

On MR^3 -type Algorithms for the Tridiagonal Symmetric Eigenproblem and the Bidiagonal SVD



Zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

am Fachbereich Mathematik und Naturwissenschaften der
Bergischen Universität Wuppertal vorgelegte und genehmigte

Dissertation

von

Dipl.-Inf. Paul Willems

aus Köln

Gutachter: Prof. Dr. Bruno Lang

Gutachter: Prof. Beresford N. Parlett, PhD

Dissertation eingereicht am: 20. Dezember 2009

Tag der Disputation: 13. April 2010

Diese Dissertation kann wie folgt zitiert werden:

urn:nbn:de:hbz:468-20100243

[<http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3A468-20100243>]

Acknowledgments

The present thesis was written between January 2005 and December 2009, during my employment as scientific assistant at the Applied Computer Science Group within the department of Mathematics and Computer Science of the University of Wuppertal. Since December 2008 this research has been partially funded by the *Bundesministerium für Bildung und Forschung*, contract number 01 IH 08 007 B, within the project *ELPA-Eigenwert-Löser für Petaflop-Anwendungen*.

At this point I want to extend gratitude to the people that supported my work in some way or other, directly or indirectly.

First and foremost, I am indebted to my thesis advisor, Prof. Dr. Bruno Lang. I first met him in Aachen, as student attending a lab on parallel programming with Java he gave in the summer of 2000. During later collaboration in the form of student jobs, a diploma thesis and finally this dissertation, Prof. Lang has become a mentor and set a great example. While I was originally drawn more towards the theoretical side of “traditional” computer science, it is due to his instigation that I finally ended up in numerical linear algebra. Prof. Lang was always generous with his time (which is scarce) and offered support and help whenever needed. He let me find my own ways, in my own time (well, except for gentle nudges in the right direction here and there).

I want to express my appreciation to Prof. Dr. Andreas Frommer for providing shelter in his research group. I have come to admire the way he performs and conveys mathematics, and would have loved being able to attend his lectures as a student. The research group in Wuppertal has been a uniquely enjoyable working environment, due to the special ensemble of people that are and have been working there.

I am very grateful to my colleagues Dr. Katrin Schäfer and Lukas Krämer, who diligently read a near-final version of the whole thesis from cover to cover and contributed a plethora of corrections and suggestions.

This project was jump-started by a research visit to Berkeley in the end of 2004, and I want to thank the Research Centre Jülich for the financial support making it possible. During that visit I had the pleasure to meet the one and only Prof. Beresford Parlett. The number of ways in which he influenced and contributed to this thesis are too numerous to count: his wonderful book and papers forming the basis for everything, the many detailed and fruitful email-discourses we had, and ultimately his willingness to become external referee. His sincere interest for my work has been a huge source of motivation.

Also in Berkeley I met Dr. Christof Vömel, another fellow grail-hunter. Besides his theoretical contributions to MR³, the fact I maybe came to appreciate most was that he has struggled with MR³ on the code-level (and won). We had many enlightening discussions (powered by SKYPETM) about the deepest and darkest corners of MR³.

Finally, there is a real life. I want to give a big collective Thank You to my family and friends for making that life worthwhile. I know that without them providing support and understanding when needed (and distraction otherwise), I would have gotten bogged down somewhere along this road.



Contents

Introduction	iii
1 Foundations	1
1.1 Boot Camp	1
1.2 Computing with Errors	9
1.2.1 Floating-Point Arithmetic	9
1.2.2 Condition of Addition & Cancellation	12
1.2.3 A Toolset for Error Analysis	15
1.3 Angles between subspaces	21
1.4 Greatest Hits of SEP	27
1.4.1 Rayleigh's Quotient, Residuals and Gaps	28
1.4.2 The Classics	33
1.4.3 Perturbation Theory	38
2 The MR³ Algorithm	43
2.1 MR ³ in a Nutshell	46
2.1.1 The Idea	46
2.1.2 The Algorithm	50
2.2 Proof of Correctness	53
2.3 Twisted Factorizations and their role(s) in MR ³	63
2.3.1 Burn At Both Ends	64
2.3.2 Eigenvectors with a Twist	71
2.3.3 Relative Condition Numbers	76
2.4 QD-Algorithms	80
2.4.1 Different flavors of Representations	82
2.4.2 Stationary Factorization	84
2.4.3 Progressive Factorization	91
2.4.4 Twisted Factorization	97

2.4.5	Additional Remarks & Discussion	102
2.5	Towards an improved implementation of MR^3	106
3	MR^3 for the Bidiagonal SVD	117
3.1	Basics	119
3.1.1	The Problem	119
3.1.2	Associated Tridiagonal Problems	120
3.1.3	Setting the stage	126
3.2	MR^3 and the Normal Equations	128
3.3	MR^3 and the Golub-Kahan Matrix	134
3.4	Couplings	143
3.5	The Algorithm of Multiple Coupled RRRs	150
3.5.1	Description of the Algorithm	151
3.5.2	Proof of Correctness	152
3.5.3	Shifting coupled nodes	159
3.6	Numerical Results	161
4	Block Factorizations	165
4.1	Basics	166
4.1.1	A note on 2-by-2 matrices	166
4.1.2	Structural implications of 2-by-2 pivots	167
4.1.3	When not to choose a 2-by-2 pivot	174
4.2	Stationary Block Factorization	176
4.2.1	Keep a block	185
4.2.2	Break a block	188
4.2.3	Creating blocks & handling overlap	199
4.2.4	The complete algorithm	209
4.3	Numerical Results	214
	Summary	221
	List of Figures	223
	List of Tables	226
	List of Algorithms	227
	Bibliography	234
	Summary of Notation	235

Introduction

A beginning is the time for taking
the most delicate care that
the balances are correct.
— FRANK HERBERT, *Dune* (1965)

This thesis concerns two very basic problems:

- TSEP — Compute the *eigendecomposition* of a real symmetric tridiagonal matrix.
- BSVD — Compute the *singular value decomposition* of a real bidiagonal matrix.

Both problems have in common that they look deceptively simple, since, after all, the number of inputs is only linear in the matrix dimension. They also have in common that solving them provides the means to compute eigen-/singular value decompositions of Hermitian/general dense complex matrices, and those are two of the most challenging tasks in numerical linear algebra.



In 1997, Inderjit Dhillon finished his thesis [15] where he proposed the *Algorithm of Multiple Relatively Robust Representations* (MR^3) as new solution method for TSEP. It offered optimal complexity together with embarrassing parallelism, even if not the full eigendecomposition was desired. Thus it was heralded as breakthrough achievement, and rightly so. Indeed, this was just what everyone had been looking for, so in some inner circles of eigenvalue-hunters, MR^3 came to be known by the more catchy title “The Grail”.

The singular value decomposition (SVD) of a bidiagonal matrix can be obtained in multiple ways by computing eigendecompositions of related symmetric

tridiagonal matrices. Hence the natural desire arose to employ MR^3 for that purpose. However, all the resulting solution strategies for BSVD turned out to be unstable. This was investigated by Benedikt Großer, who found that the reason lay in the most elementary feature distinguishing BSVD from TSEP: the former requires two full systems of orthonormal vectors. These must not be treated independently, since the left and right singular vectors have to fit together on a one-to-one basis. None of the “black box” applications of MR^3 to related tridiagonal problems could guarantee to get the connection right. For his thesis [35], Großer devised a new scheme based on *coupling relations* to run MR^3 implicitly on two tridiagonal matrices at once, in a synchronized fashion. This seemed to solve the problem and delivered properly paired singular vectors.

The coupling-based approach was promising enough to be deemed worthy of inclusion into the renowned LAPACK [1] software library. This is where my involvement with the topic began, since Benedikt Großer had already left the academic field at that time. During a visit to Berkeley at the end of 2004, I worked on an implementation of the “coupled MR^3 ” which should incorporate the latest developments for MR^3 . The version that was produced worked fairly well, but there were also too many test cases where the results were not satisfactory. Further investigation revealed the cause to be gaps in the existing theory for the coupled approach.

Our main motivation for this thesis has been to close these gaps and extend Großer’s work to finally obtain a MR^3 -based solution method for BSVD that retains all benefits of MR^3 , rests on a solid body of theory, and is accompanied by a stable, reliable and efficient software implementation. To achieve these goals, we also had to reexamine how some aspects of MR^3 itself might be improved.

◇

A general outline of this work is as follows. In Chapter 1 we fix notation and assemble the necessary tools for later. This includes an introduction to our way of handling error analysis, which is rather technical by nature. Furthermore we give a compressed selection of those parts of the beautiful and deep theory of the symmetric eigenproblem that will be referred to later.

The MR^3 -algorithm has the whole Chapter 2 to its own. We will present the algorithm in a revised form that is both simpler and more abstract than earlier versions. Building on this we provide a complete proof of correctness and error analysis, which will become the foundation for the theory in Chapter 3. A prominent role will be played by shifted bidiagonal factorizations of symmetric tridiagonal matrices, in the form

$$\text{LDL}^* - \tau = \text{L}^+\text{D}^+(\text{L}^+)^*, \quad (\star)$$

where the factors D , D^+ are diagonal and L , L^+ are (unit lower) bidiagonal. MR^3 owes part of its success to the fact that such decompositions, and variants of

them, can be computed with componentwise mixed relative stability. To this end, customized variants of Rutishauer’s qd-transformations are employed. We will study those, and also adapt them to different interpretations of what the defining data for a bidiagonal factorization can be. Our treatment of MR^3 closes with numerical experiments; we have engineered an implementation of MR^3 that incorporates some new techniques and want to compare it with code from LAPACK.

Chapter 3 is devoted to problem BSVD and how the MR^3 -algorithm can be harnessed for its solution. We give a thorough theoretical analysis of the problems involved with the pure black box approaches. Then we show how one of these—namely using MR^3 on the so-called *Golub-Kahan matrix*—can actually be repaired to solve BSVD reliably. After this we treat the coupled approach, for which we can now provide a complete proof of correctness including rigorous error bounds. Thus we have now in fact two working methods for BSVD, which are intimately related but nevertheless different. We have provided implementations for both, and at the end of Chapter 3 numerical experiments shall ascertain that both are indeed effective solutions for BSVD.

The final Chapter 4 is about what we call *block factorizations*. It could, in principle, be regarded on its own. The motivation is that standard bidiagonal factorizations like in (\star) above are vulnerable to element growth, just like all other variants of Gaussian Elimination. This vulnerability extends to MR^3 and as such also to the MR^3 -based methods for BSVD.

One way to counter element growth before it even arises is to allow 2×2 -pivots, or *blocks*, in \mathbf{D} and \mathbf{D}^+ . We have developed a new algorithm to compute shifted block factorizations, for which we can prove componentwise mixed relative stability. The latter makes Chapter 4 a very technical one, but is unavoidable to employ block factorizations within MR^3 .

The implementations for TSEP and BSVD that were introduced in the chapters before have been alternatively realized using blocks, and at the end of Chapter 4 we compare the versions by the way of numerical experiments. The conclusion will be that block factorizations can lift MR^3 -based methods to a whole new level of robustness, but at some cost to efficiency.

Chapter 1

Foundations

I do not define time, space, place, and motion, as being well known to all.
— ISAAC NEWTON, *Philosophiae Naturalis Principia Mathematica* (1687)

The enjoyment of one's tools is an essential ingredient of successful work.
— DONALD E. KNUTH, *The Art of Computer Programming, Volume 2, Seminumerical Algorithms* (1998)

As the title conveys, this chapter will lay some necessary groundwork. In §1.1 we recapitulate some elementary facts and introduce our notation. The content of §1.2 is floating-point arithmetic and error analysis. §1.3 is devoted to the concept of angles between subspaces, which is derived from first principles. Finally, in §1.4 we cover parts of the theory of the symmetric eigenproblem.

With the sole exception of §1.1, everything is presented in full detail, including proofs, and everything that is presented will be needed at some later point. The motivation for doing so is that this chapter makes the whole work self-contained while being as compact as possible. Except for marginal issues, each result or proof that will have to be built upon later is right here.

Readers experienced in the field will probably know most of the contents of this chapter in some form and can safely skip parts of it. To simplify this, many parts of §1.2–§1.4 contain at the beginning some pointers to the most important aspects that should be noticed. In any case we recommend to read §1.1 in full to get a feeling for our notation.

1.1 Boot Camp

This section is intended as a refresher course with emphasis on introducing the parts of our notation that might be considered nonstandard. The topics are

among those any basic linear algebra course should convey; for the most part they will be leaned upon implicitly in the rest of this thesis. Therefore the speed will be rapid, proofs omitted, and only a bare minimum of equations are numbered to allow for future explicit reference.

The core of our notation follows [56] in using upper and lower case sans-serif letters for matrices \mathbf{A} and vectors \mathbf{x} . Most of the time we will also reserve symmetric letters \mathbf{A} , \mathbf{H} , \mathbf{M} and \mathbf{T} for symmetric matrices.

The main problems considered in this thesis are the symmetric eigenproblem, in particular the tridiagonal case, and the bidiagonal SVD. Those can all be completely and satisfactorily solved while sticking to the reals, so complex numbers will not be an issue. Hence we focus the spotlight of our attention on \mathbb{R}^n , identified with the n -dimensional Euclidean Space over \mathbb{R} . Elements of \mathbb{R}^n are understood to be *column*-vectors.

Although considering only real numbers, for purely aesthetic reasons we still use the superscript $*$ to indicate transposition, instead of the usual T . Thus, \mathbf{x}^* is a row-vector.

A summary of the notation can be found on page 235.

Numbers & Indices

The natural numbers are $\mathbb{N} = \{1, 2, \dots\}$ without zero in contrast to $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$.

An *index* can in principle be any integer in \mathbb{Z} , although we assume them to be in-range implicitly if used with vectors and matrices. Any *nonempty* set of indices is an *index set*. If it consists of a sequence $a, a + 1, \dots, b$ of consecutive indices, we call it *index range* or *interval* and write $a:b$, in MATLABTM-notation. If the lower and/or upper index bound is omitted it should be replaced by the smallest/largest index that makes sense in the context, respectively.

We define the sign of any real (number) $x \in \mathbb{R}$, also denoted as *scalar*, as

$$\text{sign}(x) = \begin{cases} -1, & \text{if } x < 0, \\ 0, & \text{if } x = 0, \\ +1, & \text{if } x > 0. \end{cases}$$

Vectors & Matrices

Let us showcase our notation with the help of some vector $\mathbf{x} \in \mathbb{R}^n$ and some matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$. To access the entries of \mathbf{x} we use parentheses, i.e.,

$$\mathbf{x} = \begin{pmatrix} \mathbf{x}(1) \\ \mathbf{x}(2) \\ \vdots \\ \mathbf{x}(n) \end{pmatrix}.$$

Note that we would write x_1 to denote a vector with index one but not the first entry of x . For an index set I the subvector of x built from just the entries whose indices are in the set is denoted as x_I ; for example:

$$x = \begin{pmatrix} 5 \\ 7 \\ 11 \end{pmatrix} \Rightarrow x_{\{1,3\}} = \begin{pmatrix} 5 \\ 11 \end{pmatrix}, x_{:2} = \begin{pmatrix} 5 \\ 7 \end{pmatrix}.$$

The identity matrix is I_n or simply I if the dimension is clear from the context. Its columns e_k are called *coordinate vectors* or *canonical vectors* and obey

$$e_k(i) = \begin{cases} 1, & \text{if } k = i, \\ 0, & \text{otherwise.} \end{cases}$$

The entries of a matrix A can also be accessed using parentheses, or alternatively via subscripts, that is,

$$A(i, j) = A_{i,j} = e_i^* A e_j.$$

Note that the latter conveys implicitly that the e_i^* on the left has dimension m and the e_j on the right has dimension n .

For structured matrices it can be very useful to have access to individual bands. As is common practice we identify a band of an $m \times n$ -matrix with its distance k from the main diagonal, $-(m-1) \leq k \leq n-1$.

To extract a band from a given matrix, $\text{diag}_k[A]$ yields a matrix of same dimensions as A but with all entries zero except for the k th band, which is copied from A . Thus we can decompose a matrix by summing its bands, i.e.,

$$A = \sum_{k=-(m-1)}^{n-1} \text{diag}_k[A] \quad \text{for all } A \in \mathbb{R}^{m \times n}.$$

A complementary tool is $\text{diag}_k(a_1, \dots, a_{n-k})$ to specify a square matrix of dimension $n \times n$ that is zero except for the entries a_i on its k th band. For both notations we omit k if it is zero and allow to replace k by $\pm k$ for symmetric structure or even an arbitrary index set to work with a whole bunch of bands at once. Thus we have $I = \text{diag}(1, \dots, 1)$ and

$$A = \sum_{k=0}^{n-1} \text{diag}_{\pm k}(A(1, 1+k), \dots, A(n-k, n))$$

for all symmetric $A \in \mathbb{R}^{n \times n}$.

Given index sets I and J , both $A_{I,J}$ and $A(I, J)$ can be used to denote the submatrix that consists only of those entries with row-indices from I and column-indices from J . In particular, the i th row and j th column of A are

$$A(i, :) = A_{i,:} = \mathbf{e}_i^* A, \quad A(:, j) = A_{:,j} = A \mathbf{e}_j.$$

As you can see singleton sets $\{i\}$ may simply be written as i . For example:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \Rightarrow A_{:2,\{1,3\}} = A(:, \{1, 3\}) = \begin{pmatrix} 1 & 3 \\ 4 & 6 \end{pmatrix}.$$

For $I = J$, $A_I := A_{I,I}$ is called a *principal submatrix* of A .

Transpose and Inverse

For any matrices F and G we have

$$(FG)^* = G^*F^*,$$

so for any invertible $A \in \mathbb{R}^{n \times n}$,

$$AA^{-1} = \mathbf{I} = (AA^{-1})^* = (A^{-1})^*A^*.$$

Hence, the fact that the identity is symmetric implies that transposition and inversion commute, so

$$\boxed{A^{-*} := (A^{-1})^* = (A^*)^{-1}}$$

is well-defined.

Norms

The distinguishing feature of Euclidean Space \mathbb{R}^n is the standard scalar product

$$\mathbf{x}^* \mathbf{y} := \sum_{i=1}^n x(i)y(i).$$

It allows for a notion of *length* of vectors via the Euclidean or standard *norm*

$$\|\mathbf{x}\| := \sqrt{\mathbf{x}^* \mathbf{x}} = \left(\sum_{i=1}^n x(i)^2 \right)^{1/2},$$

also sometimes called 2-norm (in symbols: $\|\cdot\|_2$). There are other useful norms, like for instance the maximum norm

$$\|\mathbf{x}\|_\infty := \max_i \{|x(i)|\},$$

but for us, when we write $\|\mathbf{x}\|$ we always mean the Euclidean norm. A vector \mathbf{x} with $\|\mathbf{x}\| = 1$ is called *normalized* or *unit vector*.

The standard scalar product on \mathbb{R}^n allows to define an *angle* between nonzero vectors \mathbf{x} and \mathbf{y} via

$$\cos \phi := \frac{\mathbf{x}^* \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|},$$

such that the usual geometrical meaning (in \mathbb{R}^2 and \mathbb{R}^3) is retained. Replacing $\mathbf{x}^* \mathbf{y}$ by $|\mathbf{x}^* \mathbf{y}|$ on the right hand side gives the *acute* angle in $[0, \pi/2]$, a more suitable measure of angle between the one-dimensional subspaces $\text{span}\{\mathbf{x}\}$ and $\text{span}\{\mathbf{y}\}$.

This geometrical connotation of the standard scalar product is why \mathbf{x} and \mathbf{y} are called *orthogonal* when $\mathbf{x}^* \mathbf{y} = 0$. A set of pairwise orthogonal unit vectors is called an *orthonormal set*. If this pertains to the columns of a matrix $\mathbf{Q} \in \mathbb{R}^{m \times n}$ we have $\mathbf{Q}^* \mathbf{Q} = \mathbf{I}_n$ and \mathbf{Q} is *orthonormal*. We use the term *unitary* to denote a real square orthonormal matrix ($\mathbf{Q}^* = \mathbf{Q}^{-1}$), avoiding the common—but misleading—reuse of *orthogonal* for that purpose. More will be said about orthogonality and angles in §1.3.

For a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, any pair of vector norms on \mathbb{R}^m and \mathbb{R}^n induces an *operator norm*. Taking the standard norm both times, we get the *spectral norm*

$$\|\mathbf{A}\| := \max_{\mathbf{o} \neq \mathbf{x} \in \mathbb{R}^n} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \|\mathbf{x}\|=1}} \|\mathbf{A}\mathbf{x}\| = \max_{\substack{\mathbf{x} \in \mathbb{R}^n \\ \|\mathbf{x}\|=1}} \sqrt{\mathbf{x}^* \mathbf{A}^* \mathbf{A} \mathbf{x}}.$$

The spectral norm is invariant under transposition,

$$\|\mathbf{A}\| = \|\mathbf{A}^*\|$$

for any matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ (proof: use the SVD).

For orthonormal $\mathbf{Q} \in \mathbb{R}^{m \times n}$ ($\mathbf{Q}^* \mathbf{Q} = \mathbf{I}_n$) and any $\mathbf{x} \in \mathbb{R}^n$,

$$\|\mathbf{Q}\mathbf{x}\| = \sqrt{\mathbf{x}^* \mathbf{Q}^* \mathbf{Q} \mathbf{x}} = \|\mathbf{x}\|.$$

As a direct consequence, for arbitrary $\mathbf{F} \in \mathbb{R}^{n \times p}$,

$$\|\mathbf{Q}\mathbf{F}\| = \|\mathbf{F}^* \mathbf{Q}^*\| = \|\mathbf{F}\|.$$

Thus the Euclidean norm is invariant under orthonormal (not necessarily unitary, i.e., maybe non-square) transformations, and the spectral norm is invariant under multiplication by orthonormal columns from the left or by orthonormal rows from the right. (The same is true only for the Frobenius-norm, which we won't need here.)

Vector (sub-)spaces

We use uppercase letters like $\mathcal{S}, \mathcal{U}, \mathcal{Q}$ to denote subspaces of \mathbb{R}^n . A *nontrivial* subspace is one that is not equal to $\{\mathbf{o}\}$. For a set $X \subseteq \mathbb{R}^n$, the space of all linear combinations of vectors from X is denoted $\text{span } X$.

Two subspaces \mathcal{U} and \mathcal{V} are *complementary* if they have only the zero vector in common and sum to \mathbb{R}^n ,

$$\mathcal{U} + \mathcal{V} = \mathbb{R}^n, \quad \mathcal{U} \cap \mathcal{V} = \{\mathbf{o}\}.$$

Spaces are orthogonal, written $\mathcal{U} \perp \mathcal{V}$, if this holds for every pair of vectors from them, i.e., $\mathbf{u}^* \mathbf{v} = 0$ for all $\mathbf{u} \in \mathcal{U}, \mathbf{v} \in \mathcal{V}$. For each \mathcal{U} there is one, and only one, other space that is both orthogonal and complementary, aptly called the *orthogonal complement* of \mathcal{U} and written \mathcal{U}^\perp .

Useful relations for subspaces to have in one's toolbelt are the dimension rule

$$\dim(\mathcal{U} + \mathcal{V}) + \dim(\mathcal{U} \cap \mathcal{V}) = \dim \mathcal{U} + \dim \mathcal{V},$$

as well as

$$\dim \mathcal{U}^\perp = n - \dim \mathcal{U} \quad \text{and} \quad (\mathcal{U} + \mathcal{V})^\perp = \mathcal{U}^\perp \cap \mathcal{V}^\perp.$$

concerning orthogonal complements.

Eigenvalues and Eigenvectors

If for $\mathbf{A} \in \mathbb{C}^{n \times n}$ we have

$$\mathbf{A}\mathbf{q} = \lambda\mathbf{q}, \quad \mathbf{q} \in \mathbb{C}^n, \mathbf{q} \neq \mathbf{o}, \lambda \in \mathbb{C},$$

then λ is an *eigenvalue* of \mathbf{A} and for every nonzero scalar α the vector $\alpha\mathbf{q}$ is an *eigenvector* belonging to λ ; together $(\lambda, \alpha\mathbf{q})$ form an *eigenpair* of \mathbf{A} .

A transformation of the kind $\mathbf{A} \mapsto \mathbf{F}\mathbf{A}\mathbf{F}^{-1}$ is called *similarity transformation*. It leaves eigenvalues untouched and changes eigenvectors in a simple way, because, if (λ, \mathbf{q}) is an eigenpair of \mathbf{A} then $(\lambda, \mathbf{F}\mathbf{q})$ is an eigenpair of $\mathbf{F}\mathbf{A}\mathbf{F}^{-1}$.

Another useful transformation is *shifting*, $\mathbf{A} \mapsto \mathbf{A} - \tau\mathbf{I}$. An eigenpair (λ, \mathbf{q}) of \mathbf{A} becomes an eigenpair $(\lambda - \tau, \mathbf{q})$ of $\mathbf{A} - \tau\mathbf{I}$, so the main effect of shifting is that the eigenvalues are shifted. It will be ubiquitous in this thesis which is why we drop the identity,

$$\boxed{\mathbf{A} - \tau := \mathbf{A} - \tau\mathbf{I}.}$$

A matrix \mathbf{A} is similar to a diagonal matrix via a unitary similarity transformation (possibly complex) if, and only if, it is *normal*, meaning $\mathbf{A}^*\mathbf{A} = \mathbf{A}\mathbf{A}^*$. A special subclass of normal matrices are *real symmetric* ones. They have a full set of n real eigenvalues and one can find an orthonormal basis for \mathbb{R}^n consisting of eigenvectors.

From here on we assume that $\mathbf{A} \in \mathbb{R}^{n \times n}$ is symmetric.

The aforementioned similarity to a diagonal matrix gives the *eigendecomposition*

$$\boxed{\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^*}$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ contains the eigenvalues in ascending order and $\mathbf{Q} = [\mathbf{q}_1 | \dots | \mathbf{q}_n] \in \mathbb{R}^{n \times n}$ is unitary, $\mathbf{Q}^* \mathbf{Q} = \mathbf{I}$, having normalized eigenvectors as columns. Then \mathbf{A} can be regarded as sum of rank-1 matrices,

$$\mathbf{A} = \sum_{i=1}^n \lambda_i \mathbf{q}_i \mathbf{q}_i^*.$$

Notation. We use $\lambda_i[\mathbf{A}]$ and $\lambda_{-i}[\mathbf{A}]$ to denote the i th smallest and largest eigenvalue of \mathbf{A} , respectively, that is,

$$\begin{aligned} \lambda_1[\mathbf{A}] &\leq \lambda_2[\mathbf{A}] \leq \dots \leq \lambda_n[\mathbf{A}] \\ = &= &= \\ \lambda_{-n}[\mathbf{A}] &\leq \lambda_{-(n-1)}[\mathbf{A}] \leq \dots \leq \lambda_{-1}[\mathbf{A}]. \end{aligned}$$

Negating the matrix negates the eigenvalues, so an alternative formulation is

$$\lambda_{-i}[\mathbf{A}] = \lambda_{n-i+1}[\mathbf{A}] = -\lambda_i[-\mathbf{A}].$$

Based on the same indexing, $\mathbf{q}_{\pm i}[\mathbf{A}]$ denotes the corresponding member of an orthonormal basis of eigenvectors (be careful here in the face of multiple eigenvalues), so that

$$\boxed{(\mathbf{A} - \lambda_i[\mathbf{A}])\mathbf{q}_i[\mathbf{A}] \equiv \mathbf{o}, \quad \|\mathbf{q}_i[\mathbf{A}]\| \equiv 1.}$$

As is common practice, the symbol “ \equiv ” shall emphasize an equality that holds for all indices.

Invariant Subspaces. If \mathbf{A} maps a subspace \mathcal{S} to itself, $\mathbf{A}\mathcal{S} \subseteq \mathcal{S}$, then \mathcal{S} is called an *invariant subspace* of (for, under) \mathbf{A} . Equivalent would be to say that the space is spanned by a subset of \mathbf{A} 's eigenvectors. An *eigenspace* is a special kind of invariant subspace spanned by eigenvectors belonging to the same (multiple) eigenvalue—within an eigenspace, multiplication by \mathbf{A} is the same as multiplying by a scalar.

For an index set I ,

$$\mathcal{Q}_I[\mathbf{A}] := \text{span} \{ \mathbf{q}_i[\mathbf{A}] \mid i \in I \}$$

denotes the invariant subspace belonging to eigenvalues with index in I . Again we can use MATLABTM-notation here, like for any j ,

$$\begin{aligned} \mathcal{Q}_{:j}[\mathbf{A}] &= \mathcal{Q}_{\{1, \dots, j\}}[\mathbf{A}] = \text{span} \{ \mathbf{q}_1[\mathbf{A}], \dots, \mathbf{q}_j[\mathbf{A}] \}, \\ \mathcal{Q}_{j:}[\mathbf{A}] &= \mathcal{Q}_{\{j, \dots, n\}}[\mathbf{A}] = \text{span} \{ \mathbf{q}_j[\mathbf{A}], \dots, \mathbf{q}_n[\mathbf{A}] \}. \end{aligned} \tag{1.1}$$

The Spectrum. Let symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$ with eigenvalues $\lambda_i \equiv \lambda_i[\mathbf{A}]$ be fixed. All eigenvalues taken together give the *spectrum*

$$\text{spec}(\mathbf{A}) := \{\lambda_1, \dots, \lambda_n\}.$$

Nice to know is that multiplying all eigenvalues gives the determinant and summing them up gives the trace

$$\det(\mathbf{A}) = \prod_{i=1}^n \lambda_i, \quad \text{trace}(\mathbf{A}) = \sum_{i=1}^n \mathbf{A}(i, i) = \sum_{i=1}^n \lambda_i.$$

The spectral norm (also known as spectral *radius*) of a symmetric matrix equals the largest extremal eigenvalue in magnitude, or more concisely

$$\|\mathbf{A}\| = \max\{|\lambda_1|, |\lambda_n|\} = \max\{-\lambda_1, \lambda_n\}. \quad (1.2)$$

In some situations more appropriate than the spectral norm is the *spectral diameter*

$$\text{spdiam}[\mathbf{A}] := \lambda_n - \lambda_1,$$

because many results can be stated with respect to *gaps* in the spectrum, for which it is the smallest upper bound.

We can quantify the distribution of the spectrum by considering how distant the eigenvalues associated with an index set I (need not be a range) are to the rest, in either an absolute or relative sense, using

$$\text{gap}_{\mathbf{A}}(I) := \min\{|\lambda_j - \lambda_i| : i \in I, j \notin I\}, \quad (1.3)$$

$$\text{relgap}_{\mathbf{A}}(I) := \min\{|\lambda_j - \lambda_i|/|\lambda_i| : i \in I, j \notin I\}. \quad (1.4)$$

Both are considered undefined if $I = \{1, \dots, n\}$. For the relative gap we stipulate that $0/0 = 0$ (two zero eigenvalues, one in I , one not) and $x/0 = \infty$ for $x \neq 0$ (simple zero eigenvalue in I).

An alternative notion is that we have a sample value μ and want to know the smallest positive distance to an eigenvalue. This is captured by

$$\begin{aligned} \text{gap}_{\mathbf{A}}(\mu) &:= \min\{|\mu - \lambda| : \mu \neq \lambda \in \text{spec}(\mathbf{A})\}, \\ \text{relgap}_{\mathbf{A}}(\mu) &:= \min\{|\mu - \lambda|/|\mu| : \mu \neq \lambda \in \text{spec}(\mathbf{A})\}. \end{aligned} \quad (1.5)$$

Note that the two forms coincide if $\mu = \lambda_i$ is a simple eigenvalue, since then $\text{gap}_{\mathbf{A}}(\mu) = \text{gap}_{\mathbf{A}}(\{i\})$ and $\text{relgap}_{\mathbf{A}}(\mu) = \text{relgap}_{\mathbf{A}}(\{i\})$.

The relative gaps as they were defined above can be confusing, because the denominator varies over I . One would expect the minimum to be attained at the “border(s)” of I , but this might not be so. However, we are usually only interested in cases where $\text{relgap}_{\mathbf{A}}(I) < 1$. Then there must exist $i \in I, j \notin I$ with $\text{sign } \lambda_i = \text{sign } \lambda_j$. Based on this, a straightforward (but somewhat tedious) case distinction yields

$$\text{relgap}_{\mathbf{A}}(I) = \min\left\{|\lambda_j - \lambda_i|/|\lambda_i| : i \in I, j \notin I, |i - j| = 1\right\}. \quad (1.6)$$

1.2 Computing with Errors

1.2.1 Floating-Point Arithmetic

Computers can only handle a finite set of numbers. As consequence, computations have to be restricted to a finite subset of the reals, and a proper machine-compatible encoding for them has to be found. Unfortunately, no finite set that is useful can be closed under all arithmetic operations. So results will have to be rounded to the nearest representable number. In other words, we have to adapt to *finite precision*.

The following pages compress the core issues of finite precision arithmetic that concern us. Our exposition partially conforms with each of [43] and [32] (but not fully with both).

The encoding of choice (at least today) is scientific or *floating-point* notation. It can be characterized by four positive integer parameters: a *base* β (usually 2), a *precision* p and bounds $e_{\min} < e_{\max}$ to define a range of allowed exponents. With these fixed, the considered subset $\mathbb{F} \subset \mathbb{R}$ consists of all numbers x that can be written in the form

$$x = \pm 0.d_1 d_2 \dots d_p \beta^e, \quad d_i \in \{0, \dots, \beta - 1\}, \quad e_{\min} \leq e \leq e_{\max}, \quad (1.7)$$

or equivalently,

$$x = \pm m \beta^{e-p}, \quad m = (d_1 d_2 \dots d_p)_\beta = \sum_{i=1}^p d_i \beta^{p-i}. \quad (1.8)$$

The fractional part given by $m \in \mathbb{N}$ is called *mantissa* or *significand* and e is the *exponent*. To enforce uniqueness we require $m \geq \beta^{p-1}$ for nonzero x , that is, the first digit d_1 of the significand (or the most significant bit for $\beta = 2$) should be nonzero; such a representation is called *normalized*. Then we have $\beta^{p-1} \leq m \leq \beta^p - 1$ for all $x \neq 0$, so all representable numbers satisfy the constraint

$$x = 0 \quad \text{or} \quad \beta^{e_{\min}-1} \leq |x| \leq (1 - \beta^{-p}) \beta^{e_{\max}}. \quad (1.9)$$

A fundamental implication of using floating-point numbers is that they are not uniformly distributed in \mathbb{R} , because the absolute distance between neighboring numbers grows with their size. The smallest modification one can make to increase a positive x as in (1.8) without leaving the system is to add one to m , thus revolving around the last digit of the significand. Doing so effects an absolute change of

$$\text{ulp}(x) := 0.\underbrace{0\dots 01}_p \cdot \beta^e = \beta^{e-p},$$

called one *unit in the last place*. We just write ulp for $\text{ulp}(1)$. Note that $\text{ulp}(x)$ depends only on the exponent of x . Thus, for $x > 0$, the next floating-point number after x is

$$\text{next}(x) = x + \text{ulp}(x),$$

provided the operation does not leave the system, i.e., $x < (1 - \beta^{-p})\beta^{e_{\max}}$. Analogously, for $x > \beta^{e_{\min}-1}$, the floating-point number coming before x is

$$\text{prev}(x) = \begin{cases} x - \text{ulp}(x), & \text{if } m \neq \beta^{p-1}, \\ x - \text{ulp}(x)/\beta, & \text{if } m = \beta^{p-1}. \end{cases}$$

Measuring differences of floating-point numbers in terms of ulps is the right thing to do within the system \mathbb{F} , but the concept of ulp has no meaning for general real numbers. One way to interpret the embedding of \mathbb{F} within \mathbb{R} is to regard each $z \in \mathbb{R}$ as represented by its best approximation $\text{fl}(z) \in \mathbb{F}$. This concept is commonly known as rounding. A number $z \in \mathbb{R}$ lies within representable range if it satisfies the constraint (1.9). Assuming this is so and $z \neq 0$, we can write $z = \pm\mu\beta^{e-p}$ with $\beta^{p-1} \leq \mu < \beta^p$. Then rounding z to $\text{fl}(z)$ corresponds to rounding μ to an integer. Thus the absolute distance of z to the best approximation in \mathbb{F} satisfies

$$|\text{fl}(z) - z| \leq \frac{1}{2}\beta^{e-p}.$$

Hence, the relative error incurred by rounding will obey

$$\frac{|\text{fl}(z) - z|}{\min\{|\text{fl}(z)|, |z|\}} \leq \frac{\frac{1}{2}\beta^{e-p}}{\beta^{e-1}} = \frac{1}{2}\beta^{1-p} = \frac{1}{2}\text{ulp}(1) =: \epsilon_{\text{rep}},$$

defining the *maximal relative representation error* ϵ_{rep} , also known as *machine epsilon*. Then we can state

$$\text{fl}(z) = z(1 + \gamma) = z/(1 + \delta) \quad \text{with} \quad |\gamma|, |\delta| \leq \epsilon_{\text{rep}}. \quad (1.10)$$

Some authors characterize ϵ_{rep} as the smallest floating-point number that added to one gives a floating-point number other than one. Depending on how ties are broken this might not be identical to the definition above. A better formulation denotes ϵ_{rep} as half the distance between one and the next larger floating-point number.

There seems to be no real consensus in the community on what machine epsilon should be. Our ϵ_{rep} is the one used by Goldberg [32], whereas Higham [43] calls $\beta^{1-p}/2$ *unit roundoff* and defines machine epsilon to be twice that (which is the full distance between one and the next larger floating-point number). With regard to numerical software, `xLAMCH('E')` from LAPACK returns ϵ_{rep} as machine epsilon but the `eps` in MATLABTM is $2\epsilon_{\text{rep}}$.

At this point we make a step back. For analyzing the algorithms in this thesis, we generally do not need to know how floating-point numbers look like

Axiom FP

The floating-point system provides the arithmetic operations $+$, $-$, $*$, $/$ and $\sqrt{\cdot}$ over a (sensible) set \mathbb{F} of machine numbers. The implementation of the arithmetic obeys that when an operation is applied to arguments from \mathbb{F} for which the exact result $z \in \mathbb{R}$ is defined, exactly one of three things may happen:

- The computation is marked as overflow, indicating that $|z|$ exceeds the overflow threshold o_\diamond .
- The computation is marked as underflow, indicating that $0 < |z| < u_\diamond$, where u_\diamond is the underflow threshold.
- A number $x \in \mathbb{F}$ is returned that satisfies

$$x = z(1 + \gamma) = z/(1 + \delta), \quad |\gamma|, |\delta| \leq \epsilon_\diamond,$$

with machine epsilon ϵ_\diamond .

The three parameters o_\diamond , u_\diamond and ϵ_\diamond are supposed to be uniformly chosen so that the above holds for all arguments from \mathbb{F} .

or even what base is chosen. Instead it suffices that the operations executed on the machine flag results that would not be representable and otherwise behave in principle as if they were computed exactly and then rounded, such that (1.10) holds for some uniform constant ϵ_{rep} . These requirements are formulated as Axiom FP on this page.

Remark 1.1 (About Axiom FP). The exact result might exceed o_\diamond without causing overflow if it is rounded to o_\diamond , and similar for underflow. Furthermore, for underflow and overflow, nothing is said about the produced result of the operation, it might well but need not be undefined. This is done deliberately to encompass a broader range of implementations. For instance, both the use of denormalized numbers or flushing the result to zero would be acceptable treatments of underflow. We merely need some way of telling that an exceptional case has occurred.

The system \mathbb{F} will rarely be mentioned, because we will implicitly assume all inputs for algorithms to be representable floating-point numbers. Provided no underflow or overflow occurs during execution, the same will then be true for intermediate and final results of the computation. \diamond

	single precision 32 bit	double precision 64 bit
β	2	2
p	24	53
e_{\min}	-125	-1021
e_{\max}	+128	+1024
u_{\diamond}	2^{-126} $\approx 1.8 \cdot 10^{-38}$	2^{-1022} $\approx 2.2 \cdot 10^{-308}$
ϵ_{\diamond}	2^{-24} $\approx 6.0 \cdot 10^{-8}$	2^{-53} $\approx 1.1 \cdot 10^{-16}$
o_{\diamond}	$(1 - 2^{-24}) \cdot 2^{128}$ $\approx 3.4 \cdot 10^{38}$	$(1 - 2^{-53}) \cdot 2^{1024}$ $\approx 1.8 \cdot 10^{308}$

Table 1.1: Formats defined by the IEEE 754 standard and the parameters with which they fulfill Axiom FP.

The IEEE Standard

Most modern architectures adhere to the IEEE 754 standard for floating-point arithmetic which was first established in 1985 [44] and recently revised [45]. It requires that all operations produce exactly rounded results, that is, $\epsilon_{\diamond} = \epsilon_{\text{rep}}$. The original standard defines two formats **single precision** and **double precision**. Table 1.1 summarizes the respective parameters of these formats and how they fulfill Axiom FP. Notable features of the standard are the use of *denormalized* numbers to provide for *gradual* underflow, signed infinities $\pm\infty \in \mathbb{F}$ for overflow and special constants called *not-a-number* (NaN) to indicate undefined results (like $0/0$ or $\infty - \infty$). For more information, see [32, 49].

1.2.2 Condition of Addition & Cancellation

The field of numerical analysis is driven by the ambition to let machines compute numbers for which we know beforehand what they will mean. Generally, suitable models are used to reduce the concrete application area at hand (e.g., the optimization of a plane-wing's aerodynamics) to one or more *problems* with an underlying well-defined mathematical relation (like a linear system, differential equations or the eigensystem of a matrix).

Given (sets of) inputs to the problem we want to compute outputs which reflect the true result as accurately as is possible with reasonable use of computing resources. It depends on the application to define when a solution is accurate enough.

There are basically two kinds of obstacles in our way. The first is that we

cannot do exact arithmetic on a machine or even represent each real number, as we have seen in the previous section. The second obstacle is in fact more subtle and concerns the fact that the input data we get will generally be inexact, due to inaccurate measurements in the “real” world. So, before we start thinking about computing outputs, we have to ask ourselves if the problem is well-posed, that is, is it even possible to compute nearly right outputs from slightly wrong inputs? If the answer to this question is yes, we call the problem *well-conditioned*, otherwise *ill-conditioned*. This notion of a problem’s *condition* is completely independent of any algorithm that might be deployed to solve it.

In the previous section we did already brush the dichotomy of relative vs. absolute error. In the context of floating-point numbers, relative errors are way more meaningful and should always be preferred because they indicate roughly the number of correct digits in a result. For this thesis, relative errors and relative condition numbers are far more prominent than absolute ones.

At this point we do not want to give a general definition of what a problem and its absolute and relative condition numbers are. Those can be found in any introductory textbook. What we will do is focus on the relative condition of a specific problem which seems trivial but is, arguably, one of the most influential: the addition of two real numbers. Let $x, y \in \mathbb{R}$ be fixed and assume $x + y \neq 0$. For small relative perturbations ξ, γ —think of them as about $\mathcal{O}(\epsilon_\diamond)$ —we surely have

$$\begin{aligned} x(1 + \xi) + y(1 + \gamma) &= (x + y) \left(1 + \frac{x\xi + y\gamma}{x + y} \right) \\ &= (x + y)(1 + \zeta), \end{aligned}$$

with

$$|\zeta| \leq \max\{|\xi|, |\gamma|\} \cdot \frac{|x| + |y|}{|x + y|}.$$

This reveals that the relative change in the result is controlled by the quotient $(|x| + |y|)/|x + y|$, which is just the condition of adding x and y . The resulting condition number is important enough to warrant its own name. Note that our initial assumption $x + y \neq 0$ can be dropped by allowing the condition to reach infinity.

Definition 1.2. For any $x, y \in \mathbb{R}$ define

$$\kappa_+(x, y) := \frac{|x| + |y|}{|x + y|}, \quad \kappa_-(x, y) := \frac{|x| + |y|}{|x - y|}.$$

For both cases, $z/0$ may evaluate to 1 if $z = 0$ and to ∞ otherwise. \diamond

The separate definition of $\kappa_-(x, y)$ is just there to avoid having to write ugly things like $\kappa_+(x, -y)$. Let us note some fairly obvious properties of κ_+ .

Lemma 1.3. For any $x, y \in \mathbb{R}$:

- (i) $\kappa_-(x, y) = \kappa_+(x, -y)$.
- (ii) $\kappa_+(x, y) \geq 1$.
- (iii) $\kappa_+(x, y) = 1 \Leftrightarrow (\text{sign } x \cdot \text{sign } y) \neq -1$.
- (iv) $\kappa_+(\alpha x, \alpha y) = \kappa_+(x, y)$ for all $\alpha \in \mathbb{R} \setminus \{0\}$.

It is also worthwhile to state explicitly the connection between the condition and the effect of input data perturbations on the result.

Lemma 1.4. For any scalars $x, y, \xi, \gamma \in \mathbb{R}$ with $x + y \neq 0$

$$x(1 + \xi) + y(1 + \gamma) = (x + y)(1 + \zeta)$$

for some $\zeta \in \mathbb{R}$ with $|\zeta| \leq \kappa_+(x, y) \max\{|\xi|, |\gamma|\}$.

One interesting remark is that we did not (need to) make any assumptions about the size of the perturbation quantities ξ and γ , although we will mainly use this result in a context where they are of magnitude $\mathcal{O}(\epsilon_\circ)$.

The intuitive notion is that if $\kappa_-(x, y)$ is large, then x and y must be close in a relative sense. For floating-point numbers this would mean that x and y agree to some of their leading digits, so evaluating $x - y$ involves *cancellation*. The next result captures this.

Lemma 1.5. Let $x, y \in \mathbb{R}$ with $|x| < |y|$ and $\kappa_+(x, y) > 1$. Then

- (i) $\frac{|x|}{\kappa_+(x, y) - 1} = \frac{|x + y|}{2} = \frac{|y|}{\kappa_+(x, y) + 1}$,
- (ii) $|x|(\kappa_+(x, y) + 1) = |y|(\kappa_+(x, y) - 1)$.

