Even when $|\hat{\lambda} - \lambda_j| \approx n\varepsilon\|\tilde{J}\|$, (3.0.7) shows that $x^{(r)}$ may not be close to v_j's direction if $\xi_j = e_r^* v_j = v_j(r)$ is tiny. In Example 3.0.1, $r = 1$ and $\xi_j \approx O(\varepsilon^{3/2})$ while in Wilkinson's example, $r = n = 21$ and $\xi_j = 10^{-17}$. As a result, in both these cases the natural method does not approximate the eigenvector well.

Given an accurate $\hat{\lambda}$, (3.0.7) implies that $x^{(r)}$ will be a good approximation to v_j provided the rth component of v_j is not small. Wilkinson notes this in [136, p.318] and concludes,

> 'Hence if the largest component of u_k [v_j in our notation] is the rth, then it is the rth equation which should be omitted when computing u_k. This result is instructive but not particularly useful, since we will not know *a priori* the position of the largest component of u_k. In fact, u_k is precisely what we are attempting to compute!'

In the absence of a reliable and cheap procedure to find r, Wilkinson compromised by choosing the starting vector for inverse iteration as PLe, where $\tilde{J} - \hat{\lambda}I = PLU$ and e

is the vector of all 1's (this led to the choice made in EISPACK [128]). A random starting vector is used in the LAPACK implementation of inverse iteration [87].

In this chapter, we show the following

1. Sections 3.1 and 3.2 introduce twisted factorizations that provide an answer to Wilkinson's problem of choosing which equation to omit. This is due to pioneering work by Fernando, and enables us to discard LAPACK's random choice of starting vector to compute an eigenvector of an *isolated* eigenvalue [57, 117].

2. In Section 3.3 we show how to adapt the results of Sections 3.1 and 3.2 when triangular factorizations don't exist. Some of the material presented in Sections 3.1-3.3 has appeared in [117].

3. Section 3.4 shows how to eliminate the divisions in the method outlined in Section 3.2.

4. In Section 3.5, we introduce twisted Q factorizations and give an alternate method to compute an eigenvector. We also show how the *perfect shift* strategy suggested by Parlett [110, p.173] can be made to succeed.

5. In Section 3.6, we show that twisted factorizations may also be used to detect singularity of Hessenberg and dense matrices.

In preparation for the upcoming theory, we state a few well known results without proof. The informed reader should be able to furnish their proofs without much difficulty. We expect that the reader knows the LDU decomposition and theorem concerning existence and uniqueness of triangular factorization and the expressions for the pivots, as the diagonal entries of D are often called.

Lemma 3.0.1 *Let $\tilde{J}$ be an unreduced tridiagonal matrix and v be an eigenvector. Then the first and last components of v are non-zero.*

Lemma 3.0.2 *Let $\tilde{J}$ be an unreduced normal tridiagonal matrix. Then every eigenvalue of $\tilde{J}$ is simple.*

Lemma 3.0.3 *Let $J = \tilde{J} - \lambda I$ be an unreduced singular tridiagonal matrix where λ is an eigenvalue of $\tilde{J}$ and v is the corresponding eigenvector. Then the following are equivalent.*

i. J admits the LDU and UDL triangular factorizations.

ii. All strictly leading and trailing submatrices of J are nonsingular.

Furthermore when J is normal, the following is equivalent to the above

iii. No component of v is zero.

Proof. The latter is not so obvious and follows from the fact that when J is normal

$$|v(i)|^2\chi'(\lambda) \;=\; \chi^{1:i-1}(\lambda)\cdot\chi^{i+1:n}(\lambda) \tag{3.0.8}$$

where $\chi^{l:m}(\mu) = \det(\mu I - \tilde{J}^{l:m})$. See [110, Section 7-9] to derive the above. $\square$

3.1 Twisted Factorizations

Despite Wilkinson's pessimism, we ask the question: can we find reliable indicators of the sizes of various components of a normalized eigenvector without knowing the eigenvector itself? We turn to triangular factorization in search of an answer and examine the LDU decomposition of $\tilde{J} - \hat{\lambda}I$. In this representation, both L and U have 1's on the diagonal. We will denote the ith diagonal element of D by $D(i)$ and $L(i+1,i), U(i,i+1)$ by $L(i)$ and $U(i)$ respectively.

If $\hat{\lambda}$ is an eigenvalue of $\tilde{J}$, the following theorem implies that $D(n)$ must be zero in exact arithmetic.

Theorem 3.1.1 *Let B be a matrix of order n such that the triangular factorization*

$$B = LDU$$

exists, i.e., its principal submatrices $B^{1:i}$, $i = 1, 2, \ldots, n-1$ are nonsingular. Then if B is singular,

$$D(n) = 0.$$

Proof. The expression

$$D(n) = \frac{\det(B)}{\det(B^{1:n-1})}$$

implies that if B is singular, $D(n) = 0$. $\square$

We now examine the values $D(n)$ can take when $\hat{\lambda}$ is not an eigenvalue. Suppose $\hat{\lambda}$ approximates λ_j. Since $\tilde{J} - \hat{\lambda}I = LDU$,

$$\begin{aligned} U^{-1}D^{-1}L^{-1} &= (\tilde{J} - \hat{\lambda}I)^{-1} \\ \Rightarrow\; e_n^*U^{-1}D^{-1}L^{-1}e_n &= e_n^*(\tilde{J} - \hat{\lambda}I)^{-1}e_n \end{aligned}$$

Since L and U^* are unit lower triangular, $Le_n = e_n$ and $U^* e_n = e_n$. Letting $\tilde{J} = V\Lambda V^*$, we get

$$\begin{aligned} \frac{1}{D(n)} &= e_n^* V(\Lambda - \hat{\lambda} I)^{-1} V^* e_n \\ \Rightarrow \quad \frac{1}{D(n)} &= \frac{|v_j(n)|^2}{\lambda_j - \hat{\lambda}} + \sum_{i \neq j} \frac{|v_i(n)|^2}{\lambda_i - \hat{\lambda}}, \end{aligned} \tag{3.1.9}$$

where $v_i(n)$ denotes the nth component of v_i. (3.1.9) implies that when $\hat{\lambda}$ is close to λ_j and λ_j is isolated, a large value of $v_j(n)$ results in $D(n)$ being small. But, when $v_j(n)$ is tiny, $D(n)$ can be as large as $O(\|\tilde{J}\|)$. Thus the value of $D(n)$ reflects the value of the last component of the desired eigenvector in addition to the accuracy of $\hat{\lambda}$.

We can consider another triangular factorization of a matrix, i.e., the UDL decomposition that may be obtained by taking rows in decreasing order of their index. To differentiate this process from the standard LDU factorization, we make a notational innovation. We will use $+$ to indicate a process taking rows in increasing order and $-$ to indicate the process going in decreasing order, i.e., LDU will henceforth be written as $L_+D_+U_+$ while UDL will be written as $U_-D_-L_-$.

By repeating the analysis that led to (3.1.9), it can be shown that in the $U_-D_-L_-$ factorization, $D_-(1)$ is small when $v_j(1)$ is large, but not otherwise. Thus, the value of $D_-(1)$ mirrors the value of $v_j(1)$. Besides $D_+(n)$ and $D_-(1)$ can we find quantities that indicate the magnitudes of other components of the desired eigenvector?

Natural candidates in this quest are elements of various *twisted factorizations.* Instead of completing the $L_+D_+U_+$ factorization of a matrix, we can stop it at an intermediate row k. Now we can start the $U_-D_-L_-$ factorization from the bottom of the matrix, going up till we have a singleton in the kth row, which we will denote by γ_k. Such a twisted factorization, with $n = 5$ and $k = 3$ is shown in Figure 3.1.

We formally define a twisted triangular factorization of a tridiagonal matrix J at twist position k as

$$J = N_k D_k \tilde{N}_k, \tag{3.1.10}$$

$$
\begin{bmatrix} x & x & & & \\ x & x & x & & \\ & x & x & x & \\ & & x & x & x \\ & & & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & & & \\ & x & x & & \\ & x & x & x & \\ & & x & x & \\ & & & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & & & \\ & x & x & & \\ & & x & x & \\ & & x & x & \\ & & & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & & & \\ & x & x & & \\ & & x & & \\ & & x & x & \\ & & & x & x \end{bmatrix}
$$

Figure 3.1: Twisted Triangular Factorization at $k = 3$ (the next elements to be annihilated are circled)

where

$$
N_k = \begin{bmatrix}
1 & & & & & & & & \\
L_+(1) & 1 & & & & & & & \\
 & \ddots & \ddots & & & & & & \\
 & & L_+(k-2) & 1 & & & & & \\
 & & & L_+(k-1) & 1 & U_-(k) & & & \\
 & & & & & 1 & U_-(k+1) & & \\
 & & & & & & 1 & \ddots & \\
 & & & & & & & \ddots & U_-(n-1) \\
 & & & & & & & & 1
\end{bmatrix},
$$

$\tilde{N}_k$ has the same non-zero structure as N_k^* with $L_+(1), \ldots, L_+(k-1), U_-(k), \ldots, U_-(n-1)$ being replaced by $U_+(1), \ldots, U_+(k-1), L_-(k), \ldots, L_-(n-1)$ respectively,

$$D_k = \text{diag}(D_+(1), \ldots, D_+(k-1), \gamma_k, D_-(k+1), \ldots, D_-(n))$$

and

$$J \;=\; L_+D_+U_+ \;=\; U_-D_-L_-.$$

The perceptive reader may already have grasped the significance of γ_k, which we now explain.

Theorem 3.1.2 (Double Factorization) *Let J be a tridiagonal $n \times n$ complex matrix that permits triangular factorization in both increasing and decreasing order of rows:*

$$L_+D_+U_+ = J = U_-D_-L_-. \tag{3.1.11}$$

Consider the twisted factorization at k given by (3.1.10). Then for $1 \le k \le n$,

$$\gamma_k = D_+(k) + D_-(k) - J_{kk} \tag{3.1.12}$$

and if J is invertible,

$$\frac{1}{\gamma_k} = e_k^* J^{-1} e_k. \tag{3.1.13}$$

Proof. By construction of the twisted factorization,

$$\begin{aligned}
\gamma_k &= -J_{k-1,k}L_+(k-1) + J_{kk} - J_{k+1,k}U_-(k) \qquad (3.1.14)\\
&= (J_{kk} - J_{k-1,k}L_+(k-1)) - J_{kk} + (J_{kk} - J_{k+1,k}U_-(k))\\
&= D_+(k) - J_{kk} + D_-(k)
\end{aligned}$$

for $1 \le k \le n$. Here we take $J_{1,0} = J_{0,1} = J_{n+1,n} = J_{n,n+1} = 0$, and $L_+(0) = U_-(n) = 0$.

Since $N_k e_k = e_k$ and $e_k^* \tilde{N}_k = e_k^*$,

$$\begin{aligned}
e_k^* J^{-1} e_k &= e_k^* \tilde{N}_k^{-1} D_k^{-1} N_k^{-1} e_k\\
&= \frac{1}{\gamma_k}
\end{aligned}$$

□

Note that $\gamma_n = D_+(n)$ and $\gamma_1 = D_-(1)$. The above theorem shows how to get *all possible* n twisted factorizations at the cost of $2n$ divisions. The alert reader would have noted that the above theorem makes no assumption about the nearness to singularity of J.

We want to emphasize that in the theory developed here the existence of the triangular factorizations (3.1.11) is not restrictive. Neither is the occasional requirement that J be nonsingular. In fact, the singularity of J is desirable in our application of computing an eigenvector. As we will show in Section 3.3, in the absence of these requirements, all the theory developed in this section and the next is easily extended if ∞ is added to the number system.

The following corollary gives alternate expressions for γ_k.

Corollary 3.1.1 *With the notation of Theorem 3.1.2, for* $1 < k < n$*,*

$$\gamma_k = \begin{cases} D_+(k) - J_{kk} + D_-(k), \\ \\ -J_{k-1,k}L_+(k-1) + J_{kk} - J_{k+1,k}U_-(k), \\ \\ -J_{k,k-1}U_+(k-1) + J_{kk} - J_{k,k+1}L_-(k), \\ \\ D_+(k) - J_{k+1,k}U_-(k), \\ \\ -J_{k-1,k}L_+(k-1) + D_-(k). \end{cases}$$

For $k = 1$ *and* $k = n$ *omit terms with invalid indices.*

Proof. The first and second expressions are just (3.1.12) and (3.1.14). The others come from rewriting (3.1.14) as

$$\gamma_k = -J_{k,k-1}J_{k-1,k}/D_+(k-1) + J_{kk} - J_{k,k+1}J_{k+1,k}/D_-(k+1) \tag{3.1.15}$$

and using $J_{k,k+1} = U_-(k)D_-(k+1) = D_+(k)U_+(k)$ etc., and the formula for the backward pivots, $D_-(k) = J_{kk} - J_{k,k+1}J_{k+1,k}/D_-(k+1)$, etc. $\square$

As shown below, double factorization makes available the so called *Newton correction.*

Corollary 3.1.2 *Assume the hypothesis (3.1.11) of Theorem 3.1.2, and let* J *be nonsingular. Then*

$$\sum_{i=1}^{n} \gamma_i^{-1} \;=\; \mathrm{Trace}(J^{-1}) \;=\; -\frac{\chi'(0)}{\chi(0)}, \tag{3.1.16}$$

where $\chi(\mu) = \det(\mu I - J)$.

Proof. By (3.1.13) in Theorem 3.1.2,

$$\sum_i \gamma_i^{-1} \;=\; \sum_i e_i^* J^{-1} e_i \;=\; \mathrm{Trace}(J^{-1}) \;=\; -\frac{\chi'(0)}{\chi(0)}$$

since

$$\frac{\chi'(\mu)}{\chi(\mu)} \;=\; \frac{\sum_j \prod_{i\neq j}(\mu - \lambda_i)}{\prod_i (\mu - \lambda_i)} \;=\; \sum_i \frac{1}{\mu - \lambda_i}.$$

$\square$

Double factorization also allows a wide choice of expressions for $\det(J)$, and a recurrence for computing γ_i that involves only multiplications and divisions.

Theorem 3.1.3 *Assume the hypothesis of Theorem 3.1.2. Then for $k = 1, \ldots, n$ (omit invalid indices),*

$$\det(J) = D_+(1) \cdots D_+(k-1)\gamma_k D_-(k+1) \cdots D_-(n) \tag{3.1.17}$$

and

$$\gamma_k D_-(k+1) = D_+(k)\gamma_{k+1}. \tag{3.1.18}$$

Proof. From (3.1.10),

$$\det(J) = \det(N_k)\det(D_k)\det(\tilde{N}_k) = \det(D_k),$$

and we get (3.1.17). Applying (3.1.17) to two successive k's leads to (3.1.18). □

The following are immediate consequences of (3.1.17),

Corollary 3.1.3 *Assuming the hypothesis of Theorem 3.1.2,*

$$\begin{aligned} \gamma_k &= D_+(k) \prod_{i=k+1}^{n} (D_+(i)/D_-(i)), \\ &= D_-(k) \prod_{i=1}^{k-1} (D_-(i)/D_+(i)). \end{aligned}$$

Corollary 3.1.4 *Assuming the existence of the twisted factorization (3.1.10),*

$$\gamma_k = \frac{\det(J)}{\det(J^{1:k-1})\det(J^{k+1:n})}.$$

Note that the above corollary shows that $\gamma_k = 0$ if J is singular and admits triangular factorizations (3.1.11). See also Theorem 3.1.1 which proves that $\gamma_n = 0$ when J is singular.

The Double Factorization theorem is not new. In 1992, in [108], Meurant reviewed a significant portion of the literature on the inverses of band matrices and presented the main ideas in a nice unified framework. The inexpensive additive formulae for $(J^{-1})_{kk}$ (Theorem 3.1.2 and Corollary 3.1.1) are included in Theorem 3.1 of [108], while our Corollary 3.1.3 that gives the quotient/product form of $(J^{-1})_{kk}$ is given in Theorem 2.3 of [108]. We believe that such formulae have been known for quite some time in the differential equations community. However, these researchers were not interested in computing eigenvectors but in obtaining analytic expressions for elements of the inverse, when possible, and in the decay rate in terms of distance from the main diagonal. Our contribution is in recognizing the importance of twisted factorizations and successfully applying them to solve some elusive problems in numerical linear algebra. We will show some of these applications in this thesis, for other applications see [42, 115, 74]. In addition to the papers reviewed in [108], twisted factorizations have appeared in various contexts in the literature, see [77, 94, 5, 58, 130, 133, 44]. For a brief review, see Section 4.1 of [117]. Twisted factorizations have also been referred to as BABE factorizations (Begin, or Burn, at Both Ends) in [78, 57, 74].

3.2 The Eigenvector Connection

Given an eigenvalue approximation $\hat{\lambda}$ of $\tilde{J}$, we can compute the double factorization of $\tilde{J} - \hat{\lambda} I$ by Theorem 3.1.2. In this section, we see how double factorization can be used to find a "good" equation to omit when solving the system $(\tilde{J} - \hat{\lambda} I)x = 0$, thereby obtaining a good approximation to the desired eigenvector. Some of the results of this section have appeared in [117].

Since $\gamma_k^{-1} = e_k^*(\tilde{J} - \hat{\lambda} I)^{-1} e_k$, if we let $\tilde{J} = V \Lambda V^*$, we get an expression for γ_k that is similar to (3.1.9),

$$\begin{aligned} \frac{1}{\gamma_k} &= e_k^* V (\Lambda - \hat{\lambda} I)^{-1} V^* e_k, \\ \Rightarrow \quad \frac{1}{\gamma_k} &= \frac{|v_j(k)|^2}{\lambda_j - \hat{\lambda}} + \sum_{i \neq j} \frac{|v_i(k)|^2}{\lambda_i - \hat{\lambda}}. \end{aligned} \tag{3.2.19}$$

Thus when $\hat{\lambda}$ is close to an isolated λ_j and λ_j is isolated, the value of γ_k reflects the value of $v_j(k)$, i.e, an above average value of $v_j(k)$ leads to a tiny value of γ_k while a small $v_j(k)$ implies a large γ_k. We now make this claim more precise.

Lemma 3.2.1 *Let $\tilde{J} - \mu I$ be a normal, unreduced nonsingular tridiagonal matrix that satisfies the hypotheses of Theorem 3.1.2 for all μ in some open interval containing the eigenvalue λ_j. Let $\gamma_k = \gamma_k(\mu)$ be as in (3.1.13), for each k. As $\mu \longrightarrow \lambda_j$,*

$$\frac{\gamma_k^{-1}}{\sum_{m=1}^{n} \gamma_m^{-1}} \longrightarrow |v_j(k)|^2, \quad k = 1, 2, \ldots, n. \tag{3.2.20}$$

Proof. For $\mu \neq \lambda_j$,

$$\left(\tilde{J} - \mu I\right)^{-1} = V(\Lambda - \mu I)^{-1} V^* = \sum_i v_i v_i^* (\lambda_i - \mu)^{-1}.$$

From (3.1.13) in Theorem 3.1.2, $\gamma_k^{-1} = \left[\left(\tilde{J} - \mu I\right)^{-1}\right]_{kk}$. Since λ_j is simple by Lemma 3.0.2, as $\mu \longrightarrow \lambda_j$,

$$
\begin{aligned}
(\lambda_j - \mu)\left(\tilde{J} - \mu I\right)^{-1} &\longrightarrow v_j v_j^*, \\
(\lambda_j - \mu)\gamma_k^{-1} &\longrightarrow |v_j(k)|^2, \quad k = 1, 2, \ldots, n \\
(\lambda_j - \mu)\sum_{m=1}^{n} \gamma_m^{-1} &\longrightarrow \|v_j\|^2 = 1.
\end{aligned}
$$

□

The following theorem replaces the limits of Lemma 3.2.1 with error bounds. It implies that when μ is sufficiently close to λ_j then one of the γ_k's must be small. Note that in the following theorem, the matrix need not be tridiagonal and the eigenvalues may be complex.

Theorem 3.2.1 *Let $\tilde{J} - \mu I$ be a normal, invertible matrix, and let*

$$\gamma_k^{-1} = e_k^*(\tilde{J} - \mu I)^{-1} e_k, \quad \text{for } k = 1, 2, \ldots, n.$$

Then if $v_j(k) \neq 0$,

$$\gamma_k = \frac{\lambda_j - \mu}{|v_j(k)|^2}\left[1 + \left(|v_j(k)|^{-2} - 1\right)\mathcal{A}_1\right]^{-1}. \tag{3.2.21}$$

Here $\mathcal{A}_1$ is a weighted arithmetic mean of $\left\{\frac{\lambda_j - \mu}{\lambda_i - \mu},\ i \neq j\right\}$, $0 \le |\mathcal{A}_1| < |\lambda_j - \mu|/gap(\mu)$, where $gap(\mu) = \min_{i \neq j} |\lambda_i - \mu|$. Furthermore, if λ_j is isolated enough, i.e.,

$$\frac{|\lambda_j - \mu|}{gap(\mu)} \le \frac{1}{M} \cdot \frac{1}{n-1} \quad \text{where} \quad M > 1 \tag{3.2.22}$$

then for k such that $|v_j(k)| \ge 1/\sqrt{n}$,

$$|\gamma_k| \le \frac{|\lambda_j - \mu|}{|v_j(k)|^2} \cdot \frac{M}{M-1} \le n|\lambda_j - \mu| \cdot \frac{M}{M-1} \tag{3.2.23}$$

Proof.

$$
\begin{aligned}
\gamma_k^{-1} &= e_k^*(\tilde{J} - \mu I)^{-1} e_k, \\
&= \sum_{i=1}^{n} \frac{|v_i(k)|^2}{\lambda_i - \mu}.
\end{aligned}
$$

Extract the jth term to find

$$\gamma_k^{-1} = \left(\frac{|v_j(k)|^2}{\lambda_j - \mu}\right)\left[1 + \sum_{i \neq j} \left|\frac{v_i(k)}{v_j(k)}\right|^2 \left(\frac{\lambda_j - \mu}{\lambda_i - \mu}\right)\right]. \tag{3.2.24}$$

Since

$$\sum_{i \neq j} \left| \frac{v_i(k)}{v_j(k)} \right|^2 = \frac{1 - |v_j(k)|^2}{|v_j(k)|^2}$$

the $\sum_{i \neq j}$ term in (3.2.24) may be written as

$$\left(|v_j(k)|^{-2} - 1 \right) \mathcal{A}_1, \qquad |\mathcal{A}_1| \leq |\lambda_j - \mu| / \text{gap}(\mu),$$

where

$$\mathcal{A}_1 \;=\; \sum_{i \neq j} w_i \left(\frac{\lambda_j - \mu}{\lambda_i - \mu} \right), \quad 1 = \sum_{i \neq j} w_i, \;\; w_i \geq 0, \;\; \text{gap}(\mu) = \min_{i \neq j} |\lambda_i - \mu|,$$

to yield the equality in (3.2.21). If (3.2.22) holds, then

$$|\gamma_k| \;\leq\; \frac{|\lambda_j - \mu|}{|v_j(k)|^2} \left[1 - \left(|v_j(k)|^{-2} - 1 \right) \left(\frac{1}{M \cdot (n-1)} \right) \right]^{-1}.$$

For k such that $|v_j(k)| \geq 1/\sqrt{n}$,

$$|\gamma_k| \;\leq\; \frac{|\lambda_j - \mu|}{|v_j(k)|^2} \left[1 - \frac{1}{M} \right]^{-1}.$$

□

In cases of interest, $|\lambda_j - \mu| / \text{gap}(\mu) = O(\varepsilon)$ and hence $M \gg 1$. Thus the factor $M/(M-1)$ in (3.2.23) is ≈ 1.

The following theorem shows a way to compute the vector z that satisfies all the equations of $(\tilde{J} - \mu I)z = 0$ except the kth.

Theorem 3.2.2 *Let $J = \tilde{J} - \mu I$ be an unreduced tridiagonal matrix that permits the twisted factorization (3.1.10) for a fixed value of k. Then the system*

$$(\tilde{J} - \mu I) z^{(k)} = \gamma_k e_k \tag{3.2.25}$$

has a unique solution given by

$$\begin{aligned} z^{(k)}(k) &= 1, \\ z^{(k)}(j) &= -U^+_{j,j+1} z^{(k)}(j+1), \quad j = k-1, \ldots, 1, \\ z^{(k)}(i) &= -L^-_{i,i-1} z^{(k)}(i-1), \quad i = k+1, \ldots, n. \end{aligned}$$

Proof. Since $N_k e_k = e_k$ and $D_k e_k = \gamma_k e_k$,

$$(\tilde{J} - \mu I) z^{(k)} = \gamma_k e_k \;\;\Rightarrow\;\; \tilde{N}_k z^{(k)} = e_k$$

from which the result follows. $\square$

Note that when J is unreduced, the above theorem precludes $z^{(k)}$ from having a zero entry. This is consistent with Theorem 3.0.3. A benefit of computing $z^{(k)}$ is the availability of the Rayleigh Quotient.

Corollary 3.2.1 *Let $\tilde{J} - \mu I$ satisfy the hypothesis of Theorem 3.1.2, and $z = z^{(k)}$ be as in (3.2.25) for $1 \leq k \leq n$. Then the Rayleigh Quotient of z is given by*

$$\frac{z^* \tilde{J} z}{z^* z} = \mu + \frac{\gamma_k}{\|z\|^2} \tag{3.2.26}$$

Proof. By (3.2.25),

$$z^*(\tilde{J} - \mu I) z = \gamma_k z^* e_k = \gamma_k$$

since $z(k) = 1$. The result (3.2.26) follows. $\square$

Since $\|z^{(k)}\| \geq 1$, Theorem 3.2.1 implies that the residual norm

$$\frac{\|(\tilde{J} - \mu I) z^{(k)}\|}{\|z^{(k)}\|} = \frac{|\gamma_k|}{\|z^{(k)}\|} \tag{3.2.27}$$

is small when μ is a good approximation to an isolated eigenvalue λ, and k is chosen appropriately. The following theorem gives a better bound on $\min_k |\gamma_k| / \|z^{(k)}\|$.

Theorem 3.2.3 *Let $\tilde{J} - \mu I$ be a normal, invertible matrix, and let*

$$(\tilde{J} - \mu I) z^{(k)} = e_k \gamma_k, \quad \text{for } k = 1, 2, \ldots, n.$$

Then if $v_j(k) \neq 0$,

$$\begin{aligned} \frac{|\gamma_k|}{\|z^{(k)}\|} &= \frac{|\mu - \lambda_j|}{|v_j(k)|} \left[1 + \left(|v_j(k)|^{-2} - 1\right) \mathcal{A}_2\right]^{-1/2}, \\ &\leq \frac{|\mu - \lambda_j|}{|v_j(k)|} \leq \sqrt{n}\, |\mu - \lambda_j| \text{ for at least one } k. \end{aligned} \tag{3.2.28}$$

Here $\mathcal{A}_2$ is a weighted arithmetic mean of $\left\{ \left| \frac{\mu - \lambda_j}{\mu - \lambda_i} \right|^2, \; i \neq j \right\}$, $0 < \mathcal{A}_2 < [|\lambda_j - \mu| / gap(\mu)]^2$.

Proof.

$$\begin{aligned} z^{(k)} &= (\tilde{J} - \mu I)^{-1} e_k \gamma_k, \\ \|z^{(k)}\|^2 &= |\gamma_k|^2 e_k^* V (\bar{\Lambda} - \bar{\mu} I)^{-1} (\Lambda - \mu I)^{-1} V^* e_k \\ &= |\gamma_k|^2 \sum_{i=1}^{n} \frac{|v_i(k)|^2}{|\mu - \lambda_i|^2}. \end{aligned}$$

Extract the jth term to find

$$\left(\frac{\|z^{(k)}\|}{|\gamma_k|}\right)^2 = \left(\frac{|v_j(k)|}{|\mu-\lambda_j|}\right)^2\left[1+\sum_{i\neq j}\left|\frac{v_i(k)}{v_j(k)}\right|^2\left|\frac{\mu-\lambda_j}{\mu-\lambda_i}\right|^2\right]$$

$$\Rightarrow \frac{|\gamma_k|}{\|z^{(k)}\|} = \frac{|\mu-\lambda_j|}{|v_j(k)|}\left[1+\left(|v_j(k)|^{-2}-1\right)\mathcal{A}_2\right]^{-1/2},$$

where

$$\mathcal{A}_2 = \sum_{i\neq j} w_i\left|\frac{\mu-\lambda_j}{\mu-\lambda_i}\right|^2, \quad 1=\sum_{i\neq j} w_i, \quad w_i\geq 0, \quad 0<\mathcal{A}_2<[|\lambda_j-\mu|/\mathrm{gap}(\mu)]^2.$$

Since $(|v_j(k)|^{-2}-1)\mathcal{A}_2\geq 0$, (3.2.28) follows easily from the above. □

The above theorems suggest a way to compute a vector $z^{(k)}$ such that the residual norm (3.2.27) is small. In 1995, Fernando, in an equivalent formulation, proposed choosing the index for which $|\gamma_k|$ is minimum, say r, and then solving $(\tilde{J}-\mu I)z^{(r)}=\gamma_r e_r$ to compute an approximate eigenvector $z^{(r)}$. See [57] for his subsequent work. Earlier, in 1985, Godunov and his collaborates proposed a similar but more obscure method for obtaining a provably accurate approximation to an eigenvector by 'sewing' together two "Sturm Sequences" that start at either end of the matrix. See [64] and [63] for their work, and Section 4.2 of [117] for interpretation in our notation. Fernando's approach leads to the following algorithm

Algorithm 3.2.1 [Computing an eigenvector of an isolated eigenvalue.]

1. Compute $\tilde{J}-\mu I = L_+D_+U_+ = U_-D_-L_-$.
2. Compute γ_k for $k=1,2,\ldots,n$ using the expressions in Corollary 3.1.1 or (3.1.18) in Theorem 3.1.3 and choose the index r where $|\gamma_k|$ is minimum.
3. Form the vector $z^{(r)}$ as in Theorem 3.2.2.

□

Note that $z^{(r)}$ is formed by multiplications only, thus promising greater accuracy and a more favorable error analysis than standard methods that involve additions. Using (3.2.23) and the fact that $\|z^{(r)}\|\geq 1$, the above algorithm bounds the residual norm by

$$\frac{\|(\tilde{J}-\mu I)z^{(r)}\|}{\|z^{(r)}\|} \leq n|\lambda_j-\mu|\cdot\frac{M}{M-1}. \tag{3.2.29}$$

This solves Wilkinson's problem of choosing which equation to omit, modulo the mild assumption (3.2.22) about the separation of λ_j. As we shall emphasize in later chapters, we will use Algorithm 3.2.1 *only when* λ_j *is sufficiently isolated.*

We have noted, and so has Jessie Barlow [8], that a simple recurrence will yield all values of $\|z^{(k)}\|$ for $O(n)$ operations. Consequently it would be feasible to minimize $|\gamma_k|/\|z^{(k)}\|$ instead of $|\gamma_k|$ to obtain a possibly smaller residual norm. At present we feel that the extra expense is not warranted. The bound given by (3.2.28) is certainly better than the above bound (3.2.29). However, the latter bound is much too pessimistic since $\|z^{(r)}\|$ can be as large as $\sqrt{n}$ when all eigenvector entries are equal in magnitude.

3.3 Zero Pivots

We now show that the procedure to compute an eigenvector suggested by the previous section needs to be modified only slightly when $J = \tilde{J} - \mu I$ does not admit the triangular factorizations (3.1.11) in Theorem 3.1.2. We will continue to assume exact arithmetic — the effect of roundoff errors in a computer implementation is examined in the next chapter.

Triangular factorization is said to fail, or not exist, if a zero 'pivot', $D_+(j)$ or $D_-(j)$ is encountered prematurely. The last pivot is allowed to vanish because it does not occur as a denominator in the computation.

One of the attractions of an unreduced tridiagonal matrix is that the damage done by a zero pivot is localized. Indeed, if $\pm\infty$ is added to the number system then triangular factorization cannot break down and the algorithm always maps J into unique triplets L_+, D_+, U_+ and U_-, D_-, L_-. There is no need to spoil the inner loop with tests. It is no longer true that $L_+D_+U_+ = J = U_-D_-L_-$ but equality does hold for all entries except for those at or adjacent to any infinite pivot. IEEE arithmetic [2] allows computer implementations to handle $\pm\infty$ and take advantage of this feature of tridiagonals.

We now show that proceeding thus, it is meaningful to pick $|\gamma_r| = \min_k |\gamma_k|$, omit the rth equation and solve for $z^{(r)}$ even when triangular factorization breaks down. We first handle the case when J is singular.

When J is singular, it may appear that any equation can be omitted to solve $Jz = 0$ by the method suggested in Theorem 3.2.2. However, zero entries in z do not allow such a simple solution.

Theorem 3.3.1 *Let J be $n \times n$, tridiagonal, unreduced, and singular. For each k, $1 < k < n$, $J^{1:k-1}$ is singular if, and only if, $J^{k+1:n}$ is singular. They are singular if, and only if, $z(k) = 0$ whenever $Jz = 0$.*

Proof. Write

$$z = \begin{bmatrix} z_+ \\ z(k) \\ z_- \end{bmatrix}$$

and partition $Jz = 0$ conformably. Thus

$$J^{1:k-1} z_+ + J_{k-1,k} z(k) e_{k-1} = 0, \tag{3.3.30}$$

$$e_1 J_{k+1,k} z(k) + J^{k+1:n} z_- = 0, \tag{3.3.31}$$

and $z_+(1) \neq 0$, $z_-(n) \neq 0$ by Lemma 3.0.1.

If $z(k) = 0$ then (3.3.30) shows that $z_+ (\neq 0)$ is in $J^{1:k-1}$'s null space and (3.3.31) shows that $z_- (\neq 0)$ is in $J^{k+1:n}$'s null space. So both matrices are singular.

Now consider the converse, $z(k) \neq 0$. Since J is unreduced, $\text{rank}(J) = n - 1$ and its null space is one dimensional. So the system

$$Jz = 0, \quad z(k) = 1,$$

has a unique solution. Thus both (3.3.30) and (3.3.31) are inhomogeneous equations with unique solutions. Thus $J^{1:k-1}$ and $J^{k+1:n}$ are invertible. □

Clearly when $z(k) = 0$, Theorem 3.2.2 should not be used to compute z. When $z(k) = 0$, Theorem 3.3.1 implies that $D_+(k-1) = D_-(k+1) = 0$, and the formulae for γ_k in Corollary 3.1.1 give $\gamma_k = \infty + \infty$ or $\infty - \infty$. When $z(k) \neq 0$, Corollary 3.1.4 implies that $\gamma_k = 0$. By Lemma 3.0.1, $z(1) \neq 0$ and $z(n) \neq 0$, and consequently $\gamma_1 = \gamma_n = 0$. Thus the index r where $|\gamma_k|$ is minimum is such that $z(r) \neq 0$. To account for possible breakdown in triangular factorization, the method of Theorem 3.2.2 may be modified slightly.

Algorithm 3.3.1 [Vectors with zeros.]

$$\begin{aligned}
z(r) &= 1, \\
z(j) &= \left\{ \begin{array}{ll} -U_+(j) z(j+1), & z(j+1) \neq 0, \\ -(J_{j+1,j+2}\, z(j+2)/J_{j+1,j}), & \text{otherwise} \end{array} \right\}, \ j = r-1, \ldots, 1 \\
z(i) &= \left\{ \begin{array}{ll} -L_-(i-1) z(i-1), & z(i-1) \neq 0, \\ -(J_{i-1,i-2}\, z(i-2)/J_{i-1,i}), & \text{otherwise} \end{array} \right\}, \ i = r+1, \ldots, n.
\end{aligned}$$

□

Thus the case of singular J is handled correctly. Note that the multiplicative recurrence given by (3.1.18) breaks down when computing γ_k as

$$\gamma_k = \gamma_{k+1} \frac{D_+(k)}{D_-(k+1)}$$

if $D_+(k) = 0$, $D_-(k+2) = 0$ and $\gamma_{k+1} = \infty$.

Now we consider the case when J is nonsingular and triangular factorization breaks down. When $D_+(k-1)$ or $D_-(k+1)$ equals zero, the expressions in Corollary 3.1.1 imply that $\gamma_k = \infty$. Note that both $D_+(k-1)$ and $D_-(k+1)$ cannot be zero, otherwise J would be singular. Unlike the singular case, it is possible that all $|\gamma_k|$ are ∞. We now show that this occurs only in very special circumstances. We recall that $\gamma_k^{-1} = (J^{-1})_{kk}$.

Lemma 3.3.1 *Let J be a complex tridiagonal matrix of order n with* $\operatorname{diag}(J) = 0$. *Then*

$$\begin{aligned} &\text{if } n \text{ is odd}, \quad \chi_n(\mu) = -\chi_n(-\mu), \quad \text{for all } \mu \in \mathcal{C}, \quad \text{and} \\ &\text{if } n \text{ is even}, \quad \chi_n(\mu) = \chi_n(-\mu), \quad \text{for all } \mu \in \mathcal{C}. \end{aligned}$$

where $\chi_i(\mu) = \det(\mu I - J^{1:i})$.

Proof. If $\operatorname{diag}(J) = 0$,

$$\chi_i(\mu) \;=\; \mu\chi_{i-1}(\mu) - J_{i,i-1}J_{i-1,i}\chi_{i-2}(\mu).$$

We prove the result by induction on i. $\chi_1(\mu) = \mu$ is an odd function, while $\chi_2(\mu) = \mu^2 - J_{i,i-1}J_{i-1,i}$ is even and the result holds for the base cases. In the inductive step, if i is odd

$$\begin{aligned} \chi_i(-\mu) &= (-\mu)\chi_{i-1}(-\mu) - J_{i,i-1}J_{i-1,i}\chi_{i-2}(-\mu) \\ &= -\mu\chi_{i-1}(\mu) + J_{i,i-1}J_{i-1,i}\chi_{i-2}(\mu) \;=\; -\chi_i(\mu). \end{aligned}$$

Similarly, if i is even

$$\chi_i(-\mu) \;=\; (-\mu)\chi_{i-1}(-\mu) - J_{i,i-1}J_{i-1,i}\chi_{i-2}(-\mu) \;=\; \chi_i(\mu).$$

□

Nonsingular J that have zero diagonals have a special pattern of zeros and nonzeros, as illustrated by the following example.

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & -1 & 0 & -1 & 0 \end{bmatrix}. \tag{3.3.32}$$

Theorem 3.3.2 *Let J be a nonsingular tridiagonal matrix. Then*

$$\mathrm{diag}(J) = 0 \;\Leftrightarrow\; \mathrm{diag}(J^{-1}) = 0.$$

Proof. In this proof, we will use the famous Cauchy-Binet formula

$$J \cdot \mathrm{adj}(J) \;=\; \det(J) \cdot I \tag{3.3.33}$$

where $\mathrm{adj}(J)$ is the *adjugate* of J and is the transpose of the matrix of cofactors [129, p.402], to get expressions for elements of J^{-1}. By (3.3.33),

$$(J^{-1})_{ii} \;=\; \frac{\det(J^{1:i-1})\det(J^{i+1:n})}{\det(J)}. \tag{3.3.34}$$

Suppose $\mathrm{diag}(J) = 0$. The nonsingularity of J, and Lemma 3.3.1 imply that n must be even. Since one of $J^{1:i-1}$ or $J^{i+1:n}$ must be of odd order, Lemma 3.3.1 and (3.3.34) implies that $(J^{-1})_{ii} = 0$ for $1 \leq i \leq n$.

Now suppose that $\mathrm{diag}(J^{-1}) = 0$. In this case, the key fact needed for the proof is that every leading (and trailing) principal submatrix of J of odd dimension is singular. This behavior can be observed in (3.3.32). We now prove this claim.

First, we observe that no two consecutive leading submatrices can be singular since otherwise, by the three-term recurrence

$$\det(J^{1:i}) \;=\; J_{ii}\det(J^{1:i-1}) - J_{i-1,i}J_{i,i-1}\det(J^{1:i-2}), \tag{3.3.35}$$

J would be singular. Similarly no two consecutive trailing submatrices can be singular. Now

$$(J^{-1})_{11} \;=\; 0 \quad\Rightarrow\quad \det(J^{2:n}) \;=\; 0 \quad \text{by} \quad (3.3.34),$$

and hence, $\det(J^{3:n}) \neq 0$. Consequently,

$$(J^{-1})_{22} \;=\; 0 \quad \Rightarrow \quad \det(J^{1:1}) \;=\; 0, \qquad \text{again by (3.3.34).}$$

Similarly by setting $i = 3, 4$ and so on in (3.3.34), we can conclude that the principal submatrices $J^{4:n}, J^{1:3}, J^{6:n}, J^{1:5}, \ldots$ must be singular. In particular, all leading principal submatrices of odd dimension are singular. Now, if n is odd then $n-2$ is also odd and by the above reasoning, $J^{1:n-2}$ is singular. However, by setting $i = n$ in (3.3.34), we see that

$$\det(J^{1:n-1}) \;=\; 0,$$

but no two consecutive principal submatrices of a nonsingular J can be singular. Hence n cannot be odd, and must be even.

When i is odd, the submatrices $J^{1:i}$ and $J^{1:i-2}$ are singular and by (3.3.35), we get

$$0 \;=\; \det(J^{1:i}) \;=\; J_{ii}\det(J^{1:i-1}),$$

and J_{ii} must be 0 since $\det(J^{1:i-1}) \neq 0$ in this case. Similarly, since n is even and by considering the trailing recurrence

$$\det(J^{i:n}) \;=\; J_{i,i}\det(J^{i+1:n}) - J_{i,i+1}J_{i+1,i}\det(J^{i+2:n}),$$

we can conclude that $J_{ii} = 0$ for even i. □

The pattern of zeros and nonzeros in J^{-1} that is used to prove the above result may also be deduced by using the following lemma which states that the upper (lower) triangular part of J^{-1} is a rank-one matrix. This result has appeared in [4, 14, 83, 78, 42].

Lemma 3.3.2 *Let J be a nonsingular unreduced tridiagonal matrix of order n. Then there exist vectors x, y, p and q such that*

$$(J^{-1})_{ik} = \begin{cases} x_i y_k, & i \leq k, \\ p_i q_k, & i \geq k. \end{cases}$$

The following corollary says that the spectrum of tridiagonals that have a zero diagonal is symmetric.

Corollary 3.3.1 *Let J be a tridiagonal matrix such that* $\mathrm{diag}(J) = 0$. *Then if λ is an eigenvalue of J, so is $-\lambda$. If v and w are the corresponding normalized eigenvectors, then $|v(i)| = |w(i)|$ for $1 \leq i \leq n$.*

Proof. Lemma 3.3.1 implies that eigenvalues occur in $\pm$ pairs. The corresponding eigenvector entries are equal in magnitude by (3.0.8) since $\chi'(\lambda) = -\chi'(-\lambda)$. $\square$

We have shown that for all $|\gamma_k|$ to be ∞, all eigenvalues of $J = \tilde{J} - \mu I$ must occur in $\pm$ pairs. Thus μ is equidistant to the two closest eigenvalues. Note that this result is consistent with Theorem 3.2.1. We will be careful to avoid this situation in our application. As we stated in the previous section, we will use Algorithm 3.2.1 to compute an eigenvector only when the corresponding eigenvalue is "sufficiently isolated" (we make the meaning of "isolated" more precise in the upcoming chapters). Then, as Theorem 3.2.1 shows, at least one γ_k must be small if μ is close to the eigenvalue. Breakdown in the triangular factorization does not hinder finding such a γ_k. The corresponding eigenvector is approximated well by Algorithm 3.3.1 given earlier in this section.

We now show how the ideas developed in this chapter enable us to correctly handle the matrix T of (3.0.4).

Example 3.3.1 [Minimum $|\gamma_k|$ leads to a good eigenvector approximation.] Consider T as in (3.0.4) of Example 3.0.1, and the eigenvalue approximation $\hat{\lambda} = 1$ so that the error in the eigenvalue is $\varepsilon + O(\varepsilon^2)$.

$$(T - I)^{-1} = \begin{bmatrix} 4/(-4+3\varepsilon) & 0 & -\sqrt{\varepsilon}/(-4+3\varepsilon) \\ 0 & 0 & 1/\sqrt{\varepsilon} \\ -\sqrt{\varepsilon}/(-4+3\varepsilon) & 1/\sqrt{\varepsilon} & (-4+10\varepsilon-5\varepsilon^2)/\varepsilon(-4+3\varepsilon) \end{bmatrix}$$

and since $\gamma_k^{-1} = (T^{-1})_{kk}$,

$$\gamma_1 \approx -1, \quad |\gamma_2| = \infty \quad \text{and} \quad \gamma_3 \approx \varepsilon.$$

Since $|\gamma_3|$ is the minimum, our theory predicts that $z = (T - I)^{-1} e_3$ is a good eigenvector approximation. Indeed,

$$\frac{z}{\|z\|} = \begin{bmatrix} \varepsilon^{3/2}/4 + O(\varepsilon^{5/2}) \\ \sqrt{\varepsilon} + O(\varepsilon^{3/2}) \\ 1 - \varepsilon/2 + O(\varepsilon^2) \end{bmatrix}.$$

and from (3.0.5) we see that each entry of the above vector agrees with the corresponding eigenvector entry in all its figures, i.e., the relative error in each entry is $O(\varepsilon)$. $\square$

3.4 Avoiding Divisions

On modern computers, divisions may be much more costly than multiplications. For example, on an IBM RS/6000 computer, a divide operation takes 19 cycles while a multiplication can be done in 1 cycle. In this section, we see how to eliminate all divisions from Algorithm 3.2.1. This may result in a faster computer implementation to compute an eigenvector. However, elimination of divisions results in a method that is susceptible to over/underflow and requires some care to ensure correctness. This section was inspired by Fred Gustavson's observation that n divisions are sufficient for a related application [74].

In this section, we will assume that T is the given unreduced real, symmetric tridiagonal matrix with diagonal elements $\alpha_1, \alpha_2, \ldots, \alpha_n$ and off-diagonal elements $\beta_1, \ldots, \beta_{n-1}$. Extensions to the normal case are trivial.