Proof. From Lemma 1.3 we get $\text{sign } x = -\text{sign } y$. Together with $|x| < |y|$ this gives $|x + y| = |y| - |x|$. Now both claims are immediate. \square

The result can be employed backwards to use information about x and y to obtain a bound on κ_+ . For example, Lemma 1.5 leads to

$$|x| < K|y|, 0 < K < 1 \implies \kappa_+(x, y) < \frac{1 + K}{1 - K}. \quad (1.11)$$

This formula captures the notion of *cancellation*: Adding numbers of different signs that are of about the same magnitude ($K \approx 1$) is badly conditioned and can therefore lose information.

1.2.3 A Toolset for Error Analysis

The floating-point model introduced previously allows us to bound errors in computations at the most basic level. In this section we develop a custom notation and some tools to streamline working with those bounds.

Readers familiar with componentwise relative error analysis but not that interested in the details of our notation can probably safely skip most of this section. The important points that will be needed later on are the Chart 1.11 on page 20 and Corollary 1.12. In most minimal terms the notation we will use is captured in (1.19).

An Example

Consider the innocuous computation

$$y := (a + 1) \cdot b^2 - c. \quad (1.12)$$

The symbol “ $:=$ ” is how we write down computational statements in algorithms.

If (1.12) is executed by a machine satisfying Axiom FP (where we assume the data a, b, c to be given as floating-point values) the computed result y can be described as

$$y = [(a + 1)(1 + \alpha_+)b^2(1 + \alpha_{\text{sq}})(1 + \alpha_*) - c](1 + \alpha_-) \quad (1.13)$$

with $|\alpha_+|, |\alpha_{\text{sq}}|, |\alpha_*|, |\alpha_-| \leq \epsilon_\diamond$. Most of the time we will use fittingly subscripted Greek letters ($\alpha, \beta, \gamma, \dots$) to denote (small) relative errors or perturbations. In this case it was possible to find subscripts that link the errors to the operations that caused them, although this will not always be feasible or necessary. Note that Axiom FP would have allowed us to place the perturbation $(1 + \alpha_-)$ on the left hand side of the equation, but for this example we chose not to.

What can we say about the computed result y in (1.13)? Well, in order to link y to what would be the exact result $(a + 1) \cdot b^2 - c$, that is, to do what is formally called a *forward error analysis*, we would need to consider the *condition* of the subtraction involved, which might be bad if there is cancellation.

However, we can also look at (1.13) from a slightly different angle and see it as

$$y(1 + \alpha_-)^{-1} = (a + 1) \left[b \sqrt{(1 + \alpha_+)(1 + \alpha_{\text{sq}})(1 + \alpha_*)} \right]^2 - c.$$

Note that we implicitly exploited $\epsilon_\diamond < 1$ here in order to put the perturbations under the square root. With

$$\tilde{y} := y(1 + \alpha_-)^{-1} \quad \text{and} \quad \tilde{b} := b \sqrt{(1 + \alpha_+)(1 + \alpha_{\text{sq}})(1 + \alpha_*)} \quad (1.14)$$

we arrive at

$$\tilde{y} = (a + 1)\tilde{b}^2 - c.$$

Basically, this relation reveals that if we change the input data b and the output y according to (1.14), the perturbed values do satisfy the desired relation *exactly*. What we just did is called (*componentwise*) *mixed relative error analysis*; it is a powerful technique and will play a prominent role in this work.

Standard Approaches

In general, we call an algorithm *mixed relatively stable*, or just *stable*, if it allows for a mixed relative error analysis and the resulting relative errors to inputs and outputs are “small”. With rudimentary knowledge of infinite series we can interpret (1.14) to get the crude bounds

$$\tilde{y} = y(1 + \gamma + \mathcal{O}(\epsilon_\diamond^2)), \quad |\gamma| \leq \epsilon_\diamond, \quad \tilde{b} = b(1 + \beta + \mathcal{O}(\epsilon_\diamond^2)), \quad |\beta| \leq \frac{3}{2}\epsilon_\diamond. \quad (1.15)$$

The usual way is just to forget about any $\mathcal{O}(\epsilon_\diamond^2)$ terms and read this as “perturb y and b each by about ϵ_\diamond ($\approx .5\text{ulp}$)”. For instance this is how results are presented in [15, 18, 35–37].

Once you start using them, $\mathcal{O}(\epsilon_\diamond^2)$ -terms tend to spawn rapidly until they occur almost everywhere. This is not only unsatisfactory but plain ugly. Thus we will use two techniques to handle relative perturbation factors more fluently.

To motivate the following, let us cast what we did in the example above into more general terms. The sole origin of perturbations is Axiom FP and most of the time one has to juggle around factors $(1 + \alpha)$, $|\alpha| \leq \epsilon_\diamond$. One could call these *basic* or *elementary* perturbations. It is possible to accumulate them by multiplying, dividing and taking square roots. This can take you quite far; in the example above until (1.14). In general it will lead to something like

$$\tilde{x} = x \prod_i \frac{(1 + \alpha_i)^{s_i}}{(1 + \beta_i)^{t_i}}, \quad |\alpha_i|, |\beta_i| \leq \epsilon_\diamond, \quad s_i, t_i \in [0, 1]. \quad (1.16)$$

But there comes a point when one wants to write $\tilde{x} = x(1 + \xi)$ again with a concise bound for ξ , that is, for $|\tilde{x}/x - 1|$. First-order perturbation theory would give $|\tilde{x}/x - 1| \leq \epsilon_\diamond \sum_i (s_i + t_i) + \mathcal{O}(\epsilon_\diamond^2)$.

A Refined Notation

Higham uses in his book [43] terms of the form $\gamma_n = n\epsilon_\diamond/(1 - n\epsilon_\diamond)$ for accumulating relative perturbation factors. The denominator $(1 - n\epsilon_\diamond)$ takes care of all second-order contributions in a clearly defined way. We decided to adopt his approach. However, the bounds γ_n do not lend themselves well for taking square roots of errors like in (1.14), nor for dividing them. For instance, the best one can do to bound $(1 + \gamma_k)/(1 + \gamma_j)$ is by $(1 + \gamma_{k+2j})$ (cf. [43, Lemma 3.3]), which overestimates the first-order contribution by $j\epsilon_\diamond$. Therefore we developed a slight generalization. Using our notation is a little more cumbersome, but it gives

sharper first-order bounds. The essence is that we have to keep better track of second-order contributions, not because we are interested in them, but because neglecting them would cause the first-order bounds to suffer.

The type of bounds we will use are of the form $x/(1 - kx)$ for small x , e.g., $x = n\epsilon_\diamond$. To analyze their properties let us, for $k \in \mathbb{N}_0$, define the functions

$$\Theta_k : \left[0, \frac{1}{\max\{1, k\}}\right) \rightarrow \mathbb{R}, \quad x \mapsto \frac{x}{1 - kx}. \quad (1.17)$$

Note that Θ_0 is just the identity on $[0, 1)$. The restriction on the definition domain is just a formal necessity since we are interested in $\Theta_k(x)$ only for tiny arguments x . The salient feature of $\Theta_k(x)$ is that it is strictly monotonically increasing in both k and x . Furthermore we define

$$\theta_k(x) := \text{placeholder for quantity bounded by } \Theta_k(x) \text{ in magnitude.} \quad (1.18)$$

By “placeholder” we mean that occurrences of θ_k -terms should be interpreted similarly to the traditional $\mathcal{O}(\cdot)$ notation. To pronounce relations that deal with unspecified quantities like these we will write “ \doteq ” instead of “ $=$ ”. The expressions become unambiguous if interpreted from left to right.

Figure 1.1 shows plots of Θ_k for $k = 1, 2, 4$ as well as a possible incarnation of θ_1 . The following theorem is of rather technical nature but forms the foundation of our notation.

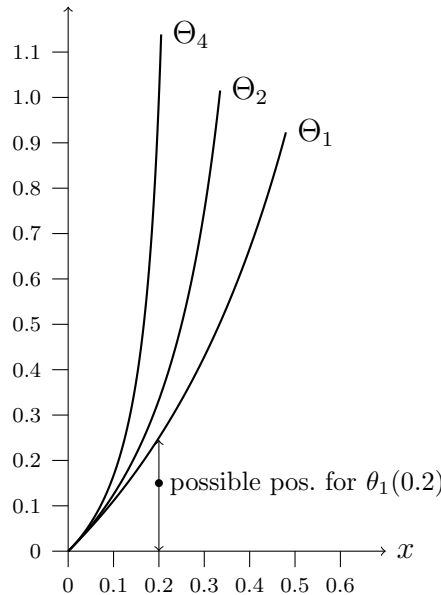


Figure 1.1: Functions Θ_k and placeholders θ_k underlying the notation for error analysis.

Theorem 1.6 (Properties of θ_k)

With the notation developed above, and provided all θ_k terms are defined according to (1.17) and (1.18), the following holds:

- (i) $(1 + \theta_k(x))(1 + \theta_j(y)) \doteq (1 + \theta_{\max\{k,j\}}(\max\{x, y\}))^2,$
- (ii) $(1 + \theta_k(x))(1 + \theta_j(y)) \doteq 1 + \theta_{\max\{1,k,j\}}(x + y),$
- (iii) $(1 + \theta_k(x))^s \doteq 1 + \theta_{(k+1)/|s|}(|s|x), \quad 0 \neq |s| \leq 1,$
- (iv) $\theta_k(x) \pm \theta_j(y) \doteq \theta_{\max\{1,k,j\}}(x + y),$
- (v) $\lambda\theta_k(x) \doteq \theta_k(\lambda x) \quad \text{for all } \lambda \geq 1.$

The proof will require a couple of lemmas for support.

Lemma 1.7. For any $x, s \in \mathbb{R}$ with $|x| < 1$ and $|s| \leq 1$ we have

$$|(1+x)^s - 1| \leq \frac{|sx|}{1-|x|}.$$

Proof. Recalling the general binomial coefficients

$$\binom{s}{k} := \begin{cases} s(s-1)\cdots(s-k+1)/k!, & \text{if } k > 0, \\ 1, & \text{if } k = 0, \\ 0, & \text{otherwise,} \end{cases}$$

we employ the general binomial expansion to obtain

$$\begin{aligned} (1+x)^s &= \sum_{k=0}^{\infty} \binom{s}{k} x^k \\ &= 1 + sx \sum_{k=0}^{\infty} \frac{(s-1)(s-2)\cdots(s-k)}{2 \cdot 3 \cdots (k+1)} x^k. \end{aligned}$$

Since $|s| \leq 1$, the coefficients of x^k in the latter sum are bounded by one in magnitude. Hence the geometric series $\sum |x|^k = (1-|x|)^{-1}$ is a majorant. \square

Lemma 1.8. Let $a, b \in \mathbb{R}$ and $|s| \leq 1$. Then, for any $\alpha \in \mathbb{R}$,

$$|\alpha| \leq \frac{a}{1-b} < 1 \quad \implies \quad |(1+\alpha)^s - 1| \leq \frac{|s|a}{1-(a+b)}.$$

Proof. Apply Lemma 1.7 and note that $|x|/(1-|x|) = \Theta_1(|x|)$ is monotonically increasing. \square

Lemma 1.9. *Let $a, b, c, d \in \mathbb{R}$ with $0 \leq a \leq b$, $0 \leq c \leq d$ and $b + d < 1$. Then, for any $\alpha, \beta \in \mathbb{R}$,*

$$|\alpha| \leq \frac{a}{1-b} \wedge |\beta| \leq \frac{c}{1-d} \implies |(1+\alpha)(1+\beta) - 1| \leq \frac{a+c}{1-(b+d)}.$$

Proof.

$$\begin{aligned} |(1+\alpha)(1+\beta) - 1| &= |\alpha + \beta + \alpha\beta| \leq \frac{a}{1-b} + \frac{c}{1-d} + \frac{ac}{(1-b)(1-d)} \\ &= \frac{a+c - (ad+bc-ac)}{1-(b+d)+bd}. \end{aligned}$$

From the prerequisites we can infer $0 \leq bd < b+d$ as well as

$$0 \leq (ad+bc-ac) = a(d-c) + bc < a+c$$

and the result follows. \square

Proof of Theorem 1.6. Remember that the claims only concern nonnegative arguments x and y .

(i). This is just the monotonicity of $\Theta_k(x)$ in k and x .

(ii). For $k \geq 1$, $j \geq 1$ we employ Lemma 1.9 to get

$$\begin{aligned} \left| (1 + \theta_k(x))(1 + \theta_j(y)) - 1 \right| &\leq \frac{x+y}{1-(kx+jy)} \\ &\leq \frac{x+y}{1-\max\{k,j\}(x+y)} = \Theta_{\max\{k,j\}}(x+y). \end{aligned}$$

The special cases $k = 0$ or $j = 0$ follow then as well, since we can exploit $\Theta_0(x) \leq \Theta_1(x)$ for all $x \in [0, 1)$, that is, $\theta_0(x) \doteq \theta_1(x)$.

(iii). Lemma 1.8 gives

$$\left| (1 + \theta_k(x))^s - 1 \right| \leq \frac{|s|x}{1-(k+1)x} \stackrel{s \neq 0}{\leq} \frac{|s|x}{1-\frac{k+1}{s}sx} = \Theta_{(k+1)/|s|}(|s|x).$$

(iv). This follows from (ii), since for scalars $\alpha \doteq \theta_k(x), \beta \doteq \theta_j(y)$ we have

$$|\alpha \pm \beta| \leq |\alpha| + |\beta| + |\alpha\beta| = |(1+|\alpha|)(1+|\beta|) - 1|.$$

(v). Use that $1/(1-z) \leq 1/(1-\lambda z)$ for all $z \leq \lambda z < 1$. \square

Now we can define our notation for running error analysis. As we are effectively only going to need perturbations expressed as multiple of ϵ_\diamond , we will define a special notation for those, namely

$$\boxed{\epsilon^{[k]}(n) := \theta_k(n\epsilon_\diamond)}.$$

The following definition captures this and the subsequent chart compiles the properties we are going to need, all of which are immediate consequences of Theorem 1.6. The Θ_k 's are completely left out from the presentation, since their sole purpose was to get us here. Naturally, Theorem 1.6 would also offer an analogon to the general exponentiation rule (iii) for the $\epsilon^{[\cdot]}(\cdot)$ -terms. But to ease later reference to the chart we have opted to forego it. Instead, we did explicitly split off only the two special rules that will be needed, namely (3) for inversion and (4) for taking square roots.

Definition 1.10. For any $p, n \in \mathbb{R}^{\geq 0}$ and $0 \leq pn\epsilon_\diamond < 1$, we define

$$\epsilon^{[p]}(n) := \text{placeholder for quantity } \alpha \text{ with } |\alpha| \leq \frac{n\epsilon_\diamond}{1 - pn\epsilon_\diamond}.$$

We abbreviate $\epsilon^{[1]}(n) =: \epsilon(n)$ and write ϵ_\diamond instead of $\epsilon^{[0]}(1)$ if in placeholder context (on the right hand side of an “ \doteq ”). \diamond

Chart 1.11 (Rules for running error analysis)

Let $r = \max\{p, q\}$, $R = \max\{1, p, q\}$.

$$\begin{aligned} (1) \quad & (1 + \epsilon^{[p]}(n))(1 + \epsilon^{[q]}(m)) \doteq (1 + \epsilon^{[r]}(\max\{m, n\}))^2 \\ (2) \quad & (1 + \epsilon^{[p]}(n))(1 + \epsilon^{[q]}(m)) \doteq 1 + \epsilon^{[R]}(n + m) \\ (3) \quad & (1 + \epsilon^{[p]}(n))^{-1} \doteq 1 + \epsilon^{[p+1]}(n) \\ (4) \quad & (1 + \epsilon^{[p]}(n))^{1/2} \doteq 1 + \epsilon^{[2p+2]}(\frac{1}{2}n), \\ (5) \quad & \epsilon^{[p]}(n) \pm \epsilon^{[q]}(m) \doteq \epsilon^{[R]}(n + m) \\ (6) \quad & \lambda\epsilon^{[p]}(n) \doteq \epsilon^{[p]}(\lambda n), \lambda \geq 1 \end{aligned}$$

We need to stress, again, that although the use of $\epsilon^{[\cdot]}(\cdot)$ -terms as placeholder for an unspecified quantity simplifies the notation in general, it may cause ambiguity if employed carelessly. Just as with the traditional \mathcal{O} -notation, expressions which contain placeholders should be interpreted from left to right. Consider rule (6) in the previous chart: It would have been erroneous to state it the other way around since $\lambda n\epsilon_\diamond / (1 - p\lambda n\epsilon_\diamond) > \lambda n\epsilon_\diamond / (1 - pn\epsilon_\diamond)$ for $\lambda > 1$.

To demonstrate the power of the notation, the following corollary shows how accumulated elementary perturbations like those in (1.16) can be handled rather effortlessly.

Corollary 1.12 (Accumulation of basic rounding errors).

Given for $i = 1, \dots, n$ scalars $m_i \in \mathbb{R}$ and $s_i \in \{-1, 1\}$, let $D := \sum_{i=1}^n |m_i|$. Then, provided $D < 1/\epsilon_\diamond$,

$$\prod_{i=1}^n (1 + m_i\epsilon_\diamond)^{s_i} \doteq 1 + \epsilon(D).$$

Proof. Because $|s_i|$ is limited to one, we can with help of Lemma 1.8 rewrite the factors as $(1 + m_i \epsilon_\diamond)^{s_i} \doteq 1 + \epsilon(|m_i|)$. Then the result follows from rule (2). \square

Note that if one would bound each elementary perturbation from the start by $(1 + m_i \epsilon_\diamond) \doteq 1 + \epsilon(|m_i|)$, then using rules (2) and (3) the bound would become $1 + \epsilon^{[2]}(D)$. The difference concerns only second-order contributions, so a pragmatic reader might remark that this seeming improvement provided by the corollary is of no practical consequence whatsoever.

We want to leave this section by taking a second look at example (1.12) from before. As was done there, a tilde superscript is reserved to denote perturbed quantities.

To provide for a more visual presentation we use a zigzag-arrow \rightsquigarrow to pronounce the transition from unperturbed to perturbed data. Furthermore, for perturbations of scalars we also identify $x \rightsquigarrow \tilde{x}$ with the relative factor involved; formally defined this means

$$x \rightsquigarrow \tilde{x} := (\tilde{x} - x)/x, \quad x \neq 0.$$

So, $\tilde{x} = x(1 + \xi)$ is the same as $x \rightsquigarrow \tilde{x} = \xi$.

For the perturbations in (1.14) we would use Corollary 1.12 to compress the factors under the square root as $(1 + \epsilon(3))$, and then use rules (3) and (4) from Chart 1.11 to obtain

$$y \rightsquigarrow \tilde{y} \doteq \epsilon(1), \quad b \rightsquigarrow \tilde{b} \doteq \epsilon^{[4]}(\frac{3}{2}).$$

As closing remark, it is nice to have control over second-order terms, but rarely of practical relevance. Most readers will probably be already familiar with the traditional use of the \mathcal{O} -notation. To ease the transition for them the rule

$$\boxed{\epsilon^{[k]}(n) = n\epsilon_\diamond + k(n\epsilon_\diamond)^2 + \mathcal{O}(\epsilon_\diamond^3)} \quad (1.19)$$

really covers everything one needs to keep in mind.

1.3 Angles between subspaces

For understanding the MR³-algorithm we will have to analyze how invariant subspaces of matrices are affected by perturbations. To do so, we need to develop a strong grip on the concept of *angles* between subspaces. This topic can become rather tricky to handle in a general way, as one inevitably has to venture beyond 3-dimensional space where intuition and examples might help. On top of that, treatment in the literature [9, 34, 55, 56] is somewhat varying in depth and often strongly biased by the authors' intended application.

Therefore we decided to give a stand-alone presentation of the topic. Experienced readers might want to flip ahead to the results starting on page 25 to

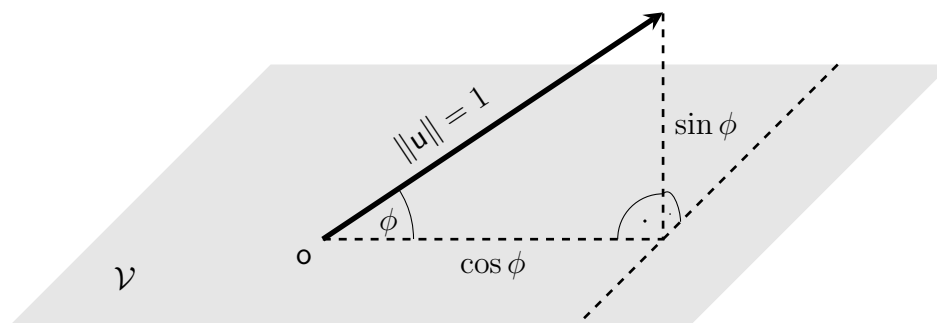


Figure 1.2: Geometrical interpretation of an angle between a vector (line) and a 2-dimensional subspace (plane) in \mathbb{R}^3 .

evaluate the potential gain from studying this section. The notion of angle we will end up using is just the *largest canonical angle* as defined in [34], see the remarks at the end of this section.

Our presentation up to Definition 1.14 was strongly influenced by [55, §5.15].

The standard scalar product with which \mathbb{R}^n was embodied to become the Euclidean space carries with it the concept of an *angle* between vectors, or more precisely, between one-dimensional subspaces. Let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ have unit norm, then

$$\cos \angle(\text{span}\{\mathbf{u}\}, \text{span}\{\mathbf{v}\}) = |\mathbf{u}^* \mathbf{v}| \quad (1.20)$$

defines this angle, measured as acute in $[0, \pi/2]$ for uniqueness. For brevity we will just write $\angle(\mathbf{u}, \mathbf{v})$ given nonzero \mathbf{u} and \mathbf{v} .

If we lift the restriction of being one-dimensional from one of the spaces, say by replacing $\mathbf{v} \in \mathbb{R}^n$ with $\mathcal{V} \subseteq \mathbb{R}^n$, it becomes natural to ask for the smallest rotation to bring \mathbf{u} into \mathcal{V} . This gives rise to the *minimal angle* ϕ between those two, defined by

$$\cos \phi = \max \{ |\mathbf{u}^* \mathbf{v}| : \mathbf{v} \in \mathcal{V}, \|\mathbf{v}\| = 1 \}, \quad (1.21)$$

or equivalently,

$$\begin{aligned} \sin \phi &= \min \{ \|\mathbf{u} - \mathbf{v}\| : \mathbf{v} \in \mathcal{V} \} \\ &= \max \{ |\mathbf{u}^* \mathbf{w}| : \mathbf{w} \in \mathcal{V}^\perp, \|\mathbf{w}\| = 1 \}. \end{aligned} \quad (1.22)$$

Both definitions are conceptually based on orthogonal decompositions, see Figure 1.2.

However, for making the next step to define an angle between general subspaces $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^n$, the intuition that brought us thus far can become misleading. At first one could try to define a minimum angle

$$\cos \theta_m := \max \{ |\mathbf{u}^* \mathbf{v}| : \mathbf{u} \in \mathcal{U}, \mathbf{v} \in \mathcal{V} \}, \quad (1.23)$$

only to realize θ_m will be zero as soon as the spaces have a non-trivial intersection. The alternative approach to define a maximum angle asymmetrically via

$$\sin \theta_M := \max \{ |u^* w| : u \in \mathcal{U}, w \in \mathcal{V}^\perp \}, \quad (1.24)$$

does not work either, because then $\theta_M = \pi/2$ whenever $\dim \mathcal{U} > \dim \mathcal{V}$ (we will prove this in Lemma 1.13 below).

Projectors

To work with arbitrary subspaces we first need a way to identify them. One could employ any basis for this purpose, but that would not be unique. Better is to use that each subspace has a unique linear operator associated with it—its *orthogonal projector*.

Any idempotent matrix $P = P^2$ is called a *projector*. Then

$$I - P = I - 2P + P^2 = (I - P)(I - P)$$

is also a projector. For any $x \in \mathbb{R}^n$ we have $P^2 x = P x$, so $\text{range}(P) = \text{null}(I - P)$ and $\text{null}(P) = \text{range}(I - P)$. Hence, for every projector P ,

$$\text{range}(P) + \text{null}(P) = \mathbb{R}^n, \quad \text{range}(P) \cap \text{null}(P) = \{0\},$$

meaning $\text{range}(P)$ and $\text{null}(P)$ are *complementary subspaces*.

Conversely, for any two complementary subspaces \mathcal{R} and \mathcal{N} , there is a unique linear operator whose matrix P (with respect to the standard basis on \mathbb{R}^n) combines the properties $P^2 = P$, $\text{range}(P) = \mathcal{R}$ and $\text{null}(P) = \mathcal{N}$. In this situation we say that P *projects onto \mathcal{R} along \mathcal{N}* .

A projector is called *orthogonal* if its range and nullspace are, well, orthogonal, otherwise it is *oblique*. The preceding thoughts show that there is a one-to-one correspondence between a subspace \mathcal{S} and an orthogonal projector $P_{\mathcal{S}}$ that projects onto \mathcal{S} (along \mathcal{S}^\perp), thus we can identify the space \mathcal{S} uniquely with the matrix $P_{\mathcal{S}}$.

There is a nice connection between orthogonal projectors and orthonormal bases of the subspace. For any S with orthonormal columns spanning \mathcal{S} , we clearly have $\text{range}(SS^*) = \mathcal{S}$ and $\text{null}(SS^*) = \mathcal{S}^\perp$. Hence, as the orthogonal projector onto \mathcal{S} is unique,

$$P_{\mathcal{S}} = SS^* \quad \text{and} \quad P_{\mathcal{S}^\perp} = I - SS^*. \quad (1.25)$$

An immediate consequence is

$$\|P\| = 1 \quad \text{for any orthogonal projector } P. \quad (1.26)$$

Furthermore, any symmetric projector, $P = P^*$, must be orthogonal, because $(I - P)^* P = (I - P) P = 0$. Together with (1.25) we get the elegant characterization that orthogonal projectors are just the symmetric ones, i.e.,

$$\text{range}(P) \perp \text{null}(P) \quad \Leftrightarrow \quad P = P^*. \quad (1.27)$$

In the remainder of this thesis we will exclusively be dealing with orthogonal projectors, so from here on, we will just say projector when we mean orthogonal projector.

Distance between subspaces

Given subspaces $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^n$, one way to gauge their difference is to measure the maximal distance of a vector in one space, say \mathcal{U} , to its (orthogonal) projection onto the other space. This leads to

$$\delta(\mathcal{U}, \mathcal{V}) := \max_{\substack{\mathbf{u} \in \mathcal{U} \\ \|\mathbf{u}\|=1}} \|(I - P_{\mathcal{V}})\mathbf{u}\|. \quad (1.28)$$

Note that this definition of distance is similar to (1.24) but *not* symmetric. Using that orthogonal projectors are symmetric, we can rewrite it

$$\delta(\mathcal{U}, \mathcal{V}) = \|(I - P_{\mathcal{V}})P_{\mathcal{U}}\| = \|P_{\mathcal{U}}(I - P_{\mathcal{V}})\|. \quad (1.29)$$

Take any unitary matrices $Q_{\mathcal{U}} = [U|\bar{U}]$ and $Q_{\mathcal{V}} = [V|\bar{V}]$ such that the columns of U and V form orthonormal bases of \mathcal{U} and \mathcal{V} , respectively, to see

$$\delta(\mathcal{U}, \mathcal{V}) = \|(I - P_{\mathcal{V}})P_{\mathcal{U}}\| = \|\bar{V}\bar{V}^*UU^*\| = \|\bar{V}^*U\|. \quad (1.30)$$

The next lemma compiles some rudimentary properties.

Lemma 1.13. *For any subspaces $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^n$:*

- (i) $0 \leq \delta(\mathcal{U}, \mathcal{V}) \leq 1$
- (ii) $\mathcal{U} \cap \mathcal{V}^{\perp} \neq \{\mathbf{o}\} \Rightarrow \delta(\mathcal{U}, \mathcal{V}) = 1$
- (iii) $\dim \mathcal{U} > \dim \mathcal{V} \Rightarrow \delta(\mathcal{U}, \mathcal{V}) = 1$
- (iv) $\dim \mathcal{U} = \dim \mathcal{V} \Rightarrow \delta(\mathcal{U}, \mathcal{V}) = \delta(\mathcal{V}, \mathcal{U})$
- (v) $\max\{\delta(\mathcal{U}, \mathcal{V}), \delta(\mathcal{V}, \mathcal{U})\} = \|P_{\mathcal{U}} - P_{\mathcal{V}}\|$

Proof. $I - P_{\mathcal{V}}$ is an (orthogonal) projector (onto \mathcal{V}^{\perp}), thus (i) follows from (1.29) and (1.26). (ii) is a direct consequence of the way δ is defined. (iii) follows from (ii) if one exploits that \mathbb{R}^n is finite dimensional and therefore $\dim \mathcal{U} > \dim \mathcal{V}$ implies the existence of a nonzero $\mathbf{x} \in \mathcal{U} \cap \mathcal{V}^{\perp}$.

Now fix unitary matrices as in (1.30). Then

$$Q_{\mathcal{U}}^*(P_{\mathcal{U}} - P_{\mathcal{V}})Q_{\mathcal{V}} = \begin{bmatrix} U^* \\ \bar{U}^* \end{bmatrix} (UU^* - VV^*) [V|\bar{V}] = \begin{bmatrix} 0 & U^*\bar{V} \\ -\bar{U}^*V & 0 \end{bmatrix},$$

so together with (1.30) we have (v).

Finally, it remains to show (iv). With (1.25) and (1.30),

$$\begin{aligned}\delta(\mathcal{U}, \mathcal{V})^2 &= \|\bar{\mathbf{V}}^* \mathbf{U}\|^2 = \max_{\|\mathbf{x}\|=1} \mathbf{x}^* \mathbf{U}^* \bar{\mathbf{V}} \bar{\mathbf{V}}^* \mathbf{U} \mathbf{x} \\ &= \max_{\|\mathbf{x}\|=1} \mathbf{x}^* \mathbf{U}^* (\mathbf{I} - \mathbf{V} \mathbf{V}^*) \mathbf{U} \mathbf{x} \\ &= 1 - \min_{\|\mathbf{x}\|=1} \|\mathbf{V}^* \mathbf{U} \mathbf{x}\|^2,\end{aligned}$$

and analogously,

$$\delta(\mathcal{V}, \mathcal{U})^2 = \|\bar{\mathbf{U}}^* \mathbf{V}\|^2 = 1 - \min_{\|\mathbf{x}\|=1} \|\mathbf{U}^* \mathbf{V} \mathbf{x}\|^2.$$

Since $\dim \mathcal{U} = \dim \mathcal{V}$, the matrix $\mathbf{U}^* \mathbf{V}$ is square. Should it be singular, then the spaces $\mathcal{U} \cap \mathcal{V}^\perp$ and $\mathcal{U}^\perp \cap \mathcal{V}$ are both nontrivial, and (ii) gives $\delta(\mathcal{U}, \mathcal{V}) = \delta(\mathcal{V}, \mathcal{U}) = 1$. Otherwise, $\mathbf{U}^* \mathbf{V}$ is invertible and hence $\mathbf{V}^* \mathbf{U} = (\mathbf{U}^* \mathbf{V})^*$ is, too. Thus the above calculations give

$$\delta(\mathcal{U}, \mathcal{V})^2 = 1 - \|(\mathbf{V}^* \mathbf{U})^{-1}\|^{-2} = 1 - \|(\mathbf{U}^* \mathbf{V})^{-1}\|^{-2} = \delta(\mathcal{V}, \mathcal{U})^2,$$

completing the argument. \square

Angles

The concept of angles between general subspaces that we will use is based on the distances δ that were just introduced. However, the angle really should not depend on the order in which we write down the subspaces, so the definition will be symmetrical, in contrast to the δ -measure. After all we have $\angle(\mathbf{x}, \mathbf{y}) = \angle(\mathbf{y}, \mathbf{x})$ for vectors as well. Point (iii) of Lemma 1.13 shows that it is really only sensible to use δ -distances in a way that the maximization in (1.28) runs over the smaller of the spaces, and (iv) conveys that we can take either one if they have equal dimension.

Definition 1.14. For any two nontrivial subspaces $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^n$, define the (*acute angle*) between them to be the number $\angle(\mathcal{U}, \mathcal{V}) \in [0, \pi/2]$ with

$$\sin \angle(\mathcal{U}, \mathcal{V}) = \min \left\{ \delta(\mathcal{U}, \mathcal{V}), \delta(\mathcal{V}, \mathcal{U}) \right\}.$$

\diamond

Next we establish that our definition of angles is indeed consistent with the intuitively clear notions of angle between vectors and between vectors and subspaces as they were introduced at the beginning of this section; in particular we retain the geometrical connections from (1.21) and (1.22) between angles and orthogonal decompositions from Figure 1.2.

Lemma 1.15. *Let \mathcal{S} be a nontrivial subspace of \mathbb{R}^n . Then each $\mathbf{x} \in \mathbb{R}^n$ with $\|\mathbf{x}\| = 1$ has a unique orthogonal decomposition*

$$\mathbf{x} = \mathbf{s} \cos \alpha + \mathbf{r} \sin \alpha, \quad \alpha = \angle(\mathbf{x}, \mathcal{S}),$$

with $\mathbf{s} \in \mathcal{S}, \mathbf{r} \in \mathcal{S}^\perp$ and $\|\mathbf{s}\| = \|\mathbf{r}\| = 1$.

Proof. There is really not much to do except to take the orthogonal decomposition as defined by the projector $\mathbf{P}_{\mathcal{S}}$, note that $\sin \alpha = \delta(\text{span}\{\mathbf{x}\}, \mathcal{S}) = \|(I - \mathbf{P}_{\mathcal{S}})\mathbf{x}\|$ and leave the rest to Pythagoras. \square

The following minimax-characterization builds on the well-understood angles between vectors in Euclidean space from (1.20) and avoids the explicit use of norms.

Lemma 1.16. *For any two subspaces $\mathcal{U}, \mathcal{V} \subseteq \mathbb{R}^n$ with $0 < \dim \mathcal{U} \leq \dim \mathcal{V}$,*

$$\angle(\mathcal{U}, \mathcal{V}) = \max_{\mathbf{u} \in \mathcal{U}} \min_{\mathbf{v} \in \mathcal{V}} \angle(\mathbf{u}, \mathbf{v}).$$

Proof. In this situation we have $\sin \angle(\mathcal{U}, \mathcal{V}) = \delta(\mathcal{U}, \mathcal{V})$. As sine is strictly monotonically increasing on $[0, \pi/2]$, the claim is just (1.28) if one notes that for \mathbf{x} of unit norm, $\|(I - \mathbf{P}_{\mathcal{V}})\mathbf{x}\| = \min_{\mathbf{v} \in \mathcal{V}} \sin \angle(\mathbf{x}, \mathbf{v})$. \square

This characterization makes the following result immediate. We just write it down here because it is needed for Chapter 2.

Corollary 1.17. *Let \mathcal{U}, \mathcal{V} be nontrivial subspaces of \mathbb{R}^n with $\mathcal{U} \subseteq \mathcal{V}$. Then for any nontrivial \mathcal{Z}*

$$\begin{cases} \angle(\mathcal{U}, \mathcal{Z}) \leq \angle(\mathcal{V}, \mathcal{Z}), & \text{if } \dim \mathcal{V} \leq \dim \mathcal{Z}, \\ \angle(\mathcal{Z}, \mathcal{V}) \leq \angle(\mathcal{Z}, \mathcal{U}), & \text{if } \dim \mathcal{Z} \leq \dim \mathcal{U}. \end{cases}$$

Finally, we will show that general subspace angles do still enjoy some notion of transitivity. This fact will also be put to frequent use in Chapters 2 and 3.

Lemma 1.18. *Let $\mathcal{U}, \mathcal{V}, \mathcal{W}$ be nontrivial subspaces of \mathbb{R}^n with*

$$\min\{\dim \mathcal{U}, \dim \mathcal{W}\} \leq \dim \mathcal{V} \leq \max\{\dim \mathcal{U}, \dim \mathcal{W}\}.$$

Then

$$\sin \angle(\mathcal{U}, \mathcal{W}) \leq \sin \angle(\mathcal{U}, \mathcal{V}) + \sin \angle(\mathcal{V}, \mathcal{W}).$$

Proof. The symmetric definition of angles allows without loss of generality to restrict ourselves to the case $\dim \mathcal{U} \leq \dim \mathcal{V} \leq \dim \mathcal{W}$. Then the claim is equivalent to

$$\delta(\mathcal{U}, \mathcal{W}) \leq \delta(\mathcal{U}, \mathcal{V}) + \delta(\mathcal{V}, \mathcal{W}).$$

First assume $\dim \mathcal{V} = \dim \mathcal{W}$. Use that operator norms are matrix norms, (1.26) and Lemma 1.13 to see

$$\begin{aligned} \delta(\mathcal{U}, \mathcal{W}) &= \|(I - P_{\mathcal{W}})P_{\mathcal{U}}\| \\ &\leq \|(I - P_{\mathcal{V}})P_{\mathcal{U}}\| + \|(P_{\mathcal{V}} - P_{\mathcal{W}})P_{\mathcal{U}}\| \\ &\leq \|(I - P_{\mathcal{V}})P_{\mathcal{U}}\| + \|P_{\mathcal{V}} - P_{\mathcal{W}}\| \\ &= \delta(\mathcal{U}, \mathcal{V}) + \delta(\mathcal{V}, \mathcal{W}). \end{aligned} \quad (\star)$$

For the case $\dim \mathcal{V} < \dim \mathcal{W}$, we can find a subspace $\mathcal{S} \subseteq (\mathcal{V}^{\perp} \cap \mathcal{W})$ with $\dim \mathcal{S} = \dim \mathcal{W} - \dim \mathcal{V}$. Then $\tilde{\mathcal{V}} := \mathcal{V} + \mathcal{S}$ has $\dim \tilde{\mathcal{V}} = \dim \mathcal{W}$, so (\star) is applicable and yields

$$\delta(\mathcal{U}, \mathcal{W}) \leq \delta(\mathcal{U}, \tilde{\mathcal{V}}) + \delta(\tilde{\mathcal{V}}, \mathcal{W}).$$

It is not hard to see that our choice for $\tilde{\mathcal{V}}$ implies $\delta(\tilde{\mathcal{V}}, \mathcal{W}) = \delta(\mathcal{V}, \mathcal{W})$. Furthermore, we have $\mathcal{V} \subseteq \tilde{\mathcal{V}}$ and $\sin \angle(\mathcal{U}, \tilde{\mathcal{V}}) = \delta(\mathcal{U}, \tilde{\mathcal{V}})$, so Lemma 1.17 gives $\delta(\mathcal{U}, \tilde{\mathcal{V}}) \leq \delta(\mathcal{U}, \mathcal{V})$. \square

At this point, we have all the tools concerning angles assembled that we will need on the following pages. However, before we leave this section, we want at least to remark on one prominent concept that was left out so far.

Remark 1.19 (Principal Angles). For general subspaces \mathcal{U}, \mathcal{V} there is actually a whole collection

$$\phi_1 \leq \dots \leq \phi_k, \quad k = \min \{ \dim \mathcal{U}, \dim \mathcal{V} \},$$

of angles between them, called the *canonical* or *principal* angles. These are usually defined in an algorithmic fashion, by taking the minimal angle $\phi_1 := \theta_m$ from (1.23), factoring out the directions (\mathbf{u} and \mathbf{v}) for which it was attained, and continuing iteratively until the smaller space is reduced to the nullspace. Geometrically, the canonical angles do in a way define a minimal sequence of rotations around suitable axes to bring the smaller space into the larger one. Mathematically, the cosines of the canonical angles are the singular values of $\mathbf{V}^* \mathbf{U}$ (see [34, p. 603f]) and their sines are among the singular values of $P_{\mathcal{U}} - P_{\mathcal{V}}$, (cf. (v) in Lemma 1.13). Our definition gives nothing else than the *largest* canonical angle, that is, $\angle(\mathcal{U}, \mathcal{V}) = \phi_k$. \diamond

1.4 Theory of the Symmetric Eigenproblem (Greatest Hits)

This section will take us on a trip to visit a selection of topics from the theory of the symmetric eigenproblem.

We only cover results that are in some way used at a later point; this includes that we need to reference them directly for proving something, but also

the introduction of crucial techniques or just valuable insights. Everything that is presented is done so in detail, including complete proofs.

Experts in the field will probably not find anything new on the following pages. The contents can be found in most standard textbooks. Because of this we have opted not to give a detailed reference to the literature for each lemma or theorem. Except for small modifications here and there our exposition follows the lead of [56], with some additions from [10, 69] mixed in.

1.4.1 Rayleigh's Quotient, Residuals and Gaps

Any matrix \mathbf{A} ($\in \mathbb{R}^{n \times n}$, symmetric) defines a bilinear form $\mathbf{x} \mapsto \mathbf{x}^* \mathbf{A} \mathbf{x}$. By factoring out the vector's norm one obtains a map that is of the utmost importance for the theory of symmetric matrices and their eigenvalues, namely

$$\rho_{\mathbf{A}}(\mathbf{x}) := \frac{\mathbf{x}^* \mathbf{A} \mathbf{x}}{\mathbf{x}^* \mathbf{x}}, \quad \mathbf{x} \neq \mathbf{o}, \quad (1.31)$$

called the *Rayleigh Quotient* of \mathbf{A} applied to \mathbf{x} . If the matrix can be inferred from the context we just write $\rho(\mathbf{x})$. The undefinedness at zero is unlucky as it becomes cumbersome to heed $\mathbf{x} \neq \mathbf{o}$ at each occasion. In the sake of brevity let us thus stipulate that whenever we use Rayleigh's Quotient, the involved vectors are implicitly assumed to be nonzero.

From its design, $\rho_{\mathbf{A}}(\mathbf{x})$ is independent of the norm of \mathbf{x} , that is, it is invariant under scaling, $\rho_{\mathbf{A}}(\mathbf{x}) = \rho_{\mathbf{A}}(\alpha \mathbf{x})$ for all $\alpha \in \mathbb{R}^{\neq 0}$. Despite this feature being admittedly obvious, it is worth mentioning, as it allows to restrict any analysis of the Rayleigh Quotient to the unit sphere.

For a deeper understanding of $\rho_{\mathbf{A}}$ we have to put \mathbf{A} 's eigendecomposition into play. Let $\lambda_i = \lambda_i[\mathbf{A}]$, $\mathbf{q}_i = \mathbf{q}_i[\mathbf{A}]$, $i = 1, \dots, n$, so that we can expand \mathbf{x} in the basis of \mathbf{A} 's eigenvectors as

$$\mathbf{x} = \sum_{i=1}^n \xi_i \mathbf{q}_i \quad \text{with} \quad \|\mathbf{x}\|^2 = \sum_{i=1}^n \xi_i^2.$$

This gives

$$\rho_{\mathbf{A}}(\mathbf{x}) = \sum_{i=1}^n \lambda_i \xi_i^2 = \sum_{i=1}^n \lambda_i (\mathbf{q}_i^* \mathbf{x})^2, \quad (1.32)$$

revealing the following crucial property:

$\rho_{\mathbf{A}}(\mathbf{x})$ is a weighted average of the eigenvalues of \mathbf{A} , where the weight of λ is just the square of the weight of the corresponding unit eigenvector in the representation of \mathbf{x} .

One can also use the term *convex combination* instead of weighted average, this makes it even more clear that $\rho_{\mathbf{A}}(\mathbf{x})$ ranges over the convex hull of \mathbf{A} 's eigenvalues, i.e.,

$$\lambda_1[\mathbf{A}] \leq \rho_{\mathbf{A}}(\mathbf{x}) \leq \lambda_n[\mathbf{A}] \quad \text{and} \quad \max_{\mathbf{x} \neq \mathbf{0}} |\rho_{\mathbf{A}}(\mathbf{x})| = \|\mathbf{A}\|. \quad (1.33)$$

Still just building on (1.32), we can improve on this result for arbitrary invariant subspaces. If $\mathbf{x} \neq \mathbf{0}$ is known to lie within $\mathcal{Q}_I[\mathbf{A}]$, then

$$\lambda_{\min I}[\mathbf{A}] \leq \rho_{\mathbf{A}}(\mathbf{x}) \leq \lambda_{\max I}[\mathbf{A}]. \quad (1.34)$$

A nice application is to consider the index sets $\{j, \dots, n\}$ and $\{1, \dots, j\}$ with their associated invariant subspaces. They have, respectively, only one index j and one eigenvector $\mathbf{q}_j[\mathbf{A}]$ in common. Thus a straightforward argument based on (1.34) gives the characterization

$$\max \left\{ \rho_{\mathbf{A}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{Q}_{1:j}[\mathbf{A}] \right\} = \lambda_j[\mathbf{A}] = \min \left\{ \rho_{\mathbf{A}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{Q}_{j:n}[\mathbf{A}] \right\}. \quad (1.35)$$

Experienced readers will recognize this as an instance of the Courant-Fischer Theorem which we will present on its own at a later point.

Clearly, if $\mathbf{x} = \mathbf{q}_i$ happens to be an eigenvector of \mathbf{A} , then $\rho(\mathbf{x}) = \lambda_i$. But note that the converse is not necessarily true, that is, $\rho(\mathbf{x}) = \lambda_i$ does not imply that \mathbf{x} is an eigenvector. Just consider for example a matrix with eigenvalues that satisfy $\lambda_i = \frac{1}{2}(\lambda_{i-1} + \lambda_{i+1})$ and then take $\mathbf{x} = \mathbf{q}_{i-1} + \mathbf{q}_{i+1}$.

However, on a related note, it is not hard to show that there is a one-to-one correspondence between eigenpairs of \mathbf{A} and pairs of extremal values and stationary points of the map $\rho_{\mathbf{A}}$ constrained to the unit sphere. Using Lagrange multipliers, we know that those are just the pairs (λ, \mathbf{x}) where the derivative of

$$L(\mathbf{x}) = \mathbf{x}^* \mathbf{A} \mathbf{x} - \lambda(\mathbf{x}^* \mathbf{x} - 1)$$

vanishes, and this gives

$$L'(\mathbf{x}) = 2\mathbf{x}^* \mathbf{A} - 2\lambda \mathbf{x}^* = \mathbf{0} \quad \Leftrightarrow \quad \mathbf{A} \mathbf{x} = \lambda \mathbf{x}.$$

An algorithm that computes eigenpairs has to employ some means to evaluate the quality of an approximation. A natural way to do so is by monitoring an eigenpair's residual. The following result states that the residual norm is an upper bound for—and can therefore never be smaller than—the absolute accuracy in the eigenvalue approximation.