The crucial observation is that the rth column of $(T - \mu I)^{-1}$ may be written as

$$\chi^{1:n} w_r = \begin{bmatrix} (1 \cdot \chi^{r+1:n})(\beta_1 \cdots \beta_{r-1}) \\ \cdot \\ \cdot \\ (\chi^{1:r-2} \cdot \chi^{r+1:n})\beta_{r-1} \\ (\chi^{1:r-1} \cdot \chi^{r+1:n}) \\ (\chi^{1:r-1} \cdot \chi^{r+2:n})\beta_r \\ \cdot \\ \cdot \\ (\chi^{1:r-1} \cdot 1)(\beta_r \cdots \beta_{n-1}) \end{bmatrix} \tag{3.4.36}$$

for $r = 1, 2, \ldots, n$, where $\chi^{i:j} = \chi^{i:j}(\mu) = \det(\mu I - T^{i:j})$, taking $\chi^{n+1:n}(\mu) = \chi^{1:0}(\mu) = 1$. The above is easily derived from the Cauchy-Binet formula (3.3.33) for the entries of the inverse of a matrix. Emphasizing that

$$\left[(T - \mu I)^{-1}\right]_{kk} = \frac{\chi^{1:k-1}(\mu) \cdot \chi^{k+1:n}(\mu)}{\chi^{1:n}(\mu)} \tag{3.4.37}$$

we give an alternate method to compute an eigenvector of an isolated eigenvalue.

Algorithm 3.4.1 [Computing an eigenvector with no divisions.]

1. Compute $\chi^{1:1}(\mu), \chi^{1:2}(\mu), \ldots, \chi^{1:n-1}(\mu)$ using the 3-term recurrence

$$\chi^{1:i}(\mu) = (\mu - \alpha_i)\chi^{1:i-1}(\mu) - \beta_{i-1}^2 \chi^{1:i-2}(\mu), \quad \chi^{1:0} = 1, \quad \beta_0 = 0.$$

2. Compute $\chi^{n:n}(\mu), \chi^{n:n-1}(\mu), \ldots, \chi^{n:2}(\mu)$ using the 3-term recurrence

$$\chi^{i:n}(\mu) = (\mu - \alpha_i)\chi^{i+1:n}(\mu) - \beta_i^2 \chi^{i+2:n}(\mu), \quad \chi^{n+1:n} = 1, \quad \beta_n = 0.$$

3. Compute

$$\Delta_k = \chi^{1:k-1}(\mu) \cdot \chi^{k+1:n}(\mu) \quad \text{for} \quad k = 1, 2, \ldots, n$$

   and choose the index r where $|\Delta_k|$ is maximum. In exact arithmetic, by (3.1.13) and (3.4.37), this index r is identical to the one found in Step 2 of Algorithm 3.2.1.

4. Form the vector $w_r \cdot \chi^{1:n}(\mu)$ by multiplying the appropriate χ's and β's, as given in (3.4.36). □

The total cost involved in the above algorithm is $8n$ multiplications and $3n$ additions. Some multiplications may be saved when computing more than one eigenvector by forming and saving the products of β's that are used more than once. There is also some cost involved in monitoring and handling potential overflow and underflow. The cost of the above algorithm should be compared to that of Algorithm 3.2.1 which requires $2n$ divisions, $3n$ multiplications and $4n$ additions if an additive formula to compute γ_k is used. The vector output by both these algorithms may be normalized at a further cost of 1 square root, 1 division, $2n$ multiplications and $n - 1$ additions.

However, as is well known, the quantities $\chi^{1:i}(\mu)$ and $\chi^{i:n}(\mu)$ can grow and decay rapidly, and hence the recurrences to compute them are susceptible to severe overflow and underflow problems. To overcome this, these quantities need to be monitored and rescaled when approaching overflow or underflow. Some modern computers allow such monitoring to be done in hardware but alas, this facility is not presently visible to the software developer who programs in a high level language [92].

3.4.1 Heuristics for choosing r

In Algorithm 3.2.1, the $2n$ quantities $D_+(1), \ldots, D_+(n)$ and $D_-(1), \ldots, D_-(n)$ are computed to find the minimum among all the γ_k's. Once the right index r is chosen, half of these quantities are discarded and the other half used to compute the desired eigenvector approximation. The situation is similar in Algorithm 3.4.1.

If we have a reasonable guess at a "good" index r, we can check the value of γ_r by only computing the n quantities $D_+(1), \ldots, D_+(r-1), D_-(r+1), \ldots, D_-(n)$ and γ_r. If

γ_r is small enough we can accept r, otherwise we can compute all the γ's and choose the minimum as before. In case the guess is accurate, we can save approximately half the work in the computation of the eigenvector. We now examine the proof of the Gerschgorin Disk Theorem that suggests a heuristic choice of r.

Theorem 3.4.1 [Gerschgorin Disk Theorem]. *Let B be a matrix of order n with an eigenvalue λ. Then λ lies in one of the Gerschgorin disks, i.e., $\exists r$ such that*

$$|\lambda - B_{rr}| \;\le\; \sum_{k \neq r} |B_{rk}|$$

Proof. Let v be the eigenvector corresponding to λ, and let $|v(r)| = \|v\|_\infty$. Consider the rth equation of $Bv = \lambda v$,

$$\begin{aligned} \lambda v(r) &= B_{rr} v(r) + \sum_{k \neq r} B_{rk} v(k) \\ \Rightarrow\ |\lambda - B_{rr}| &\le \sum_{k \neq r} |B_{rk}||v(k)/v(r)| \end{aligned}$$

from which the result follows. □

Thus if the kth entry of v is the largest, then λ must lie in the kth Gerschgorin disk. Consequently, if λ lies in just one Gerschgorin disk then the corresponding entry of v must be the largest. However λ may lie in many Gerschgorin disks and there is no guarantee that the corresponding entries of v are large. But we may use the following heuristic

Pick r such that λ lies in the rth Gerschgorin disk with minimum radius $\sum_{k \neq r} |B_{rk}|$.

Such a heuristic is inexpensive to compute and can lead to considerable savings on large matrices, especially ones that are even mildly graded. Such matrices seem to arise in many real life applications. Note that the above heuristic gives the correct choice of r in the example tridiagonals of (3.0.4) and (3.0.6).

We do not make any claim about the optimality of the above heuristic, and it may be possible to obtain better ones. The purpose of this section was to furnish the idea of heuristically picking r and checking it in order to potentially save half the computations.

3.5 Twisted Q Factorizations — A Digression*

In this section, we introduce twisted QR-type factorizations. The reader who is pressed for time and is primarily interested in seeing how to adapt inverse iteration to

$$
\begin{bmatrix} x & x & & & \\ x & x & x & & \\ & x & x & x & \\ & & x & x & x \\ & & & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & & & & \\ x & x & x & & \\ x & x & x & x & x \\ & & x & x & x \\ & & & & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & & & & \\ x & x & & & \\ x & x & x & x & x \\ & x & x & x & x \\ & & & & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & & & & \\ x & x & & & \\ x & x & x & x & x \\ & x & & x & x \\ & & & & x \end{bmatrix}
$$

Figure 3.2: Twisted Q Factorization at $k = 3$ (the next elements to be annihilated are circled): forming N_k.

compute mutually orthogonal eigenvectors may skip to the next chapter.

We define a twisted orthogonal or Q factorization at position k as

$$J = N_k Q_k^* \tag{3.5.38}$$

where Q_k is an orthogonal matrix and N_k is such that $N_k(i,j) = 0$ for i,j such that $j > i$ when $i < k$ and $k \leq j < i$ when $i \geq k$. Note that N_k is a "permuted" triangular matrix, since there exists a symmetric permutation of N_k that is a triangular matrix. Figure 3.2 illustrates how to compute such a factorization — it may be obtained by starting an LQ factorization from the left of the matrix stopping it at column k, and then doing an RQ factorization from the right of the matrix till there is *a singleton in the kth column* (note that we are doing column operations here in contrast to row operations in Section 3.1 — hence we refer to the left and right of the matrix rather than the top and bottom). Using the fact that N_k is "essentially" triangular and assuming that J is of full rank, we can show that the twisted Q factorization (3.5.38) is unique, modulo the signs of the diagonal elements of N_k. We could have done row operations instead and looked at twisted Q factorizations that are obtained by doing QR from the top of the matrix, and QL from the bottom to obtain a singleton in the kth row. We have chosen instead to look at column twisted Q factorizations in order to make a direct connection with the twisted triangular factorizations of Section 3.1.

We now develop some theory about twisted Q factorizations along the lines of Section 3.1.

Theorem 3.5.1 *Let J be a nonsingular tridiagonal $n \times n$ complex matrix with the column twisted Q factorization at k given by (3.5.38) for $1 \leq k \leq n$. Let δ_k be the singleton in the* *kth column of N_k, i.e., $\delta_k = N_k(k,k)$. Then*

$$\frac{1}{|\delta_k|^2} = e_k^*(JJ^*)^{-1}e_k. \tag{3.5.39}$$

Proof.

$$\begin{aligned} JJ^* &= N_k Q_k^* Q_k N_k^* \\ \Rightarrow\ e_k^*(JJ^*)^{-1}e_k &= e_k^* N_k^{-*} N_k^{-1} e_k. \end{aligned}$$

Since $N_k e_k = \delta_k e_k$, the result (3.5.39) follows. □

We could endeavor to get expressions for δ_k, as we did for γ_k in (3.1.12), in terms of the left and right factorizations. In particular, we can attempt to express δ_k in terms of the *Schur parameters* $\{\cos\theta_k^+, k = 1, \ldots, n-1\}$, $\{\cos\theta_k^-, k = 1, \ldots, n-1\}$ that determine the orthogonal matrices in the L_+Q_+ and R_-Q_- factorizations, and the diagonal elements of L_+ and R_-. However, we choose not to do so since the expressions are not stated simply with our present notation, and do not add to our exposition.

The following results parallel those of Sections 3.1 and 3.2 as do the corresponding proofs.

Corollary 3.5.1 *Let J be a nonsingular tridiagonal matrix. With the notation of Theorem 3.5.1,*

$$\sum_{i=1}^n |\delta_i|^{-2} \;=\; \mathrm{Trace}((JJ^*)^{-1}) \;=\; \sum_{i=1}^n \sigma_i^{-2},$$

where $\sigma_1, \sigma_2, \ldots, \sigma_n$ are the singular values of J.

Theorem 3.5.2 *Let $J = \tilde{J} - \mu I$ be a tridiagonal matrix with the twisted factorization (3.5.38) for a fixed value of k. Then*

$$(\tilde{J} - \mu I)q_k = \delta_k e_k,$$

where q_k is the kth column of Q_k.

Proof. Since $N_k e_k = \delta_k e_k$ and $\tilde{J} - \mu I = N_k Q_k^*$,

$$(\tilde{J} - \mu I)Q_k \;=\; N_k \;\Rightarrow\; (\tilde{J} - \mu I)q_k \;=\; \delta_k e_k.$$

□

Thus the vector q_k is just a normalized version of $z^{(k)}$ defined in (3.2.25) in Theorem 3.2.2. The following theorems confirm that the corresponding residual norms are equal.

Theorem 3.5.3 *Let* $J = \tilde{J} - \mu I$ *be a nonsingular tridiagonal matrix and let* δ_k, γ_k *and* $z^{(k)}$ *be as defined in (3.5.39), (3.1.13) and (3.2.25) respectively. Then*

$$\frac{|\gamma_k|}{\|z^{(k)}\|} = |\delta_k| \tag{3.5.40}$$

Proof. Since $(\tilde{J} - \mu I)z^{(k)} = \gamma_k e_k$,

$$\begin{aligned} \frac{\|z^{(k)}\|^2}{|\gamma_k|^2} &= e_k^*(\tilde{J} - \mu I)^{-*}(\tilde{J} - \mu I)^{-1}e_k \\ &= \frac{1}{|\delta_k|^2}. \end{aligned}$$

The last equality is just (3.5.39). □

Thus when J is normal, the bound (3.2.28) for $|\gamma_k|/\|z^{(k)}\|$ applies to $|\delta_k|$. The connection between q_k and $z^{(k)}$ suggests the following alternative to Algorithm 3.2.1 that is guaranteed to find a small δ_k for a nearly singular $\tilde{J} - \mu I$, even when the eigenvalue near μ is not isolated.

Algorithm 3.5.1 [Computing an eigenvector of any eigenvalue.]

1. Compute the diagonal elements δ_k for all n twisted Q factorizations of $\tilde{J}$.
2. Choose the index r where $|\delta_k|$ is minimum.
3. Form the vector q_r, where q_r is as in Theorem 3.5.2.

□

The above results may appear somewhat surprising but the reader is reminded of the intimate connection between the LR algorithm, the QR algorithm and inverse iteration [110]. For a non-normal matrix also, as the following theorem implies, there is at least one δ_k that indicates its nearness to singularity.

Theorem 3.5.4 *Let* B *be a nonsingular matrix of order* n. *Let* $\sigma_{min} = \sigma_n$ *be the smallest singular value of* B *and* u_{min} *be the corresponding left singular vector. If*

$$\frac{1}{|\delta_k|^2} = e_k^*(BB^*)^{-1}e_k \quad \text{for } k = 1, 2, \ldots, n$$

then

$$\begin{aligned} |\delta_k| &= \frac{\sigma_{min}}{|u_{min}(k)|}\left[1 + \left(|u_{min}(k)|^{-2} - 1\right)\mathcal{A}_2\right]^{-1/2}, \\ &\le \frac{\sigma_{min}}{|u_{min}(k)|} \le \sqrt{n}\,\sigma_{min} \quad \text{for at least one } k \end{aligned}$$

where $\mathcal{A}_2$ is a weighted arithmetic mean of $\left\{ \left(\frac{\sigma_{min}}{\sigma_i}\right)^2, \; i < n \right\}$, $0 < \mathcal{A}_2 < (\sigma_{min}/\sigma_{n-1})^2$.

Proof. The proof is similar to that of Theorem 3.2.3 using the following eigendecomposition of BB^*,

$$BB^* = U\Sigma^2 U^*.$$

$\square$

In Section 3.1, we showed that $\gamma_k = 0$ when J is singular and admits the triangular decompositions (3.1.11) (see Corollary 3.1.4 and the comments immediately following it). Similarly, it can be shown that if J is an unreduced tridiagonal that is singular, then $\delta_k = 0$ in the factorization of (3.5.38) for any k.

3.5.1 "Perfect" Shifts are perfect

In this section, we give an application of twisted Q factorizations. If λ is an eigenvalue of the unreduced tridiagonal $\tilde{J}$, then the last diagonal element of R, R_{nn}, in the factorization

$$\tilde{J} - \lambda I = QR, \quad Q^*Q = I, \quad R \text{ upper triangular}$$

must be zero in exact arithmetic. The QR iteration on $\tilde{J}$ with shift μ yields $\tilde{J}_1$:

$$\tilde{J} - \mu I = QR, \quad RQ + \mu I = \tilde{J}_1.$$

When $\mu = \lambda$, the $(n-1, n)$ entry of the tridiagonal $\tilde{J}_1$ is zero and hence λ may be *deflated* from $\tilde{J}_1$ by simply removing the last row and column.

It may be expected that if μ is close to λ, then R_{nn} would be small and deflation would again occur in one iteration. Based on this expectation, Parlett proposed the following strategy to compute all the eigenvalues and eigenvectors of a real, symmetric tridiagonal matrix T [110, p.173].

1. Compute eigenvalues by a fast algorithm such as the PWK algorithm, which is a QR algorithm free of square roots.

2. Run the QR algorithm using the computed eigenvalues as shifts, accumulating the rotations to form the eigenvector matrix.

The expectation was that deflation would occur in one step, thus saving approximately 50% of the dominant $O(n^3)$ computation in the QR algorithm, which typically requires an average of 2-3 iterations per eigenvalue.

However,

$$\begin{aligned} T-\mu I &= QR \\ \Rightarrow\ e_n^T(T-\mu I)^{-2}e_n &= \frac{1}{|R_{nn}|^2} \\ \Rightarrow\ \sum_{i=1}^{n}\frac{|v_i(n)|^2}{(\lambda_i-\mu)^2} &= \frac{1}{|R_{nn}|^2} \end{aligned}$$

showing that if the bottom entry of the desired eigenvector is tiny, then R_{nn} is not small. As a result, immediate deflation is not always seen in the perfect-shift QR algorithm, and it was found not to perform any better than the standard QR algorithm with Wilkinson's shift [69]. The situation is similar to that with $D_+(n)$ discused in Section 3.1. And it is natural for us, as in Section 3.1, to turn to twisted factorizations to detect singularity of $T-\mu I$. Let

$$T-\mu I = Q_k N_k \tag{3.5.41}$$

be a twisted Q factorization of $T-\mu I$. This is essentially identical to the column twisted L factorization of J in (3.5.38) since T is symmetric. In the above factorization the kth row of N_k is a singleton with a non-zero on the diagonal. By the theory developed in the previous section, in particular Theorem 3.5.4, there must be a k such that $N_k(k,k)$ is small if μ is a good approximation to an eigenvalue of T.

With this choice of k, we can form the matrix

$$A_1 = N_k Q_k + \mu I \tag{3.5.42}$$

following the standard QR algorithm. Since δ_k is small all the off-diagonal entries of the kth row of A_1 must be small. Since $A_1 = Q_k^* T Q_k$ is symmetric the off-diagonal entries in the kth column are also tiny. An eigenvalue close to μ may be deflated by *crossing out the entire kth row and column of* A_1. Thus deflation occurs in exactly one step when perfect shifts are used. For a more detailed treatment, the reader should wait for [41].

Each matrix in the sequence obtained in the standard QR algorithm is tridiagonal. Does (3.5.42) preserve tridiagonal form? The reduction uniqueness theorem in [110, Section 7-2] states that a tridiagonal formed by an orthogonal similarity transformation, $T = Q^*AQ$, is determined by A and the first or last column of Q. It follows that A_1 cannot be tridiagonal in general since, except when $k=1$ or n, the first and last columns of Q_k in (3.5.41) are independent. However, it can be shown that the matrix obtained by deleting the kth row and column of A_1 is tridiagonal with an additional bulge at new position

$(k+1, k)$ after deflation. This bulge can then be "chased" out of the top or bottom of the matrix, whichever is nearer. This yields a tridiagonal matrix to which the next perfect shift may now be applied.

To form the eigenvector matrix the $n-1$ rotations that form Q_k must be accumulated as must be the rotations used to chase the bulge out of the end of the deflated matrix. Thus in the worst case 1.5 iterations are needed per eigenvalue to compute the eigenvectors. Assuming 1-1.5 iterations per eigenvalue, all the eigenvectors may be obtained at a cost of $3n^3$-$4.5n^3$ operations. This is about twice as fast as current QR algorithms, assuming that n is large enough to disregard the extra computation involved in the initial computation of the eigenvalues and in finding k at each step. Heuristics for guessing k may also be used as discussed in Section 3.4.1.

Standard QR algorithms use an implicit version to achieve greater accuracy. It remains to be seen if similar techniques may be used in the algorithm outlined above. It should also be possible to find k by using a procedure free from square roots as in the PWK algorithm. But as we mentioned earlier, this section is a digression from the main theme of this thesis and we plan to explore these questions elsewhere [41].

The reader may wonder about our inconsistency in finding the eigenvalues by a standard QR algorithm but using twisted factorizations in finding the eigenvectors. Is it not possible to use twisted factorizations to find the eigenvalues also? We believe that the latter might indeed be the right approach not only in speeding up the algorithm but also to get higher accuracy since the standard QR algorithm may violently rearrange the matrix thereby destroying accuracy [32]. However, this is not straightforward due to the complication that A_1 in (3.5.42) is not tridiagonal. When deflation does not occur, chasing the bulges in A_1 off the top or bottom of the matrix will give the standard QR or QL algorithm respectively (by the reduction uniqueness theorem in [110, Section 7-2]). Thus in order to develop a "twisted algorithm", we must give up the tridiagonal nature of the intermediate matrices. The number of bulges increase by 2 at each step, and it is for future research to determine if it is computationally feasible to handle this increase. We also believe that such a twisted algorithm may lead to better shifts than the Francis or Wilkinson shifts.

3.6 Rank Revealing Factorizations*

In earlier sections of this study, we exhibited twisted triangular and twisted orthogonal (or Q) factorizations for tridiagonals. In our applications, we were successfully able to reveal the singularity of the tridiagonal by one of the n possible twisted factors.

In this section, we show that such factorizations can also be done for denser matrices and may be used to reveal the rank of the matrix. We convey our ideas through pictures and make no pretence to being complete in this section.

Formally, we define a row twisted triangular factorization of a matrix B at position k as the decomposition

$$B = N_k D_k \tilde{N}_k \tag{3.6.43}$$

where D_k is diagonal, while N_k and $\tilde{N}_k$ are such that $(N_k)_{ij} = 0$ for i,j such that $j > i$ when $i < k$ and $k \leq j < i$ otherwise, and $\tilde{N}_k^*$ has the same requirement on its zeros. We take both N_k and $\tilde{N}_k$ to have 1's on their diagonals. Note that N_k and $\tilde{N}_k$ are simultaneously permuted triangular matrices, i.e., $\exists$ a permutation P_k such that $P_k^T N_k P_k$ is lower triangular and $P_k^T \tilde{N}_k P_k$ is upper triangular. (3.6.43) may be written as

$$P_k^T B P_k = (P_k^T N_k P_k)(P_k^T D_k P_k)(P_k^T \tilde{N}_k P_k)$$

and this is just the LDU decomposition of $P_k^T B P_k$. This implies the uniqueness of the factorization in (3.6.43).

Figures 3.3 and 3.5 suggest a way to compute the twisted LU decomposition for Hessenberg and dense matrices respectively. Note that when B is normal, Theorem 3.2.1 is applicable with $\gamma_k = D_k(k,k)$. Thus when B is nearly singular with an isolated small eigenvalue, Theorem 3.2.1 implies the existence of a small bottom pivot in all possible LU-factorizations of B that can be obtained by symmetric permutations of the rows and columns of B. A generalization of such a result was proved by Tony Chan in 1984 [19] (he considered non-normal B as well, and arbitrary row and column permutations of B). We briefly compare our results with Chan's results of [19] at the end of this section.

We now formally define a row twisted Q decomposition of B at position k as

$$B = Q_k N_k \tag{3.6.44}$$

where Q_k is orthogonal and N_k is a permuted triangular matrix as made precise earlier in this section. Figures 3.4 and 3.5 suggest computational procedures to find such a factorization

$$
\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \\ & & x & x & \\ & & & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & x & x & \\ & & & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & \\ & & x & x & \\ & & & x & x \end{bmatrix}
$$

$$
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & & \\ & & x & x & \\ & & & x & x \end{bmatrix}
$$

Figure 3.3: Twisted Triangular Factorization of a Hessenberg matrix at $k = 3$ (the next elements to be annihilated are circled)

$$
\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \\ & & x & x & \\ & & x & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & x & x & \\ & & x & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & \\ & & x & x & \\ & & x & x & x \end{bmatrix}
$$

$$
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & & \\ & & x & x & \\ & & x & x & x \end{bmatrix}
$$

Figure 3.4: Twisted Orthogonal Factorization of a Hessenberg matrix at $k = 3$ (the next elements to be annihilated are circled)

$$
\begin{bmatrix} x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \\ x & x & x & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \\ & x & x & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & x \\ & & x & x & x \\ & & x & x & x \end{bmatrix}
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & x & \\ & & x & x & \\ & & x & x & x \end{bmatrix}
$$

$$
\longrightarrow
\begin{bmatrix} x & x & x & x & x \\ & x & x & x & x \\ & & x & & \\ & & x & x & \\ & & x & x & x \end{bmatrix}
$$

Figure 3.5: The above may be thought of as a twisted triangular or orthogonal factorization of a dense matrix at $k = 3$ (the next elements to be annihilated are circled)

for Hessenberg and dense matrices respectively. Since $\exists\ P_k$ such that $P_k^T N_k P_k$ is upper triangular, (3.6.44) may be written as

$$BP_k \; = \; Q_k P_k (P_k^T N_k P_k)$$

to give a QR decomposition of B after a column permutation. Thus the factorization (3.6.44) is unique in the same sense as the QR factorization of B is unique.

The result of Theorem 3.5.4 holds for the decomposition (3.6.44) with J replaced by B^*. This proves the existence of a column permutation of B such that the bottom element of R in B's QR decomposition is tiny when B is nearly singular. This result was proved by Chan in [18] and earlier by Golub, Klema and Stewart in [66].

Thus twisted factorizations can be rank-revealing. Rank-revealing LU and QR factorizations have been extensively studied and several algorithms to compute such factorizations exist. Twisted factorizations seem to have been overlooked and may offer computational advantages. We consider our results outlined above to be stronger than those of Chan [19, 18] and Golub *et al.* [66] since the permutations we consider are restricted. In particular, as seen in the tridiagonal and Hessenberg case, twisted factorizations *respect the sparsity structure of the given matrix*, and thus may offer computational advantages in terms of speed and accuracy.

We believe that twisted factorizations of Hessenberg matrices can be used for a better solution to the non-symmetric eigenproblem. For banded and sparse matrices, twisted factorizations may offer considerable savings in computing rank-revealing factorizations. We intend to conduct further research in these two areas. The innovative reader may be able to think of other applications.

Chapter 4

Computing orthogonal eigenvectors when relative gaps are large

In the last chapter, we showed how to compute the eigenvector corresponding to an isolated eigenvalue of a normal, unreduced tridiagonal matrix. In particular, we showed how to choose a right hand side for inverse iteration that has a guaranteed large component in the direction of the desired eigenvector. Even though this is both a theoretical and practical advance, it does not solve the most pressing problem with inverse iteration — that of *computing approximate eigenvectors that are numerically orthogonal.*

When eigenvalues are well separated, vectors that give a small residual norm, as in (1.1.1), are numerically orthogonal. In such a case, the methods of Chapter 3 are sufficient. However this is not the case with close eigenvalues and current implementations of inverse iteration resort to explicit orthogonalization. As mentioned in Chapter 1 and Section 2.8.1 this can take an inordinately large amount of computation, and even then, the computed vectors may not be numerically orthogonal.

The rest of this thesis is devoted to the computation of "good eigenvectors" that are *automatically* numerically orthogonal thus avoiding the need for explicit orthogonalization. In this chapter, we show that an alternate representation of a tridiagonal is the key to better approximations. Coupled with the theory developed in Chapter 3 this allows us to compute orthogonal eigenvectors when the corresponding eigenvalues have large relative gaps, even though the eigenvalues may be very close together in an absolute sense. More precisely,

1. In Section 4.1, we extol the virtues of high accuracy. Tridiagonals do not always "allow" such high accuracy computations and in Section 4.2, we advocate representing the tridiagonal as a product of bidiagonal matrices.

2. In Section 4.3, we recap earlier work on relative perturbation theory as applied to bidiagonal matrices.

3. In Section 4.4.1, we give qd-like recurrences that allow us to exploit the properties of bidiagonals in order to compute highly accurate approximations to eigenvectors. In Section 4.4.2, we do a roundoff error analysis of their computer implementations while in Section 4.4.3 we give an algorithm to compute eigenvectors based on these qd-like recurrences.

4. In Section 4.5 we prove that the dot products between the eigenvectors computed by the algorithm given in Section 4.4.3 are inversely proportional to the relative gaps between eigenvalues. As a consequence, the computed eigenvectors are numerically orthogonal if the corresponding eigenvalues are relatively far apart. *These results are new and are a major advance towards our goal of obtaining guaranteed numerical orthogonality in $O(n^2)$ time.*

5. In Section 4.6, we present numerical results that support the above claims.

We consider the case of eigenvalues that have small relative gaps in the next two chapters.

4.1 Benefits of High Accuracy

Consider the matrix T_0 in (2.8.16) of Section 2.8.1,

$$T_0 = \begin{bmatrix} 1 & \sqrt{\varepsilon} & 0 \\ \sqrt{\varepsilon} & 7\varepsilon/4 & \varepsilon/4 \\ 0 & \varepsilon/4 & 3\varepsilon/4 \end{bmatrix}$$

where ε is the machine precision. T_0 has two small eigenvalues $\lambda_1 = \varepsilon/2 + O(\varepsilon^2)$ and $\lambda_2 = \varepsilon + O(\varepsilon^2)$, while $\lambda_3 \approx 1$. Suppose that $\hat\lambda_1$ and $\hat\lambda_2$ are approximations to λ_1 and λ_2, and inverse iteration as in (2.7.4) is used to find the corresponding eigenvectors. In exact arithmetic, taking $b_1 = \sum_i \xi_i v_i$ and $b_2 = \sum_i \eta_i v_i$, $\|b_1\| = \|b_2\| = 1$, we get

$$y_1 \;=\; (T_0 - \hat\lambda_1 I)^{-1} b_1 \;=\; \frac{\xi_1}{\lambda_1 - \hat\lambda_1}\left(v_1 + \frac{\xi_2}{\xi_1}\frac{\lambda_1 - \hat\lambda_1}{\lambda_2 - \hat\lambda_1} v_2 + \frac{\xi_3}{\xi_1}\frac{\lambda_1 - \hat\lambda_1}{\lambda_3 - \hat\lambda_1} v_3\right), \quad (4.1.1)$$

$$y_2 = (T_0 - \hat\lambda_2 I)^{-1} b_2 = \frac{\eta_2}{\lambda_2 - \hat\lambda_2}\left(\frac{\eta_1}{\eta_2}\frac{\lambda_2 - \hat\lambda_2}{\lambda_1 - \hat\lambda_2} v_1 + v_2 + \frac{\eta_3}{\eta_2}\frac{\lambda_2 - \hat\lambda_2}{\lambda_3 - \hat\lambda_2} v_3\right). \quad (4.1.2)$$

We assume that ξ_1, ξ_2, η_1 and η_2 are $O(1)$. Earlier we showed that y_1 and y_2 are nearly parallel if $\hat\lambda_1 \approx \hat\lambda_2$, and in such a case, EISPACK and LAPACK implementations fail to deliver orthogonal vectors despite explicit orthogonalization. See Section 2.8.1 and Case Study A for more details. It is clearly desirable that $\hat\lambda_1$ and $\hat\lambda_2$ be more accurate so that their difference is discernible. Given limited precision to represent numbers in a computer, how accurate can $\hat\lambda_1$ and $\hat\lambda_2$ be? We say that $\hat\lambda_i$ agrees with λ_i to d digits if

$$\frac{|\lambda_i - \hat\lambda_i|}{|\lambda_i|} = 10^{-d}.$$

The IEEE double precision format allows for 53 bits of precision, thus $\varepsilon = 2^{-52} \approx 2.2 \cdot 10^{-16}$ and d can take a maximum value of about 16. Suppose $\hat\lambda_1$ and $\hat\lambda_2$ agree with λ_1 and λ_2 respectively to d digits, where $d \geq 1$. Since λ_1 and λ_2 do not agree in any digit,

$$\frac{|\lambda_i - \hat\lambda_i|}{|\lambda_j - \hat\lambda_i|} = O(10^{-d}) \quad \text{where} \quad i = 1, 2, \;\; j \neq i$$

and by (4.1.1) and (4.1.2),

$$\frac{|y_1^T y_2|}{\|y_1\| \cdot \|y_2\|} = O\left(\frac{|\lambda_1 - \hat\lambda_1|}{|\lambda_2 - \hat\lambda_1|} + \frac{|\lambda_2 - \hat\lambda_2|}{|\lambda_1 - \hat\lambda_2|}\right) = O(10^{-d}).$$

Thus the more accurate $\hat\lambda_1$ and $\hat\lambda_2$ are, the more orthogonal are y_1 and y_2. In fact when $d = 16$, i.e., when $\hat\lambda_1$ and $\hat\lambda_2$ have full relative accuracy, y_1 *and* y_2 *are numerically orthogonal and no further orthogonalization is needed.*

Of course, the above is true only in exact arithmetic. We now see why the standard representation of a tridiagonal matrix does not allow computation of eigenvalues to such high accuracy.

4.2 Tridiagonals Are Inadequate

A real, symmetric tridiagonal matrix T is traditionally represented by its $2n-1$ diagonal and off-diagonal elements. In this section, we show that for our computational purposes it is better to represent T by its bidiagonal factors.

To account for the roundoff errors that occur in a computer implementation, it is common practice to show that the computed eigenvalues and eigenvectors are exact for a slightly perturbed matrix. For highly accurate algorithms such as bisection we can show that the eigenvalues computed for T are exact eigenvalues of $T+\delta T$ where δT represents a small *componentwise* perturbation in only the off-diagonals of T. In the following example, we show that such a perturbation can cause a large relative change in the eigenvalues and eigenvector entries.

Example 4.2.1 [Tridiagonals are inadequate.] Consider the tridiagonal

$$T_1 = \begin{bmatrix} 1-\sqrt{\varepsilon} & \varepsilon^{1/4}\sqrt{1-7\varepsilon/4} & 0 \\ \varepsilon^{1/4}\sqrt{1-7\varepsilon/4} & \sqrt{\varepsilon}+7\varepsilon/4 & \varepsilon/4 \\ 0 & \varepsilon/4 & 3\varepsilon/4 \end{bmatrix},$$

and a small relative perturbation

$$T_1+\delta T_1 = \begin{bmatrix} 1-\sqrt{\varepsilon} & \varepsilon^{1/4}(1+\varepsilon)\sqrt{1-7\varepsilon/4} & 0 \\ \varepsilon^{1/4}(1+\varepsilon)\sqrt{1-7\varepsilon/4} & \sqrt{\varepsilon}+7\varepsilon/4 & \varepsilon(1+\varepsilon)/4 \\ 0 & \varepsilon(1+\varepsilon)/4 & 3\varepsilon/4 \end{bmatrix}.$$

where ε is the machine precision. The two smallest eigenvalues of T_1 and $T_1+\delta T_1$ are[1]:

$$\begin{aligned} \lambda_1 &= \varepsilon/2+\varepsilon^{3/2}/8+O(\varepsilon^2), \\ \lambda_1+\delta\lambda_1 &= \varepsilon/2-7\varepsilon^{3/2}/8+O(\varepsilon^2) \end{aligned}$$

and

$$\begin{aligned} \lambda_2 &= \varepsilon-\varepsilon^{3/2}/8+O(\varepsilon^2), \\ \lambda_2+\delta\lambda_2 &= \varepsilon-9\varepsilon^{3/2}/8+O(\varepsilon^2). \end{aligned}$$

Thus

$$\left|\frac{\delta\lambda_i}{\lambda_i}\right| = O(\sqrt{\varepsilon}), \quad i=1,2$$

and the relative change in these eigenvalues is much larger than the initial relative perturbations in the entries of T_1. Similarly the corresponding eigenvectors of T_1 and $T_1+\delta T_1$ are:

$$v_1 = \begin{bmatrix} \sqrt{\frac{\varepsilon}{2}}(1+\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon^{5/4}) \\ -\frac{1}{\sqrt{2}}(1-\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon) \\ \frac{1}{\sqrt{2}}(1-\frac{3\varepsilon}{4})+O(\varepsilon^{3/2}) \end{bmatrix} \qquad v_1+\delta v_1 = \begin{bmatrix} \sqrt{\frac{\varepsilon}{2}}(1+\frac{5\sqrt{\varepsilon}}{2})+O(\varepsilon^{5/4}) \\ -\frac{1}{\sqrt{2}}(1+\frac{3\sqrt{\varepsilon}}{2})+O(\varepsilon) \\ \frac{1}{\sqrt{2}}(1-2\sqrt{\varepsilon})+O(\varepsilon) \end{bmatrix}.$$

[1]we artfully constructed this matrix to have the desired behavior which may be verified by using a symbol manipulator such as Maple [21] or Mathematica [137]

and

$$v_2 = \begin{bmatrix} -\sqrt{\frac{\varepsilon}{2}}(1+\frac{\sqrt{\varepsilon}}{2}) + O(\varepsilon^{5/4}) \\ \frac{1}{\sqrt{2}}(1-\frac{\sqrt{\varepsilon}}{2}) + O(\varepsilon) \\ \frac{1}{\sqrt{2}}(1+\frac{3\varepsilon}{4}) + O(\varepsilon^{3/2}) \end{bmatrix}, \quad v_2 + \delta v_2 = \begin{bmatrix} -\sqrt{\frac{\varepsilon}{2}}(1-\frac{3\sqrt{\varepsilon}}{2}) + O(\varepsilon^{5/4}) \\ \frac{1}{\sqrt{2}}(1-\frac{5\sqrt{\varepsilon}}{2}) + O(\varepsilon) \\ \frac{1}{\sqrt{2}}(1+2\sqrt{\varepsilon}) + O(\varepsilon) \end{bmatrix},$$

whereby

$$\left|\frac{\delta v_i(j)}{v_i(j)}\right| = O(\sqrt{\varepsilon}) \quad \text{for} \quad i = 1, 2 \quad \text{and} \quad j = 1, 2, 3.$$

Since a small relative change of ε in the off-diagonal entries of T_1 results in a much larger relative change in its eigenvalues and eigenvectors, we say that T_1 does not determine its eigenvalues and eigenvector components to high relative accuracy. Consequently, it is unlikely that we can compute numerically orthogonal eigenvectors without explicit orthogonalization. To confirm this, we turned off all orthogonalization in the EISPACK and LAPACK implementations of inverse iteration and found the computed vectors to have dot products as large as $O(\sqrt{\varepsilon})$. For more details, see Case Study B. □

The situation can be retrieved by computing the bidiagonal Cholesky factor of T_1,

$$T_1 = \tilde{L}_1 \tilde{L}_1^T.$$

It is now known that small relative changes to the entries of a bidiagonal matrix cause small relative changes to its singular values (note that the singular values of $\tilde{L}_1$ are the square roots of the eigenvalues of T_1). When relative gaps between the singular values of $\tilde{L}_1$ are large, it is also known that the changes in the corresponding singular vectors are small. In the rest of this chapter, we show how to exploit this property of bidiagonal matrices to compute numerically orthogonal eigenvectors without any explicit orthogonalization.

The relative perturbation results for bidiagonal matrices mentioned above have appeared in [89, 29, 35, 49, 50, 51, 100, 101]. We state the precise results in the next section.

4.3 Relative Perturbation Theory for Bidiagonals

In 1966, Kahan proved the remarkable result that a real, symmetric tridiagonal with zero entries on the diagonal determines its eigenvalues to high relative accuracy with respect to small relative perturbations in its off-diagonal elements. This result appears to have lain neglected in the technical report [89] until Demmel and Kahan used it in 1990 to devise a method to compute the singular values of a bidiagonal matrix to high relative accuracy [35]. Subsequently, these results have been extended and simplified, and we cite some contributions during the course of this section.

Consider the lower bidiagonal matrix

$$\tilde{L} = \begin{bmatrix} a_1 & & & & 0 \\ b_1 & a_2 & & & \\ & b_2 & a_3 & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ 0 & & & b_{n-1} & a_n \end{bmatrix}, \tag{4.3.3}$$

and a componentwise perturbation

$$\tilde{L} + \delta\tilde{L} = \begin{bmatrix} a_1\alpha_1 & & & & 0 \\ b_1\alpha_2 & a_2\alpha_3 & & & \\ & b_2\alpha_4 & a_3\alpha_5 & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ 0 & & & b_{n-1}\alpha_{2n-2} & a_n\alpha_{2n-1} \end{bmatrix} \tag{4.3.4}$$

where $\alpha_i > 0$.

The key fact is that $\tilde{L} + \delta\tilde{L}$ may be written as

$$\begin{aligned} \tilde{L} + \delta\tilde{L} &= D_1 \tilde{L} D_2 \\ \text{where} \quad D_1 &= \operatorname{diag}\left(1, \frac{\alpha_2}{\alpha_1}, \frac{\alpha_2\alpha_4}{\alpha_1\alpha_3}, \frac{\alpha_2\alpha_4\alpha_6}{\alpha_1\alpha_3\alpha_5}, \ldots\ldots, \frac{\alpha_2\alpha_4\alpha_6\cdots\alpha_{2n-2}}{\alpha_1\alpha_3\alpha_5\cdots\alpha_{2n-3}}\right), \\ \text{and} \quad D_2 &= \operatorname{diag}\left(\alpha_1, \frac{\alpha_1\alpha_3}{\alpha_2}, \frac{\alpha_1\alpha_3\alpha_5}{\alpha_2\alpha_4}, \ldots\ldots, \frac{\alpha_1\alpha_3\alpha_5\cdots\alpha_{2n-1}}{\alpha_2\alpha_4\cdots\alpha_{2n-2}}\right). \end{aligned}$$

In [49], Eisenstat and Ipsen considered such multiplicative perturbations for singular value problems and proved the results presented below which show that if D_1 and D_2 are close to unitary matrices then the singular values of $\tilde{L} + \delta\tilde{L}$ are close in a relative measure to the corresponding singular values of $\tilde{L}$. These results are also an immediate consequence of Ostrowski's theorem in [80, Thm. 4.5.9]. In the following, σ_j and $\sigma_j + \delta\sigma_j$ denote the jth singular values of $\tilde{L}$ and $\tilde{L} + \delta\tilde{L}$ respectively, while u_j, $u_j + \delta u_j$, v_j and $v_j + \delta v_j$ denote the corresponding left and right singular vectors.

Theorem 4.3.1 (Eisenstat and Ipsen [49, Thm. 3.1]). *Let $\tilde{L} + \delta\tilde{L} = D_1^T \tilde{L} D_2$, where D_1 and D_2 are nonsingular matrices. Then*

$$\frac{\sigma_j}{\|D_1^{-1}\| \cdot \|D_2^{-1}\|} \;\le\; \sigma_j + \delta\sigma_j \;\le\; \sigma_j \|D_1\| \cdot \|D_2\|.$$

Corollary 4.3.1 (Barlow and Demmel [9, Thm. 1], Deift et al. [29, Thm. 2.12], Demmel and Kahan [35, Cor. 2], Eisenstat and Ipsen [49, Cor. 4.2]). *Let $\tilde{L}$ and $\tilde{L} + \delta\tilde{L}$ be bidiagonal matrices as in (4.3.3) and (4.3.4). Then*

$$\frac{1}{1+\eta}\sigma_j \;\le\; \sigma_j + \delta\sigma_j \;\le\; (1+\eta)\sigma_j,$$

where $\eta = \prod_{i=1}^{2n-1} \max\{|\alpha_i|, 1/|\alpha_i|\} - 1$.

Proof. The proof follows by noting that $\|D_1\| \cdot \|D_2\| \le 1 + \eta$ and $\|D_1^{-1}\| \cdot \|D_2^{-1}\| \le 1 + \eta$, and then applying Theorem 4.3.1. □

More recently, in [116] Parlett gives relative condition numbers that indicate the precise amount by which a singular value changes due to a relative perturbation in a particular element of a bidiagonal matrix.

Theorem 4.3.2 (Parlett [116, Thm. 1]) *Let $\tilde{L}$ be a bidiagonal matrix as in (4.3.3), with $a_i \ne 0$, $b_i \ne 0$. Let σ denote a particular singular value of $\tilde{L}$ and let u, v be the corresponding singular vectors. Then, since $\sigma \ne 0$,*

$$\begin{aligned}
(a) \quad \frac{\partial\sigma}{\partial a_k} \cdot \frac{a_k}{\sigma} \;&=\; \sum_{i=1}^{k} v(i)^2 - \sum_{j=1}^{k-1} u(j)^2 = \sum_{m=k}^{n} u(m)^2 - \sum_{l=k+1}^{n} v(l)^2, \\
(b) \quad \frac{\partial\sigma}{\partial b_k} \cdot \frac{b_k}{\sigma} \;&=\; \sum_{i=1}^{k} (u(i)^2 - v(i)^2) = \sum_{m=k+1}^{n} (v(m)^2 - u(m)^2).
\end{aligned}$$

Traditional error bounds on singular vector perturbations show them to be inversely proportional to the absolute gaps between the corresponding singular values. Recent work has shown that in the case of multiplicative perturbations, as in Theorem 4.3.1 above, absolute gaps may be replaced by relative gaps.

Before we quote the relevant results, we introduce some notation. We define the relative distance between two numbers α and β as

$$\operatorname{reldist}(\alpha, \beta) \;\stackrel{\text{def}}{=}\; \frac{|\alpha - \beta|}{|\alpha|}. \tag{4.3.5}$$

By the above definition, $\text{reldist}(\alpha,\beta) \neq \text{reldist}(\beta,\alpha)$. On the other hand, the measures

$$\text{reldist}_p(\alpha,\beta) \stackrel{\text{def}}{=} \frac{|\alpha-\beta|}{\sqrt[p]{|\alpha|^p+|\beta|^p}}. \tag{4.3.6}$$

are symmetric for $1 \le p \le \infty$. Note that

$$\text{reldist}_\infty(\alpha,\beta) \stackrel{\text{def}}{=} \frac{|\alpha-\beta|}{\max(|\alpha|,|\beta|)}. \tag{4.3.7}$$

For more discussion and comparison between these measures, see [100]. Relative gaps between singular values will figure prominently in our discussion, and we define

$$\text{relgap}(\nu,\{\sigma\}) \stackrel{\text{def}}{=} \min_{y\in\{\sigma\}} \text{reldist}(\nu,y), \tag{4.3.8}$$

where ν is a real number while $\{\sigma\}$ denotes a set of real numbers. Typically, $\{\sigma\}$ will denote a subset of the singular values. Similarly, we define the relative gaps for the symmetric measures to be

$$\text{relgap}_p(\nu,\{\sigma\}) \stackrel{\text{def}}{=} \min_{y\in\{\sigma\}} \text{reldist}_p(\nu,y).$$

The following theorem bounds the perturbation angle in terms of the relative gaps. It is a special case of Li's Theorem 4.7 [101].

Theorem 4.3.3 *Let $\tilde{L}+\delta\tilde{L} = D_1^T\tilde{L}D_2$, where D_1 and D_2 are nonsingular matrices. If $\sigma_i+\delta\sigma_i \neq \sigma_j$ for $i \neq j$, then*

$$|\sin\angle(u_j, u_j+\delta u_j)| \le \frac{\sqrt{\|I-D_1^T\|^2+\|I-D_1^{-1}\|+\|I-D_2^T\|^2+\|I-D_2^{-1}\|}}{\text{relgap}_2(\sigma_j,\{\sigma_i+\delta\sigma_i | i\neq j\})}.$$

Corollary 4.3.2 *Let $\tilde{L}$ and $\tilde{L}+\delta\tilde{L}$ be bidiagonal matrices as in (4.3.3) and (4.3.4). If $\sigma_i+\delta\sigma_i \neq \sigma_j$ for $i\neq j$, then*

$$|\sin\angle(u_j, u_j+\delta u_j)| \le \frac{\eta}{\text{relgap}_2(\sigma_j,\{\sigma_i+\delta\sigma_i | i\neq j\})},$$

where $\eta = \prod_{i=1}^{2n-1}\max\{|\alpha_i|, 1/|\alpha_i|\} - 1$.