Theorem 1.20

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric. Then for every nonzero vector $\mathbf{x} \in \mathbb{R}^n$ and every $\mu \in \mathbb{R}$ there is an eigenvalue λ of \mathbf{A} such that

$$|\lambda - \mu| \leq \|\mathbf{A} \mathbf{x} - \mathbf{x} \mu\| / \|\mathbf{x}\|.$$

Proof. There are different approaches, we chose one that exhibits the Rayleigh Quotient:

$$\|Ax - x\mu\|^2 / \|x\|^2 = \rho_{(A-\mu)^2}(x) \stackrel{(1.33)}{\geq} \lambda_1[(A-\mu)^2] = \min_i (\lambda_i[A] - \mu)^2.$$

□

A special residual is the one that comes with the Rayleigh Quotient,

$$r_A(x) := Ax - \rho_A(x)x. \quad (1.36)$$

Its characteristic feature is being orthogonal to x , as $x^*r_A(x) = x^*Ax - \|x\|^2\rho_A(x) = 0$. Hence $Ax = r_A(x) + \rho_A(x)x$ is the orthogonal decomposition of the map of x under A , with respect to $\text{span}\{x\}$, see Figure 1.3. This realization has several consequences. First of all it shows that the Rayleigh Quotient-residual is the best residual one can get with any eigenvalue for a given vector x ,

$$\|Ax - x\rho_A(x)\| \leq \|Ax - x\mu\| \quad \text{for all } x \neq 0, \mu \in \mathbb{R}. \quad (1.37)$$

Furthermore, elementary trigonometry (or Lemma 1.15) gives

$$\|Ax\| \cos\angle(x, Ax) = \rho_A(x)\|x\|, \quad \|Ax\| \sin\angle(x, Ax) = \|r_A(x)\|, \quad (1.38)$$

linking $r_A(x)$ to the rotation effected by A on x .

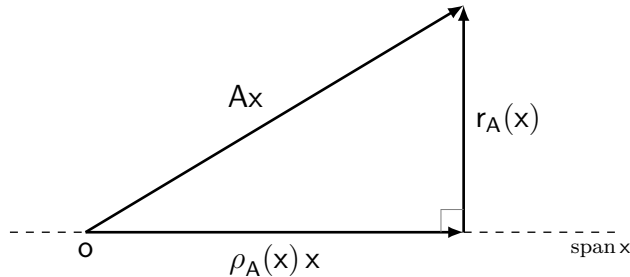


Figure 1.3: The Rayleigh Quotient, its associated residual and their role in the orthogonal decomposition of the mapped vector.

Now we come up to two highlights. Theorem 1.20 told us that small residual norms imply accuracy in the eigenvalue. The *Gap Theorems* we will present now do essentially the same for eigenvectors, as they connect the residual norm with how close the vector is to a true eigenvector or an invariant subspace, expressed similar to (1.38) by considering angles.

It turns out that the quality of information about vectors that can be inferred from the residual hinges on gaps in the spectrum. For stating the following results,

the gaps we defined in §1.1 are unfortunately not sufficient. Instead we need to define for any index set I the rather exotic

$$\text{gap}_{\mathbf{A}}(I; \mu) := \min \left\{ |\lambda_j - \mu| : j \notin I \right\}, \quad (1.39)$$

which is the distance of μ to any eigenvalue of \mathbf{A} whose index is *not* in I . The connection to the traditional gap-measures from (1.3) and (1.5) is as follows:

$$i \in I \text{ and } |\mu - \lambda_i| \leq \frac{1}{2} \text{gap}_{\mathbf{A}}(\lambda_i) \implies \frac{1}{2} \text{gap}_{\mathbf{A}}(I) \leq \text{gap}_{\mathbf{A}}(I; \mu). \quad (1.40)$$

The condition states that μ is closer to an eigenvalue λ_i with index $i \in I$ than to any other eigenvalue of \mathbf{A} ; this is usually the case in situations where we want to invoke one of the Gap Theorems.

The following result about subspaces is usually not found in introductory textbooks; it can be generalized to angles between subspaces of equal dimensions [56, Thm. 11.7.2] or even arbitrary ones [9]. Note the lack of requirements except for a nonzero gap, which will only not be fulfilled if the eigenvalue approximation μ is really bad and happens to coincide with an exact eigenvalue outside the set we are interested in.

Theorem 1.21 (Gap Theorem for an invariant subspace)

For every symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$, unit vector \mathbf{x} , scalar μ and index set I ,

$$\sin \angle(\mathcal{Q}_I[\mathbf{A}], \mathbf{x}) \leq \frac{\|\mathbf{A}\mathbf{x} - \mathbf{x}\mu\|}{\text{gap}_{\mathbf{A}}(I; \mu)},$$

provided $\text{gap}_{\mathbf{A}}(I; \mu) \neq 0$.

Proof. Decompose \mathbf{x} along $\mathcal{Q}_I[\mathbf{A}]$ as

$$\mathbf{x} = \sum_{i \in I} \xi_i \mathbf{q}_i[\mathbf{A}] + \sum_{j \notin I} \omega_j \mathbf{q}_j[\mathbf{A}]$$

to see

$$\begin{aligned} \|\mathbf{A}\mathbf{x} - \mathbf{x}\mu\|^2 &= \sum_{i \in I} \xi_i^2 (\lambda_i[\mathbf{A}] - \mu)^2 + \sum_{j \notin I} \omega_j^2 (\lambda_j[\mathbf{A}] - \mu)^2 \\ &\geq \text{gap}_{\mathbf{A}}(I; \mu)^2 \sum_{j \notin I} \omega_j^2 \\ &= \text{gap}_{\mathbf{A}}(I; \mu)^2 \sin^2 \angle(\mathcal{Q}_I[\mathbf{A}], \mathbf{x}), \end{aligned}$$

where we needed Lemma 1.15 to make the last step. \square

Now we present the Gap Theorem that is probably better known. Additionally to the same upper bound on the sine as in Theorem 1.21, the use of Rayleigh Quotients makes it possible to give both a lower bound on the sine, as well as a tight upper bound on the distance of $\rho_{\mathbf{A}}(\mathbf{x})$ to an exact eigenvalue of \mathbf{A} that is proportional to the *square* of the residual norm and generally superior to the one from Theorem 1.20. In fact, this latter bound is the driving force behind Rayleigh Quotient Iteration (RQI, [56]) and its cubic convergence. However, in contrast to Theorem 1.21, the new bounds *cannot* be generalized for invariant subspaces with dimension exceeding one, cf. Lemma 3.5.

The proof is rather long and quite subtle, but highly instructive. Therefore we decided to present it in full length; except for minor adjustments it is taken from [56].

Theorem 1.22 (Gap Theorem with Rayleigh's Quotient)

For symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$ and unit vector \mathbf{x} with $\theta = \rho_{\mathbf{A}}(\mathbf{x})$, let $\lambda = \lambda_i[\mathbf{A}]$ be an eigenvalue of \mathbf{A} such that no other eigenvalue lies between (or equals) λ and θ , and $\mathbf{q} = \mathbf{q}_i[\mathbf{A}]$ the corresponding normalized eigenvector. With

$$\mathbf{r} := (\mathbf{A} - \theta)\mathbf{x}, \quad \text{gap} := \text{gap}_{\mathbf{A}}(\{i\}; \theta), \quad \text{spdiam} := \text{spdiam}[\mathbf{A}],$$

we will have $\text{gap} > 0$ and

$$\frac{\|\mathbf{r}\|}{\text{spdiam}} \leq \sin \angle(\mathbf{x}, \mathbf{q}) \leq \frac{\|\mathbf{r}\|}{\text{gap}} \quad \text{and} \quad |\theta - \lambda| \leq \frac{\|\mathbf{r}\|^2}{\text{gap}}.$$

Proof. Let $\angle(\mathbf{x}, \mathbf{q}) =: \psi$. The upper bound for $\sin \psi$ is just a special case for Theorem 1.21 (with $I = \{i\}, \mu = \theta$). We can without loss of generality assume $\mathbf{x} \notin \text{span}\{\mathbf{q}\}$ as there is nothing to prove otherwise. Then there is a unique unit vector \mathbf{w} orthogonal to \mathbf{q} such that we have the decomposition $\mathbf{x} = \mathbf{q} \cos \psi + \mathbf{w} \sin \psi$. This gives

$$\mathbf{r} = (\mathbf{A} - \theta)\mathbf{x} = \mathbf{q}(\lambda - \theta) \cos \psi + (\mathbf{A} - \theta)\mathbf{w} \sin \psi.$$

Because \mathbf{w} is orthogonal to \mathbf{q} , so is $(\mathbf{A} - \theta)\mathbf{w}$. Hence, using Pythagoras, we obtain

$$\|\mathbf{r}\|^2 = (\lambda - \theta)^2 \cos^2 \psi + \|(\mathbf{A} - \theta)\mathbf{w}\|^2 \sin^2 \psi. \quad (\text{A})$$

On the other hand, we know that $\mathbf{r} = \mathbf{r}_{\mathbf{A}}(\mathbf{x})$, so $\mathbf{x} \perp \mathbf{r}$ and therefore

$$0 = \mathbf{x}^* \mathbf{r} = (\lambda - \theta) \cos^2 \psi + \mathbf{w}^* (\mathbf{A} - \theta) \mathbf{w} \sin^2 \psi. \quad (\text{B})$$

Solve (B) for $(\lambda - \theta) \cos^2 \psi$ and combine with (A) to see

$$\begin{aligned}
\|r\|^2 &= \|(A - \theta)w\|^2 \sin^2 \psi - (\lambda - \theta)w^*(A - \theta)w \sin^2 \psi \\
&= (w^*(A - \theta)(A - \theta)w - w^*(\lambda - \theta)(A - \theta)w) \sin^2 \psi \\
&= w^*((A - \theta) - (\lambda - \theta))(A - \theta)w \sin^2 \psi \\
&= w^*(A - \lambda)(A - \theta)w \sin^2 \psi.
\end{aligned} \tag{C}$$

The Cauchy-Schwartz inequality now gives

$$w^*(A - \lambda)(A - \theta)w \leq \|(A - \lambda)w\| \|(A - \theta)w\| \leq \text{spdiam}^2,$$

which provides us with the desired lower bound on $\sin \psi$.

We still have to come up with the upper bound on $|\theta - \lambda|$. Write $w = \sum_{j \neq i} \omega_j \mathbf{q}_j[A]$ for

$$w^*(A - \lambda)(A - \theta)w = \sum_{j \neq i} \omega_j^2 (\lambda_j - \lambda)(\lambda_j - \theta) \stackrel{(C)}{\geq} 0.$$

At this point we deploy the as yet unused condition that no eigenvalue of A comes between λ and θ to conclude that for all $j \neq i$, the scalars $(\lambda_j - \lambda)$ and $(\lambda_j - \theta)$ have identical signs. In other words, the matrix $(A - \lambda)(A - \theta)$ is positive definite on $\text{span}\{\mathbf{q}\}^\perp$. Hence,

$$\begin{aligned}
w^*(A - \lambda)(A - \theta)w &= \sum_{j \neq i} \omega_j^2 \cdot |\lambda_j - \lambda| \cdot |\lambda_j - \theta| \\
&\geq \text{gap} \left| \sum_{j \neq i} \omega_j^2 (\lambda_j - \lambda) \right| \\
&= \text{gap} |w^*(A - \lambda)w|.
\end{aligned}$$

Thus, looking back at (C), we realize that the proof is complete as soon as we can establish $|w^*(A - \lambda)w| \sin^2 \psi \geq |\theta - \lambda|$. Indeed, we can even show equality here: Plug $\cos^2 \psi = 1 - \sin^2 \psi$ into (B) and rearrange for

$$(\theta - \lambda) = (w^*(A - \theta)w + (\theta - \lambda)) \sin^2 \psi = w^*(A - \lambda)w \sin^2 \psi.$$

□

1.4.2 The Classics

In the following we present a series of timeless results from the theory of the symmetric eigenproblem, all of which are going to be needed at some later point. Concretely these are the Cauchy Interlace Theorem, the Courant-Fischer Theorem giving minmax- and maxmin-characterizations, Sylvester's Law of Inertia and Weyl's Theorem.

Interestingly, all four build directly on the following fact from basic linear algebra:

In a finite dimensional space, if the dimensions of two subspaces sum to more than the whole, then they must have a nontrivial intersection.

The Cauchy Interlace Theorem is usually known in the form that deleting from a symmetric matrix for one index the corresponding row and column gives a symmetric matrix whose eigenvalues lie between (*interlace*) the original one's. An alternative (but equivalent) formulation links the eigenvalues to those of a principal submatrix. See also Figure 1.4 for a comparison of these viewpoints.

The eigenvalues of a principal submatrix are also called *Ritz Values*, although they are in fact only a special kind of those, namely with respect to columns of the identity.

Theorem 1.23 (Cauchy Interlace Theorem)

For every symmetric A with principal submatrix A_I , and every k with $1 \leq k \leq |I|$,

$$\lambda_k[A] \leq \lambda_k[A_I] \quad \text{and} \quad \lambda_{-k}[A_I] \leq \lambda_{-k}[A].$$

Proof. We start with simplifying the setting a bit, by noting that, as any symmetric permutation $A \mapsto P^*AP$ has no effect on the eigenvalues, we can just as well limit ourselves to the case $I = \{1, \dots, m\}$, where A can be partitioned

$$A = \begin{bmatrix} H & B \\ B^* & X \end{bmatrix}, \quad H := A_I. \quad (\star)$$

For any given k we then have

$$\dim \mathcal{Q}_{k:n}[A] + \dim \mathcal{Q}_{1:k}[H] = n + 1 > n.$$

If we append zeros to the eigenvectors of H to get them to dimension n , the previous dimensional argument remains valid for the thusly prolonged invariant subspaces of H . Hence we can find a nonzero vector $x \in \mathcal{Q}_{1:k}[H]$ such that

$$y = \begin{bmatrix} x \\ \mathbf{o} \end{bmatrix} \in \mathcal{Q}_{k:n}[A]. \quad (\star\star)$$

This gives

$$\begin{aligned} \lambda_k[A] &= \min \left\{ \rho_A(z) \mid z \in \mathcal{Q}_{k:n}[A] \right\} && \text{by (1.35)} \\ &\leq \rho_A(y) && \text{since } y \in \mathcal{Q}_{k:n}[A], \\ &= \rho_H(x) && \text{by } (\star) \text{ and } (\star\star), \\ &\leq \max \left\{ \rho_H(z) \mid z \in \mathcal{Q}_{1:k}[H] \right\} && \text{since } x \in \mathcal{Q}_{1:k}[H], \\ &= \lambda_k[H], \end{aligned}$$

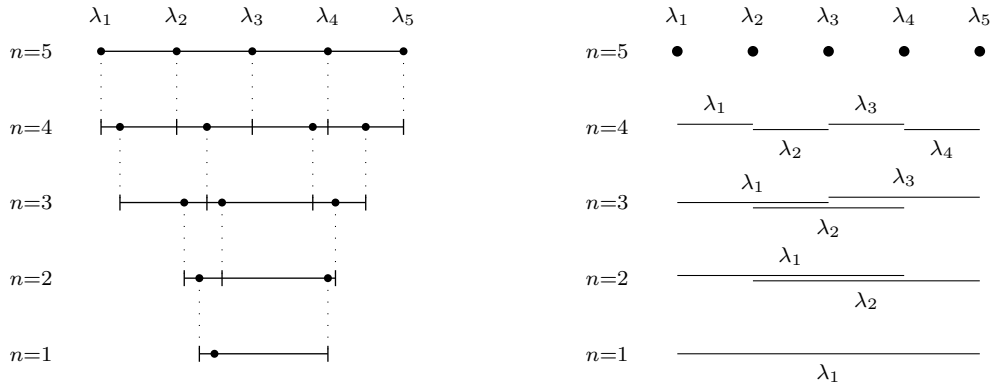


Figure 1.4: The Cauchy Interlace Theorem. *On the left:* Sample eigenvalue distribution for sequence of submatrices obtained by successively deleting one row and column; dotted lines indicate the bounds as given by the next larger matrix. *On the right:* The lines indicate ranges where eigenvalues of submatrices with the respective dimensions can lie, deduced directly from the original $n = 5$ matrix.

proving the first inequality. The second inequality is just the first one written for $-\mathbf{A}$. □

The Courant-Fischer Theorem provides the well-known minmax-characterization of eigenvalues and is a generalization of (1.35). The following formulation [56, Thm. 10.2.1] differentiates between *subspaces* and *constraint spaces*.

Theorem 1.24 (Courant-Fischer “Minimax” Theorem)

Let \mathcal{S} and \mathcal{C} stand for subspaces of \mathbb{R}^n with dimensions attached as superscripts. For $j = 1, \dots, n$, the j 'th eigenvalue of symmetric $\mathbf{A} \in \mathbb{R}^{n \times n}$ is characterized by

$$\min_{\mathcal{S}^j} \max_{\mathbf{x} \in \mathcal{S}^j} \rho_{\mathbf{A}}(\mathbf{x}) = \lambda_j[\mathbf{A}] = \max_{\mathcal{C}^{j-1}} \min_{\mathbf{x} \perp \mathcal{C}^{j-1}} \rho_{\mathbf{A}}(\mathbf{x}),$$

or equivalently stated for the j 'th largest eigenvalue of \mathbf{A}

$$\max_{\mathcal{S}^j} \min_{\mathbf{x} \in \mathcal{S}^j} \rho_{\mathbf{A}}(\mathbf{x}) = \lambda_{-j}[\mathbf{A}] = \min_{\mathcal{C}^{j-1}} \max_{\mathbf{x} \perp \mathcal{C}^{j-1}} \rho_{\mathbf{A}}(\mathbf{x}).$$

Proof. To see that both formulas are equivalent, replace j by $n - j + 1$ in the first and note that

$$\min_{\mathcal{S}^{n-j+1}} \max_{\mathbf{x} \in \mathcal{S}^{n-j+1}} f(\mathbf{x}) = \min_{\mathcal{C}^{j-1}} \max_{\mathbf{x} \perp \mathcal{C}^{j-1}} f(\mathbf{x})$$

for any f .

Because $\dim \mathcal{S}^j + \dim (\mathcal{C}^{j-1})^\perp = j + (n - j + 1) > n$ we can find a nonzero vector $\mathbf{w} \in \mathcal{S}^j \cap (\mathcal{C}^{j-1})^\perp$. Then we surely have

$$\min_{\mathbf{x} \perp \mathcal{C}^{j-1}} \rho_{\mathbf{A}}(\mathbf{x}) \leq \rho_{\mathbf{A}}(\mathbf{w}) \leq \max_{\mathbf{x} \in \mathcal{S}^j} \rho_{\mathbf{A}}(\mathbf{x}).$$

As this holds for every possible configuration of \mathcal{S}^j and \mathcal{C}^{j-1} , we can conclude

$$\max_{\mathcal{C}^{j-1}} \min_{\mathbf{x} \perp \mathcal{C}^{j-1}} \rho_{\mathbf{A}}(\mathbf{x}) \leq \min_{\mathcal{S}^j} \max_{\mathbf{x} \in \mathcal{S}^j} \rho_{\mathbf{A}}(\mathbf{x}).$$

Now consider the choices $\mathcal{S}^j = \mathcal{Q}_{1:j}[\mathbf{A}]$ and $(\mathcal{C}^{j-1})^\perp = \mathcal{Q}_{j:n}[\mathbf{A}]$. Use (1.35) to see

$$\begin{aligned} \lambda_j[\mathbf{A}] &= \min \{ \rho_{\mathbf{A}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{Q}_{j:n}[\mathbf{A}] \} \\ &\leq \max_{\mathcal{C}^{j-1}} \min_{\mathbf{x} \perp \mathcal{C}^{j-1}} \rho_{\mathbf{A}}(\mathbf{x}) \\ &\leq \min_{\mathcal{S}^j} \max_{\mathbf{x} \in \mathcal{S}^j} \rho_{\mathbf{A}}(\mathbf{x}) \\ &\leq \max \{ \rho_{\mathbf{A}}(\mathbf{x}) \mid \mathbf{x} \in \mathcal{Q}_{1:j}[\mathbf{A}] \} = \lambda_j[\mathbf{A}]. \end{aligned}$$

Thus all inequalities must in fact be equalities and the first characterization is proven. \square

Similarity transformations $\mathbf{A} \mapsto \mathbf{X}^{-1}\mathbf{A}\mathbf{X}$ leave eigenvalues unchanged. For the symmetric eigenproblem, *congruence transformations* $\mathbf{A} \mapsto \mathbf{X}^*\mathbf{A}\mathbf{X}$ with nonsingular \mathbf{X} are far more relevant in practice. Indeed, starting with Chapter 2 everything will quite literally revolve around them.

Congruent matrices need not have the same eigenvalues, but Sylvester's Law establishes that at least the *signs* of eigenvalues are identical. This is equivalent to saying that they have the same *inertia*, which is defined as triplet containing the numbers of negative, zero, and positive eigenvalues of the matrix. Methods that compute eigenvalues using bisection rely on Sylvester's Law implicitly.

The proof combines parts from [10, Theorem 5.3] and [55, p. 586].

Theorem 1.25 (Sylvester's Inertia Theorem)

Symmetric matrices are congruent if, and only if, they have the same inertia.

Proof. (\Leftarrow) Every symmetric matrix is congruent to a diagonal matrix with entries from $\{-1, 0, +1\}$, the signs of the eigenvalues. By reordering the eigenvalues conformably we see that if two symmetric matrices have the same inertia, they are in fact congruent to the same diagonal matrix. Then just note that congruence is an equivalence relation and hence transitive.

(\Rightarrow) Let $M = X^*AX \in \mathbb{R}^{n \times n}$ for nonsingular X and symmetric A, M . Choose a sign $s \in \{-1, 0, +1\}$ and let \mathcal{S}_A and \mathcal{S}_M be the invariant subspaces spanned by the eigenvectors belonging to the eigenvalues of A and M with sign s , respectively. Recall (1.34) to see that ρ_A and ρ_M both have constant sign s while varying over these spaces.

We will give a proof by contradiction. Assume without loss of generality $\dim \mathcal{S}_A > \dim \mathcal{S}_M$, otherwise swap A and M . Because X is nonsingular,

$$\dim \mathcal{S}_A + \dim(X\mathcal{S}_M^\perp) = \dim \mathcal{S}_A + n - \dim \mathcal{S}_M > n.$$

Therefore we can find nonzero vectors u, v with $u = Xv$,

$$u \in \mathcal{S}_A \Rightarrow \text{sign } \rho_A(u) = s \quad \text{and} \quad v \in \mathcal{S}_M^\perp \Rightarrow \text{sign } \rho_M(v) \neq s.$$

This is a contradiction to $\rho_A(u) = \rho_A(Xv) = \rho_M(v)$. \square

Last but surely not least, Weyl's Theorem provides a bound that links eigenvalues of matrices to those of their sum. It is extremely valuable for analyzing additive perturbations, but its principal setup is far more general. In fact, some textbooks, e.g., [10, Thm. 5.1], give only the error analysis viewpoint, which we will state as Corollary 1.28 below. The formulation here is taken from [56, Thm. 10.3.1].

Theorem 1.26 (Weyl's Theorem)

Let $W = U + V \in \mathbb{R}^{n \times n}$ and $i, j \in \mathbb{N}$ with $1 \leq i + j - 1 \leq n$. Then

$$\lambda_i[U] + \lambda_j[V] \leq \lambda_{i+j-1}[W] \quad \text{and} \quad \lambda_{-(i+j-1)}[W] \leq \lambda_{-i}[U] + \lambda_{-j}[V].$$

Proof. Using the dimension rule it is straightforward to show that

$$\dim(\mathcal{S}_1 \cap \mathcal{S}_2 \cap \mathcal{S}_3) \geq \dim \mathcal{S}_1 + \dim \mathcal{S}_2 + \dim \mathcal{S}_3 - 2n$$

holds for any three subspaces $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ of an n -dimensional vector space. In the situation at hand we have

$$\begin{aligned} \dim \mathcal{Q}_{i:n}[U] + \dim \mathcal{Q}_{j:n}[V] + \dim \mathcal{Q}_{1:(i+j-1)}[W] \\ = (n - i + 1) + (n - j + 1) + (i + j - 1) = 2n + 1, \end{aligned}$$

hence there is a nonzero vector z in the intersection of these three spaces. Thus,

$$\begin{aligned} \lambda_i[U] + \lambda_j[V] &= \min \{ \rho_U(x) \mid x \in \mathcal{Q}_{i:n}[U] \} + \min \{ \rho_V(x) \mid x \in \mathcal{Q}_{j:n}[V] \} \\ &\leq \rho_U(z) + \rho_V(z) \\ &= \rho_W(z) \\ &\leq \max \{ \rho_W(x) \mid x \in \mathcal{Q}_{1:(i+j-1)}[W] \} \\ &= \lambda_{i+j-1}[W]. \end{aligned}$$

The second inequality is just the first applied to $-A$. \square

1.4.3 Perturbation Theory

The field of perturbation theory for the symmetric eigenproblem is vast. We hand-picked the following results with regard to our needs in subsequent chapters. For more information, consult for instance [23, 24, 40, 51, 52, 63] and the references therein.

Remark 1.27 (Multiple Eigenvalues). For simplicity we will limit the presentation of the two results concerning perturbed eigenvectors (Theorems 1.29 and 1.31) to simple eigenvalues. Using the subspace-formulation of the Gap Theorem 1.21 it is easy to extend both to multiple eigenvalues by replacing $\text{gap}_A(\lambda)$ by $\text{gap}_A(I)$ in the statements and proofs, where $I = \{i \mid \lambda_i[A] = \lambda\}$. \diamond

Additive Perturbations

Every perturbation can be written in the form

$$A \mapsto A + E,$$

which makes the study of these perturbations useful in almost any application. The only thing we have to require is that E does not destroy symmetry, i.e., E must be symmetric itself.

As an immediate consequence of Weyl's Theorem 1.26 we get that eigenvalues of symmetric matrices are *perfectly conditioned*, at least in a normwise absolute sense. Related results are the Wielandt-Hoffmann inequality for a bound in the Frobenius-Norm and the Bauer-Fike Theorem as generalization to diagonalizable matrices, cf. [34, 56].

Corollary 1.28 (Eigenvalues under Additive Perturbations). *For symmetric $A, E \in \mathbb{R}^{n \times n}$,*

$$\max_k \left\{ |\lambda_k[A + E] - \lambda_k[A]| \right\} \leq \|E\|.$$

Proof. For any fixed k invoke Theorem 1.26 (with $U = A, V = E, j = 1$ and $i = k$ for the left bound, $i = n - k + 1$ for the right) to get

$$\lambda_k[A] + \lambda_1[E] \leq \lambda_k[A + E] \leq \lambda_k[A] + \lambda_n[E],$$

then use $\|E\| = \max \{|\lambda_1[E]|, |\lambda_n[E]|\}$. \square

Right of the bat we can get a cheap bound on the change in the eigenvector. Let $\lambda = \lambda_k[A]$, $\mathbf{q} = \mathbf{q}_k[A]$, $\tilde{\lambda} = \lambda_k[A + E]$, $\tilde{\mathbf{q}} = \mathbf{q}_k[A + E]$ and $\theta = \angle(\mathbf{q}, \tilde{\mathbf{q}})$. Noting that then $\text{gap}_A(\{k\}; \lambda) = \text{gap}_A(\{k\}) = \text{gap}_A(\lambda)$, the Gap Theorem 1.21 yields

$$\begin{aligned} \sin \theta \text{gap}_A(\lambda) &\leq \|A\tilde{\mathbf{q}} - \tilde{\mathbf{q}}\lambda\| = \|(A + E - E)\tilde{\mathbf{q}} - \tilde{\mathbf{q}}\lambda\| \\ &= \|(\tilde{\lambda} - \lambda)\tilde{\mathbf{q}} - E\tilde{\mathbf{q}}\| \leq 2\|E\|, \end{aligned} \tag{1.41}$$

where we needed Corollary 1.28 for a bound on $|\tilde{\lambda} - \lambda|$ in the last step. It turns out that the bound can be improved by (nearly) a factor of two, cf. [10, Thm. 5.4], but the price we pay is considerable more work for a proof. To extend the claim to multiple eigenvalues, see Remark 1.27.

Theorem 1.29 (Eigenvectors under Additive Perturbations)

Use the same notation as in (1.41). If λ is simple, then

$$\frac{1}{2} \sin 2\theta \leq \frac{\|E\|}{\text{gap}_A(\lambda)},$$

provided $2\|E\| < \text{gap}_A(\lambda)$. Note that $\theta \approx 0$ implies $\sin 2\theta \approx 2 \sin \theta$.

Proof. Redo (1.41) up until before the last step, where we bounded the norm of $r := (\tilde{\lambda} - \lambda)\tilde{q} - E\tilde{q}$ by $2\|E\|$. Now write

$$q^* \tilde{\lambda} \tilde{q} = q^*(A + E)\tilde{q} = q^* \lambda \tilde{q} + q^* E \tilde{q} \quad \Rightarrow \quad q^* E \tilde{q} = (\tilde{\lambda} - \lambda) q^* \tilde{q}.$$

Using (1.41) and the assumption $2\|E\| < \text{gap}_A(k)$ we know $\theta < \pi/2$, i.e., q and \tilde{q} cannot be orthogonal. Hence we see

$$r = RE\tilde{q} \quad \text{with} \quad R := \frac{\tilde{q} q^*}{q^* \tilde{q}} - I.$$

The matrix R is rather interesting since $R^2 = -R$. We will now proceed to prove $\|R\| = (q^* \tilde{q})^{-1}$; indeed this holds generally apart from the setting here for all such R and unit vectors q, \tilde{q} (cf. [10, Question 5.7]). To validate this claim, define $\eta := (q^* \tilde{q})^{-1} = (\cos \theta)^{-1}$, so that we have $\eta \tilde{q} = q + d$ with d orthogonal to q . Then $R = (q + d)q^* - I$ and hence $RR^* = (I - qq^*) + dd^*$, leading to

$$\|R\|^2 \leq 1 + \|d\|^2 = \eta^2 \quad \Rightarrow \quad \|R\| \leq \eta.$$

And since $d^* RR^* d = (1 + \|d\|^2)\|d\|^2$, this inequality is indeed an equality.

Now we can put everything together to obtain

$$\sin \theta \text{gap}_A(\lambda) \leq \|r\| = \|RE\tilde{q}\| \leq (\cos \theta)^{-1} \|E\|,$$

from where the identity $2 \sin \theta \cos \theta = \sin 2\theta$ completes the argument. \square

The preceding results are purely norm-based, as they rely solely on $\|E\|$. They are satisfactory and useful in many applications, but are too coarse to convey meaningful information for eigenvalues that are small or tightly clustered. In §2.3.3 we will present a more sophisticated approach to get bounds of finer granularity.

Multiplicative Perturbations

By “multiplicative” we mean perturbations in the form of congruence transformations

$$A \mapsto X^*AX,$$

but with the understanding that X is somehow “close to” the identity. Sometimes these are also called *outer* perturbations. We will consider the effects on eigenvalues and eigenvectors separately. In principle, the following results can be extended for nonsymmetric permutations $A \mapsto XAY$ [24].

The crucial fact to take from here is that multiplicative perturbations are more benign in the sense that they allow us to say something about the *relative* changes in eigenvalues as well as changes in the eigenvectors based on *relative* gaps in the spectrum.

Theorem 1.30 (Eigenvalues under Multiplicative Perturbations)

Let $A \in \mathbb{R}^{n \times n}$ be symmetric, $X \in \mathbb{R}^{n \times n}$ be nonsingular and $\alpha, \tilde{\alpha}$ be the k 'th eigenvalues of A and $\tilde{A} = X^*AX$, respectively. Then

$$|\alpha| \lambda_1[X^*X] \leq |\tilde{\alpha}| \leq |\alpha| \lambda_n[X^*X],$$

which implies

$$\tilde{\alpha} = \alpha(1 + \varepsilon) \quad \text{with} \quad |\varepsilon| \leq \|X^*X - I\|.$$

Proof. By Sylvester's Law, the k th eigenvalue of $X^*(A - \alpha)X$ is zero. Written as $\tilde{A} - \alpha X^*X$ we can interpret it as additive perturbation of \tilde{A} . Then Corollary 1.28, combined with the fact that X^*X is positive definite, leads to

$$|\tilde{\alpha}| = |\tilde{\alpha} - 0| \leq |\alpha| \|X^*X\| = |\alpha| \lambda_n[X^*X].$$

The same argument, based on \tilde{A} as the original matrix and $A = X^{-*}\tilde{A}X^{-1}$ as the perturbed one, and then using $\lambda_n[X^{-*}X^{-1}] = \lambda_n[X^{-1}X^{-*}] = \lambda_1[X^*X]^{-1}$ allows us to conclude

$$|\alpha| \lambda_1[X^*X] \leq |\tilde{\alpha}| \leq |\alpha| \lambda_n[X^*X].$$

Subtract $|\alpha|$ and use that by Sylvester's Law we have $\text{sign } \alpha = \text{sign } \tilde{\alpha}$, to see

$$|\tilde{\alpha}| - |\alpha| = |\tilde{\alpha} - \alpha| \leq |\alpha| \max \left\{ |\lambda_1[X^*X] - 1|, |\lambda_n[X^*X] - 1| \right\} = |\alpha| \|X^*X - I\|.$$

□

The following result concerning the effect of multiplicative perturbations on eigenvectors and its proof were taken from [10, Thm. 5.7] but modified to measure the relative gaps with respect to the original matrix instead of the perturbed one. Therefore the stated bound differs slightly. A generalization for invariant subspaces can be found in [52, Thm. 3.1] and for multiple eigenvalues, see Remark 1.27.

Theorem 1.31 (Eigenvectors under Multiplicative Perturbations)

Let $A \in \mathbb{R}^{n \times n}$ be symmetric, X be nonsingular and $\tilde{A} = X^*AX$. Let $\alpha, \tilde{\alpha}$ be the respective k 'th eigenvalues, both having multiplicity one, and with associated normalized eigenvectors $\mathbf{q}, \tilde{\mathbf{q}}$. Then

$$\frac{1}{2} \sin 2\theta \leq \frac{\varepsilon}{\text{relgap}_A(\alpha)} + \gamma, \quad \theta := \angle(\mathbf{q}, \tilde{\mathbf{q}}),$$

with $\varepsilon = \|X^*X\| \|X^{-*}X^{-1} - I\|$, $\gamma = \|X - I\|$, and where for $\alpha = 0$, $\varepsilon/\text{relgap}_A(\alpha)$ should be evaluated as zero.

Proof. Define $M = A - \tilde{\alpha}$ and $\delta M = \tilde{\alpha}(I - X^{-*}X^{-1})$ to get

$$M + \delta M = A - \tilde{\alpha}X^{-*}X^{-1} = X^{-*}(\tilde{A} - \tilde{\alpha})X^{-1}.$$

Hence, the k th eigenvalue of $M + \delta M$ is zero and $X\tilde{\mathbf{q}}$ is an (unnormalized) eigenvector belonging to it. Now it gets a bit subtle. We invoke Theorem 1.29 for

$$\frac{1}{2} \sin 2\theta_1 \leq \frac{\|\delta M\|}{\text{gap}_M(\alpha - \tilde{\alpha})}, \quad \theta_1 := \angle(\mathbf{q}, X\tilde{\mathbf{q}}).$$

Because M is just A shifted, $\text{gap}_M(\alpha - \tilde{\alpha}) = \text{gap}_A(\alpha)$. Furthermore, we can bound $\|\delta M\|$ by $|\alpha|\varepsilon$ using Theorem 1.30. Together this gives us

$$\frac{1}{2} \sin 2\theta_1 \leq \frac{\varepsilon}{\text{relgap}_A(\alpha)}.$$

The rest is trigonometry. With $\theta_2 := \angle(X\tilde{\mathbf{q}}, \tilde{\mathbf{q}})$ the triangle inequality gives $\sin \theta_2 \leq \|X - I\|$ and $\theta \leq \theta_1 + \theta_2$, as all angles lie in $[0, \pi/2]$. Hence

$$\begin{aligned} \sin 2\theta &\leq \sin(2\theta_1 + 2\theta_2) \\ &\leq \sin 2\theta_1 \cos 2\theta_2 + \sin 2\theta_2 \cos 2\theta_1 \\ &\leq \sin 2\theta_1 + \sin \theta_2, \end{aligned}$$

and therefore

$$\frac{1}{2} \sin 2\theta \leq \frac{\varepsilon}{\text{relgap}_A(\alpha)} + \|X - I\|.$$

□

Outer perturbations by a diagonal matrix. We want to leave this section with a small example to apply the previous results in a more concrete setting. Frequently, the arising multiplicative perturbations are given by a diagonal matrix that is a componentwise relative perturbation of the identity, i.e.,

$$A \mapsto DAD, \quad D = \text{diag}((1 + \delta_1), \dots, (1 + \delta_n)), \quad |\delta_i| \ll 1.$$

Then the bounds from Theorems 1.30 and 1.31 become

$$|\lambda_k[A]| \min_i \{(1 + \delta_i)^2\} \leq |\lambda_k[DAD]| \leq |\lambda_k[A]| \max_i \{(1 + \delta_i)^2\}, \quad (1.42)$$

for the eigenvalues and

$$\frac{1}{2} \sin 2\theta \leq \frac{\max_{i,j} \{(1 + \delta_i)^2 \cdot |(1 + \delta_j)^{-2} - 1|\}}{\text{relgap}_A(\{k\})} + \max_i \{|\delta_i|\}$$

for the eigenvectors.

These expressions are rather unwieldy. We can state them more compactly using our notation for error analysis from §1.2.3; indeed it might be a good idea to recall that notation before we embark to the next chapter, where we will make frequent use of it. If we have a matrix D as above, it is reasonable to assume that the entries stem from some kind of error analysis which gave us concrete bounds on the perturbations. Assume we know for some $m, p \in \mathbb{R}^{\geq 0}$ that

$$\delta_i \doteq \epsilon^{[p]}(m), \quad i = 1, \dots, n.$$

Then the rules (2) and (3) from Chart 1.11 let us compute $\|X - I\| \doteq \epsilon^{[p]}(m)$ and

$$\begin{aligned} \|X^*X\| &= \max_i \{(1 + \delta_i)^2\} \doteq 1 + \epsilon^{[p]}(2m), \\ \|X^*X - I\| &= \max_i \{|(1 + \delta_i)^2 - 1|\} \doteq \epsilon^{[p]}(2m), \\ \|X^{-*}X^{-1} - I\| &= \max_i \{|(1 + \delta_i)^{-2} - 1|\} \doteq \epsilon^{[p+1]}(2m). \end{aligned}$$

Hence, the bounds for eigenvalues and eigenvectors can be stated as

$$\begin{aligned} \lambda_k[DAD] &\doteq \lambda_k[A](1 + \epsilon^{[p]}(2m)), \\ \frac{1}{2} \sin 2\theta &\doteq (1 + \epsilon^{[p]}(2m)) \frac{\epsilon^{[p+1]}(2m)}{\text{relgap}_A(\{k\})} + \epsilon^{[p]}(m). \end{aligned} \quad (1.43)$$

Chapter 2

The MR³ Algorithm

“Elsa never really believed in the grail.
She thought she’d found a prize.”
“And what did you find, Dad?”
“Me? Illumination.”

— in *Indiana Jones and the Last Crusade* (1989)

The proven strategy to solve a dense real or complex symmetric eigenproblem numerically is to reduce the matrix to real tridiagonal form by an orthogonal similarity transformation [1, 34] and then solve the tridiagonal problem. Before there was MR³, usually one of three standard algorithms was employed for the second part.

QR-Iteration has been the workhorse since, well, forever. The current implementation `xSTEQR` in LAPACK [1] is rock-solid. At the time of this writing, the undisputed best method for computing all eigenvalues is the `dqds` algorithm [30, 64], which is nothing else but a supercharged QR-iteration. But for computing eigenvectors, QR exhibits a true $\mathcal{O}(n^3)$ complexity, making it very slow in practice. In fact, for the dense symmetric problem, the solution of the tridiagonal problem using QR can outweigh the reduction phase by far [18].

Divide & Conquer (DC) has been known quite a long time [8] but it took nearly 15 years until a stable implementation was found [39]. Complexity is $\mathcal{O}(n^3)$ in theory but on average DC behaves more like $\mathcal{O}(n^{2.5})$ in practice [10, 13], due to a large amount of work that can be outsourced to BLAS3-operations. The current implementation in LAPACK (`xSTEDC`) is fast, accurate and the method of choice for computing all eigenvalues and eigenvectors. The only conceivable drawback is that n^2 temporary workspace is needed.

Bisection & Inverse Iteration (BI) has as main advantage that a subset of k eigenpairs can be computed. But due to accuracy limitations of the inverse

iteration scheme employed, explicit Gram-Schmidt reorthogonalization is required for eigenvalues that are too close to each other (in an absolute sense), so the complexity remains $\mathcal{O}(k^2n)$.

The latest addition to the field has been the algorithm of *multiple relatively robust representations* [15, 17, 18], in short MRRR or MR³. It offers to compute k eigenpairs $(\bar{\lambda}_i, \bar{\mathbf{q}}_i)$, $\|\bar{\mathbf{q}}_i\| = 1$ of a symmetric tridiagonal matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ with residual norms and orthogonality levels of

$$\|(\mathbf{T} - \bar{\lambda}_i)\bar{\mathbf{q}}_i\| = \mathcal{O}(\|\mathbf{T}\|n\epsilon_\diamond), \quad |\bar{\mathbf{q}}_i^* \bar{\mathbf{q}}_j| = \mathcal{O}(n\epsilon_\diamond), \quad i \neq j, \quad (2.1)$$

in optimal $\mathcal{O}(kn)$ time. As such, it is a whole order of magnitude faster than BI.

The feature of being able to compute subsets of the spectrum is called *adaptability*. Combined with the fact that MR³ requires *no* communication for Gram-Schmidt reorthogonalization, this makes for an embarrassingly parallel algorithm.

In a recent thorough study [13] of the performances of the current tridiagonal eigensolvers in LAPACK, Demmel et al. made the following observations:

- (a) MR³ does the least amount of floating-point operations across the board.
- (b) DC is faster than MR³ if all or most of the eigendecomposition is wanted.
- (c) MR³ is not as accurate as DC or QR in general. The bounds in (2.1) are valid, but for some of their test cases the hidden constant in the \mathcal{O} -term lies in the hundreds.

The explanation for the seeming contradiction between (a) and (b) is that MR³ operates at a lower flop-rate. There are two reasons for this. First, MR³ does more divisions; as will become apparent during this chapter, MR³ needs $\mathcal{O}(kn)$ divisions to compute k eigenpairs. And second, DC can be arranged to do most of its work within matrix-matrix multiplications.

Their conclusion is that DC remains the method of choice when the full eigendecomposition is wanted, but for subsets MR³ is the clear winner.

Outline

In this chapter we will exhibit the theoretical side of MR³-algorithm in full splendor. It plays a central role for this thesis, because everything else quite literally revolves around MR³.

A short road map of this chapter is as follows: In §2.1 we present the algorithm in an abstract setting, just detailed enough to be able to give a full proof of correctness and error bounds in §2.2. Only then will we study twisted factorizations of symmetric tridiagonal matrices: §2.3 contains their definition, exposition of the main features and how they mesh with MR³, then §2.4 presents in-depth means—both old and new—for their computation. Finally, §2.5 holds topics of

a more practical flavor. There we will describe various techniques that we explored while engineering our own implementation of MR³. Our exposition will close with numerical experiments to compare our version with the most recent MR³-implementation from LAPACK 3.2.1.

Although our task for this thesis was focused on the bidiagonal SVD, out of necessity we gathered quite a bit of theoretical and practical expertise with MR³, some of which lead to improvements in the core algorithm itself. We regard the following as our main contributions in this chapter:

- A streamlined formulation of MR³ to provide the framework for integrating alternative kinds of representations for symmetric tridiagonal matrices; in particular we completely disentangled twisted factorizations from the presentation. This was in fact necessary groundwork for both Chapters 3 and 4, but on its own it might also serve as catalyst for future developments.
- Better ways to work with twisted factorizations, leading to increased accuracy and better efficiency.

Setting the stage

For the given symmetric tridiagonal we usually name the diagonal entries c_i and the offdiagonals e_i , that is,

$$\mathbb{T} = \text{diag}(c_1, \dots, c_n) + \text{diag}_{\pm 1}(e_1, \dots, e_{n-1}) \in \mathbb{R}^{n \times n}. \quad (2.2)$$

It is advisable to preprocess the given input matrix with regard to a couple of points. First of all, the entries should be scaled properly to fall into practicable range; specific guidelines can for example be found in [48]. If some offdiagonal entry is zero the matrix is *reducible* to two independent subproblems that can be handled separately. On the same note, if some offdiagonal entries are smaller than $\mathcal{O}(\epsilon_\diamond \|\mathbb{T}\|)$ they can safely be set to zero without serious ramifications for the quality of the results in (2.1). The process of eliminating small offdiagonal entries and breaking the problem into unreduced subproblems is called *splitting* the matrix. Once this has been done, one can always find a diagonal matrix \mathbb{X} with entries ± 1 (called a *signature* matrix) such that the offdiagonals of $\mathbb{X}\mathbb{T}\mathbb{X}$ are positive.

Combining these thoughts allows without loss of generality to restrict our attention to a matrix \mathbb{T} as in (2.2) that obeys

$$\epsilon_\diamond \|\mathbb{T}\| < e_i, \quad i = 1, \dots, n-1. \quad (2.3)$$

Remark 2.1. For future reference it will be useful to compile some of the special properties of an unreduced symmetric tridiagonal $\mathbb{T} \in \mathbb{R}^{n \times n}$:

1. All eigenvalues are simple (cf. [56, Thm. 7.7.1]).

2. No eigenvector has a zero in its first or last component and eigenvectors to the extremal eigenvalues have no zero components at all (cf. [56, Cor. 7.9.3]).
3. Let (λ, \mathbf{q}) be an eigenpair. Then for $k = 1, \dots, n$ the following are equivalent:
 - $k \neq 1$ and λ is an eigenvalue of $\mathbb{T}_{1:k-1}$
 - $k \neq n$ and λ is an eigenvalue of $\mathbb{T}_{k+1:n}$
 - $\mathbf{q}(k) = 0$

(follows from [56, Thm. 7.9.2]). In particular λ is not an eigenvalue of $\mathbb{T}_{1:n-1}$ nor of $\mathbb{T}_{2:n}$.

◇

2.1 MR³ in a Nutshell

There is a long line of research results that led to and contribute to the MR³-algorithm, some to a lesser and some to a larger degree. But the first time where all pieces were assembled and harnessed fully to form the algorithm we now call MR³, was in Inderjit Dhillon's thesis [15] under the supervision of Beresford Parlett. A completely revised description and a formal proof of correctness followed, distributed across two subsequent publications [17] and [18], the latter of which won its authors the SIAM/SIAG Linear Algebra Prize in 2006.

The discovery spawned a multitude of further results and developments. We cannot give an exhaustive overview but some publications that stand out are [59–61, 63] concerning relative perturbation theory, [19, 20, 53] about issues of relevance for a robust implementation, and [4, 70] to address problematic aspects in the original version and how they can be overcome.

In this section we want to develop the driving principles behind MR³ and then describe the algorithm in full. Our presentation differs in some aspects notably from the original version in the sources mentioned above.

We will restrict ourselves to what we call the *core* algorithm, meaning that we assume reasonable preparations like splitting and scaling have been done, so that we have an unreduced tridiagonal matrix as in (2.3) for which a subset of eigenpairs is to be computed.

2.1.1 The Idea

From a distant point of view, MR³ can be seen as a sophisticated variant of inverse iteration. Its salient feature is that no explicit reorthogonalization is needed, resulting in an $\mathcal{O}(n^2)$ complexity. This is achieved, basically, by computing the

vectors to such accuracy that they really have no other choice but to be orthogonal *automatically*. How this works can be explained in layman’s terms: If we manage to compute approximate eigenvectors $\bar{\mathbf{q}}_i$ and $\bar{\mathbf{q}}_j$ so that they have small angles to the respective exact eigenvectors,

$$\sin\angle(\bar{\mathbf{q}}_i, \mathbf{q}_i) = \mathcal{O}(n\epsilon_\diamond), \quad \sin\angle(\bar{\mathbf{q}}_j, \mathbf{q}_j) = \mathcal{O}(n\epsilon_\diamond), \quad (2.4)$$

then, provided $\|\bar{\mathbf{q}}_i\| = \|\bar{\mathbf{q}}_j\| = 1$, simple trigonometry gives

$$\bar{\mathbf{q}}_i^* \bar{\mathbf{q}}_j = \cos\angle(\bar{\mathbf{q}}_i, \bar{\mathbf{q}}_j) \leq \sin\angle(\bar{\mathbf{q}}_i, \mathbf{q}_i) + \sin\angle(\bar{\mathbf{q}}_j, \mathbf{q}_j) = \mathcal{O}(n\epsilon_\diamond). \quad (2.5)$$

The details of this idea will go into hibernation here and not resurface until the proof of Theorem 2.14 (Orthogonality of MR³) way down the road. Right now it will suffice to keep in mind that *accuracy yields orthogonality*.

Of course, accuracy has to come from somewhere and in particular on the levels stipulated in (2.4) it usually has to be earned. Standard inverse iteration can only deliver such accuracy if you got lucky in the choice of starting vector [46]. One step closer towards the intricacies that constitute MR³ reveals that there are two principles at work to overcome this hurdle.

Assume the matrix \mathbf{T} does “define” an eigenpair (λ, \mathbf{q}) that we want to “high relative accuracy”—the precise meaning of these terms will be defined later. Then we are able to compute an approximation $\bar{\lambda}$ of λ with

$$|\bar{\lambda} - \lambda| = \mathcal{O}(|\lambda|n\epsilon_\diamond) \text{ or even } \mathcal{O}(|\lambda|\epsilon_\diamond).$$

The first ingredient to MR³ is a new technique that allows in this situation to compute an approximation $\bar{\mathbf{q}}$ to the eigenvector with a residual norm that is small *compared to the eigenvalue*,

$$\|(\mathbf{T} - \bar{\lambda})\bar{\mathbf{q}}\| \leq \mathcal{O}(|\lambda|n\epsilon_\diamond). \quad (2.6)$$

The key is to exploit tridiagonal form. Using *twisted factorizations* one can determine, in $\mathcal{O}(n)$ time, a position where the true eigenvector has a large entry [27, 62]. Starting with the corresponding canonical vector on the right hand side, one step of inverse iteration will deliver a residual norm satisfying (2.6) and that is at most a factor \sqrt{n} away from being optimal [46]. We will explore twisted factorizations and their use for computing eigenvectors in detail in §2.3.

The reward of (2.6) is revealed by the Gap Theorem 1.21, since it gives

$$\sin\angle(\bar{\mathbf{q}}, \mathbf{q}) = \mathcal{O}(n\epsilon_\diamond/\text{relgap}_\mathbf{T}(\lambda)). \quad (2.7)$$

Hence, we can reach the goal (2.4) of highly accurate eigenvectors for all eigenvalues whose relative gap to the rest of the spectrum exceeds a constant *gaptol*, the *gap tolerance* (think of *gaptol* between 0.01 and 0.001 for double precision). Such eigenvalues are called *isolated* or *singletons*, the others are grouped into *clusters*.

It is surprisingly easy to state a formal definition for the concept of a cluster that looks fine at first glance but turns out to be ambiguous, due to relative gaps being nonsymmetric and centered around one eigenvalue. The following measure will help to avoid this trap.

Definition 2.2. The *relative distance* between two scalars $a, b \in \mathbb{R}$ is 0 if $a = b = 0$ and

$$\text{reldist}(a, b) := \frac{|a - b|}{\max\{|a|, |b|\}}$$

otherwise. ◇

Note that if λ_i are the eigenvalues of \mathbb{T} , then

$$\min \{ \text{reldist}(\lambda_{i-1}, \lambda_i), \text{reldist}(\lambda_i, \lambda_{i+1}) \} \leq \text{relgap}_{\mathbb{T}}(\lambda_i),$$

and with (1.6) one can extend this to sets of eigenvalues.

Definition 2.3 (Singletons & Clusters). Let $\lambda_i = \lambda_i[\mathbf{A}]$ be the eigenvalues of a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. Given a tolerance *gaptol*, we say that the spectrum of \mathbf{A} has a (*relative*) *gap* between i and $i + 1$ if

$$\text{reldist}(\lambda_i, \lambda_{i+1}) \geq \text{gaptol}.$$

A nonempty set $\{\lambda_a, \lambda_{a+1}, \dots, \lambda_b\}$ of consecutive eigenvalues of \mathbf{A} is *relatively separated* if

- $a = 1$ or there is a gap between $a - 1$ and a , and
- $b = n$ or there is a gap between b and $b + 1$, and
- for $i = a, \dots, b - 1$ there is no gap between i and $i + 1$.

A relatively separated set is called a *singleton* if it has just one element, otherwise it is a *cluster*. ◇

The second principle at work within MR³ deals with eigenvalues that are too close in a relative sense to make the bound (2.7) useful. It is just this: *Eigenvectors are shift-invariant, but relative gaps are not.* To exemplify the idea, take a shift $\tau \approx \lambda$ close to an eigenvalue to see that the relative gap with respect to the shifted matrix becomes

$$\text{relgap}_{\mathbb{T}-\tau}(\lambda - \tau) = \text{relgap}_{\mathbb{T}}(\lambda) \frac{|\lambda|}{|\lambda - \tau|} \gg \text{relgap}_{\mathbb{T}}(\lambda).$$

Thus we can increase relative gaps by shifting close.

Both principles taken together give the crude outline for a strategy:

- (1) Compute the eigenvalues to reveal relative gaps.
- (2) *For all singletons:* Compute the eigenvector such that (2.6) is fulfilled.
- (3) *For all clusters:* Shift close to the cluster. This will increase the relative gaps, so at least some should now exceed the tolerance; rinse and repeat.

This looks deceptively simple, but there is a catch. Doing the shifting in floating-point arithmetic will inevitably incur rounding errors so that the eigenvector-invariance is lost. To get numerically orthogonal vectors as in (2.5) one must ensure that the computed vectors will still be accurate *with respect to the original matrix*. Hence, we will basically need that rounding errors from the shifting do not affect the invariant subspace we are aiming for too severely.

We can summarize that for the strategy outlined above to work, each encountered shifted matrix has to define the desired eigenvalues and eigenvectors to high relative accuracy, and furthermore the shifting must be done so as not to spoil the invariant subspaces. The key ingredient to overcome both issues is what could be regarded as third principle underlying MR³, namely: *do not try to compute results to more accuracy than is warranted by the data* (cf. [15, p. 155]). The question of what data one should actually use to define a matrix leads to a crucial new concept.

Definition 2.4 (Representation). A set of $m \leq 2n - 1$ scalars, called the *data*, together with a mapping $f : \mathbb{R}^m \rightarrow \mathbb{R}^{2n-1}$ to define the entries of a symmetric tridiagonal matrix $\mathbb{T} \in \mathbb{R}^{n \times n}$ is called a *representation* of \mathbb{T} . \diamond

Definition 2.5 (Perturbation of a Representation). Let x_1, \dots, x_m be the scalars used to represent the symmetric tridiagonal matrix \mathbb{T} and $\tilde{x}_i = x_i(1 + \xi_i)$ be relative perturbations of them; using the same representational mapping they define a matrix $\tilde{\mathbb{T}}$.

If $|\xi_i| \leq \bar{\xi}$ for $i = 1, \dots, m$, we call $\mathbb{T} \rightsquigarrow \tilde{\mathbb{T}}$ a *componentwise* or *elementwise relative perturbation (erp)* bounded by $\bar{\xi}$, in short $\tilde{\mathbb{T}} = \text{erp}(\mathbb{T}, \bar{\xi})$. \diamond

A (partial) *relatively robust representation (RRR)* of a matrix \mathbb{T} is one where small relative changes in the data, in the form of an elementwise relative perturbation bounded by some constant $\bar{\xi}$, will cause only relative changes proportional to $\bar{\xi}$ in (some of) the eigenvalues and eigenvectors. We will quantify these notions in §2.2.

The obvious way to represent a matrix uses the diagonal and offdiagonal entries directly, that is, the representational mapping is just the identity on \mathbb{R}^{2n-1} . Unfortunately this will rarely be an RRR, at least not for the small eigenvalues. An alternative is to take the entries of a decomposition, e.g., an LU-decomposition $\mathbb{T} = \text{LDL}^*$ with L unit bidiagonal. In [15] compelling evidence was gathered that this will usually result in at least a partial RRR for the smallest eigenvalues,

which is why LDL*-factorizations were chosen as workhorse for the original MR³-algorithm. We will investigate them in-depth in §2.3.

The seminal paper [12] showed that bidiagonal factorizations of definite matrices will always give RRRs, as do entrywise representations of tridiagonals with a zero diagonal. The latter will play a prominent role in Chapter 3 and also provides an example why it is useful to allow $m < 2n - 1$ in the definition above.

On the notational side, we will not explicitly mention the representational mappings from here on but just say \mathbb{T} where we mean a representation of \mathbb{T} .

2.1.2 The Algorithm

The previously developed ideas are formalized in Algorithm 2.1. One particular non-standard feature of our version is that nothing is said about the kind of representation used at the nodes. The computed eigenpairs are denoted $(\bar{\lambda}_i, \bar{\mathbf{q}}_i)$ to distinguish them from the exact ones.

The Representation Tree

Although we have chosen an iterative layout, the basic nature of algorithm MR³ is a recursive one. Hence, the natural way to model the computation is as traversal of a tree, the *representation tree*. The *nodes* in this tree are the objects placed into and retrieved from the set \mathcal{S} ; they correspond to the iterations of the outermost loop. Each node has three features associated with it:

- a representation of a matrix \mathbf{M} ,
- an index set I of the eigenpairs that are to be computed for \mathbf{M} , called the *local eigenpairs*,
- the accumulated shift $\bar{\tau}$ from the root matrix.

Hence we can write nodes as triplet $(\mathbf{M}, I, \bar{\tau})$, but sometimes we may just write (\mathbf{M}, I) if the shift is not of interest. The *root* node is $(\mathbf{M}_0, I_0, 0)$ at the top of the tree.

Step 5 partitions the index set of a node (\mathbf{M}, I) as $I = I_1 \cup \dots \cup I_m$. This basically defines the layout of the tree. Each I_j is called a *child index set* of the node, but only if $|I_j| > 1$ does it lead to a *child node* $(\mathbf{M}_c, I_j, \bar{\tau}_c)$. Note that this implies that the index sets of all nodes except possibly the root have at least two elements.

For each computed eigenpair there is exactly one singleton child index set of a node somewhere. Conceptually we regard these as the *leaves* of the tree, but they are *not* nodes in the above sense, because no representation is attached. If $\{i\}$ is a child index set of (\mathbf{M}, I) , we say that the eigenpair $(\bar{\lambda}_i, \bar{\mathbf{q}}_i)$ was *computed at* the node (\mathbf{M}, I) .

ALGORITHM 2.1 *The MR³ Algorithm*

Compute selected eigenvalues and eigenvectors for symmetric tridiagonal T .

Input: Symmetric tridiagonal $T \in \mathbb{R}^{n \times n}$. Index set $I_0 \subseteq \{1, \dots, n\}$.

Output: Eigenpairs $(\bar{\lambda}_i, \bar{q}_i), i \in I_0$

Params: *gaptol* : The Gap Tolerance

-
- 1: Find a suitable representation M_0 for T , preferably definite, possibly by shifting T .
 - 2: $\mathcal{S} := \{(M_0, I_0, \bar{\tau} = 0)\}$
 - 3: **while** $\mathcal{S} \neq \emptyset$ **do**
 - 4: Remove one node $(M, I, \bar{\tau})$ from \mathcal{S}
 - 5: Approximate eigenvalues $[\lambda_i^{\text{loc}}], i \in I$ of M such that they can be classified into singletons and clusters according to *gaptol*; this gives a partition $I = I_1 \cup \dots \cup I_m$.
 - 6: **for** $r = 1$ **to** m **do**
 - 7: **if** $I_r = \{i\}$ **then** *// singleton*
 - 8: Refine eigenvalue approximation $[\lambda_i^{\text{loc}}]$ if necessary and use it to compute eigenvector \bar{q}_i
 - 9: $\bar{\lambda}_i := \lambda_i^{\text{loc}} + \bar{\tau}$
 - 10: **else** *// cluster*
 - 11: Refine the eigenvalue approximations at the borders (and/or inside) of the cluster if desired for more accurate shift-selection.
 - 12: Choose a suitable shift τ near the cluster and compute a representation of $M^+ = M - \tau$.
 - 13: Add new node $(M^+, I_r, \bar{\tau} + \tau)$ to \mathcal{S}
 - 14: **endif**
 - 15: **endfor**
 - 16: **endwhile**

For each node we define $d(\mathbf{M})$ to be its *depth*, that is, the number of edges on the (one and only) path leading from the root to it, so $d(\mathbf{M}_0) = 0$. Furthermore, for each $j \in I_0$ let $d(j)$ stand for the depth of the node where the eigenpair $(\bar{\lambda}_i, \bar{\mathbf{q}}_i)$ is computed. (Would we regard leaves $\{j\}$ as nodes, their depth should be $d(j) + 1$.)

Note that it is theoretically possible that a child node has the same index set as its father, but a node is uniquely identified by the combination of index set and depth.

(Some) Implementation Issues

At this point we would like to raise some more general points that pertain to any implementation of the algorithm. More specific information will follow in §2.5.

Remark 2.6 (Computation of eigenvalues; line 5). Because only a subset of eigenvalues is to be computed, some kind of bisection method is advisable to get intervals $[\lambda_i^{\text{loc}}]$ for each eigenvalue. The accuracy needs only be good enough to enable reliable classification; e.g., for $\text{gaptol} = 0.001$ getting about the first three decimal digits would suffice. For clusters the bounds can then be inflated a little and shifted by τ to serve as useful starting points for the eigenvalue refinement at child nodes. \diamond

Remark 2.7 (Computation of eigenvectors; line 8). At this point an initial approximation to the eigenvalue will be available from the classification step. In principle one could again use bisection to refine the eigenvalue to full accuracy before computing the vector, but for faster convergence a Rayleigh-Quotient Iteration (RQI) is usually deployed in practice. The eigenpair is improved until the residual norm was driven below some threshold. In §2.2 we discuss concrete criteria to gauge when the pair is acceptable and §2.3.2 contains more information about how the RQI can be set up. \diamond

Remark 2.8 (Shifting; line 12). This is one of the most critical and also most subtle steps in the whole algorithm. On one hand, we need to shift as close as possible or even inside the cluster to break up the relative gaps. For this it is usually sensible to follow the classification phase by further refinement of at least the intervals for cluster boundaries. On the other hand, the representation of the shifted matrix has to satisfy a series of requirements that will be compiled in §2.2, so one may have to back off from the cluster to get them right. \diamond

Remark 2.9 (The set \mathcal{S}). Management of \mathcal{S} determines the order in which the representation tree is traversed. This is irrelevant for the algorithm itself, but there are practical considerations. A *breadth-first* strategy can save workspace memory by storing the $2n - 1$ data elements of the node representations in the eigenvector matrix, because each node covers at least two eigenvalues. This is how MR³ was originally designed [20]. Alternatively a *depth-first* strategy can be employed, as for example proposed in [35]. The advantage is that it keeps the

option of backtracking, should insurmountable problems within a subtree arise—for instance that no suitable shift candidate can be found. Also, the copying of representation data to and from the eigenvector matrix is avoided. The only disadvantage of a *depth-first* strategy is that it requires to cap the maximal depth of the tree (to, say, 10) so that all representations on the longest path can be stored at once in a reserved area of workspace. \diamond

2.2 Proof of Correctness

We now set out to present, from scratch, a complete error analysis of the MR^3 -algorithm. Originally this was presented by the inventors in [17] and [18]; what is to follow will combine and, to some degree, extend their work.

Our presentation is more abstract, because we do not specify the type of representation used. In [17, 18] the proof was tightly intertwined with the use of bidiagonal and twisted factorizations; our intention is to drive home the notion that those do not need to be an integral part of the algorithm per se. Furthermore, the line of argumentation could be compressed and streamlined. We invested a considerable amount of work to do so, and the resulting proof is partly new. Nevertheless we wish to emphasize at this point that it cannot be regarded as novel, because since the beginnings it has been clear to everyone involved with MR^3 that one could in principle use different kinds of representations, it just had not been justified formally.

We need to stress that we feel the following pages to be an essential part of this thesis, because two of our main contributions hinge on it.

First and foremost, in Chapter 3 we will use MR^3 to compute the singular value decomposition of a bidiagonal matrix and establish a proof of the new method. For this to succeed we need to disassemble MR^3 and its proof, polish and tune the parts, and put them back together in a new way; in other words, we will need to refer to intermediate results in the proof that were not easily accessible in the original version [17].

And second, in §2.4 and Chapter 4 we investigate alternative ways to represent tridiagonal matrices and how they can be used within MR^3 . To be able to do so we need a clear understanding of where the representation's features interconnect with the dependencies of the algorithm.

The basic strategy we use is the same as in [17]: Identify a minimal set of (reasonable and practicable) requirements that are assumed to be fulfilled and on whose shoulders the proof can rest. The advantage of this approach is that it provides, as a side effect, a concise set of guidelines for an implementation, e.g., the conditions that have to be heeded when selecting a shift in step 12.

Recall that $(\bar{\lambda}_i, \bar{\mathbf{q}}_i)$ are the eigenpairs computed by algorithm MR^3 . We can make the simplifying assumption that the computed vectors are normed to unity *exactly*, $\|\bar{\mathbf{q}}_i\| \equiv 1$. Otherwise just replace $\bar{\mathbf{q}}_i$ by $\bar{\mathbf{q}}_i/\|\bar{\mathbf{q}}_i\|$ everywhere.

Name	Source	Function	Expected Size
C_{vecs}	RRR	controls change in eigenvectors	moderate (≈ 10)
C_{elg}	ELG	factor for element growth	moderate (≈ 10)
α_{\downarrow}	SHIFTREL	erp at father to link with child	$\mathcal{O}(\epsilon_{\diamond})$
α_{\uparrow}	SHIFTREL	erp at child to link with father	$\mathcal{O}(\epsilon_{\diamond})$
α_{\ddagger}	GETVEC	erp for vector computations	$\mathcal{O}(\epsilon_{\diamond})$
β_{\ddagger}	GETVEC	relative change to vector entries	$\mathcal{O}(n\epsilon_{\diamond})$
R_{gv}	GETVEC	factor for the residual bound	up to $\mathcal{O}(1/\text{gaptol})$

Table 2.1: Summary of parameters in the requirements for MR³. All are expected to be uniform over the whole tree. The rightmost column gives magnitudes of what might come up in practice for an average problem.

A List of Requirements

There are five distinct requirements a computational run of MR³ has to fulfill; they are compiled on page 55. The first three (RRR, ELG, RELGAPS) can be found in similar form in [17, p. 10-12]. How to attain the fourth (SHIFTREL) and fifth (GETVEC) for bidiagonal factorizations is the topic of [18]; stating them as requirements provides the capsule in which we hide the particular kind of representation used. The requirements declare a small set of constants to control the error bounds globally; an overview of them is given in Table 2.1, together with reasonable assumptions about their sizes for an average test case.

The first requirement RRR was already motivated in §2.1.1, namely that the local invariant subspaces have to be relatively robust in the face of small changes to a node’s representation data.

The stated condition only concerns eigenvectors, because that is all we need explicitly. We do *not* need that each eigenvalue within a cluster has to be relatively robust. As will become clear when we study relative condition numbers in §2.3.3, such a requirement would be very hard to guarantee for a cluster with moderate relative width (for example $\approx \sqrt{\epsilon_{\diamond}}$). In that regard our formulation differs notably from [17]. However, using the Gap Theorem 1.21, we see that fulfillment of the stated condition will imply that the *boundaries* of a cluster, as well as singleton eigenvalues, will be relatively robust, since they cannot change by more than $\mathcal{O}(C_{\text{vecs}}n\alpha|\lambda|)$.

Note the connection to Theorem 1.31: for erps which can be written as multiplicative perturbations one could basically set $C_{\text{vecs}} = 1$.

The requirement ELG concerns the *absolute* changes to matrix entries that result from *relative changes* to the representation data. For decomposition-based

Requirement 1: RRR (Relatively Robust Representations)

There is a constant C_{vecs} such that for any perturbation $\tilde{\mathbf{M}} = \text{erp}(\mathbf{M}, \alpha)$ at a node (\mathbf{M}, I) , the effect on the eigenvectors can be controlled as

$$\sin \angle(\mathcal{Q}_J[\mathbf{M}], \mathcal{Q}_J[\tilde{\mathbf{M}}]) \leq C_{\text{vecs}} n \alpha / \text{relgap}_{\mathbf{M}}(J),$$

for all $J \in \{I, I_1, \dots, I_r\}$ with $|J| < n$.

Requirement 2: ELG (Conditional Element Growth)

There is a constant C_{elg} such that for any perturbation $\tilde{\mathbf{M}} = \text{erp}(\mathbf{M}, \alpha)$ at a node (\mathbf{M}, I) , the incurred element growth is bounded by

$$\begin{aligned} \|\tilde{\mathbf{M}} - \mathbf{M}\| &\leq \text{spdiam}[\mathbf{M}_0], \\ \|(\tilde{\mathbf{M}} - \mathbf{M})\tilde{\mathbf{q}}_i\| &\leq C_{\text{elg}} n \alpha \text{spdiam}[\mathbf{M}_0] \quad \text{for each } i \in I. \end{aligned}$$

Requirement 3: RELGAPS (Relative Gaps)

For each node (\mathbf{M}, I) , the classification of I into child index sets in step 5 is done such that for $r = 1, \dots, m$, $\text{relgap}_{\mathbf{M}}(I_r) \geq \text{gaptol}$ (if $|I_r| < n$).

Requirement 4: SHIFTRREL (Shift Relation)

There exist constants $\alpha_{\downarrow}, \alpha_{\uparrow}$ such that for every node with matrix \mathbf{H} that was computed using shift τ as child of \mathbf{M} , there are perturbations

$$\check{\mathbf{M}} = \text{erp}(\mathbf{M}, \alpha_{\downarrow}) \quad \text{and} \quad \hat{\mathbf{H}} = \text{erp}(\mathbf{H}, \alpha_{\uparrow})$$

with which the exact shift relation $\check{\mathbf{M}} - \tau = \hat{\mathbf{H}}$ is attained.

Requirement 5: GETVEC (Computation of Eigenvectors)

There exist constants $\alpha_{\ddagger}, \beta_{\ddagger}$ and R_{gv} with the following property: Let $(\bar{\lambda}^{\text{leaf}}, \bar{\mathbf{q}})$ with $\bar{\mathbf{q}} = \bar{\mathbf{q}}_i$ be computed at node (\mathbf{M}, I) , where $\bar{\lambda}^{\text{leaf}}$ is the final local eigenvalue approximation. Then we can find elementwise perturbations to the matrix and the vector,

$$\tilde{\mathbf{M}} = \text{erp}(\mathbf{M}, \alpha_{\ddagger}), \quad \bar{\mathbf{q}} \rightsquigarrow \tilde{\mathbf{q}}, \quad \tilde{\mathbf{q}}(j) \doteq \bar{\mathbf{q}}(j)(1 + \beta_{\ddagger}),$$

for which the residual norm is bounded as

$$\|\mathbf{r}^{\text{leaf}}\| := \|(\tilde{\mathbf{M}} - \bar{\lambda}^{\text{leaf}})\tilde{\mathbf{q}}\| / \|\tilde{\mathbf{q}}\| \leq R_{\text{gv}} n \epsilon_{\diamond} \text{gap}_{\tilde{\mathbf{M}}}(\{i\}; \bar{\lambda}^{\text{leaf}}).$$

representations this is called *element growth* (*elg*). Note that the requirement becomes mute if the matrix is represented by its entries directly.

The formulation of the two conditions that constitute the requirement are quite deliberately chosen. Combined they convey that even huge element growth is permissible (first condition), but only in those entries where the local eigenvectors of interest have tiny entries (second condition). This meaning is not really affected by the fact that the second condition is stated using the *computed* vectors (as it is done in [17] as well) instead of the exact ones. An advantage of doing it this way is that it provides a direct and easily checkable criterion to see if the computation is on the right track. In fact one could still do the proof if $\bar{\mathbf{q}}_i$ were replaced by \mathbf{q}_i here, but it would require more work.

A final note concerning element growth. In the requirement it is measured globally, with respect to the spectral diameter (of the root). A stronger condition would compare $\tilde{\mathbf{M}}(i, j)$ to $\mathbf{M}(i, j)$ directly (in particular the diagonal entries) to evaluate what we like to call *local* element growth, but that is not necessary at this point. Dhillon conjectured [15, p. 123] that absence of local element growth implies relative robustness, at least for the eigenvalues of small magnitude, but as of now this claim has not been proven. We will come back to the connection between element growth and relative robustness in §2.5.

The third requirement RELGAPS conveys that the structure of the representation tree has to be consistent with our definition of relative gaps. With regard to the coming proof, RELGAPS is the key to allow the use of both RRR and GETVEC. It influences two separate aspects of algorithm MR³.

The first is of course step 5. But it should be clear that, if the eigenvalues are approximated accurately enough and the classification is done in accordance with Definition 2.3, fulfillment of the requirement should not be an issue.

The other concern is slightly hidden: the requirement also touches on the *outer* relative gaps of the whole local subset at the node. This is because

$$\max_r \{ \text{relgap}_{\mathbf{M}}(I_r) \} \leq \text{relgap}_{\mathbf{M}}(I),$$

so fulfillment of the requirement implies $\text{relgap}_{\mathbf{M}}(I) \geq \text{gaptol}$. Stated the other way around, the requirement cannot be fulfilled if $\text{relgap}_{\mathbf{M}}(I) < \text{gaptol}$. This fact has to be kept in mind when the node is created, in particular during evaluation of shifts for a new child in step 12. Furthermore, it has to hold for the root node and I_0 . If it does not when only a subset of eigenpairs is desired, $I_0 \subsetneq \{1, \dots, n\}$, one has to deploy special countermeasures to deal with this problem [53].

Requirement SHIFTRREL provides the glue that connects the nodes in the tree. It basically states that the computations of the shifted representations have to be done in a mixed relatively stable way. In §2.4 we will deal extensively with algorithms to achieve this for twisted factorizations.

Note that the perturbation $\tilde{M} = \text{erp}(M, \alpha_{\downarrow})$ at the father will in general be different for each of its child nodes, but each child node has just one perturbation governed by α_{\uparrow} to establish the link to its father.

Finally, GETVEC captures what was postulated at the beginning of §2.1.1, in (2.6), namely that the vectors computed in step 8 have to have residual norms that are small, even when compared to the eigenvalue. The formulation of the requirement is relaxed in that we may perturb the representation and the vector entries to actually obtain the bound. In §2.3.2 we will explore how twisted factorizations can be employed to achieve GETVEC.

It must seem strange that the required residual bound is given in terms of the gap (of the perturbed matrix) instead of the eigenvalue. In the situation at hand we are talking about a singleton eigenvalue, meaning that the gap will be of about the same magnitude as the eigenvalue. Assuming $\bar{\lambda} = \bar{\lambda}^{\text{leaf}}$ approximates $\lambda = \lambda_i[M]$ to high relative accuracy, that is, $\bar{\lambda} = \lambda(1 + \mathcal{O}(n\epsilon_{\diamond}))$, for all intents and purposes we will have $\text{gap}_{\tilde{M}}(\bar{\lambda}) \sim \text{gap}_M(\lambda)$ and therefore

$$|\bar{\lambda}| \text{gap}_{tol} \lesssim \text{gap}_{\tilde{M}}(\bar{\lambda}).$$

Hence we could also have required a residual norm of $\mathcal{O}(n\epsilon_{\diamond}|\bar{\lambda}|\text{gap}_{tol})$. However, formulating the condition based on the gap eases access to the Gap Theorem 1.21. Furthermore, quite often we have the situation that the eigenvalue of interest has a relative separation far better than gap_{tol} . Then this formulation clearly states that high relative accuracy in the eigenvalue is not strictly necessary. As such, it also provides a practicable convergence criterion for the iteration in step 8.

This closes the requirements and we can now begin actually proving something. The two goals will be to obtain error bounds for the final residuals (Theorem 2.10) and orthogonality levels (Theorem 2.14). Three intermediate results in the form of lemmas will pave the way. Not all of the above requirements have to be fulfilled for each result: see Figure 2.1 to get an overview of the logical dependencies.

Residual Norms

In the following we will derive a first-order bound for the residual norms of the computed vectors \bar{q}_i , with respect to the root matrix and an ideal eigenvalue approximation λ^* which one would get if the shift-accumulations in lines 9 and 13 were done exactly. For the actually returned eigenvalue $\bar{\lambda}_i$, the bound would have to be adjusted minimally to take into account the rounding errors from adding the shifts together. Alternatively, it might be a better idea to return the Rayleigh Quotient of the computed vector instead to minimize the residual norm.

The residual r^{leaf} used in the statement of the theorem is the one with respect to the local eigenvalue $\bar{\lambda}^{\text{leaf}}$, as specified by GETVEC.

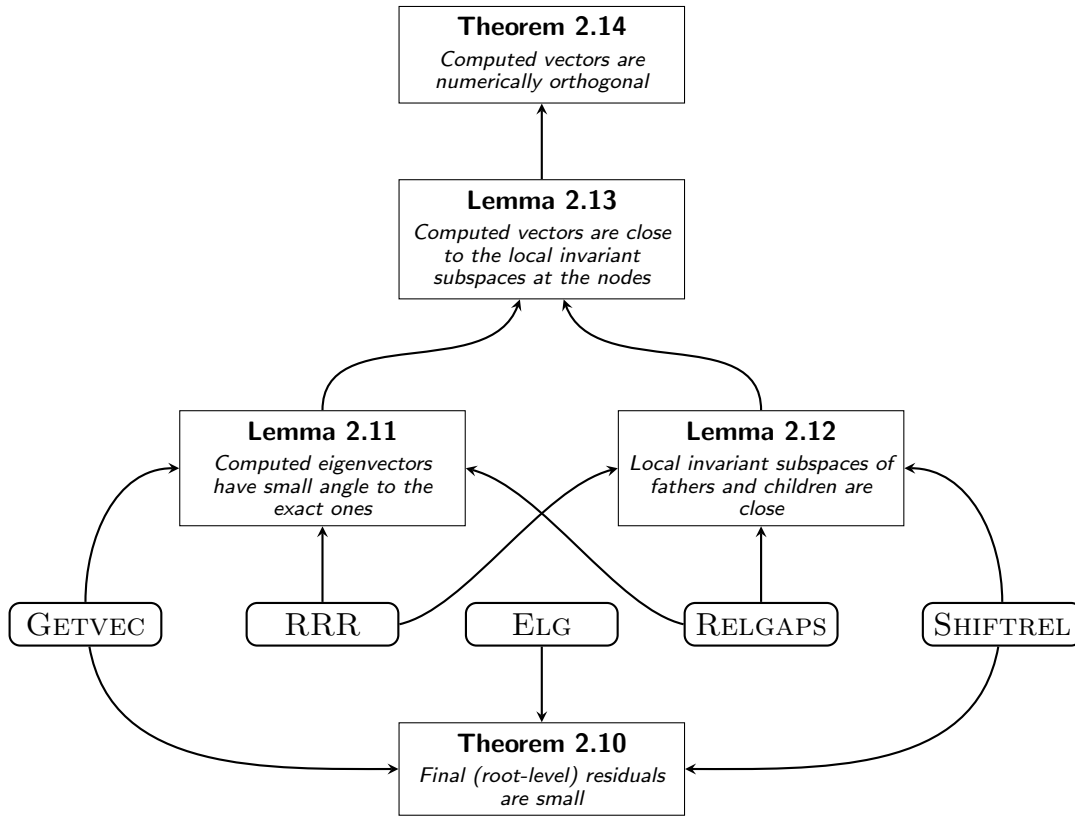


Figure 2.1: Outline and logical structure of the proof of correctness for MR³.

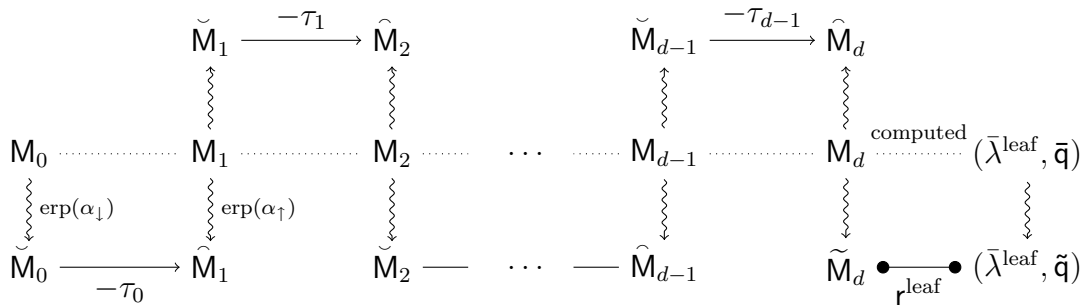


Figure 2.2: The computed eigenpairs relate to the root along an alternating sequence of matrix perturbations and exact shift relations.

Theorem 2.10 (Residual norms for MR³)

Let the representation tree traversed by Algorithm MR³ satisfy requirements ELG, SHIFTRREL and GETVEC.

For given index $j \in I_0$, let $d = d(j)$ and $\mathbf{M}_0, \mathbf{M}_1, \dots, \mathbf{M}_d$ be the representations along the path from the root (\mathbf{M}_0, I_0) to the node where $\bar{\mathbf{q}} = \bar{\mathbf{q}}_j$ was computed, with shifts τ_i linking \mathbf{M}_i and \mathbf{M}_{i+1} , respectively (cf. Figure 2.2).

Define $\lambda^* := \tau_0 + \dots + \tau_{d-1} + \bar{\lambda}^{\text{leaf}}$. Then

$$\|(\mathbf{M}_0 - \lambda^*)\bar{\mathbf{q}}\| \leq \left(\|r^{\text{leaf}}\| + \gamma \text{spdiam}[\mathbf{M}_0] \right) \frac{(1 + \beta_{\ddagger})}{(1 - \beta_{\ddagger})},$$

where $\gamma := C_{\text{elg}} n (d(\alpha_{\downarrow} + \alpha_{\uparrow}) + \alpha_{\ddagger}) + 2(d + 1)\beta_{\ddagger}$.

Proof. Requirement SHIFTRREL gives us along the path perturbations

$$\check{\mathbf{M}}_k = \text{erp}(\mathbf{M}_k, \alpha_{\downarrow}) \quad \text{and} \quad \hat{\mathbf{M}}_{k+1} = \text{erp}(\mathbf{M}_{k+1}, \alpha_{\uparrow}),$$

such that

$$\check{\mathbf{M}}_k - \tau_k = \hat{\mathbf{M}}_{k+1}, \quad k = 0, \dots, d-1.$$

Furthermore, GETVEC specifies another perturbed matrix

$$\tilde{\mathbf{M}}_d = \text{erp}(\mathbf{M}_d, \alpha_{\ddagger}), \quad \tilde{\mathbf{q}} = \bar{\mathbf{q}} + \delta\bar{\mathbf{q}}, \quad \|\delta\bar{\mathbf{q}}\| \leq \beta_{\ddagger},$$

with $r^{\text{leaf}} = (\tilde{\mathbf{M}}_d - \bar{\lambda}^{\text{leaf}})\tilde{\mathbf{q}}/\|\tilde{\mathbf{q}}\|$. The complete situation is depicted in Figure 2.2. We can link the representations on the path using a telescope sum:

$$\begin{aligned} \mathbf{M}_0 - \lambda^* &= \mathbf{M}_d - \bar{\lambda}_{\text{leaf}} + \sum_{k=0}^{d-1} [(\mathbf{M}_k - \tau_k) - \mathbf{M}_{k+1}] \\ &= \mathbf{M}_d - \bar{\lambda}_{\text{leaf}} \\ &\quad + \sum_{k=0}^{d-1} [(\mathbf{M}_k - \check{\mathbf{M}}_k) + \underbrace{(\check{\mathbf{M}}_k - \tau_k) - \hat{\mathbf{M}}_{k+1}}_{=0} + (\hat{\mathbf{M}}_{k+1} - \mathbf{M}_{k+1})] \\ &= (\tilde{\mathbf{M}}_d - \bar{\lambda}_{\text{leaf}}) + (\mathbf{M}_d - \tilde{\mathbf{M}}_d) \\ &\quad + \sum_{k=0}^{d-1} [(\mathbf{M}_k - \check{\mathbf{M}}_k) + (\hat{\mathbf{M}}_{k+1} - \mathbf{M}_{k+1})]. \end{aligned}$$

Multiply by $\tilde{\mathbf{q}} = \bar{\mathbf{q}} + \delta\bar{\mathbf{q}}$ and take norms. Then employ $\|\delta\bar{\mathbf{q}}\| \leq \beta_{\ddagger}$, $\|\tilde{\mathbf{q}}\| \leq 1 + \beta_{\ddagger}$ and the bounds provided by GETVEC and ELG to obtain

$$\begin{aligned} \|(\mathbf{M}_0 - \lambda^*)\tilde{\mathbf{q}}\| &\leq \|\mathbf{r}^{\text{leaf}}\|(1 + \beta_{\ddagger}) \\ &\quad + C_{\text{elg}}(d(\alpha_{\downarrow} + \alpha_{\uparrow}) + \alpha_{\ddagger}) n \text{spdiam}[\mathbf{M}_0] \\ &\quad + \beta_{\ddagger}(2d + 1) \text{spdiam}[\mathbf{M}_0] \\ &= \|\mathbf{r}^{\text{leaf}}\|(1 + \beta_{\ddagger}) + (\gamma - \beta_{\ddagger}) \text{spdiam}[\mathbf{M}_0] \\ &=: \Delta. \end{aligned}$$

Now we only need to replace $\tilde{\mathbf{q}}$ by $\bar{\mathbf{q}}$ on the left. By Theorem 1.20 the above bound on the residual norm implies the existence of an eigenvalue λ of \mathbf{M}_0 with $|\lambda - \lambda^*| \leq \Delta/\|\tilde{\mathbf{q}}\|$. This gives

$$\begin{aligned} \|(\mathbf{M}_0 - \lambda^*)\bar{\mathbf{q}}\| &\leq \Delta + \|(\mathbf{M}_0 - \lambda^*)\delta\bar{\mathbf{q}}\| \\ &\leq \Delta + \|(\mathbf{M}_0 - \lambda)\delta\bar{\mathbf{q}}\| + \|(\lambda - \lambda^*)\delta\bar{\mathbf{q}}\| \\ &\leq \Delta + \beta_{\ddagger} \text{spdiam}[\mathbf{M}_0] + \Delta\|\delta\bar{\mathbf{q}}\|/\|\tilde{\mathbf{q}}\| \\ &\leq (\Delta + \beta_{\ddagger} \text{spdiam}[\mathbf{M}_0]) (1 + \|\delta\bar{\mathbf{q}}\|/\|\tilde{\mathbf{q}}\|), \end{aligned}$$

and noting $\|\delta\bar{\mathbf{q}}\|/\|\tilde{\mathbf{q}}\| \leq \beta_{\ddagger}/(1 - \beta_{\ddagger})$ the claim follows. \square

Orthogonality

For proving that the computed vectors are numerically orthogonal, it will again be necessary to propagate them up the tree. The key is to show that each $\bar{\mathbf{q}}_i$ has a small angle to *all* invariant subspaces $\mathcal{Q}_I[\mathbf{M}]$ of ancestors in the tree, which are just all nodes (\mathbf{M}, I) with $I \ni i$. Note that if the shifting were done in exact arithmetic, we would have containment, $\bar{\mathbf{q}}_i \in \mathcal{Q}_I[\mathbf{M}]$, so all these angles would be zero.

At this point it might be a good idea to recall §1.3 about angles between subspaces. In particular we will have to make frequent use of the transitivity law in Lemma 1.18.

We start by harnessing GETVEC to establish that the computed vectors are close to the corresponding exact eigenvector at the node where they are computed.

Lemma 2.11 (Accuracy of computed vectors). *Let the representation tree traversed by Algorithm MR³ satisfy Requirements GETVEC, RRR and RELGAPS. Then a vector $\bar{\mathbf{q}}_i$ computed at node (\mathbf{M}, I) will obey*

$$\sin\angle(\mathbf{q}_i[\mathbf{M}], \bar{\mathbf{q}}_i) \leq C_{\text{vecs}}\alpha_{\ddagger}n/\text{gaptol} + R_{\text{gv}}n\epsilon_{\diamond} + \beta_{\ddagger} =: \mathcal{R}n\epsilon_{\diamond}, \quad (2.8)$$

defining \mathcal{R} .

Proof. Let $\tilde{\mathbf{M}}$ and $\tilde{\mathbf{q}}$ be the perturbed matrix and vector specified by GETVEC. Then we can use Lemma 1.18 for

$$\sin\angle(\mathbf{q}_i[\mathbf{M}], \bar{\mathbf{q}}_i) \leq \sin\angle(\mathbf{q}_i[\mathbf{M}], \mathbf{q}_i[\tilde{\mathbf{M}}]) + \sin\angle(\mathbf{q}_i[\tilde{\mathbf{M}}], \tilde{\mathbf{q}}_i) + \sin\angle(\tilde{\mathbf{q}}_i, \bar{\mathbf{q}}_i).$$

We will proceed by considering each of the three summands on the right separately. Plugging the residual bound from GETVEC into the Gap Theorem 1.21 yields

$$\sin\angle(\mathbf{q}_i[\tilde{\mathbf{M}}], \tilde{\mathbf{q}}_i) \leq R_{\text{gv}} n \epsilon_{\diamond},$$

which takes care of the middle term. If $\bar{\mathbf{q}}_i$ is computed at node (\mathbf{M}, I) then i must be a singleton eigenvalue, i.e., $\{i\}$ is a child index set of I . Thus, by requirement RELGAPS, its relative gap exceeds *gaptol*. Consequently, requirement RRR allows to bound the first term as

$$\sin\angle(\mathbf{q}_i[\mathbf{M}], \mathbf{q}_i[\tilde{\mathbf{M}}]) \leq C_{\text{vecs}} \alpha_{\dagger} n / \text{gaptol}.$$

Finally we have to consider the componentwise perturbation $\bar{\mathbf{q}}_i \rightsquigarrow \tilde{\mathbf{q}}_i$ to the computed vector's entries. This we can write as $\tilde{\mathbf{q}}_i = \mathbf{X} \bar{\mathbf{q}}_i$, where \mathbf{X} is diagonal holding the individual relative perturbations. But then the triangle inequality gives

$$\sin\angle(\tilde{\mathbf{q}}_i, \bar{\mathbf{q}}_i) \leq \|(\mathbf{X} - \mathbf{I}) \bar{\mathbf{q}}_i\| \leq \|\mathbf{X} - \mathbf{I}\| \leq \beta_{\dagger}.$$

□

The next result provides control over the shifts, as it shows how to relate an invariant subspace of a child to the corresponding subspace at the father.