See also Eisenstat and Ipsen [49, Theorem 3.3 and Cor. 4.5].

The above results can be generalized to singular subspaces of larger dimension. Before we present the results, we need to introduce additional notation. We partition the singular value decomposition of the square matrices $\tilde{L}$ and $\tilde{L}+\delta\tilde{L}$ as

$$\tilde{L} = U\Sigma V^T = (U_1, U_2)\begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix}\begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix},$$

and

$$\tilde{L} + \delta\tilde{L} \;=\; (U_1 + \delta U_1, U_2 + \delta U_2) \begin{pmatrix} \Sigma_1 + \delta\Sigma_1 & 0 \\ 0 & \Sigma_2 + \delta\Sigma_2 \end{pmatrix} \begin{pmatrix} V_1^T + \delta V_1^T \\ V_2^T + \delta V_2^T \end{pmatrix},$$

where $\Sigma_1 = \mathrm{diag}(\sigma_1, \ldots, \sigma_k)$ and $\Sigma_2 = \mathrm{diag}(\sigma_{k+1} + \delta\sigma_{k+1}, \ldots, \sigma_n + \delta\sigma_n)$ for $1 \le k < n$, and the other arrays are partitioned conformally. In such a case, we measure the angle between the eigenvector u_j and the invariant subspace $U_1 + \delta U_1$, $j \le k$, as

$$\|\sin \angle(u_j, U_1 + \delta U_1)\| \;=\; \|(U_2^T + \delta U_2^T) u_j\|.$$

For more on angles between subspaces, see [67, Section 12.4.3]. The following is an easily obtained extension of Theorem 4.8 in [101].

Theorem 4.3.4 *Let $\tilde{L} + \delta\tilde{L} = D_1^T \tilde{L} D_2$, where D_1 and D_2 are nonsingular matrices. If*

$$\max_{k < i \le n} \sigma_i + \delta\sigma_i \;<\; \min_{1 \le l \le k} \sigma_l,$$

then for $j \le k$ and $1 \le p \le \infty$,

$$\|\sin \angle(u_j, U_1 + \delta U_1)\| \;\le\; \frac{\max\left\{\sqrt{\|I - D_2^{-1}\|^2 + \|I - D_1^T\|^2}, \sqrt{\|I - D_1^{-1}\|^2 + \|I - D_2^T\|^2}\right\}}{\mathrm{relgap}_p(\sigma_j, \{\sigma_i + \delta\sigma_i | k < i \le n\})}.$$

Corollary 4.3.3 *Let $\tilde{L}$ and $\tilde{L} + \delta\tilde{L}$ be bidiagonal matrices as in (4.3.3) and (4.3.4). If*

$$\max_{k < i \le n} \sigma_i + \delta\sigma_i \;<\; \min_{1 \le l \le k} \sigma_l,$$

then for $j \le k$ and $1 \le p \le \infty$,

$$\|\sin \angle(u_j, U_1 + \delta U_1)\| \;\le\; \frac{\eta}{\mathrm{relgap}_p(\sigma_j, \{\sigma_i + \delta\sigma_i | k < i \le n\})},$$

where $\eta = \prod_{i=1}^{2n-1} \max\{|\alpha_i|, 1/|\alpha_i|\} - 1$.

See also Eisenstat and Ipsen [50, Theorem 3.1].

4.4 Using Products of Bidiagonals

We now show how to exploit the properties of a bidiagonal matrix that were outlined in the previous section. Consider the tridiagonal matrix T_1 given in Example 4.2.1. In section 4.2, we gave the inherent limitations of using the $2n-1$ diagonal and off-diagonal elements of T_1. Since T_1 is positive definite we can compute its bidiagonal Cholesky factor $\tilde{L_1}$. The singular values, σ_j, of $\tilde{L_1}$ may now be computed to high relative accuracy using either bisection or the much faster and more elegant **dqds** algorithm given in [56] (remember that in exact arithmetic the eigenvalues of T_1 are the squares of the singular values of $\tilde{L_1}$ and the eigenvectors of T_1 are the left singular vectors of $\tilde{L_1}$). Recall that the singular values of $\tilde{L_1}$ are such that

$$\sigma_1^2 = \varepsilon/2 + O(\varepsilon^2), \quad \sigma_2^2 = \varepsilon + O(\varepsilon^2), \text{ and } \quad \sigma_3^2 = 1 + O(\varepsilon).$$

As a consequence of the large relative gaps between the singular values, the singular vectors of $\tilde{L_1}$ are "well-determined" with respect to small componentwise perturbations in entries of $\tilde{L_1}$. We can now compute these singular vectors by using a method similar to Algorithm 3.2.1 of Chapter 3. In spite of being computed independently, these vectors will turn out to be numerically orthogonal!

The errors made in computing the Cholesky factor $\tilde{L_1}$ are irrelevant to the orthogonality of the computed vectors. Cholesky factorization is known to be backward stable and it can be shown that the computed $\tilde{L_1}$ is the exact Cholesky factor of $T_1 + \delta T_1$ where $\|\delta T_1\| = O(\varepsilon \|T_1\|)$. Thus small residual norms with respect to $\tilde{L_1}\tilde{L_1}^T$ translate to small residual norms with respect to T_1, i.e.,

$$\begin{aligned} \|(\tilde{L}_1\tilde{L}_1^T - \sigma_j^2 I)v\| &= O(\varepsilon\|\tilde{L}_1\tilde{L}_1^T\|) \\ \Rightarrow \quad \|(T_1 - \sigma_j^2 I)v\| &= O(\varepsilon\|T_1\|). \end{aligned}$$

The dual goals of orthogonality and small residual norms will thus be satisfied in this case (see (1.1.1) and (1.1.2)).

The tridiagonal matrix of Example 4.2.1 is positive definite. In general, the matrix T whose eigendecomposition is to be computed will be indefinite. In such a case, we modify slightly the strategy outlined in the above paragraph by shifting T to make it definite. We can apply this transformation since the eigenvectors of any matrix are shift invariant, i.e.,

$$\text{Eigenvectors of } T \quad \equiv \quad \text{Eigenvectors of } T + \mu I, \quad \text{for all } \mu.$$

Our strategy can be summarized by the following algorithm.

Algorithm 4.4.1 [Computes eigenvectors using bidiagonals (preliminary version).]

1. Find $\mu \leq \|T\|$ such that $T + \mu I$ is positive (or negative) definite.
2. Compute $T + \mu I = \tilde{L}\tilde{L}^T$.
3. Compute the singular values of $\tilde{L}$ to high relative accuracy (by bisection or the **dqds** algorithm).
4. For each computed singular value of $\tilde{L}$, compute its left singular vector by a method similar to Algorithm 3.2.1 of Chapter 3. □

We now complete Step 4 of the above algorithm that computes an individual singular vector of $\tilde{L}$. We need to implement this step with care in order to get guaranteed orthogonality, whenever possible. Note that each singular vector is computed independently of the others. Corollary 4.3.2 implies that such vectors do exist when the singular values have large relative gaps and when the effects of roundoff can be attributed to small relative changes in the entries of $\tilde{L}$.

4.4.1 qd-like Recurrences

Before we proceed to the main content of this section, we make a slight change in our representation. Instead of the Cholesky factorization $\tilde{L}\tilde{L}^T$ of $T + \mu I$, we will consider its triangular decomposition LDL^T, where L is unit lower bidiagonal of the form

$$L = \begin{bmatrix} 1 & & & & & 0 \\ l_1 & 1 & & & & \\ & l_2 & 1 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ 0 & & & & l_{n-1} & 1 \end{bmatrix}, \tag{4.4.9}$$

and

$$D = \operatorname{diag}(d_1, d_2, \ldots, d_{n-1}, d_n). \tag{4.4.10}$$

Both factorizations are obviously linked and $\tilde{L} = LD^{1/2}$. The following theorem indicates that in terms of the relative perturbation theory presented in Section 4.3, both these representations are equivalent.

Theorem 4.4.1 *Let $LDL^T = \tilde{L}\Omega\tilde{L}^T$, where L, D and $\tilde{L}$ are as in (4.4.9), (4.4.10) and (4.3.3) respectively, while Ω is a diagonal matrix with ± 1 entries on its diagonal. Let $\lambda = \mathrm{sign}(\lambda)\sigma^2$ ($\neq 0$) be a typical eigenvalue of LDL^T. Then*

$$(a) \quad \frac{\partial \lambda}{\partial d_k} \cdot \frac{d_k}{\lambda} = \frac{\partial \sigma}{\partial a_k} \cdot \frac{a_k}{\sigma} + \frac{\partial \sigma}{\partial b_k} \cdot \frac{b_k}{\sigma}, \tag{4.4.11}$$

$$(b) \quad \frac{\partial \lambda}{\partial l_k} \cdot \frac{l_k}{\lambda} = 2\frac{\partial \sigma}{\partial b_k} \cdot \frac{b_k}{\sigma}. \tag{4.4.12}$$

Proof. Let $(\Omega)_{kk} = \omega_k$. The two factorizations are related by

$$d_k = w_k a_k^2, \qquad l_k = b_k/a_k.$$

By applying the chain rule for derivatives,

$$\frac{\partial \lambda}{\partial a_k} = \frac{\partial \lambda}{\partial d_k} \cdot \frac{\partial d_k}{\partial a_k} + \frac{\partial \lambda}{\partial l_k} \cdot \frac{\partial l_k}{\partial a_k}, \tag{4.4.13}$$

and

$$\frac{\partial \lambda}{\partial b_k} = \frac{\partial \lambda}{\partial d_k} \cdot \frac{\partial d_k}{\partial b_k} + \frac{\partial \lambda}{\partial l_k} \cdot \frac{\partial l_k}{\partial b_k}. \tag{4.4.14}$$

By substituting

$$\frac{\partial \lambda}{\partial x} = 2\sigma\,\mathrm{sign}(\lambda)\frac{\partial \sigma}{\partial x}, \qquad \frac{\partial d_k}{\partial a_k} = 2\omega_k a_k,$$

$$\frac{\partial l_k}{\partial a_k} = \frac{-b_k}{a_k^2}, \qquad \text{and} \qquad \frac{\partial l_k}{\partial b_k} = \frac{1}{a_k}$$

in (4.4.13) and (4.4.14), we get

$$2\sigma\,\mathrm{sign}(\lambda)\frac{\partial \sigma}{\partial a_k} = \frac{\partial \lambda}{\partial d_k} \cdot 2\omega_k a_k - \frac{\partial \lambda}{\partial l_k} \cdot \frac{b_k}{a_k^2}, \tag{4.4.15}$$

$$\text{and} \qquad 2\sigma\,\mathrm{sign}(\lambda)\frac{\partial \sigma}{\partial b_k} = \frac{\partial \lambda}{\partial l_k} \cdot \frac{1}{a_k}. \tag{4.4.16}$$

The result (4.4.12) now follows from multiplying (4.4.16) by b_k/λ, while (4.4.11) is similarly obtained by substituting (4.4.12) in (4.4.15). □

From now on, we will deal exclusively with the LDL^T representation instead of the Cholesky factorization. This choice avoids the need to take square roots when forming the Cholesky factor.

To find an individual eigenvector by Algorithm 3.2.1, we needed to form the $L_+D_+L_+^T$ and $U_-D_-U_-^T$ decompositions of $T - \hat{\lambda}I$. Instead of T we now have the factored matrix LDL^T. Algorithm 4.4.2 listed below implements the transformation

$$LDL^T - \mu I = L_+D_+L_+^T. \tag{4.4.17}$$

We call this the "**st**ationary **q**uotient-**d**ifference with **s**hift"(stqds) transformation for historical reasons. This term was first coined by Rutishauser for similar transformations that formed the basis of his qd algorithm first developed in 1954 [121, 122, 123]. Although (4.4.17) is not identical to the stationary transformation given by Rutishauser, the differences are not significant enough to warrant inventing new terminology. More recently, Fernando and Parlett have developed another qd algorithm that gives a fast way of computing the singular values of a bidiagonal matrix to high relative accuracy [56]. The term 'stationary' is used for (4.4.17) since it represents an identity transformation when $\mu = 0$. Rutishauser used the term 'progressive' instead for the formation of $U_-D_-U_-^T$ from LDL^T.

In the rest of this chapter, we will denote $L_+(i+1,i)$ by $L_+(i)$, $U_-(i,i+1)$ by $U_-(i)$ and the ith diagonal entries of D_+ and D_- by $D_+(i)$ and $D_-(i)$ respectively.

Algorithm 4.4.2 (stqds)

$$
\begin{aligned}
& D_+(1) \;:=\; d_1 - \mu \\
& \text{for } i = 1, n-1 \\
& \quad L_+(i) \;:=\; (d_i l_i)/D_+(i) && (4.4.18) \\
& \quad D_+(i+1) \;:=\; d_i l_i^2 + d_{i+1} - L_+(i) d_i l_i - \mu && (4.4.19) \\
& \text{end for}
\end{aligned}
$$

We now see how to eliminate some of the additions and subtractions from the above algorithm. We introduce the intermediate variable

$$
\begin{aligned}
s_{i+1} &= D_+(i+1) - d_{i+1}, && (4.4.20) \\
&= d_i l_i^2 - L_+(i) d_i l_i - \mu, \quad \text{by } (4.4.19) \\
&= L_+(i) l_i (D_+(i) - d_i) - \mu, \quad \text{by } (4.4.18) \\
&= L_+(i) l_i s_i - \mu. && (4.4.21)
\end{aligned}
$$

Using this intermediate variable, we get the so-called *differential form* of the stationary qd transformation (dstqds). This term was again coined by Rutishauser in the context of similar transformations in [121, 122].

Algorithm 4.4.3 (dstqds)

$$s_1 \;:=\; -\mu$$

$$
\begin{aligned}
&\text{for } i = 1, n-1 \\
&\quad D_+(i) \ := \ s_i + d_i \\
&\quad L_+(i) \ := \ (d_i l_i)/D_+(i) \\
&\quad s_{i+1} \ := \ L_+(i) l_i s_i - \mu \\
&\text{end for} \\
&D_+(n) \ := \ s_n + d_n
\end{aligned}
$$

In the next section we will show that the above differential algorithm has some nice properties in the face of roundoff errors.

We also need to compute the transformation

$$LDL^T - \mu I \ = \ U_- D_- U_-^T.$$

which we call the "progressive **q**uotient-**d**ifference with **s**hift"(qds) transformation. The following algorithm gives an obvious way to implement this transformation.

Algorithm 4.4.4 (qds)

$$
\begin{aligned}
&U_-(n) \ := \ 0 \\
&\text{for } i = n-1, 1, -1 \\
&\quad D_-(i+1) \ := \ d_i l_i^2 + d_{i+1} - U_-(i+1) d_{i+1} l_{i+1} - \mu && (4.4.22) \\
&\quad U_-(i) \ := \ (d_i l_i)/D_-(i+1) && (4.4.23) \\
&\text{end for} \\
&D_-(1) \ := \ d_1 - U_-(1) d_1 l_1 - \mu
\end{aligned}
$$

As in the stationary transformation, we introduce the intermediate variable

$$
\begin{aligned}
p_i &= D_-(i) - d_{i-1} l_{i-1}^2, && (4.4.24) \\
&= d_i - U_-(i) d_i l_i - \mu, \quad \text{by (4.4.22)} \\
&= \frac{d_i}{D_-(i+1)} (D_-(i+1) - d_i l_i^2) - \mu, \quad \text{by (4.4.23)} \\
&= \frac{d_i}{D_-(i+1)} \cdot p_{i+1} - \mu. && (4.4.25)
\end{aligned}
$$

Using this intermediate variable, we get the *differential form* of the progressive qd transformation,

Algorithm 4.4.5 (dqds)

$$
\begin{aligned}
& p_n \ := \ d_n - \mu \\
& \text{for } i = n-1, 1, -1 \\
& \quad D_-(i+1) \ := \ d_i l_i^2 + p_{i+1} \\
& \quad t \ := \ d_i / D_-(i+1) \\
& \quad U_-(i) \ := \ l_i t \\
& \quad p_i \ := \ p_{i+1} t - \mu \\
& \text{end for} \\
& D_-(1) \ := \ p_1
\end{aligned}
$$

Note that we have denoted the intermediate variables by the symbols s_i and p_i to stand for *stationary* and *progressive* respectively.

As in Algorithm 3.2.1, we also need to find all the γ_k's in order to choose the appropriate twisted factorization for computing the eigenvector. Since $(LDL^T)_{k,k+1} = d_k l_k$, by the fourth formula for γ_k in Corollary 3.1.1, we have

$$
\begin{aligned}
\gamma_k &= D_+(k) - \frac{(d_k l_k)^2}{D_-(k+1)}, \\
&= s_k + d_k - \frac{(d_k l_k)^2}{D_-(k+1)}, \quad \text{by } (4.4.20) \\
&= s_k + \frac{d_k}{D_-(k+1)} \left(D_-(k+1) - d_k l_k^2 \right).
\end{aligned}
$$

By (4.4.24), (4.4.25) and (4.4.21), we can express γ_k as

$$
\gamma_k = \begin{cases} s_k + \frac{d_k}{D_-(k+1)} \cdot p_{k+1}, \\ s_k + p_k + \mu, \\ p_k + L_+(k-1) l_{k-1} s_{k-1}. \end{cases} \tag{4.4.26}
$$

In the next section, we will see that the top and bottom formulae in (4.4.26) are "better" for computational purposes. We can now choose r as the index where $|\gamma_k|$ is minimum. The twisted factorization at position r is given by

$$
LDL^T - \mu I = N_r D_r N_r^T,
$$

where $D_r = \text{diag}(D_+(1), \ldots, D_+(r-1), \gamma_r, D_-(r+1), \ldots, D_-(n))$ and N_r is the corresponding twisted factor (see (3.1.10)). It may be formed by the following "**d**ifferential **tw**isted **q**uotient-**d**ifference with **s**hift"(dtwqds) transformation.

Algorithm 4.4.6 (dtwqds)

$$
\begin{aligned}
& s_1 := -\mu \\
& \text{for } i = 1, r-1 \\
& \quad D_+(i) := s_i + d_i \\
& \quad L_+(i) := (d_i l_i)/D_+(i) \\
& \quad s_{i+1} := L_+(i) l_i s_i - \mu \\
& \text{end for} \\
& p_n := d_n - \mu \\
& \text{for } i = n-1, r, -1 \\
& \quad D_-(i+1) := d_i l_i^2 + p_{i+1} \\
& \quad t := d_i / D_-(i+1) \\
& \quad U_-(i) := l_i t \\
& \quad p_i := p_{i+1} t - \mu \\
& \text{end for} \\
& \gamma_r := s_r + \frac{d_r}{D_-(r+1)} \cdot p_{r+1}
\end{aligned}
$$

Note: In cases where we have already computed the stationary and progressive transformations, i.e., we have computed L_+, D_+, U_- and D_-, the only additional work needed for dtwqds is one multiplication and one addition to compute γ_r.

In the next section, we exhibit desirable properties of the differential forms of our qd-like transformations in the face of roundoff errors. Before we do so, we emphasize that *the particular qd-like transformations presented in this section are new.* Similar qd recurrences have been studied by Rutishauser [121, 122, 123], Henrici [76], Fernando and Parlett [56], Yao Yang [138] and David Day [28].

4.4.2 Roundoff Error Analysis

First, we introduce our model of arithmetic. We assume that the floating point result of a basic arithmetic operation $\circ$ satisfies

$$fl(x \circ y) = (x \circ y)(1 + \eta) = (x \circ y)/(1 + \delta)$$

where η and δ depend on x, y, $\circ$, and the arithmetic unit but satisfy

$$|\eta| < \varepsilon, \quad |\delta| < \varepsilon$$

for a given ε that depends only on the arithmetic unit. We shall choose freely the form (η or δ) that suits the analysis. As usual, we will ignore $O(\varepsilon^2)$ terms in our analyses. We also adopt the convention of denoting the computed value of x by $\hat{x}$.

Ideally, we would like to show that the differential qd transformations introduced in the previous section produce an output that is exact for data that is very close to the input matrix. Since we desire relative accuracy, we would like this backward error to be relative. However, our algorithms do not admit such a pure backward analysis (see [138, 111] for a backward analysis where the backward errors are absolute but not relative). Nevertheless, we will give a hybrid interpretation involving both backward and forward relative errors. Our error analysis is on the lines of that presented in [56].

The best way to understand our first result is by studying Figure 4.1. Following Rutishauser, we merge elements of L and D into a single array,

$$Z := \{d_1, l_1, d_2, l_2, \ldots, d_{n-1}, l_{n-1}, d_n\}.$$

Likewise, the array $\vec{Z}$ is made up of elements $\vec{d}_i$ and $\vec{l}_i$, $\hat{Z}_+$ contains elements $\hat{D}_+(i)$, $\hat{L}_+(i)$ and so on. The acronym *ulp* in Figure 4.1 stands for *u*nits in the *l*ast *p*lace held. It is the natural way to refer to *relative* differences between numbers. When a result is correctly rounded the error is not more than half an *ulp*.

In all our results of this section, numbers in the computer are represented by letters without any overbar, such as Z, or by "hatted" symbols, such as $\hat{Z}_+$. For example in Figure 4.1, Z represents the input data while $\hat{Z}_+$ represents the output data obtained by executing the dstqds algorithm in finite precision. Intermediate arrays, such as $\vec{Z}$ and $\overset{\frown}{Z}_+$, are introduced for our analysis but are typically unrepresentable in a computer's limited precision. Note that we have chosen the symbols $\rightarrow$ and $\frown$ in Figure 4.1 to indicate a process that takes rows and columns in increasing order, i.e., from "left to right" and "top to bottom". Later, in Figure 4.2 we use $\leftarrow$ and $\smile$ to indicate a "right to left" and "bottom to top" process.

Figure 4.1 states that the computed outputs of the dstqds transformation (see Algorithm 4.4.3), $\hat{D}_+(i)$ and $\hat{L}_+(i)$ are small relative perturbations of the quantities $\overset{\frown}{D}_+(i)$ and $\overset{\frown}{L}_+(i)$ which in turn are the results of an exact dstqds transformation applied to the perturbed matrix represented by $\vec{Z}$. The elements of $\vec{Z}$ are obtained by small relative changes

$$\begin{tikzcd}[column sep=huge, row sep=huge]
Z \arrow[r, "\textbf{dstqds}", "\text{computed}"'] \arrow[d, "\begin{array}{l}\text{change each}\\ d_i \text{ by } 1\ ulp,\\ l_i \text{ by } 3\ ulps.\end{array}"'] & \hat{Z}_+ \\
\vec{Z} \arrow[r, "\textbf{dstqds}", "\text{exact}"'] & \overset{\frown}{Z}_+ \arrow[u, "\begin{array}{l}\text{change each}\\ \overset{\frown}{D}_+(i) \text{ by } 2\ ulps,\\ \overset{\frown}{L}_+(i) \text{ by } 3\ ulps.\end{array}"']
\end{tikzcd}$$

Figure 4.1: Effects of roundoff — **dstqds** transformation

in the inputs L and D. Analogous results hold for the dqds and dtwqds transformation (see Algorithms 4.4.5 and 4.4.6). As we mentioned above, this is not a pure backward error analysis. We have put small perturbations not only on the input but also on the output in order to obtain an exact dstqds transform. This property is called mixed stability in [30] but note that our perturbations are relative ones.

Theorem 4.4.2 *Let the dstqds transformation be computed as in Algorithm 4.4.3. In the absence of overflow and underflow, the diagram in Figure 4.1 commutes and* $\vec{d}_i$ ($\vec{l}_i$) *differs from* d_i (l_i) *by* 1 (3) *ulps, while* $\hat{D}_+(i)$ ($\hat{L}_+(i)$) *differs from* $\overset{\frown}{D}_+(i)$ ($\overset{\frown}{L}_+(i)$) *by* 2 (3) *ulps.*

Proof. We write down the exact equations satisfied by the computed quantities.

$$\begin{aligned}
\hat{D}_+(i) &= (\hat{s}_i + d_i)/(1+\varepsilon_+), \\
\hat{L}_+(i) &= d_i\, l_i(1+\varepsilon_*)(1+\varepsilon_/)/\hat{D}_+(i) \;=\; \frac{d_i\, l_i(1+\varepsilon_*)(1+\varepsilon_/)(1+\varepsilon_+)}{\hat{s}_i + d_i}, \\
\text{and} \quad \hat{s}_{i+1} &= \frac{\hat{L}_+(i)\, l_i \hat{s}_i(1+\varepsilon_\circ)(1+\varepsilon_{**}) - \mu}{1+\varepsilon_{i+1}}.
\end{aligned}$$

In the above, all ε's depend on i but we have chosen to single out the one that accounts for the subtraction as it is the only one where the dependence on i must be made explicit. In more detail the last relation is

$$(1+\varepsilon_{i+1})\hat{s}_{i+1} = \frac{d_i\, l_i^2\, \hat{s}_i}{\hat{s}_i + d_i}(1+\varepsilon_*)(1+\varepsilon_/)(1+\varepsilon_+)(1+\varepsilon_\circ)(1+\varepsilon_{**}) - \mu.$$

The trick is to define $\vec{d}_i$ and $\vec{l}_i$ so that the exact dstqds relation

$$\vec{s}_{i+1} = \frac{\vec{d}_i \, \vec{l}_i^{\,2} \, \vec{s}_i}{\vec{s}_i + \vec{d}_i} - \mu \tag{4.4.27}$$

is satisfied. This may be achieved by setting

$$\begin{aligned} \vec{d}_i &= d_i(1+\varepsilon_i), \\ \vec{s}_i &= \hat{s}_i(1+\varepsilon_i), \\ \vec{l}_i &= l_i \sqrt{\frac{(1+\varepsilon_*)(1+\varepsilon_/)(1+\varepsilon_+)(1+\varepsilon_\circ)(1+\varepsilon_{**})}{1+\varepsilon_i}}. \end{aligned} \tag{4.4.28}$$

In order to satisfy the exact mathematical relations of dstqds,

$$\widehat{D}_+(i) = \vec{s}_i + \vec{d}_i, \tag{4.4.29}$$

$$\widehat{L}_+(i) = \frac{\vec{d}_i \, \vec{l}_i}{\vec{s}_i + \vec{d}_i}, \tag{4.4.30}$$

we set

$$\begin{aligned} \widehat{D}_+(i) &= \hat{D}_+(i)(1+\varepsilon_+)(1+\varepsilon_i), \\ \widehat{L}_+(i) &= \hat{L}_+(i)\sqrt{\frac{(1+\varepsilon_\circ)(1+\varepsilon_{**})}{(1+\varepsilon_*)(1+\varepsilon_/)(1+\varepsilon_+)(1+\varepsilon_i)}} \end{aligned} \tag{4.4.31}$$

and the result holds. □

A similar result holds for the dqds transformation.

Theorem 4.4.3 *Let the dqds transformation be computed as in Algorithm 4.4.5. In the absence of overflow and underflow, the diagram in figure 4.2 commutes and* $\overleftarrow{d}_i$ ($\overleftarrow{l}_i$) *differs from* d_i (l_i) *by* 3 (3) *ulps, while* $\hat{D}_-(i)$ ($\hat{U}_-(i)$) *differs from* $\breve{D}_-(i)$ ($\breve{U}_-(i)$) *by* 2 (4) *ulps.*

Proof. The proof is similar to that of Theorem 4.4.2. The computed quantities satisfy

$$\hat{D}_-(i+1) = (d_i\, l_i^2(1+\varepsilon_*)(1+\varepsilon_{**}) + \hat{p}_{i+1})/(1+\varepsilon_+), \tag{4.4.32}$$

$$\begin{aligned} \hat{t} &= d_i(1+\varepsilon_/)/\hat{D}_-(i+1), \\ \hat{U}_-(i) &= l_i\hat{t}(1+\varepsilon_\circ) = \frac{d_i\, l_i(1+\varepsilon_/)(1+\varepsilon_\circ)(1+\varepsilon_+)}{d_i\, l_i^2(1+\varepsilon_*)(1+\varepsilon_{**}) + \hat{p}_{i+1}}, \\ \hat{p}_i &= \frac{(d_i/\hat{D}_-(i+1))\hat{p}_{i+1}(1+\varepsilon_/)(1+\varepsilon_{\circ\circ}) - \mu}{1+\varepsilon_i}, \\ \Rightarrow \quad (1+\varepsilon_i)\hat{p}_i &= \frac{d_i\, \hat{p}_{i+1}}{d_i\, l_i^2(1+\varepsilon_*)(1+\varepsilon_{**}) + \hat{p}_{i+1}}(1+\varepsilon_/)(1+\varepsilon_{\circ\circ})(1+\varepsilon_+) - \mu. \end{aligned}$$

$$\begin{tikzcd}[column sep=huge, row sep=huge]
Z \arrow[r, "\textbf{dqds}", "\text{computed}"'] \arrow[d, "\begin{array}{l}\text{change each}\\ d_i \text{ by 3 } ulp\text{s},\\ l_i \text{ by 3 } ulp\text{s}.\end{array}"'] & \hat{Z}_- \\
\overleftarrow{Z} \arrow[r, "\textbf{dqds}", "\text{exact}"'] & \breve{Z}_- \arrow[u, "\begin{array}{l}\text{change each}\\ \breve{D}_-(i) \text{ by 2 } ulp\text{s},\\ \breve{U}_-(i) \text{ by 4 } ulp\text{s}.\end{array}"']
\end{tikzcd}$$

Figure 4.2: Effects of roundoff — **dqds** transformation

Note that the above ε's are different from the ones in the proof of the earlier Theorem 4.4.2. As in Theorem 4.4.2, the trick is to satisfy the exact relation,

$$\overleftarrow{p}_i = \frac{\overleftarrow{d}_i \, \overleftarrow{p}_{i+1}}{\overleftarrow{d}_i \, \overleftarrow{l}_i^{\,2} + \overleftarrow{p}_{i+1}} - \mu, \tag{4.4.33}$$

which is achieved by setting

$$\begin{aligned}
\overleftarrow{d}_i &= d_i(1+\varepsilon_/)(1+\varepsilon_{\circ\circ})(1+\varepsilon_+), \\
\overleftarrow{p}_i &= \hat{p}_i(1+\varepsilon_i), \qquad (4.4.34) \\
\text{and} \quad \overleftarrow{l}_i &= l_i\sqrt{\frac{(1+\varepsilon_*)(1+\varepsilon_{**})(1+\varepsilon_{i+1})}{(1+\varepsilon_/)(1+\varepsilon_{\circ\circ})(1+\varepsilon_+)}}, \qquad (4.4.35) \\
\text{so that} \quad \overleftarrow{d}_i \, \overleftarrow{l}_i^{\,2} &= d_i l_i^2 (1+\varepsilon_*)(1+\varepsilon_{**})(1+\varepsilon_{i+1}).
\end{aligned}$$

The other dqds relations,

$$\breve{D}_-(i+1) = \overleftarrow{d}_i \, \overleftarrow{l}_i^{\,2} + \overleftarrow{p}_{i+1}, \tag{4.4.36}$$

$$\breve{U}_-(i) = \frac{\overleftarrow{d}_i \, \overleftarrow{l}_i}{\overleftarrow{d}_i \, \overleftarrow{l}_i^{\,2} + \overleftarrow{p}_{i+1}}, \tag{4.4.37}$$

may be satisfied by setting

$$\begin{aligned}
\breve{D}_-(i+1) &= \hat{D}_-(i+1)(1+\varepsilon_+)(1+\varepsilon_{i+1}), \\
\breve{U}_-(i) &= \frac{\hat{U}_-(i)}{1+\varepsilon_\circ}\sqrt{\frac{(1+\varepsilon_*)(1+\varepsilon_{**})(1+\varepsilon_{\circ\circ})}{(1+\varepsilon_/)(1+\varepsilon_+)(1+\varepsilon_{i+1})}}. \qquad (4.4.38)
\end{aligned}$$

$$
\begin{tikzcd}
Z \arrow[r, "\textbf{dtwqds}", "\text{computed}"'] \arrow[d] & \hat{Z}_k \\
\bar{Z} \arrow[r, "\textbf{dtwqds}", "\text{exact}"'] & \tilde{Z}_k \arrow[u]
\end{tikzcd}
$$

change each
d_i by 1 *ulp*, $1 \le i < k$,
l_i by 3 *ulps*, $1 \le i < k$,
d_k by 4 *ulps*, l_k by $3\frac{1}{2}$ *ulps*,
d_i by 3 *ulps*, $k < i \le n$,
l_i by 3 *ulps*, $k < i < n$.

change each
$\overset{\frown}{D}_+(i)$ by 2 *ulps*, $1 \le i < k$,
$\overset{\frown}{L}_+(i)$ by 3 *ulps*, $1 \le i < k$.
$\tilde{\gamma}_k$ by 2 *ulps*, $\tilde{U}_-(k)$ by $4\frac{1}{2}$ *ulps*,
$\overset{\smile}{D}_-(i)$ by 2 *ulps*, $k < i \le n$,
$\overset{\smile}{U}_-(i)$ by 4 *ulps*, $k < i < n$.

Figure 4.3: Effects of roundoff — **dtwqds** transformation

By combining parts of the analyses for the dstqds and dqds transformations, we can also exhibit a similar result for the twisted factorization computed by Algorithm 4.4.6. In Figure 4.3, the various Z arrays represent corresponding twisted factors that may be obtained by "concatenating" the stationary and progressive factors. In particular, for any twist position k,

$$
\begin{aligned}
\hat{Z}_k &:= \{\hat{D}_+(1), \hat{L}_+(1), \ldots, \hat{L}_+(k-1), \hat{\gamma}_k, \hat{U}_-(k), \ldots, \hat{U}_-(n-1), \hat{D}_-(n)\}, \\
\tilde{Z}_k &:= \{\overset{\frown}{D}_+(1), \overset{\frown}{L}_+(1), \ldots, \overset{\frown}{L}_+(k-1), \tilde{\gamma}_k, \overset{\smile}{U}_-(k), \ldots, \overset{\smile}{U}_-(n-1), \overset{\smile}{D}_-(n)\},
\end{aligned}
$$

while

$$
\bar{Z} := \{\vec{d}_1, \vec{l}_1, \ldots, \vec{l}_{k-1}, \bar{d}_k, \overleftarrow{l}_k, \ldots, \overleftarrow{l}_{n-1}, \overleftarrow{d}_n\}.
$$

$\hat{Z}_k$ and $\tilde{Z}_k$ represent the twisted factorizations

$$
\hat{N}_k \hat{D}_k \hat{N}_k^T \qquad \text{and} \qquad \tilde{N}_k \tilde{D}_k \tilde{N}_k^T
$$

respectively (note that $\sim$ is a concatenation of the symbols $\frown$ and $\smile$, while $-$ may also be derived by concatenating $\leftarrow$ and $\rightarrow$).

Theorem 4.4.4 *Let the dtwqds transformation be computed as in Algorithm 4.4.6. In the absence of overflow and underflow, the diagram in Figure 4.3 commutes and* $\vec{d}_i$ ($\vec{l}_i$) *differs from* d_i (l_i) *by* 1 (3) *ulps for* $1 \le i < k$, $\bar{d}_k$ ($\bar{l}_k$) *differs from* d_k (l_k) *by* 4 ($3\frac{1}{2}$) *ulps, while*

$\overleftarrow{d}_i$ ($\overleftarrow{l}_i$) *differs from* d_i (l_i) *by* 3 (3) *ulps for* $k < i \leq n$. *On the output side,* $\hat{D}_+(i)$ ($\hat{L}_+(i)$) *differs from* $\overset{\frown}{D}_+(i)$ ($\overset{\frown}{L}_+(i)$) *by* 2 (3) *ulps for* $1 \leq i < k$, $\hat{\gamma}_k$ ($\hat{U}_-(k)$) *differs from* $\tilde{\gamma}_k$ ($\tilde{U}_-(k)$) *by* 2 ($4\frac{1}{2}$) *ulps, while* $\hat{D}_-(i)$ ($\hat{U}_-(i)$) *differs from* $\breve{D}_-(i)$ ($\breve{U}_-(i)$) *by* 2 (4) *ulps for* $k < i \leq n$.

Proof. The crucial observation is that for the exact stationary transformation ((4.4.27), (4.4.29) and (4.4.30)) to be satisfied for $1 \leq i \leq k-1$, roundoff errors need to be put only on $d_1, d_2, \ldots, d_{k-1}$ and $l_1, l_2, \ldots, l_{k-1}$. Similarly for the progressive transformation ((4.4.33), (4.4.36) and (4.4.37)) to hold for $k+1 \leq i \leq n$, roundoff errors need to be put only on the bottom part of the matrix, i.e., on $d_{k+1}, \ldots, d_n$ and $l_{k+1}, \ldots, l_{n-1}$. By the top formula in (4.4.26),

$$\hat{\gamma}_k = \left(\hat{s}_k + \frac{d_k}{\hat{D}_-(k+1)}\,\hat{p}_{k+1}(1+\varepsilon_/^-)(1+\varepsilon_{\circ\circ}^-)\right)\Big/(1+\varepsilon_k).$$

Note that in the above, we have put the superscript $-$ on some ε's to indicate that they are identical to the corresponding ε's in the proof of Theorem 4.4.3. By (4.4.28) and (4.4.32),

$$\begin{aligned}
(1+\varepsilon_k)\hat{\gamma}_k &= \frac{\vec{s}_k}{1+\varepsilon_k^+} + \frac{\hat{p}_{k+1}\cdot d_k(1+\varepsilon_/^-)(1+\varepsilon_{\circ\circ}^-)(1+\varepsilon_+^-)}{d_k\,l_k^2(1+\varepsilon_*^-)(1+\varepsilon_{**}^-)+\hat{p}_{k+1}},\\
\Rightarrow\quad (1+\varepsilon_k)(1+\varepsilon_k^+)\hat{\gamma}_k &= \vec{s}_k + \frac{\hat{p}_{k+1}(1+\varepsilon_{k+1}^-)\cdot d_k(1+\varepsilon_/^-)(1+\varepsilon_{\circ\circ}^-)(1+\varepsilon_+^-)(1+\varepsilon_k^+)}{d_k\,l_k^2(1+\varepsilon_*^-)(1+\varepsilon_{**}^-)(1+\varepsilon_{k+1}^-)+\hat{p}_{k+1}(1+\varepsilon_{k+1}^-)}.
\end{aligned}$$

Note that we are free to attribute roundoff errors to d_k and l_k in order to preserve exact mathematical relations at the twist position k. In particular, by setting

$$\begin{aligned}
\tilde{\gamma}_k &= \hat{\gamma}_k(1+\varepsilon_k)(1+\varepsilon_k^+),\\
\bar{d}_k &= d_k(1+\varepsilon_/^-)(1+\varepsilon_{\circ\circ}^-)(1+\varepsilon_+^-)(1+\varepsilon_k^+),\\
\bar{l}_k &= l_k\sqrt{\frac{(1+\varepsilon_*^-)(1+\varepsilon_{**}^-)(1+\varepsilon_{k+1}^-)}{(1+\varepsilon_/^-)(1+\varepsilon_{\circ\circ}^-)(1+\varepsilon_+^-)(1+\varepsilon_k^+)}},
\end{aligned}$$

and recalling that $\overleftarrow{p}_k = \hat{p}_k(1+\varepsilon_k^-)$ (see (4.4.34)), the following exact relation holds,

$$\tilde{\gamma}_k = \vec{s}_k + \frac{\bar{d}_k\,\overleftarrow{p}_{k+1}}{\bar{d}_k\,\bar{l}_k^2 + \overleftarrow{p}_{k+1}}.$$

In addition, the exact relation

$$\tilde{U}_-(k) = \frac{\bar{d}_k\,\bar{l}_k}{\bar{d}_k\,\bar{l}_k^2 + \overleftarrow{p}_{k+1}}$$

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
T = (a_k, b_k) \arrow[r, "LDL^T \text{ decomposition}", "\text{computed}"'] \arrow[d, "\begin{array}{l}\text{change each}\\ b_k \text{ by } 2\frac{1}{2}\ ulps.\end{array}"'] & \hat{Z} \\
\bar{T} = (a_k, \bar{b}_k) \arrow[r, "LDL^T \text{ decomposition}", "\text{exact}"'] & \tilde{Z} \arrow[u, "\begin{array}{l}\text{change each}\\ d_k \text{ by } 2\ ulps,\\ l_k \text{ by } 2\frac{1}{2}\ ulps.\end{array}"']
\end{tikzcd}
$$

Figure 4.4: Effects of roundoff — LDL^T decomposition

holds if we set

$$\tilde{U}_-(k) \;=\; \frac{\hat{U}_-(k)}{1+\varepsilon_\circ^-}\sqrt{\frac{(1+\varepsilon_*^-)(1+\varepsilon_{**}^-)(1+\varepsilon_{\circ\circ}^-)(1+\varepsilon_k^+)}{(1+\varepsilon_/^-)(1+\varepsilon_{k+1}^-)(1+\varepsilon_+^-)}}, \tag{4.4.39}$$

where $\varepsilon_\circ^-$ is identical to the $\varepsilon_\circ$ of (4.4.38). □

Note: A similar result may be obtained if γ_k is computed by the last formula in (4.4.26).

Before we proceed to the next section, we give an algorithm and error analysis for the initial decomposition

$$T + \mu I \;=\; LDL^T.$$

We denote the diagonal elements of T by a_i and off-diagonal elements by b_i.

Algorithm 4.4.7 [Computes the initial LDL^T decomposition.]

$$
\begin{array}{l}
d_1 \;:=\; a_1 + \mu \\
\text{for } i = 1, n-1 \\
\quad l_i \;:=\; b_i / d_i \\
\quad d_{i+1} \;:=\; (a_{i+1} + \mu) - l_i b_i \\
\text{end for}
\end{array}
$$

Theorem 4.4.5 *Let the LDL^T decomposition be computed as in Algorithm 4.4.7. In the absence of overflow and underflow, the diagram in Figure 4.4 commutes and $\bar{b}_i$ differs from b_i by $2\frac{1}{2}$ ulps, while $\hat{d}_i$ ($\hat{l}_i$) differs from $\tilde{d}_i$ ($\tilde{l}_i$) by 2 ($2\frac{1}{2}$) ulps.*

Proof. The computed quantities satisfy

$$\begin{aligned}
\hat{l}_i &= \frac{b_i}{\hat{d}_i} \cdot (1+\varepsilon_/),\\
\hat{d}_{i+1} &= \frac{(a_{i+1}+\mu)/(1+\varepsilon^+_{i+1}) - b_i\hat{l}_i(1+\varepsilon_*)}{1+\varepsilon^-_{i+1}},\\
\Rightarrow \quad (1+\varepsilon^+_{i+1})(1+\varepsilon^-_{i+1})\hat{d}_{i+1} &= (a_{i+1}+\mu) - \frac{b_i^2}{\hat{d}_i}(1+\varepsilon_/)(1+\varepsilon_*)(1+\varepsilon^+_{i+1}).
\end{aligned}$$

By setting

$$\begin{aligned}
\tilde{d}_{i+1} &= \hat{d}_{i+1}(1+\varepsilon^+_{i+1})(1+\varepsilon^-_{i+1}), \quad \tilde{d}_1 = \hat{d}_1(1+\varepsilon^+_1),\\
\tilde{l}_i &= \hat{l}_i\sqrt{\frac{(1+\varepsilon_*)(1+\varepsilon^+_{i+1})}{(1+\varepsilon_/)(1+\varepsilon^+_i)(1+\varepsilon^-_i)}},\\
\bar{b}_i &= b_i\sqrt{(1+\varepsilon_/)(1+\varepsilon_*)(1+\varepsilon^+_{i+1})(1+\varepsilon^+_i)(1+\varepsilon^-_i)},
\end{aligned} \tag{4.4.40}$$

the following exact relations hold

$$\begin{aligned}
\tilde{l}_i &= \frac{\bar{b}_i}{\tilde{d}_i},\\
\tilde{d}_{i+1} &= a_{i+1} + \mu - \frac{\bar{b}_i^2}{\tilde{d}_i}.
\end{aligned}$$

□

We can obtain a purely backward error analysis in the above case by showing that the LDL^T decomposition computed by Algorithm 4.4.7 is exact for $T+\delta T$, where δT represents an absolute perturbation in the nonzero elements of T.

Theorem 4.4.6 *In the absence of overflow and underflow, the $\hat{L}\hat{D}\hat{L}^T$ decomposition computed by Algorithm 4.4.7 is exact for a slightly perturbed matrix $T+\delta T$, i.e.,*

$$T + \delta T + \mu I = \hat{L}\hat{D}\hat{L}^T,$$

where

$$|\delta b_i| = |\eta_{i_1}\hat{d}_i\hat{l}_i + \eta_{i_2} b_i|, \quad \eta_{i_1}, \eta_{i_2} < 2.5\varepsilon, \tag{4.4.41}$$

$$|\delta a_{i+1}| = |\eta_{i_3}\hat{d}_i\hat{l}_i^2 + \eta_{i_4}\hat{d}_{i+1}|, \quad \eta_{i_3} < 3\varepsilon, \eta_{i_4} < 2\varepsilon. \tag{4.4.42}$$

Proof. From Theorem 4.4.5, the following relation holds

$$\bar{T} + \mu I \;=\; \tilde{L}\tilde{D}\tilde{L}^T, \tag{4.4.43}$$

where $\bar{T}$, $\tilde{L}$ and $\tilde{D}$ are as in Figure 4.4 and Theorem 4.4.5. Equating the diagonal and off-diagonal elements of (4.4.43), we get

$$\bar{b}_i \;=\; \tilde{d}_i\tilde{l}_i, \tag{4.4.44}$$

$$a_{i+1} + \mu \;=\; \tilde{d}_i\tilde{l}_i^2 + \tilde{d}_{i+1}. \tag{4.4.45}$$

By Theorem 4.4.5,

$$\begin{aligned}
\tilde{d}_i\tilde{l}_i &= \hat{d}_i\hat{l}_i\sqrt{\frac{(1+\varepsilon_i^+)(1+\varepsilon_i^-)(1+\varepsilon_*)(1+\varepsilon_{i+1}^+)}{1+\varepsilon_/}} \;=\; \hat{d}_i\hat{l}_i(1+\eta_{i_1}),\\
\tilde{d}_i\tilde{l}_i^2 &= \hat{d}_i\hat{l}_i^2\frac{(1+\varepsilon_*)(1+\varepsilon_{i+1}^+)}{1+\varepsilon_/} \;=\; \hat{d}_i\hat{l}_i^2(1+\eta_{i_3}),\\
\tilde{d}_{i+1} &= \hat{d}_{i+1}(1+\varepsilon_{i+1}^+)(1+\varepsilon_{i+1}^-) \;=\; \hat{d}_{i+1}(1+\eta_{i_4}),
\end{aligned}$$

where $\eta_{i_1} < 2.5\varepsilon$, $\eta_{i_3} < 3\varepsilon$ and $\eta_{i_4} < 2\varepsilon$. Substituting the above in (4.4.44) and (4.4.45) we get

$$\begin{aligned}
b_i + (\bar{b}_i - b_i - \eta_{i_1}\hat{d}_i\hat{l}_i) &= \hat{d}_i\hat{l}_i,\\
a_{i+1} - (\eta_{i_3}\hat{d}_i\hat{l}_i^2 + \eta_{i_4}\hat{d}_{i+1}) + \mu &= \hat{d}_i\hat{l}_i^2 + \hat{d}_{i+1}.
\end{aligned}$$

The result now follows by recalling the relation between b_i and $\bar{b}_i$ given in (4.4.40). $\square$

The backward error given above is small when there is no "element growth" in the LDL^T decomposition. The following lemma proves the well known fact that no element growth is obtained when factoring a positive definite tridiagonal matrix.