Lemma 2.12 (Invariant subspace relations). *Let the representation tree traversed by Algorithm MR^3 satisfy requirements RRR, SHIFTRREL and RELGAPS. Then for each node (\mathbf{M}, I) with child (\mathbf{H}, J) ,*

$$\sin\angle(\mathcal{Q}_J[\mathbf{M}], \mathcal{Q}_J[\mathbf{H}]) \leq C_{\text{vecs}} (\alpha_{\downarrow} + \alpha_{\uparrow}) n / \text{gaptol}.$$

Proof. Let τ be the shift used to compute \mathbf{H} and $\tilde{\mathbf{M}}, \hat{\mathbf{H}}$ be the perturbed versions of father and child as specified by requirement SHIFTRREL. Then $\hat{\mathbf{H}} = \tilde{\mathbf{M}} - \tau$ and so $\mathcal{Q}_J[\tilde{\mathbf{M}}] = \mathcal{Q}_J[\hat{\mathbf{H}}]$. Hence, Lemma 1.18 gives

$$\sin\angle(\mathcal{Q}_J[\mathbf{M}], \mathcal{Q}_J[\mathbf{H}]) \leq \sin\angle(\mathcal{Q}_J[\mathbf{M}], \mathcal{Q}_J[\tilde{\mathbf{M}}]) + \sin\angle(\mathcal{Q}_J[\hat{\mathbf{H}}], \mathcal{Q}_J[\mathbf{H}]).$$

Now, as J is a child index set of I , RELGAPS implies $\text{relgap}_{\mathbf{M}}(J) \geq \text{gaptol}$ as well as $\text{relgap}_{\mathbf{H}}(J) \geq \text{gaptol}$. With the help of requirement RRR we can therefore bound the two terms on the right in the inequality above separately as

$$\begin{aligned} \sin\angle(\mathcal{Q}_J[\mathbf{M}], \mathcal{Q}_J[\tilde{\mathbf{M}}]) &\leq C_{\text{vecs}} \alpha_{\downarrow} n / \text{gaptol}, \\ \sin\angle(\mathcal{Q}_J[\mathbf{H}], \mathcal{Q}_J[\hat{\mathbf{H}}]) &\leq C_{\text{vecs}} \alpha_{\uparrow} n / \text{gaptol}, \end{aligned}$$

and the claim follows. □

Now a straightforward argument based on Lemma 2.11 and using Lemma 2.12 inductively allows to link the computed vectors to invariant subspaces higher up in the tree.

Lemma 2.13 (Computed vectors and the local invariant subspaces). *Let the conditions for Lemmas 2.11 and 2.12 be fulfilled. Then for each node (\mathbf{M}, I) in the tree with child index set $J \subseteq I$, the computed vectors $\bar{\mathbf{q}}_j, j \in J$ will obey*

$$\sin\angle(\mathcal{Q}_J[\mathbf{M}], \bar{\mathbf{q}}_j) \leq \mathcal{R}n\epsilon_\diamond + C_{\text{vecs}}(d(j) - d(\mathbf{M}))(\alpha_\downarrow + \alpha_\uparrow)n/\text{gaptol},$$

with \mathcal{R} as in (2.8).

Proof. If the eigenvalue j is a singleton, $J = \{j\}$, then $d(j) = d(\mathbf{M})$ and the claim is just Lemma 2.11.

For the case $|J| > 1$, let (\mathbf{H}, J) be the corresponding child node and τ the shift leading to it. Fix any index $j \in J$ and let N be the unique child index set of J that contains j . Note that N may or may not be a singleton. In any case,

$$\begin{aligned} \sin\angle(\mathcal{Q}_J[\mathbf{M}], \bar{\mathbf{q}}_j) &\leq \sin\angle(\mathcal{Q}_J[\mathbf{M}], \mathcal{Q}_J[\mathbf{H}]) + \sin\angle(\mathcal{Q}_J[\mathbf{H}], \bar{\mathbf{q}}_j) \quad \text{using Lemma 1.18,} \\ &\leq \sin\angle(\mathcal{Q}_J[\mathbf{M}], \mathcal{Q}_J[\mathbf{H}]) + \sin\angle(\mathcal{Q}_N[\mathbf{H}], \bar{\mathbf{q}}_j) \quad \text{by Cor. 1.17, as } N \subseteq J. \end{aligned}$$

The first term on the right is handled by Lemma 2.12. For the second term we employ the induction hypothesis, noting that $j \in N$, N is a child index set of J and $d(\mathbf{H}) = d(\mathbf{M}) + 1$; together they complete the argument. \square

Finally, the crown jewel lies within our grasp. Based on the previous results, it just formalizes the notion raised at the beginning of §2.1.1, in (2.4) and (2.5), namely that accuracy yields orthogonality.

Theorem 2.14 (Orthogonality of MR³)

Let the representation tree traversed by Algorithm MR³ satisfy the conditions for Lemma 2.13, and let $d_{\max} := \max\{d(i) \mid i \in I_0\}$ be the maximal depth of a node in the tree. Then any two computed vectors $\bar{\mathbf{q}}_i$ and $\bar{\mathbf{q}}_j$ will obey

$$\frac{1}{2}\bar{\mathbf{q}}_i^* \bar{\mathbf{q}}_j \leq \mathcal{R}n\epsilon_\diamond + C_{\text{vecs}}d_{\max}(\alpha_\downarrow + \alpha_\uparrow)n/\text{gaptol},$$

with \mathcal{R} as in (2.8).

Proof. Let (M, N) be the unique deepest ancestor of both i and j , meaning that N has child index sets $I \neq J$ with $i \in I$ and $j \in J$. From Lemma 2.13 we know that \bar{q}_i and \bar{q}_j can be written as

$$\bar{q}_i = x + r, \quad x \in \mathcal{Q}_I[M], \quad x \perp r, \quad \|r\| = \sin \angle(\mathcal{Q}_I[M], \bar{q}_i), \quad (2.9)$$

$$\bar{q}_j = y + s, \quad y \in \mathcal{Q}_J[M], \quad y \perp s, \quad \|s\| = \sin \angle(\mathcal{Q}_J[M], \bar{q}_j), \quad (2.10)$$

and we have bounds on $\|r\|$ and $\|s\|$. The situation is depicted in Figure 2.3. Hence,

$$\bar{q}_i^* \bar{q}_j = \underbrace{x^* y}_{=0} + x^* s + r^* y + r^* s = r^* \bar{q}_j + x^* s \quad (= \bar{q}_i^* s + r^* y),$$

and as we assume $\|\bar{q}_i\| = \|\bar{q}_j\| = 1$,

$$|\bar{q}_i^* \bar{q}_j| \leq |r^* \bar{q}_j| + |x^* s| \leq \|r\| + \|s\|,$$

where we had to invoke the Cauchy-Schwartz inequality for the last step. \square

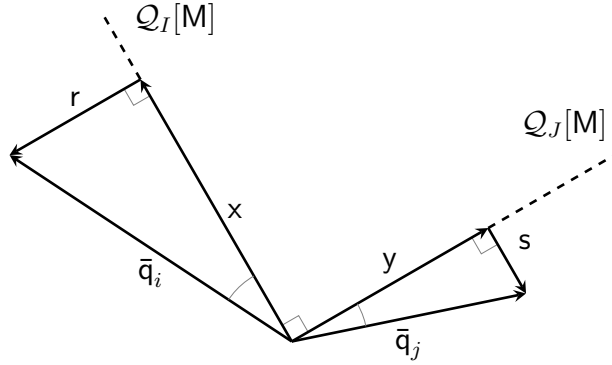


Figure 2.3: Situation for the proof of Theorem 2.14.

This completes what we set out to do. It should be noted that the error bounds are all of a worst-case nature. The accuracy delivered by MR³ is in general far better than these would suggest, as we will see in §2.5. Nevertheless they are also sharp in the following sense: running MR³ with a gap tolerance of, say, $gaptol = 0.001$, one must be prepared to live with orthogonality levels of about $\mathcal{O}(1000n\epsilon_\circ)$, because they can occur even if all of the requirements are fulfilled with very benign parameters.

2.3 Twisted Factorizations and their role(s) in MR³

We have seen that MR³ can in principle be set up using any kind of representation at the nodes, as long as the five requirements are fulfilled. However, the latter

is sometimes nontrivial, and this is particularly true for RRR and GETVEC. In general it would not be possible to attain these if tridiagonal matrices were represented by their entries.

In its original form, MR³ used decompositions resulting from Gaussian elimination, specially tailored for the tridiagonal case, to achieve both RRR and GETVEC. Top-to-bottom LDL*-type factorizations were reevaluated as favorable way to represent symmetric tridiagonals with regard to relative sensitivity of eigenvalues. For computing eigenvectors under the restrictions posed by GETVEC, an extension based on so called *twisted factorizations* was employed.

In this section we present these techniques and their interplay with MR³. We start gently, by revisiting the standard bidiagonal factorizations, which can be regarded as generalized Cholesky decompositions. Then twisted factorizations are introduced. In §2.3.2 we will see how the latter allow to compute accurate eigenvectors (GETVEC), and §2.3.3 gives a short introduction to the favorable properties of bidiagonal factorizations with regard to small relative perturbations in the data (RRR). The only thing missing, namely how to shift twisted factorizations with mixed stability (SHIFTREL), will then be the exclusive topic of §2.4.

2.3.1 Burn At Both Ends

Top-to-Bottom

If \mathbf{T} is positive definite we can compute a Cholesky-decomposition $\mathbf{B}^*\mathbf{B}$ with a bidiagonal matrix \mathbf{B} . For indefinite \mathbf{T} we can revert to using standard (symmetrical) Gaussian Elimination starting at the first row to obtain the *lower bidiagonal factorization*

$$\mathbf{T} = \mathbf{LDL}^*, \quad (2.11)$$

where $\mathbf{D} = \text{diag}(d_1, \dots, d_n)$ and

$$\mathbf{L} = \begin{pmatrix} 1 & & & & \\ \ell_1 & 1 & & & \\ & \ell_2 & 1 & & \\ & & \ddots & \ddots & \\ & & & \ell_{n-1} & 1 \end{pmatrix}$$

is *unit lower bidiagonal*. The d_i are called *pivots* and we sometimes refer to the ℓ_i as *nontrivial entries* of \mathbf{L} .

Just evaluating (2.11) gives

$$\begin{aligned} \mathbf{T}(1, 1) &= d_1, \\ \mathbf{T}(i, i) &= d_i + \ell_{i-1}^2 d_{i-1}, & \text{for } i = 2, \dots, n, \\ \mathbf{T}(i, i+1) &= \ell_i d_i, & \text{for } i = 1, \dots, n-1. \end{aligned} \quad (2.12)$$

With these at hand it is straightforward to write down a small loop to compute the decomposition.

CS 2.2: *Factorize* $T = LDL^*$

```

1:   $d_1 := T(1, 1)$ 
2:  for  $i = 1$  to  $n - 1$  do
3:     $\ell_i := T(i, i + 1)/d_i$ 
4:     $d_{i+1} := T(i + 1, i + 1) - \ell_i^2 d_i$ 
5:  endfor

```

Note that this is the only way to compute the data if T is unreduced, which is another way of saying that the decomposition must be unique, if it exists. The division by d_i is problematic because it means the computation will fail if a pivot other than the last becomes zero—this is fittingly called *breakdown*. Lemma 2.15 below will ascertain that a zero pivot is equivalent to a leading submatrix being singular. Its proof hinges on the following elementary observation. Write (2.11) in blocked form to see

$$T_{1:k} = (LDL^*)_{1:k} = L_{1:k} D_{1:k} L_{1:k}^*, \quad k = 1, \dots, n. \quad (2.13)$$

In other words, stopping the process prematurely at an index k still gives a valid lower bidiagonal factorization of the leading $k \times k$ submatrix of T .

Lemma 2.15. *Let symmetric tridiagonal $T \in \mathbb{R}^{n \times n}$ admit lower bidiagonal factorization $T = LDL^*$. Then*

$$d_k = \frac{\det T_{1:k}}{\det T_{1:k-1}}, \quad k = 2, \dots, n.$$

Consequently, the factorization exists in the first place if, and only if, all strictly leading principal submatrices $T_{1:k}, k < n$ are nonsingular.

Proof. Use (2.13) for $\det T_{1:k} = \det L_{1:k} \det D_{1:k} \det L_{1:k}^*$. The matrix $L_{1:k}$ is unit lower bidiagonal and thus has determinant one, so $\det T_{1:k} = \det D_{1:k} = \prod_{i=1}^k d_i$. \square

Bottom-to-Top

An alternative to $T = LDL^*$ is to just start the process in the last row of the matrix and proceed upwards, resulting in an *upper bidiagonal factorization*

$$T = URU^*$$

with $R = \text{diag}(r_1, \dots, r_n)$ and

$$U = \begin{pmatrix} 1 & u_2 & & & \\ & 1 & u_3 & & \\ & & 1 & \ddots & \\ & & & \ddots & u_n \\ & & & & 1 \end{pmatrix}$$

being *unit upper bidiagonal*.

Analogously to (2.12) we obtain

$$\begin{aligned} T(n, n) &= r_n, \\ T(i, i) &= r_i + u_{i+1}^2 r_{i+1}, & \text{for } i = 1, \dots, n-1, \\ T(i-1, i) &= u_i r_i, & \text{for } i = 2, \dots, n, \end{aligned} \quad (2.14)$$

together with a straightforward loop to compute the decomposition.

CS 2.3: *Factorize* $T = URU^*$

```

1:   $r_n := T(n, n)$ 
2:  for  $i = n$  down to 2 do
3:     $u_i := T(i-1, i)/r_i$ 
4:     $r_{i-1} := T(i-1, i-1) - u_i^2 r_i$ 
5:  endfor

```

Note that we index the nontrivial entries of U starting with two, although the standard way employed by other authors is to start with one. We feel our scheme beneficial mainly because it better exhibits the similarity (or symmetry) between lower and upper factorizations; in fact these are already apparent when comparing (2.12) with (2.14) and CS 2.2 with CS 2.3. Indexing the u_i 's starting with one would lead to terms $u_i^2 r_{i+1}$ and $u_{i-1} r_i$ compared to $\ell_i^2 d_i$ and $\ell_i d_i$ instead.

The mathematical equivalent of starting in the lower right corner is *flipping* the matrix beforehand. For $A \in \mathbb{R}^{n \times n}$ define its *flipped* counterpart by inverting each (off-)diagonal, that is,

$$A^{\text{flip}}(i, j) := A(n+1-j, n+1-i), \quad i, j = 1, \dots, n. \quad (2.15)$$

Using the symmetric permutation matrix $P_{\text{rev}} := [e_n | \dots | e_1] \in \mathbb{R}^{n \times n}$, this is more elegantly expressed as

$$A^{\text{flip}} = P_{\text{rev}} A^* P_{\text{rev}}.$$

Back to tridiagonal T , an upper bidiagonal factorization $T = URU^*$ is now tantamount to

$$T^{\text{flip}} = (U^{\text{flip}})^* R^{\text{flip}} U^{\text{flip}}, \quad (2.16)$$

which is a *lower* bidiagonal factorization of \mathbb{T}^{flip} . At this point we would urge the reader to verify for himself that (2.14) and CS 2.3 are really just the flipped analogons of (2.12) and CS 2.2. As an application, the flipping-connection applied to Lemma 2.15 nets us the criterion for existence of an upper bidiagonal factorization.

Corollary 2.16. *Let symmetric tridiagonal $\mathbb{T} \in \mathbb{R}^{n \times n}$ admit upper bidiagonal factorization $\mathbb{T} = \text{URU}^*$. Then*

$$r_k = \frac{\det \mathbb{T}_{k:n}}{\det \mathbb{T}_{k+1:n}}, \quad k = 1, \dots, n-1.$$

Consequently, the factorization exists in the first place if, and only if, all strictly trailing principal submatrices $\mathbb{T}_{k:n}, k > 1$ are nonsingular.

Simplifications

Bidiagonal factorizations and in particular the data elements that define them are ubiquitous in this thesis. To simplify the author's life we will now introduce two purely notational simplifications that are to become effective immediately.

First, the relations in (2.12) and (2.14) could have been stated more compactly less attention had to be spent on getting the indices right and watch for special cases. The following makes this possible:

(IMPLICIT 0)
Indexed quantities are implicitly defined as zero whenever they are used with an index that is "out-of-range".

However, reliance on implicitly defining otherwise undefined things does have a drawback in that it thrusts the doors wide open for ambiguity. As a matter of principle, we will avoid using it for writing down algorithms¹.

Second, things like $\ell_i^2 d_i$ quickly become cumbersome to write and confusing to read. Thus we define the handy abbreviations

$$\begin{aligned} \ell d_i &:= \ell_i^2 d_i, & \ell d_i &:= \ell_i d_i, \\ u u r_i &:= u_i^2 r_i, & u r_i &:= u_i r_i. \end{aligned} \tag{2.17}$$

Basically we just omit writing indices multiple times. Note that this would not be so easy to do if we had indexed the u 's starting with one. Naturally the declaration of implicit-zero will apply to those quantities as well.

For example, using both simplifications we can now write (2.12) and (2.14) as

$$\begin{aligned} (\text{LDL}^*)(i, i) &\equiv d_i + \ell d_{i-1}, & (\text{LDL}^*)(i, i+1) &\equiv \ell d_i, \\ (\text{URU}^*)(i, i) &\equiv r_i + u u r_{i+1}, & (\text{URU}^*)(i-1, i) &\equiv u r_i. \end{aligned} \tag{2.18}$$

¹ But one really should use IMPLICIT in FORTRAN, immediately followed by NONE.

Remark 2.17 (About ℓd_i and $u u r_i$). The connection between leading blocks of the matrix and blocks of the bidiagonal factors was expressed in (2.13). One key property of the quantities ℓd_i and $u u r_i$ is that they allow to do the same for trailing blocks, since

$$\begin{aligned} (\text{LDL}^*)_{k:n} - \ell d_{k-1} \mathbf{e}_1 \mathbf{e}_1^* &= \mathbf{L}_{k:n} \mathbf{D}_{k:n} \mathbf{L}_{k:n}^*, \\ (\text{URU}^*)_{1:k} - u u r_{k+1} \mathbf{e}_k \mathbf{e}_k^* &= \mathbf{U}_{1:k} \mathbf{R}_{1:k} \mathbf{U}_{1:k}^*. \end{aligned}$$

◇

Twisted Factorizations

A generalization of lower and upper bidiagonal factorizations are the so-called *twisted* factorizations. Sometimes also called *BABE-factorizations* (for “**B**urn **A**t **B**oth **E**nds”), they were resurrected by K.V. Fernando [26, 27] for the purpose of computing eigenvectors of symmetric tridiagonal matrices. We start by giving a principal motivation before delving into details.

The idea is to initiate both a top-to-bottom and a bottom-to-top factorization process at once until they meet at, say, the k th row, where they will have to be sewed together. The index k is the *twist index* or *twist position*. This results in a decomposition with both lower bidiagonal and upper bidiagonal subsystems.

What is the benefit of doing so? A near-singular matrix factorized as $\mathbf{T} = \text{LDL}^*$ may, but need not, have a tiny last pivot d_n . One, if not the essential observation is that there is always at least one twist position such that the twist element of the associated twisted factorization does reveal the near-singularity of the matrix. A cornerstone of the MR³-algorithm is that such a twist position is connected to a large eigenvector component and thus provides an excellent starting vector for inverse iteration.

Now come the details. Assume \mathbf{T} admits both a top-to-bottom and a bottom-to-top factorization $\mathbf{T} = \text{LDL}^* = \text{URU}^*$. With the insight given by (2.13) we can then construct for each twist index $k = 1, \dots, n$ a *twisted factorization* of \mathbf{T} as

$$\mathbf{T} = \mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*, \quad (2.19)$$

where $(\mathbf{N}_k)_{1:k} = \mathbf{L}_{1:k}$ is unit lower bidiagonal, $(\mathbf{N}_k)_{k:n} = \mathbf{U}_{k:n}$ is unit upper bidiagonal.

There is a delightfully clear characterization of the twist elements, linking them to the diagonal entries of the inverse. Applications of this result extend far beyond computing eigenvectors; see for example [62] for references to the solution of differential equations.

Lemma 2.18. *Let $T = N_k G_k N_k^*$. Then*

$$\gamma_k = \begin{cases} 0 & \text{if } T \text{ is singular,} \\ (e_k^* T^{-1} e_k)^{-1} & \text{otherwise.} \end{cases}$$

Proof. We have $\det T = \det G$. By Lemma 2.15 and Corollary 2.16, the twisted factorization can only exist if none of the $d_i, i < k$, or the $r_j, j > k$, is zero. Hence, T is singular if, and only if, $\gamma_k = 0$.

Now assume T is nonsingular. We have $N_k e_k = e_k$ and therefore $N_k^{-1} e_k = e_k$. This leads to

$$e_k^* T^{-1} e_k = e_k^* N_k^{-*} G_k^{-1} N_k^{-1} e_k = e_k^* G_k^{-1} e_k = \gamma_k^{-1}.$$

□

One particular consequence is that for singular T all twist elements will be zero, that is, $\gamma_k = 0$ for all k .

A remark concerning terminology. We have noted above that a top-to-bottom factorization can be regarded as a twisted one since $LDL^* = N_n G_n N_n^*$, with the last pivot of the former being the twist element of the latter, $d_n = \gamma_n$. The choice of interpretation may vary with context, but will be consistent, that is, a specific decomposition will either be regarded as LDL^* with last pivot d_n , or as $N_n G_n N_n^*$ with twist element γ_n . The same holds for the relation between URU^* and $N_1 G_1 N_1^*$.

In any case we use the terms *bidiagonal* and *twisted* sometimes interchangeably. After all, the twisted factors N_k are partially bidiagonal.

About Inverses

One feature of bidiagonal matrices is that one can write down explicit formulas for the entries of the inverse. For given unit lower bidiagonal L the solutions to $Lx = e_j, 1 \leq j \leq n$ are obtained through forward substitution

$$\begin{aligned} x(i) &= 0, & i &= 1, \dots, j-1, \\ x(j) &= 1, & & \\ x(i) &= -\ell_{i-1} x(i-1), & i &= j+1, \dots, n. \end{aligned} \tag{2.21}$$

Thus the offdiagonal entries of the inverse of L , which is dense unit lower triangular, are given by

$$(L^{-1})(i, j) = (-1)^{i-j} \prod_{k=j}^{i-1} \ell_k, \quad i > j. \quad (2.22)$$

More revealing are the recurrences

$$\begin{aligned} \mathbf{e}_1^* L^{-1} &= \mathbf{e}_1^*, \\ \mathbf{e}_i^* L^{-1} &= -\ell_{i-1} \mathbf{e}_{i-1}^* L^{-1} + \mathbf{e}_i^*, \quad i = 2, \dots, n, \end{aligned} \quad (2.23)$$

for the rows and

$$\begin{aligned} L^{-1} \mathbf{e}_n &= \mathbf{e}_n, \\ L^{-1} \mathbf{e}_i &= -\ell_i L^{-1} \mathbf{e}_{i+1} + \mathbf{e}_i, \quad i = n-1, n-2, \dots, 1, \end{aligned} \quad (2.24)$$

for the columns. One application of these is presented by Dhillon in [16] for the stable computation of condition numbers of tridiagonal matrices.

As an example, for $n = 4$ we have

$$L = \begin{pmatrix} 1 & & & \\ \ell_1 & 1 & & \\ & \ell_2 & 1 & \\ & & \ell_3 & 1 \end{pmatrix}, \quad L^{-1} = \begin{pmatrix} 1 & & & \\ -\ell_1 & 1 & & \\ \ell_1 \ell_2 & -\ell_2 & 1 & \\ -\ell_1 \ell_2 \ell_3 & \ell_2 \ell_3 & -\ell_3 & 1 \end{pmatrix}.$$

The preceding structural properties transcend to twisted factors N with twist index k , since

$$(N^{-1})_{1:k} = (N_{1:k})^{-1}.$$

Similarly, for a factored tridiagonal matrix $T = NGN^*$ we can use

$$T^{-1} = N^{-*} G^{-1} N^{-1}, \quad (2.25)$$

to make the above results applicable. Thus it becomes apparent that a representation of a nonsingular symmetric tridiagonal matrix using the nontrivial data entries of a twisted factorization can also be regarded as representing the inverse, using only multiplications for the representational mapping.

2.3.2 Eigenvectors with a Twist

In the following we will investigate how twisted factorizations can be harnessed to compute eigenvectors with residual bounds that are good enough so satisfy requirement GETVEC of MR³.

To better synchronize the exposition with GETVEC and step 8 of MR³, we consider a node in the tree with representation M and a singleton eigenpair (λ, \mathbf{q}) ,

$\|\mathbf{q}\| = 1$. The task is to compute an approximate eigenpair $(\bar{\lambda}, \mathbf{z})$. Objects of particular interest will be twisted factorizations $\mathbf{M} - \bar{\lambda} = \mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$ for some twist index k .

A structural feature of the twisted factor \mathbf{N}_k is that the k th column is just the k th unit vector. This property is retained for the inverse of the factor, as $\mathbf{N}_k \mathbf{e}_k = \mathbf{e}_k \Rightarrow \mathbf{e}_k = \mathbf{N}_k^{-1} \mathbf{e}_k$. Thus, solving the system

$$\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^* \mathbf{z} = \gamma_k \mathbf{e}_k, \quad \mathbf{z}(k) = 1, \quad (2.26)$$

is equivalent to solving $\mathbf{N}_k^* \mathbf{z} = \mathbf{e}_k$ (keep in mind that γ_k might be zero). The latter is achieved through the following recurrence, which is really just (2.21) in action.

CS 2.4: *Solve for eigenvector*

```

1:    $\mathbf{z}(k) := 1$ 
2:   for  $i = k - 1$  down to 1 do
3:        $\mathbf{z}(i) := -\ell_i \mathbf{z}(i + 1)$ 
4:   endfor
5:   for  $i = k + 1$  to  $n$  do
6:        $\mathbf{z}(i) := -u_i \mathbf{z}(i - 1)$ 
7:   endfor

```

The residual norm with respect to \mathbf{M} and $\bar{\lambda}$ is $|\gamma_k|/\|\mathbf{z}\|$; this we must be able to control for GETVEC. Note that we have not yet put any constraint on the choice of k . An obvious approach at this point is to compute $\mathbf{M} - \bar{\lambda} = \mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$ for all k and choose the one that minimizes $|\gamma_k|$.

The twist elements, in particular the smallest ones, give information about how far the matrix $\mathbf{M} - \bar{\lambda}$ is away from singularity, i.e., how close $\bar{\lambda}$ is to λ . Each twist element denotes an additive perturbation to just one entry that makes the matrix singular, since replacing γ_k by zero yields a twisted factorization of $\mathbf{M} - \bar{\lambda} - \gamma_k \mathbf{e}_k \mathbf{e}_k^*$ with eigenpair $(0, \mathbf{z})$. A normwise smaller perturbation to get a singular matrix is given by the residual, because $(0, \mathbf{z})$ is also an eigenpair of

$$\mathbf{M} - \bar{\lambda} - \frac{\gamma_k}{\|\mathbf{z}\|^2} \mathbf{e}_k \mathbf{z}^*.$$

Indeed it can be shown [46] that this is the normwise smallest change to $\mathbf{M} - \bar{\lambda}$ which yields in a singular matrix.

Assuming $\bar{\lambda}$ is closer to λ than to any other eigenvalue, Theorem 1.20 shows

$$|\bar{\lambda} - \lambda| \leq |\gamma_k|/\|\mathbf{z}\| \leq |\gamma_k|$$

for all k . More important for the purpose at hand than the connection between the twist elements and eigenvalues is that the magnitudes of the twist elements are related to the magnitudes of the eigenvector entries. Recall that \mathbf{q} is the

eigenvector of M belonging to λ . One can show [15, Lemma 3.2.1] that in the limit case $\bar{\lambda} \rightarrow \lambda$, the twist elements $\gamma_k = \gamma_k(\bar{\lambda})$ of $M - \bar{\lambda} = N_k G_k N_k^*$ behave as

$$\frac{\gamma_k^{-1}}{\sum_{i=1}^n \gamma_i^{-1}} \longrightarrow \mathbf{q}(k)^2, \quad k = 1, \dots, n.$$

Hence, for a sufficiently accurate eigenvalue approximation $\bar{\lambda}$, the twist index minimizing $|\gamma_k|$ will be one where the eigenvector has an above average entry. Therefore, solving (2.26) with this k is one step of inverse iteration with a right hand side that is *guaranteed* to be good.

The next result sheds light on these issues, but without even mentioning twisted factorizations. It combines Theorem 11 and Lemma 12 from [18] to cover the general characteristics of solving symmetric systems with a coordinate vector on the right hand side. For more results of the same flavor see [27].

Theorem 2.19

Let $A \in \mathbb{R}^{n \times n}$ be symmetric, k be an index with $1 \leq k \leq n$, $\mathbf{z} \in \mathbb{R}^n$ be a vector with $\mathbf{z}(k) = 1$ and γ, τ be scalars such that

$$(A - \tau)\mathbf{z} = \gamma \mathbf{e}_k,$$

with the added condition that $\gamma = 0$ if $A - \tau$ is singular. Write $\lambda_i = \lambda_i[A]$, $\mathbf{q}_i = \mathbf{q}_i[A]$. Then the following holds:

- (i) For every j with $\mathbf{q}_j(k) \neq 0$, the residual norm of the approximation (τ, \mathbf{z}) to the j 'th eigenpair of A is bounded by

$$\frac{\|(A - \tau)\mathbf{z}\|}{\|\mathbf{z}\|} = \frac{|\gamma|}{\|\mathbf{z}\|} \leq \frac{|\lambda_j - \tau|}{|\mathbf{q}_j(k)|}.$$

- (ii) The Rayleigh Quotient of \mathbf{z} with respect to the matrix A is

$$\rho = \frac{\mathbf{z}^* A \mathbf{z}}{\mathbf{z}^* \mathbf{z}} = \frac{\gamma}{\|\mathbf{z}\|^2} + \tau,$$

with associated residual

$$\frac{\|(A - \rho)\mathbf{z}\|}{\|\mathbf{z}\|} = \frac{|\gamma|}{\|\mathbf{z}\|} (1 - \|\mathbf{z}\|^{-2})^{1/2}.$$

Proof. (i). If $\gamma = 0$, there is nothing to prove, and because of the addendum that $\gamma = 0$ if A is singular neither is there anything to prove in that case, so assume $\gamma \neq 0$ and $A - \tau$ being nonsingular, i.e., $\lambda_j \neq \tau$ for all j .

Then $\mathbf{z} = \gamma(\mathbf{A} - \tau)^{-1}\mathbf{e}_k$ and so,

$$\begin{aligned} \|\mathbf{z}\|^2/|\gamma|^2 &= \mathbf{e}_k^*(\mathbf{A} - \tau)^{-1}(\mathbf{A} - \tau)^{-1}\mathbf{e}_k \\ &= \mathbf{e}_k^* \left(\sum_{i=1}^n (\lambda_i - \tau)^{-2} \mathbf{q}_i \mathbf{q}_i^* \right) \mathbf{e}_k \\ &= \sum_{i=1}^n \left(\frac{\mathbf{q}_i(k)}{\lambda_i - \tau} \right)^2 \geq \left(\frac{\mathbf{q}_j(k)}{\lambda_j - \tau} \right)^2 \quad \text{for all } j. \end{aligned}$$

(ii). Write $\mathbf{A} = (\mathbf{A} - \tau) + \tau$. Using $\mathbf{z}(k) = 1$, the identity for ρ is immediate. Then

$$(\mathbf{A} - \rho)\mathbf{z} = (\mathbf{A} - \tau)\mathbf{z} - \frac{\gamma}{\|\mathbf{z}\|^2}\mathbf{z} = \gamma \left(\mathbf{e}_k - \frac{\mathbf{z}}{\|\mathbf{z}\|^2} \right).$$

Again employing $\mathbf{z}(k) = 1$, we obtain

$$\|(\mathbf{A} - \rho)\mathbf{z}\|^2 = \gamma^2 \left(1 - \frac{2}{\|\mathbf{z}\|^2} + \frac{1}{\|\mathbf{z}\|^2} \right) = \gamma^2(1 - \|\mathbf{z}\|^{-2}).$$

□

Suppose we have all twisted factorizations $\mathbf{M} - \bar{\lambda} = \mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$, $k = 1, \dots, n$ together with the solutions \mathbf{z}_k of (2.26). The eigenvector \mathbf{q}_j belonging to $\lambda = \lambda_j[\mathbf{M}]$ has unit norm and therefore must have at least one entry exceeding $1/\sqrt{n}$ in magnitude. Then the first part of the theorem shows that there is at least one twist index k with

$$\|(\mathbf{M} - \bar{\lambda})\mathbf{z}_k\|/\|\mathbf{z}_k\| \leq \sqrt{n}|\bar{\lambda} - \lambda|,$$

revealing the near-singularity of $\mathbf{M} - \bar{\lambda}$. Furthermore it becomes clear that in order to obtain the accuracy we need (for GETVEC), in general $\bar{\lambda}$ will have to approximate λ to high relative accuracy. Now the second part of the theorem comes in, as it allows using the *Rayleigh Quotient Correction* $\gamma_k/\|\mathbf{z}\|^2$ to improve $\bar{\lambda}$.

Altogether this culminates in the customized Rayleigh Quotient Iteration (RQI) for twisted factorizations shown as Algorithm 2.5. For brevity we have omitted some details like how to deal with non-convergence. In any case, something similar to it should be used as subroutine for step 8 in MR³. Note that we use twisted factorizations only for computing the vectors, but the kind of representation used for \mathbf{M} is not specified, except that it must provide the operation to compute twisted factorizations of $\mathbf{M} - \tau$.

Issues in Exact Arithmetic

Theorem 2.19 provides solid motivation for the claim that choosing the twist index to minimize $|\gamma_k|$ should result in an excellent eigenvector. However, there are two conceivable blemishes to refute.

ALGORITHM 2.5 *RQI with twisted factorizations***Input:** M , initial estimate $\bar{\lambda}$ for an eigenvalue**Output:** Improved estimate $\bar{\lambda}$ and approximate eigenvector z

```

1:  loop
2:      Compute  $M - \bar{\lambda} = (N_t G_t N_t^*)$  for  $t = 1, \dots, n$ 
3:      Choose  $k$  that minimizes  $|\gamma_k|$ 
4:      Use CS 2.4 to solve  $N_k^+ G_k^+ (N_k^+)^* z = \gamma_k e_k$ 
5:      if  $|\gamma_k|/\|z\|$  is small enough then
6:          stop
7:      else
8:           $\bar{\lambda} := \bar{\lambda} + \gamma_k/\|z\|^2$ 
9:      endif
10: end loop

```

We implicitly assumed that the twisted factorizations do exist for all k . In principle every one of them might break down due to zero pivots. The problem is related to zero entries in the eigenvectors, cf. Remark 2.1. Indeed, look back at CS 2.4 to confirm that it can never produce a zero entry (in exact arithmetic, that is, barring underflow). There is an extension of the method that effectively solves the problem [15, 18].

Of more concern is that the bound for the residual norm given by Theorem 2.19 might not be good enough. After all, we can in general not expect to be able to approximate λ better than $|\bar{\lambda} - \lambda| = \mathcal{O}(|\lambda|n\epsilon_\diamond)$. Hence, even if the RQI is converged and $\bar{\lambda}$ is as accurate as we can possibly get it, the worst-case bound for the residual norm might still be $\mathcal{O}(n^{3/2}|\lambda|\epsilon_\diamond)$, exceeding the allowances of GETVEC. The main argument that can be fielded against this is practical experience showing that the residuals are usually far better than these worst-case bounds would indicate. Also keep in mind that GETVEC only requires an $\mathcal{O}(|\lambda|n\epsilon_\diamond)$ residual norm if the relative separation is close to the gap tolerance. See [18, p.890f] for an in-depth discussion about the quality of these bounds in theory and their ramifications in practice.

Issues in Floating-Point Arithmetic

Recall that GETVEC allowed both perturbing the matrix and the computed vector to attain the desired residual norm. So far we have not mentioned this and we want to leave the topic of computing eigenvectors with a note on how these perturbations arise and why they are necessary.

We expect that the algorithm used to compute a twisted factorization $\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$ of $\mathbf{M} - \tau$ is stable in the sense that

$$\tilde{\mathbf{M}}_k - \bar{\lambda} = \tilde{\mathbf{N}}_k \tilde{\mathbf{G}}_k \tilde{\mathbf{N}}_k^*$$

holds exactly for suitable elementwise relative perturbations to \mathbf{M} and $\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$.

Note the k in $\tilde{\mathbf{M}}_k$. Its purpose is to pronounce that the perturbation at the source \mathbf{M} will in general be different for each k . In theory this means we lose the statement that at least one of the twist indices leads to a residual norm not larger than $\sqrt{n}|\bar{\lambda} - \lambda|$, but the practical consequences of this are minute (cf. what we said above for exact arithmetic).

Assume the erp $\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^* \rightsquigarrow \tilde{\mathbf{N}}_k \tilde{\mathbf{G}}_k \tilde{\mathbf{N}}_k^*$ can be described by $\tilde{\ell}_i \doteq \ell_i(1 + \epsilon(m))$, $\tilde{u}_j \doteq u_j(1 + \epsilon(m))$. Let CS 2.4 be executed under the wings of Axiom FP, without underflow or overflow, yielding a vector $\bar{\mathbf{z}}$. The crucial observation is that the entries of $\bar{\mathbf{z}}$ were computed using only multiplications. Then it is not hard to see that we can account for the perturbation of $\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$ and the rounding errors during the computation by putting some small relative changes to the vector's components. This leads to

$$\bar{\mathbf{z}} \rightsquigarrow \tilde{\mathbf{z}}, \quad \tilde{\mathbf{z}}(i) \doteq \bar{\mathbf{z}}(i) \left(1 + \epsilon((m+1)|i-k| - 1) \right), \quad i \neq k,$$

such that $\tilde{\mathbf{z}}$ is the exact solution of the perturbed system, i.e.,

$$(\tilde{\mathbf{M}} - \bar{\lambda})\tilde{\mathbf{z}} = \tilde{\mathbf{N}}_k \tilde{\mathbf{G}}_k \tilde{\mathbf{N}}_k^* \tilde{\mathbf{z}} = \tilde{\gamma}_k \mathbf{e}_k.$$

Thus, in the context of GETVEC, we can take $\beta_{\ddagger} = \epsilon(mn)$ to cover all possible choices of k , cf. the corresponding entry in Table 2.1. Finally, note that the residual norm $|\gamma_k|/\|\bar{\mathbf{z}}\|$ and the Rayleigh Quotient Correction $\gamma_k/\|\bar{\mathbf{z}}\|^2$ are effectively computed by Algorithm 2.5 to high relative accuracy with respect to the perturbed system above. Hence, errors due to finite-precision arithmetic will not have an adverse effect on the iteration itself. In fact, this is another advantage of using twisted factorizations.

2.3.3 Relative Condition Numbers

Since the development of MR³, much work has been invested into the relative perturbation theory of (partial) eigensystems of tridiagonal matrices in factored form. The most general approach looks at how elementwise relative perturbations

can effect the *hyperbolic* SVD [7] of a bidiagonal matrix, see [59, 60] and the formidable [61]. A simpler strategy can be based on first- (and second-) order analysis of additive perturbations and this is what we will stick to. The loss is only theoretical, because the resulting bounds and condition numbers are essentially identical whichever way you go.

We will harness techniques from parts of [15, §5.2.1] and [59, 60, 63], but our presentation differs because our motivation does: we want to apply the results to different kinds of representations (like the block factorizations from Chapter 4), so we will do as much as possible in the setting of additive perturbations of arbitrary symmetric matrices before zooming in on the factored form.

Per-eigenpair analysis of additive perturbations

We already took a look at additive perturbations

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{E}$$

for symmetric \mathbf{E} in §1.4.3. There the focus was on uniform bounds based on $\|\mathbf{E}\|$, which led to Corollary 1.28 for the eigenvalues and Theorem 1.29 for the eigenvectors.

For MR³ we need to achieve more for less. Less because only the local invariant subspace has to be relatively robust; in principle we do not care what happens with the rest. More because the corresponding eigenvalues will be small or even tiny compared to the norm of the matrix. Added to that, for decomposition-based representations of nearly singular matrices some amount of element growth might be unavoidable, so situations where $\|\mathbf{E}\| \gg \epsilon_0 \|\mathbf{A}\|$ have to be anticipated. Under these circumstances, a bound using solely $\|\mathbf{E}\|$ gives us nothing. The following analysis is finer-grained and will provide more insight, in particular concerning the effects on parts of the spectrum.

Use $\lambda_i = \lambda_i[\mathbf{A}]$, $\mathbf{q}_i = \mathbf{q}_i[\mathbf{A}]$, $\tilde{\lambda}_i = \lambda_i[\tilde{\mathbf{A}}]$, $\tilde{\mathbf{q}}_i = \mathbf{q}_i[\tilde{\mathbf{A}}]$ for $i = 1, \dots, n$ to denote the eigensystems of \mathbf{A} and $\tilde{\mathbf{A}}$, and

$$\tilde{\lambda}_j = \lambda_j + \delta\lambda_j, \quad \tilde{\mathbf{q}}_j = \sum_{i=1}^n \eta_{ji} \mathbf{q}_i,$$

to relate them. The factors η_{ji} measure the contribution of \mathbf{q}_i to the perturbed $\tilde{\mathbf{q}}_j$. In particular we have $\eta_{jj} = \cos\angle(\mathbf{q}_j, \tilde{\mathbf{q}}_j)$ and $\sum_{i \neq j} \eta_{ji}^2 = \sin^2\angle(\mathbf{q}_j, \tilde{\mathbf{q}}_j)$. To see how the k th eigenpair of \mathbf{A} is affected write down the standard eigenequation

$$(\mathbf{A} + \mathbf{E}) \sum_{i=1}^n \eta_{ki} \mathbf{q}_i = \tilde{\mathbf{A}} \tilde{\mathbf{q}}_k = (\lambda_k + \delta\lambda_k) \sum_{i=1}^n \eta_{ki} \mathbf{q}_i.$$

Multiply this from the left by \mathbf{q}_r^* for any r to find

$$\lambda_r \eta_{kr} + \mathbf{q}_r^* \mathbf{E} \tilde{\mathbf{q}}_k = \tilde{\lambda}_k \eta_{kr}.$$

Rearranged this gives the nicely symmetric formula

$$\boxed{\eta_{kr}(\tilde{\lambda}_k - \lambda_r) = \mathbf{q}_r^* \mathbf{E} \tilde{\mathbf{q}}_k} \quad (2.27)$$

for all k, r , which becomes

$$\boxed{\eta_{kk} \delta \lambda_k = \mathbf{q}_k^* \mathbf{E} \tilde{\mathbf{q}}_k} \quad (2.28)$$

for the special case $k = r$. Note that we did not assume anything so far. These formulas are valid and sharp even for a multiple eigenvalue λ_k , and $\eta_{kk} = 0$ is possible, i.e., \mathbf{q}_k might be orthogonal to its perturbed counterpart $\tilde{\mathbf{q}}_k$.

As appealing as the symmetry in (2.27) is, the formula is nevertheless unsatisfactory because the perturbed quantities $\tilde{\lambda}_k$ and $\tilde{\mathbf{q}}_k$ are still in there. The usual approach to get rid of them is to resort to first-order analysis. A formal analytical setting can be found in chapter 2 of Wilkinson's book [71], we will constrain ourselves to the main steps.

Assume λ_k is simple and $\eta_{kk} \neq 0$. Then we can write

$$\tilde{\mathbf{q}}_k = \mathbf{q}_k + \delta \mathbf{q}_k, \quad \delta \mathbf{q}_k = \sum_{i \neq k} \omega_{ki} \mathbf{q}_i,$$

where $\sum_{i \neq k} \omega_{ki}^2 = \tan^2 \angle(\mathbf{q}_k, \tilde{\mathbf{q}}_k)$, $\omega_{ki} = \eta_{ki} / \eta_{kk}$.

The gist of the approach is to write $\mathbf{E} = \epsilon \mathbf{F}$ and investigate $\tilde{\lambda}_k = \lambda_k(\epsilon)$ and $\tilde{\mathbf{q}}_k = \mathbf{q}_k(\epsilon)$ as $\epsilon \rightarrow 0$. Using that λ_k is simple, one can expand both $\lambda_k(\epsilon)$ and $\mathbf{q}_k(\epsilon)$ as convergent power series in ϵ to conclude $|\lambda_k(\epsilon) - \lambda_k| = \mathcal{O}(\epsilon)$ and $\|\mathbf{q}_k(\epsilon) - \mathbf{q}_k\| = \mathcal{O}(\epsilon)$ for small enough ϵ . Plug these into (2.27) and (2.28) and single out all ϵ -terms to get the first-order expressions

$$\begin{aligned} \delta \lambda_k &\stackrel{1\text{st}}{=} \mathbf{q}_k^* \mathbf{E} \mathbf{q}_k, \\ \omega_{kr}(\lambda_k - \lambda_r) &\stackrel{1\text{st}}{=} \mathbf{q}_k^* \mathbf{E} \mathbf{q}_r, \end{aligned} \quad (2.29)$$

(cf. [71, p. 69-70]). These capture the principle for most applications. Just to be on the safe side one can also consider the second-order terms, leading to

$$\begin{aligned} \delta \lambda_k &\stackrel{2\text{nd}}{=} \mathbf{q}_k^* \mathbf{E} \mathbf{q}_k + \sum_{i \neq k} \frac{(\mathbf{q}_k^* \mathbf{E} \mathbf{q}_i)^2}{\lambda_k - \lambda_i}, \\ \omega_{kr}(\lambda_k - \lambda_r) &\stackrel{2\text{nd}}{=} \mathbf{q}_k^* \mathbf{E} \mathbf{q}_r + \sum_{i \neq k, r} \frac{(\mathbf{q}_k^* \mathbf{E} \mathbf{q}_i)(\mathbf{q}_r^* \mathbf{E} \mathbf{q}_i)}{\lambda_k - \lambda_i}. \end{aligned} \quad (2.30)$$

Interestingly, for our purposes the second-order terms are not that useless as they usually are. They convey that the first-order terms in (2.29) are perfectly sufficient for the k th eigenpair as long as $|\mathbf{q}_k^* \mathbf{E} \mathbf{q}_i| \ll |\lambda_k - \lambda_i|$ for all $i \neq k$. That is a much more powerful statement than to require $\|\mathbf{E}\| \ll \text{gap}_A(\lambda_k)$ as we had to

in §1.4.3, and it fits with requirement ELG for MR³ in reiterating that element growth is acceptable at entries where the eigenvector components are small.

However, for a tight cluster I of eigenvalues, we might well have $|q_k^* E q_i| \gtrsim |\lambda_k - \lambda_i|$ for $i \in I$, so the first-order expressions lose applicability. But then the second-order terms in (2.30) show *in principle* how the analysis has to be adjusted: the summands on the right become non-negligible for $i \in I$, so the right thing to do for gauging the robustness of the eigenvalues in the cluster is to evaluate $Q_I^* E Q_I$ in a suitable norm.

Application to Twisted Factorizations

Now let us see how the developed techniques can help our understanding of elementwise relative perturbations of twisted factorizations. Suppose we have NGN^* (right now we do not care about the twist index) as representation at a node in the tree. In order to fulfill requirements RRR and GETVEC, we need to be able to control the effect that any erp

$$T = NGN^* \rightsquigarrow \tilde{N}\tilde{G}\tilde{N}^* = \tilde{T}$$

can have on the local subset of eigenpairs.

The crucial observation is that elementwise relative perturbations of *bidiagonal* matrices can be cast in the form of outer multiplicative perturbations by diagonal matrices; this goes back to Demmel & Kahan [12] and was later expanded by Demmel & Gragg [11] to characterize the class of matrices whose associated bipartite graph is acyclic.

Lemma 2.20. *Let $L \in \mathbb{R}^{n \times n}$ be unit lower bidiagonal and perturbed to \tilde{L} according to $\tilde{L}(i+1, i) = L(i+1, i)(1 + \varepsilon_i)$ for $i = 1, \dots, n-1$, with $n\varepsilon_i \ll 1$. Then*

$$\tilde{L} = \begin{pmatrix} (1 + \alpha_1) & & & \\ & \ddots & & \\ & & \ddots & \\ & & & (1 + \alpha_n) \end{pmatrix} \cdot L \cdot \begin{pmatrix} (1 + \beta_1) & & & \\ & \ddots & & \\ & & \ddots & \\ & & & (1 + \beta_n) \end{pmatrix},$$

where

$$(1 + \alpha_i) = \prod_{k=1}^{i-1} (1 + \varepsilon_k), \quad (1 + \beta_i) = \prod_{k=1}^{i-1} (1 + \varepsilon_k)^{-1}, \quad i = 1, \dots, n.$$

Proof. Self-evident. □

Clearly this result extends to unit upper bidiagonal matrices, their flipped counterparts and therefore also to twisted factors. Thus an elementwise perturbation of a twisted factorization can be written as

$$\tilde{N}\tilde{G}\tilde{N}^* = Y \underbrace{NXGXN^*}_{\tilde{T}} Y$$

with diagonal matrices \mathbf{X} and \mathbf{Y} . From Theorems 1.30 and 1.31 we know the outer perturbations given by \mathbf{Y} to be of no concern. What has to be studied is the effect of the *inner* multiplicative perturbations \mathbf{X} .

At this point a slight change in notation will prove to be beneficial. Extract the signs from \mathbf{G} into a signature matrix $\Omega = \text{diag}(\pm 1)$ via $\mathbf{G} = \sqrt{|\mathbf{G}|}\Omega\sqrt{|\mathbf{G}|}$ to get $\mathbf{T} = \mathbf{K}\Omega\mathbf{K}^*$ with $\mathbf{K} = \mathbf{N}\sqrt{|\mathbf{G}|}$. As both \mathbf{X} and \mathbf{G} are diagonal, we also have $\tilde{\mathbf{T}} = \mathbf{K}\Omega\mathbf{X}^2\mathbf{K}^*$. Then $\mathbf{T} \rightsquigarrow \tilde{\mathbf{T}}$ can be written as additive perturbation

$$\tilde{\mathbf{T}} = \mathbf{T} + \mathbf{K}\Omega(\mathbf{X}^2 - \mathbf{I})\mathbf{K}^* = \mathbf{T} + \mathbf{E},$$

defining \mathbf{E} . With $\Delta := \|\mathbf{X}^2 - \mathbf{I}\|$, application of (2.29) yields

$$|\delta\lambda_k| \stackrel{\text{1st}}{=} |\mathbf{q}_k^*\mathbf{K}\Omega(\mathbf{X}^2 - \mathbf{I})\mathbf{K}^*\mathbf{q}_k| \leq \Delta\|\mathbf{K}^*\mathbf{q}_k\|^2. \quad (2.31)$$

Hence we arrive at the point where the proper *relative condition number* (relcond) of an eigenvalue λ_k emerges, namely

$$\kappa_{\text{rel}}(\lambda_k) := \frac{\|\mathbf{K}^*\mathbf{q}_k\|^2}{|\lambda_k|} = \frac{\mathbf{q}_k^*\mathbf{N}|\mathbf{G}|\mathbf{N}^*\mathbf{q}_k}{|\mathbf{q}_k^*\mathbf{N}\mathbf{G}\mathbf{N}^*\mathbf{q}_k|}. \quad (2.32)$$

This is how relative condition numbers were originally defined by Dhillon for LDL*-decompositions [15]. It is possible to get a corresponding condition number for the eigenvectors by summing up the ω_{kr} 's in (2.29) for $r \neq k$. This can in turn be expressed as sum over the relconds of the values, but we will omit a derivation here and refer the reader to [63, p. 149f]. Suffice it to say that, as end result, one can state that relative robustness of the eigenvalue implies the same for the eigenvector, provided the eigenvalue has a large enough relative separation from the rest of the spectrum.

Note that both numerator and denominator in (2.32) are Rayleigh Quotients, and $\kappa_{\text{rel}}(\lambda_k)$ is in fact nothing else but the condition number of the Rayleigh Quotient $\rho_{\mathbf{N}\mathbf{G}\mathbf{N}^*}(\mathbf{q}_k)$ under perturbations in \mathbf{G} .

Clearly $\kappa_{\text{rel}} \equiv 1$ for a semi-definite matrix ($|\mathbf{G}| = \pm\mathbf{G}$). So the above derivation can be regarded as first-order proof that a bidiagonal factorization of a semi-definite matrix will always be relatively robust for all eigenpairs. This restates the well-known result by Demmel & Kahan which we will present in Chapter 3.

2.4 QD-Algorithms

The missing link in our exposition so far remains how twisted factorizations of shifted symmetric tridiagonal matrices can be computed in a stable way. The highly customized tools that exist for this purpose are all descendants of Rutishauer's original *Quotienten-Differenzen* (quotient differences) algorithm [67]. This section is devoted to the derivation, analysis and refinement of these tools.

Our presentation of MR³ revolved around the principle that the kind of representation used for the matrices at each node can be chosen freely, as long as two operations can be provided:

- (1) One must be able to shift, that is, compute

$$\mathbf{M} - \tau = \mathbf{M}_+,$$

such that requirement SHIFTRREL is achieved.

- (2) In §2.3.2 we saw how to compute eigenvectors with residual norms good enough for GETVEC, if a twisted factorization of a shifted representation

$$\mathbf{M} - \tau = \mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$$

can be computed with mixed relative stability.

If twisted factorizations are itself to be used as representations at the nodes, both operations collapse to computing

$$\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^* - \tau = \mathbf{N}_t^+ \mathbf{G}_t^+ (\mathbf{N}_t^+)^*, \quad k, t \text{ arbitrary.} \quad (2.33)$$

Note that this would explicitly allow for varying the twist index between nodes.

In its original form, MR³ was presented using plain LDL* factorizations at each node. To facilitate their use, Dhillon developed in his thesis the following transformations [15, Alg. 4.4.{3, 4, 6}]:

$$\begin{array}{ll} \text{dstqds} & \text{to compute } \mathbf{LDL}^* - \tau = \mathbf{L}^+ \mathbf{D}^+ (\mathbf{L}^+)^*, \quad (\text{stationary}) \\ \text{dqds} & \text{to compute } \mathbf{LDL}^* - \tau = \mathbf{U}^+ \mathbf{R}^+ (\mathbf{U}^+)^*, \quad (\text{progressive}) \\ \text{dtwqds} & \text{to compute } \mathbf{LDL}^* - \tau = \mathbf{N}_k^+ \mathbf{G}_k^+ (\mathbf{N}_k^+)^*. \end{array}$$

The names stand for *differential (stationary / twisted) quotient-differences with shift*. Note that dqds is not the same as the dqds-algorithm for computing singular values, although both are intimately related, since the latter uses a tailored version of the former as subroutine; for more information see Remark 2.21 below.

The ultimate goal of the following pages is to reach an algorithm for the general task (2.33), which we will still call dtwqds, even if it generalizes Dhillon's version. Although the only previous treatment of (2.33) that we know of is in [35, Appendix A] for the restricted case $t \in \{1, n\}$, we do *not* claim the algorithm to be new in any way, because it boils down to an elementary combination of stationary and progressive parts.

Our contribution lies along the way in the investigation of two alternative approaches to represent matrices given by twisted factorizations. The traditional way of using the nontrivial entries of the factors \mathbf{N} and \mathbf{G} directly is shown to be far from optimal. Furthermore we will introduce a new technique which we call *perturbing the shift*. It allows to simplify the analysis of the shifting process while providing more robust error bounds at the same time.

2.4.1 Different flavors of Representations

A general symmetric tridiagonal matrix $\mathbf{T} \in \mathbb{R}^{n \times n}$ has exactly $2n - 1$ degrees of freedom. The concept of a representation from Definition 2.4 allows for flexibility in how these can be determined. For a twisted factorization $\mathbf{T} = \mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$, we have already come to know four important types of associated scalars in §2.3, namely:

- (1) d_i, γ_k, r_j : The pivots. They give the determinants of submatrices, provide Sturm counts and contribute to the diagonal entries as in (2.18).
- (2) ℓ_i, u_j : The nontrivial entries of the twisted factor \mathbf{N}_k . In §2.3.2 we saw how products of them give the solution of $\mathbf{T}\mathbf{x} = \gamma_k \mathbf{e}_k$, leading to the entries of computed eigenvectors. Furthermore, Lemma 2.20 established that an erp to \mathbf{N} can be cast directly into outer perturbations.
- (3) ld_i, ur_j : The offdiagonal entries e_i of the matrix. In exact arithmetic they would remain unchanged by shifting.
- (4) lld_i, uur_j : Constitute the diagonal entries of the matrix together with the pivots. Remark 2.17 revealed their connection to Schur complements.

Let us focus on a factored matrix $\mathbf{T} = \mathbf{LDL}^*$ for a moment. A natural alternative to representing \mathbf{T} by its entries is combining (1) and (2), i.e., using the nontrivial entries d_1, \dots, d_n and $\ell_1, \dots, \ell_{n-1}$ of \mathbf{D} and \mathbf{L} instead. However, in principle any two of the sets $\{d_i | i < n\}$, $\{\ell_i | i < n\}$, $\{ld_i | i < n\}$, $\{lld_i | i < n\}$ combined with d_n would also define the matrix uniquely, since we know $d_i \neq 0, i < n$ and assume the offdiagonals to be positive. As a truly exotic example, if we merely had the ld_i 's, lld_i 's and d_n , the remaining quantities could be derived as

$$\ell_i = lld_i / ld_i, \quad d_i = ld_i^2 / lld_i, \quad i < n.$$

In this case, we would denote the ld_i 's, lld_i 's and d_n as *primary* data elements of the representation, and d_i for $i < n$ together with ℓ_i as *secondary* or *derived* data.

How the matrix is defined by the representation has a direct impact on how algorithms working with the matrix have to be designed. Thus, changing the kind of representation basically requires a new algorithm. In a floating-point context, different algorithms may behave quite differently, even if they are equivalent in theory. Then the question arises if one representation and its associated algorithm(s) might be better than another in some regard.

In the following we will develop and analyze corresponding algorithms for twisted factorizations represented by combining the pivots (1) with either one of (2)–(4). This leads to three basic types of representations. All of them have

$$d_1, \dots, d_{k-1}, \gamma_k, r_{k+1}, \dots, r_n$$

	ℓ, u	ld, ur	lld, uur
N-rep	primary	$ld_i = \ell_i d_i$ $ur_j = u_j r_j$	$lld_i = \ell_i^2 d_i$ $uur_j = u_j^2 r_j$
e-rep	$\ell_i = ld_i/d_i$ $u_j = ur_j/r_j$	primary	$lld_i = ld_i^2/d_i$ $uur_j = ur_j^2/r_j$
Z-rep	$\ell_i = \text{sign}(d_i) \sqrt{lld_i/d_i}$ $u_j = \text{sign}(r_j) \sqrt{uur_j/r_j}$	$ld_i = \sqrt{lld_i d_i}$ $ur_j = \sqrt{uur_j r_j}$	primary

Table 2.2: Computation of secondary (derived) data in the three basic representation types.

in common as first n primary data elements, but differ in their take for the remaining $n - 1$:

- The N-representation is based on (2), i.e., $\ell_1, \dots, \ell_{k-1}$ and u_{k+1}, \dots, u_n .
- The e-representation takes (3), that is, ld_1, \dots, ld_{k-1} and ur_{k+1}, \dots, ur_n .
- The Z-representation uses (4), i.e., lld_1, \dots, lld_{k-1} and uur_{k+1}, \dots, uur_n .

Note that the twist index k is formally for all types a parameter of the representational mapping and not data. Table 2.2 summarizes for the three types which of (2)–(4) is primary data and how the remaining secondary items can be defined in terms of them.

Remark 2.21. The motivation for the names N- and e-representations should be self-evident, but where does the Z come from?

The Z-representation is actually not a novel concept but has long been in (albeit hidden) use within the dqds-algorithm [30, 57] by Fernando & Parlett; we selected the name as tribute to this fact. Let us elaborate on the connection. The dqds-algorithm computes singular values of an upper bidiagonal matrix

$$\mathbf{B} = \text{diag}(a_1, \dots, a_n) + \text{diag}_{+1}(b_1, \dots, b_{n-1})$$

as eigenvalues of $\mathbf{B}^*\mathbf{B}$ and $\mathbf{B}\mathbf{B}^*$ in an iterative fashion. One step consists of transforming \mathbf{B} into another upper bidiagonal $\hat{\mathbf{B}}$ such that $\hat{\mathbf{B}}^*\hat{\mathbf{B}} - \tau = \mathbf{B}\mathbf{B}^*$. Thus, an upper (non-unit) bidiagonal factorization is shifted and rewritten as a lower (non-unit) bidiagonal one.

To facilitate a more elegant implementation and to avoid the need for taking roots, the process does not use the entries of \mathbf{B} directly, but their squares. Those are accumulated by the authors of [64] into an array

$$Z = (a_1^2, b_1^2, \dots, a_{n-1}^2, b_{n-1}^2, a_n^2).$$

Now comes the connection to representations of tridiagonal matrices. The factorization $\mathbf{B}\mathbf{B}^*$ is upper bidiagonal, but with factors that have no unit diagonal. This is easy to amend, since $\mathbf{B}\mathbf{B}^* = \mathbf{U}\mathbf{R}\mathbf{U}^*$ with $r_i = a_i^2, u_{i+1} = b_i/a_{i+1}$. So $u_{i+1}r_{i+1} = b_i^2$, meaning that the Z -array contains just the data we would take for the Z -representation of $\mathbf{U}\mathbf{R}\mathbf{U}^*$. \diamond

2.4.2 Stationary Factorization

The computation of

$$\mathbf{L}\mathbf{D}\mathbf{L}^* - \tau = \mathbf{L}^+\mathbf{D}^+(\mathbf{L}^+)^* \quad (2.34)$$

is called *stationary* transformation, because it is just the identity for $\tau = 0$. An evaluation of (2.34) row-by-row gives the conditions

$$\begin{cases} d_1 - \tau \stackrel{!}{=} d_1^+, \\ d_i + lld_{i-1} - \tau \stackrel{!}{=} d_i^+ + lld_{i-1}^+, & i = 2, \dots, n, \end{cases} \quad (2.35)$$

for the diagonal entries and

$$ld_i \stackrel{!}{=} ld_i^+, \quad i = 1, \dots, n-1, \quad (2.36)$$

for the offdiagonals. Those lead directly to a straightforward algorithm; shown below is the one for a Z -representation.

CS 2.6: *Stationary qd-transform with shift*

```

1:   $d_1^+ := d_1 - \tau$ 
2:  for  $i = 1$  to  $n - 1$  do
3:     $lld_i^+ := lld_i d_i / d_i^+$ 
4:     $d_{i+1}^+ := d_{i+1} + lld_i - lld_i^+ - \tau$ 
5:  endfor

```

Unfortunately, there is no way to control the relative errors in this version. The key to obtain numerical stability is to get a handle on the differences $lld_i - lld_i^+$ in line 4. These are important enough to warrant their own name, so define

$$s_i := lld_{i-1} - lld_{i-1}^+ \quad (2.37)$$

for $i = 1, \dots, n$ (which, given our implicit-zero assumption, means $s_1 = 0$). We refer to the s_i sometimes as *auxiliary quantities*, *auxiliaries* or *adjustments*. Then

$$\begin{aligned}
s_{i+1} &= \ell d_i - \ell d_i^+ = \ell d_i^2 \left(\frac{1}{d_i} - \frac{1}{d_i^+} \right) \\
&= \frac{\ell d_i^2}{d_i d_i^+} (d_i^+ - d_i) = \frac{\ell d_i^2}{d_i d_i^+} (s_i - \tau) \\
&= \frac{\ell d_i}{d_i^+} (s_i - \tau) = \ell_i^+ \ell_i (s_i - \tau).
\end{aligned} \tag{2.38}$$

The experienced reader may notice that our definition of s_i differs from the way it is done for example in [18] (the s_i there is our $s_i - \tau$). We have various reasons for our choice, not the least of them being that we will get the stronger property that the computed s_i are exact for perturbed data.

Remark 2.22 (Recursive form). Not only can we express the auxiliaries s_{i+1} recursively in terms of s_i , but on a related note they do allow to formulate the whole factorization process in a recursive form. Assume $\mathbf{LDL}^* - \tau = \mathbf{L}^+ \mathbf{D}^+ (\mathbf{L}^+)^*$, then Remark 2.17 gives

$$\mathbf{L}_{k:n} \mathbf{D}_{k:n} \mathbf{L}_{k:n}^* + s_k \mathbf{e}_1 \mathbf{e}_1^* - \tau = \mathbf{L}_{k:n}^+ \mathbf{D}_{k:n}^+ (\mathbf{L}_{k:n}^+)^*.$$

◇

Incorporating the auxiliaries leads to the *differential* version of the stationary transformation. Code for all three representation types is given in Algorithm 2.7. It should be self-explanatory, except maybe concerning some notational issues:

- We use the keyword **branch** to indicate that an instantiation of the algorithm should be tailored to one representation type and contain only one of the branches; in particular those are *not* conditionals. This is deliberately emphasized by using identical line numbers within each branch.
- Some intermediate expressions appear multiple times, like $s_i - \tau$. These should of course be computed just once, but we prefer to repeat them over having to introduce dummy variables.
- Braces must be obeyed as given for the coming error analysis to work, but where there are none, we do not care.

On the perpetual quest to keep notation simple, we want to avoid having to mark computed values by overbars or somesuch. Thus we use unadorned symbols to refer to the *primary* data elements that are inputs and outputs of computations; e.g., for a Z-representation this would be d_i , ℓd_i , and d_i^+ , ℓd_i^+ . We can do that because we are really not interested in the data of the exactly shifted matrix. However, note that if used as a variable this way, ℓd_i is not just an abbreviation for $\ell_i \ell_i d_i$ anymore, but an independent entity, denoting a floating-point number stored in memory. Concerning secondary data we have to be

careful. Staying with the Z-representation-example, the exact matrix defined by primary data d_i , ℓd_i has an exact offdiagonal element ℓd_i , which will generally not be a floating-point number and differ notably from the value

$$\overline{\ell d}_i := \text{fl}(\sqrt{\ell d_i d_i})$$

we would compute for it on a machine.

LDL*	N-rep	e-rep	Z-rep	L ⁺ D ⁺ (L ⁺)*	N-rep	e-rep	Z-rep
$d_i \rightsquigarrow \tilde{d}_i$	1	1	1	$d_i^+ \rightsquigarrow \tilde{d}_i^+$	2	2	2
$\ell_i \rightsquigarrow \tilde{\ell}_i$	3	3	2	$\ell_i^+ \rightsquigarrow \tilde{\ell}_i^+$	3	3	2
$\ell d_i \rightsquigarrow \tilde{\ell d}_i$	3	3	2	$\ell d_i^+ \rightsquigarrow \tilde{\ell d}_i^+$	3	3	2
$\ell \ell d_i \rightsquigarrow \tilde{\ell \ell d}_i$	5	5	3	$\ell \ell d_i^+ \rightsquigarrow \tilde{\ell \ell d}_i^+$	4	4	2

Table 2.3: Error bounds to achieve mixed relative stability for `dstqds` depending on the representation type used; cf. Theorem 2.24 and Figure 2.4. Only first-order bounds are shown, i.e., an entry k stands for a bound $k\epsilon_\diamond + \mathcal{O}(\epsilon_\diamond^2)$.

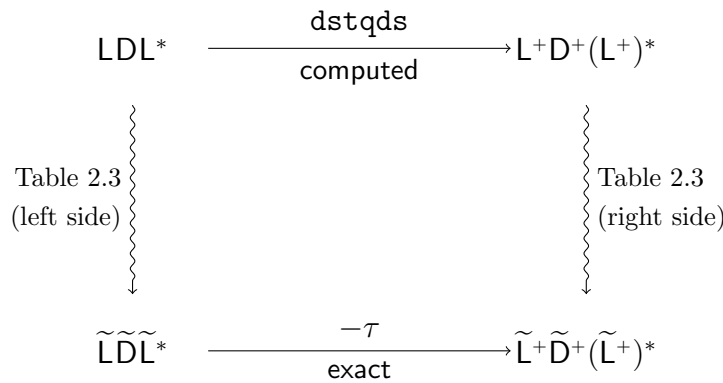


Figure 2.4: Mixed relative error analysis for `dstqds`.

Remark 2.23 (Flipped `dstqds`). It should be clear that Algorithm 2.7 can also be used to compute $\mathbf{URU}^* - \tau = \mathbf{U}^+\mathbf{R}^+(\mathbf{U}^+)^*$ by flipping the inputs beforehand and the results afterward, according to (2.16). We will refer to this simply as *flipped dstqds*. \diamond

Let us come to the (componentwise mixed relative) error analysis of the algorithm. It will be a bit of tedious work. The proof for the N-representation

ALGORITHM 2.7 *dstqds—template for all representations*

Given $\text{LDL}^* \in \mathbb{R}^{n \times n}$ and a shift $\tau \in \mathbb{R}$, compute a top-to-bottom factorization

$$\text{LDL}^* - \tau = \text{L}^+\text{D}^+(\text{L}^+)^*.$$

Input: Shift τ , data defining LDL^* in one of the representation types **N**, **e** or **Z**, i.e.,

$$d_1, x_1, \dots, d_{n-1}, x_{n-1} \quad \text{and} \quad d_n,$$

where x stands for ℓ , ld or lld , respectively.

Output: Data for $\text{L}^+\text{D}^+(\text{L}^+)^*$ in the same representation type as the input and auxiliary quantities, that is

$$d_1^+, x_1^+, \dots, d_{n-1}^+, x_{n-1}^+, d_n^+ \quad \text{and} \quad s_1, \dots, s_n.$$

```

1:   $s_1 := 0$ 
2:  for  $i = 1$  to  $n - 1$  do
3:     $d_i^+ := d_i + (s_i - \tau)$ 
      branch N:                                     // code for N-representation
4:     $\ell_i^+ := \ell_i d_i / d_i^+$ 
5:     $s_{i+1} := \ell_i^+ \ell_i (s_i - \tau)$ 
      branch e:                                     // code for e-representation
4:     $ld_i^+ := ld_i$ 
5:     $s_{i+1} := \frac{ld_i^2}{d_i d_i^+} (s_i - \tau)$ 
      branch Z:                                     // code for Z-representation
4:     $lld_i^+ := (lld_i / d_i^+) d_i$ 
5:     $s_{i+1} := (lld_i / d_i^+) (s_i - \tau)$ 
      endbranch
6:  endfor
7:   $d_n^+ := d_n + (s_n - \tau)$ 

```

was originally given by Dhillon in [15]; we repeat it here in adjusted form for the sake of completeness and for comparison purposes. The analysis for the other two types are new. The resulting bounds are stated in Table 2.3 for both primary and secondary data; the latter are useful because they are sharper than what could normally be inferred. For example, take the first column on the left (N-rep): If only bounds for $d_i \rightsquigarrow \tilde{d}_i$ and $\ell_i \rightsquigarrow \tilde{\ell}_i$ were given, the best that could be deduced for the implied perturbation $\ell d_i \rightsquigarrow \tilde{\ell} \tilde{d}_i$ would be $(3 + 3 + 1)\epsilon_\diamond = 7\epsilon_\diamond$, but we will see during the proof that it is actually bounded by $5\epsilon_\diamond$.

Theorem 2.24 (Error analysis of dstqds)

Let Algorithm 2.7 (instantiated for one representation type) be executed without underflow or overflow in an environment that satisfies Axiom FP.

Then the diagram in Figure 2.4 commutes, that is, there are perturbations to the inputs and outputs, with bounds listed in Table 2.3, such that

$$\tilde{\mathbf{L}}\tilde{\mathbf{D}}\tilde{\mathbf{L}}^* - \tau = \tilde{\mathbf{L}}^+\tilde{\mathbf{D}}^+(\tilde{\mathbf{L}}^+)^*.$$

The computed auxiliaries are exact for the perturbed data, i.e.,

$$s_1 = 0 \quad \text{and} \quad s_i = \tilde{\ell} \tilde{d}_{i-1} - \tilde{\ell} \tilde{d}_{i-1}^+, \quad i = 2, \dots, n.$$

Proof. A short note before we begin. Componentwise relative error analysis can be daunting, not because it is mathematically profound, but because one has to keep track of minutiae. This is not the only analysis of this kind we will have to do. But as it is the first, we will expose intermediate steps in more detail to highlight the techniques employed. In particular we single out the references to our rules from Chart 1.11, which we would normally use implicitly.

We will proceed per induction over i , with induction hypothesis being that the computed s_i is exact for the already specified perturbations of data $< i$. This is trivially true for $i = 1$.

We can deal with d_i and d_i^+ up to and including $i = n$ independently of the representation type, as lines 3 and 7 are always present. Their execution will give

$$d_i^+ / (1 + \alpha_+) = d_i + (s_i - \tau)(1 + \sigma), \quad |\alpha_+|, |\sigma| \leq \epsilon_\diamond. \quad (2.39)$$

The individual perturbations α_+ and σ will of course differ for each i , but the inductive setting allows to hide this dependency. Because we assume the computed s_i to be exact, we have to perturb $d_i \rightsquigarrow \tilde{d}_i$ and $d_i^+ \rightsquigarrow \tilde{d}_i^+$ such that the relation $\tilde{d}_i^+ = \tilde{d}_i + s_i - \tau$ holds. Looking at (2.39) this is achieved by

$$\tilde{d}_i := d_i \cdot (1 + \sigma)^{-1} \quad \Longrightarrow \quad d_i \rightsquigarrow d_i^+ \doteq \epsilon(1), \quad (2.40)$$

$$\tilde{d}_i^+ := d_i^+ \cdot (1 + \alpha_+)^{-1} (1 + \sigma)^{-1} \quad \Longrightarrow \quad d_i^+ \rightsquigarrow \tilde{d}_i^+ \doteq \epsilon(2), \quad (2.41)$$

where Corollary 1.12 was used to bound the errors in our $\epsilon(\cdot)$ -notation.

Branch for N-representation. The computations specific to this branch (lines 4 and 5) can be modeled as saying that the computed quantities ℓ_i^+ and s_{i+1} satisfy

$$\ell_i^+ = \ell_i d_i / d_i^+ \cdot (1 + \alpha_\ell), \quad (1 + \alpha_\ell) \doteq (1 + \epsilon_\diamond)^2, \quad (2.42)$$

$$s_{i+1} = \ell_i^+ \ell_i (s_i - \tau) \cdot (1 + \alpha_s)(1 + \sigma), \quad (1 + \alpha_s) \doteq (1 + \epsilon_\diamond)^2, \quad (2.43)$$

with σ being the one from (2.39). Note the use of $\epsilon_\diamond = \epsilon^{[0]}(1)$ on the right in “placeholder context”. The multiplication and division involved to compute ℓ_i^+ would normally be captured by two factors $(1 + \alpha_{\ell,*})(1 + \alpha_{\ell,/})$ with $|\alpha_{\ell,*}|, |\alpha_{\ell,/}| \leq \epsilon_\diamond$; we invoked rule (1) from Chart 1.11 to compress those to $\doteq (1 + \epsilon_\diamond)^2$.

Now we need to find perturbations $\ell_i \rightsquigarrow \tilde{\ell}_i$ and $\ell_i^+ \rightsquigarrow \tilde{\ell}_i^+$ such that the perturbed offdiagonals match and the computed s_{i+1} is exact. This gives two conditions

$$\tilde{\ell}_i^+ \tilde{d}_i^+ \stackrel{!}{=} \tilde{\ell}_i \tilde{d}_i \quad \text{and} \quad s_{i+1} \stackrel{!}{=} \tilde{\ell}_i^+ \tilde{\ell}_i (s_i - \tau)$$

that have to hold. For the first one, we have

$$\begin{aligned} \tilde{\ell}_i^+ \tilde{d}_i^+ \stackrel{!}{=} \tilde{\ell}_i \tilde{d}_i &\iff \frac{\tilde{\ell}_i^+ \tilde{d}_i^+}{\ell_i^+ d_i^+} \stackrel{!}{=} \frac{\tilde{\ell}_i \tilde{d}_i}{\ell_i d_i} (1 + \alpha_\ell)^{-1}, && \text{by (2.42),} \\ &\iff \frac{\tilde{\ell}_i^+}{\ell_i^+} (1 + \alpha_\ell) \stackrel{!}{=} \frac{\tilde{\ell}_i}{\ell_i} (1 + \alpha_+), && \text{by (2.40) and (2.41).} \end{aligned}$$

For the second one, just use (2.43) to see

$$s_{i+1} \stackrel{!}{=} \tilde{\ell}_i^+ \tilde{\ell}_i (s_i - \tau) \iff \tilde{\ell}_i^+ \tilde{\ell}_i \stackrel{!}{=} \ell_i^+ \ell_i \cdot (1 + \alpha_s)(1 + \sigma).$$

Both conditions are satisfied with

$$\tilde{\ell}_i := \ell_i \sqrt{\frac{(1 + \sigma)(1 + \alpha_\ell)(1 + \alpha_s)}{(1 + \alpha_+)}} \implies \ell_i \rightsquigarrow \tilde{\ell}_i \doteq \epsilon^{[4]}(3), \quad (2.44)$$

$$\tilde{\ell}_i^+ := \ell_i^+ \sqrt{\frac{(1 + \sigma)(1 + \alpha_+)(1 + \alpha_s)}{(1 + \alpha_\ell)}} \implies \ell_i^+ \rightsquigarrow \tilde{\ell}_i^+ \doteq \epsilon^{[4]}(3). \quad (2.45)$$

The error bounds were obtained by using Corollary 1.12 to write the terms under the root as $\doteq (1 + \epsilon(6))$ and then invoking rule (4) from Chart 1.11.

The resulting secondary perturbations to $\ell d_i, \ell \ell d_i, \ell d_i^+, \ell \ell d_i^+$ as they are stated in Table 2.3 (first column) can then be derived by combining (2.40), (2.41) with (2.44), (2.45) and cancelling terms. For instance, to get a bound on $\ell d_i^+ \rightsquigarrow \tilde{\ell} d_i^+$, (2.41) and (2.45) give

$$\begin{aligned} \tilde{\ell} d_i^+ &= \tilde{\ell}_i^+ \tilde{d}_i^+ = \ell_i^+ \sqrt{\frac{(1 + \sigma)(1 + \alpha_+)(1 + \alpha_s)}{(1 + \alpha_\ell)}} \cdot d_i^+ (1 + \alpha_+)^{-1} (1 + \sigma)^{-1} \\ &= \ell d_i^+ \sqrt{\frac{(1 + \alpha_s)}{(1 + \sigma)(1 + \alpha_+)(1 + \alpha_\ell)}} \doteq \ell d_i^+ (1 + \epsilon^{[4]}(3)). \end{aligned}$$

Branch for e-representation. This case is much simpler because the offdiagonal elements stay unchanged, $ld_i = ld_i^+$. The only real branch-specific computation performed is in line 5 and will result in

$$s_{i+1} = \frac{\widetilde{ld}_i^2}{\widetilde{d}_i \widetilde{d}_i^+} (s_i - \tau) \cdot (1 + \sigma)(1 + \alpha_s) \quad \text{with} \quad (1 + \alpha_s) \doteq (1 + \epsilon_\diamond)^4 \quad (2.46)$$

and σ from (2.39). Because s_i is exact by the induction hypothesis, what we have to ensure is

$$s_{i+1} = \frac{\widetilde{ld}_i^2}{\widetilde{d}_i \widetilde{d}_i^+} (s_i - \tau).$$

With the perturbations \widetilde{d}_i and \widetilde{d}_i^+ already fixed by (2.40) and (2.41), this leads to

$$\widetilde{ld}_i := ld_i \sqrt{\frac{(1 + \alpha_s)}{(1 + \sigma)(1 + \alpha_+)}} \quad \Longrightarrow \quad ld_i \rightsquigarrow \widetilde{ld}_i \doteq \epsilon^{[4]}(3), \quad (2.47)$$

$$\widetilde{ld}_i^+ := \widetilde{ld}_i \quad \Longrightarrow \quad ld_i^+ \rightsquigarrow \widetilde{ld}_i^+ \doteq \epsilon^{[4]}(3). \quad (2.48)$$

The implied secondary perturbations to $l_i, lld_i, l_i^+, lld_i^+$ as they are stated in Table 2.3 (second column) are then given by combining (2.40), (2.41) with (2.47), (2.48) and cancelling terms.

Branch for Z-representation. The quantities lld_i^+ and s_{i+1} as computed in lines 4 and 5 of this branch will come out as

$$lld_i^+ = (lld_i/d_i^+)d_i \cdot (1 + \alpha_\prime)(1 + \alpha_*), \quad (2.49)$$

$$s_{i+1} = (lld_i/d_i^+)(s_i - \tau) \cdot (1 + \alpha_\prime)(1 + \beta_*)(1 + \sigma), \quad (2.50)$$

with $|\alpha_\prime|, |\alpha_*|, |\beta_*| \leq \epsilon_\diamond$ and σ from (2.39). The rest is very similar to the N-branch. We need the perturbed data to fulfill

$$\widetilde{lld}_i^+ \widetilde{d}_i^+ \stackrel{!}{=} \widetilde{lld}_i \widetilde{d}_i \quad \text{and} \quad s_{i+1} \stackrel{!}{=} (\widetilde{lld}_i / \widetilde{d}_i^+) (s_i - \tau).$$

With the perturbations for d_i, d_i^+ already fixed by (2.40) and (2.41), both conditions can be satisfied through

$$\widetilde{lld}_i = lld_i \cdot (1 + \alpha_\prime)(1 + \beta_*) / (1 + \alpha_+) \quad \Longrightarrow \quad lld_i \rightsquigarrow \widetilde{lld}_i \doteq \epsilon(3), \quad (2.51)$$

$$\widetilde{lld}_i^+ = lld_i^+ \cdot (1 + \beta_*) / (1 + \alpha_*) \quad \Longrightarrow \quad lld_i^+ \rightsquigarrow \widetilde{lld}_i^+ \doteq \epsilon(2). \quad (2.52)$$

The derived perturbations for the secondary data $ld_i, lld_i, ld_i^+, lld_i^+$ as they are stated in Table 2.3 (third column) are then defined by the combination of (2.40), (2.41) with (2.51), (2.52), after cancelling terms. \square

2.4.3 Progressive Factorization

To compute is

$$\text{URU}^* - \tau = \text{L}^+\text{D}^+(\text{L}^+)^*, \quad (2.53)$$

called a *progressive* transformation, because—in contrast to the stationary case—the structural change is present even for $\tau = 0$. Just evaluating (2.53) line-by-line gives the conditions

$$\begin{cases} r_1 + \text{ur}_2 - \tau \stackrel{!}{=} d_1^+, \\ r_i + \text{ur}_{i+1} - \tau \stackrel{!}{=} d_i^+ + \text{lld}_{i-1}^+, & i = 2, \dots, n-1, \\ r_n - \tau \stackrel{!}{=} d_n^+ + \text{lld}_{n-1}^+, \end{cases} \quad (2.54)$$

for the diagonal entries and

$$\text{ur}_{i+1} \stackrel{!}{=} \text{ld}_i^+, \quad i = 1, \dots, n-1 \quad (2.55)$$

for the offdiagonals. A straightforward way to compute the factorization for a Z-representation is shown below.

CS 2.8: *Progressive qd-transform with shift*

```

1:   $d_1^+ := r_1 + \text{ur}_2 - \tau$ 
2:  for  $i = 2$  to  $n - 1$  do
3:       $\text{lld}_{i-1}^+ := \text{ur}_i r_i / d_{i-1}^+$ 
4:       $d_i^+ := r_i + \text{ur}_{i+1} - \text{lld}_{i-1}^+ - \tau$ 
5:  endfor
6:   $d_n^+ := r_n - \text{lld}_{n-1}^+ - \tau$ 

```

Analogously to the stationary case, the key to obtain a numerically stable formulation is to realize that the differences $r_i - \text{lld}_{i-1}^+$ can be written recursively. Define

$$\boxed{p_i := r_i - \text{lld}_{i-1}^+} \quad (2.56)$$

for $i = 1, \dots, n$ (implying $p_1 = r_1$) and recall $\text{ld}_i^+ \equiv \text{ld}_i$ to see

$$\begin{aligned} p_{i+1} &= r_{i+1} - \text{lld}_i^+ = r_{i+1} - \frac{\text{ld}_i^2}{d_i^+} \\ &= \frac{r_{i+1}}{d_i^+} \left(d_i^+ - \frac{\text{ld}_i^2}{r_{i+1}} \right) = \frac{r_{i+1}}{d_i^+} (d_i^+ - \text{ur}_{i+1}) \\ &= \frac{r_{i+1}}{d_i^+} (p_i - \tau). \end{aligned} \quad (2.57)$$

The adjusted computation for all three representation types is given in Algorithm 2.9. Concerning notation, the same conventions as for `dstqds` do apply.

What must seem like coming out-of-nowhere is the parameter *off*. For now just think of it being zero. We will need it in §2.4.2 for a seamless combination of `dstqds` and `dqds`.

Remark 2.25 (Flipped `dqds`). Analogously to Remark 2.23, Algorithm 2.7 can also be used to compute $\text{LDL}^* - \tau = \text{U}^+\text{R}^+(\text{U}^+)^*$ by flipping the inputs beforehand and the results afterwards, according to (2.16). \diamond

Perturbing the Shift. Before proceeding to the error analysis of `dqds` we need to introduce a new technique that we call *perturbing the shift*. Effectively it allows to relax the problem.

The task could have been posed more generally as shifting by an arbitrary diagonal matrix,

$$\mathbb{T} - \text{diag}(\tau_1, \dots, \tau_n) = \mathbb{T}_+.$$

The interesting thing is that for both the stationary and the progressive transformations, the definitions of the auxiliaries and their recursive formulations would remain essentially unchanged, except for writing τ_i instead of τ . This applies to the algorithms as well. For `dstqds` it suffices to replace $(s_i - \tau)$ by $(s_i - \tau_i)$ everywhere, and for `dqds` change $(r_1 - \tau)$ to $(r_1 - \tau_1)$ in line 1 and $(p_{i+1} - \tau)$ to $(p_{i+1} - \tau_{i+1})$ in line 6.

Of course, shifting by a diagonal is not very useful per se. But this changes if the shift matrix is close to a scalar multiple of the identity. If $\text{diag}(\tau_1, \dots, \tau_n) = \tau\mathbb{X}$, $\|\mathbb{X}\| \ll 1$, then

$$\begin{aligned} \mathbb{T}_+ &= \mathbb{T} - \text{diag}(\tau_1, \dots, \tau_n) = \mathbb{T} - \tau\mathbb{X} \\ &\iff \\ \mathbb{Y}\mathbb{T}_+\mathbb{Y} &= \mathbb{Y}\mathbb{T}\mathbb{Y} - \tau, \quad \mathbb{Y}^2 = \mathbb{X}^{-1}. \end{aligned} \tag{2.58}$$

Hence we obtain a standard shift relation by applying *outer* (multiplicative) perturbations \mathbb{Y} to the input matrix \mathbb{T} and the output \mathbb{T}_+ . From §1.4.3 we know that those are harmless. Indeed, outer perturbations are actually to be preferred over perturbing representation data, because they are independent of the matrices' condition. As a consequence, by transporting some perturbations from representational data elements to the shift we arrive at sharper error bounds that will be more robust.

The coming error analysis of `dqds` will be the first one where we use the technique to state the results with a modified shift τ_1 for the $(1, 1)$ entry. It will also prove to be of great use in Chapter 4.

ALGORITHM 2.9 dqds—*template for all representations*

Given $\text{URU}^* \in \mathbb{R}^{n \times n}$ and a shift $\tau \in \mathbb{R}$, compute a top-to-bottom factorization $\text{URU}^* + \text{off} \mathbf{e}_1 \mathbf{e}_1^* - \tau = \mathbf{L}^+ \mathbf{D}^+ (\mathbf{L}^+)^*$.

Input: Shift τ , initial adjustment *off* (default: 0), complete data defining URU^* in either one of **N**-, *e*- or **Z**-representation, i.e.,

$$r_1 \quad \text{and} \quad r_2, y_2, \dots, r_n, y_n,$$

where *y* stands for *u*, *ur* or *uur*, respectively.

Output: Data for $\mathbf{L}^+ \mathbf{D}^+ (\mathbf{L}^+)^*$ in the same representation type as the input and auxiliary quantities, that is,

$$d_1^+, x_1^+, \dots, d_{n-1}^+, x_{n-1}^+, d_n^+ \quad \text{and} \quad p_2, \dots, p_n,$$

where *x* denotes one of *ℓ*, *ℓd*, *ℓld*.

```

1:  Δ := off + (r1 - τ)
2:  for i = 1 to n - 1 do
      branch N:                                     // code for N-representation
3:     di+ := ui+12 ri+1 + Δ
4:     ℓi+ := (ri+1 / di+) ui+1
      branch e:                                     // code for e-representation
3:     di+ := uri+12 / ri+1 + Δ
4:     ℓdi+ := uri+1
      branch Z:                                     // code for Z-representation
3:     di+ := uuri+1 + Δ
4:     ℓldi+ := (ri+1 / di+) uuri+1
      endbranch
5:     pi+1 := (ri+1 / di+) Δ
6:     Δ := pi+1 - τ
7:  endfor
8:  dn+ := Δ

```

URU*	N-rep	e-rep	Z-rep	L+D+(L+)*	N-rep	e-rep	Z-rep
$r_i \rightsquigarrow \tilde{r}_i$	3	3	3	$d_i^+ \rightsquigarrow \tilde{d}_i^+$	2	2	2
$u_i \rightsquigarrow \tilde{u}_i$	3	3	3	$\ell_i^+ \rightsquigarrow \tilde{\ell}_i^+$	4	3	2
$ur_i \rightsquigarrow \tilde{ur}_i$	3	3	3	$ld_i^+ \rightsquigarrow \tilde{ld}_i^+$	4	3	2
$uur_i \rightsquigarrow \tilde{uur}_i$	3	3	1	$lld_i^+ \rightsquigarrow \tilde{lld}_i^+$	6	4	2

Table 2.4: Error bounds to achieve mixed relative stability for dqds depending on the representation type used; cf. Theorem 2.26 and Figure 2.5. Only first-order bounds are shown, i.e., an entry k stands for a bound $k\epsilon_\diamond + \mathcal{O}(\epsilon_\diamond^2)$.

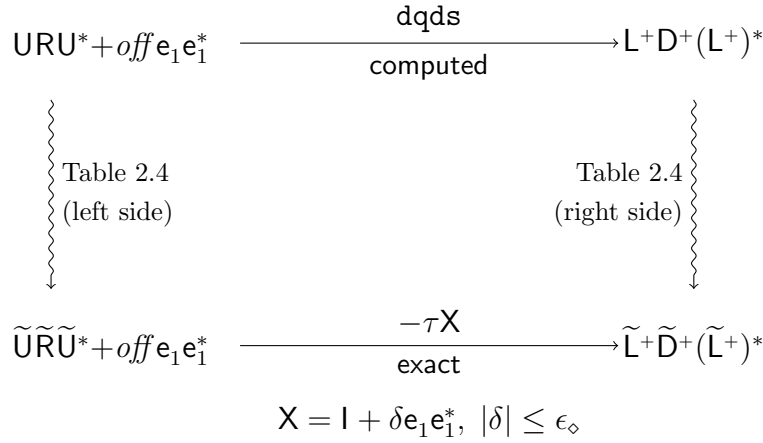


Figure 2.5: Mixed relative error analysis for dqds.

Theorem 2.26 (Error analysis of dqds)

Let Algorithm 2.9 (instantiated for one representation type) be executed without underflow or overflow in an environment that satisfies Axiom FP.

Then the diagram shown in Figure 2.5 commutes, that is, there are perturbations to the inputs and outputs, with bounds listed in Table 2.4, such that

$$\tilde{\text{UR}}\tilde{\text{U}}^* + \mathbf{e}_1 \mathbf{e}_1^* \text{off} - \tau(\mathbf{I} + \delta \mathbf{e}_1 \mathbf{e}_1^*) = \tilde{\text{L}}^+ \tilde{\text{D}}^+ (\tilde{\text{L}}^+)^*, \quad |\delta| \leq \epsilon_\diamond.$$

The computed auxiliaries are exact for the perturbed data, i.e.,

$$p_i = \tilde{r}_i - \tilde{\ell} d_{i-1}^+, \quad i = 2, \dots, n.$$

Proof. We will begin with analyzing the part involving the auxiliaries p_i and the local quantity Δ . This can be done independently of the representation type.