Lemma 4.4.1 *Suppose $\tilde{T}$ is a positive definite tridiagonal matrix. Its $\tilde{L}\tilde{D}\tilde{L}^T$ factorization, i.e.,*

$$\tilde{T} \;=\; \tilde{L}\tilde{D}\tilde{L}^T$$

satisfies

$$0 \;<\; \tilde{d}_i \;\leq\; \tilde{T}(i,i) \;\leq\; \|\tilde{T}\|, \quad \text{for} \quad i = 1, 2, \ldots, n$$

$$0 \;\leq\; \tilde{d}_{i-1}\tilde{l}_{i-1}^2 \;\leq\; \tilde{T}(i,i) \;\leq\; \|\tilde{T}\|, \quad \text{for} \quad i = 1, \ldots, n-1$$

$$\text{and} \quad \tilde{d}_i\tilde{l}_i \;=\; \tilde{T}(i+1,i) \quad\Rightarrow\quad |\tilde{d}_i\tilde{l}_i| \;\leq\; \|\tilde{T}\|, \quad \text{for} \quad i = 1, \ldots, n-1.$$

Proof. The proof is easily obtained by noting that

$$\begin{aligned} \tilde{d}_{i-1}\tilde{l}_{i-1}^2 + \tilde{d}_i &= \tilde{T}(i,i), \\ \tilde{d}_i\tilde{l}_i &= \tilde{T}(i+1,i) \end{aligned}$$

and using properties of a positive definite matrix by which the diagonal elements of $\tilde{T}$ and $\tilde{D}$ must be positive. □

Note that in the above lemma, we do not claim that the elements of $\tilde{L}$ are bounded by $\|\tilde{T}\|$. Indeed, the elements of $\tilde{L}$ can be arbitrarily large as seen from the following example,

$$\begin{bmatrix} \varepsilon^{2p} & \varepsilon^p \\ \varepsilon^p & 1+\varepsilon \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \varepsilon^{-p} & 1 \end{bmatrix} \begin{bmatrix} \varepsilon^{2p} & 0 \\ 0 & \varepsilon \end{bmatrix} \begin{bmatrix} 1 & \varepsilon^{-p} \\ 0 & 1 \end{bmatrix}.$$

Corollary 4.4.1 *Suppose $T + \mu I$ is a positive definite tridiagonal matrix. In the absence of overflow and underflow, Algorithm 4.4.7 computes $\hat{L}$ and $\hat{D}$ such that*

$$T + \delta T + \mu I = \hat{L}\hat{D}\hat{L}^T,$$

where

$$\begin{aligned} |\delta b_i| &< 5\varepsilon |b_i|, \\ |\delta a_{i+1}| &< 3\varepsilon |a_{i+1} + \mu|, \end{aligned}$$

and since $|\mu| \le \|T\|$,

$$\|\delta T\|_1 < 5\varepsilon \|T\|_1 + 3\varepsilon|\mu| < 8\varepsilon \|T\|_1.$$

Proof. The result follows by recalling that $\tilde{d}_i$ and $\tilde{l}_i$ are obtained by small relative changes in $\hat{d}_i$ and $\hat{l}_i$ respectively, and then substituting the inequalities of Lemma 4.4.1 in (4.4.41) and (4.4.42). □

Note: The backward error on T is relative if $\mu = 0$.

4.4.3 Algorithm X — orthogonality for large relative gaps

In this section, we complete the preliminary version of the method outlined in Algorithm 4.4.1. It is based on the differential qd-like transformations of Section 4.4.1. The following algorithm computes eigenvectors that are numerically orthogonal whenever the relative gaps between the eigenvalues of LDL^T are large, i.e., $O(1)$.

Algorithm X [Computes eigenvectors using bidiagonals.]

1. Find $\mu \le \|T\|$ such that $T + \mu I$ is positive (or negative) definite.

2. Compute $T + \mu I = LDL^T$.

3. Compute the eigenvalues, $\hat{\sigma}_j^2$, of LDL^T to high relative accuracy (by bisection or the **dqds** algorithm [56]).

4. For each computed eigenvalue, $\hat{\lambda} = \hat{\sigma}_j^2$, do the following

   (a) Compute $LDL^T - \hat{\lambda} I = L_+ D_+ L_+^T$ by the **dstqds** transform (Algorithm 4.4.3).

   (b) Compute $LDL^T - \hat{\lambda} I = U_- D_- U_-^T$ by the **dqds** transform (Algorithm 4.4.5).

   (c) Compute γ_k by the top formula of (4.4.26). Pick r such that $|\gamma_r| = \min_k |\gamma_k|$.

   (d) Form the approximate eigenvector $z_j = z_j^{(r)}$ by solving $\hat{N}_r \hat{D}_r \hat{N}_r^T z_j = \hat{\gamma}_r e_r$ (see Theorem 3.2.2):

$$\begin{aligned} z_j(r) &= 1, \\ z_j(i) &= -\hat{L}_+(i) \cdot z_j(i+1), \quad i = r-1, \ldots, 1, \\ z_j(l+1) &= -\hat{U}_-(l) \cdot z_j(l), \qquad l = r, \ldots, n-1. \end{aligned} \tag{4.4.46}$$

   (e) If needed, compute znrm $= \|z_j\|$ and set $\hat{v}_j = z_j/\text{znrm}$.

□

We will refer to the above method as Algorithm X in anticipation of Algorithm Y which will handle the case of small relative gaps.

4.5 Proof of Orthogonality

In this section, we prove that the pairs $(\hat{\sigma}_j^2, \hat{v}_j)$ computed by Algorithm X satisfy

$$\|(T - \hat{\sigma}_j^2 I)\hat{v}_j\| = O(n\varepsilon\|T\|), \tag{4.5.47}$$

$$|\hat{v}_j^T \hat{v}_m| = \min_{k=j}^{m-1} \left\{ \frac{O(n\varepsilon)}{\text{reldist}_2(\sigma_j, \sigma_{k+1})} + \frac{O(n\varepsilon)}{\text{reldist}_2(\sigma_k, \sigma_m)} \right\}, \quad j < m, \tag{4.5.48}$$

where reldist_2 is the relative distance as defined in Section 4.3 and ε is the machine precision. In particular, the neighboring vectors computed by Algorithm X are such that

$$|\hat{v}_j^T \hat{v}_{j+1}| = \frac{O(n\varepsilon)}{\text{reldist}_2(\sigma_j, \sigma_{j+1})}. \tag{4.5.49}$$

The O in the above bounds will be replaced by the appropriate expressions in the formal treatment given in Section 4.5.3. Here, and for the rest of this chapter, we assume that the singular values are arranged in decreasing order, i.e., $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.

As a special case, we provide a rigorous proof that Algorithm X computes numerically orthogonal eigenvectors whenever the eigenvalues of LDL^T have large relative gaps. The key to our success is that we exploit the relative perturbation properties of bidiagonal matrices given in Section 4.3 by using carefully chosen inner loops in all our computations.

The bounds of (4.5.48) and (4.5.49) are meaningful only when the relative distances are not too small. In this section, we are only interested in the order of magnitudes of these relative distances and not in their exact values. Thus in our discussions, quantities such as

$$\mathrm{relgap}_p(\sigma_j, \sigma_i) \qquad \text{and} \qquad \mathrm{reldist}_p(\hat{\sigma}_j, \sigma_i),$$

where σ_j and $\hat{\sigma}_j$ agree in almost all their digits, are to be treated equivalently. We will, of course, be precise in the formal statement of our theorems and their proofs.

4.5.1 A Requirement on r

As explained in Section 3.2, we require that the choice of the index r in Step (4c) of Algorithm X be such that the residual norm of the computed eigenpair, $|\gamma_r|/\|z_j\|$, is "small". For the rest of this thesis we assume that our new algorithms always choose such a "good" r. We now explain why we can ensure such a choice of r.

1. Theorem 3.2.3 proves that there exists a "good" choice of r, i.e., $\exists r$ such that
$$\frac{|\gamma_r|}{\|z_j\|} \leq \sqrt{n}|\hat{\sigma}_j^2 - \sigma_j^2|, \tag{4.5.50}$$
where z_j is the vector computed in Step (4d) of Algorithm X. Note that in all our methods to compute eigenvectors, $\hat{\sigma}_j$ will be a very good approximation to σ_j, i.e.,
$$\frac{|\hat{\sigma}_j - \sigma_j|}{|\sigma_j|} = O(\varepsilon), \tag{4.5.51}$$
and so the residual norm given by (4.5.50) will be small.

2. Theorem 3.2.1 showed that a "good" choice of r is often revealed by a small $|\gamma_r|$ (this fact is utilized in Step (4c) of Algorithm X). The following mild assumption on the approximations $\hat{\sigma}_j^2$ and the separation of the eigenvalues σ_j^2,
$$\frac{|\sigma_j^2 - \hat{\sigma}_j^2|}{\mathrm{gap}(\hat{\sigma}_j^2, \{\sigma_i^2 | i \neq j\})} \leq \frac{1}{2(n-1)}, \tag{4.5.52}$$

where $\text{gap}(\hat{\sigma}_j^2, \{\sigma_i^2 | i \neq j\}) = \min_{i \neq j} |\hat{\sigma}_j^2 - \sigma_i^2|$, ensures that

$$|\gamma_r| \leq 2n \cdot |\hat{\sigma}_j^2 - \sigma_j^2| \tag{4.5.53}$$

(note that we have obtained the above bound by choosing M in Theorem 3.2.1 to equal 2). Note that $\|z_j\| \geq 1$ and so a small value of $|\gamma_r|$ implies a small residual norm. Recall that $\hat{\sigma}_j$ will satisfy (4.5.51) and so the eigenvalues have to be very close together to violate (4.5.52). When the latter happens, there is a theoretical danger that no γ_k will be small and we shortly see how to handle such a situation. However, in all our extensive random numerical testing, we have never come across an example where small gaps cause all γ's to be large.

3. We can make a "good" choice of r even in the situation described above by using twisted Q factorizations that were discussed in Section 3.5. Theorems 3.5.3 and 3.5.4 indicate how to use these factorizations to choose an r that satisfies (4.5.50). Of course, Step (4c) of Algorithm X needs to be modified when γ_r is not small enough. For an alternate approach to compute a good r that does not involve orthogonal factorizations the reader is referred to [42].

In summary, we have seen how to ensure that r is "good" and either (4.5.50) or (4.5.53) is satisfied.

Of course, we look for a small residual norm since it implies that the computed vector is close to the exact eigenvector. The following $\sin\theta$ theorem, see [110, Chapter 11], is well known and shows that a small residual norm implies a good eigenvector if the corresponding eigenvalue is isolated. The theorem, which we will often refer to in the next few sections, is valid for all Hermitian matrices.

Theorem 4.5.1 *Let $A = A^*$ have an isolated eigenvalue λ with normalized eigenvector v. Consider y, $y^*y = 1$, and real μ closer to λ than to any other eigenvalue. Then*

$$|\sin \angle(v, y)| \leq \frac{\|Ay - y\mu\|_2}{gap(\mu)},$$

where $gap(\mu) = \min\{|\nu - \mu| : \nu \neq \lambda, \ \nu \in \ spectrum\ (A)\}$.

The extension of this theorem to higher-dimensional subspaces is due to Davis and Kahan, see [26] and [27] for more details.

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
(u_j)\ Z \arrow[r, "\textbf{dtwqds}", "\text{computed}"'] \arrow[d, "\begin{array}{l}\text{change each}\\ d_i \text{ by } 1\ ulp,\ 1 \le i < k,\\ l_i \text{ by } 3\ ulps,\ 1 \le i < k,\\ d_k \text{ by } 4\ ulps,\ l_k \text{ by } 3\frac{1}{2}\ ulps,\\ d_i \text{ by } 3\ ulps,\ k < i \le n,\\ l_i \text{ by } 3\ ulps,\ k < i < n.\end{array}"'] & \hat{Z}_k\ (\hat{v}_j) \\
(\bar{u}_j)\ \bar{Z} \arrow[r, "\textbf{dtwqds}", "\text{exact}"'] & \tilde{Z}_k\ (\tilde{v}_j) \arrow[u, "\begin{array}{l}\text{change each}\\ \widehat{D}_+(i) \text{ by } 2\ ulps,\ 1 \le i < k,\\ \widehat{L}_+(i) \text{ by } 3\ ulps,\ 1 \le i < k.\\ \tilde{\gamma}_k \text{ by } 2\ ulps,\ \tilde{U}_-(k) \text{ by } 4\frac{1}{2}\ ulps,\\ \breve{D}_-(i) \text{ by } 2\ ulps,\ k < i \le n,\\ \breve{U}_-(i) \text{ by } 4\ ulps,\ k < i < n.\end{array}"']
\end{tikzcd}
$$

Figure 4.5: **dtwqds** transformation applied to compute an eigenvector

4.5.2 Outline of Argument

We alert the reader to the fact that the analysis to follow involves close but different quantities such as $\hat{v}$, $\tilde{v}$, L, $\bar{L}$, $\tilde{L}$, etc. So watch the overbars carefully. Recall that quantities with a $\sim$ or $-$ on top are ideal whereas others like d_i and $\hat{D}_+(i)$ are stored in the computer.

We repeat the commutative diagram for the dtwqds transformation in Figure 4.5 for easy reference. We will relate the computed vector $\hat{v}_j$ to an eigenvector u_j of LDL^T by first relating it to the intermediate vectors $\tilde{v}_j$ and $\bar{u}_j$. We have associated these vectors with the various Z arrays in Figure 4.5, and the reader may find it helpful to refer to this figure during our upcoming exposition. Before we give the formal proofs, we sketch a detailed outline.

1. The vector computed in Step (4d) of Algorithm X is formed only by multiplications. As a result, the computed vector $\hat{v}_j$ is a small componentwise perturbation of the vector $\tilde{v}_j$ which is the exact solution to
$$(\bar{L}\bar{D}\bar{L}^T - \hat{\sigma}_j^2 I)\tilde{v}_j \;=\; \frac{\tilde{\gamma}_r}{\|\tilde{z}_j\|}e_r,$$
i.e.,
$$\frac{|\hat{v}_j(i) - \tilde{v}_j(i)|}{|\tilde{v}_j(i)|} \;=\; O(n\varepsilon) \quad \text{for} \quad i = 1, 2, \ldots, n, \tag{4.5.54}$$
and $\bar{L}$, $\bar{D}$ are the matrices represented by $\bar{Z}$ in Figure 4.5. For a formal proof, see Theorem 4.5.2 and Corollary 4.5.3 below.

2. To relate $\tilde{v}_j$ to an eigenvector $\bar{u}_j$ of $\bar{L}\bar{D}\bar{L}^T$, we invoke Theorem 4.5.1 to show that
$$|\sin\angle(\bar{u}_j, \tilde{v}_j)| \;\le\; \frac{|\tilde{\gamma}_r|}{\|\tilde{z}_j\| \cdot \mathrm{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_i^2 | i \neq j\})}, \tag{4.5.55}$$
where $\mathrm{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_i^2 | i \neq j\}) = \min_{i\neq j} |\hat{\sigma}_j^2 - \bar{\sigma}_i^2|$, $\bar{\sigma}_i^2$ being the ith eigenvalue of $\bar{L}\bar{D}\bar{L}^T$. Section 4.5.1 explained how we can always ensure that the residual norm is small, i.e., $|\tilde{\gamma}_r|/\|\tilde{z}_j\| = O(n\varepsilon\hat{\sigma}_j^2)$. By substituting this value in (4.5.55), the absolute gap turns into a relative gap and we get
$$|\sin\angle(\bar{u}_j, \tilde{v}_j)| \;=\; \frac{O(n\varepsilon)}{\mathrm{relgap}(\hat{\sigma}_j, \{\bar{\sigma}_i | i \neq j\})}. \tag{4.5.56}$$

3. Next we relate $\bar{u}_j$ to an eigenvector u_j of LDL^T. $\bar{L}$ and $\bar{D}$ are small componentwise perturbations of L and D as shown by our roundoff error analysis of the dtwqds transformation in Theorem 4.4.4. By the properties of a perturbed bidiagonal matrix $\bar{u}_j$ can be related to u_j (see Corollary 4.3.2), i.e.,
$$|\sin\angle(\bar{u}_j, u_j)| \;=\; \frac{O(n\varepsilon)}{\mathrm{relgap}_2(\sigma_j, \{\bar{\sigma}_i | i \neq j\})}. \tag{4.5.57}$$
The reader should note that the matrices $\tilde{L}$, $\tilde{D}$ and $\bar{L}$, $\bar{D}$ *depend on* $\hat{\sigma}_j^2$ whereas LDL^T is the *fixed* representation. By (4.5.54), (4.5.56) and (4.5.57), we have related vectors computed by Algorithm X to eigenvectors of LDL^T,
$$|\sin\angle(\hat{v}_j, u_j)| \;=\; \frac{O(n\varepsilon)}{\mathrm{relgap}_2(\sigma_j, \{\sigma_i | i \neq j\})}.$$
Theorem 4.5.3 below contains the details.

Similarly, using the Davis-Kahan SinΘ theorem [26, 27] and the subspace theorems given in Section 4.3, it can be shown that
$$|\sin\angle(\hat{v}_j, U^{1:j})| \;=\; \frac{O(n\varepsilon)}{\mathrm{reldist}_2(\sigma_j, \sigma_{j+1})}, \tag{4.5.58}$$
$$\text{and} \quad |\sin\angle(\hat{v}_j, U^{j:n})| \;=\; \frac{O(n\varepsilon)}{\mathrm{reldist}_2(\sigma_j, \sigma_{j-1})}, \tag{4.5.59}$$
where $U^{1:j}$ and $U^{j:n}$ denote the invariant subspaces spanned by $u_1, u_2, \ldots, u_j$ and $u_j, \ldots, u_n$ respectively. See Theorems 4.5.4 and 4.5.5 for the details.

4. The dot product between the neighboring vectors $\hat{v}_j$ and $\hat{v}_{j+1}$ can now be bounded since
$$\begin{aligned} |\cos\angle(\hat{v}_j, \hat{v}_{j+1})| &\le \left|\cos\angle\left\{\frac{\pi}{2} - \left(\angle(\hat{v}_j, U^{1:j}) + \angle(\hat{v}_{j+1}, U^{j+1:n})\right)\right\}\right|, \\ \Rightarrow \quad |\hat{v}_j^T \hat{v}_{j+1}| &\le |\sin\angle(\hat{v}_j, U^{1:j})| + |\sin\angle(\hat{v}_{j+1}, U^{j+1:n})|. \end{aligned}$$

By (4.5.58) and (4.5.59), we get

$$|\hat{v}_j^T \hat{v}_{j+1}| \;=\; \frac{O(n\varepsilon)}{\mathrm{reldist}_2(\sigma_j, \sigma_{j+1})}.$$

For details, see Corollary 4.5.2 below. The result (4.5.48) can be shown in the same way. Hence if all the relative gaps between eigenvalues of LDL^T are large, the vectors computed by Algorithm X are numerically orthogonal.

5. The final step in the proof is to show that the residual norms with respect to the input tridiagonal matrix T are small, i.e., (4.5.47) is satisfied. Since we can always ensure a small value of $|\gamma_r|/\|z_j\|$, we can show we *always* compute eigenpairs with small residual norms, irrespective of the relative gaps. See Theorem 4.5.6 for details.

4.5.3 Formal Proof

Now, the formal analysis begins. We start by showing that the computed vector is very close to an ideal vector.

Theorem 4.5.2 *Let $\hat{N}_r$ and $\hat{D}_r$, $\tilde{N}_r$ and $\tilde{D}_r$ be the twisted factors represented by $\hat{Z}_r$ and $\tilde{Z}_r$ respectively in Figure 4.5 (see Theorem 4.4.4 also). Let $\hat{z}_j$ be the value of z_j computed in Step (4d) of Algorithm X, and let $\tilde{z}_j$ be the exact solution of*

$$\tilde{N}_r \tilde{D}_r \tilde{N}_r^T \tilde{z}_j = \tilde{\gamma}_r e_r.$$

Then $\hat{z}_j$ is a small relative perturbation of $\tilde{z}_j$. More specifically,

$$\begin{aligned} \hat{z}_j(r) &= \tilde{z}_j(r) \;=\; 1, \\ \hat{z}_j(i) &= \tilde{z}_j(i) \cdot (1+\eta_i), i \neq r, \end{aligned} \tag{4.5.60}$$

where

$$|\eta_i| \;\leq\; \begin{cases} 4(r-i)\varepsilon, & 1 \leq i < r, \\ 5(i-r)\varepsilon, & r < i \leq n. \end{cases} \tag{4.5.61}$$

Proof. Accounting for the rounding error in (4.4.46) of Algorithm X, we get

$$\hat{z}_j(i) \;=\; -\hat{L}_+(i) \cdot \hat{z}_j(i+1) \cdot (1+\varepsilon_*^i).$$

Now replace $\hat{L}_+$ by $\widehat{L}_+$ using (4.4.31) in Theorem 4.4.2 and set $i = r-1$ to get

$$\hat{z}_j(r-1) \;=\; -\;\widehat{L}_+\,(r-1) \cdot (1+\eta_{r-1}^+) \hat{z}_j(r) \cdot (1+\varepsilon_*^{r-1}) \;=\; \tilde{z}_j(r-1) \cdot (1+\eta_{r-1}^+)(1+\varepsilon_*^{r-1}),$$

where $\eta_{r-1}^{+} < 3\varepsilon$. Thus (4.5.60) holds for $i = r-1$ (note that $\varepsilon_{*}^{r-1} = 0$ since $\hat{z}_j(r) = 1$), and similarly for $i < r-1$. Rounding errors can analogously be attributed to the lower half of $\hat{z}$ by

$$\hat{z}_j(l+1) \;=\; -\hat{U}_-(l) \cdot \hat{z}_j(l) \cdot (1 + \varepsilon_{\circ}^{l+1}).$$

Replacing $\hat{U}_-$ by $\tilde{U}_-$ using (4.4.39) and setting $l = r$, we obtain

$$\hat{z}_j(r+1) \;=\; -\tilde{U}_-(r) \cdot \hat{z}_j(r) \cdot (1 + \eta_{r+1}^{-})(1 + \varepsilon_{\circ}^{r+1}) \;=\; \tilde{z}_j(r-1) \cdot (1 + \eta_{r+1}^{-})(1 + \varepsilon_{\circ}^{r+1}),$$

where $\eta_{r+1}^{-} < 4.5\varepsilon$. Thus (4.5.60) holds for $i = r+1$ (note that $\varepsilon_{\circ}^{r+1} = 0$), and by using (4.4.38) we can similarly show that it holds for $i > r+1$. $\square$

The following theorem is at the heart of the results of this section.

Theorem 4.5.3 *Let (σ_j^2, u_j) denote the jth eigenpair of LDL^T, and let $\hat{\sigma}_j^2$ be the approximation used to compute $z_j = z_j^{(r)}$ by Step (4d) of Algorithm X, where r is an index such that*

$$\frac{|\tilde{\gamma}_r|}{\|\tilde{z}_j\|} \;\le\; \sqrt{n}\,|\hat{\sigma}_j^2 - \bar{\sigma}_j^2|. \tag{4.5.62}$$

Then

$$|\sin \angle(\hat{z}_j, u_j)| \;\le\; 5n\varepsilon + \frac{\sqrt{n}\,|\hat{\sigma}_j^2 - \bar{\sigma}_j^2|}{\mathrm{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_i^2 | i \ne j\})} + \frac{3n\varepsilon}{\mathrm{relgap}_2(\sigma_j, \{\bar{\sigma}_i | i \ne j\})},$$

where ${\bar{\sigma}_i}^2$ is the ith eigenvalue of $\bar{L}\bar{D}\bar{L}^T$ (see Figure 4.5) and differs from σ_i^2 by a few ulps.

Proof. As we outlined in Section 4.5.2, to prove this result we will relate $\hat{z}_j$ to u_j by first relating $\hat{z}_j$ to an eigenvector of the intermediate matrix $\bar{L}\bar{D}\bar{L}^T$ (see Figure 4.5). Theorem 4.5.2 links $\hat{z}_j$ to $\tilde{z}_j$ and implies that

$$|\sin \angle(\hat{z}_j, \tilde{z}_j)| \;\le\; \|\hat{z}_j - \tilde{z}_j\| \;\le\; 5n\varepsilon. \tag{4.5.63}$$

Note that $\tilde{z}_j$ is the exact solution to

$$(\bar{L}\bar{D}\bar{L}^T - \hat{\sigma}_j^2 I)\tilde{z}_j \;=\; \tilde{\gamma}_r e_r.$$

Let $(\bar{\sigma}_j^2, \bar{u}_j)$ denote the jth eigenpair of $\bar{L}\bar{D}\bar{L}^T$. By Theorem 4.5.1,

$$|\sin \angle(\tilde{z}_j, \bar{u}_j)| \;\le\; \frac{|\tilde{\gamma}_r|}{\|\tilde{z}_j\| \cdot \mathrm{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_i^2 | i \ne j\})}.$$

Since r is such that (4.5.62) is satisfied (recall that Section 4.5.1 explains why this bound can always be satisfied), we get

$$|\sin\angle(\tilde{z}_j, \bar{u}_j)| \quad \leq \quad \frac{\sqrt{n}\,|\hat{\sigma}_j^2 - \bar{\sigma}_j^2|}{\mathrm{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_i^2 | i \neq j\})}. \tag{4.5.64}$$

Since $\bar{L}$ and $\bar{D}$ are small relative perturbations of L and D, $\bar{\sigma}_i$ is a small relative perturbation of σ_i by Corollary 4.3.1. In addition,

$$|\sin\angle(\bar{u}_j, u_j)| \quad \leq \quad \frac{3n\varepsilon}{\mathrm{relgap}_2(\sigma_j, \{\bar{\sigma}_i | i \neq j\})}, \tag{4.5.65}$$

where we obtained the above bound by converting all the ulp changes in entries of L and D given in Theorem 4.4.4 to ulp changes in the off-diagonal and diagonal elements of $LD^{1/2}$, and then applying Corollary 4.3.2. The result now follows from (4.5.63), (4.5.64) and (4.5.65) since

$$|\sin\angle(\hat{z}_j, u_j)| \;\leq\; |\sin\angle(\hat{z}_j, \tilde{z}_j)| + |\sin\angle(\tilde{z}_j, \bar{u}_j)| + |\sin\angle(\bar{u}_j, u_j)|.$$

□

We can generalize the above result to bound the angle between the computed vector and the invariant subspaces of LDL^T.

Theorem 4.5.4 *Let $\hat{\sigma}_j^2$ be the approximation used to compute z_j by Step (4d) of Algorithm X, and r be such that (4.5.62) is satisfied. Let (σ_j^2, u_j) be the jth eigenpair of LDL^T and let $U^{1:k}$ denote the subspace spanned by $u_1, \ldots, u_k$. Then for $j \leq k < n$,*

$$|\sin\angle(\hat{z}_j, U^{1:k})| \quad \leq \quad 5n\varepsilon + \frac{\sqrt{n}\,|\hat{\sigma}_j^2 - \bar{\sigma}_j^2|}{\mathrm{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_{k+1}^2\})} + \frac{3n\varepsilon}{\mathrm{relgap}_2(\sigma_j, \{\bar{\sigma}_{k+1}\})},$$

where ${\bar{\sigma}_i}^2$ is the ith eigenvalue of $\bar{L}\bar{D}\bar{L}^T$ (see Figure 4.5) and differs from σ_i^2 by a few ulps.

Proof. The proof is almost identical to that of the above Theorem 4.5.3 except that instead of applying Theorem 4.5.1 and Corollary 4.3.2 to get bounds on the angles between individual vectors, we apply the Davis-Kahan SinΘ Theorem [26, 27] and Corollary 4.3.3 to get similar bounds on the angles between corresponding subspaces. □

Theorem 4.5.5 *Let $\hat{\sigma}_j^2$ be the approximation used to compute z_j by Step (4d) of Algorithm X, and r be such that (4.5.62) is satisfied. Let (σ_j^2, u_j) be the jth eigenpair of LDL^T and let $U^{k:n}$ denote the subspace spanned by $u_k, \ldots, u_n$. Then for $1 < k \leq j$,*

$$|\sin\angle(\hat{z}_j, U^{k:n})| \quad \leq \quad 5n\varepsilon + \frac{\sqrt{n}\,|\hat{\sigma}_j^2 - \bar{\sigma}_j^2|}{\mathrm{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_{k-1}^2\})} + \frac{3n\varepsilon}{\mathrm{relgap}_2(\sigma_j, \{\bar{\sigma}_{k-1}\})},$$

where ${\bar{\sigma}_i}^2$ is the ith eigenvalue of $\bar{L}\bar{D}\bar{L}^T$ (see Figure 4.5) and differs from σ_i^2 by a few ulps.

Finally we can bound the dot products between the computed eigenvectors using the above results.

Corollary 4.5.1 *Let $\hat{\sigma}_j^2$ and $\hat{\sigma}_m^2$ be the approximations used to compute z_j and z_m respectively by Step (4d) of Algorithm X. Let the twist indices r for both these computations satisfy (4.5.62). Then*

$$\begin{aligned}\frac{|\hat{z}_j^T \hat{z}_m)|}{\|\hat{z}_j\| \cdot \|\hat{z}_m\|} \;&\leq\; 10n\varepsilon + \min_{k=j,m-1}\left\{\frac{\sqrt{n}\,|\hat{\sigma}_j^2 - \bar{\sigma}_j^2|}{\text{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_{k+1}^2\})} + \frac{\sqrt{n}\,|\hat{\sigma}_m^2 - \bar{\sigma}_m^2|}{\text{gap}(\hat{\sigma}_m^2, \{\bar{\sigma}_k^2\})} + \right.\\ &\left.\frac{3n\varepsilon}{\text{relgap}_2(\sigma_j, \{\bar{\sigma}_{k+1}\})} + \frac{3n\varepsilon}{\text{relgap}_2(\sigma_m, \{\bar{\sigma}_k\})}\right\}, \quad \text{for} \quad j < m.\end{aligned}$$

where ${\bar{\sigma}_i}^2$ is the ith eigenvalue of $\bar{L}\bar{D}\bar{L}^T$ (see Figure 4.5) and differs from σ_i^2 by a few ulps.

Proof. The cosine of the angle between the computed vectors can be bounded by

$$\begin{aligned}|\cos\angle(\hat{z}_j, \hat{z}_m)| \;&\leq\; \left|\cos\angle\left\{\frac{\pi}{2} - \Big(\angle(\hat{z}_j, U^{1:k}) + \angle(\hat{z}_m, U^{k+1:n})\Big)\right\}\right|\\ &=\; |\sin\angle(\hat{z}_j, U^{1:k})| + |\sin\angle(\hat{z}_m, U^{k+1:n})|,\end{aligned}$$

where $U^{1:k}$ and $U^{k+1:n}$ are as in Theorems 4.5.4 and 4.5.5. The result now follows by applying the results of these theorems, and then choosing k to be the index where the bound is minimum. □

Corollary 4.5.2 *Let $\hat{\sigma}_j^2$ and $\hat{\sigma}_{j+1}^2$ be the approximations used to compute z_j and z_{j+1} respectively by Step (4d) of Algorithm X. Let the indices r for both these computations satisfy (4.5.62). Then*

$$\begin{aligned}\frac{|\hat{z}_j^T \hat{z}_{j+1})|}{\|\hat{z}_j\| \cdot \|\hat{z}_{j+1}\|} \;&\leq\; 10n\varepsilon + \frac{\sqrt{n}\,|\hat{\sigma}_j^2 - \bar{\sigma}_j^2|}{\text{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_{j+1}^2\})} + \frac{\sqrt{n}\,|\hat{\sigma}_{j+1}^2 - \bar{\sigma}_{j+1}^2|}{\text{gap}(\hat{\sigma}_{j+1}^2, \{\bar{\sigma}_j^2\})} + \\ &\frac{3n\varepsilon}{\text{relgap}_2(\sigma_j, \{\bar{\sigma}_{j+1}\})} + \frac{3n\varepsilon}{\text{relgap}_2(\sigma_{j+1}, \{\bar{\sigma}_j\})}.\end{aligned}$$

where ${\bar{\sigma}_i}^2$ is the ith eigenvalue of $\bar{L}\bar{D}\bar{L}^T$ (see Figure 4.5) and differs from σ_i^2 by a few ulps.

Proof. This is a special case of Corollary 4.5.1. □

Instead of assuming (4.5.62), if we assume that

$$|\tilde{\gamma}_r| \;\leq\; 2n \cdot |\hat{\sigma}_j^2 - \bar{\sigma}_j^2|, \tag{4.5.66}$$

we get a bound that is weaker by a factor of only $2\sqrt{n}$. If such an index r is chosen in Algorithm X, we can modify the middle terms in Corollaries 4.5.1 and 4.5.2 by using the above bound. See Section 4.5.1 to see how we can ensure that either one of (4.5.62) or (4.5.66) is satisfied.

Since we can compute the singular values of a bidiagonal matrix to high relative accuracy, we can find $\hat{\sigma}_j$ such that

$$|\sigma_j^2 - \hat{\sigma}_j^2| \;\le\; h(n) \cdot \varepsilon \cdot \hat{\sigma}_j^2 \tag{4.5.67}$$

where h is a slowly growing function of n.

For the sake of completeness, we prove the well known fact that normalizing $\hat{z}_j$, as in Step (4e) of Algorithm X, does not change the accuracy of the computed eigenvectors.

Corollary 4.5.3 *Let $\hat{z}_j$ and $\tilde{z}_j$ be as in Theorem 4.5.2. Let $\tilde{v}_j = \tilde{z}_j/\|\tilde{z}_j\|$, and $\hat{v}_j$ be the computed value of $\hat{z}_j/\|\hat{z}_j\|$ (see Step (4e) of Algorithm X). Then*

$$\hat{v}_j(i) \;=\; \tilde{v}_j(i) \cdot (1 + \varepsilon_i), \tag{4.5.68}$$

where $|\varepsilon_i| \le (n+2)\varepsilon + |\eta_i| + \max_i |\eta_i|$, and η_i is as in (4.5.61).

Proof. The computed value of $\|\hat{z}_j\|$ equals $\|\hat{z}_j\| \cdot (1 + \varepsilon_{\|})$ where $|\varepsilon_{\|}| < (n+1)\varepsilon$. Thus

$$\hat{v}_j(i) \;=\; \frac{\hat{z}_j(i)}{\|\hat{z}_j\|} \cdot \frac{1+\varepsilon_/}{1+\varepsilon_{\|}}. \tag{4.5.69}$$

By (4.5.60) of Theorem 4.5.2,

$$\|\hat{z}_j\| \;=\; \left(\sum_{i=1}^{n} \tilde{z}_j(i)^2 \cdot (1+\eta_i)^2\right)^{1/2} \;=\; \|\tilde{z}_j\| \left(\sum_i \frac{\tilde{z}_j(i)^2}{\|\tilde{z}_j\|^2} \cdot (1+\eta_i)^2\right)^{1/2}.$$

It follows that

$$\|\hat{z}_j\| \;=\; \|\tilde{z}_j\| \cdot (1+\eta_{\max}), \quad \text{where} \quad |\eta_{\max}| = \max_i |\eta_i|. \tag{4.5.70}$$

Substituting (4.5.60) and (4.5.70) in (4.5.69), we get

$$\hat{v}_j(i) \;=\; \frac{\tilde{z}_j(i)}{\|\tilde{z}_j\|} \cdot \frac{(1+\varepsilon_/)(1+\eta_i)}{(1+\varepsilon_{\|})(1+\eta_{\max})},$$

and the result follows. □

Thus for the eigenpairs computed by Algorithm X, the following result holds.

Corollary 4.5.4 *Let $(\hat{\sigma}_j^2, \hat{v}_j)$ be the approximate eigenpairs computed by Algorithm X. Assuming that (4.5.66) holds,*

$$\begin{aligned} |\hat{v}_j^T \hat{v}_{j+1})| \;\le\; & 22n\varepsilon + \frac{2n|\hat{\sigma}_j^2 - \bar{\sigma}_j^2|}{\operatorname{gap}(\hat{\sigma}_j^2, \{\bar{\sigma}_{j+1}^2\})} + \frac{2n|\hat{\sigma}_{j+1}^2 - \bar{\sigma}_{j+1}^2|}{\operatorname{gap}(\hat{\sigma}_{j+1}^2, \{\bar{\sigma}_j^2\})} + \\ & \frac{3n\varepsilon}{\operatorname{relgap}_2(\sigma_j, \{\bar{\sigma}_{j+1}\})} + \frac{3n\varepsilon}{\operatorname{relgap}_2(\sigma_{j+1}, \{\bar{\sigma}_j\})}, \end{aligned} \tag{4.5.71}$$

and by the relative accuracy of $\hat{\sigma}_j^2$ (see (4.5.67)),

$$|\hat{v}_j^T \hat{v}_{j+1})| \;\leq\; 22n\varepsilon + \frac{2nh(n)\varepsilon}{\mathrm{relgap}(\hat{\sigma}_j^2, \{\bar{\sigma}_{j+1}^2\})} + \frac{2nh(n)\varepsilon}{\mathrm{relgap}(\hat{\sigma}_{j+1}^2, \{\bar{\sigma}_j^2\})} + \frac{3n\varepsilon}{\mathrm{relgap}_2(\sigma_j, \{\bar{\sigma}_{j+1}\})} + \frac{3n\varepsilon}{\mathrm{relgap}_2(\sigma_{j+1}, \{\bar{\sigma}_j\})}. \tag{4.5.72}$$

where $\bar{\sigma}_i{}^2$ is the ith eigenvalue of $\bar{L}\bar{D}\bar{L}^T$ (see Figure 4.5) and differs from σ_i^2 by a few ulps.

Note that we have chosen to exhibit the result in two forms, namely, (4.5.71) and (4.5.72). The difference is in the second and third terms of the two bounds. Since we only care about each term in the bound being $O(\varepsilon)$, often it may not be necessary to find an eigenvalue to full relative accuracy. For example, suppose $\sigma_n^2 = \varepsilon$ and $\sigma_{n-1}^2 = 1$, then there is *no need to find all the correct digits of σ_n^2*. Instead, absolute accuracy will suffice and be more efficient in such a case. The first result (4.5.71) makes this clear.

We now show that irrespective of the relative gaps, Algorithm X always computes eigenpairs with small residual norms.

Theorem 4.5.6 *Let $T + \mu I$ be a positive definite tridiagonal matrix and let $(\hat{\sigma}_j^2, \hat{v}_j)$ be the approximate eigenpairs computed by Algorithm X. Then their residual norms are small, i.e., for $j = 1, 2, \ldots, n$,*

$$\|(T + \mu I - \hat{\sigma}_j^2 I)\hat{v}_j\| \;\leq\; \Big(2\|T\|_1(2nh(n) + 11n^{3/2} + 4) + 48\Big) \cdot \varepsilon, \tag{4.5.73}$$

assuming that (4.5.66) holds.

Proof. We refer the reader to Figure 4.5 for notation used in this proof. Recall that by definition the vector $\tilde{v}_j$ is the exact solution to

$$(\bar{L}\bar{D}\bar{L}^T - \hat{\sigma}_j^2 I)\tilde{v}_j \;=\; \frac{\tilde{\gamma}_r}{\|\tilde{z}\|} e_r. \tag{4.5.74}$$

The elements of $\bar{L}$ and $\bar{D}$ are small relative perturbations of the elements of L and D. In particular,

$$\bar{d}_i \;=\; d_i(1 + \eta_1^i), \tag{4.5.75}$$

$$\bar{d}_i \bar{l}_i \;=\; d_i l_i (1 + \eta_2^i) \tag{4.5.76}$$

$$\bar{d}_i \bar{l}_i^2 \;=\; d_i l_i^2 (1 + \eta_3^i) \tag{4.5.77}$$

where

$$|\eta_1^i| < 4\varepsilon, \quad |\eta_2^i| < 3.5\varepsilon, \quad \text{and} \quad |\eta_3^i| < 5\varepsilon \tag{4.5.78}$$

(these bounds on η^i's can be deduced from Theorems 4.4.2, 4.4.3 and 4.4.4). Consider the ith equation of (4.5.74),

$$\bar{d}_{i-1}\bar{l}_{i-1}\tilde{v}_j(i-1) + (\bar{d}_{i-1}\bar{l}_{i-1}^2 + \bar{d}_i - \hat{\sigma}_j^2)\tilde{v}_j(i) + \bar{d}_i\bar{l}_i\tilde{v}_j(i+1) \;=\; K_i,$$

where $K_i = 0$ if $i \neq r$ and $K_r = \tilde{\gamma}_r/\|\tilde{z}\|$. Substituting (4.5.68), (4.5.75), (4.5.76) and (4.5.77) into the above equation, we get

$$d_{i-1}l_{i-1}\hat{v}_j(i-1) + (d_{i-1}l_{i-1}^2 + d_i - \hat{\sigma}_j^2)\hat{v}_j(i) + d_il_i\hat{v}_j(i+1) \;=\; K_i + \delta_i,$$

where

$$\begin{aligned} |\delta_i| \;<\; & |\varepsilon_{i-1}d_{i-1}l_{i-1}| + |\varepsilon_i| \cdot (|d_{i-1}l_{i-1}^2| + |d_i|) + |\varepsilon_{i+1}d_il_i|) + \\ & |\eta_2^{i-1}\tilde{v}_j(i-1)| + (|\eta_3^{i-1}| + |\eta_1^i|) \cdot |\tilde{v}_j(i)| + |\eta_2^i\tilde{v}_j(i+1)|. \end{aligned}$$

In the above, the ε_i's are as in (4.5.68). Since $|\varepsilon_i| < 11n\varepsilon$ and by (4.5.78), we can bound δ_i by

$$|\delta_i| \;<\; 11n\varepsilon \cdot \|LDL^T\|_1 + 16\varepsilon \cdot \max\{\tilde{v}_j(i-1), \tilde{v}_j(i), \tilde{v}_j(i+1)\}.$$

Thus

$$(LDL^T - \hat{\sigma}_j^2 I)\hat{v}_j \;=\; r_j. \tag{4.5.79}$$

Substituting the bound for $|\gamma_r|$ from (4.5.66), we get

$$\|r_j\| \;<\; (2nh(n)\hat{\sigma}_j^2 + 11n^{3/2}\|LDL^T\|_1 + 48) \cdot \varepsilon \;<\; \Big((2nh(n) + 11n^{3/2})\|LDL^T\|_1 + 48\Big) \cdot \varepsilon.$$

By the backward stability of the Cholesky factorization (see Corollary 4.4.1),

$$T + \delta T + \mu I \;=\; LDL^T, \tag{4.5.80}$$

where $\|\delta T\|_1 \leq 8\varepsilon\|T\|_1$. From (4.5.79) and (4.5.80),

$$(T + \mu I - \hat{\sigma}_j^2 I)\hat{v}_j \;=\; (LDL^T - \hat{\sigma}_j^2 I - \delta T)\hat{v}_j \;=\; r_j - \delta T \cdot \hat{v}_j,$$

and the result follows since $\|LDL^T\|_1 \leq 2\|T\|_1$ (recall that $|\mu| \leq \|T\|$). □

4.5.4 Discussion of Error Bounds

The careful reader may worry about the $nh(n)$ and $n^{3/2}$ terms in the bounds on the residual norms of (4.5.73), and the $nh(n)$ term in the dot product bound given by (4.5.72).

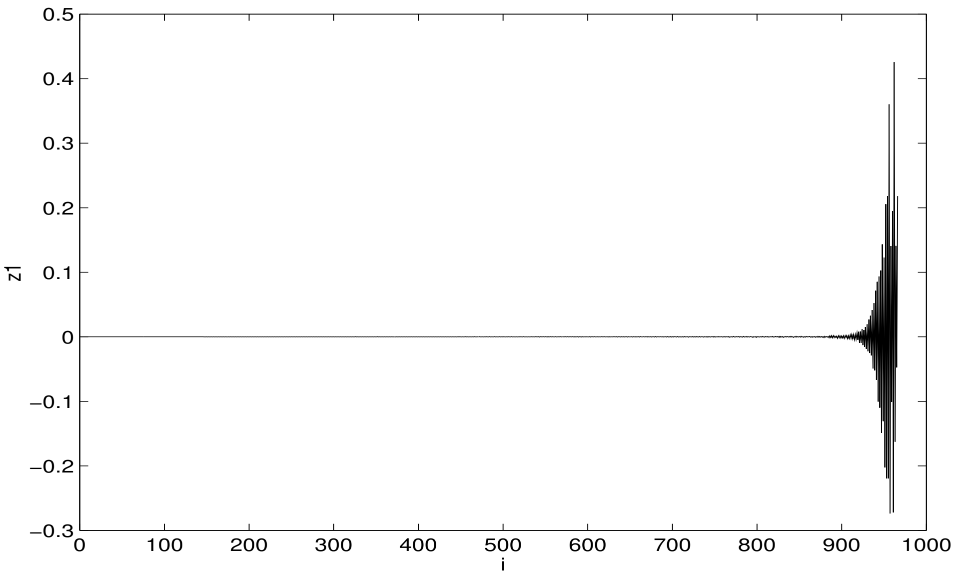

Figure 4.6: An eigenvector of a tridiagonal : most of its entries are negligible

The fear is that when n is large enough, these terms are no longer negligible and may lead to a loss in accuracy. We reassure the reader that our error bounds can be overtly pessimistic and this is borne out by numerical experience.