Concerning Δ , we will have

$$\Delta = [\text{off} + (r_1 - \tau)(1 + \eta)](1 + \pi_1), \quad |\eta|, |\pi_1| \leq \epsilon_\diamond$$

initially (line 1) and

$$\Delta = (p_{i+1} - \tau)(1 + \pi_2), \quad |\pi_2| \leq \epsilon_\diamond$$

after subsequent updates (line 6). Now perturb r_1 and the shift for the (1,1) diagonal entry as

$$\tilde{r}_1 := r_1(1 + \eta) \quad \Longrightarrow \quad r_1 \rightsquigarrow \tilde{r}_1 \doteq \epsilon_\diamond \quad (2.59)$$

$$\tilde{\tau}_1 := \tau(1 + \eta) \quad \Longrightarrow \quad \tau \rightsquigarrow \tilde{\tau}_1 \doteq \epsilon_\diamond. \quad (2.60)$$

Then we can state that the computed Δ will always be within one machine epsilon of the value it should have in exact arithmetic for the perturbed data, that is,

$$\Delta/(1 + \beta_\Delta) = \begin{cases} \text{off} + \tilde{r}_1 - \tilde{\tau}_1, & \text{initially, at loop entry for } i = 1, \\ p_i - \tau, & \text{at loop entry for } 1 < i < n, \\ p_n - \tau, & \text{at the end,} \end{cases} \quad (2.61)$$

for $|\beta_\Delta| \leq \epsilon_\diamond$. As d_n^+ is just a copy of the last Δ produced, this already shows how d_n^+ must be perturbed, namely

$$\tilde{d}_n^+ := d_n^+/(1 + \beta_\Delta) \quad \Longrightarrow \quad d_n^+ \rightsquigarrow \tilde{d}_n^+ \doteq \epsilon(1). \quad (2.62)$$

Execution of line 5 will give

$$p_{i+1} = (r_{i+1}/d_i^+)\Delta \cdot (1 + \beta_l)(1 + \beta_*), \quad |\beta_l|, |\beta_*| \leq \epsilon_\diamond. \quad (2.63)$$

With (2.61) we see that in order to get p_{i+1} to be exact for the perturbed data,

$$p_{i+1} \stackrel{!}{=} (\tilde{r}_{i+1}/\tilde{d}_i^+)\Delta/(1 + \beta_\Delta)$$

has to hold. Hence, the perturbations $r_{i+1} \rightsquigarrow \tilde{r}_{i+1}$ and $d_i^+ \rightsquigarrow \tilde{d}_i^+$ must be chosen to satisfy the condition

$$\frac{\tilde{r}_{i+1}}{\tilde{d}_i^+} \stackrel{!}{=} \frac{r_{i+1}}{d_i^+} \cdot (1 + \beta_l)(1 + \beta_*)(1 + \beta_\Delta). \quad (2.64)$$

Now we will have to start paying respect to the actual representation type and branch taken. The first line in each branch computes d_i^+ and can be captured by

$$d_i^+ = [z(1 + \zeta) + \Delta](1 + \beta_+), \quad \text{where } |\beta_+| \leq \epsilon_\diamond \text{ and} \quad (2.65)$$

$$\begin{cases} z = u_{i+1}^2 r_{i+1} & (1 + \zeta) \doteq (1 + \epsilon_\diamond)^2, & \text{for branch N,} \\ z = ur_{i+1}^2/r_{i+1}, & (1 + \zeta) \doteq (1 + \epsilon_\diamond)^2, & \text{for branch e,} \\ z = uur_{i+1}, & \zeta = 0, & \text{for branch Z.} \end{cases}$$

The quantity z that was just introduced is defined using primary data specific to the representation; let \tilde{z} be the corresponding value for the perturbed data. From (2.61) we know that the perturbed \tilde{d}_i^+ will then have to obey $\tilde{d}_i^+ \stackrel{!}{=} \tilde{z} + \Delta/(1+\beta_\Delta)$. Combined with (2.65) this leads us to

$$\tilde{z} \stackrel{!}{=} z \cdot (1 + \zeta)(1 + \beta_\Delta)^{-1} \quad (2.66)$$

and

$$\tilde{d}_i^+ := d_i^+ \cdot (1 + \beta_\Delta)^{-1}(1 + \beta_+)^{-1} \implies d_i^+ \rightsquigarrow \tilde{d}_i^+ \doteq \epsilon(2). \quad (2.67)$$

The latter combined with the condition (2.64) fixes the perturbation for r_{i+1} to

$$\tilde{r}_{i+1} := r_{i+1} \cdot (1 + \beta_l)(1 + \beta_*)(1 + \beta_+)^{-1} \implies r_{i+1} \rightsquigarrow \tilde{r}_{i+1} \doteq \epsilon(3). \quad (2.68)$$

The remaining analysis has to tread on different paths depending on the representation type, although not much remains to be done. To summarize where we stand, the perturbations $d_i^+ \rightsquigarrow \tilde{d}_i^+$ and $r_{i+1} \rightsquigarrow \tilde{r}_{i+1}$ have already been fixed and for the rest we will have to uphold (2.66).

Branch for N-representation. Here we have $\tilde{z} = \tilde{u}_{i+1}^2 \tilde{r}_{i+1}$, so with (2.65) and (2.68) the condition (2.66) requires us to perturb u_{i+1} according to

$$\tilde{u}_{i+1} := u_{i+1} \sqrt{\frac{(1 + \zeta)(1 + \beta_+)}{(1 + \beta_l)(1 + \beta_*)(1 + \beta_\Delta)}} \implies u_{i+1} \rightsquigarrow \tilde{u}_{i+1} \doteq \epsilon^{[4]}(3). \quad (2.69)$$

The remaining ℓ_i^+ is in line 4 effectively computed to be

$$\ell_i^+ = (r_{i+1}/d_i^+)u_{i+1} \cdot (1 + \beta_l)(1 + \alpha_*),$$

with $|\alpha_*| \leq \epsilon_\diamond$ and β_l being the same as in (2.63). Because \tilde{d}_i^+ and \tilde{r}_{i+1} were chosen to fulfill (2.64), (2.69) means that in order to attain $\tilde{\ell}_i^+ \tilde{d}_i^+ \stackrel{!}{=} \tilde{r}_{i+1} \tilde{u}_{i+1}$ we have to perturb ℓ_i^+ as

$$\tilde{\ell}_i^+ := \ell_i^+ \cdot \sqrt{\frac{(1 + \zeta)(1 + \beta_+)(1 + \beta_*)(1 + \beta_\Delta)}{(1 + \beta_l)(1 + \alpha_*)^2}} \doteq \ell_i^+ (1 + \epsilon^{[4]}(4)). \quad (2.70)$$

The derived perturbations for the secondary data ur_{i+1} , uur_{i+1} , ld_i^+ , lld_i^+ as they are stated in Table 2.4 (first column) are then defined by the combination of (2.67) and (2.68) with (2.69) and (2.70), after cancelling terms.

Branch for e-representation. As $\tilde{z} = \tilde{ur}_{i+1}^2/\tilde{r}_{i+1}$ and \tilde{r}_{i+1} is given by (2.68), the condition (2.66) can only be met with

$$\tilde{ur}_{i+1} := ur_{i+1} \cdot \sqrt{\frac{(1 + \zeta)(1 + \beta_l)(1 + \beta_*)}{(1 + \beta_\Delta)(1 + \beta_+)}} \doteq ur_{i+1} (1 + \epsilon^{[4]}(3)), \quad (2.71)$$

$$\tilde{ld}_i^+ := \tilde{ur}_{i+1} \doteq ld_i^+ (1 + \epsilon^{[4]}(3)). \quad (2.72)$$

For the derived perturbations of the secondary data $u_{i+1}, wur_{i+1}, \ell_i^+, lld_i^+$ as stated in Table 2.4 (second column), combine (2.67) and (2.68) with (2.71) and (2.72) and cancel terms.

Branch for Z-representation. In this case we have $z = wur_{i+1}$ and $\zeta = 0$, so the condition (2.66) does already spell out the perturbation for wur_{i+1} to be

$$\widetilde{wur}_{i+1} := wur_{i+1}(1 + \beta_\Delta)^{-1} \implies wur_{i+1} \rightsquigarrow \widetilde{wur}_{i+1} \doteq \epsilon(1). \quad (2.73)$$

The rest is nearly identical to the N-branch. The computation of lld_i^+ in line 4 will produce

$$lld_i^+ = (r_{i+1}/d_i^+)wur_{i+1} \cdot (1 + \beta_\gamma)(1 + \alpha_*),$$

with $|\alpha_*| \leq \epsilon_\diamond$ and the β_γ from (2.63). The perturbations \widetilde{d}_i^+ and \widetilde{r}_{i+1} were suitably chosen to give (2.64), so the desired relation $\widetilde{lld}_i^+ \widetilde{d}_i^+ \stackrel{!}{=} \widetilde{wur}_{i+1} \widetilde{r}_{i+1}$ is attained by

$$\widetilde{lld}_i^+ := lld_i^+ \cdot \frac{(1 + \beta_*)}{(1 + \alpha_*)} \doteq lld_i^+ (1 + \epsilon(2)). \quad (2.74)$$

The derived perturbations for secondary data, in this case $u_{i+1}, ur_{i+1}, \ell_i^+, ld_i^+$, with bounds as they are stated in Table 2.4 (third column), are obtained by combining (2.41) and (2.68) with (2.73) and (2.74), keeping $\zeta = 0$ in mind. \square

2.4.4 Twisted Factorization

Now we will investigate how stationary and progressive transformations can be combined to yield

$$\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^* - \tau = \mathbf{N}_t^+ \mathbf{G}_t^+ (\mathbf{N}_t^+)^* \quad (2.75)$$

for arbitrary combinations of twist indices k and t . For brevity we will sometimes drop the subscripts k and t and just write $\mathbf{N} \mathbf{G} \mathbf{N}^* - \tau = \mathbf{N}^+ \mathbf{G}^+ (\mathbf{N}^+)^*$. Evaluate (2.75) for the diagonal entries to see

$$\begin{aligned} d_i + lld_{i-1} - \tau &= d_i^+ + lld_{i-1}^+, & 1 \leq i < \min\{k, t\}, \\ r_i + wur_{i+1} - \tau &= d_i^+ + lld_{i-1}^+, & k < i < t, \\ d_i + lld_{i-1} - \tau &= r_i^+ + wur_{i+1}^+, & t < i < k, \\ r_i + wur_{i+1} - \tau &= r_i^+ + wur_{i+1}^+, & \max\{k, t\} < i \leq n, \end{aligned} \quad (2.76)$$

and

$$\begin{aligned} \text{if } k < t : & \begin{cases} lld_{k-1} + \gamma_k + wur_{k+1} - \tau = d_k^+ + lld_{k-1}^+, \\ r_t + wur_{t+1} - \tau = lld_{t-1}^+ + \gamma_t^+ + wur_{t+1}^+, \end{cases} \\ \text{if } k = t : & lld_{k-1} + \gamma_k + wur_{k+1} - \tau = lld_{t-1}^+ + \gamma_t^+ + wur_{t+1}^+, \\ \text{if } k > t : & \begin{cases} d_t + lld_{t-1} - \tau = lld_{t-1}^+ + \gamma_t^+ + wur_{t+1}^+, \\ lld_{k-1} + \gamma_k + wur_{k+1} - \tau = r_k^+ + wur_{k+1}^+. \end{cases} \end{aligned} \quad (2.77)$$

Define $a := \min\{k, t\}$ and $b := \max\{k, t\}$. Then the above conditions reveal that the task can be broken into four parts:

- A stationary transformation from the top down to a , except for the (a, a) -entry; in matrix notation this becomes

$$\underbrace{N_{1:a} G_{1:a} N_{1:a}^*}_{\text{LDL}} + e_a e_a^* u u r_{a+1} - \tau = \underbrace{N_{1:a}^+ G_{1:a}^+ (N_{1:a}^+)^*}_{\text{LDL}} + e_a e_a^* u u r_{a+1}^+.$$

(Note that for $k \neq t$, one of $u u r_{a+1}$, $u u r_{a+1}^+$ will be undefined and hence equal zero.)

- Analogously, a (flipped) stationary transformation from n up to b , violated only at the (b, b) -entry, that is,

$$\underbrace{N_{b:n} G_{b:n} N_{b:n}^*}_{\text{URU}} + e_1 e_1^* l l d_{b-1} - \tau = \underbrace{N_{b:n}^+ G_{b:n}^+ (N_{b:n}^+)^*}_{\text{URU}} + e_1 e_1^* l l d_{b-1}^+.$$

- If $k < t$ a progressive transformation from k down to t except at the first and last diagonal entries, that is,

$$\begin{aligned} \underbrace{N_{k:t} G_{k:t} N_{k:t}^*}_{\text{URU}} - \tau + e_1 e_1^* l l d_{b-1} + e_m e_m^* u u r_{t+1} \\ = e_1 e_1^* l l d_{k-1}^+ + e_m e_m^* u u r_{t+1}^+ + \underbrace{N_{k:t}^+ G_{k:t}^+ (N_{k:t}^+)^*}_{\text{LDL}} \end{aligned}$$

with $m := t - k + 1 > 1$.

- Analogously, if $t < k$ a (flipped) progressive transformation from t up to k except at the first and last diagonal entries, that is,

$$\begin{aligned} \underbrace{N_{t:k} G_{t:k} N_{t:k}^*}_{\text{LDL}} - \tau + e_1 e_1^* l l d_{t-1} + e_m e_m^* u u r_{k+1} \\ = e_1 e_1^* l l d_{t-1}^+ + e_m e_m^* u u r_{k+1}^+ + \underbrace{N_{t:k}^+ G_{t:k}^+ (N_{t:k}^+)^*}_{\text{URU}}, \end{aligned}$$

with $m := k - t + 1 > 1$.

Note that there is at most one progressive part and none if $k = t$. They are welded together at the twist positions k and t for which we will have to devise special treatment. The formulas above already suggest that the auxiliaries from the stationary and progressive processes can be harnessed for this purpose. To differentiate from which part they did originate we use decorations $\tilde{}$ for a top-to-bottom process and $\hat{}$ for bottom-to-top. Thus, recalling (2.37) and (2.56) and

adjusting to the fact that the twist elements $\gamma_k = \mathbf{G}(k, k)$ and $\gamma_t^+ = \mathbf{G}^+(t, t)$ can be regarded both as d and as r depending on the part we are in, we get

$$\begin{aligned}\check{s}_i &= \ell d_{i-1} - \ell d_{i-1}^+, & 1 \leq i \leq \min\{k, t\}, \\ \check{p}_i &= \mathbf{G}(i, i) - \ell d_{i-1}^+, & k < i \leq t, \\ \hat{p}_i &= \mathbf{G}(i, i) - u u r_{i+1}^+, & t \leq i < k, \\ \hat{s}_i &= u u r_{i+1} - u u r_{i+1}^+, & \max\{k, t\} \leq i \leq n.\end{aligned}\tag{2.78}$$

Plug these into (2.77) to get a formulation for how the computation can be completed at the twist positions:

$$\begin{aligned}\text{if } k < t : & \quad \begin{cases} d_k^+ = u u r_{k+1} + \gamma_k + \check{s}_k - \tau, \\ \gamma_t^+ = \check{p}_t + \hat{s}_t - \tau, \end{cases} \\ \text{if } k = t : & \quad \gamma_k^+ = \gamma_k + \check{s}_k + \hat{s}_k - \tau, \\ \text{if } k > t : & \quad \begin{cases} \gamma_t^+ = \check{s}_t + \hat{p}_t - \tau, \\ r_k^+ = \ell d_{k-1} + \gamma_k + \hat{s}_k - \tau. \end{cases}\end{aligned}\tag{2.79}$$

The finishing touch is to realize that the expressions for d_k^+ and r_k^+ in the case $k \neq t$ are nearly what the corresponding progressive transformation would do in its first step anyways, except for \hat{s}_k and \check{s}_k . Here it comes to fruit that we formulated `dqds` with the adjustment *off* for the first diagonal entry. Altogether this culminates in Algorithm 2.10. The name `dtwqds` was coined by Dhillon for his algorithm to compute $\text{LDL}^* - \tau = \mathbf{N}_t^+ \mathbf{G}_t^+ (\mathbf{N}_t^+)^*$; we reuse it here for what is, in fact, a generalization.

$\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$	N-rep	e-rep	Z-rep	$\mathbf{N}_t^+ \mathbf{G}_t^+ (\mathbf{N}_t^+)^*$	N-rep	e-rep	Z-rep
d_i, r_j	3	3	3	d_i^+, r_j^+	2	2	2
ℓ_i, u_j	3	3	3	ℓ_i^+, u_j^+	4	3	2
$\ell d_i, u r_j$	3	3	3	$\ell d_i^+, u r_j^+$	4	3	2
$\ell \ell d_i, u u r_j$	5	5	3	$\ell \ell d_i^+, u u r_j^+$	6	4	2
γ_k	3	3	3	γ_t^+	3	3	3

Table 2.5: Worst-case error bounds to achieve mixed relative stability for `dtwqds` depending on the representation type used; cf. Theorem 2.28 and Figure 2.6. Only first-order bounds are shown, i.e., an entry p stands for a perturbation $x \rightsquigarrow \tilde{x} \doteq p\epsilon_\circ + \mathcal{O}(\epsilon_\circ^2)$.

ALGORITHM 2.10 `dtwqds`—*template for all representations*

Given twisted factorization $\text{NGN}^* \in \mathbb{R}^{n \times n}$ with twist k and a shift $\tau \in \mathbb{R}$ Factorize $\text{NGN}^* - \tau = \text{N}^+ \text{G}^+ (\text{N}^+)^*$ with twist t .

Input: Shift τ , data for NGN^* in either one of a N^- , e^- or Z^- representation, desired twist t in the target.

Output: Data for $\text{N}^+ \text{G}^+ (\text{N}^+)^*$ in the same representation type as the input.

Note: Let `dstqds_part` and `dqds_part` stand for versions of Algorithms 2.7 and 2.9 where the last pivots are discarded, i.e., without lines 7 and 8, respectively.

- 1: Use `dstqds_part` on $\text{N}_{1:a} \text{G}_{1:a} \text{N}_{1:a}^*$, where $a = \min\{k, t\}$, to compute pivots d_1^+, \dots, d_{a-1}^+ and auxiliaries $\check{s}_1, \dots, \check{s}_a$.
// Amounts to just $\check{s}_1 = 0$ if $k = 1$ or $t = 1$.
- 2: Use flipped `dstqds_part` on $\text{N}_{b:n} \text{G}_{b:n} \text{N}_{b:n}^*$, where $b = \max\{k, t\}$, to compute pivots $r_n^+, r_{n-1}^+, \dots, r_{b+1}^+$ and auxiliaries $\hat{s}_n, \hat{s}_{n-1}, \dots, \hat{s}_b$.
// Amounts to just $\hat{s}_n = 0$ if $k = n$ or $t = n$.
- 3: **if** $k < t$ **then**
- 4: Use `dqds_part` with *off* = \check{s}_k on $\text{N}_{k:t} \text{G}_{k:t} \text{N}_{k:t}^*$ to compute pivots d_k^+, \dots, d_{t-1}^+ and auxiliaries $\check{p}_{k+1}, \dots, \check{p}_t$.
- 5: $\gamma_t^+ := (\check{p}_t + \hat{s}_t) - \tau$
- 6: **else if** $k = t$ **then**
- 7: $\gamma_k^+ := \gamma_t + ((\check{s}_t + \hat{s}_t) - \tau)$
- 8: **else**
- 9: Use flipped `dqds_part` with *off* = \hat{s}_k on $\text{N}_{t:k} \text{G}_{t:k} \text{N}_{t:k}^*$ to compute pivots $r_k^+, r_{k-1}^+, \dots, r_{t+1}^+$ and auxiliaries $\hat{p}_{k-1}, \hat{p}_{k-2}, \dots, \hat{p}_t$.
- 10: $\gamma_t^+ := (\check{s}_t + \hat{p}_t) - \tau$
- 11: **endif**

Proof. Naturally we deploy Theorems 2.24 and 2.26 for the executed stationary and progressive parts. The thing to realize is that this fixes perturbations for everything except the twist elements and does provide exact auxiliaries and delivers all entrywise desired relations except for the diagonal entries at the twist positions.

Case $k \neq t$. Due to symmetry it suffices to prove the case $k < t$. Here, γ_k was gobbled in the progressive part (interpreted as r_k), so its perturbation is already fixed. The offset given to `dqds` ensures that the perturbed data fulfills the desired (k, k) diagonal entry relation.

Let the new twist element be computed in line 5 according to

$$\gamma_t^+ = [(\check{p}_t + \hat{s}_t)/(1 + \pi) - \tau]/(1 + \sigma), \quad |\pi|, |\sigma| \leq \epsilon_\diamond.$$

We have \check{p}_t and \hat{s}_t being exact, so by perturbing γ_t^+ and the shift as

$$\begin{aligned} \tilde{\gamma}_t^+ &:= \gamma_t^+(1 + \pi)(1 + \sigma) &\implies &\gamma_t^+ \rightsquigarrow \tilde{\gamma}_t^+ \doteq \epsilon(2), \\ \tilde{\tau}_t &:= \tau(1 + \pi) &\implies &\tau \rightsquigarrow \tilde{\tau}_t \doteq \epsilon(1), \end{aligned}$$

we get $\tilde{\gamma}_t^+ = \check{p}_t + \hat{s}_t - \tilde{\tau}_t$, as desired.

Case $k = t$. Let the new twist element be computed in line 5 as

$$\gamma_t^+ = \left[\gamma_t + ((\check{s}_t + \hat{s}_t)/(1 + \pi) - \tau)/(1 + \sigma) \right] / (1 + \omega) \quad |\pi|, |\sigma|, |\omega| \leq \epsilon_\diamond.$$

In this case there is no progressive part, so as yet neither γ_t nor γ_t^+ have been touched. Building on the auxiliaries \check{s}_t, \hat{s}_t being exact, we fix

$$\begin{aligned} \tilde{\gamma}_t &:= \gamma_t(1 + \pi)(1 + \sigma) &\implies &\gamma_t \rightsquigarrow \tilde{\gamma}_t \doteq \epsilon(2), \\ \tilde{\gamma}_t^+ &:= \gamma_t^+(1 + \pi)(1 + \sigma)(1 + \omega) &\implies &\gamma_t^+ \rightsquigarrow \tilde{\gamma}_t^+ \doteq \epsilon(3), \\ \tilde{\tau}_t &:= \tau(1 + \pi) &\implies &\tau \rightsquigarrow \tilde{\tau}_t \doteq \epsilon(1), \end{aligned}$$

and obtain the last missing piece, namely $\tilde{\gamma}_t^+ = \tilde{\gamma}_t + \check{s}_t + \hat{p}_t - \tilde{\tau}_t$.

A final remark on how the error bounds in Table 2.5 are obtained. These are worst-case bounds in the sense that they do not depend on k and t . Then each data element can belong to a stationary or a progressive part, flipped or not. Hence we have to take the respective worse bounds from Tables 2.3 and 2.4. \square

2.4.5 Additional Remarks & Discussion

Comparison of Representation Types

We have seen that N-, e- and Z-representations have different characteristics concerning rounding errors. The natural question is, which one is best. Tables 2.3, 2.4 and 2.5 pronounce a Z-representation as the clear winner when accuracy is the only concern, but so far we have not yet taken efficiency into account.

To optimize for speed, one would implement the algorithms such that subterms can be cached for repeated use. A prominent candidate to be precomputed and cached is ℓd_i , because then an optimized loop for bisection (where only the signs of the d_i^+ are wanted) could for all representation types use the same formula for s_{i+1} as the Z-branch, needing just one division and one multiplication.

CS 2.11: *Optimized bisection, all representation-types*

```

1:   ...
2:    $d_i^+ := d_i + (s_i - \tau)$ 
3:   if  $d_i^+ < 0$  then
4:      $negcount := negcount + 1$ 
5:   endif
6:    $s_{i+1} := (\ell d_i / d_i^+)(s_i - \tau)$ 
7:   ...

```

Let us focus on the N- and e -branches of `dstqds`. To precompute whichever one of ℓ_i and ℓd_i is not primary data too is less useful in general, but at least the ℓ_i 's will be needed if the eigenvector-equation associated with the matrix is to be solved. Below, two modified loop bodies for the N- and e -branches are shown that will compute everything:

branch N:

```

1:    $\ell_i^+ := \ell d_i / d_i^+$ 
2:    $s_{i+1} := \ell_i^+ \ell_i (s_i - \tau)$ 
3:    $\ell d_i^+ := \ell_i^+ d_i^+$ 
4:    $\ell \ell d_i^+ := \ell d_i^+ \ell_i^+$ 

```

branch e :

```

1:    $\ell_i^+ := \ell d_i / d_i^+$ 
2:    $s_{i+1} := \ell_i^+ \ell_i (s_i - \tau)$ 
3:    $\ell d_i^+ := \ell d_i$ 
4:    $\ell \ell d_i^+ := \ell d_i^+ \ell_i^+$ 

```

Both require one division and two multiplications at minimum (lines 1 and 2), and one or two further multiplications to compute ℓd_i^+ and $\ell \ell d_i^+$ (lines 3 and 4). So we see that working with an e -representation is in all situations at least as efficient as using a N-representation.

The nice thing is that our error analysis in Theorem 2.24 is still valid for these modified loops, and also for the optimized bisection in CS 2.11, because we did not specify the order in which the multiplications and divisions in lines 4–5 of `dstqds` are executed.

Concerning the Z-representation, examining the Z-branch of `dstqds` shows that bisection can be done equally fast (1 div, 1 mult) and a plain factorization giving only primary data too (1 div, 2 mult). That is all nice and well as long as we do not need ℓ_i . If we do, it gets costly, in the form of one *division* and one *square root*², cf. Table 2.2. Hence an RQI to compute eigenvectors would become

²On most modern machines a division and a square root should take about the same number of cycles.

prohibitively expensive using Z-representations. To make them nevertheless usable for MR³, we propose the following workaround:

- Use plain Z-representations for all nodes, without computing secondary data.
- For nodes with singleton eigenvalues (and only for those), switch to an e -representation (of a nearby matrix) by computing $ld_i := \sqrt{\overline{\ell d_i d_i}}$ and use that one to do RQIs for all singletons. The cost is then basically n additional square roots per node with singleton eigenvalues and should amortize over the whole run.

A similar argumentation for **dqds** comes to the same conclusions, in fact, even the optimized operation counts are identical. To summarize the pros and cons for the representation types:

e -representation

- + (Minimally) fastest computation, if properly optimized.
- + The offdiagonal entries staying the same means they need only be stored once for the root and can then be reused for every node in the tree. This basically halves the memory traffic for MR³.
- + Error analysis is simpler because there is less data to perturb, cf. the proofs of Theorems 2.24 and 2.26.
 - Accuracy is comparable to an N-representation.

Z-representation

- + By far the sharpest error bounds.
 - As fast as an e -representation as long as secondary data items (ℓ_i, ld_i, u_j, ur_j) are not required, i.e., for bisection and shifting.
 - The previous makes it impracticable for computing eigenvectors. A workaround like the one mentioned above will have to be used.

N-representation

- + As fast as an e -representation as long as the offdiagonals (ld_i, ur_j) are not needed, otherwise minimally slower.
 - Accuracy is comparable to an e -representation.

As result, we can state that a Z-representation should be used when accuracy is of paramount concern. In particular, the greatly reduced effected perturbations $\ell d_i \rightsquigarrow \widetilde{\ell d_i}$ and $ur_i \rightsquigarrow \widetilde{ur_i}$ will lessen the impact of moderate local element growth on the diagonal. Otherwise, an e -representation is best as it gives superior performance. There is really no reason to stick with the traditional N-representation.

Remark 2.29 (A word of caution). The optimized loop bodies for the N- and e-branches above are deceptively similar (only difference in line 3), and so are the perturbation bounds. Nevertheless they are not numerically equivalent and may lead MR³ onto different computational paths. The thing is that the floating-point incarnations of d_i , ℓ_i , ld_i and lld_i as they are stored in the machine do not enjoy the correlation they would have in exact arithmetic. In order to keep reasoning clear one must keep track of the matrix, meaning the exact mathematical object which is defined by the primary data and for which we do have mixed relative stability. This is basically the reason why we have been so pedantic about the distinction between primary and secondary data. \diamond

Alternative Problems

There are two related problem formulations:

- (a) Factorize $\mathbf{T} - \tau = \mathbf{N}_k^+ \mathbf{G}_k^+ (\mathbf{N}_k^+)^*$ for symmetric tridiagonal \mathbf{T} represented by its entries. MR³ needs this to find a root RRR for the initial matrix.
- (b) Factorize $\mathbf{B}\mathbf{B}^* - \tau = \mathbf{N}^+ \mathbf{G}^+ (\mathbf{N}^+)^*$ and $\mathbf{B}^*\mathbf{B} - \tau = \mathbf{N}^+ \mathbf{G}^+ (\mathbf{N}^+)^*$ for upper bidiagonal \mathbf{B} given by its entries, cf. Remark 2.21; this will be needed in Chapter 3.

In his groundbreaking technical report [48], Kahan provided a thorough treatment of (a) for $k = n$ in the context of eigenvalue computations. He gives a componentwise relative backward error analysis for the computed Sturm counts and takes into account proper scaling to avoid underflow and overflow (cf. also breakdowns below). As icing on the cake he even provides a rigorous proof that the computed counts of negative eigenvalues (Sturm counts) are monotonic in the shift if one can assume that the floating-point arithmetic unit obeys a set of reasonable conditions on top of Axiom FP. It is an open problem if the differential algorithms presented in this section enjoy a similar property.

The problem (b), as well as (a) for matrices with a zero diagonal, are investigated by Fernando in [28], again in the context of computing eigenvalues via bisection. He also explores internal connections between the factorizations. In turn, those led to the coupling relations by Großer and Lang [36] that will become a core topic in Chapter 3. Tailored Algorithms for (b) are used within the dqds algorithm and can also be found in [35]; by writing $\mathbf{B}^*\mathbf{B} = \mathbf{L}\mathbf{D}\mathbf{L}^*$ and $\mathbf{B}\mathbf{B}^* = \mathbf{U}\mathbf{R}\mathbf{U}^*$ it is not hard to recognize them as special cases of `dstqds` and `dqds` for Z-representations.

Treatment of Breakdowns

It is possible to safeguard the factorizations with regard to breakdowns. One approach, dating back to [48], is to deflect pivots away from an interval around zero. For `dstqds` and `dqds` this would look as follows:

```

1:    $d_i^+ := \dots$ 
2:   if  $|d_i^+| < pivmin$  then
3:      $d_i^+ := -pivmin$ 
4:   endif
5:    $\dots$ 

```

Normally the matrix is scaled (and splitted) beforehand to lie in a sensible range and *pivmin* is set to something like $o_{\diamond}^{-1} \max_i |ld_i^2|$, with the intent that $lld_i^+ = ld_i^2/d_i^+$ cannot overflow anymore. The modification can be modeled as changing the diagonal entries by no more than $2|pivmin|$. Hence, in the context of the shift-relation we want to preserve for MR³, it is harmless as long as $|pivmin|$ does not exceed $\epsilon_{\diamond}|\tau|$, because we can subsume it as perturbation to the shift.

The disadvantage is that the extra conditional causes a performance hit. An alternative is possible if IEEE-conform handling of NaN's and ∞ is present, see for example [54]. The resulting matrix will have invalid rows and columns due to entries being ∞ , but a Sturm count can still be used, making the technique very useful to speed up bisection. It is also possible to compute eigenvectors for twisted factorizations with infinities [15], but there the net gain is reduced compared to loosing the exact mathematical relation provided by the *pivmin*-scheme and the fact that underflow in the vector entries will still have to be guarded against (cf. §4 in [19]).

2.5 Towards an improved implementation of MR³

Much of the research work leading to this thesis has been spent in software. During our work on MR³-based solutions to compute the singular value decomposition of a bidiagonal matrix (problem BSVD, which will be the topic of the next chapter), it quickly became apparent that there were many more extreme numerical cases than for the tridiagonal problem. This led us to investigate how parts of existing implementations of MR³ could be improved, with particular emphasis on reliability and robustness.

MR³ is an algorithm that can be explained rather compactly, if you leave out the details. But to get the subcomponents—like the RQI for computing vectors, or finding a good shift—working just “right” is a tricky business. We desired, for example, to study what the use of twisted factorizations with somehow optimally chosen twist index as representations would bring, or how a different heuristic for evaluating shift candidates does perform.

It is not really feasible to facilitate greater algorithmic changes in reasonable time while sticking to an optimized FORTRAN 77 codebase. We have developed an extensive software framework to prototype MR³-based solution approaches for TSEP (and also BSVD)³. For ultimate flexibility and code-reusability we have

³At the time of this writing, more than 30000 lines of source code.

chosen an object-oriented approach in C++. It is completely configurable using plain text files, and contains copious instrumentation for purposes of debugging, logging, and generation of statistics. Furthermore it can call external LAPACK-solvers to evaluate their results in the same setting.

The disadvantage of a prototypical approach is that we cannot provide timing results right now, but only extrapolated cost based on counting iterations. However, we feel this to be well outweighed by the fact that we can now rather easily try out algorithmic changes to MR³ that were not practicable before.

The purpose of this section is not to describe our software framework, which is, after all, just means to an end. Instead we want to pit our so far best MR³-implementation against LAPACK.

In the following, when we write `DSTEMR` we always mean the implementation of MR³ in LAPACK 3.2.1, which is—at the time of this writing—still the most recent one. For the lack of a better name, let us denote the chosen MR³-instantiation of our framework as `XMR`.

How to achieve the requirements

We know that any implementation of MR³ has to heed the five requirements on page 55. Three of those are actually of no concern:

- `RELGAPS` comes mostly for free provided the representation is robust in even the most rudimentary terms, and the implied conditions for the outer gaps are easily checked.
- `SHIFTREL` is fulfilled if the routine to compute the child representation is mixed relatively stable. From §2.4 we know how to do that for twisted factorizations with arbitrary twist index.
- `GETVEC` is not always easy to achieve, but if some variant of Algorithm 2.5 is employed it will deliver a bound for the residual norm, so fulfillment of `GETVEC` can be monitored.

The remaining two, `RRR` and `ELG`, have to be attained in step 12 of Algorithm 2.1, where the shifts are selected. The problem with them is that we cannot always know *a priori*, that is, at the time of computing the child representation, if the requirements are met. Sometimes this is only possible to know for sure *after* the vectors have been computed, called *a posteriori*.

That the vectors are needed for evaluation is evident from how `ELG` is stated, and for `RRR` it is clear from (2.32). The bounds that can be inferred without eigenvector information are more often than not too crude to be useful. However, it turns out that to obtain practicable *a priori* bounds, it usually suffices to know where all vectors in the local invariant subspace of interest have small entries.

This information is captured in the *envelope* of the subspace, which is formally defined as vector $\text{env}_{\mathcal{S}} \in \mathbb{R}^n$ with entries

$$\text{env}_{\mathcal{S}}(i) := \max \{ |x(i)| : x \in \mathcal{S}, \|x\| = 1 \}.$$

The amount of accurate information that can be cheaply determined about an envelope can sometimes be astonishing [58, 65].

We see that the components of the envelope are upper bounds on entries of any unit vector from the subspace. If some of them are small we say there is *eigenvector localization*. Localization can only reasonably be expected for clusters $I = \{c, \dots, d\}$ that are small, say $|d - c| < 0.3n$, as well as *tight* in the sense

$$|\lambda_c - \lambda_d| \leq \text{tfac} \text{gap}_{\mathbf{A}}(I), \quad \text{tfac} \ll 1, \quad (2.80)$$

with, for instance, $\text{tfac} = 2^{-8}$.

Suppose we have an approximation \mathbf{s} for $\text{env}_{\mathcal{S}}$. It should be faithful in the sense $\text{env}_{\mathcal{S}}(i) \leq \mathbf{s}(i) \leq 1$ for all entries, but might have $\mathbf{s}(i) \equiv 1$ if no useful information could be deduced. For a twisted factorization $\mathbf{M} = \mathbf{N}\mathbf{G}\mathbf{N}^*$ as representation it is not hard to see that ELG boils down to the condition

$$\|\mathbf{G}\mathbf{s}\| \leq C_{\text{elg}} \text{spdiam}[\mathbf{M}_0]. \quad (2.81)$$

For requirement RRR, the formula (2.32) for relative condition numbers is not well suited to be used with an envelope. A better one can be found in [63, p. 126]; we adapt it now to twisted factorizations. The first step is to eliminate the explicit dependence on \mathbf{G} , noting that

$$\begin{aligned} \delta\lambda &\leq |\mathbf{q}^* \mathbf{N} \Omega \mathbf{G} \mathbf{N}^* \mathbf{q}| && \text{by (2.31), } \Omega \mathbf{G} = |\mathbf{G}|, \\ &\leq |\mathbf{q}^* \mathbf{N} \Omega \mathbf{N}^{-1} \mathbf{q} \lambda| && \text{since } \mathbf{N} \mathbf{G} \mathbf{N}^* \mathbf{q} = \mathbf{q} \lambda. \end{aligned}$$

This implies

$$\frac{\delta\lambda}{|\lambda|} \leq |\mathbf{q}^* \mathbf{N} \Omega \mathbf{N}^{-1} \mathbf{q}| \leq \|\Upsilon \mathbf{N}^* \mathbf{q}\| \|\Upsilon^{-1} \mathbf{N}^{-1} \mathbf{q}\|$$

for any (nonsingular) diagonal matrix Υ . Hence, requirement RRR can be checked a priori via the condition

$$\|\Upsilon \mathbf{N}^* \mathbf{s}\| \|\Upsilon^{-1} \mathbf{N}^{-1} \mathbf{s}\| \leq \text{maxrc}, \quad (2.82)$$

where Υ can be chosen freely to minimize the resulting bound, and maxrc is a parameter of moderate size, say 10.

Outline of DSTEMR

For comparison purposes, let us first describe in a few words how the latest DSTEMR in LAPACK 3.2.1 was set up; for more information see [20] or refer directly to the source code [2].

The basic design is faithfully depicted by Algorithm 2.1. Only plain LDL*-factorizations are used as representations at nodes.

Concerning selection of shifts in step 12, **DSTEMR** refines the eigenvalues at the borders of clusters to full precision and then tries to shift close to one of them, but still outside, backing off if the first tries are not successful. To evaluate a shift, **DSTEMR** only checks for element growth, that is, condition (2.81) above, with $C_{\text{elg}} = 8$. If no candidate satisfies the condition, **DSTEMR** takes the one with minimal element growth. Envelope information is obtained in a very minimal fashion: for a shift on the left, **DSTEMR** uses a rudimentary approximation to the “rightmost” eigenvector of the cluster as envelope, and vice versa. This approach is very efficient, but runs the danger of using wrong envelope information. To alleviate this concern, it is only employed if the plain element growth (unweighted) still lies below a safeguard threshold.

Once a shift has been selected and the corresponding representation is computed, **DSTEMR** will initialize the child’s eigenvalue bounds. Let $[\lambda_i]$ be the bounds at the father, then

$$[\lambda_i^+] := [\lambda_i] - \tau$$

become the bounds for the child. The first order of business before classification starts in line 5 for the child is then to inflate the child bounds until they have consistent Sturm counts.

Outline of our implementation **XMR**

The basic principle guiding the design of our implementation **XMR** was reliability. One cannot completely avoid situations where no shift candidate is acceptable a priori, in particular if no useful envelope is available. Furthermore we made the experience that the sole testing of element growth, like **DSTEMR** does it, can lead you astray: for some (admittedly rare, synthetic) test cases we encountered two candidates with comparable, acceptable element growth, but manual experimentation revealed only one of them to be relatively robust.

Hence we want to accumulate as much information about the candidates as possible, to make a better-informed choice. This is why we included the a priori checking of condition (2.82) for relative robustness.

For the same reason we restructured Algorithm 2.1 a little, by incorporating some bisection steps directly into step 12. Once a shift is deemed worthy of selection because it seems best—either passing the a priori tests or failing them less badly than the other candidates—the shifted representation is computed and initial bounds for its local eigenvalues are set up. Let $[\lambda_i], i \in I$ be the local approximations (intervals) at the father. Then we initialize the bounds for the child as

$$[\lambda_i^+] := ([\lambda_i](1 \pm \alpha) - \tau)(1 \pm \alpha), \quad (2.83)$$

where α is a parameter (currently $\alpha = 100n\epsilon_\diamond$). In our experience, one of the first

signs that something is wrong is that the eigenvalues start to jump out of even these inflated intervals. Hence the bounds are then “verified” to be consistent on the spot by doing two Sturm counts each. Should they prove to be inconsistent, the candidate is immediately discarded. In that case the Sturm counts could be seen as wasted, but they did prevent us from making a wrong choice. However, if the bounds pass basic consistency, we can save the (costly) inflation of the child bounds that DSTEMR has to do. We are then also in a position to use just two more Sturm counts to “verify” that the outer relative gaps at the child exceed $gaptol$, thus establishing that part of RELGAPS.

Another major new ingredient in XMR is that we exploit cluster substructure. As motivation consider for example a cluster of twenty eigenvalues with

$$\lambda_1, \dots, \lambda_{10} \doteq 1 + \mathcal{O}(\epsilon_\diamond), \quad \lambda_{11}, \dots, \lambda_{20} \doteq 1 + gaptol/2 + \mathcal{O}(\epsilon_\diamond).$$

We have to treat all twenty as one cluster, although we would like to separate the two subgroups. Considering envelopes, the condition (2.80) indicates that both subgroups may have significant eigenvector localization, and thus the whole cluster may have, too, although (2.80) considered solely for the whole cluster would make us think otherwise.

Our general strategy is to refine eigenvalues in clusters to reveal interior gaps exceeding $\sqrt{\epsilon_\diamond}$. Then we try to find envelopes for each subgroup and combine them to form an envelope for the whole cluster. To construct the envelopes for the subgroups we have implemented the new techniques from [65]. The whole approach works nicely so far and gives us usable and faithful envelope information for many cases where DSTEMR cannot use any.

Once cluster substructure has been determined, it is just too tempting not to exploit it further by also trying to shift inside the cluster, and this is what we do. It was feared before that doing so runs more risk of coming too close to Ritz values, thereby causing element growth. And indeed, just blindly taking the cluster midpoint is most often a fatal decision (we tried). However, to our experience inside shifts can work very well, *as long as* one makes sure to keep the representations nearly singular, that is, still places the shift within an interior gap, close to a boundary. In the example above we would try to shift slightly above λ_{10} or slightly below λ_{11} , additionally to trying shifts on the outside.

Further significant differences to DSTEMR are the following:

- We use e -representations of twisted factorizations at the nodes. The twist index is chosen on a case-by-case basis to minimize the bounds for both conditions (2.81) and (2.82) simultaneously.
- The tolerances for RQI and bisection were initially taken from DSTEMR and then optimized. In particular for singletons we switch from bisection to RQI far earlier to benefit from the faster convergence. Also the acceptance thresholds in RQI are rather strict in DSTEMR, causing the iteration to continue (or worse: fallback bisection to be invoked) needlessly sometimes.

- We use a different bisection design for **XMR**. The bisection process keeps a list of nonoverlapping intervals that contain one or more eigenvalues each. For instance, the child bounds in (2.83) are setup using the bounds in that list, and not for each eigenvalue separately. In contrast, **DSTEMR** uses a simpler scheme, with one interval per eigenvalue. This can be rather wasteful if the eigenvalues are close together, since overlapping intervals are refined independently.

Finally we want to mention that we have discovered rare cases where as yet unregarded *underflow* in the qd-transformations leads to loss of relative accuracy.

Our code is now completely underflow-safe, whereas **DSTEMR** still has the problem. Indeed we have at least one test case where we are quite sure that exactly this underflow-issue causes **DSTEMR** to return a zero vector⁴, without signaling an error code. However, at the moment it seems that introducing the necessary changes into an optimized code would either require IEEE 754-conform handling of exceptions, or a complete rethinking of the *pivmin*-scheme from §2.4.5.

We omit further details here, since these problems occurred only recently and warrant further study.

The Testsets

We have prepared two testsets **Pract** and **Synth** that consist of *upper bidiagonal* matrices **B**. They will also be used in Chapters 3 and 4. For the purpose at hand we will compare **DSTEMR** and **XMR** on the tridiagonal problems **B*B**.

Most of the bidiagonal matrices in the sets were obtained from tridiagonal problems **T** in two steps:

- (1) **T** was scaled and split according to (2.3).
- (2) For each unreduced subproblem we chose a shift to allow a Cholesky decomposition, yielding an upper bidiagonal matrix.

The composition of the two testsets **Pract** and **Synth** is as follows.

Pract contains 75 bidiagonal matrices with dimensions up to 6245. They were obtained in the manner above from tridiagonal matrices from various applications, which we got from Osni Marques and Christof Vömel. More information about the specific matrices can be found in [13], where the same set was used to evaluate the symmetric eigensolvers in LAPACK.

The testset **Synth** contains 19240 bidiagonal matrices that stem from artificially generated tridiagonal problems. The latter include standard types like Wilkinson matrices as well as matrices with eigenvalue distributions built into

⁴ The loss of relative accuracy leads to a twist that is completely wrong and in turn to a vector of huge norm that overflows.

LAPACK's test matrix generator DLATMS. In fact, all artificial types listed in [14] are present.

For each of these basic types, all tridiagonal matrices up to dimension 100 were generated. Then those were split according to step (1) above. For the resulting tridiagonal subproblems we made two further versions by glueing [19, 66] them to themselves: two copies with a small glue of $\gtrsim \|T\|n\epsilon_\diamond$; three copies with two medium glues of $\mathcal{O}(\|T\|n\sqrt{\epsilon_\diamond})$. Finally, step (2) above was used to obtain bidiagonal factors of all unreduced tridiagonal matrices.

Further additions to **Synth** include some special bidiagonal matrices **B** that were originally devised by Benedikt Großer. These were glued as well. However, special care was taken that step (1) above would not affect the matrix $\mathbf{B}^*\mathbf{B}$ for any one of these extra additions.