Often, an eigenvector of a tridiagonal is non-negligible only in a tiny fraction of its entries, see Figure 4.6. When this happens we say that the eigenvector has small "support". In such a case, the error bounds of (4.5.72) and (4.5.73) can effectively be reduced by replacing n with $|\mathrm{supp}|$ which denotes the support of the eigenvector under consideration. For example, as in Figure 4.6, we may have $n = 1000$ but $|\mathrm{supp}|$ may be just over 50 for some eigenvectors.

We give some pointers to the various places where our bounds may be pessimistic.

1. The small $|\mathrm{supp}|$ of an eigenvector can lead to smaller error bounds in Theorem 4.5.2 and Corollary 4.5.3. This would mean that n can be replaced by $|\mathrm{supp}|$ in the terms $22n\varepsilon$ and $11n^{3/2}$ that occur in (4.5.72) and (4.5.73) respectively.

2. The bounds of Corollaries 4.3.2 and 4.3.3 that are used frequently may also permit n to be replaced by $|\mathrm{supp}|$.

3. The term $h(n)$ in (4.5.67) may be quite small. Indeed, if bisection is used for computing the eigenvalues, then $h(n) = O(|\mathrm{supp}|)$.

4. The $\sqrt{n}$ term in (4.5.62) is really $\|\tilde{v}_j\|_\infty^{-1}$ (see Theorem 3.2.3) and can be $O(1)$ if the largest entry of the normalized eigenvector is $O(1)$. Consequently, our results will be more accurate than as suggested by our error bounds.

4.5.5 Orthogonality in Extended Precision Arithmetic

Suppose the user wants d digits of accuracy, i.e., residual norms and dot products of about 10^{-d} are acceptable. Until now, we have implicitly assumed that the arithmetic precision in which we compute is identical to the acceptable level of accuracy, e.g., we desire $O(\varepsilon)$ accuracy in our goals of (1.1.1) and (1.1.2) where ε is the precision of the arithmetic. Can a desired level of accuracy by guaranteed by arithmetic of a higher precision? For example, the user may want single precision accuracy when computing in double precision, or we may aim for double precision accuracy by computing in quadruple precision arithmetic. The IEEE standard also specifies a Double-Extended precision format (sometimes referred to as "80-bit" arithmetic) and on some machines, these *extra precise* computations may be performed in hardware [2, 65]. Quadruple precision arithmetic is generally simulated in software [91, 120].

In order to get single precision accuracy, we may try and execute Algorithm X in double precision arithmetic. However, there are cases when this simple strategy will not work. Consider Wilkinson's matrix W_{21}^+ where the largest pair of eigenvalues agree to more than 16 digits (see [136, p.309] for more details). By the theory developed in Section 4.5, the corresponding eigenvectors computed by Algorithm X can be nearly parallel even if we compute in double precision! And indeed in a numerical run, we observe large dot products. Thus we cannot use the doubled precision accuracy in a naive manner.

We now indicate without proof that Algorithm X can easily be modified to deliver results that are accurate to a desired accuracy when operating in arithmetic of *doubled* precision, i.e., by slightly modifying Algorithm X, **it almost always delivers eigenpairs with $O(\sqrt{\varepsilon})$ residual norms and dot products when using arithmetic of precision** ε. The modification is that after computing the LDL^T decomposition in Step 2 of Algorithm X, random componentwise perturbations of $O(\sqrt{\varepsilon})$ should be made in the elements of L and D. This perturbation makes it unlikely that relative gaps between the eigenvalues of the perturbed LDL^T will be smaller than $O(\sqrt{\varepsilon})$. In many cases, the above effect can be achieved just by performing the computation to get the LDL^T decomposition in $\sqrt{\varepsilon}$ precision arithmetic. The computation of the eigenvalues and eigenvectors by Steps 3 and 4 of Algorithm X must, of course, be performed in the doubled precision. The results of Theorem 4.5.6 and Corollary 4.5.4 now imply that both residual norms and dot products are $O(\sqrt{\varepsilon})$.

4.6 Numerical Results

We now provide numerical evidence to verify our claims of the last two sections. We consider two types of matrices with the following distribution of eigenvalues.

Type 1. $n-1$ eigenvalues uniformly distributed from ε to $(n-1)\varepsilon$, and the nth eigenvalue at 1, i.e.,

$$\lambda_i = i \cdot \varepsilon, \quad i = 1, 2, \ldots, n-1, \text{ and } \lambda_n = 1.$$

Type 2. One eigenvalue at ε, $n-2$ eigenvalues uniformly distributed from $1+\sqrt{\varepsilon}$ to $1+(n-2)\sqrt{\varepsilon}$, and the last eigenvalue at 2, i.e.,

$$\lambda_1 = \varepsilon, \ \ \lambda_i = 1 + (i-1) \cdot \sqrt{\varepsilon}, \quad i = 2, \ldots, n-1, \text{ and } \lambda_n = 2.$$

We generated matrices of the above type using the LAPACK test matrix generator [36], which first forms a random dense symmetric matrix with the given spectrum and Householder reduction of this dense matrix then yields a tridiagonal of the desired type.

In Table 4.1, we compare the times taken by our new algorithm with the LAPACK and EISPACK implementations of inverse iteration on matrices of type 1. The $O(n^3)$ behavior of the LAPACK and EISPACK codes is seen in this table while Algorithm X takes $O(n^2)$ time. We see that Table 4.1 shows the new algorithm to be consistently faster — it is about 3 times faster on a matrix of size 50 and nearly 23 times faster on a 1000×1000 matrix. As we proved in the last section, Algorithm X delivers vectors that are numerically orthogonal, and this is seen in Table 4.2.

On matrices of type 2, our theory predicts that the vectors we compute will have dot products of about $\sqrt{\varepsilon}$ (see Corollary 4.5.4). Indeed, as Table 4.3 shows, that is what we observe. The times taken by Algorithm X to compute the vectors in this case are approximately the same as the times listed in Table 4.1. Vectors that have dot products of $O(\sqrt{\varepsilon})$ are sometimes referred to as a *semi-orthogonal basis*. In some cases, such a basis may be as good as an orthogonal basis, see [114].

Matrix Size	Time(LAPACK) (in s.)	Time(EISPACK) (in s.)	Time(Alg. X) (in s.)	Time(Alg. X) / Time(LAPACK)
50	0.10	0.09	0.03	3.33
100	0.45	0.34	0.07	6.43
250	3.60	2.32	0.37	9.73
500	19.88	11.21	1.35	14.73
750	57.53	29.65	2.98	19.31
1000	124.98	60.81	5.51	22.68

Table 4.1: Timing results on matrices of type 1

Matrix	$\max_i \|T\hat{v}_i - \hat{\sigma}_i^2\hat{v}_i\|$			$\max_{i\neq j} \|\hat{v}_i^T\hat{v}_j\|$		
Size	LAPACK	EISPACK	Alg. X	LAPACK	EISPACK	Alg. X
50	$1.6\cdot10^{-16}$	$5.9\cdot10^{-15}$	$1.2\cdot10^{-16}$	$1.1\cdot10^{-15}$	$2.8\cdot10^{-15}$	$2.5\cdot10^{-15}$
100	$3.1\cdot10^{-17}$	$1.5\cdot10^{-15}$	$1.3\cdot10^{-17}$	$1.1\cdot10^{-15}$	$5.7\cdot10^{-15}$	$2.0\cdot10^{-15}$
250	$1.1\cdot10^{-16}$	$2.1\cdot10^{-14}$	$1.1\cdot10^{-16}$	$1.7\cdot10^{-15}$	$1.4\cdot10^{-14}$	$1.5\cdot10^{-14}$
500	$1.1\cdot10^{-16}$	$4.9\cdot10^{-14}$	$5.5\cdot10^{-18}$	$3.5\cdot10^{-15}$	$2.0\cdot10^{-14}$	$4.2\cdot10^{-14}$
750	$1.1\cdot10^{-16}$	$3.6\cdot10^{-14}$	$3.2\cdot10^{-18}$	$4.6\cdot10^{-15}$	$3.9\cdot10^{-14}$	$4.1\cdot10^{-14}$
1000	$1.2\cdot10^{-17}$	$5.1\cdot10^{-14}$	$2.2\cdot10^{-16}$	$4.4\cdot10^{-15}$	$6.0\cdot10^{-14}$	$8.3\cdot10^{-14}$

Table 4.2: Accuracy results on matrices of type 1

Matrix Size	$\max_i \|T\hat{v}_i - \hat{\sigma}_i^2\hat{v}_i\|$	$\max_{i\neq j} \|\hat{v}_i^T\hat{v}_j\|$
50	$7.2\cdot10^{-16}$	$5.2\cdot10^{-9}$
100	$1.0\cdot10^{-15}$	$3.9\cdot10^{-9}$
250	$1.5\cdot10^{-16}$	$2.5\cdot10^{-9}$
500	$2.1\cdot10^{-15}$	$2.1\cdot10^{-9}$
750	$2.6\cdot10^{-15}$	$2.4\cdot10^{-9}$
1000	$2.9\cdot10^{-15}$	$1.7\cdot10^{-9}$

Table 4.3: Accuracy results on matrices of type 2

A nice property of Algorithm X is that the dot products of the vectors computed can be predicted quite accurately, based solely on the relative separation of the eigenvalues. As exhibited in Sections 2.8 and 4.2, existing implementations do not have such a property. This feature is useful when larger dot products are acceptable, such as in the case of semi-orthogonal vectors.

Algorithm X is a major step in obtaining an $O(n^2)$ algorithm for the symmetric tridiagonal problem. However, it does not always deliver vectors that are numerically orthogonal. In the next chapter, we investigate how to extend Algorithm X in order to always achieve the desired accuracy while doing only $O(n^2)$ work.

Chapter 5

Multiple Representations

In the previous chapter, we saw how to obtain vectors that are guaranteed to be numerically orthogonal when eigenvalues have large relative gaps. In this chapter and the next, we concentrate our energies on the case when relative gaps are smaller than a threshold, say $1/n$. Such eigenvalues will be called a *cluster* throughout this chapter.

The following is our plan of attack:

1. In Section 5.1, we present two examples which suggest that orthogonality may be achieved by shifting the matrix close to a cluster and then forming a bidiagonal factorization of the shifted matrix. The aim is to clearly distinguish between the individual eigenvalues of the cluster so that we can treat each eigenvalue as isolated and compute the corresponding eigenvector as before. If the bidiagonal factorization is "good", the computed vectors will be nearly orthogonal.

2. In Section 5.2, we list the properties that a "good" bidiagonal factorization must satisfy. We call such a factorization a *relatively robust representation.* Section 5.2.1 introduces relative condition numbers that indicate when a representation is relatively robust. In Section 5.2.3, we investigate factorizations of nearly singular tridiagonal matrices and attempt to explain why these representations are almost always relatively robust.

3. In Section 5.3, we explain why the vectors computed using different relatively robust representations turn out to be numerically orthogonal. We do so by introducing a *representation tree* which we use as a visual tool for our exposition. A representation

tree summarizes the computation and helps in relating the various representations to each other.

4. In Section 5.4, we present Algorithm Y which is an enhancement to Algorithm X that was earlier presented in Chapter 4. Algorithm Y takes $O(n^2)$ time and handles the remaining case of small relative gaps. Unlike Algorithm X, we do not have a proof of correctness of Algorithm Y as yet. In all our numerical testing, which we present in the next chapter, we have always found it to deliver accurate answers.

5.1 Multiple Representations

Example 5.1.1 [Small Relative Gaps.] Consider the matrix

$$\mathrm{T}_0 = \begin{bmatrix} .520000005885958 & .519230209355285 & & \\ .519230209355285 & .589792290767499 & .36719192898916 & \\ & .36719192898916 & 1.89020772569828 & 2.7632618547882\cdot 10^{-8} \\ & & 2.7632618547882\cdot 10^{-8} & 1.00000002235174 \end{bmatrix}$$

with eigenvalues

$$\lambda_1 \approx \varepsilon, \quad \lambda_2 \approx 1+\sqrt{\varepsilon}, \quad \lambda_3 \approx 1+2\sqrt{\varepsilon}, \quad \lambda_4 \approx 2.0,$$

where $\varepsilon \approx 2.2\times 10^{-16}$ (all these results are in IEEE double precision arithmetic). Since

$$\mathrm{relgap}_2(\lambda_2,\lambda_3) \;\approx\; \sqrt{\varepsilon},$$

Corollary 4.5.4 implies that the vectors computed by Algorithm X have a dot product of

$$|\hat{v}_2^T\hat{v}_3| \;=\; O(n\varepsilon/\sqrt{\varepsilon}) \;=\; O(n\sqrt{\varepsilon}).$$

The challenge is to obtain approximations to v_2 and v_3 that are numerically orthogonal without resorting to Gram-Schmidt or a similar technique that explicitly orthogonalizes vectors.

Note that the eigenvectors of T_0 are identical to the eigenvectors of $T_0 - I$. We can form

$$T_0 - I \;=\; L_0D_0L_0^T, \tag{5.1.1}$$

to get

$$\mathrm{diag}(\mathrm{D}_0) = \begin{bmatrix} -.4799999941140420 \\ .1514589857947483 \\ 3.074504323352656 \cdot 10^{-7} \\ 1.986821250068485 \cdot 10^{-8} \end{bmatrix}, \qquad \mathrm{diag}(\mathrm{L}_0, -1) = \begin{bmatrix} -1.081729616088125 \\ 2.424365428451800 \\ .08987666186707277 \end{bmatrix},$$

where we have used the MATLAB notation "diag$(L_0, -1)$" to give the subdiagonal entries of L_0. Note that the interior eigenvalues of this shifted matrix are $\sqrt{\varepsilon}$ and $2\sqrt{\varepsilon}$. *The relative gap between these numbers is now large!* Can we exploit these large relative gaps as we did in the previous chapter?

Suppose LDL^T is a factorization where small relative changes in L and D result in tiny changes in all its eigenvalues and a corresponding small change in the eigenvectors. By revisiting the proofs that lead to Corollary 4.5.4, we discover that in such a case if Steps 3 and 4 of Algorithm X are applied to LDL^T, then the computed vectors will be nearly orthogonal. Indeed, if we apply these steps to $L_0D_0L_0^T$, the vectors computed are

$$\hat{\mathrm{v}}_2 = \begin{bmatrix} .4999999955491866 \\ .4622227251939223 \\ -.1906571596350473 \\ .7071067841882251 \end{bmatrix}, \qquad \hat{\mathrm{v}}_3 = \begin{bmatrix} .4999999997942006 \\ .4622227434674882 \\ -.1906571264658161 \\ -.7071067781848689 \end{bmatrix},$$

and $\hat{v}_2^T \hat{v}_3 = 2\varepsilon$! It appears to be a miracle that by considering a translate of the original T that makes the relative gap small, we are able to compute eigenvectors that are orthogonal to working accuracy. □

Clearly, success in the above example is due to the property of the decomposition $L_0D_0L_0^T$ by which all of its eigenvalues change by small relative amounts under small componentwise perturbations. In Section 4.3, we saw that every positive definite tridiagonal LDL^T enjoys this benign property. However, not every decomposition of an indefinite tridiagonal shares this property, see Example 5.2.3 for one such decomposition. But the important question is: can we always find such "relatively robust" representations "near" a cluster?

One distinguishing feature of the LDL^T decomposition of a positive definite matrix is the lack of any element growth. For the decomposition $T - \mu I = LDL^T$, let us define[1]

$$\text{Element Growth} \stackrel{\text{def}}{=} \max \left| \frac{D(i,i)}{T(i,i) - \mu} \right|^{1/2}. \tag{5.1.2}$$

[1] we do not have a strong preference for this particular definition of element growth; indeed, it may be possible to get a "better" definition

When $T - \mu I$ is positive definite, the element growth always equals 1 (see Lemma 4.4.1). For the matrix T_0 of Example 5.1.1, we can verify that the element growth at $\mu = 1$ also equals 1. We might suspect a correlation between the lack of element growth and high relative accuracy. We speculate further on this correlation in Section 5.2.3.

Suppose there is a large element growth when forming LDL^T. Small relative perturbations in the large elements of L and D result in large *absolute* perturbations. Thus it appears unlikely that the eigenvalues of such an LDL^T can be computed to absolute accuracy. Our goal of computing the small eigenvalues of LDL^T to high relative accuracy seems to hold out even less hope. However, as the following example shows, we can be in a peculiar but lucky situation.

Example 5.1.2 [Large Element Growth.] Consider the matrix

$$\mathrm{T}_1 = \begin{bmatrix} 1.00000001117587 & .707106781186547 & & \\ .707106781186547 & .999999977648258 & .707106781186546 & \\ & .707106781186546 & 1.00000003352761 & 1.05367121277235 \cdot 10^{-8} \\ & & 1.05367121277235 \cdot 10^{-8} & 1.00000002235174 \end{bmatrix}$$

whose eigenvalues are approximately equal to those of T_0 (see Example 5.1.1). To get orthogonal approximations to the interior eigenvectors we can try the same trick as before. However when we form $T_1 - I = L_1 D_1 L_1^T$, we get

$$\mathrm{diag}(\mathrm{D}_1) = \begin{bmatrix} 1.117587089538574 \cdot 10^{-8} \\ -4.473924266666669 \cdot 10^{7} \\ 4.470348380358755 \cdot 10^{-8} \\ 1.986821493746602 \cdot 10^{-8} \end{bmatrix}, \qquad \mathrm{diag}(\mathrm{L}_1, -1) = \begin{bmatrix} 6.327084877781628 \cdot 10^{7} \\ -1.580506693551128 \cdot 10^{-8} \\ .2357022791274500 \end{bmatrix}.$$

There is appreciable element growth in forming the $(2,2)$ element of D_1, and we may be skeptical of using this factorization to get the desired results. But there is no harm in trying to apply Steps 3 and 4 of Algorithm X to $L_1 D_1 L_1^T$ to compute the interior eigenvectors. By doing so, we get

$$\hat{\mathrm{v}}_2 = \begin{bmatrix} .4999999981373546 \\ 2.634178061370114 \cdot 10^{-9} \\ -.4999999981373549 \\ .7071067838207259 \end{bmatrix}, \qquad \hat{\mathrm{v}}_3 = \begin{bmatrix} -.5000000018626457 \\ -1.317089024797209 \cdot 10^{-8} \\ .5000000018626453 \\ .7071067785523688 \end{bmatrix}, \qquad (5.1.3)$$

and miraculously $|\hat{v}_2^T \hat{v}_3| < 3\varepsilon$*!* The corresponding residual norms are also small! □

The above example appears to be an anomaly. Due to the large entry $D_1(2,2)$, we should not even expect absolute accuracy in the computed eigenvalues and eigenvectors. Note that in the above example we used $L_1D_1L_1^T$ to compute only $\hat{v}_2$ and $\hat{v}_3$. It turns out that small componentwise perturbations in L_1 and D_1 result in small relative changes in only the two small eigenvalues of $L_1D_1L_1^T$, but large relative (and hence, absolute) changes in the extreme eigenvalues. We get inaccurate answers if we attempt to compute approximations to v_1 and v_4 using $L_1D_1L_1^T$. These extreme eigenvectors must be computed using a different representation, say the Cholesky decomposition of T_1. We investigate this seemingly surprising phenomenon in Section 5.2.1. More details on the two examples discussed above may be found in Case Study C.

5.2 Relatively Robust Representations (RRRs)

In Algorithm X and the two examples of the previous section, triangular factorizations of translates of T allow us to compute orthogonal eigenvectors whenever *the eigenvalues of these factorizations* have large relative gaps. We now identify the crucial property of these decompositions that enables such computation.

Informally, a *relatively robust representation* is a set of numbers that define a matrix A such that small componentwise perturbations in these numbers result in a small relative change in the eigenvalues, and the change in the eigenvectors is inversely proportional to the relative gaps in the eigenvalues. For example, a unit lower bidiagonal L and a *positive* diagonal matrix D are the triangular factors of the tridiagonal matrix LDL^T, and form a relatively robust representation as shown by the theory outlined in Section 4.3. In the following, we denote the jth eigenvalue and eigenvector of A by λ_j and v_j respectively, while the corresponding perturbed eigenvalue and eigenvector are denoted by $\lambda_j + \delta\lambda_j$ and $v_j + \delta v_j$. More precisely,

Definition 5.2.1 A *Relatively Robust Representation* or RRR is a set of numbers $\{x_i\}$ that define a matrix A such that if x_i is perturbed to $x_i(1+\varepsilon_i)$, then for $j = 1, 2, \ldots, n$,

$$\begin{aligned} \frac{|\delta\lambda_j|}{|\lambda_j|} &= O\left(\sum_i \varepsilon_i\right), \\ |\sin\angle(v_j, v_j + \delta v_j)| &= O\left(\frac{\sum_i \varepsilon_i}{\text{relgap}(\lambda_j, \{\lambda_k | k \neq j\})}\right), \end{aligned}$$

where relgap is the relative gap as defined in Section 4.3.

Typically, the RRRs we consider will be triangular factors L and D of the shifted matrix $T - \mu I$. We will frequently refer to such a factorization as a representation of T (based) at the shift μ. Also, for the sake of brevity, we will often refer to the underlying matrix LDL^T as the RRR (instead of the individual factors L and D). Sometimes a representation may determine only a few but not *all* its eigenvalues to high relative accuracy. For example, a representation may only determine the eigenvalues $\lambda_j, \lambda_{j+1}, \ldots, \lambda_k$ to high relative accuracy. In such a case, we say that we have a partial RRR and denote it by RRR$(j, j+1, \ldots, k)$. In the next section, we will show that the representation of T_0 at the shift 1 is an RRR while that of T_1 at 1 is a partial RRR$(2, 3)$, where T_0 and T_1 are the matrices of Examples 5.1.1 and 5.1.2 respectively.

5.2.1 Relative Condition Numbers

We now find a criterion to judge if the factors L and D form an RRR. Instead of dealing with LDL^T we switch to a Cholesky-like decomposition for the purpose of our analysis,

$$LDL^T \;=\; \tilde{L}\Omega\tilde{L}^T,$$

where $\Omega = \mathrm{sign}(D)$ is a matrix with ± 1 entries and explicitly captures the "indefiniteness" of the matrix, and $\tilde{L}$ is lower triangular. Note that Ω is the identity matrix when LDL^T is positive definite. We made a similar switch in Chapter 5 where we analyzed the perturbation properties of the Cholesky factor while performing our computations with LDL^T. The relationship between these alternate representations is easy to see and is given by

$$\tilde{L} \;=\; L\,|D|^{1/2}.$$

By Theorem 4.4.1, both these representations are similar in terms of their behavior under small relative perturbations.

Thus we consider the factorization

$$T \;= \tilde{L}\Omega\tilde{L}^T. \tag{5.2.4}$$

As shown in Section 4.3, if we make small componentwise perturbations in $\tilde{L}$, the perturbed bidiagonal may be represented as $D_1\tilde{L}D_2$ where D_1 and D_2 are diagonal matrices close to the identity matrix. We are interested in answering the following question. Are the eigenvalues of the perturbed matrix

$$T + \delta T \;=\; D_1\tilde{L}D_2\Omega D_2^T\tilde{L}^T D_1^T$$

relatively close to the eigenvalues of $\tilde{L}\Omega\tilde{L}^T$ and if so, when? Note that the answer is *always* in the affirmative when $\Omega = I$, irrespective of the individual entries in $\tilde{L}$.

By Eisenstat and Ipsen's Theorem 2.1 in [49], the eigenvalues of $D_1^T A D_1$ are small relative perturbations of the eigenvalues of A if $\|D_1\| \approx 1$. By applying their result to our case, we get

$$\frac{\lambda_j[\tilde{L}D_2\Omega D_2^T\tilde{L}^T]}{\|D_1\|^2} \;\leq\; \lambda_j[T+\delta T] \;\leq\; \lambda_j[\tilde{L}D_2\Omega D_2^T\tilde{L}^T]\cdot\|D_1\|^2, \tag{5.2.5}$$

where $\lambda_j[A]$ denotes the jth eigenvalue of A.

We now write D_2 as $I+\Delta_2$ where $\|\Delta_2\| = O(\varepsilon)$. In proving the following result, we use the fact that an eigenvalue of a matrix is a continuous function of its entries.

Theorem 5.2.1 *Let $\lambda = \lambda_j$ be a simple eigenvalue of $\tilde{L}\Omega\tilde{L}^T$ with eigenvector $v = v_j$. Let $\lambda + \delta\lambda$ be the corresponding eigenvector of $\tilde{L}(I+\Delta_2)\Omega(I+\Delta_2^T)\tilde{L}^T$. Then*

$$\delta\lambda \;=\; v^T\tilde{L}(\Delta_2\Omega + \Omega\Delta_2^T)\tilde{L}^T v + O(\|\Delta_2\|^2\cdot\|\tilde{L}^T v\|^2), \tag{5.2.6}$$

or

$$\frac{|\delta\lambda|}{|\lambda|} \;\leq\; \frac{|v^T\tilde{L}\tilde{L}^T v|}{|v^T\tilde{L}\Omega\tilde{L}^T v|}\cdot 2\|\Delta_2\| + O\left(\frac{|v^T\tilde{L}\tilde{L}^T v|}{|v^T\tilde{L}\Omega\tilde{L}^T v|}\cdot\|\Delta_2\|^2\right). \tag{5.2.7}$$

Proof. By continuity of the eigenvalues, we can write

$$\left(\tilde{L}(I+\Delta_2)\Omega(I+\Delta_2^T)\tilde{L}^T\right)(v+\delta v) \;=\; (\lambda+\delta\lambda)(v+\delta v).$$

Expanding the terms, we get

$$\begin{aligned}(\tilde{L}\Delta_2\Omega\tilde{L}^T + \tilde{L}\Omega\Delta_2\tilde{L}^T + \tilde{L}\Delta_2\Omega\Delta_2\tilde{L}^T)\cdot v + \tilde{L}\Omega\tilde{L}^T\cdot\delta v + &\\ (\tilde{L}\Delta_2\Omega\tilde{L}^T + \tilde{L}\Omega\Delta_2\tilde{L}^T + \tilde{L}\Delta_2\Omega\Delta_2\tilde{L}^T)\cdot\delta v \;&=\; \lambda\delta v + \delta\lambda\cdot v + \delta\lambda\cdot\delta v.\end{aligned}$$

Premultiplying both sides by v^T,

$$v^T\tilde{L}\Delta_2\Omega\tilde{L}^T v + v^T\tilde{L}\Omega\Delta_2\tilde{L}^T v + \lambda v^T\delta v + O(\|L^T v\|^2\cdot\|\Delta_2\|^2) \;=\; \lambda v^T\delta v + \delta\lambda,$$

and by canceling the common term on both sides, we obtain (5.2.6). Dividing both sides by the eigenvalue λ and taking norms yields the result (5.2.7). □

We call the ratio of the Rayleigh Quotients given in (5.2.7) as the *relative condition number of* λ_j, and denote it by κ_{rel}, i.e.,

$$\kappa_{rel}(\lambda_j) \;=\; \frac{|v_j^T\tilde{L}\tilde{L}^T v_j|}{|v_j^T\tilde{L}\Omega\tilde{L}^T v_j|}. \tag{5.2.8}$$

By combining (5.2.5) and (5.2.7) and using the above notation, we get

$$\begin{aligned}\{1 - 2\,\|I - D_2\| \cdot \kappa_{rel}(\lambda_j[T])\}\,\frac{\lambda_j[T]}{\|D_1\|^2} \;&\le\; \lambda_j[T + \delta T] \\ &\le\; \lambda_j[T]\,\|D_1\|^2\,\{1 + 2\,\|I - D_2\| \cdot \kappa_{rel}(\lambda_j[T])\}\,, \end{aligned} \quad (5.2.9)$$

which is correct to first-order terms.

We draw the reader's attention to the following facts :

1. Unlike the case of $\Omega = I$ where all eigenvalues are relatively robust (see Section 4.3), here a condition number measures the relative robustness of an eigenvalue.

2. The relative condition number κ_{rel} is *different for each eigenvalue.* Thus some eigenvalues may be determined to high relative accuracy whereas others may not be. See Example 5.2.2 for such an example.

3. A uniform bound for κ_{rel} is $\|\tilde{L}\| \cdot \|\tilde{L}^{-1}\|$, and this bound holds for all eigenvalues (see (5.2.13) below). However this bound can be a severe over-estimate as is shown by the examples given later in Section 5.2.2.

4. Note that we did not constrain $\tilde{L}$ to a bidiagonal form in deriving (5.2.9). However, relative perturbations in the individual entries of a dense matrix $\tilde{L}$ *cannot* be expressed as $D_1\tilde{L}D_2$. There is only a restricted class of matrices whose componentwise perturbations can be written in the form $D_1\tilde{L}D_2$. Matrices that belong to this class are completely characterized by the sparsity pattern of their non-zeros and have been called the set of *biacyclic matrices*, a term coined by Demmel and Gragg in [33]. Besides bidiagonal matrices, the twisted factors introduced in Section 3.1 and triangular factors of *arrowhead* matrices (see [73, p.175]) belong to this class of matrices. Thus the theory developed in this section is applicable not only to a bidiagonal matrix but also to any biacyclic matrix.

By defining the vector

$$f_j \;\stackrel{\text{def}}{=}\; \frac{\tilde{L}^T v_j}{\sqrt{|\lambda_j|}}, \quad (5.2.10)$$

the relative condition number defined in (5.2.8) may be simply written as a norm,

$$\kappa_{rel}(\lambda_j) \;=\; f_j^T f_j \;=\; \|f_j\|^2. \quad (5.2.11)$$

Note that the following relationships hold,

$$\tilde{L}^T v_j \;=\; f_j\sqrt{|\lambda_j|}, \quad \tilde{L}\Omega f_j \;=\; v_j \mathrm{sign}(\lambda_j)\sqrt{|\lambda_j|},$$

$$\text{and} \qquad f_j^T \Omega f_j \;=\; \mathrm{sign}(\lambda_j), \quad v_j^T v_j \;=\; 1,$$

and so we call f_j, the jth right Ω-singular vector of $\tilde{L}$. Since $\tilde{L}\Omega\tilde{L}^T v_j = \lambda_j v_j$, and by (5.2.10), f_j may alternately be expressed as

$$f_j \;=\; \mathrm{sign}(\lambda_j)\sqrt{|\lambda_j|}\cdot \Omega \tilde{L}^{-1} v_j. \tag{5.2.12}$$

(5.2.10) and (5.2.12) suggest a bound for $f_j^T f_j$,

$$\kappa_{rel}(\lambda_j) \;=\; f_j^T f_j \;\le\; \mathrm{cond}(\tilde{L}) \;=\; \|\tilde{L}\|\cdot\|\tilde{L}^{-1}\|, \qquad \forall j. \tag{5.2.13}$$

However the above bound is rather pessimistic since we want to compute the small eigenvalues of $\tilde{L}\Omega\tilde{L}^T$ to high relative accuracy when the matrix is nearly singular, i.e., $\tilde{L}$ *is ill-conditioned.*

In [116], Parlett also arrived at such condition numbers for a bidiagonal $\tilde{L}$ but by using calculus.

Theorem 5.2.2 (Parlett [116, Thm. 2]) *Let $\tilde{L}$ be a bidiagonal matrix as in (4.3.3), with $a_k \neq 0$, $b_k \neq 0$ and let $\Omega = \mathrm{diag}(\omega_1, \ldots, \omega_n)$ with $\omega_k = \pm 1$. Let (λ, v) denote a particular eigenpair of $\tilde{L}\Omega\tilde{L}^T$ and let f be as defined in (5.2.10). Then by writing $\lambda = \mathrm{sign}(\lambda)\theta^2$, since $\lambda \neq 0$,*

$$(a) \quad \frac{\partial\theta}{\partial a_k}\cdot\frac{a_k}{\theta} \;=\; \sum_{i=1}^{k} v(i)^2 - \mathrm{sign}(\lambda)\sum_{j=1}^{k-1}\omega_j f(j)^2 = \mathrm{sign}(\lambda)\sum_{m=k}^{n}\omega_m f(m)^2 - \sum_{l=k+1}^{n} v(l)^2,$$

$$(b) \quad \frac{\partial\theta}{\partial b_k}\cdot\frac{b_k}{\theta} \;=\; \mathrm{sign}(\lambda)\sum_{i=1}^{k}\omega_i p(i)^2 - \sum_{j=1}^{k} v(j)^2 = \sum_{m=k+1}^{n} v(m)^2 - \mathrm{sign}(\lambda)\sum_{l=k+1}^{n}\omega_l p(l)^2.$$

For more work of a similar flavor, see [34].

5.2.2 Examples

We now examine relative condition numbers of some factorizations

$$T - \mu I = \tilde{L}\Omega\tilde{L}^T,$$

where T is tridiagonal and $\tilde{L}$ is lower bidiagonal. In particular, we show that

i. The representation of T_0 at 1 is an RRR, where T_0 is as given in Example 5.1.1.

ii. The representation of T_1 at 1 is a partial RRR, where T_1 is the matrix of Example 5.1.2.

iii. A representation of $T - \mu I$ at an arbitrary shift μ may not determine *any* eigenvalue to high relative accuracy.

Example 5.2.1 [An RRR (All Eigenvalues are Relatively Well-Conditioned).] Consider the $L_0 D_0 L_0^T$ decomposition where D_0 and L_0 are as in Example 5.1.1. The corresponding Cholesky-like decomposition

$$\tilde{L}_0 \Omega \tilde{L}_0^T \;=\; L_0 D_0 L_0^T$$

has

$$\mathrm{diag}(\tilde{\mathrm{L}}_0) \;=\; \begin{bmatrix} .6928203187797266 \\ .3891773192193351 \\ 5.544821298610673 \cdot 10^{-4} \\ 1.409546469637835 \cdot 10^{-4} \end{bmatrix}, \qquad \mathrm{diag}(\tilde{\mathrm{L}}_0, -1) \;=\; \begin{bmatrix} -.7494442574516461 \\ .9435080382529064 \\ 4.983500289685748 \cdot 10^{-5} \end{bmatrix},$$

while $\mathrm{diag}(\Omega) = (-1, 1, 1, 1)$. The eigenvalues of $\tilde{L}_0 \Omega \tilde{L}_0^T$ are approximately

$$\lambda_1 = -1, \qquad \lambda_2 = 1.49 \cdot 10^{-8}, \qquad \lambda_3 = 2.98 \cdot 10^{-8}, \qquad \lambda_4 = 1,$$

and

$$\mathrm{cond}(\tilde{L}) \;=\; \|\tilde{L}\| \cdot \|\tilde{L}^{-1}\| \;=\; 9.46 \cdot 10^3.$$

We find the Ω-right singular vectors f_j to be

$$\mathrm{f}_1 = \begin{bmatrix} 1.01 \\ -.144 \\ 7.5 \cdot 10^{-5} \\ -5.3 \cdot 10^{-13} \end{bmatrix}, \; \mathrm{f}_2 = \begin{bmatrix} 8.8 \cdot 10^{-5} \\ -3.1 \cdot 10^{-4} \\ .577 \\ -.816 \end{bmatrix}, \; \mathrm{f}_3 = \begin{bmatrix} 1.2 \cdot 10^{-4} \\ -4.4 \cdot 10^{-4} \\ .816 \\ .577 \end{bmatrix}, \text{ and } \mathrm{f}_4 = \begin{bmatrix} -.144 \\ 1.01 \\ 5.2 \cdot 10^{-4} \\ 3.7 \cdot 10^{-12} \end{bmatrix}.$$

and by (5.2.11), the individual relative condition numbers are

$$\kappa_{rel}(\lambda_1) \;=\; 1.042, \qquad \kappa_{rel}(\lambda_2) \;=\; 1.000, \qquad \kappa_{rel}(\lambda_3) \;=\; 1.000, \qquad \kappa_{rel}(\lambda_4) \;=\; 1.042.$$

All the relative condition numbers are close to 1 and explain our success in computing orthogonal eigenvectors as exhibited in Example 5.1.1. □

In the above example, we found the bidiagonal factorization of a nearly singular tridiagonal to be an RRR. In our numerical experience, the above situation appears to be typical. However, sometimes the factorization may determine only a few eigenvalues to high relative accuracy. Somewhat surprisingly, *eigenvalues that are small in magnitude may be determined to high relative accuracy but not the large ones!*

Example 5.2.2 [Partial RRR (Only Small Eigenvalues are Well-Conditioned).] Consider the factors L_1 and D_1 given in Example 5.1.2. The corresponding Cholesky-like decomposition

$$\tilde{L}_1\Omega_1\tilde{L}_1^T \;=\; L_1D_1L_1^T$$

has

$$\mathrm{diag}(\tilde{\mathrm{L}}_1) \;=\; \begin{bmatrix} 1.05715987472128\cdot 10^{-4} \\ 6.68874025674659\cdot 10^{3} \\ 2.11431974944257\cdot 10^{-4} \\ 1.40954648562579\cdot 10^{-4} \end{bmatrix}, \qquad \mathrm{diag}(\tilde{\mathrm{L}}_1,-1) \;=\; \begin{bmatrix} 6.68874025674659\cdot 10^{3} \\ -1.05715987472128\cdot 10^{-4} \\ 4.98349983747794\cdot 10^{-5} \end{bmatrix},$$

and $\mathrm{diag}(\Omega_1) = (1,-1,1,1)$. The eigenvalues of $\tilde{L}_1\Omega_1\tilde{L}_1^T$ are approximately

$$\lambda_1 = -1, \qquad \lambda_2 = 1.49\cdot 10^{-8}, \qquad \lambda_3 = 2.98\cdot 10^{-8}, \qquad \lambda_4 = 1,$$

and

$$\mathrm{cond}(\tilde{L}_1) \;=\; \|\tilde{L}_1\|\cdot\|\tilde{L}_1^{-1}\| \;=\; 1.37\cdot 10^{8}.$$

The Ω-right singular vectors for the two small eigenvalues are

$$\mathrm{f}_2 = \begin{bmatrix} .577 \\ .577 \\ -.577 \\ .816 \end{bmatrix}, \quad \text{and} \quad \mathrm{f}_3 = \begin{bmatrix} .816 \\ .816 \\ -.816 \\ -.577 \end{bmatrix},$$

and the corresponding relative condition numbers are

$$\kappa_{rel}(\lambda_2) \;=\; 1.666, \qquad \kappa_{rel}(\lambda_3) \;=\; 2.333.$$

Thus the two eigenvalues that are small in magnitude are relatively well conditioned but for the extreme eigenvalues we have

$$\mathrm{f}_1 = \begin{bmatrix} -4.7\cdot 10^{3} \\ -4.7\cdot 10^{3} \\ 1.1\cdot 10^{-4} \\ -7.4\cdot 10^{-13} \end{bmatrix}, \quad \text{and} \quad \mathrm{f}_4 = \begin{bmatrix} 4.7\cdot 10^{3} \\ 4.7\cdot 10^{3} \\ 1.1\cdot 10^{-4} \\ 7.4\cdot 10^{-13} \end{bmatrix},$$

and

$$\kappa_{rel}(\lambda_1) \;=\; 4.5 \cdot 10^7, \qquad \kappa_{rel}(\lambda_4) \;=\; 4.5 \cdot 10^7!$$

□

There are also factorizations where none of the eigenvalues is determined to high relative accuracy.

Example 5.2.3 [No Eigenvalue is Relatively Well-Conditioned.] Let the matrix T_0 be as in Example 5.1.1, and consider

$$T_0 - 1.075297677018139I \;=\; \tilde{L}\Omega\tilde{L}^T.$$

We have intentionally chosen the above shift to be very close to an eigenvalue of the leading 2×2 submatrix of T_0. Consequently, there is a large element growth in this factorization,

$$\mathrm{diag}(\tilde{\mathrm{L}}) \;=\; \begin{bmatrix} .7451829782893468 \\ 2.018543658277998 \cdot 10^{-7} \\ 1.819093322471722 \cdot 10^{6} \\ .2744041812115826 \end{bmatrix}, \qquad \mathrm{diag}(\tilde{\mathrm{L}}, -1) \;=\; \begin{bmatrix} -.6967821655658823 \\ 1.819093322471946 \times 10^{6} \\ -1.519032487587594 \times 10^{-14} \end{bmatrix},$$

with $\mathrm{diag}(\Omega) = (-1, 1, -1, -1)$. The approximate eigenvalues of $\tilde{L}\Omega\tilde{L}^T$ are,

$$\lambda_1 = -1.0753, \qquad \lambda_2 = -.0752976, \qquad \lambda_3 = -.0752954, \qquad \lambda_4 = .92473,$$

and

$$\mathrm{cond}(\tilde{L}) \;=\; \|\tilde{L}\| \cdot \|\tilde{L}^{-1}\| \;=\; 2.46 \cdot 10^{13},$$

where this large condition number is primarily due to the large norm of $\tilde{L}$. The relative condition numbers are found to be

$$\kappa_{rel}(\lambda_1) \;=\; 1.1 \cdot 10^{11}, \;\; \kappa_{rel}(\lambda_2) \;=\; 6.4 \cdot 10^{12}, \;\; \kappa_{rel}(\lambda_3) \;=\; 6.8 \cdot 10^{7}, \;\; \kappa_{rel}(\lambda_4) \;=\; 6.5 \cdot 10^{12},$$

and indicate that *none of the eigenvalues is determined to high relative accuracy.* Note that none of the eigenvalues is small and the lack of relative accuracy implies an absence of any absolute accuracy! □

Other examples of RRRs may be found in Section 6 of [116]. In order to see how the relative robustness of LDL^T representations of $T_0 - \mu I$ varies with μ, see Figures C.1, C.3 and C.3 in Case Study C.

5.2.3 Factorizations of Nearly Singular Tridiagonals

The purpose of taking multiple representations is to differentiate between the individual eigenvalues in a cluster. It is crucial that the "refined" eigenvalues of the representation based near the cluster have modest relative condition numbers, which were defined in (5.2.8). In this section we indicate why most triangular factorizations of nearly singular tridiagonals are relatively robust, at least for the small eigenvalues.

Since we do not have a complete theory as yet to explain all our numerical experiments, we shall present some conjectures in this section and give some insight into why they might be true.

In Section 5.1, we speculated on a possible correlation between the element growth when forming

$$T - \mu I \;=\; \tilde{L}\Omega\tilde{L}^T,$$

and the relative robustness of the triangular factorization. In all our numerical experience, we have always found $\tilde{L}$ and Ω to be relatively robust whenever there is no element growth. Thus we believe the following conjecture.

Conjecture 1 *If*

$$\frac{|\,(\tilde{L}\tilde{L}^T)_{ii}\,|}{|\,(\tilde{L}\Omega\tilde{L}^T)_{ii}\,|} \;=\; O(1), \quad \text{for} \quad 1 \le i \le n, \tag{5.2.14}$$

then $\tilde{L}$ and Ω form a relatively robust representation (RRR), where an RRR is as defined in Section 5.2.

Note that

$$\frac{|(\tilde{L}\tilde{L}^T)_{ii}|}{|(\tilde{L}\Omega\tilde{L}^T)_{ii}|} \;=\; \frac{|e_i^T\tilde{L}\tilde{L}^Te_i|}{|e_i^T\tilde{L}\Omega\tilde{L}^Te_i|},$$

whereas

$$\kappa_{rel}(\lambda_j) \;=\; \frac{|v_j^T\tilde{L}\tilde{L}^Tv_j|}{|v_j^T\tilde{L}\Omega\tilde{L}^Tv_j|}.$$

The quantity on the left hand side of (5.2.14) is a measure of the element growth in forming $\tilde{L}\Omega\tilde{L}^T$. The above conjecture speculates that $\kappa_{rel}(\lambda_j)$ is $O(1)$ for **all** λ_j whenever (5.2.14) is satisfied.

Note: All the conjectures presented here should be taken "in spirit only", and not as precise mathematical conjectures. For example, the condition (5.2.14) might not be exactly correct, but a measure of element growth should be $O(1)$ to guarantee an RRR.

Even if we get a large element growth, the eigenvalues of interest may still be determined to high relative accuracy. We saw one such occurrence in Example 5.1.2. Again, extensive numerical testing leads us to believe the following.

Conjecture 2 *Suppose that $\tilde{L}\Omega\tilde{L}^T$ is "nearly" singular. Then the eigenvalue of $\tilde{L}\Omega\tilde{L}^T$ that is smallest in magnitude is always determined to high relative accuracy with respect to small relative changes in entries of $\tilde{L}$.*

We consider an extreme example in support of the above conjecture. Suppose $\tilde{L}\Omega\tilde{L}^T$ is a singular matrix. Since

$$\det(\tilde{L}\Omega\tilde{L}^T) \;=\; \pm\det(\tilde{L}^2),$$

singularity of $\tilde{L}\Omega\tilde{L}^T$ implies that a diagonal element of $\tilde{L}$ must be zero. Relative changes in the individual entries of such an $\tilde{L}$ keep the matrix singular no matter how large the non-zero elements of $\tilde{L}$ are (since a relative perturbation of a zero entry keeps it zero). Thus we may expect that when $\tilde{L}\Omega\tilde{L}^T$ is close to singularity, its smallest eigenvalue will not change appreciably due to small componentwise changes in $\tilde{L}$.

Note that in the above conjecture, we have not specified how close $\tilde{L}\Omega\tilde{L}^T$ should be to a singular matrix. We believe that the exact condition will be a natural outcome of a proof of the conjecture, if true.

Although, Conjecture 2 is interesting in its own right, it is insufficient for our purposes. We want *all* locally small eigenvalues, and not just the smallest, to be determined to high relative accuracy. An example sheds some light on the possible scenarios.

Example 5.2.4 [2nd Smallest Eigenvalue should be Relatively Well-Conditioned.] Consider Wilkinson's matrix [136, p.308],

$$W_{21}^+ \;=\; \begin{bmatrix} 10 & 1 & & & & & 0 \\ 1 & 9 & 1 & & & & \\ & 1 & 8 & . & & & \\ & & . & . & . & & \\ & & & . & 8 & 1 & \\ & & & & 1 & 9 & 1 \\ 0 & & & & & 1 & 10 \end{bmatrix}, \tag{5.2.15}$$

which has pairs of eigenvalues that are close to varying degrees. For example, the eigenvalues

$$\begin{aligned}\hat{\lambda}_{14} &= 7.003951798616376,\\ \hat{\lambda}_{15} &= 7.003952209528675,\end{aligned}$$

agree in 6 leading digits. We might try and form either

$$W_{21}^{+} - \hat{\lambda}_{14} I \;=\; L_1 D_1 L_1^T, \tag{5.2.16}$$

or

$$W_{21}^{+} - \hat{\lambda}_{15} I \;=\; L_2 D_2 L_2^T \tag{5.2.17}$$

in order to compute orthogonal approximations to v_{14} and v_{15}. We encounter a large element growth when forming (5.2.16) but not in (5.2.17). Consistent with Conjectures 1 and 2, we find $L_2 D_2 L_2^T$ to be an RRR whereas $L_1 D_1 L_1^T$ only determines its smallest eigenvalue to high relative accuracy. In particular,

$$\kappa_{rel}(\lambda_{14}[L_1 D_1 L_1^T]) \;=\; 2.246, \qquad \kappa_{rel}(\lambda_{15}[L_1 D_1 L_1^T]) \;=\; 7.59 \times 10^8,$$

while

$$\kappa_{rel}(\lambda_{14}[L_2 D_2 L_2^T]) \;=\; 1.0, \qquad \kappa_{rel}(\lambda_{15}[L_2 D_2 L_2^T]) \;=\; 1.0.$$

Thus $L_1 D_1 L_1^T$ is inadequate for our purposes whereas $L_2 D_2 L_2^T$ enables us to compute $\hat{v}_{14}$ and $\hat{v}_{15}$ that are numerically orthogonal. □

Finally, as in the above example, we believe we can always find a partial RRR based at one of the eigenvalues in the cluster.

Conjecture 3 *Suppose T is a tridiagonal matrix and $\hat{\lambda}_j, \hat{\lambda}_{j+1}, \ldots, \hat{\lambda}_{j+m-1}$ are approximations to m "close" eigenvalues of T, that agree in almost all their digits. Then at least one of the factorizations*

$$T - \hat{\lambda}_s I \;=\; L_s D_s L_s^T, \qquad j \le s < j+m,$$

is a partial RRR($j, j+1, \ldots, j+m-1$), i.e., $L_s D_s L_s^T$ determines its locally small eigenvalues to high relative accuracy.

The relative robustness of factorizations of nearly singular tridiagonals is consistent with our earlier Conjecture 1. This is because we do not get any element growth in the generic case when forming

$$T - \hat{\lambda} I \;=\; LDL^T,$$

where $\hat{\lambda}$ is close to an eigenvalue of T.

Conjecture 3 is **exactly** what we need to achieve our goal of computing orthogonal eigenvectors. Furthermore, it would be even more efficient if we could easily find the particular $\hat{\lambda}_s$ in Conjecture 3 that leads to a relatively robust representation.

We leave it for future studies to provide further numerical evidence and theory to prove or disprove the validity of the conjectures made in this section. From now on, we will assume that Conjecture 3 is true so that we can find the desired RRRs.

5.2.4 Other RRRs

Until now, we have only considered computing a triangular decomposition, LDL^T or $\tilde{L}\Omega\tilde{L}^T$, as an RRR. However, we could also consider the twisted factorizations introduced in Section 3.1 as RRRs. Indeed, since the set of n possible twisted factorizations

$$T - \mu I \;=\; N^{(k)}\Delta N^{(k)^T}, \qquad\qquad 1 \leq k \leq n,$$

includes the LDL^T decomposition, it may be easier to form such RRRs. Besides, it is conceivable that a twisted factorization may be qualitatively better than a triangular decomposition in terms of its relative perturbation properties. We believe that *all* conjectures of the previous section also hold for twisted factorizations, and not only for triangular factorizations.

5.3 Orthogonality using Multiple Representations

The examples of the previous section indicate that eigenvectors computed using a single RRR turn out to be numerically orthogonal. If relative gaps are small, we form an RRR near the cluster to get locally small eigenvalues that have large relative gaps. Sometimes, as in Example 5.1.2, we can only obtain a partial RRR the use of which guarantees orthogonality of a limited set of eigenvector approximations. In such a case, the remaining eigenvectors need to be computed using a different RRR. In this section, we explain why the approximations computed from one RRR are numerically orthogonal to the approximations computed using another RRR. An added benefit of the *representation trees* that we will introduce for our exposition is that they summarize the computations involved and help us in identifying computationally efficient strategies.

5.3.1 Representation Trees

In Section 4.5, we provided a detailed proof of orthogonality of the eigenvectors computed by Algorithm X using the LDL^T decomposition of a positive definite tridiagonal matrix when the eigenvalues have large relative gaps. This proof can easily be generalized to the case when any fixed RRR is used for the computation. Similarly we can furnish a detailed proof of orthogonality when different RRRs are used for computing different eigenvectors. However, to avoid being drowned in detail, we choose an alternate approach. We now introduce *representation trees* that informally indicate why vectors computed using one RRR are orthogonal to those computed from another RRR. A formal treatment and greater level of detail as in Corollary 4.5.4 can also be reproduced by mimicking its proof.

We remind the reader of our indexing convention for eigenvalues, by which

$$\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n.$$

A *representation tree* consists of nodes that are denoted by (R, I) where R stands for an RRR and I is a subset of the index set $\{1, 2, \ldots, n\}$. Informally, each node denotes an RRR that is used enroute to computing eigenvectors of the eigenvalues $\lambda_i[R]$ indexed by I, i.e., $i \in I$ (note that we have used R in two ways — to denote the RRR and also for the underlying matrix defined by the RRR). A parent node (R_p, I_p) can have m children (R_j, I_j), $1 \le j \le m$, such that

$$\cup_j I_j \;=\; I_p.$$

Any pair of the child nodes $(R_1, I_1), (R_2, I_2)$ must satisfy

$$I_1 \subseteq I_p, \qquad I_2 \subseteq I_p, \qquad I_1 \cap I_2 = \phi, \tag{5.3.18}$$

$$\text{and} \quad \mathrm{relgap}_2(\lambda_i[R_p], \lambda_j[R_p]) \;\ge\; 1/n, \qquad \forall i \in I_1, j \in I_2,\ i \ne j, \tag{5.3.19}$$

where relgap_2 is as defined in Section 4.3. Nodes that have no children will be called *leaf nodes* while all other nodes are called *intermediate nodes*. The index set associated with any leaf node must be a singleton. Each edge connecting a parent node to a child node will be labeled by a real number μ. Informally, an edge denotes the action of forming an RRR by applying a shift of μ to the matrix denoted by the parent RRR. Additionally, the shifts used to form the leaf representations must be very accurate approximations to the appropriate eigenvalues of the leaf's parent RRR. The condition on the relative gaps given by (5.3.19) ensures that different RRRs are formed in order to compute eigenvectors when relative gaps are smaller than $1/n$. Note that we choose $1/n$ as the cut-off point for relative gaps since this translates into an acceptable bound of $O(n\varepsilon)$ for the dot products, see Corollary 4.5.4 for details. The above conditions should become clearer when we examine some example representation trees.

In all the representation trees we consider in this thesis, the RRRs associated with intermediate nodes will be bidiagonal factors of either the given tridiagonal T or its translate. Each RRR must determine to high relative accuracy all the eigenvalues associated with its index set. However, the representations associated with leaf nodes are, in general, twisted factorizations. As we saw in Section 4.5, relative robustness of these twisted factorizations is not necessary for the orthogonality of the computed eigenvectors. However, we have always found such a twisted factorization to determine its smallest eigenvalue to high relative accuracy, see Conjecture 2 and Section 5.2.4. Recall that the eigenvector approximations are formed by multiplying elements of these twisted factorizations, see Step (4d) of Algorithm X.

A representation tree denotes the particular computations involved in computing orthogonal eigenvectors. Since we are concerned about forming representations that are relatively robust, it is crucial that we use the differential transformations of Section 4.4.1 when forming

$$L_p D_p L_p^T - \mu I \;=\; L_1 D_1 L_1^T,$$

so that we can relate the computed decomposition to a small componentwise perturbation of L_p and D_p.

Example 5.3.1 [A Representation Tree.] The RRR tree in Figure 5.1 summarizes a computation of the eigenvectors of the matrix T_0 presented in Example 5.1.1. Figure 5.1, where ovals denote intermediate representations while rectangles stand for leaf nodes, reflects the following information. The initial decomposition

$$T_0 \;=\; L_p D_p L_p^T$$

has eigenvalues

$$\lambda_1 \approx \varepsilon, \qquad \lambda_2 \approx 1+\sqrt{\varepsilon}, \qquad \lambda_3 \approx 1+2\sqrt{\varepsilon}, \qquad \lambda_4 \approx 2.$$

The extreme eigenvectors are computed using the twisted factorizations

$$L_p D_p L_p^T - \hat{\lambda}_1 I \;=\; N_1^{(1)} \Delta_1 N_1^{(1)^T}, \qquad L_p D_p L_p^T - \hat{\lambda}_4 I \;=\; N_4^{(3)} \Delta_4 N_4^{(3)^T},$$

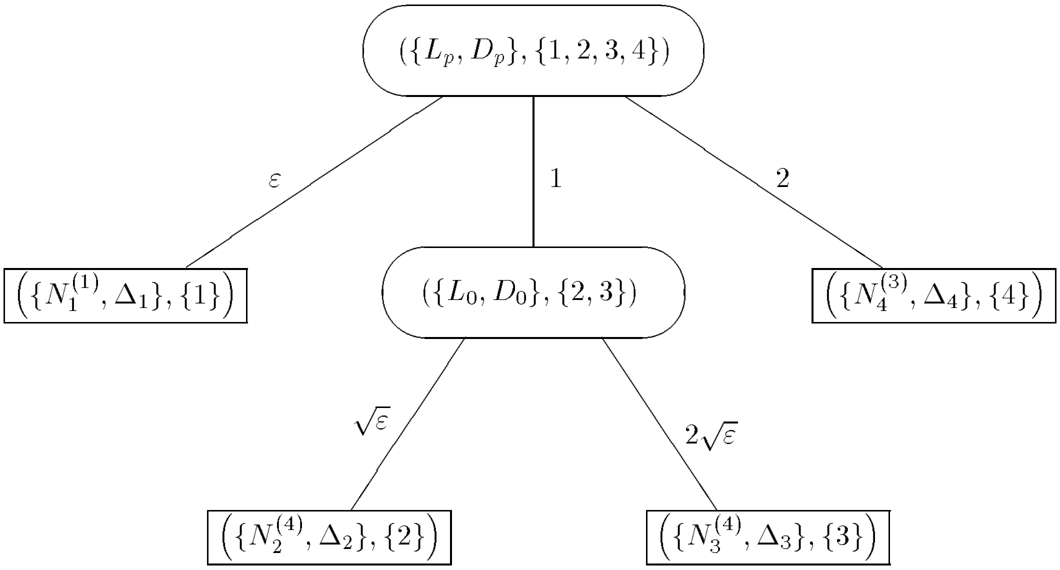

Figure 5.1: Representation Tree — Forming an extra RRR based at 1

where $|\hat{\lambda}_i - \lambda_i| = O(\varepsilon|\lambda_i|)$ and the superscript r in $N_i^{(r)}$ denotes the twist position. The intermediate representation

$$L_p D_p L_p^T - I \;=\; L_0 D_0 L_0^T$$

is needed since $\mathrm{relgap}_2(\lambda_2, \lambda_3) = O(\sqrt{\varepsilon}) < 1/\sqrt{n}$. The two smallest eigenvalues of $L_0 D_0 L_0^T$ are computed to be

$$\delta\hat{\lambda}_2 \;\approx\; \sqrt{\varepsilon}, \qquad \text{and} \qquad \delta\hat{\lambda}_3 \;\approx\; \sqrt{2\varepsilon}.$$

The corresponding eigenvectors may now be computed as

$$L_0 D_0 L_0^T - \delta\hat{\lambda}_2 I \;=\; N_2^{(4)} \Delta_2 N_2^{(4)^T}, \qquad L_0 D_0 L_0^T - \delta\hat{\lambda}_3 I \;=\; N_3^{(4)} \Delta_3 N_3^{(4)^T}.$$

As we mentioned earlier, all the factorizations in the representation tree should be formed by the differential transformations of Section 4.4.1. □

Example 5.3.2 [A Better Representation Tree.] From Example 5.2.1, we know that the $L_0 D_0 L_0^T$ decomposition of $T_0 - I$ determines *all* its eigenvalues to high relative accuracy. Thus the scheme given by the representation tree of Figure 5.2 can alternatively be used to compute orthogonal eigenvectors. The bulk of the work lies in computing the relevant subset of eigenvalues of the intermediate RRRs. Thus the representation tree of Figure 5.2 yields a more efficient computation scheme than Figure 5.1. □

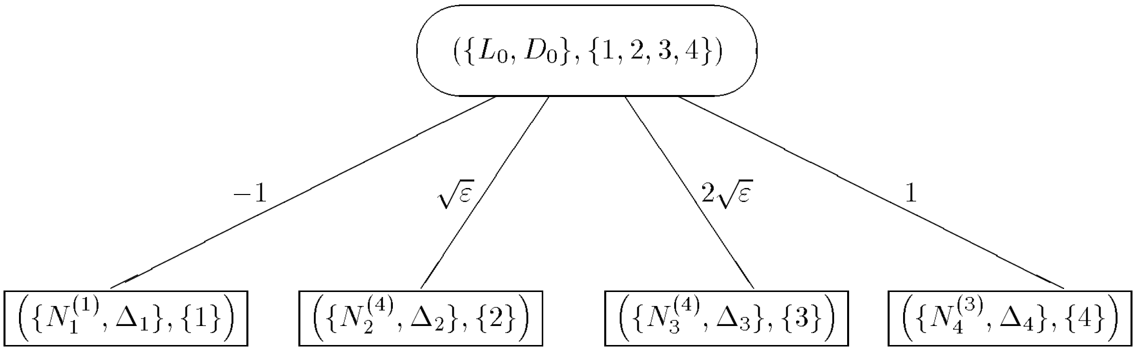

Figure 5.2: Representation Tree — Only using the RRR based at 1

Example 5.3.3 [Only One Representation Tree is Possible.] Orthogonal eigenvectors of the matrix T_1 given in Example 5.1.2 may be computed as shown by the representation tree of Figure 5.3. By Example 5.2.2, $L_1 D_1 L_1^T$ is only a partial RRR, and cannot be used to compute the extreme eigenvectors. Figure C.3 in Case Study C shows that there is no representation based near 1 that is relatively robust for *all* its eigenvalues and hence, there is no representation tree for T_1 that looks like Figure 5.2. $\square$

We now indicate why the computed eigenvectors are numerically orthogonal when the particular computation scheme is described by a representation tree. We re-emphasize the following facts that are important for showing orthogonality.

i. Each intermediate node is a partial RRR for the eigenvalues associated with its index set.

ii. Each node, or representation, is formed by the differential transformations given in Section 4.4.1.

iii. The approximation used to form a leaf representation agrees in almost all its digits with the relevant eigenvalue of its parent node (as given by the index of the leaf node).

iv. The eigenvector approximation is formed solely by multiplications of elements of the twisted factorizations that are represented by the leaf nodes.

v. Whenever the relative gap between eigenvalues of a representation is smaller than $1/n$, a new representation that is relatively robust for its locally small eigenvalues is formed.

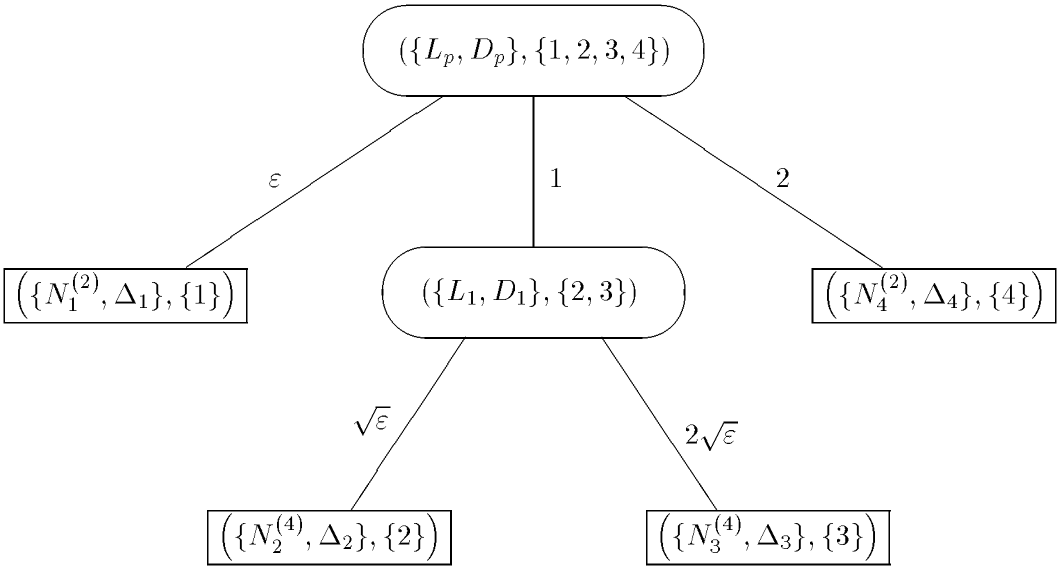

Figure 5.3: Representation Tree — An extra RRR based at 1 is essential

The first four facts outlined above can be used, as in Section 4.5, to prove the orthogonality of eigenvector approximations computed from leaf representations that have a common parent. Note that the above statement is analogous to saying that the vectors computed when Steps 3 and 4 of Algorithm X are applied to an RRR can be shown to be orthogonal. Recall that in Section 4.5, orthogonality is proved by relating each computed vector to an exact eigenvector of the matrix defined by the parent RRR.

Representation trees come in handy in seeing why the computed vectors are orthogonal when more than one intermediate RRR is used. Let us consider Figure 5.1. The computed vectors $\hat{v}_2$ and $\hat{v}_3$ are orthogonal for the reasons given in the above paragraph. But why is $\hat{v}_2$ orthogonal to $\hat{v}_1$? Note that $\hat{v}_1$ is computed using

$$L_p D_p L_p^T - \hat{\lambda}_1 I \;=\; N_1^{(1)} \Delta_1 N_1^{(1)^T},$$

while $\hat{v}_2$ is computed from

$$L_0 D_0 L_0^T - \delta\hat{\lambda}_2 I \;=\; N_2^{(4)} \Delta_2 N_2^{(4)^T}.$$

However,

- both $L_p D_p L_p^T$ and $L_0 D_0 L_0^T$ are RRRs;

- $L_0 D_0 L_0^T$ is a translate of $L_p D_p L_p^T$ and is computed by the differential *dstqds* transformation of Section 4.4.1, and the roundoff error analysis given in Theorem 4.4.2 shows that an exact relation exists between small componentwise perturbations of L_p, D_p and L_0, D_0;
- the relative gap between λ_1 and λ_2 is large.

The vector $\hat{v}_1$ can directly be shown to be close to the first eigenvector of $L_p D_p L_p^T$, while $\hat{v}_2$ can be similarly related to the second eigenvector of $L_p D_p L_p^T$ but via $L_0 D_0 L_0^T$. The relative robustness of both these representations along with the above mentioned facts can now be used to prove that $\hat{v}_1$ and $\hat{v}_2$ are orthogonal.

In general, any two eigenvectors computed from the twisted factorizations represented by the leaf nodes of a representation tree will be "good" approximations to distinct eigenvectors of a common ancestor RRR. The properties satisfied by a representation tree now imply that these computed vectors must be orthogonal. Note that the detailed bounds on the dot products of the computed vectors can get rather messy and would involve the number of intermediate representations employed in computing an eigenvector. On the other hand, representation trees provide a visual tool that makes it easy to see why the computed vectors are nearly orthogonal. Besides this use, representation trees also allow us to clearly see which computations are more efficient.

5.4 Algorithm Y — orthogonality even when relative gaps are small

We now present an algorithm that also handles small relative gaps but still performs the computation in $O(n^2)$ time.

Algorithm Y [Computes orthogonal eigenvectors.]

1. Find $\mu \leq \|T\|$ such that $T + \mu I$ is positive (or negative) definite.
2. Compute $T + \mu I = L_p D_p L_p^T$.
3. Compute the eigenvalues, $\hat{\lambda}_j$, of $L_p D_p L_p^T$ to high relative accuracy by the **dqds** algorithm [56].
4. Set $l \leftarrow 1$, $m \leftarrow n$.

5. Group the computed eigenvalues $\hat{\lambda}_l, \ldots, \hat{\lambda}_m$ into the categories:

   isolated. $\hat{\lambda}_j$ is isolated if

$$\min(\mathrm{relgap}_2(\hat{\lambda}_j, \hat{\lambda}_{j+1}), \mathrm{relgap}_2(\hat{\lambda}_{j-1}, \hat{\lambda}_j)) \;\geq\; 1/n.$$

   clustered. $\hat{\lambda}_j, \ldots, \hat{\lambda}_{j+k-1}$ form a "cluster" of k eigenvalues if

$$\mathrm{relgap}_2(\hat{\lambda}_i, \hat{\lambda}_{i+1}) \;\leq\; 1/n, \qquad j \leq i < j+k-1,$$

$$\text{while} \quad \mathrm{relgap}_2(\hat{\lambda}_{j-1}, \hat{\lambda}_j) \;\geq\; 1/n, \quad \text{and} \quad \mathrm{relgap}_2(\hat{\lambda}_{j+k-1}, \hat{\lambda}_{j+k}) \;\geq\; 1/n.$$

6. For each isolated eigenvalue, $\hat{\lambda} = \hat{\lambda}_j$, $l \leq j \leq m$, do the following

   (a) Compute $L_p D_p L_p^T - \hat{\lambda} I = L_+ D_+ L_+^T$ by the **dstqds** transform (Algorithm 4.4.3).

   (b) Compute $L_p D_p L_p^T - \hat{\lambda} I = U_- D_- U_-^T$ by the **dqds** transform (Algorithm 4.4.5).

   (c) Compute γ_k as in the top formula of (4.4.26). Pick r such that $|\gamma_r| = \min_k |\gamma_k|$.

   (d) Form the approximate eigenvector $z_j = z_j^{(r)}$ by solving $\hat{N}_r \hat{D}_r \hat{N}_r^T z_j = \hat{\gamma}_r e_r$ (see Theorem 3.2.2):

$$\begin{aligned} z_j(r) &= 1, \\ z_j(i) &= -\hat{L}_+(i) \cdot z_j(i+1), \quad i = r-1, \ldots, 1, \\ z_j(l+1) &= -\hat{U}_-(l) \cdot z_j(l), \qquad l = r, \ldots, n-1. \end{aligned}$$

7. For each cluster $\hat{\lambda}_j, \ldots, \hat{\lambda}_{j+k-1}$ do the following.

   (a) Get a partial RRR$(j, \ldots, j+k-1)$ by forming the dstqds transformation

$$L_p D_p L_p^T - \hat{\lambda}_s I \;=\; L_s D_s L_s^T,$$

   where $j \leq s \leq j+k-1$.

   (b) Compute the jth through $(j+k-1)$th eigenvalues of $L_s D_s L_s^T$ to high relative accuracy and call them $\delta\hat{\lambda}_j, \ldots, \delta\hat{\lambda}_{j+k-1}$.

   (c) Set $l \leftarrow j$, $m \leftarrow i+k-1$, $\hat{\lambda}_i \leftarrow \delta\hat{\lambda}_i$ for $j \leq i \leq j+k-1$, $L_p \leftarrow L_s$, $D_p \leftarrow D_s$, and go to Step 5.

□

The previous section indicates why the vectors computed by Algorithm Y are numerically orthogonal. We now present detailed numerical results comparing a computer implementation of Algorithm Y with existing software routines.

Chapter 6

A Computer Implementation

In this chapter, we give detailed timing and accuracy results of a computer implementation of Algorithm Y, whose pseudocode was given in Section 5.4. The only uncertainty in implementing this algorithm is in its Step (7a), where we need to choose a shift μ near a cluster in order to form the relatively robust representation

$$L_p D_p L_p^T - \mu I \;=\; LDL^T. \tag{6.0.1}$$

We briefly discuss our strategy in Section 6.1. Having found a suitable representation, we need to find its locally small eigenvalues to high relative accuracy in Step (7b) of Algorithm Y. In Section 6.2, we give an efficient scheme to find these small eigenvalues to the desired accuracy. Note that the earlier steps of Algorithm Y have been discussed in great detail in Chapter 4.

Finally, in Section 6.4, we give detailed timing and accuracy results comparing our computer implementation of Algorithm Y with existing LAPACK and EISPACK software. Our test-bed contains a variety of tridiagonal matrices, some from quantum chemistry applications, that highlight the sensitivity of earlier algorithms to different distributions of eigenvalues. The test matrices are discussed in Section 6.4.1.

We find that our implementation of Algorithm Y is uniformly faster than earlier implementations of inverse iteration, while still being accurate. This speed is by virtue of the $O(n^2)$ running time of Algorithm Y as opposed to the earlier algorithms that take $O(n^3)$ time in general. We want to stress that the results presented in the chapter are preliminary — we can envisage more algorithmic enhancements and a better use of the memory hierachy and architectural design (such as multiple functional units in the CPU) of modern computers to further speed up our computer implementation. Some of these enhancements are briefly discussed in Section 6.5, and we hope to incorporate such code improvements in the near future.

6.1 Forming an RRR

Two questions, that need to be resolved to get a computer implementation of Step (7a) of Algorithm Y, were raised in Sections 5.2.1 and 5.2.3:

1. What shift μ near a cluster should we choose so that the LDL^T decomposition of (6.0.1) is an RRR?

2. Given LDL^T, how can we cheaply decide if it is an RRR?

As we saw earlier in Example 5.2.3, not every μ in (6.0.1) leads to an RRR. We do not have any *a priori* way of knowing whether an arbitrary choice of μ would lead to a desired RRR. Indeed, as Example 5.2.4 suggests, there may not be any alternative other than actually forming LDL^T at a judicious choice of μ and then, *a posteriori*, checking whether this decomposition forms an RRR. Since we want an efficient procedure for the latter purpose, we cannot afford to evaluate the relative condition numbers of Section 5.2.1. Thus, answers to the two questions posed above are crucial to a correct implementation.

We made several conjectures in Section 5.2.3 that attempt to answer these questions. In our computer implementation, we have made the following decisions which reflect our belief in these conjectures.

1. After identifying a cluster of eigenvalues $\hat\lambda_j, \hat\lambda_{j+1}, \ldots, \hat\lambda_{j+k-1}$, we restrict our search for an RRR to a factorization based at one of these $\hat\lambda$'s, i.e., to

$$L_pD_pL_p^T - \hat\lambda_s I \;=\; L_sD_sL_s^T, \quad j \le s \le j+k-1. \tag{6.1.2}$$

   This is the strategy given in Step (7a) of Algorithm Y and is consistent with Conjecture 3.

2. When trying to form the desired RRR in (6.1.2), we try $\mu = \hat\lambda_s$ in the order $s = j, j+1, \ldots, j+k-1$. If we find the element growth, as defined in (5.1.2), at some $\hat\lambda_s$ to be less than an acceptable tolerance, say 200, then we immediately accept $L_sD_sL_s^T$ as the desired RRR. By this strategy, we are often able to form an RRR in the first

attempt, i.e., based at the leftmost eigenvalue $\mu = \hat{\lambda}_j$. This saves us the extra work of forming all possible k factorizations of (6.1.2). Note that this approach reflects our belief in Conjecture 1.

3. If all the above choices of μ lead to element growths bigger than 200, as in Example 5.2.2 (see also Case Study C), we choose the $L_s D_s L_s^T$ decomposition that leads to the least element growth as our partial RRR.

We want to emphasize that even though we cannot prove the correctness of the above decisions as yet, our computer implementation gives accurate answers on all our tests. We have tested our implementation on tridiagonal matrices that are quite varied in their eigenvalue distributions. See Section 6.4.1 for details.

6.2 Computing the Locally Small Eigenvalues

Until now, we have not discussed ways of efficiently computing the eigenvalues of a relatively robust representation. All eigenvalues of the LDL^T decomposition of a positive definite matrix may be efficiently found by the dqds algorithm, and this is the method employed in Step 3 of Algorithm Y. In its present form, the dqds algorithm finds the eigenvalues in sequential order, from the smallest to the largest, and always operates on a positive definite matrix. See [56] for more details. The main difficulty in trying to employ the dqds algorithm to find the locally small eigenvalues of an RRR is that in most cases, the RRR will be the factorization of an *indefinite* matrix. It is not known, as yet, if the dqds algorithm can be adapted to an indefinite case and hence we need to find an alternate method.

One means of finding the locally small eigenvalues is the bisection algorithm, using any of the differential transformations given in Section 4.4.1 as the inner loop. However, since bisection is a rather slow method, it could become the dominant part of the computation. So we use a faster scheme which is a slight variant of the Rayleigh Quotient Iteration (RQI). A traditional RQI method starts with some (well-chosen) vector q_0 and progresses by computing Rayleigh Quotients to get increasingly better approximations to an eigenvalue.

Algorithm 6.2.1 [Traditional RQI.]

1. Choose a vector q_0 ($\|q_0\| = 1$), and a scalar θ_0. Set $i \leftarrow 0$.

2. Solve $(T - \theta_i I)x_{i+1} = q_i$ for x_{i+1}.

3. Set $q_{i+1} \leftarrow x_{i+1}/\|x_{i+1}\|$, $\theta_{i+1} \leftarrow q_{i+1}^T T q_{i+1}$, $i \leftarrow i+1$, and repeat Step 2. $\square$

As shown in Corollary 3.2.1, a twisted factorization allows us to cheaply compute the Rayleigh Quotient of the vector z where

$$(T - \theta I)z \;=\; \gamma_r e_r, \quad z(r) = 1, \tag{6.2.3}$$

and γ_r is an element of the twisted factorization at twist position r. It is immediately seen from (6.2.3) that

$$\frac{z^T(T-\theta I)z}{z^T z} \;=\; \frac{\gamma_r}{z^T z}.$$

As discussed earlier, it is possible to choose r so that γ_r is proportional to the distance of θ from an eigenvalue of T. The index r where $|\gamma_r| = \min_k |\gamma_k|$ is one such choice, see Section 3.1 for more details. Thus we get the following iteration scheme.

Algorithm 6.2.2 [RQI-Like (Computes an eigenvalue of $L_s D_s L_s^T$).]

1. Choose a scalar θ_0. Set $i \leftarrow 0$.

2. Choose an index r as follows :

   (a) Compute $L_s D_s L_s^T - \theta_i I = L_+ D_+ L_+^T$ by the **dstqds** transform (Algorithm 4.4.3).

   (b) Compute $L_s D_s L_s^T - \theta_i I = U_- D_- U_-^T$ by the **dqds** transform (Algorithm 4.4.5).

   (c) Compute γ_k as in the top formula of (4.4.26). Pick r such that $|\gamma_r| = \min_k |\gamma_k|$.

3. Solve $(L_s D_s L_s^T - \theta_i I)z_i = \gamma_r e_r$ as follows

$$\begin{aligned} z_i(r) &= 1, \\ z_i(p) &= -L_+(p) \cdot z_i(p+1), \quad p = r-1, \ldots, 1, \\ z_i(q+1) &= -U_-(q) \cdot z_i(q), \qquad q = r, \ldots, n-1. \end{aligned}$$

4. Set $\theta_i = \gamma_r / \|z_i\|^2$, $i \leftarrow i+1$. Repeat Step 2. $\square$

In the above RQI-like scheme, we obtain the Rayleigh Quotient as a by-product of computing the vector z_i. One iteration of the above algorithm is only 2-3 times as expensive as one bisection step, but convergence is at least quadratic. Note that Step 3 given above differs from traditional RQI in its choice of e_r as the right hand side of the linear system to be solved.

As outlined in Step (7a) of Algorithm Y (see Section 5.4), the representation $L_s D_s L_s^T$ is a translate of the original matrix $L_p D_p L_p^T$, i.e.,

$$L_p D_p L_p^T - \mu I \;=\; L_s D_s L_s^T. \qquad (6.2.4)$$

If λ is an eigenvalue of $L_p D_p L_p^T$, $\lambda - \mu$ is the corresponding eigenvalue of $L_s D_s L_s^T$ if the relation (6.2.4) holds exactly. However, we only know an approximate eigenvalue $\hat{\lambda}$, and roundoff errors are inevitable in forming L_s and D_s. But, even though $\hat{\lambda} - \mu$ is not an exact eigenvalue of the computed $L_s D_s L_s^T$, it does give us a very good starting approximation. As a result, θ_0 can be initialized to $\hat{\lambda} - \mu$ in Step 1 of our RQI-like scheme. Of course, as emphasized in Chapter 5, we need to compute each small eigenvalue of $L_s D_s L_s^T$ to high relative accuracy. Since we compute both forward and backward pivots, D_+ and D_-, we can include the safeguards of bisection in our iteration.

6.3 An Enhancement using Submatrices

In this section, we briefly mention a novel idea that facilitates the computation of orthogonal "eigenvectors" of eigenvalues that are very close to each other but are well-separated from the rest of the spectrum.

Eigenvalues of a tridiagonal matrix can be equal only if an off-diagonal element is zero. In such a case, the tridiagonal matrix is a direct sum of smaller tridiagonals, and orthogonal eigenvectors are trivially obtained from the eigenvectors of these disjoint submatrices by padding them with zeros. However, as Wilkinson observed, eigenvalues can be arbitrarily close without any off-diagonal element being small [136]. It turns out that even in such a case, a good orthogonal basis of the invariant subspace can be computed by using suitable, possibly overlapping, submatrices. Thus we can use the following scheme:

Algorithm 6.3.1 [Computes orthogonal "eigenvectors" for tight clusters using submatrices.]

1. For each of the "close" eigenvalues $\lambda_j, \ldots, \lambda_k$ (that are well-separated from the rest of the spectrum), do the following:

   (a) "Find" a submatrix $T^{p:q}$ with an *isolated* eigenvalue λ which is "close" to the cluster of eigenvalues.

(b) Compute the eigenvector of λ, i.e., solve $(T^{p:q} - \lambda I)s = 0$ for s.

(c) Output the vector v as an eigenvector, where $v^{p:q} = s$ and the rest of v is padded with zeroes. □

For some of the theory underlying this scheme, the reader is referred to Parlett [112, 115]. Clearly, besides the existence of suitable submatrices, the crucial question is: how do we choose the submatrices in Step (1a) of the above scheme. The computation of the eigenvector of an isolated eigenvalue in Step (1b) is easily done by using the methods discussed earlier.

We now have a more robust way of picking the appropriate submatrices than the approaches outlined in [112, 115]. We have included this enhancement in our implementation of Algorithm Y and found it to work accurately in practice. Note that the above algorithm is an alternate way of computing orthogonal eigenvectors without doing any explicit orthogonalization. The smaller the submatrix sizes in Step (1b) above, the less is the work required to produce orthogonal eigenvectors.

It is beyond the scope of this thesis to discuss the above approach in greater detail. The theory that justifies this scheme is quite involved and intricate, and a cursory treatment would not do it justice. We hope to present more details in the near future [43].

6.4 Numerical Results

In this section, we present a numerical comparison between Algorithm Y and four other software routines for solving the symmetric tridiagonal eigenproblem that are included in the EISPACK [128] and LAPACK [1] libraries. These are

1. LAPACK INVIT : The LAPACK implementation of bisection and inverse iteration [88, 84, 87] (subroutines DSTEBZ and DSTEIN);
2. EISPACK INVIT : The EISPACK implementation of inverse iteration after finding the eigenvalues by bisection [118] (subroutine DSTEBZ from LAPACK followed by TINVIT from EISPACK);
3. LAPACK D&C : The LAPACK implementation of the divide and conquer method that uses a rank-one tear to subdivide the problem [71, 124] (subroutine DSTEDC);

4. LAPACK QR : The LAPACK implementation of the QR algorithm that uses Wilkinson's shifts to compute both eigenvalues and eigenvectors [69] (subroutine DSTEQR).

6.4.1 Test Matrices

We have chosen many different types of tridiagonals as our test matrices. They differ mainly in their eigenvalue distributions which highlight the sensitivity of the above algorithms as discussed in Chapter 2. Some of our example tridiagonals come from quantum chemistry applications. The first eleven among the following types of tridiagonal matrices are obtained by Householder reduction of random dense symmetric matrices that have the given eigenvalue distributions (see [36] for more on the generation of such matrices). The matrix sizes for our tests range from 125-2000.

1) Uniform Distribution (ε apart). $n-1$ eigenvalues uniformly distributed from ε to $(n-1)\varepsilon$, and the nth eigenvalue at 1, i.e.,

$$\lambda_i = i \cdot \varepsilon, \quad i = 1, 2, \ldots, n-1, \ \text{ and } \ \lambda_n = 1.$$

These matrices are identical to the Type 1 matrices of Section 4.6.

2) Uniform Distribution ($\sqrt{\varepsilon}$ apart). One eigenvalue at ε, $n-2$ eigenvalues uniformly distributed from $1+\sqrt{\varepsilon}$ to $1+(n-2)\sqrt{\varepsilon}$, and the last eigenvalue at 2, i.e.,

$$\lambda_1 = \varepsilon, \ \ \lambda_i = 1 + (i-1) \cdot \sqrt{\varepsilon}, \quad i = 2, \ldots, n-1, \ \text{ and } \ \lambda_n = 2.$$

These are also identical to the Type 2 matrices of Section 4.6.

3) Uniform Distribution (ε to 1). The eigenvalues are equi-spaced between ε and 1, i.e.,

$$\lambda_i \ = \ \varepsilon + (i-1) * \tau, \quad i = 1, 2, \ldots, n$$

where $\tau = (1-\varepsilon)/(n-1)$.

4) Uniform Distribution (ε to 1 with random signs). Identical to the above type of matrices except that a random $\pm$ sign is attached to the eigenvalues.

5) Geometric Distribution (ε to 1). The eigenvalues are geometrically arranged between ε and 1, i.e.,

$$\lambda_i \ = \ \varepsilon^{(n-i)/(n-1)}, \quad i = 1, 2, \ldots, n.$$

6) Geometric Distribution (ε to 1 with random signs). Identical to the above type except that a random $\pm$ sign is attached to the eigenvalues.

7) Random. The eigenvalues come from a random, normal $(0,1)$ distribution.

8) Clustered at 1. $\lambda_1 = \varepsilon$, and $\lambda_2 \approx \lambda_3 \approx \cdots \approx \lambda_n \approx 1$.

9) Clustered at ± 1. Identical to the above type except that a random $\pm$ sign is attached to the eigenvalues.

10) Clustered at ε. $\lambda_1 \approx \lambda_2 \approx \cdots \approx \lambda_{n-1} \approx \varepsilon$, and $\lambda_n = 1$.

11) Clustered at $\pm\varepsilon$. Identical to the above type of matrices except that a random $\pm$ sign is attached to the eigenvalues.

12) (1,2,1) Matrix. These are the Toeplitz tridiagonal matrices with 2's on the diagonals and 1's as the off-diagonal elements. An $n \times n$ version of such a matrix has eigenvalues $4\sin^2[k\pi/2(n+1)]$, and for the values of n under consideration, these eigenvalues are not too close.

Matrices of types 3 through 11 above are LAPACK test matrices and are used to check the accuracy of all LAPACK software for the symmetric tridiagonal eigenproblem. In addition, the following symmetric tridiagonal matrices arise in certain quantum chemistry computations. For more details on these problems, the reader is referred to [10, 55].

13) Biphenyl. This positive definite matrix with $n = 966$ occurs in the modeling of biphenyl using Møller-Plesset theory. Most of its eigenvalues are quite small compared to the norm. See Figure 6.1 for a plot of its eigenvalue distribution.

14) SiOSi$_6$. Density functional theory methods for determining bulk properties for the molecule SiOSi$_6$ lead to this positive definite matrix with $n = 1687$. Most of the eigenvalues are quite close to their neighbors and there is no obvious subdivision of the spectrum into separate clusters. Figure 6.2 gives this distribution.

15) Zeolite ZSM-5. This 2053×2053 matrix occurs in the application of the self-consistent field(SCF) Hartree-Fock method for solving a non-linear Schrödinger problem. See Figure 6.3 for the spectrum.

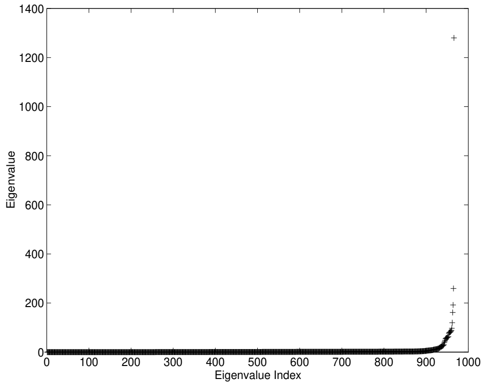

Figure 6.1: Eigenvalue distribution for Biphenyl

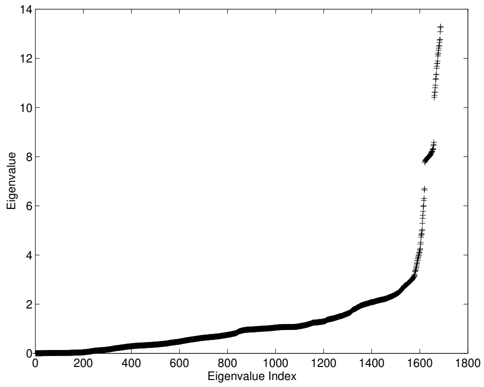

Figure 6.2: Eigenvalue distribution for $SiOSi_6$

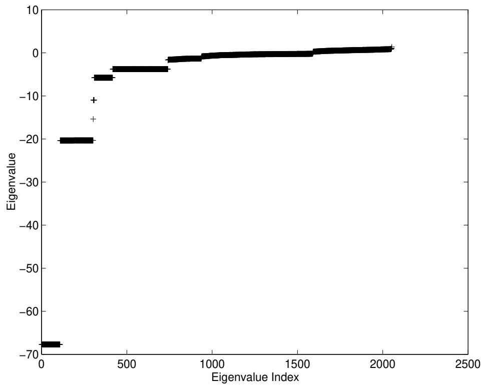

Figure 6.3: Eigenvalue distribution for Zeolite ZSM-5

6.4.2 Timing and Accuracy Results

Tables 6.1 and 6.2 give a comparison of the times taken by the different algorithms to find *all* the eigenvalues and eigenvectors of symmetric tridiagonal matrices of the type discussed above. All the numerical experiments presented in this section were conducted on an IBM RS/6000-590 processor that has a peak rating of 266 MFlops. Fortran BLAS, instead of those from the machine optimized ESSL library [82], were used in this preliminary testing[1]. We hope to include the ESSL BLAS in our future comparisons. The rest of the tables, Table 6.3 through 6.6, indicate the accuracy of the methods tested.

Our Algorithm Y may be thought of as an alternate way of doing inverse iteration. Thus LAPACK INVIT and EISPACK INVIT are the two earlier methods that most closely resemble Algorithm Y. Tables 6.1 and 6.2 show that Algorithm Y is **always** faster than both these existing implementations. The difference in speed varies according to the eigenvalue distribution — on matrices of order 2000, Algorithm Y is over 3500 times faster than LAPACK INVIT when the eigenvalues are clustered around ε while it is 4 times faster when the eigenvalues are well-separated. These different speedups highlight the sensitivity of the various algorithms to the eigenvalue distribution. When eigenvalues are clustered, EISPACK and LAPACK inverse iteration need $O(n^3)$ time, as is clear from Tables 6.1 and 6.2. On the other hand, they take $O(n^2)$ time when eigenvalues are isolated, see the results for matrices of type 4 and type 7 in Table 6.1. We also draw the reader's attention to the varying behavior on matrices with uniformly distributed eigenvalues (from ε to 1), and the $(1, 2, 1)$ matrix. For small n, we see an $O(n^2)$ behavior but for larger n, both EISPACK and LAPACK inverse iteration take $O(n^3)$ time. This discrepancy is due to the clustering criterion which is independent of n — more specifically, reorthogonalization is done when eigenvalues differ by less than $10^{-3}\|T\|$. See Section 2.8.1 for more details. EISPACK's implementation is always faster than LAPACK's but is generally, less accurate. In fact, the latter was designed to improve the accuracy of EISPACK [87].

Algorithm Y is much less sensitive to the arrangement of eigenvalues — on matrices of order 2000, the time taken by it ranges from about 30 seconds to about 60 seconds in most cases. The two notable exceptions are the matrices where almost all eigenvalues are clustered around ε or $\pm\varepsilon$. It turns out that the eigenvectors of such matrices can be very sparse — in fact, an identity matrix is a good approximation to an eigenvector matrix. By

[1] all code was compiled with the command line **f77 -u -O**, where the -O compiler option is identical to the -O2 level of optimization

Matrix Type	Matrix Size	Time Taken (in seconds)				
		LAPACK INVIT	EISPACK INVIT	LAPACK D&C	LAPACK QR	Algorithm Y
Uniform Distribution (ε apart)	125	0.72	0.53	0.02	0.17	**0.14**
	250	3.52	2.27	0.10	1.23	**0.52**
	500	19.78	11.12	0.42	8.92	**1.96**
	1000	123.91	60.24	2.13	68.06	**8.91**
	2000	858.01	369.68	13.18	534.88	**31.87**
Uniform Distribution ($\sqrt{\varepsilon}$ apart)	125	0.55	0.35	0.14	0.16	**0.40**
	250	2.87	1.63	0.69	1.16	**1.54**
	500	17.76	8.76	4.09	7.33	**5.94**
	1000	114.32	50.94	26.34	63.35	**23.17**
	2000	825.91	332.95	178.74	453.93	**92.26**
Uniform Distribution (ε to 1)	125	0.53	0.46	0.16	0.18	**0.14**
	250	2.06	1.81	0.74	1.21	**0.56**
	500	8.08	7.08	4.27	8.61	**1.96**
	1000	124.65	27.55	26.85	68.32	**8.04**
	2000	858.55	370.35	182.64	535.94	**31.91**
Uniform Distribution (ε to 1 with random signs)	125	0.54	0.47	0.15	0.18	**0.14**
	250	2.11	1.85	0.73	1.19	**0.54**
	500	8.20	7.18	4.25	8.35	**1.97**
	1000	32.05	27.81	26.85	61.58	**8.17**
	2000	127.53	109.39	183.26	468.01	**32.30**
Geometric Distribution (ε to 1)	125	0.69	0.54	0.05	0.14	**0.13**
	250	3.29	2.29	0.25	1.03	**0.48**
	500	17.72	10.59	1.22	7.66	**1.86**
	1000	108.90	55.60	6.70	58.98	**7.25**
	2000	759.77	335.68	41.50	442.96	**28.73**
Geometric Distribution (ε to 1 with random signs)	125	0.66	0.50	0.05	0.15	**0.72**
	250	3.29	2.29	0.24	0.96	**2.97**
	500	17.44	10.37	1.17	5.96	**12.79**
	1000	104.68	53.95	6.37	37.34	**53.39**
	2000	711.83	313.82	38.18	247.43	**191.82**
Random	125	0.53	0.47	0.14	0.18	**0.21**
	250	2.07	1.83	0.61	1.21	**0.99**
	500	8.20	7.18	3.54	8.20	**5.43**
	1000	32.18	27.81	21.87	60.75	**13.38**
	2000	132.16	109.83	147.50	459.67	**61.92**

Table 6.1: Timing Results

Matrix Type	Matrix Size	Time Taken (in seconds) LAPACK INVIT	EISPACK INVIT	LAPACK D&C	LAPACK QR	Algorithm Y
Clustered at ε	125	0.69	0.47	0.02	0.14	**0.01**
	250	3.34	2.10	0.04	0.98	**0.001**
	500	18.87	10.19	0.13	7.22	**0.04**
	1000	120.34	56.56	0.42	56.65	**0.09**
	2000	843.89	353.89	1.42	432.27	**0.23**
Clustered at $\pm\varepsilon$	125	0.68	0.49	0.02	0.14	**0.01**
	250	3.36	2.21	0.03	1.03	**0.02**
	500	18.98	0.11	7.26	4.0	**0.03**
	1000	120.89	57.09	0.350	54.68	**0.09**
	2000	846.84	356.07	1.18	416.25	**0.25**
Clustered at 1	125	0.30	0.19	0.01	0.07	**0.04**
	250	3.54	2.72	0.04	0.46	**0.15**
	500	13.56	12.66	0.14	3.96	**0.68**
	1000	100.04	101.29	0.39	27.64	**3.77**
	2000	767.49	952.61	1.75	223.66	**17.05**
Clustered at ± 1	125	0.23	0.14	0.01	0.09	**0.06**
	250	1.27	0.74	0.03	0.60	**0.17**
	500	7.76	4.83	0.12	4.24	**0.70**
	1000	54.03	36.17	0.350	29.98	**2.44**
	2000	398.60	313.47	1.23	225.42	**17.17**
(1,2,1) Matrix	125	0.51	0.46	0.15	0.18	**0.14**
	250	1.82	0.70	0.45	1.18	**0.57**
	500	8.21	7.13	2.69	8.47	**2.33**
	1000	40.16	29.30	18.15	64.31	**7.89**
	2000	857.84	194.14	129.05	491.40	**32.87**
Biphenyl	966	107.72	53.26	19.58	44.25	**10.26**
$SiOSi_6$	966	360.21	172.86	97.49	248.07	**41.61**
Zeolite ZSM-5	2053	281.02	161.28	145.65	376.65	**106.25**

Table 6.2: Timing Results

Matrix Type	Matrix Size	Maximum Residual Norm LAPACK INVIT	EISPACK INVIT	LAPACK D&C	LAPACK QR	Algorithm Y
Uniform Distribution (ε apart)	125	0.005	0.377	0.630	0.004	**0.013**
	250	0.002	0.601	0.627	0.002	**0.002**
	500	0.0001	0.043	0.425	0.0003	**0.0001**
	1000	0.0005	0.728	0.299	0.0005	**0.0005**
	2000	0.00002	0.153	0.276	0.00003	**0.002**
Uniform Distribution ($\sqrt{\varepsilon}$ apart)	125	0.035	1.83	0.057	0.224	**0.130**
	250	0.024	2.60	0.040	0.200	**0.197**
	500	0.030	7.62	0.073	0.358	**0.015**
	1000	0.012	5.94	0.255	0.181	**0.146**
	2000	0.001	11.15	0.012	0.197	**0.089**
Uniform Distribution (ε to 1)	125	0.056	3.31	0.117	0.365	**0.029**
	250	0.039	6.19	0.098	0.375	**0.023**
	500	0.023	5.33	0.139	0.610	**0.025**
	1000	0.023	10.92	0.070	0.410	**0.012**
	2000	0.017	7.19	0.133	0.334	**0.008**
Uniform Distribution (ε to 1 with random signs)	125	0.029	1.96	0.234	0.624	**0.046**
	250	0.034	4.77	0.145	0.586	**0.036**
	500	0.023	5.33	0.139	0.610	**0.025**
	1000	0.019	9.71	0.134	0.589	**0.018**
	2000	0.012	11.74	0.135	0.569	**0.012**
Geometric Distribution (ε to 1)	125	0.008	0.483	0.119	0.048	**0.008**
	250	0.005	0.882	0.084	0.034	**0.006**
	500	0.003	1.11	0.098	0.038	**0.004**
	1000	0.003	1.55	0.063	0.040	**0.003**
	2000	0.002	1.90	0.057	0.040	**0.002**
Geometric Distribution (ε to 1 with random signs)	125	0.006	0.391	0.081	0.041	**0.127**
	250	0.003	0.420	0.065	0.045	**0.213**
	500	0.003	0.902	0.051	0.048	**0.153**
	1000	0.003	1.91	0.041	0.061	**0.417**
	2000	0.002	2.07	0.029	0.055	**0.430**
Random	125	0.016	0.993	0.083	0.373	**0.154**
	250	0.011	1.75	0.087	0.256	**0.149**
	500	0.007	1.79	0.079	0.354	**0.403**
	1000	0.011	1.75	0.087	0.256	**0.168**
	2000	0.005	4.15	0.092	0.383	**0.392**

Table 6.3: Maximum Residual Norms $\equiv \max_i \|T\hat{v}_i - \hat{\lambda}_i \hat{v}_i\| / n\varepsilon\|T\|$

Matrix Type	Matrix Size	Maximum Residual Norm				
		LAPACK INVIT	EISPACK INVIT	LAPACK D&C	LAPACK QR	Algorithm Y
Clustered at ε	125	0.004	0.429	0.011	0.004	**0.006**
	250	0.0005	0.041	0.002	0.004	**0.001**
	500	0.0002	0.115	0.002	0.001	**0.002**
	1000	0.000007	0.125	0.001	0.0005	**0.0006**
	2000	0.0003	0.400	0.0003	0.00004	**0.0003**
Clustered at $\pm\varepsilon$	125	0.004	0.739	0.014	0.006	**0.007**
	250	0.004	0.854	0.007	0.002	**0.002**
	500	0.00003	0.079	0.004	0.001	**0.0003**
	1000	0.0005	0.395	0.002	0.0005	**0.0008**
	2000	0.000002	0.109	0.001	0.0002	**0.0008**
Clustered at 1	125	1.31	1.23	0.156	0.245	**0.251**
	250	1.45	1.14	0.190	0.269	**0.168**
	500	1.46	1.37	0.096	0.308	**0.146**
	1000	1.52	1.35	0.067	0.346	**0.107**
	2000	1.62	1.45	0.036	0.340	**0.091**
Clustered at ± 1	125	0.704	9.53	0.224	0.399	**0.190**
	250	0.712	4.55	0.138	0.611	**0.109**
	500	0.664	17.78	0.074	0.565	**0.103**
	1000	0.604	22.10	0.048	0.563	**0.083**
	2000	0.625	33.95	0.034	0.549	**0.076**
(1,2,1) Matrix	125	0.047	4.54	0.164	0.431	**0.100**
	250	0.034	8.31	0.130	0.411	**0.314**
	500	0.031	21.32	0.109	0.400	**0.459**
	1000	0.035	57.72	0.105	0.484	**0.116**
	2000	0.029	181.37	0.102	0.618	**0.227**
Biphenyl	966	0.0009	0.708	0.010	0.004	**0.0009**
$SiOSi_6$	966	0.004	179.87	0.021	0.054	**0.020**
Zeolite ZSM-5	2053	0.004	16.98	0.0145	0.097	**0.150**

Table 6.4: Maximum Residual Norms $\equiv \max_i \|T\hat{v}_i - \hat{\lambda}_i\hat{v}_i\|/n\varepsilon\|T\|$

Matrix Type	Matrix Size	Maximum Dot Product LAPACK INVIT	EISPACK INVIT	LAPACK D&C	LAPACK QR	Algorithm Y
Uniform Distribution (ε apart)	125	0.056	0.272	0.096	0.204	**0.067**
	250	0.046	0.220	0.032	0.108	**0.120**
	500	0.024	0.216	0.018	0.068	**0.085**
	1000	0.023	0.257	0.017	0.045	**0.084**
	2000	0.017	0.260	0.010	0.032	**0.073**
Uniform Distribution ($\sqrt{\varepsilon}$ apart)	125	0.056	0.200	0.064	0.136	**0.091**
	250	0.044	0.196	0.060	0.108	**0.108**
	500	0.030	0.234	0.036	0.074	**0.084**
	1000	0.022	0.217	0.047	0.059	**0.159**
	2000	0.014	0.230	0.044	0.028	**0.091**
Uniform Distribution (ε to 1)	125	0.048	6.20	0.080	0.184	**0.107**
	250	0.049	25.48	0.056	0.106	**0.062**
	500	0.046	12.60	0.058	0.081	**0.107**
	1000	0.020	61.44	0.048	0.062	**0.118**
	2000	0.015	0.242	0.046	0.047	**0.170**
Uniform Distribution (ε to 1 with random signs)	125	0.119	3.93	0.096	0.104	**0.070**
	250	0.106	8.23	0.052	0.076	**0.114**
	500	0.132	12.67	0.044	0.105	**0.132**
	1000	0.028	17.67	0.047	0.060	**0.140**
	2000	0.028	17.23	0.046	0.025	**0.228**
Geometric Distribution (ε to 1)	125	0.052	0.432	0.096	0.164	**0.060**
	250	0.066	0.308	0.054	0.104	**0.050**
	500	0.036	0.528	0.036	0.102	**0.032**
	1000	0.024	0.360	0.018	0.058	**0.025**
	2000	0.017	0.330	0.014	0.052	**0.020**
Geometric Distribution (ε to 1 with random signs)	125	0.056	0.192	0.088	0.192	**1.77**
	250	0.036	0.248	0.054	0.144	**0.731**
	500	0.035	0.246	0.042	0.140	**0.337**
	1000	0.022	0.244	0.022	0.066	**0.330**
	2000	0.016	0.575	0.020	0.053	**127499.0**
Random	125	0.105	6.03	0.096	0.184	**0.839**
	250	0.095	13.63	0.050	0.112	**0.337**
	500	0.042	10.36	0.050	0.062	**0.254**
	1000	0.044	26.16	0.031	0.112	**0.415**
	2000	0.031	7.30	0.042	0.039	**0.610**

Table 6.5: Maximum Dot Products $\equiv \max_{i \neq j} |\hat{v}_i^T \hat{v}_j| / n\varepsilon$

Matrix Type	Matrix Size	Maximum Dot Product				
		LAPACK INVIT	EISPACK INVIT	LAPACK D&C	LAPACK QR	Algorithm Y
Clustered at ε	125	0.052	0.829	0.056	0.160	**0.008**
	250	0.044	0.605	0.032	0.144	**0.004**
	500	0.034	0.226	0.020	0.088	**0.002**
	1000	0.022	0.626	0.008	0.067	**0.0007**
	2000	0.021	2.78	0.005	0.048	**0.0005**
Clustered at $\pm\varepsilon$	125	0.048	0.705	0.064	0.152	**0.014**
	250	0.032	0.455	0.044	0.120	**0.008**
	500	0.029	0.194	0.022	0.092	**0.003**
	1000	0.032	14.94	0.024	0.082	**0.168**
	2000	0.015	0.157	0.004	0.043	**0.0005**
Clustered at 1	125	0.048	1.26	0.044	0.064	**0.008**
	250	0.036	2.96	0.020	0.062	**0.0008**
	500	0.023	0.738	0.012	0.052	**0.003**
	1000	0.019	3.80	0.006	0.032	**0.001**
	2000	0.015	17720798.0	0.003	0.034	**0.0002**
Clustered at ± 1	125	0.052	3.44	0.048	0.116	**0.020**
	250	0.032	14.94	0.024	0.082	**0.020**
	500	0.022	625.94	0.016	0.068	**0.027**
	1000	0.015	296.76	0.007	0.045	**0.018**
	2000	0.016	63357.9	0.004	0.048	**0.007**
(1,2,1) Matrix	125	0.101	63.24	0.084	0.112	**0.623**
	250	0.064	20.03	0.028	0.068	**0.284**
	500	0.036	15.52	0.030	0.044	**0.308**
	1000	0.072	5.02	0.032	0.035	**0.759**
	2000	0.084	1.13	0.027	0.044	**0.662**
Biphenyl	966	0.018	0.162	0.030	0.073	**0.859**
$SiOSi_6$	966	0.012	70.86	0.033	0.070	**0.698**
Zeolite ZSM-5	2053	0.010	1.41	0.045	0.049	**0.382**

Table 6.6: Maximum Dot Products $\equiv \max_{i\neq j} |\hat{v}_i^T \hat{v}_j|/n\varepsilon$

invoking the ideas of Section 6.3, submatrices of very small size are needed by our new algorithm to compute orthogonal eigenvectors. As a result, Algorithm Y takes as little as 0.25 seconds to find all the eigenpairs of such a matrix of size 2000, while traditional LAPACK inverse iteration takes 840 seconds. All our timing results confirm Algorithm Y to be an $O(n^2)$ method. The new technique of finding the eigenvector of an isolated eigenvalue by using twisted factorizations (see Chapter 3 for details) implies that our new algorithm is faster even in cases that are favorable for EISPACK and LAPACK's inverse iteration, i.e., matrices of type 4 and type 7.

We mentioned the uncertainty in implementing certain steps of Algorithm Y at the beginning of this chapter. The reason is the lack of a complete theory behind multiple representations. This uncertainty has led to some redundancy and less than satisfactory fixes (such as a tolerance on the acceptable element growth) in our preliminary code which will hopefully be eliminated in the near future. In addition, whenever we form a relatively robust representation we need to recompute the locally small eigenvalues to high relative accuracy. Thus, even though it is still an $O(n^2)$ process, our code slows down somewhat when it needs to form many representations. Due to these reasons, the matrix with eigenvalues arranged in geometric order from ε to 1 (with random signs) is the hardest one for Algorithm Y, as evinced by the time needed and an occasionally larger than acceptable deviation from orthogonality. In contrast, Algorithm Y performs very well on a similar type of matrix where the eigenvalues are geometrically distributed from ε to 1 (without any random signs attached to them). The upcoming Section 6.5 lists some possible algorithmic improvements that should remove this discrepancy in performance on these similar types of matrices. Tables 6.3-6.6 show that the residual norms and orthogonality of the eigenpairs computed by our current implementation of Algorithm Y are generally, very good.

The divide and conquer method is the fastest among all the earlier algorithms and on examples where eigenvalues are close, it outperforms our preliminary implementation of Algorithm Y. This success is due to the deflation process, where an eigenpair of a submatrix is found to be an acceptable eigenpair of the full matrix. However, in cases where the eigenvalues are well-separated, the divide and conquer algorithm takes $O(n^3)$ time and is slower than Algorithm Y. One major drawback of LAPACK D&C is its extra workspace requirement of more than $2n^2$ double precision words, which can be prohibitively excessive. In fact, in the recent past, some people have turned to LAPACK INVIT instead of divide and conquer to solve their large problems solely for this reason [6, 53, 52]. Of course, workspace of only about 5-10 n double precision words is required by Algorithm Y, and EISPACK and LAPACK INVIT.

The QR method is seen to uniformly take $O(n^3)$ time and is less sensitive to the eigenvalue distribution. It is quite competitive with the other methods on matrices of small size. Both the QR method and the divide and conquer algorithm produce accurate eigenpairs, as seen from Tables 6.3, 6.4, 6.5 and 6.6.

We now discuss performance of the various algorithms on matrices that arise from quantum chemistry applications. Table 6.2 shows that our Algorithm Y is the fastest method for these matrices. In the results discussed above, the matrices were characterized by their eigenvalue distributions. In Figures 6.1, 6.2 and 6.3, we did give the eigenvalues of the three matrices under consideration. However, the quantities of interest are the absolute and relative gaps between the eigenvalues. The left halves of Figures 6.4, 6.5 and 6.6 plot the logarithms of absolute gaps, i.e.,

$$\log_{10}\left(\frac{\mathrm{absgap}(i)}{\|T\|}\right) = \log_{10}\left(\frac{\min(\lambda_{i+1}-\lambda_i, \lambda_i-\lambda_{i-1})}{\|T\|}\right), \tag{6.4.5}$$

while the right halves plot the relative gaps on a logarithmic scale, i.e.,

$$\log_{10}(\mathrm{relgap}(i)) = \log_{10}\left(\frac{\min(\lambda_{i+1}-\lambda_i, \lambda_i-\lambda_{i-1})}{|\lambda_i|}\right). \tag{6.4.6}$$

Recall that in (6.4.5) and (6.4.6), the eigenvalues are arranged in ascending order.

From Figure 6.4, we see that almost all the absolute gaps for the biphenyl matrix are less than $10^{-3}\|T\|$ and consequently, both LAPACK and EISPACK INVIT spend $O(n^3)$ time on this matrix of order 966. Most of the eigenvalues, however, agree in less than 3 leading digits and as a result, Algorithm Y is about 10 times faster than LAPACK INVIT. This translates to a 3-fold increase in speed of the total dense symmetric eigenproblem. Similar behavior is observed in the $SiOSi_6$ matrix. We see a different distribution in the Zeolite example where the relative gaps are similar to the absolute gaps. Despite the need to form many representations, we observe that Algorithm Y is still faster than the earlier methods.

We have also implemented a parallel variant of Algorithm Y in collaboration with computational chemists at Pacific Northwest National Laboratories (PNNL). This will replace the earlier tridiagonal eigensolver based on inverse iteration that was part of the PeIGS version 2.0 library [52]. Our new method is more easily parallelized, and coupled with its lower operation count, offers an even bigger computational advantage on a parallel

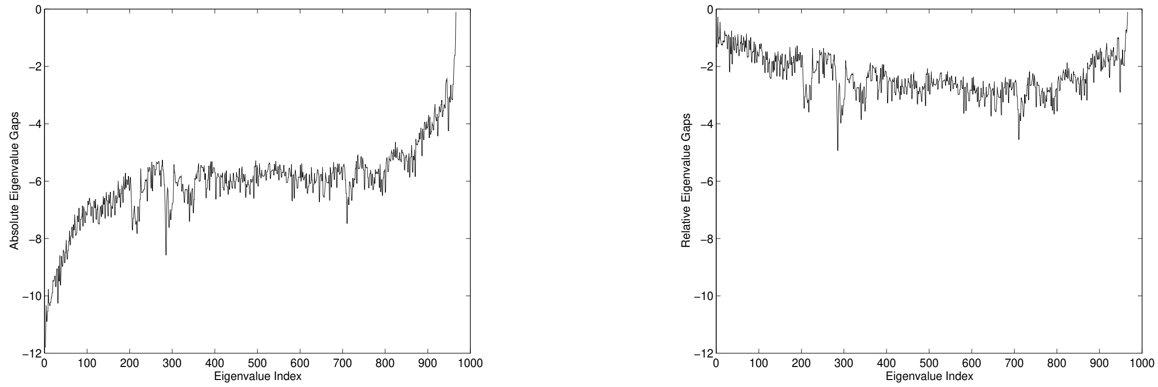

Figure 6.4: Absolute and Relative Eigenvalue Gaps for Biphenyl

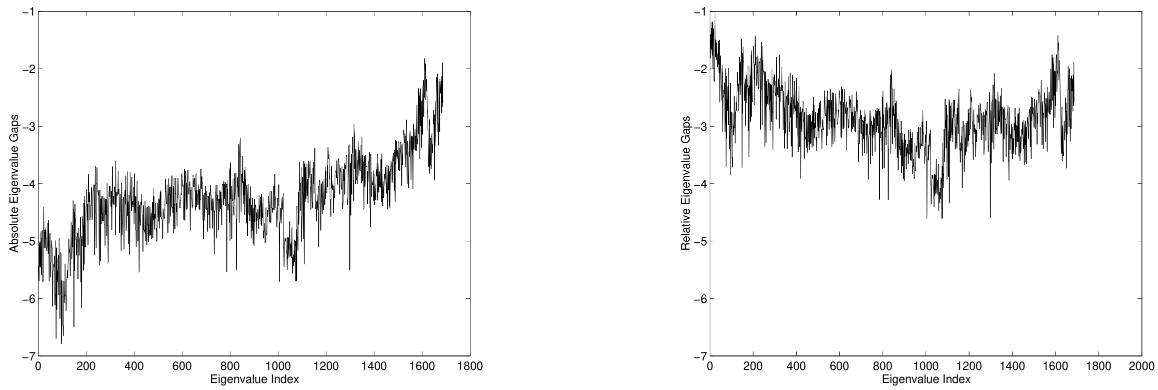

Figure 6.5: Absolute and Relative Eigenvalue Gaps for $SiOSi_6$

computer. For example, on the biphenly matrix our new parallel implementation is 100 times faster on a 128 processor IBM SP2 machine. More performance results are given in [39].

6.5 Future Enhancements to Algorithm Y

We now list various ways in which our current implementation of Algorithm Y may be improved.

Multiple Representations. We would like to complete the relative perturbation theory for factorizations of indefinite matrices. Some preliminary results and conjectures were presented in Chapter 5. We plan to investigate using twisted factorizations as relatively robust representations. We believe that completing this theory will lead to

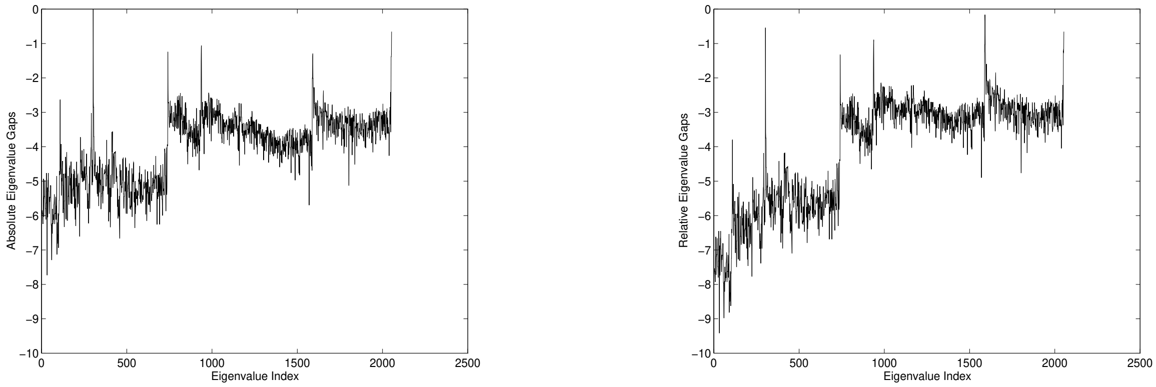

Figure 6.6: Absolute and Relative Eigenvalue Gaps for Zeolite ZSM-5

a more elegant and efficient implementation.

Choice of Base Representation. In both Algorithms X and Y, the first step is to form a *base representation* by translating the matrix and making it positive definite. We do so because of the well known relative perturbation properties of the resulting bidiagonal Cholesky factor. However, we could instead form a relatively robust factorization of an indefinite matrix. As seen from the RRR trees in Figures 5.1 and 5.2, a judicious choice of the base representation may save us from forming multiple representations and hence, result in greater speed. For example, by forming a base representation near zero, we may be able to substantially reduce the amount of time spent by Algorithm Y on matrices with eigenvalues that are geometrically distributed from ε to 1 (with random signs). In the near future, we hope to incorporate a measure of "goodness" with each shift at which we can form a relatively robust representation, and start with the "best" possible base representation.

Exploiting sparse eigenvectors. As seen in Figure 4.6, a majority of the entries in an eigenvector may be negligible (this figure actually plots an eigenvector of the biphenyl matrix). We can make our implementation more efficient by setting these negligible entries to zero, in an *a priori* manner. In fact, the submatrix ideas of Section 6.3 do produce sparse eigenvectors for clusters, and can greatly reduce the amount of work needed as seen in the previous section. Even isolated eigenvalues can have sparse eigenvectors, and we intend to exploit this sparsity and also push the submatrix ideas further to get a more efficient code. We believe that by doing so, we can get an algorithm that is uniformly faster than the divide & conquer method.

Exploiting Level-2 BLAS-like operations. Different eigenvectors are computed independently of each other by Algorithm Y. By computing multiple eigenvectors at the same time, we would be able to vectorize the arithmetic operations, and exploit multiple functional units in modern computer architectures such as the IBM RS/6000 processor. Such a strategy has been adopted to speed up the LAPACK implementation of bisection on some computer architectures [107, 1]. We say that such a strategy leads to a level-2 BLAS-like operation since it involves a quadratic amount of data and a quadratic amount of work [67].

We hope to resolve the above mentioned algorithmic enhancements in the near future. These should result in Algorithm Z, and be included in future releases of LAPACK [1] and and ScaLAPACK [22].

Finally, we remind the reader that the timing and accuracy results presented in this chapter must be considered to be transitory. Future improvements could lead to further increase in speed of our new $O(n^2)$ algorithm. All the earlier algorithms to which we compare our new methods have been researched for more than 15 years, and we ask the kind reader for some patience as we try to produce an algorithm that is provably correct in floating point arithmetic while simultaneously being (i) $O(n^2)$ and (ii) embarrassingly parallel.

Chapter 7

Conclusions

In this thesis, we have shown how to compute numerically orthogonal approximations to eigenvectors without resorting to any reorthogonalization technique such as the Gram-Schmidt process. Consequently, our new algorithm can compute orthogonal eigenvectors in $O(n^2)$ time. The individual eigenvectors are computed independently of each other which is well-suited for parallel computation. Moreover, our algorithm can deliver any subset of the eigenvalues and eigenvectors at a cost of $O(n)$ operations per eigenpair.

We now discuss some of the guiding principles behind the advances mentioned above. We feel that these principles are general in the sense that they could be applied to other numerical procedures. In the following, we list these principles and show how we have applied them to get a faster $O(n^2)$ solution to the tridiagonal eigenproblem.

Do not try to compute a quantity that is not well-determined. When eigenvalues are close, individual eigenvectors are extremely sensitive to small changes in the matrix entries. On the other hand, the invariant subspace corresponding to these close eigenvalues is well-determined and any orthogonal basis of this subspace will suffice as approximate eigenvectors. Earlier EISPACK and LAPACK software attempt to compute such a (non-unique) basis by explicit orthogonalization and hence, can take $O(n^3)$ time. However, as explained in Chapter 5, we have been able to find alternate representations of the initial matrix so that the refined eigenvalues are no longer "close" and the corresponding eigenvectors are now well-determined. These new representations allow us to identify a robust orthogonal basis of the desired subspace and hence, compute it cheaply. Another major advance in our methods comes by recognizing that the bidiagonal factors of a tridiagonal determine the desired quantities much "better" than its diagonal and off-diagonal entries.

Exploiting transformations for which the solution is invariant. Eigenvalues of a matrix A are invariant under the similarity transformations $S^{-1}AS$, and change in a simple way under affine transformations of the form $\alpha A + \beta$, $\alpha \neq 0$. Several methods such as the QR, LR and **qd** algorithms take advantage of such transformations. Eigenvectors of A are easily seen to be invariant under any affine transformation. We exploit this property to great advantage by obtaining various representations of the given problem, each of which is "better" suited for computing a subset of the spectrum.

Finding "good" representations. We need to make sure that the above transformations do not affect the accuracy of the desired quantities. Thus to produce eigenpairs that have small residual norms, as specified in (1.1.1), we need to avoid large element growths in forming the intermediate factorizations

$$T - \mu I \;=\; LDL^T. \tag{7.0.1}$$

See Chapter 5 for details.

High Accuracy in key computations can lead to speed. The main reason for forming multiple representations, as in (7.0.1), is to avoid Gram-Schmidt like methods in computing orthogonal approximations to the eigenvectors. Our faster $O(n^2)$ scheme becomes possible only because the intermediate representations determine their small eigenvalues to high relative accuracy. We can then compute these small eigenvalues to full relative accuracy and use these highly accurate eigenvalues to compute eigenvectors. The resulting eigenvectors we compute are "faithful" and *orthogonal as a consequence*. In numerical methods, there is generally believed to be a trade-off between speed and accuracy, i.e., higher accuracy can be achieved only at the expense of computing time. *However, we have demonstrated that high accuracy in the right places can actually speed up the computation.*

7.1 Future Work

In Section 6.5, we mentioned further enhancements to Algorithm Y which we intend to incorporate in the near future. Here we see how some of our ideas may be applied to other problems in numerical linear algebra.

Perfect Shifts. In Section 3.5.1 we showed how to effectively use the perfect shift strategy in a QR-like algorithm. This new scheme enables us to use "ultimate" or "perfect" shifts as envisaged by Parlett in [110]. It also solves the problem of immediately deflating a known eigenpair from a symmetric matrix by deleting one of its rows and columns. We intend to substantiate these claims with numerical experiments in the near future [41].

Non-symmetric Eigenproblem. Since Algorithm Y does not invoke any Gram-Schmidt like process, it does not explicitly try to compute eigenvectors that are orthogonal. Orthogonality is a result of the matrix being symmetric. We believe that many of our ideas can be applied to the non-symmetric eigenproblem in order to obtain the "best-conditioned" eigenvectors.

Lanczos Algorithm. The method proposed by Lanczos in 1952 [97], after many modifications to account for roundoff [125, 110, 126], has become the champion among algorithms to find some of the extreme eigenvalues of a sparse symmetric matrix. It proceeds by incrementally forming, one row and column at a time, a tridiagonal matrix that is similar to the sparse matrix. We would like to investigate if there are any benefits in using the LDL^T decomposition of this tridiagonal or its translates. The sparsity of the eigenvectors of the tridiagonal, see Figure 4.6 for an example, may also be exploited in reducing the amount of work in the selective orthogonalization phase. See [125] for details on the latter phase.

Rank-Revealing Factorizations. The twisted factorizations introduced in Section 3.1 enable us to accurately compute a null vector of a nearly singular tridiagonal matrix. The particular twisted factorization used is successful because it transparently reveals the near singularity of the tridiagonal. As discussed in Section 3.6, twisted factorizations can also be formed for denser matrices. A nice property is that they tend to preserve the sparsity pattern of the matrix. Coupled with their rank-revealing properties, twisted factorizations may become an invaluable computational tool in banded (sparse) matrix computations.

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide (second edition)*. SIAM, Philadelphia, 1995. 324 pages.

[2] ANSI/IEEE, New York. *IEEE Standard for Binary Floating Point Arithmetic*, Std 754-1985 edition, 1985.

[3] P. Arbenz, K. Gates, and Ch. Sprenger. A parallel implementation of the symmetric tridiagonal QR algorithm. In *Frontier's 92*. McLean, 1992.

[4] S. O. Asplund. Finite boundary value problems solved by Green's matrix. *Math. Scand.*, 7:49–56, 1959.

[5] I. Babuska. Numerical stability in problems of linear algebra. *SIAM J. Num. Anal.*, 9:53–77, 1972.

[6] Z. Bai, 1996. private communication.

[7] V. Bargmann, D. Montgomery, and J. von Neumann. Solution of linear systems of high order. Report prepared for Navy Bureau of Ordnance, 1946. Reprinted in [131, pp. 421–477].

[8] J. Barlow, 1996. private communication.

[9] J. Barlow and J. Demmel. Computing accurate eigensystems of scaled diagonally dominant matrices. *SIAM J. Num. Anal.*, 27(3):762–791, June 1990.

[10] D. Bernholdt and R. Harrison. Orbital invariant second order many-body perturbation on parallel computers: An approach for large molecules. *J. Chem. Physics.*, 1995.

[11] H. J. Bernstein and M. Goldstein. Parallel implementation of bisection for the calculation of eigenvalues of a tridiagonal symmetric matrices. *Computing*, 37:85–91, 1986.

[12] Christian H. Bischof and Charles F. Van Loan. The WY representation for products of Householder matrices. *SIAM J. Sci. Stat. Comput.*, 8(1):s2–s13, 1987.

[13] R. P. Brent. *Algorithms for minimization without derivatives*. Prentice-Hall, 1973.

[14] B. Bukhberger and G.A. Emel'yanenko. Methods of inverting tridiagonal matrices. *USSR Computat. Math. and Math. Phy.*, 13:10–20, 1973.

[15] J. Bunch, P. Nielsen, and D. Sorensen. Rank-one modification of the symmetric eigenproblem. *Num. Math.*, 31:31–48, 1978.

[16] J. Carrier, L. Greengard, and V. Rokhlin. A fast adaptive multipole algorithm for particle simulations. *SIAM, J. Sci. Stat. Comput.*, 9(4):669–686, July 1988.

[17] S. Chakrabarti, J. Demmel, and K. Yelick. Modeling the benefits of mixed data and task parallelism. In *Symposium on Parallel Algorithms and Architectures (SPAA)*, Santa Barbara, California, july 1995.

[18] T. Chan. Rank revealing QR factorizations. *Lin. Alg. Appl.*, 88/89:67–82, 1987.

[19] T. F. Chan. On the existence and computation of LU-factorizations with small pivots. *Math. Comp.*, 42:535–547, 1984.

[20] S. Chandrasekaran. *When is a Linear System Ill-Conditioned?* PhD thesis, Departement of Computer Science, Yale University, New Haven, CT, 1994.

[21] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, Benton L. Leong, Michael B. Monagan, and Stephen M. Watt. *Maple V Library Reference Manual.* Springer-Verlag, Berlin, 1991.

[22] J. Choi, J. Demmel, I. Dhillon, J. Dongarra, Ostrouchov, S., A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. ScaLAPACK, a portable linear algebra library for distributed memory computers-design issues and performance. *Computer Physics Communications*, 97(1-2):1–15, 1996.

[23] M. Chu. A simple application of the homotopy method to symmetric eigenvalue problems. *Lin. Alg. Appl*, 59:85–90, 1984.

[24] M. Chu. A note on the homotopy method for linear algebraic eigenvalue problems. *Lin. Alg. Appl*, 105:225–236, 1988.

[25] J.J.M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36:177–195, 1981.

[26] C. Davis and W. Kahan. Some new bounds on perturbation of subspaces. *Bulletin of the American Mathematical Society*, 77(4):863–868, July 1969.

[27] C. Davis and W. Kahan. The rotation of eigenvectors by a perturbation III. *SIAM J. Num. Anal.*, 7:248–263, 1970.

[28] D. Day. The differential qd algorithm for the tridiagonal eigenvalue problem. 1997. in preparation.

[29] P. Deift, J. Demmel, L.-C. Li, and C. Tomei. The bidiagonal singular values decomposition and Hamiltonian mechanics. *SIAM J. Num. Anal.*, 28(5):1463–1516, October 1991. (LAPACK Working Note #11).

[30] L. S. DeJong. Towards a formal definition of numerical stability. *Numer. Math.*, 28:211–220, 1977.

[31] T. J. Dekker. Finding a zero by means of successive linear interpolation. In B. Dejon and P. Henrici, editors, *Constructive aspects of the fundamanetal theorem of algebra*. New York: Wiley-Interscience, 1969.

[32] J. Demmel. The inherent inaccuracy of implicit tridiagonal QR. Technical Report Report 963, IMA, University of Minnesota, April 1992.

[33] J. Demmel and W. Gragg. On computing accurate singular values and eigenvalues of acyclic matrices. *Lin. Alg. Appl.*, 185:203–218, 1993.

[34] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač. Computing the singular value decomposition with high relative accuracy. Computer Science Dept. Technical Report CS-97-348, University of Tennessee, Knoxville, February 1997. (LAPACK Working Note #119, available electronically on netlib).

[35] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Stat. Comput.*, 11(5):873–912, September 1990.

[36] J. Demmel and A. McKenney. A test matrix generation suite. Computer science dept. technical report, Courant Institute, New York, NY, July 1989. (LAPACK Working Note #9).

[37] J. Demmel and K. Veselić. Jacobi's method is more accurate than QR. *SIAM J. Mat. Anal. Appl.*, 13(4):1204–1246, 1992. (also LAPACK Working Note #15).

[38] J. W. Demmel. personal communication, 1996.

[39] I. Dhillon, G. Fann, and B. Parlett. Application of a new algorithm for the symmetric eigenproblem to computational quantum chemistry. In *Proceedings of the Eighth SIAM Conference on Parallel Proceesing for Scientific Computing*. SIAM, March 1997.

[40] I. S. Dhillon. Current inverse iteration software can fail. 1997. Submitted for publication.

[41] I. S. Dhillon. Perfect shifts and twisted factorizations. 1997. In preparation.

[42] I. S. Dhillon. Stable computation of the condition number of a tridiagonal matrix in $O(n)$ time. 1997. Submitted for publication.

[43] I. S. Dhillon and B.N. Parlett. Orthogonal eigenvectors without Gram-Schmidt. 1997. in preparation.

[44] J. Dongarra, J. Bunch, C. Moler, and G. W. Stewart. *LINPACK User's Guide.* SIAM, Philadelphia, PA, 1979.

[45] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling. A set of Level 3 Basic Linear Algebra Subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.

[46] J. Dongarra and D. Sorensen. A fully parallel algorithm for the symmetric eigenproblem. *SIAM J. Sci. Stat. Comput.*, 8(2):139–154, March 1987.

[47] J. Dongarra, R. van de Geijn, and D. Walker. A look at scalable dense linear algebra libraries. In *Scalable High-Performance Computing Conference.* IEEE Computer Society Press, April 1992.

[48] J. DuCroz, December 1994. private communication.

[49] S. Eisenstat and I. Ipsen. Relative perturbation techniques for singular value problems. *SIAM J. Numer. Anal.*, 32(6), 1995.

[50] S. C. Eisenstat and I. C. F. Ipsen. Relative perturbation bounds for eigenspaces and singular vector subspaces. In J. G. Lewis, editor, *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra*, pages 62–65. SIAM, 1994.

[51] S.C. Eisenstat and I.C.F. Ipsen. Relative perturbation results for eigenvalues and eigenvectors of diagonalisable matrices. Technical Report CRSC-TR96-6, Center for Research in Scientific Computation, Department of Mathematics, North Carolina State University, 1996. (14 pages).

[52] D. Elwood, G. Fann, and D. Littlefield. *PeIGS User's Manual.* Pacific Northwest National Laboratory, Richland, WA, 1993.

[53] G.I. Fann, 1996. private communication.

[54] G.I. Fann and R. J. Littlefield. Parallel inverse iteration with re-orthogonalization. In R. Sincovec et. al., editor, *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing*, volume 1, pages 409–413. SIAM, 1993.

[55] G.I. Fann, R. J. Littlefield, and D.M. Elmwood. Performance of a fully parallel dense real symmetric eigensolver in quantum chemistry appliacation. In *Proceedings of the High Performance Computing '95, Simulation MultiConference.* The Society of Computer Simulation, San Diego, 1995.

[56] K. Fernando and B. Parlett. Accurate singular values and differential qd algorithms. *Numerische Mathematik*, 67:191–229, 1994.

[57] K. V. Fernando. On computing an eigenvector of a tridiagonal matrix. Technical Report TR4/95, Nag Ltd., Oxford, UK, 1995. submitted for publication.

[58] D. Fischer, G. H. Golub, O. Hald, C. Leiva, and O. Widlund. On Fourier Toeplitz methods for separable elliptic problems. *Math. Comp.*, 28:349–368, 1974.

[59] G. J. F. Francis. The QR transformation, parts I and II. *Computer J.*, 4:265–271,332–345, 1961-62.

[60] K. E. Gates. Using inverse iteration to improve the divide and conquer algorithm. Technical Report 159, ETH Department Informatik, May 1991.

[61] W. Givens. The characteristic value-vector problem. *J. ACM*, 4:298–307, 1957. also unpublished report.

[62] Wallace J. Givens. Numerical computation of the characteristic values of a real symmetric matrix. Technical Report ORNL-1574, Oak Ridge National Laboratory, Oak Ridge, TN, USA, 1954.

[63] S. K. Godunov, A. G. Antonov, O. P. Kiriljuk, and V. I. Kostin. *Guaranted Accuracy in Numerical Linear Algebra.* Kluwer Academic Publishers, Dordrecht, Netherlands, 1993. A revised translation of a Russian text first published in 1988 in Novosibirsk.

[64] S. K. Godunov, V. I. Kostin, and A. D. Mitchenko. Computation of an eigenvector of symmetric tridiagonal matrices. *Siberian Math. J.*, 26:71–85, 1985.

[65] D. Goldberg. What every computer scientist should know about floating point arithmetic. *ACM Computing Surveys*, 23(1), 1991.

[66] G. H. Golub, V. Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Technical Report STAN-CS-76-559, Computer Science Dept., Stanford Univ., 1976.

[67] Gene H. Golub and Charles F. Van Loan. *Matrix computations.* Johns Hopkins Studies in the Mathematical Sciences. The Johns Hopkins University Press, Baltimore, MD, USA, third edition, 1996.

[68] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science.* Addison-Wesley, Reading, MA, USA, 1989.

[69] A. Greenbaum and J. Dongarra. Experiments with QL/QR methods for the symmetric tridiagonal eigenproblem. Computer Science Dept. Technical Report CS-89-92, University of Tennessee, Knoxville, 1989. (LAPACK Working Note #17, available electronically on netlib).

[70] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comp. Phys.*, 73:325–348, 1987.

[71] M. Gu. Studies in numerical linear algebra. Ph.D. thesis, 1993.

[72] M. Gu and S. C. Eisenstat. A stable and efficient algorithm for the rank-1 modification of the symmetric eigenproblem. *SIAM J. Mat. Anal. Appl.*, 15(4):1266–1276, October 1994. Yale Tech report YALEU/DCS/RR-916, Sept 1992.

[73] M. Gu and S. C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Mat. Anal. Appl.*, 16(1):172–191, January 1995.

[74] F. G. Gustavson and A. Gupta. A new parallel algorithm for tridiagonal symmetric positive definite systems of equations. unpublished report, 1996.

[75] B. Hendrickson, E. Jessup, and C. Smith. A parallel eigensolver for dense symmetric matrices. Technical Report SAND96–0822, Sandia National Labs, Albuquerque, NM, March 1996.

[76] P. Henrici. The quotient-difference algorithm. *Nat. Bur. Standards Appl. Math. Series*, 19:23–46, 1958.

[77] P. Henrici. Bounds for eigenvalues of certain tridiagonal matrices. *SIAM J.*, 11:281–290, 1963.

[78] N. J. Higham. Efficient algorithms for computing the condition number of a tridiagonal matrix. *SIAM J. Sci. Stat. Comput.*, 7:150–165, 1986.

[79] Hoffman and B. N. Parlett. A new proof of global convergence for the tridiagonal QL algorithm. *SIAM J. Num. Anal.*, 15:929–937, 1978.

[80] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, 1985. Includes Ostrowski's relative perturbation theory on pp 224-225.

[81] Alston S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comput. Mach.*, 5:339–342, 1958.

[82] IBM. *Engineering and Scientific Subroutine Library, Guide and Reference, Release 3, Program 5668-863*, 4 edition, 1988.

[83] I. Ikebe. On inverses of Hessenberg matrices. *Linear Algebra and its Applications*, 24:93–97, 1979.

[84] I. Ipsen and E. Jessup. Solving the symmetric tridiagonal eigenvalue problem on the hypercube. *SIAM J. Sci. Stat. Comput.*, 11(2):203–230, 1990.

[85] I. C. F. Ipsen. Computing an eigenvector with inverse iteration. *SIAM Review*, 1997. to appear.

[86] C. G. F. Jacobi. Concerning an easy process for solving equations occurring in the theory of secular disturbances. *J. Reine Angnew. Math.*, 30:51–94, 1846.

[87] E. Jessup and I. Ipsen. Improving the accuracy of inverse iteration. *SIAM J. Sci. Stat. Comput.*, 13(2):550–572, 1992.

[88] E. R. Jessup. *Parallel Solution of the Symmetric Tridiagonal Eigenproblem.* PhD thesis, Yale University, New Haven, CT, 1989.

[89] W. Kahan. Accurate eigenvalues of a symmetric tridiagonal matrix. Computer Science Dept. Technical Report CS41, Stanford University, Stanford, CA, July 1966 (revised June 1968).

[90] W. Kahan. Notes on Laguerre's iteration. University of California Computer Science Division preprint, 1992.

[91] W. Kahan. Lecture notes on the status of IEEE standard 754 for binary floating point arithmetic. http://HTTP.CS.Berkeley.EDU/ wkahan/ieee754status/ieee754.ps, 1995.

[92] W. Kahan, 1996. private communication.

[93] L. Kaufman. A parallel QR algorithm for the symmetric tridiagonal eigenvalue problem. *J. of Parallel and Distributed Computing*, 23(3):429–434, December 1994.

[94] D. Kershaw. Inequalities on elements of the inverse of a certain tridiagonal matrix. *Math. Comp.*, 24:155–158, 1970.

[95] V. N. Kublanovskaya. On some algorithms for the solution of the complete eigenvalue problem. *Zh. Vych. Mat.*, 1:555–570, 1961.

[96] D. Kuck and A. Sameh. A parallel QR algorithm for symmetric tridiagonal matrices. *IEEE Trans. Computers*, C-26(2), 1977.

[97] C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. Res. Natl. Bur. Stand*, 49:33–53, 1952.

[98] J. Le and B. Parlett. Forward instability of tridiagonal QR. *SIAM J. Mat. Anal. Appl.*, 14(1):279–316, January 1993.

[99] K. Li and T.-Y. Li. An algorithm for symmetric tridiagonal eigenproblems — divide and conquer with homotopy continuation. *SIAM J. Sci. Comp.*, 14(3), May 1993.

[100] R.-C. Li. Relative perturbation theory: (I) eigenvalue and singular value variations. Technical Report UCB//CSD-94-855, Computer Science Division, Department of EECS, University of California at Berkeley, 1994. (revised January 1996).

[101] R.-C. Li. Relative perturbation theory: (II) eigenspace and singular subspace variations. Technical Rep University of California at Berkeley UCB//CSD-94-856, Computer Science Division, Department of EECS, University of California at Berkeley, 1994. (revised January 1996 and April 1996).

[102] R.-C. Li. Solving secular equations stably and efficiently. Computer Science Dept. Technical Report CS-94-260, University of Tennessee, Knoxville, November 1994. (LAPACK Working Note #89).

[103] T.-Y. Li and N. H. Rhee. Homotopy algorithm for symmetric eigenvalue problems. *Num. Math.*, 55:265–280, 1989.

[104] T. Y. Li and Z. Zeng. The Laguerre iteration in solving the symmetric tridiagonal eigenproblem, revisited. *SIAM J. Sci. Stat. Comput.*, 15(5):1145–1173, September 1994.

[105] T.-Y. Li, H. Zhang, and X. H. Sun. Parallel homotopy algorithm for symmetric tridiagonal eigenvalue problems. *SIAM J. Sci. Stat. Comput.*, 12:464–485, 1991.

[106] S.-S. Lo, B. Phillipe, and A. Sameh. A multiprocessor algorithm for the symmetric eigenproblem. *SIAM J. Sci. Stat. Comput.*, 8(2):155–165, March 1987.

[107] A. McKenney, 1997. private communication.

[108] G. Meurant. A review on the inverse of symmetric tridiagonal and block tridiagonal matrices. *SIAM J. Mat. Anal. Appl.*, 13(3):707–728, July 1992.

[109] J. M. Ortega and H. F. Kaiser. The LL^T and QR Methods for Symmetric Tridiagonal Matrices. *Numer. Math.*, 5:211–225, 1963.

[110] B. Parlett. *The Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, NJ, 1980. A new SIAM edition is under preparation.

[111] B. Parlett. *Acta Numerica*, chapter The new qd algorithms, pages 459–491. Cambridge University Press, 1995.

[112] B. Parlett. The construction of orthogonal eigenvectors for tight clusters by use of submatrices. Center for Pure and Applied Mathematics PAM-664, University of California, Berkeley, CA, January 1996. submitted to SIMAX.

[113] B. N. Parlett. Laguerre's Method Applied to the Matrix Eigenvalue Problem. *Math. Comp.*, 18:464–485, 1964.

[114] B. N. Parlett. The rewards for maintaining semi-orthogonality among Lanczos vectors. *Journal of Numerical Linear Algebra with Applications*, 1(2):243–267, 1992.

[115] B. N. Parlett. Invariant subspaces for tightly clustered eigenvalues of tridiagonals. *BIT*, 36(3):542–562, 1996.

[116] B. N. Parlett. Spectral sensitivity of products of bidiagonals. *Linear Algebra and its Applications*, 1996. submitted for inclusion in the proceedings of the 6th ILAS conference held in Chemnitz, Germany in August 1996.

[117] B.N. Parlett and I.S. Dhillon. Fernando's solution to Wilkinson's problem: an application of double factorization. *Linear Algebra and its Applications*, 1997. to appear.

[118] G. Peters and J.H. Wilkinson. *The calculation of specified eigenvectors by inverse iteration, contribution II/18*, volume II of *Handbook of Automatic Computation*, pages 418–439. Springer-Verlag, New York, Heidelberg, Berlin, 1971.

[119] G. Peters and J.H. Wilkinson. Inverse iteration, ill-conditioned equations and Newton's method. *SIAM Review*, 21:339–360, 1979.

[120] D. Priest. Algorithms for arbitrary precision floating point arithmetic. In P. Kornerup and D. Matula, editors, *Proceedings of the 10th Symposium on Computer Arithmetic*, pages 132–145, Grenoble, France, June 26-28 1991. IEEE Computer Society Press.

[121] H. Rutishauser. Der Quotienten-Differenzen-Algorithmus. *Z. Angew. Math. Phys.*, 5:223–251, 1954.

[122] H. Rutishauser. *Vorlesungen über numerische Mathematik.* Birkhäuser, Basel, 1976.

[123] H. Rutishauser. *Lectures on Numerical Mathematics.* Birkhäuser, Boston, 1990.

[124] J. Rutter. A serial implementation of Cuppen's divide and conquer algorithm for the symmetric eigenvalue problem. Mathematics Dept. Master's Thesis available by anonymous ftp to tr-ftp.cs.berkeley.edu, directory pub/tech-reports/csd/csd-94-799, file all.ps, University of California, 1994.

[125] D. Scott. *Analysis of the Symmetric Lanczos Algorithm.* PhD thesis, University of California, Berkeley, California, 1978.

[126] H. Simon. The Lanczos algorithm with partial reorthogonalization. *Math. Comp.*, 42(165):115–142, January 1984.

[127] H. Simon. Bisection is not optimal on vector processors. *SIAM J. Sci. Stat. Comput.*, 10(1):205–209, January 1989.

[128] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines – EISPACK Guide*, volume 6 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1976.

[129] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, New York, 1973.

[130] W. G. Strang. Implicit difference methods for initial boundary value problems. *J. Math. Anal. Appl.*, 16:188–198, 1966.

[131] A. H. Taub, editor. *John von Neumann Collected Works*, volume V, Design of Computers, Theory of Automata and Numerical Analysis. Pergamon, Oxford, 1963.

[132] R. van de Geijn. Deferred shifting schemes for parallel QR methods. *SIAM J. Mat. Anal. Appl.*, 14(1):180–194, 1993.

[133] H. A. van der Vorst. Analysis of a parallel solution method for tridiagonal linear systems. *Parallel Computing*, 5:303–311, 1987.

[134] J. H. Wilkinson. The calculation of the eigenvectors of codiagonal matrices. *Computer J.*, 1:90–96, 1958.

[135] J. H. Wilkinson. *Rounding Errors in Algebraic Processes.* Prentice Hall, Englewood Cliffs, 1963.

[136] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford University Press, Oxford, 1965.

[137] Stephen Wolfram. *Mathematica: A System for Doing Mathematics by Computer.* Addison-Wesley, Reading, MA, USA, second edition, 1991.

[138] Y. Yang. *Backward error analysis for dqds.* PhD thesis, University of California, Berkeley, California, 1994.

Case Studies

The following case studies present examples that illustrate several important aspects of finding eigenvectors of a symmetric tridiagonal matrix. Much of the upcoming material appears in the main text of this thesis. However, we feel that these examples offer great insight to the problem under scrutiny, and so we have chosen to present them in greater detail now. The raw numbers given in the examples may also reveal more than what we have been able to extract! We have tried to make these case studies independent of the main text so the reader should be able to follow them without having to read the whole thesis. Of course, for more theoretical details the reader should read the main text.

All the analytical expressions in the examples to follow were obtained by using the symbol manipulators, Maple [21] and Mathematica [137].

Case Study A

The need for accurate eigenvalues

This example demonstrates the need for high accuracy in the computed eigenvalues in order for inverse iteration to succeed. We will show how LAPACK and EISPACK implementations fail in the absence of such accuracy. Let

$$T = \begin{bmatrix} 1 & \sqrt{\varepsilon} & 0 \\ \sqrt{\varepsilon} & 7\varepsilon/4 & \varepsilon/4 \\ 0 & \varepsilon/4 & 3\varepsilon/4 \end{bmatrix} \tag{A.0.1}$$

where $\varepsilon \ll 1$ and is of the order of the machine precision. The eigenvalues of T are :

$$\begin{aligned} \lambda_1 &= \varepsilon/2 + O(\varepsilon^2), \\ \lambda_2 &= \varepsilon + O(\varepsilon^2), \\ \lambda_3 &= 1 + \varepsilon + O(\varepsilon^2). \end{aligned}$$

Let v_1, v_2, v_3 denote the corresponding eigenvectors with $||\cdot||_2 = 1$.

$$v_1 = \begin{bmatrix} -\sqrt{\varepsilon/2} + O(\varepsilon^{3/2}) \\ \frac{1}{\sqrt{2}}(1 + \frac{\varepsilon}{4}) + O(\varepsilon^2) \\ \frac{1}{\sqrt{2}}(-1 + \frac{3\varepsilon}{4}) + O(\varepsilon^2) \end{bmatrix}, v_2 = \begin{bmatrix} -\sqrt{\varepsilon/2} + O(\varepsilon^{3/2}) \\ \frac{1}{\sqrt{2}}(1 - \frac{5\varepsilon}{4}) + O(\varepsilon^2) \\ \frac{1}{\sqrt{2}}(1 + \frac{3\varepsilon}{4}) + O(\varepsilon^2) \end{bmatrix}, v_3 = \begin{bmatrix} 1 - \varepsilon/2 + O(\varepsilon^3) \\ \sqrt{\varepsilon} + O(\varepsilon^{3/2}) \\ \varepsilon^{3/2}/4 + O(\varepsilon^{5/2}) \end{bmatrix}.$$

Typically eigenvalues of a symmetric tridiagonal can be computed only to a guaranteed accuracy of $O(\varepsilon||T||)$. The eigenvalues of T as computed by MATLAB's **eig** function, that uses the QR algorithm, are

$$\begin{aligned} \hat{\lambda}_1 &= \varepsilon, \\ \hat{\lambda}_2 &= \varepsilon, \\ \hat{\lambda}_3 &= 1 + \varepsilon. \end{aligned}$$

The first eigenvector may be computed as

$$y_1 = (T - \hat{\lambda}_1 I)^{-1} b_1$$

where $b_1 = \xi_1 v_1 + \xi_2 v_2 + \xi_3 v_3$, $\xi_1^2 + \xi_2^2 + \xi_3^2 = 1$, and $\xi_2 \neq 0$.
In exact arithmetic,

$$\begin{aligned} y_1 &= \frac{\xi_1}{\lambda_1 - \hat{\lambda}_1} v_1 + \frac{\xi_2}{\lambda_2 - \hat{\lambda}_1} v_2 + \frac{\xi_3}{\lambda_3 - \hat{\lambda}_1} v_3 \\ &= \frac{\xi_2}{\lambda_2 - \hat{\lambda}_1} \left(v_2 + \frac{\xi_1}{\xi_2} \frac{\lambda_2 - \hat{\lambda}_1}{\lambda_1 - \hat{\lambda}_1} v_1 + \frac{\xi_3}{\xi_2} \frac{\lambda_2 - \hat{\lambda}_1}{\lambda_3 - \hat{\lambda}_1} v_3 \right) \\ &= \frac{1}{O(\varepsilon^2)} \left(v_2 + O(\varepsilon) v_1 + O(\varepsilon^2) v_3 \right) \end{aligned}$$

provided (ξ_1/ξ_2) and (ξ_3/ξ_2) are $O(1)$. Due to the inevitable roundoff errors in finite precision, the best we can hope to compute is

$$\hat{y}_1 = \frac{1}{O(\varepsilon^2)} \left(v_2 + O(\varepsilon) v_1 + O(\varepsilon) v_3 \right).$$

This vector is normalized to remove the $1/O(\varepsilon^2)$ factor and give $\hat{v}_1$. The second eigenvector may be computed as

$$y_2 = (T - \hat{\lambda}_2 I)^{-1} b_2.$$

Since $\hat{\lambda}_2 = \hat{\lambda}_1$, the computed value of y_2 is almost parallel to $\hat{v}_1$ (assuming that b_2 has a non-negligible component in the direction of v_2), i.e.,

$$\hat{y}_2 = \frac{1}{O(\varepsilon^2)} \left(v_2 + O(\varepsilon) v_1 + O(\varepsilon) v_3 \right).$$

Since $\hat{\lambda}_1$ and $\hat{\lambda}_2$ are coincident, typical implementations of inverse iteration orthogonalize the vectors $\hat{v}_1$ and $\hat{y}_2$ by the modified Gram-Schmidt(MGS) method in an attempt to reveal the second eigenvector :

$$\begin{aligned} z &= \hat{y}_2 - (\hat{y}_2^T \hat{v}_1) \hat{v}_1 \\ &= \frac{1}{\varepsilon^2} \left(O(\varepsilon) v_1 + O(\varepsilon) v_2 + O(\varepsilon) v_3 \right) \\ \hat{v}_2 &= z / ||z||. \end{aligned}$$

As we see above, the resulting vector $\hat{v}_2$ is far from being orthogonal to $\hat{v}_1$. A second step of MGS as recommended in [110, pgs.105–109] does not remedy the situation. Although the second step may remove the unwanted component of $\hat{v}_1$ from $\hat{v}_2$, the undesirable component of v_3 will remain (remember that λ_2 and λ_3 are well separated and most implementations will never explicitly orthogonalize $\hat{v}_2$ and $\hat{v}_3$). Some implementations may repeat the above process of inverse iteration followed by one or two steps of MGS in order to compute the second eigenvector. However the same sort of computations as described above are repeated and iterating does not satisfactorily solve the problem.

We now exhibit the above behavior by giving a numerical example. Set ε to the machine precision ($\approx 2.2 \times 10^{-16}$ in IEEE double precision arithmetic) in the matrix of (A.0.1).

The EISPACK implementation TINVIT does a step of inverse iteration followed by MGS and repeats this until convergence. See figure 2.3 and [118] for more details. Given the computed eigenvalues as $\hat{\lambda}_1 = \varepsilon$, $\hat{\lambda}_2 = \varepsilon(1+\varepsilon)$ and $\hat{\lambda}_3 = 1+\varepsilon$, TINVIT computes the first eigenvector as :

$$\hat{v}_1 = \begin{bmatrix} -1.05 \cdot 10^{-08} \\ 0.707 \\ 0.707 \end{bmatrix}$$

Note that here we chose $\hat{\lambda}_2$ to be a floating point number just bigger than $\hat{\lambda}_1$ to prevent TINVIT from perturbing $\hat{\lambda}_2$ by $\varepsilon||T||$. The computation of the second eigenvector is summarized in the following table :

Step	1	
	Before MGS	After MGS
Iterate (y)	$-1.05 \cdot 10^{-8}$ 0.707 0.707	$8.34 \cdot 10^{-9}$ -0.738 0.674
$y^T \hat{v}_1$	1.000	0.0452
$y^T \hat{v}_3$	$5.8 \cdot 10^{-25}$	$2.7 \cdot 10^{-9}$

TINVIT accepts the above iterate as having converged and hence the eigenvectors output are such that $\hat{v}_2^T \hat{v}_1 = 0.0452$ and $\hat{v}_2^T \hat{v}_3 = 2.7 \cdot 10^{-9}$! Due to the unwanted component of $\hat{v}_3$, the residual norm $||(T - \hat{\lambda}_2 I)\hat{v}_2|| = 2.7 \cdot 10^{-9}$. All the other residual norms are $O(\varepsilon||T||)$.

The LAPACK implementation DSTEIN performs two additional iterations after detecting convergence(see figure 2.4 and [87]). As in TINVIT, the first eigenvector is computed accurately by DSTEIN. The iterations for the computation of the second eigenvector are summarized in the following table.

Step	1		2		3	
	Before MGS	After MGS	Before MGS	After MGS	Before MGS	After MGS
Iterate (y)	$1.05 \cdot 10^{-8}$ -0.707 -0.707	$1.04 \cdot 10^{-8}$ -0.697 0.716	$1.05 \cdot 10^{-8}$ -0.707 -0.707	$1.05 \cdot 10^{-8}$ -0.7069 0.7073	$1.05 \cdot 10^{-8}$ -0.707 -0.707	$1.05 \cdot 10^{-8}$ -0.707108 0.707105
$\|y^T \hat{v}_1\|$	1.000	0.0014	1.000	$3.0 \cdot 10^{-4}$	1.000	$1.9 \cdot 10^{-6}$
$\|y^T \hat{v}_3\|$	$3.9 \cdot 10^{-24}$	$4.1 \cdot 10^{-11}$	$5.8 \cdot 10^{-25}$	$2.9 \cdot 10^{-12}$	$2.2 \cdot 10^{-24}$	$1.1 \cdot 10^{-13}$

Thus the eigenvectors output by DSTEIN are such that

$$\hat{v}_2^T \hat{v}_1 = 1.9 \cdot 10^{-6},$$

$\hat{v}_2^T \hat{v}_3 = 1.1 \cdot 10^{-13}$ and $||(T - \hat{\lambda}_2 I)\hat{v}_2|| = 1.1 \cdot 10^{-13}$.

Some implementations such as the PeIGS software library [52, 54] perform two steps of the MGS process after every inverse iteration step. To exhibit that this may also not be satisfactory, we modified TINVIT to perform two MGS steps. The following tabulates the iteration in computing the second eigenvector.

Step	1		
	Before MGS	After 1st MGS	After 2nd MGS
Iterate (y)	$1.05 \cdot 10^{-8}$ 0.707 0.707	$8.34 \cdot 10^{-9}$ -0.738 0.674	$-7.9 \cdot 10^{-9}$ -0.70710678118 0.70710678118
$y^T \hat{v}_1$	1.000	0.0452	$6.6 \cdot 10^{-16}$
$y^T \hat{v}_3$	$5.8 \cdot 10^{-25}$	$2.7 \cdot 10^{-9}$	$2.7 \cdot 10^{-9}$

Hence even though the vectors $\hat{v}_1$ and $\hat{v}_2$ are numerically orthogonal, $\hat{v}_2$ contains an unwanted component of $\hat{v}_3$. As a result $||(T - \hat{\lambda}_2)\hat{v}_2|| = 2.7 \cdot 10^{-9}$.

Note that the above problems occur because the eigenvalues are not accurate enough. Current inverse iteration implementations do not detect such inaccuracies.

Eigenvalues of the matrix in (A.0.1) may be computed to high relative accuracy by the bisection algorithm. If such eigenvalues are input to TINVIT or DSTEIN, the computed eigenvectors are found to be numerically orthogonal. In fact, even if no explicit orthogonalization is done inside TINVIT and DSTEIN, the eigenvectors output turn out to be numerically orthogonal. We will investigate this phenomenon in Case Study B.

Case Study B

Bidiagonals are Better

The purpose of this case study is to show that a tridiagonal matrix is "better" represented by a product of bidiagonal matrices for the task of computing eigenvalues and eigenvectors.

When any numerical algorithm is implemented on a computer, roundoff error needs to be accounted for, and it is good practice to show that calculations are not destroyed by the limited precision of the arithmetic operations. Wilkinson made popular the art of *backward error analysis*, whereby the computed solution is shown (if possible) to be the exact solution corresponding to data that is slightly different from the input data [135]. When combined with knowledge about the sensitivity of the solution to small changes in the data, this approach enables us to prove the accuracy of various numerical procedures.

A real, symmetric tridiagonal matrix T is traditionally represented by its n diagonal and $n-1$ off-diagonal elements. Some highly accurate algorithms for finding the eigenvalues of T, such as bisection, can be shown to find an exact eigenpair of $T+\delta T$, where the perturbation δT is a small relative perturbation in only the off-diagonals of T. In this Case Study, by examining the effect of such perturbations on the spectrum of T, we show that for our purposes it is better to represent a tridiagonal matrix by its bidiagonal factors.

First, we examine the matrix we encountered in Case Study A earlier, and observe that it does have the perturbation properties we desire. However not all tridiagonals share this property and we will illustrate this with another matrix.

Consider the matrix T_1 and a small perturbation :

$$T_1 = \begin{bmatrix} 1 & \sqrt{\varepsilon} & 0 \\ \sqrt{\varepsilon} & 7\varepsilon/4 & \varepsilon/4 \\ 0 & \varepsilon/4 & 3\varepsilon/4 \end{bmatrix}, \quad T_1 + \delta T_1 = \begin{bmatrix} 1 & \sqrt{\varepsilon}(1+\varepsilon) & 0 \\ \sqrt{\varepsilon}(1+\varepsilon) & 7\varepsilon/4 & \varepsilon(1+\varepsilon)/4 \\ 0 & \varepsilon(1+\varepsilon)/4 & 3\varepsilon/4 \end{bmatrix} \tag{B.0.1}$$

where ε is of the order of the machine precision. The eigenvalues of T_1 are

$$\lambda_1 = \varepsilon/2 - \varepsilon^2/4 + O(\varepsilon^3), \;\; \lambda_2 = \varepsilon - \varepsilon^2/2 + O(\varepsilon^3), \;\; \lambda_3 = 1 + \varepsilon + O(\varepsilon^2),$$

and those of $T_1 + \delta T_1$ are

$$\lambda_1 + \delta\lambda_1 = \varepsilon/2 - 3\varepsilon^2/2 + O(\varepsilon^3), \; \lambda_2 + \delta\lambda_2 = \varepsilon - 5\varepsilon^2/4 + O(\varepsilon^3), \; \lambda_3 + \delta\lambda_3 = 1 + \varepsilon + O(\varepsilon^2).$$

The eigenvector matrix of T_1 is

$$V = \begin{bmatrix} -\sqrt{\varepsilon/2} + O(\varepsilon^{3/2}) & -\sqrt{\varepsilon/2} + O(\varepsilon^{3/2}) & 1 - \varepsilon/2 + O(\varepsilon^2) \\ \frac{1}{\sqrt{2}}(1 + \frac{\varepsilon}{4}) + O(\varepsilon^2) & \frac{1}{\sqrt{2}}(1 - \frac{5\varepsilon}{4}) + O(\varepsilon^2) & \sqrt{\varepsilon} + O(\varepsilon^{3/2}) \\ \frac{1}{\sqrt{2}}(-1 + \frac{3\varepsilon}{4}) + O(\varepsilon^2) & \frac{1}{\sqrt{2}}(1 + \frac{3\varepsilon}{4}) + O(\varepsilon^2) & \varepsilon^{3/2}/4 + O(\varepsilon^{5/2}) \end{bmatrix},$$

while the perturbed matrix $T_1 + \delta T_1$ has the eigenvectors

$$V + \delta V = \begin{bmatrix} -\sqrt{\varepsilon/2} + O(\varepsilon^{3/2}) & -\sqrt{\varepsilon/2} + O(\varepsilon^{3/2}) & 1 - \varepsilon/2 + O(\varepsilon^2) \\ \frac{1}{\sqrt{2}}(1 + \frac{9\varepsilon}{4}) + O(\varepsilon^2) & \frac{1}{\sqrt{2}}(1 - \frac{13\varepsilon}{4}) + O(\varepsilon^2) & \sqrt{\varepsilon} + O(\varepsilon^{3/2}) \\ \frac{1}{\sqrt{2}}(-1 + \frac{11\varepsilon}{4}) + O(\varepsilon^2) & \frac{1}{\sqrt{2}}(1 + \frac{11\varepsilon}{4}) + O(\varepsilon^2) & \varepsilon^{3/2}/4 + O(\varepsilon^{5/2}) \end{bmatrix}.$$

From the above expressions, it is easy to see that $|\delta\lambda_j|/|\lambda_j| = O(\varepsilon)$ and $|\delta V_{ij}|/|V_{ij}| = O(\varepsilon)$ for $i, j = 1, 2, 3$. Hence, the small relative perturbations in the off-diagonals of T_1 cause small relative changes in the eigenvalues and small relative changes in the eigenvector components. The eigenvalues of such a matrix may be computed to high relative accuracy by the bisection algorithm. It turns out that the high accuracy of these eigenvalues obviates the need to orthogonalize in implementations of inverse iteration. Indeed, we modified the LAPACK and EISPACK implementations of inverse iteration (DSTEIN and TINVIT respectively) by turning off all orthogonalization, and found the computed eigenvectors to be numerically orthogonal. Note that orthogonality is achieved despite the fact that λ_1 and λ_2 are quite close together!

The theory developed in [37] does predict that the eigenvalues of matrix T_1 given in (B.0.1) are determined to high relative accuracy with respect to small relative perturbations in the off-diagonal elements. However there are tridiagonals that behave differently.

To demonstrate this, we consider the tridiagonals :

$$T_2 = \begin{bmatrix} 1-\sqrt{\varepsilon} & \varepsilon^{1/4}\sqrt{1-7\varepsilon/4} & 0 \\ \varepsilon^{1/4}\sqrt{1-7\varepsilon/4} & \sqrt{\varepsilon}+7\varepsilon/4 & \varepsilon/4 \\ 0 & \varepsilon/4 & 3\varepsilon/4 \end{bmatrix},$$

$$T_2+\delta T_2 = \begin{bmatrix} 1-\sqrt{\varepsilon} & \varepsilon^{1/4}\sqrt{1-7\varepsilon/4}(1+\varepsilon) & 0 \\ \varepsilon^{1/4}\sqrt{1-7\varepsilon/4}(1+\varepsilon) & \sqrt{\varepsilon}+7\varepsilon/4 & \varepsilon(1+\varepsilon)/4 \\ 0 & \varepsilon(1+\varepsilon)/4 & 3\varepsilon/4 \end{bmatrix}.$$

The eigenvalues of T_2 are

$$\lambda_1 = \varepsilon/2+\varepsilon^{3/2}/8+O(\varepsilon^2),\ \ \lambda_2 = \varepsilon-\varepsilon^{3/2}/8+O(\varepsilon^2),\ \ \lambda_3 = 1+\varepsilon+O(\varepsilon^2),$$

while those of the perturbed matrix $T_2+\delta T_2$ are

$$\lambda_1+\delta\lambda_1 = \varepsilon/2-7\varepsilon^{3/2}/8+O(\varepsilon^2),\ \lambda_2+\delta\lambda_2 = \varepsilon-9\varepsilon^{3/2}/8+O(\varepsilon^2),\ \lambda_3+\delta\lambda_3 = 1+\varepsilon+O(\varepsilon^2).$$

The eigenvectors of T_2 are given by

$$V = \begin{bmatrix} \sqrt{\frac{\varepsilon}{2}}(1+\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon^{5/4}) & -\sqrt{\frac{\varepsilon}{2}}(1+\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon^{5/4}) & 1-\sqrt{\varepsilon}/2+O(\varepsilon) \\ -\frac{1}{\sqrt{2}}(1-\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon) & \frac{1}{\sqrt{2}}(1-\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon) & \varepsilon^{1/4}(1+\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon^{5/4}) \\ \frac{1}{\sqrt{2}}(1-\frac{3\varepsilon}{4})+O(\varepsilon^{3/2}) & \frac{1}{\sqrt{2}}(1+\frac{3\varepsilon}{4})+O(\varepsilon^{3/2}) & \frac{\varepsilon^{5/4}}{4}(1+\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon^{9/4}) \end{bmatrix},$$

and the eigenvectors of $T_2+\delta T_2$ are

$$V+\delta V = \begin{bmatrix} \sqrt{\frac{\varepsilon}{2}}(1+\frac{5\sqrt{\varepsilon}}{2})+O(\varepsilon^{5/4}) & -\sqrt{\frac{\varepsilon}{2}}(1-\frac{3\sqrt{\varepsilon}}{2})+O(\varepsilon^{5/4}) & 1-\sqrt{\varepsilon}/2+O(\varepsilon) \\ -\frac{1}{\sqrt{2}}(1+\frac{3\sqrt{\varepsilon}}{2})+O(\varepsilon) & \frac{1}{\sqrt{2}}(1-\frac{5\sqrt{\varepsilon}}{2})+O(\varepsilon) & \varepsilon^{1/4}(1+\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon^{5/4}) \\ \frac{1}{\sqrt{2}}(1-2\sqrt{\varepsilon})+O(\varepsilon) & \frac{1}{\sqrt{2}}(1+2\sqrt{\varepsilon})+O(\varepsilon) & \frac{\varepsilon^{5/4}}{4}(1+\frac{\sqrt{\varepsilon}}{2})+O(\varepsilon^{9/4}) \end{bmatrix}.$$

Note that from the above expressions, we see that

$$\begin{aligned} |\delta\lambda_j/\lambda_j| &= O(\sqrt{\varepsilon}), \text{ and} \\ |\delta V_{ij}/V_{ij}| &= O(\sqrt{\varepsilon}) \quad \text{for} \quad i=1,2,3 \quad \text{and} \quad j=1,2. \end{aligned}$$

Thus T_2 does not determine its eigenvalues and eigenvector components to high relative accuracy and consequently, it is unlikely that we would be able to compute numerically orthogonal eigenvectors by inverse iteration without taking recourse to explicit orthogonalization. To confirm this, we repeated the experiment of turning off all orthogonalization in DSTEIN and TINVIT but found the computed "eigenvectors" *to have dot products as large as* $O(\sqrt{\varepsilon})$.

However, there is a way out of this impasse. Since T_2 is positive definite, we can form its Cholesky decomposition,

$$T_2 \;=\; LL^T.$$

Due to the tridiagonal form of T_2, the Cholesky factor L is bidiagonal. It is now known that small relative changes to the entries of a bidiagonal matrix cause small relative changes to its singular values, and small changes in the directions of the singular vectors if the relative gaps between the singular values are large. Note that the eigenvalues of T_2 are the squares of the singular values of L while its eigenvectors are the left singular vectors of L. This property of bidiagonal matrices gives us an impetus to produce algorithms where the backward errors may be posed as small relative changes in L rather than in T. By doing so we can eliminate the need for orthogonalization when finding eigenvectors of T_2 since its eigenvalues are *relatively far apart*. We have accomplished this task in Chapter 4.

The observant reader will notice that both T_1 and T_2 are positive definite and thus admit Cholesky decomposition. What can we do if the tridiagonal is indefinite? In such a case, we could turn to the LU decomposition of the tridiagonal. However, the relative perturbation theory in the indefinite case has not been extensively studied and is not well understood. We attempt to answer some questions in Chapter 5.

Case Study C

Multiple representations lead to orthogonality

In Chapter 4, we showed how to compute orthogonal approximations to eigenvectors of a positive definite tridiagonal when the eigenvalues are separated by large relative gaps. For example, for the eigenvalues ε and 2ε, we are able to compute orthogonal eigenvectors without resorting to Gram-Schmidt (see Chapter 4 for more details).

However, eigenvalues $1+\sqrt{\varepsilon}$ and $1+2\sqrt{\varepsilon}$ have a relative gap of $O(\sqrt{\varepsilon})$ and the methods of Chapter 4 can only guarantee that the dot product of the approximate eigenvectors is $O(\sqrt{\varepsilon})$.

In this section, we present examples to demonstrate ways of obtaining orthogonality *even when relative gaps appear to be tiny.* Consider the matrix

$$\mathrm{T}_0 = \begin{bmatrix} .520000005885958 & .519230209355285 & & \\ .519230209355285 & .589792290767499 & .36719192898916 & \\ & .36719192898916 & 1.89020772569828 & 2.7632618547882\cdot 10^{-8} \\ & & 2.7632618547882\cdot 10^{-8} & 1.00000002235174 \end{bmatrix}$$

with eigenvalues

$$\lambda_1 \approx \varepsilon, \quad \lambda_2 \approx 1+\sqrt{\varepsilon}, \quad \lambda_3 \approx 1+2\sqrt{\varepsilon}, \quad \lambda_4 \approx 2.0, \tag{C.0.1}$$

where $\varepsilon \approx 2.2\times 10^{-16}$ (all our results are in IEEE double precision arithmetic).

Standard methods can compute approximate eigenvectors of λ_1 and λ_4 that are orthogonal to working accuracy. Algorithm X of Section 4.4.3 computes the following approximations to the interior eigenvectors

$$\hat{\mathrm{v}}_2 = \begin{bmatrix} .5000000027411224 \\ .4622227318424748 \\ -.1906571623774349 \\ .7071067740172921 \end{bmatrix}, \qquad \hat{\mathrm{v}}_3 = \begin{bmatrix} -.5000000074616407 \\ -.4622227505556183 \\ .1906571293895208 \\ .7071067673414712 \end{bmatrix},$$

but $|\hat{v}_2^T \hat{v}_3| = O(\sqrt{\varepsilon})$. By standard perturbation theory, the subspace spanned by v_2 and v_3 is not very sensitive to small changes in T_0 and thus, $\hat{v}_2$ and $\hat{v}_3$ are orthogonal to the eigenvectors v_1 and v_4. However, the individual eigenvectors v_2 and v_3 are more sensitive and the $O(\sqrt{\varepsilon})$ dot products are a consequence.

In spite of the above observation, we can ask ourselves if it is possible to obtain orthogonal $\hat{v}_2$ and $\hat{v}_3$ without resorting to Gram-Schmidt. A natural approach is to try and extend the ideas of Chapter 4 which enable us to compute orthogonal eigenvectors even when eigenvalues are as close as ε and 2ε. We make two crucial observations

1. Eigenvectors are shift invariant, i.e.,

$$\text{Eigenvectors of } T \;\equiv\; \text{Eigenvectors of } T - \mu I, \quad \text{for all } \mu.$$

2. Relative gaps can be changed by shifting, i.e.,

$$\mathrm{relgap}(\lambda_i, \lambda_{i+1}) \;\neq\; \mathrm{relgap}(\lambda_i - \mu, \lambda_{i+1} - \mu).$$

For example, the interior eigenvalues of $T_0 - I$ are $\sqrt{\varepsilon}$ and $2\sqrt{\varepsilon}$, and these shifted eigenvalues now have a large relative gap! As we did for the positive definite case, we can find bidiagonal factors of the indefinite matrix $T_0 - I$. However, unlike the positive definite case there is no guarantee that these bidiagonal factors will determine their small eigenvalues to high relative accuracy.

A distinguishing feature of factoring a positive definite matrix is the absence of any element growth. When we factor an indefinite matrix, the near singularity of principal submatrices can cause large element growths. However, in many cases we will be able to form the L, U factors of a tridiagonal matrix without any element growth.

When we proceed to find the decomposition

$$T_0 - \mu I \;=\; L_0 D_0 L_0^T, \tag{C.0.2}$$

where $\mu = 1$, we get the following elements on the diagonal of D_0 and off-diagonal of L_0.

$$\mathrm{diag}(\mathrm{D}_0) = \begin{bmatrix} -.4799999941140420 \\ .1514589857947483 \\ 3.074504323352656 \cdot 10^{-7} \\ 1.986821250068485 \cdot 10^{-8} \end{bmatrix}, \qquad \mathrm{diag}(\mathrm{L}_0, -1) = \begin{bmatrix} -1.081729616088125 \\ 2.424365428451800 \\ .08987666186707277 \end{bmatrix}.$$

We observe that there is no element growth in forming this decomposition. In particular,

$$\text{Element Growth} \stackrel{\text{def}}{=} \max \left| \frac{D_0(i,i)}{T_0(i,i) - \mu} \right|^{1/2} = 1. \tag{C.0.3}$$

Note that the element growth always equals 1 when factoring a positive definite tridiagonal (see Lemma 4.4.1).

We say that the decomposition C.0.2 is a new *representation* of T (based) at $\mu = 1$. We can now compute the interior eigenvalues of $L_0 D_0 L_0^T$ accurately by the bisection algorithm using any of the differential qd-like transformations given in Section 4.4.1 as the inner loop. These "refined" eigenvalues are

$$\delta\hat{\lambda}_2 = 1.4901161182018 \cdot 10^{-8} \approx \sqrt{\varepsilon}, \quad \text{and} \quad \delta\hat{\lambda}_3 = 2.9802322498846 \cdot 10^{-8} \approx 2\sqrt{\varepsilon}.$$

The relative gap between $\delta\hat{\lambda}_2$ and $\delta\hat{\lambda}_3$ is large and so we can attempt to compute the eigenvectors of $L_0 D_0 L_0^T$ corresponding to these eigenvalues. By employing methods similar to those given in Chapter 4 (more specifically, Step (4d) of Algorithm X), we find

$$\hat{\mathrm{v}}_2 = \begin{bmatrix} .4999999955491866 \\ .4622227251939223 \\ -.1906571596350473 \\ .7071067841882251 \end{bmatrix}, \qquad \hat{\mathrm{v}}_3 = \begin{bmatrix} .4999999997942006 \\ .4622227434674882 \\ -.1906571264658161 \\ -.7071067781848689 \end{bmatrix}.$$

Miraculously the above vectors are orthogonal to working accuracy with $|\hat{v}_2^T \hat{v}_3| = 2\varepsilon$*!* As before, $\hat{v}_2$ and $\hat{v}_3$ are orthogonal to v_1 and v_4. On further reflection, we realize that the success of the above approach is due to the *relative robustness* of the representation obtained by (C.0.2). We say that a representation is *relatively robust* if it determines all its eigenvalues to high relative accuracy with respect to small relative changes in the individual entries of the representation.

As opposed to the case of bidiagonal Cholesky factors, there is no existing theory to show that factors of an indefinite tridiagonal are relatively robust. Indeed bidiagonal factors of an indefinite $T - \mu I$ may be far from being relatively robust (see Example 5.2.3).

However, we suspect a strong correlation between the lack of element growth and relative robustness. We now present some evidence in support of this conjecture.

In Section 5.2, we introduced *relative condition numbers* $\kappa_{rel}(\lambda_j)$ for each eigenvalue λ_j of an LDL^T representation, These condition numbers indicate the relative change in an eigenvalue due to small relative perturbations in entries of L and D. We define

$$\kappa_{rel}(LDL^T) \stackrel{\text{def}}{=} \max_j \kappa_{rel}(\lambda_j), \tag{C.0.4}$$

Note that if $\kappa(LDL^T) = O(1)$ then LDL^T determines *all* its eigenvalues to high relative accuracy. In some cases, an LDL^T representation may determine only a few of its eigenvalues to high relative accuracy.

In Figure C.1, we have plotted the element growths and the relative condition numbers, as defined by (C.0.3)) and (C.0.4) (see also (5.2.8)) respectively, for representations of T_0 (based) at various shifts. The X-axis plots the shifts μ of (C.0.2), while the Y-axis plots the element growths (represented by the green circles **o**) and relative condition numbers (represented by the red pluses $+$) on a logarithmic scale. We have taken extra data points near Ritz values (eigenvalues of proper leading submatrices), and eigenvalues of T_0 since we may fear large element growths close to these singularities. The eigenvalues of T_0 are indicated by the solid vertical lines, while the Ritz values are indicated by the dotted vertical lines.

In Figure C.1, we see that whenever the element growth is $O(1)$ the representation is relatively robust. In particular, relatively robust representations exist near all the eigenvalues. Earlier, we observed no element growth at $\mu = 1$. The enlarged picture in Figure C.2 shows that there are many other relatively robust representations near λ_2 and λ_3. However, there are large element growths near the mid-point of these eigenvalues. From Figure C.1, we also observe that representations based or anchored near the Ritz values of T_0 are not relatively robust.

To see a slightly different behavior, we examine another matrix with identical spectrum (as in (C.0.1)),

$$\mathrm{T}_1 = \begin{bmatrix} 1.00000001117587 & .707106781186547 & & \\ .707106781186547 & .999999977648258 & .707106781186546 & \\ & .707106781186546 & 1.00000003352761 & 1.05367121277235 \cdot 10^{-8} \\ & & 1.05367121277235 \cdot 10^{-8} & 1.00000002235174 \end{bmatrix}.$$

Here also, in order to compute orthogonal approximations to v_2 and v_3 we can

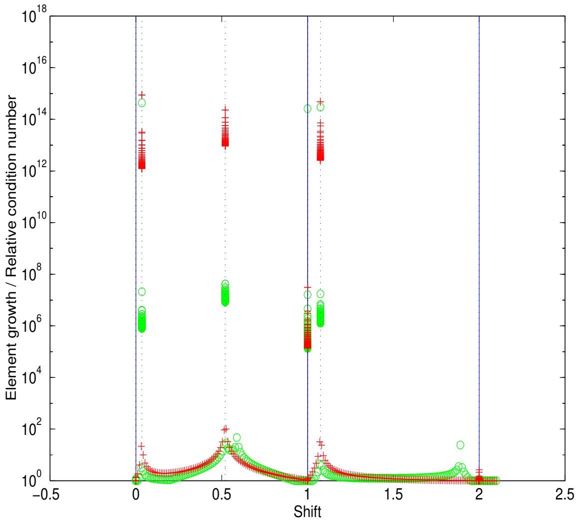

Figure C.1: The strong correlation between element growth and relative robustness

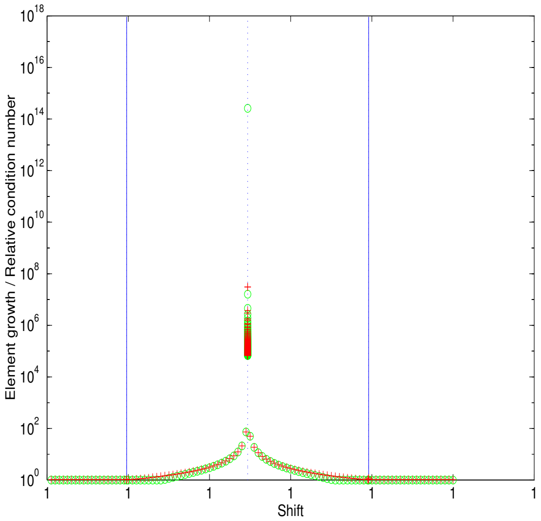

Figure C.2: Blow up of earlier figure : X-axis ranges from $1 + 10^{-8}$ to $1 + 3.5 \times 10^{-8}$

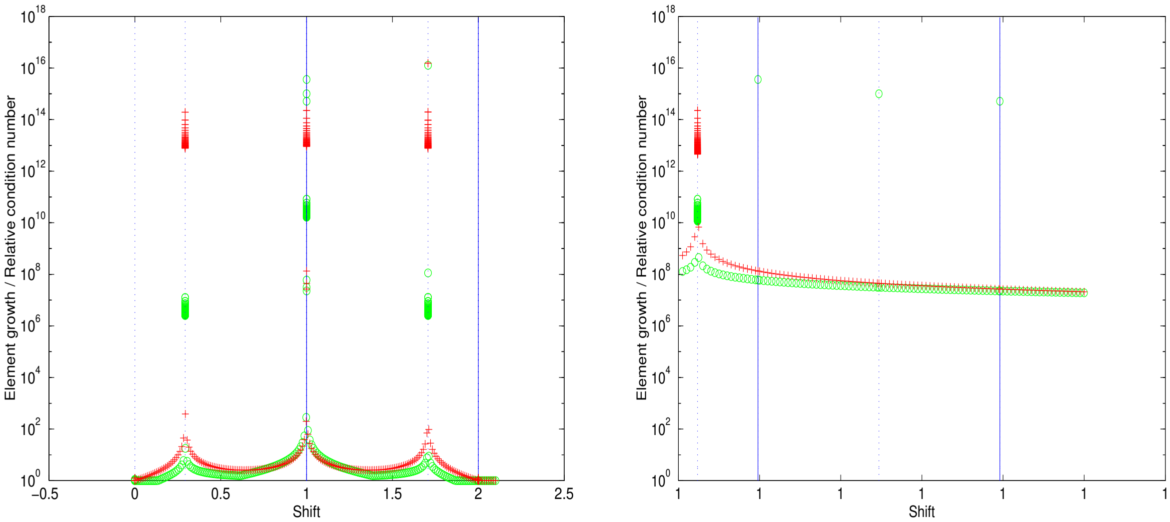

Figure C.3: Relative condition numbers (in the figure to the right, X-axis ranges from $1 + 10^{-8}$ to $1 + 3.5 \times 10^{-8}$)

attempt to form the representation

$$T_1 - I \;=\; L_1 D_1 L_1^T. \tag{C.0.5}$$

However,

$$\mathrm{diag(D_1)} \;=\; \begin{bmatrix} 1.117587089538574 \cdot 10^{-8} \\ -4.473924266666669 \cdot 10^{7} \\ 4.470348380358755 \cdot 10^{-8} \\ 1.986821493746602 \cdot 10^{-8} \end{bmatrix}, \qquad \mathrm{diag(L_1,-1)} \;=\; \begin{bmatrix} 6.327084877781628 \cdot 10^{7} \\ -1.580506693551128 \cdot 10^{-8} \\ .2357022791274500 \end{bmatrix},$$

and since $T_1(2,2) - 1 \;=\; -2.24 \cdot 10^{-8}$, the element growth is $\approx 10^8$! Due to this large element growth, we may suspect that this representation is *not* relatively robust. Indeed, $\kappa(L_1 D_1 L_1^T) \approx 4.5 \cdot 10^7$. To see if there are any good representations near 1, we again plot the element growths and relative condition numbers on the Y-axis against the shifts on the X-axis. Figure C.3 suggests that there are no good representations near λ_2 and λ_3 (the reader should contrast this with Figures C.1 and C.2, especially the blow ups).

However, there is no cause for alarm. Remember that we are interested in getting a "good" representation near λ_2 and λ_3 in order to compute the corresponding eigenvectors. Figure C.3 indicates that there is no representation at these shifts that is relatively robust for *all* eigenvalues. However, all we desire of this representation is that it determine *its two* *smallest* eigenvalues to high relative accuracy. Indeed, we discover that despite the large element growths, the relative condition numbers $\kappa_{rel}(\lambda_2)$ and $\kappa_{rel}(\lambda_3)$ for the representations close to λ_2 and λ_3 are $O(1)$, i.e., *the representations near 1 do determine their two smallest eigenvalues to high relative accuracy.* Thus, just as before, we can compute $\hat{v}_2$ and $\hat{v}_3$ from the representation based at 1, and get

$$\hat{\mathrm{v}}_2 = \begin{bmatrix} .4999999981373546 \\ 2.634178061370114 \cdot 10^{-9} \\ -.4999999981373549 \\ .7071067838207259 \end{bmatrix}, \qquad \hat{\mathrm{v}}_3 = \begin{bmatrix} -.5000000018626457 \\ -1.317089024797209 \cdot 10^{-8} \\ .5000000018626453 \\ .7071067785523688 \end{bmatrix}, \qquad \text{(C.0.6)}$$

where $|\hat{v}_2^T \hat{v}_3| < 3\varepsilon$!

We have observed numerical behavior similar to T_0 and T_1 in many larger matrices. The more common case shows no element growth for shifts near the eigenvalues, whereas large element growths, as in the case of T_1, occur rarely. The study of relative condition numbers of individual eigenvalues, especially the tiny ones, is pursued in Chapter 5.