The nature of the set **Pract** is that we expect it to represent the kinds of problems that could be expected in actual applications. Therefore we deliberately chose not to include any glued matrices in **Pract**. This does not mean the problems in there are easy to solve, far from it. Nevertheless we will see that the really “hard” cases are in **Synth**, but due to its comprehensive nature, **Synth** also contains many cases that are borderline trivial.

A final remark concerning the possible danger of testset-bias. We did in fact not use the above testsets for testing and tuning of our software; at least the synthetic matrices we used then were fewer and generated with a different random seed.

Setup of the Tests

There is another reason besides reusability in later chapters that led us to consider bidiagonal matrices as the principal test problems. We want a fair comparison of DSTEMR and XMR, which is only possible if both solve the same problem. This may sound easier than it is. First of all, the gap tolerance is fixed to $gaptol = 0.001$ for XMR just as DSTEMR does it. Furthermore we desire both to start with the same root representation, namely $\mathbf{B}^*\mathbf{B}$.

To force the root representation to be $\mathbf{B}^*\mathbf{B}$, we do not call DSTEMR directly, because it expects the entries of a tridiagonal matrix. Instead we invoke its subroutine DLARRV which contains the core MR³-algorithm and is called by DSTEMR after preparatory steps have been done. Upon call, DLARRV expects to be given two things:

- An LDL*-factorization to take as root representation. We can write $\mathbf{B}^*\mathbf{B} = \mathbf{L}_0\mathbf{D}_0\mathbf{L}_0^*$ and provide one.
- Initial approximations for the eigenvalues. If all eigenpairs are desired, DSTEMR would compute them using dqds, by calling the routine DLASQ2 in LAPACK. We do just that. Identical values are initially fed to XMR as well.

Pract 75 cases		ORTH	Synth 19240 cases	
DSTEMR	XMR		DSTEMR	XMR
13.47	5.28	AVG	1001	3.09
1.98	1.78	MED	0.97	0.91
300	91	MAX	965000	608
82.67 %	89.33 %	0 ... 10	90.43 %	94.39 %
13.33 %	10.67 %	10 ... 100	7.38 %	5.43 %
2.67 %		100 ... 200	0.38 %	0.12 %
1.33 %		200 ... 500	0.29 %	0.05 %
		500 ... 10 ³	0.25 %	0.01 %
		10 ³ ... 10 ⁶	1.04 %	
		> 10 ⁶	0.23 %	

Table 2.6: Orthogonality levels $|Q^*Q - I|$ of DSTEMR compared to XMR, as multiples of $n\epsilon_\diamond$. The results were capped at 10^6 , meaning the .23% from Synth exceeding that for DSTEMR were not included in the upper statistics.

Recall that we took deliberate care to ensure that all the bidiagonal matrices B in the testsets have in common that the corresponding tridiagonal problems B^*B are already properly preprocessed. Hence we can state that DSTEMR, hypothetically confronted with the exact B^*B , would not change the matrix anymore but just shift it to get a positive definite root, which might well be B . Thus DLARRV is called within the limits it was designed for.

There is one additional step DSTEMR applies before calling DLARRV: it perturbs the data of the root representation by small random amounts of $\mathcal{O}(\epsilon_\diamond)$. This technique was shown in [20] to be an effective countermeasure against too deep trees for some kinds of glued matrices. Again we mimic the behavior of DSTEMR by perturbing the entries of B before passing them to XMR or DLARRV; this does not change the fact that both will start with exactly the same root representation. Of course we perturb the root representation *before* computing the initial eigenvalue approximations.

Numerical Results

Now come the actual results of the tests. Tables 2.6 and 2.7 depict the orthogonality levels and residual norms of both methods XMR and DSTEMR on both testsets. A first impression is that both handle the problems in Pract well. It becomes evident that the hard cases are indeed in Synth, and here is also where our efforts to improve reliability for XMR show effect. There are not too few cases in there for which DSTEMR does not produce satisfactory results.

Pract 75 cases		RESID	Synth 19240 cases	
DSTEMR	XMR		DSTEMR	XMR
0.18	0.21	AVG	1.01	0.37
0.02	0.05	MED	0.03	0.09
1.54	3.10	MAX	5228	56.5
96.00 %	97.33 %	0 ... 1	98.79 %	88.24 %
4.00 %	2.67 %	1 ... 10	1.16 %	11.75 %
		10 ... 100		0.01 %
		> 100	0.05 %	

Table 2.7: Residual norms $\|B^*Bq - q\lambda\|$ of DSTEMR compared to XMR, as multiples of $\|B^*B\|_{n\epsilon_\diamond}$.

The natural question is if we did buy the increased robustness of XMR with decreased efficiency. To answer that question we have compiled detailed profiling data in Table 2.8.

MR³-based methods spend virtually their whole execution time doing one of four kinds of factorizations:

- (A) compute Sturm counts for bisection,
- (B) compute twisted factorizations during the RQI for singletons,
- (C) compute shifted representations.

There is a fourth kind, namely further twisted factorizations to compute envelope information, but those really apply only to XMR.

The first three rows of Table 2.8 contain information about the number of steps for each kind. To obtain these numbers for DSTEMR we took the public, official code from [2] and instrumented it manually. For each test problem, the number of executed steps was determined and divided by n , to get an average number of steps per computed eigenvalue. Then we accumulated average, median and maximum information of these results over the whole testset. For instance, the table conveys that for any case in Pract, DSTEMR can be expected to take about 14.03 bisection steps per eigenvalue, and not more than 48.56. Note that these counts do not include the initial refinement of root eigenvalues, since we mentioned above that those are computed beforehand to full precision using dqds.

Now that we know what the data in the first three rows is supposed to mean, we see that our implementation XMR achieves its better accuracy doing in fact *less* work. In particular XMR requires significantly fewer bisection steps, due to the redesigned bisection scheme and the earlier switch to RQI. But the number of RQI steps is only minimally larger on average, and even lower in the worst

case, which comes from the optimized tolerances. Clearly **XMR** tries more shift candidates—in fact nearly twice as many as **DSTEMR** does. But the overhead is minimal considering the whole amount of tries for shifts in relation to bisection and RQI steps.

At this point we would love to give timing results, but since our code has not been optimized yet we cannot do that. Instead we propose an abstract cost measure to get a feeling for the performance. We took the accumulated numbers of steps for each case and weighted them according to the relative cost of each step, where a single bisection has the weight one. The remaining weights we estimated by considering mainly the divisions involved, so the number of shift tries in **DSTEMR** was weighted by about two, and RQI steps as well as the shift tries in **XMR** were weighted by a little more than three. Looking at the data in the central row, we would therefore deduce that for any problem in **Pract**, **XMR** has an expected overall runtime equivalent to doing 16.96 bisection steps per eigenvalue. Note that the stated cost for **XMR** seems too high compared to the counts in the upper three rows. This is because the additional factorizations for envelope computations were of course included in the overall cost estimate, but are not represented in the counts.

These are estimates at best, so the stated costs are to be taken in spirit only. Nevertheless we believe they will prove to be quite faithful indicators for the real situation.

Assuming for now the cost-measure is not completely off, we thus can expect an optimized version of **XMR** to be about 30% faster than **DSTEMR** across the board. The bottom three rows apportion the costs to the three major subtasks of dealing with singleton eigenvalues, bisection for classification and refining clusters, and finding shifts. This data clearly conveys that we reinvested some of the time that was saved from bisection and RQI into finding better shift candidates.

	Pract			Synth	
	DSTEMR	XMR		DSTEMR	XMR
<i># bisection steps</i>	14.03 <i>n</i>	7.28 <i>n</i>	AVG	24.70 <i>n</i>	14.50 <i>n</i>
	7.38 <i>n</i>	3.90 <i>n</i>	MED	33.05 <i>n</i>	17.17 <i>n</i>
	48.56 <i>n</i>	25.13 <i>n</i>	MAX	65.57 <i>n</i>	42.00 <i>n</i>
<i># RQI steps (no fallback bisection)</i>	1.92 <i>n</i>	2.09 <i>n</i>	AVG	1.89 <i>n</i>	2.17 <i>n</i>
	1.87 <i>n</i>	1.99 <i>n</i>	MED	2.00 <i>n</i>	2.16 <i>n</i>
	4.36 <i>n</i>	3.01 <i>n</i>	MAX	3.87 <i>n</i>	3.78 <i>n</i>
<i># tries for shifts</i>	0.25 <i>n</i>	0.55 <i>n</i>	AVG	0.32 <i>n</i>	0.78 <i>n</i>
	0.22 <i>n</i>	0.46 <i>n</i>	MED	0.33 <i>n</i>	0.67 <i>n</i>
	1.05 <i>n</i>	2.33 <i>n</i>	MAX	1.97 <i>n</i>	2.41 <i>n</i>
<i>estimated cost (as plain Sturm counts)</i>	21.37 <i>n</i>	16.96 <i>n</i>	AVG	31.21 <i>n</i>	25.57 <i>n</i>
	17.43 <i>n</i>	13.75 <i>n</i>	MED	40.62 <i>n</i>	28.95 <i>n</i>
	63.26 <i>n</i>	44.92 <i>n</i>	MAX	81.23 <i>n</i>	58.59 <i>n</i>
<i>... bisection to classify & refine clusters</i>	42%	30%	AVG	59%	43%
	47%	29%	MED	80%	58%
	85%	60%	MAX	91%	75%
<i>... to refine singleton eigs & compute vectors</i>	57%	53%	AVG	40%	43%
	52%	44%	MED	19%	29%
	100%	100%	MAX	100%	100%
<i>... to determine en- velopes & find shifts</i>	1%	17%	AVG	1%	14%
	1%	14%	MED	1%	15%
	5%	48%	MAX	6%	49%

Table 2.8: Efficiency of DSTEMR compared to XMR.

	Pract			Synth	
	DSTEMR	XMR		DSTEMR	XMR
<i>Number of nodes in the tree</i>	126	132	AVG	24	25
	28	28	MED	5	5
	1176	1313	MAX	175	161
<i>Depth of tree</i>	1.47	1.46	AVG	1.3	1.26
	1	1	MED	1	1
	4	4	MAX	14	5
<i>Maximal # of RQI steps for any single eigenpair</i>	6.93	3.44	AVG	3.98	2.94
	7	3	MED	3	3
	12	7	MAX	12	8
<i>Maximal final sine- bound $\gamma /(\ \bar{q}\ \text{gap})$ for a computed vector</i>	18.06 <i>nε_◊</i>	10.3 <i>nε_◊</i>	AVG	2.2 <i>nε_◊</i>	10 <i>nε_◊</i>
	1.96 <i>nε_◊</i>	4.2 <i>nε_◊</i>	MED	1.2 <i>nε_◊</i>	3 <i>nε_◊</i>
	408 <i>nε_◊</i>	98.0 <i>nε_◊</i>	MAX	765 <i>nε_◊</i>	583 <i>nε_◊</i>

Table 2.9: Some additional statistics for comparison of DSTEMR and XMR.

Chapter 3

MR³ for the Bidiagonal SVD

The singular values σ_i of J are known [...] to be related to the eigenvalues of the $2n \times 2n$ matrix $\tilde{J} = [\dots]$, whose eigenvalues are just $+\sigma_i$ and $-\sigma_i$ for $i = 1, 2, \dots, n$.

The calculation of the eigenvalues of \tilde{J} is simplified conceptually by a transformation to tridiagonal form via a permutation similarity which will be exhibited below.

— GENE GOLUB and WILLIAM KAHAN,
Calculating the Singular Values and Pseudo-Inverse of a Matrix (1965)

The Singular Value Decomposition (SVD) must be regarded as (one of) the most fundamental and powerful decompositions a numerical analyst has at his or her disposal. This is partly due to generality, since every complex rectangular matrix has an SVD, but also to versatility, because most problems just dissipate once the SVD of a certain related matrix can be computed. Applications range from pure theory to image processing.

The principal algorithm for computing the SVD of an arbitrary dense complex rectangular matrix is reduction to real bidiagonal form using unitary similarity transformations, followed by computing the SVD of the obtained bidiagonal matrix. The method to do the reduction was pioneered by Golub and Kahan [33,34]. Later improvements include reorganization to do most of the work within BLAS3-calls [5, 6, 50].

We call the problem to compute the singular value decomposition of a bidiagonal matrix BSVD. There is a long tradition to solve singular value problems by casting them into related symmetric eigenproblems. For BSVD this leads to a variety of symmetric tridiagonal eigenproblems. It is then natural and tempting to solve these using the MR³ algorithm, to benefit from its many desirable features. How to do so stably and efficiently is the focus of this chapter.

At least in passing we would like to mention that an alternative and highly competitive solution strategy for the SVD was only recently discovered by Drmač

and Veselić [21, 22]. They revived the ancient Jacobi method and found a way to overcome its inherent stability issues, thus skipping bidiagonal reduction altogether.

Outline. In §3.1 we specify the problem to be solved formally, introduce the associated tridiagonal problems, and set up some notational adjustments. The focus of §3.2 is using MR³ to solve the eigenproblems associated with the *normal equations*, and we study the arising problems in detail. Invoking MR³ on symmetric tridiagonal matrices of even dimension that have a zero diagonal, so-called *Golub-Kahan matrices*, will then be investigated in §3.3. The remaining two sections are devoted to the *coupled approach* for the problem BSVD, which consists of using MR³ on the normal equations and the Golub-Kahan matrix simultaneously. The all-important *coupling relations* that make this possible are derived in §3.4, after which §3.5 contains the presentation and study of the resulting algorithm MCR³. Finally, §3.6 contains numerical experiments to evaluate our implementations.

The material in this chapter is not entirely new, but builds on pioneering work by Benedikt Groß and Bruno Lang [35–37]. We regard our main contributions to be the following:

- A thorough theoretical study to explain the problems with MR³ on the normal equations. Existing explanations before were mostly experimental in nature.
- For a long time the standing opinion was that using MR³ (or any other TSEP-solver) on the Golub-Kahan matrix was fundamentally flawed. We will refute that notion, at least with regard to MR³. Indeed we will provide a complete proof, including error bounds, showing that just a minor modification makes using MR³ on the Golub-Kahan matrix a valid solution strategy for BSVD.
- The coupled approach has been significantly revised and simplified since its original conception. Some of this work has been published in [72]. Furthermore we can now provide a rigorous proof of correctness and give concise error bounds.
- The coupling relations are presented in more detail, with a new proof. We exhibit the crucial dependence on an ominous property called a *nearly constant diagonal* and also discuss how to couple arbitrary twisted factorizations stably in practice.
- We stumbled upon a subtle error in the proof of Theorem 5.2 in [37]. So far we were not able to repair it, but the revised coupled approach lets us skirt having to use the claim for now.

3.1 Basics

3.1.1 The Problem

Throughout this chapter we will keep one particular upper bidiagonal matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ fixed and in focus. We name its diagonal entries a_i and its offdiagonal elements b_i , so that

$$\mathbf{B} = \text{diag}(a_1, \dots, a_n) + \text{diag}_{+1}(b_1, \dots, b_{n-1}). \quad (3.1)$$

The goal is to compute the full *singular value decomposition*

$$\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}^*, \quad (3.2)$$

where $\mathbf{U}^*\mathbf{U} = \mathbf{V}^*\mathbf{V} = \mathbf{I}$ and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$ with $\sigma_1 \leq \dots \leq \sigma_n$.

The columns $\mathbf{u}_i = \mathbf{U}(:, i)$ and $\mathbf{v}_i = \mathbf{V}(:, i)$ are called *left* and *right singular vectors*, respectively, and the σ_i 's are the *singular values*. Note that we do break with the convention to order the singular values descendingly; the justification is to simplify the transition between BSVD and TSEP and will become apparent soon. Taken together, $(\sigma_i, \mathbf{u}_i, \mathbf{v}_i)$ form a *singular triplet* of \mathbf{B} .

Similarly to TSEP, we have concrete requirements in mind that any algorithm to solve BSVD has to meet. The computed singular triplets $(\bar{\sigma}_i, \bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i)$ should enjoy *numerical orthogonality* in the sense

$$\max \{ |\mathbf{U}^*\mathbf{U} - \mathbf{I}|, |\mathbf{V}^*\mathbf{V} - \mathbf{I}| \} = \mathcal{O}(n\epsilon_\diamond), \quad (3.3)$$

where $|\cdot|$ is meant componentwise. Furthermore we desire small *residual norms*,

$$\max_i \{ \|\mathbf{B}\bar{\mathbf{v}}_i - \bar{\mathbf{u}}_i\bar{\sigma}_i\|, \|\mathbf{B}^*\bar{\mathbf{u}}_i - \bar{\mathbf{v}}_i\bar{\sigma}_i\| \} = \mathcal{O}(\|\mathbf{B}\|n\epsilon_\diamond). \quad (3.4)$$

In the literature the latter is sometimes stated as the singular vector pairs being “(well) coupled”. We avoid this phrase as the term *coupled* will be embodied with a related, but subtly different meaning later in this chapter.

Singular Values to high relative accuracy

The monumental paper [12] established that every bidiagonal matrix (represented by entries) determines its singular values to high relative accuracy.

At the moment, the state-of-the-art for computing singular values is the dqds-algorithm by Fernando and Parlett [30, 64], which builds upon [12] as well as Rutishauers’s original qd-algorithm [67]. An excellent implementation of dqds is included in LAPACK in the form of routine `xLASQ1`. Alternatively, bisection could be used, but this is normally much slower—in our experience it becomes worthwhile to use bisection instead of dqds only if less than ten percent of the singular values are desired (dqds can only compute all singular values at once).

The condition (3.4) alone does merely convey that each computed $\bar{\sigma}_i$ must lie within distance $\mathcal{O}(\|\mathbf{B}\|n\epsilon_\diamond)$ of *some* exact singular value of \mathbf{B} . A careful but elementary argument based on the Gap Theorem 1.21 (applied to the Golub-Kahan matrix, see below) shows that (3.3) and (3.4) combined actually provide for *absolute accuracy* in the singular values, meaning each computed $\bar{\sigma}_i$ lies within distance $\mathcal{O}(\|\mathbf{B}\|n\epsilon_\diamond)$ of the exact σ_i . To achieve *relative accuracy*, a straightforward modification is just to recompute the singular values afterwards using, for example, dqds. It is clear that doing so cannot spoil (3.4), at least as long as $\bar{\sigma}_i$ was computed with absolute accuracy.

The recomputation does not even necessarily be overhead; for MR³-type algorithms like those we study in this chapter one needs initial approximations to the singular values anyway, the more accurate the better, so there is actually a gain from computing them up front to full precision.

3.1.2 Associated Tridiagonal Problems

There are two standard approaches to reduce the problem BSVD to TSEP, involving three different symmetric tridiagonal matrices.

The Normal Equations

From (3.2) we can see the eigendecompositions of the symmetric tridiagonal matrices $\mathbf{B}\mathbf{B}^*$ and $\mathbf{B}^*\mathbf{B}$ to be

$$\mathbf{B}\mathbf{B}^* = \mathbf{U}\Sigma^2\mathbf{U}^*, \quad \mathbf{B}^*\mathbf{B} = \mathbf{V}\Sigma^2\mathbf{V}^*. \quad (3.5)$$

These two are called *normal equations*, in deference to the problem of linear least squares. It will be useful to know how the individual entries of $\mathbf{B}\mathbf{B}^*$ and $\mathbf{B}^*\mathbf{B}$ can be expressed using those of \mathbf{B} :

$$\begin{aligned} \mathbf{B}\mathbf{B}^* &= \text{diag}(a_1^2 + b_1^2, \dots, a_{n-1}^2 + b_{n-1}^2, a_n^2) \\ &\quad + \text{diag}_{\pm 1}(a_2 b_1, \dots, a_n b_{n-1}), \\ \mathbf{B}^*\mathbf{B} &= \text{diag}(a_1^2, a_2^2 + b_1^2, \dots, a_n^2 + b_{n-1}^2) \\ &\quad + \text{diag}_{\pm 1}(a_1 b_1, \dots, a_{n-1} b_{n-1}). \end{aligned} \quad (3.6)$$

The Golub-Kahan Matrix

Given a bidiagonal matrix \mathbf{B} we can go to a symmetric eigenproblem of twice the size by forming

$$\begin{bmatrix} 0 & \mathbf{B} \\ \mathbf{B}^* & 0 \end{bmatrix}, \quad (3.7)$$

which is sometimes called the *Jordan-Wielandt form* [28, 47, 68]. If \mathbf{B} has the singular value decomposition $\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}^*$, then $\mathbf{B}^* = \mathbf{V}\Sigma\mathbf{U}^*$, leading to the eigen-decomposition

$$\begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^* & \mathbf{0} \end{bmatrix} = \mathbf{J} \begin{bmatrix} -\Sigma & \mathbf{0} \\ \mathbf{0} & \Sigma \end{bmatrix} \mathbf{J}^* \quad \text{with} \quad \mathbf{J} = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{U} & \mathbf{U} \\ -\mathbf{V} & \mathbf{V} \end{bmatrix}. \quad (3.8)$$

Note that in this formulation the negative eigenvalues are ordered descendingly. If we assume the σ_i are simple, \mathbf{J} is unique up to reordering and negating columns.

The crucial idea first formulated in the seminal paper [33] is that the Jordan-Wielandt form can be permuted to become tridiagonal. To this end define \mathbf{P}_{ps} to be the permutation matrix on \mathbb{R}^{2n} which maps any $\mathbf{x} \in \mathbb{R}^{2n}$ to

$$\mathbf{P}_{\text{ps}}\mathbf{x} = [\mathbf{x}(n+1), \mathbf{x}(1), \mathbf{x}(n+2), \mathbf{x}(2), \dots, \mathbf{x}(2n), \mathbf{x}(n)]^*, \quad (3.9)$$

or, equivalently stated,

$$\mathbf{P}_{\text{ps}}^*\mathbf{x} = [\mathbf{x}(2), \mathbf{x}(4), \dots, \mathbf{x}(2n), \mathbf{x}(1), \mathbf{x}(3), \dots, \mathbf{x}(2n-1)]^*. \quad (3.10)$$

The expression (3.9) motivates why \mathbf{P}_{ps} is called a *perfect shuffle* permutation. Symmetric application to the rows and columns of the Jordan-Wielandt form yields

$$\mathbf{T}_{\text{GK}}(\mathbf{B}) := \mathbf{P}_{\text{ps}} \begin{bmatrix} \mathbf{0} & \mathbf{B} \\ \mathbf{B}^* & \mathbf{0} \end{bmatrix} \mathbf{P}_{\text{ps}}^*, \quad (3.11)$$

the *Golub-Kahan (GK) matrix* or *Golub-Kahan form* of \mathbf{B} . As far as we know this name was coined by Fernando in [28]. A short calculation verifies that $\mathbf{T}_{\text{GK}}(\mathbf{B})$ is indeed tridiagonal with a zero diagonal and the entries of \mathbf{B} interleaved on the offdiagonals, that is,

$$\mathbf{T}_{\text{GK}}(\mathbf{B}) = \text{diag}_{\pm 1}(a_1, b_1, a_2, b_2, \dots, a_{n-1}, b_{n-1}, a_n).$$

From (3.8) we then see that the relationship of the eigenpairs of $\mathbf{T}_{\text{GK}}(\mathbf{B})$ and the singular triplets of \mathbf{B} is:

$$\begin{aligned} (\sigma, \mathbf{u}, \mathbf{v}) \text{ is a singular triplet of } \mathbf{B} \text{ with } \|\mathbf{u}\| = \|\mathbf{v}\| = 1. \\ \iff \\ (\pm\sigma, \mathbf{q}) \text{ are eigenpairs of } \mathbf{T}_{\text{GK}}(\mathbf{B}), \text{ where } \|\mathbf{q}\| = 1, \sqrt{2}\mathbf{q} = \mathbf{P}_{\text{ps}} \begin{bmatrix} \mathbf{u} \\ \pm\mathbf{v} \end{bmatrix}. \end{aligned} \quad (3.12)$$

The interplay between the vectors \mathbf{u} , \mathbf{v} and \mathbf{q} is that \mathbf{v} makes up the odd entries in \mathbf{q} and \mathbf{u} the even ones:

$$\sqrt{2}\mathbf{q}^* = [\mathbf{v}(1), \mathbf{u}(1), \mathbf{v}(2), \mathbf{u}(2), \dots, \mathbf{v}(n), \mathbf{u}(n)]. \quad (3.13)$$

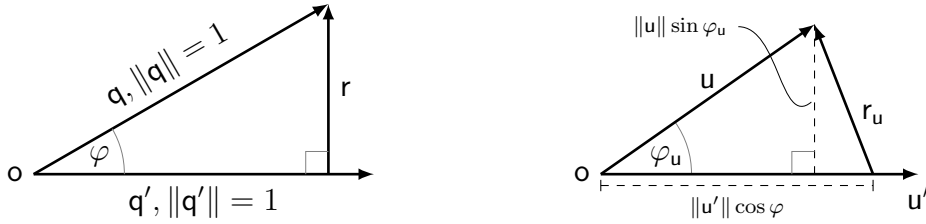


Figure 3.1: Situation for the proof of Lemma 3.1. The global setting is on the left, the right side zooms in just on the u -components. Note that in general $\varphi_u \neq \varphi$ and r_u will not be orthogonal to u , nor to u' .

It will frequently be necessary to relate rotations of GK-eigenvectors \mathbf{q} to rotations of their u - and v -components. This is actually rather straightforward and captured in the following lemma. The formulation has been kept fairly general; in particular the permutation P_{ps} is left out, but the claim does extend naturally if it is reintroduced.

Lemma 3.1. *Let \mathbf{q}, \mathbf{q}' be non-orthogonal unit vectors that admit a conforming partition*

$$\mathbf{q} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}, \quad \mathbf{q}' = \begin{bmatrix} \mathbf{u}' \\ \mathbf{v}' \end{bmatrix}, \quad \mathbf{u}, \mathbf{v} \neq \mathbf{o}.$$

Let $\varphi_u := \angle(\mathbf{u}, \mathbf{u}')$, $\varphi_v := \angle(\mathbf{v}, \mathbf{v}')$ and $\varphi := \angle(\mathbf{q}, \mathbf{q}')$. Then

$$\begin{aligned} \max \left\{ \|\mathbf{u}\| \sin \varphi_u, \|\mathbf{v}\| \sin \varphi_v \right\} &\leq \sin \varphi, \\ \max \left\{ \left| \|\mathbf{u}'\| - \|\mathbf{u}\| \right|, \left| \|\mathbf{v}'\| - \|\mathbf{v}\| \right| \right\} &\leq \frac{\sin \varphi + (1 - \cos \varphi)}{\cos \varphi}. \end{aligned}$$

Proof. Define r such that

$$\mathbf{q} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \mathbf{q}' \cos \varphi + \mathbf{r} = \begin{bmatrix} \mathbf{u}' \cos \varphi + \mathbf{r}_u \\ \mathbf{v}' \cos \varphi + \mathbf{r}_v \end{bmatrix}.$$

The resulting situation is depicted in Figure 3.1. Consequently,

$$\|\mathbf{u}\| \sin \varphi_u \leq \|\mathbf{r}_u\| \leq \|\mathbf{r}\| = \sin \varphi.$$

Now $\mathbf{u}' \cos \varphi = \mathbf{u} - \mathbf{r}_u$ implies $(\mathbf{u}' - \mathbf{u}) \cos \varphi = (1 - \cos \varphi)\mathbf{u} - \mathbf{r}_u$. Use the reverse triangle inequality, $\|\mathbf{u}\| < 1$ and $\cos \varphi \neq 0$ to obtain the desired relationship for $\|\mathbf{u}'\|$. The claims pertaining to the v -components are shown analogously. \square

Application to a given approximation \mathbf{q}' for an exact GK-eigenvector \mathbf{q} merely requires to exploit $\|\mathbf{u}\| = \|\mathbf{v}\| = 1/\sqrt{2}$. In particular, the second claim of Lemma 3.1 will then enable us to control how much the norms of \mathbf{u}' and \mathbf{v}' can

deviate from $1/\sqrt{2}$, namely basically by no more than $\sin \varphi + \mathcal{O}(\sin^2 \varphi)$, provided φ is small.

A nice exploit of the connection between a bidiagonal \mathbf{B} and the associated $\mathsf{T}_{\text{GK}}(\mathbf{B})$ gives an elegant alternative proof for the fact that \mathbf{B} determines its SVD to high relative accuracy. In the context of Chapter 2 we can state this as saying that the offdiagonal entries of a symmetric tridiagonal matrix with zero diagonal form an RRR.

Here we only want to sketch the main idea for a proof. Lemma 2.20 showed that an entrywise perturbation $\mathbf{B} \rightsquigarrow \tilde{\mathbf{B}}$ to a bidiagonal matrix can be written in multiplicative form $\tilde{\mathbf{B}} = \mathbf{X}\mathbf{B}\mathbf{Y}$ with diagonal matrices \mathbf{X} and \mathbf{Y} . This transcends to an outer perturbation of the Golub-Kahan matrix since

$$\mathsf{T}_{\text{GK}}[\tilde{\mathbf{B}}] = \mathsf{P}_{\text{ps}} \begin{bmatrix} \mathbf{X} & 0 \\ 0 & \mathbf{Y} \end{bmatrix} \mathsf{P}_{\text{ps}}^* \mathsf{T}_{\text{GK}}(\mathbf{B}) \mathsf{P}_{\text{ps}} \begin{bmatrix} \mathbf{X} & 0 \\ 0 & \mathbf{Y} \end{bmatrix} \mathsf{P}_{\text{ps}}^*.$$

Then the perturbation theory from §1.4.3 can be put into play. Theorem 1.30 allows to bound the relative change in the singular values and Theorem 1.31 controls the rotation effected on the GK-eigenvectors. The latter can be translated to rotations of the singular vectors using Lemma 3.1 above, together with Lemma 3.9 which will come below.

Eisenstat & Ipsen [23] use basically the outlined approach to establish their comprehensive perturbation results for singular values and vectors. The original proof given by Demmel & Kahan [12] is of a more direct nature and gives in fact sharper bounds, but they execute it only for the singular values.

Black Box Approaches

Once the associated tridiagonal problems have been identified, essentially the problem BSVD seems to collapse, as it becomes reducible to TSEP. Two basic strategies come to mind.

First, we could employ algorithm MR^3 to compute eigendecompositions of $\mathbf{B}\mathbf{B}^*$ and $\mathbf{B}^*\mathbf{B}$ separately. This gives both left and right singular vectors as well as the singular values (twice). A slight variation on this theme would compute just the vectors on one side, for example $\mathbf{B}\mathbf{B}^* = \mathbf{U}\Sigma^2\mathbf{U}^*$, and then get the rest through solving $\mathbf{B}\mathbf{v} = \mathbf{u}\sigma$.

Alternatively we can compute the SVD via (3.12), extracting singular triplets from computed eigenpairs of $\mathsf{T}_{\text{GK}}(\mathbf{B})$. Here the ability of MR^3 to compute partial spectra comes in handy, as we need only concern ourselves with one half of the spectrum of $\mathsf{T}_{\text{GK}}(\mathbf{B})$.

These approaches have come to be called *black-box approaches*, as they are based on employing MR^3 as is, without need for any deeper modifications to its internals. Note that in both cases, MR^3 would offer to compute only a subset of singular triplets; current solution methods for BSVD like DC or QR do not provide this feature.

So, one could think the problem BSVD is solved. But, alas, the black-box approaches do all run into major numerical problems at some point. These were investigated and reported by Großer [35]. We will describe his findings and provide a deeper theoretical analysis in §3.2 and §3.3.

Couplings

The fundamental idea originated by Großer [35] was that the problems with the black-box approaches can be bypassed if one runs MR³ on the Golub-Kahan matrix and the normal equations *simultaneously*. What makes this possible is that translates of these three matrices are intimately related, or *coupled*, provided the shifts are compatible. This insight forms a cornerstone of this chapter and permeates §3.2—§3.5, although it will not fully enter the stage before §3.4, where we provide the detailed theoretical background. Right now we want to give just a short preview.

As eigenvectors are shift-invariant, the connection between eigenpairs of the normal equations and the Golub-Kahan matrix that was expressed in (3.5) and (3.12) remains intact after shifting. This is an insight worth capturing.

Definition 3.2 (Coupled Representations). Let \check{M} , \hat{M} and M be representations of symmetric tridiagonal matrices such that there is an upper bidiagonal matrix B and a scalar $\bar{\mu} \geq 0$ with

$$\check{M} = BB^* - \bar{\mu}^2, \quad M = T_{\text{GK}}(B) - \bar{\mu}, \quad \hat{M} = B^*B - \bar{\mu}^2.$$

Then \check{M} , \hat{M} and M are (*perfectly*) *coupled*. ◇

Since the diagonal of $T_{\text{GK}}(B)$ is zero, the diagonal of M equals $-\bar{\mu}$ everywhere, i.e., is constant. This aspect will become a repeating theme on the following pages.

The next definition relaxes the concept of coupled representations and casts it into a more MR³-conform setting, where eigenvalues and invariant subspaces are regarded locally.

Definition 3.3 (Coupled Eigensystems). Let $\check{M}, \hat{M} \in \mathbb{R}^{n \times n}$ and $M \in \mathbb{R}^{2n \times 2n}$ be representations of symmetric tridiagonal matrices such that, for all indices i from a set $I \subseteq \{1, \dots, n\}$, the following conditions hold:

(i) $\lambda_i[\check{M}] = \lambda_i[\hat{M}]$.

(ii) If \mathbf{u}_i and \mathbf{v}_i are i th eigenvectors of \check{M} and \hat{M} , respectively, then $\mathbf{P}_{\text{ps}} \begin{bmatrix} \mathbf{u}_i \\ \pm \mathbf{v}_i \end{bmatrix}$ are eigenvectors of M .

Then the local eigensystems I of \check{M}, \hat{M} and M are (*perfectly*) *coupled*. ◇

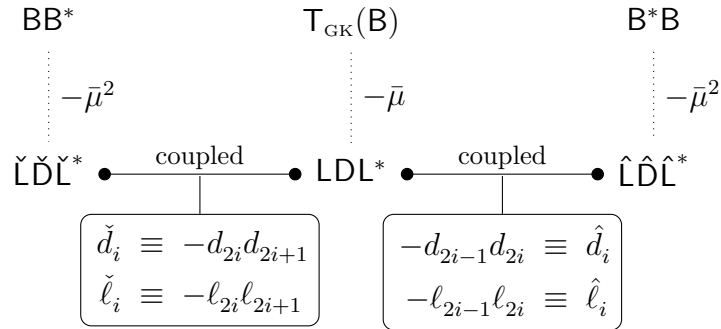


Figure 3.2: The coupling relations at work for standard LDL^* bidiagonal factorizations.

Obviously, coupled representations have complete coupled eigensystems. Note that we avoided using indices of \mathbf{M} explicitly, because they might be non-conform to those of $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$, due to the differing dimensions. We will introduce a suitable indexing convention further below.

The palpable feature of coupled representations is that their defining data elements are connected. If twisted factorizations are employed one can relate them in an elementary and stable way involving only multiplications and divisions, using the *coupling relations* that will be developed in §3.4. As appetizer, Figure 3.2 depicts how they work for plain top-to-bottom bidiagonal factorizations, cf. (3.24) and (3.25).

Thus, not only are the eigenvectors of an exact GK-translate composed of both the left and the right singular vectors, but the representation data itself carries the full information for the corresponding translates of the normal equations.

Notation

The preceding remarks about couplings already indicated that we will constantly be dealing with representations that are translates of one of our three roots \mathbf{BB}^* (“u-side”), $\mathbf{B}^*\mathbf{B}$ (“v-side”) or $\mathbf{T}_{\text{GK}}(\mathbf{B})$ (“central”). To streamline working within these three “layers” we will adjust our notation for this chapter somewhat.

As major step, we use consistent accents to differentiate the layers: translates of \mathbf{BB}^* and $\mathbf{B}^*\mathbf{B}$ will be denoted as $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$, respectively, and only descendants of $\mathbf{T}_{\text{GK}}(\mathbf{B})$ will be written as plain \mathbf{M} . These accents transcend to eigenvalues, thus $\check{\mathbf{M}}$ has eigenvalues $\check{\lambda}_i$ and $\hat{\mathbf{M}}$ has $\hat{\lambda}_i$. To indicate perturbed quantities, we replace $\check{\cdot}$ by $\check{\sim}$ and $\hat{\cdot}$ by $\hat{\sim}$.

There is no need to use separate entities for eigenvalues of $\mathbf{T}_{\text{GK}}(\mathbf{B})$, since those are just the singular values of \mathbf{B} . Because of this it is useful to retain their indexing: the Golub-Kahan Matrix $\mathbf{T}_{\text{GK}}(\mathbf{B})$ and all its shifted descendants will

The Three Layers

	\mathbf{BB}^*	$\mathbf{T}_{\text{GK}}(\mathbf{B})$	$\mathbf{B}^*\mathbf{B}$
representation	$\check{\mathbf{M}}$	\mathbf{M}	$\hat{\mathbf{M}}$
perturbed rep.	$\tilde{\mathbf{M}}$	$\tilde{\mathbf{M}}$	$\hat{\mathbf{M}}$
eigenvalues	$\check{\lambda}_i$	$\lambda_{\pm i}$	$\hat{\lambda}_i$
eigenvectors	\mathbf{u}_i	$\mathbf{q}_{\pm i}$	\mathbf{v}_i
inv. subspaces	\mathcal{U}_I	$\mathcal{Q}_{\pm I}$	\mathcal{V}_I

Table 3.1: Notational adjustments to handle representations associated with the three root matrices.

have their eigenvalues indexed from $-n$ to n (omitting 0), i.e.,

$$\lambda_{-i}[\mathbf{T}_{\text{GK}}(\mathbf{B})] \equiv -\sigma_i, \quad \lambda_i[\mathbf{T}_{\text{GK}}(\mathbf{B})] \equiv \sigma_i,$$

and naturally the same indexing applies to eigenvectors.

Concerning eigenvectors, we break with using \mathbf{q} and $\mathcal{Q}_I[\cdot]$ for everything but only continue to use them for the GK-layer. For the other two we reserve letters \mathbf{u}, \mathcal{U} to convey the same meaning on the \mathbf{BB}^* -side and \mathbf{v}, \mathcal{V} for the $\mathbf{B}^*\mathbf{B}$ -side. This has the benefit of being conform with the structural relation (3.13) and keeps the singular vector semantic, but note that we may use \mathbf{u}_i for some eigenvector of an $\check{\mathbf{M}}$ without \mathbf{u}_i being an exact left singular vector of our \mathbf{B} , since the computation of $\check{\mathbf{M}}$ might not have been exact.

A summary of the notational changes is given in Table 3.1.

3.1.3 Setting the stage

Just as was done for the tridiagonal problem in Chapter 2, it will be worthwhile to preprocess the given input matrix \mathbf{B} with regard to some points. For TSEP, it sufficed to deal with the offdiagonal elements, but now all entries of \mathbf{B} are involved with the offdiagonals of \mathbf{BB}^* , $\mathbf{B}^*\mathbf{B}$ and $\mathbf{T}_{\text{GK}}(\mathbf{B})$, which makes a sensible preprocessing a bit more difficult.

Should the input matrix be lower bidiagonal, work with \mathbf{B}^* instead and swap the roles of \mathbf{U} and \mathbf{V} . Multiplication on both sides by suitable diagonal signature matrices gets all entries nonnegative, and we can scale to get the largest elements into proper range. Then, in order to avoid a whole plethora of numerical problems later on, it is highly advisable to get rid of tiny entries by setting them to zero and splitting the problem. To summarize, we should strive for the analogon to (2.3), namely

$$\epsilon_{\circ} \|\mathbf{B}\| < \min\{a_i, b_i\}. \quad (3.14)$$

Note that this condition is equivalent to (2.3) for $\mathbf{T}_{\text{GK}}(\mathbf{B})$.

Splitting a bidiagonal matrix to attain (3.14) by setting all violating entries to zero is not that straightforward—there are two issues which must be addressed.

If an offdiagonal element b_i is zero, \mathbf{B} is reducible and can be partitioned into two smaller bidiagonal problems. If a diagonal element a_i is zero, then \mathbf{B} is singular. An elegant way to “deflate” one zero singular value is to apply one sweep of the implicit zero-shift QR method, which will yield a matrix \mathbf{B}' with $b'_{i-1} = b'_{n-1} = a'_n = 0$, cf. [12, p. 21]. Thus the zero singular value has been revealed and can now be removed by splitting into three upper bidiagonal parts $\mathbf{B}_{1:i-1}$, $\mathbf{B}_{i:n-1}$ and $\mathbf{B}_{n,n}$, the latter of which is trivial. An additional benefit of the QR-sweep is a possible preconditioning effect for the problem [35], but of course we will also have to rotate the computed vectors afterwards.

The second obstacle is that using (3.14) as criterion for setting entries to zero will impede computing the singular values to high relative accuracy with respect to the input matrix. There are splitting criteria which retain relative accuracy, for instance those employed within the zero-shift QR algorithm [12, p. 18] and the slightly stronger ones by Li [51, 64]. However, they do not naturally allow for less splitting than (3.14).

To get the best of both, that is, extensive splitting with all its benefits as well as relatively accurate singular values, we propose a 2-phase splitting as follows:

- 1) Split the matrix as much as possible *without* spoiling relative accuracy. This results in a partition

$$\mathbf{B}_{\text{rs}} = \begin{bmatrix} \mathbf{B}_{\text{rs}}^{(1)} & & \\ & \ddots & \\ & & \mathbf{B}_{\text{rs}}^{(N)} \end{bmatrix},$$

which we call the *relative split* of \mathbf{B} .

- 2) Split each block $\mathbf{B}_{\text{rs}}^{(i)}$ further aggressively to achieve (3.14), resulting in

$$\mathbf{B}_{\text{rs}}^{(i)} = \begin{bmatrix} \mathbf{B}_{\text{as}}^{(i,1)} & & \\ & \ddots & \\ & & \mathbf{B}_{\text{as}}^{(i,N_i)} \end{bmatrix}, \quad i = 1, \dots, N.$$

We denote the collection of subblocks $\mathbf{B}_{\text{as}}^{(i,j)}$ as *absolute split* of \mathbf{B} .

- 3) Solve BSVD for each block in the absolute split independently by invoking the desired MR³-based method.
- 4) Use bisection to refine the computed singular values of each block $\mathbf{B}_{\text{as}}^{(i,j)}$ to high relative accuracy with respect to the father-block $\mathbf{B}_{\text{rs}}^{(i)}$ in the relative split.

Since the singular values of the blocks in the absolute split retain absolute accuracy with respect to \mathbf{B} , the requirements (3.3) and (3.4) will still be upheld. In

ALGORITHM 3.1 *MR³ as black box on the normal equations*

Compute specified singular triplets of bidiagonal \mathbf{B} using the MR³ algorithm separately on $\mathbf{B}\mathbf{B}^*$ and $\mathbf{B}^*\mathbf{B}$.

Input: Upper bidiagonal $\mathbf{B} \in \mathbb{R}^{n \times n}$. Index set $I_0 \subseteq \{1, \dots, n\}$.

Output: Singular triplets $(\bar{\sigma}_i, \bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i), i \in I_0$.

- 1: Execute Alg. 2.1, but take $\mathbf{M}_0 := \mathbf{B}\mathbf{B}^*$ using the entries of \mathbf{B} directly as root representation in step 1.
This gives eigenpairs $(\check{\lambda}_i, \check{\mathbf{u}}_i), i \in I_0$.
- 2: Execute Alg. 2.1, but take $\mathbf{M}_0 := \mathbf{B}^*\mathbf{B}$ using the entries of \mathbf{B} directly as root representation in step 1.
This gives eigenpairs $(\hat{\lambda}_i, \hat{\mathbf{v}}_i), i \in I_0$.
- 3: Choose one of the following methods to determine $\bar{\sigma}_i$:
 - (a) take some combination of $\check{\lambda}_i^{1/2}$ and $\hat{\lambda}_i^{1/2}$
 - (b) recompute to high relative accuracy (bisection or dqds)
 - (c) set $\bar{\sigma}_i := \check{\mathbf{u}}_i^* \mathbf{B} \hat{\mathbf{v}}_i$

fact, if dqds is used to precompute the singular values (cf. §3.1.1), one can even skip steps 1) and 4), since the singular values that are computed for the blocks of the absolute split are discarded anyways. The sole purpose of the separate relative split is to speed up the refinement in step 4).

We want to stress that we propose the 2-phase splitting also when only a subset of singular triplets is desired. The additional obstacle is to get the association of triplet-indices between the blocks consistent. This can be done efficiently, but it is not entirely trivial.

3.2 MR³ and the Normal Equations

Arguably the most straightforward approach to tackle BSVD would be to just call MR³ twice, on $\mathbf{B}\mathbf{B}^*$ for the left and on $\mathbf{B}^*\mathbf{B}$ for the right singular vectors. As $\mathbf{B}\mathbf{B}^*$ and $\mathbf{B}^*\mathbf{B}$ are already positive definite bidiagonal factorizations, we would naturally take them directly as root representations, omitting the amateurish mistake to form either one explicitly. The resulting method is summarized in Algorithm 3.1.

In short, this approach is a bad idea, as the results of the following numerical experiment do convey.

Experiment 3.4. We realized Algorithm 3.1 within our software framework by calling `DSTEMR` from `LAPACK 3.2.1`. For step 3 we chose the average in option (a),

$\sigma_1, \dots, \sigma_{12}$	σ_{13}	σ_{14}	σ_{15}	σ_{16}	$\sigma_{17}, \dots, \sigma_{20}$
$= \sigma_{i+4}/100$	0.9	$1 - 10^{-7}$	$1 + 10^{-7}$	1.1	$= 100 \cdot \sigma_{i-4}$

Table 3.2: Singular value distribution of test matrix in Experiment 3.4.

but this makes no essential difference for the results below (we checked).

To facilitate explicitly forcing the root representations, we called again the internal subroutine DLARRV directly, just as we did for the tests in §2.5. For the left singular vectors we employed $(\mathbf{B}\mathbf{B}^*)^{\text{flip}} = (\mathbf{B}^{\text{flip}})^* \mathbf{B}^{\text{flip}}$ to obtain an upper bidiagonal factorization as root; the computed vectors then had to be reversed afterwards.

We used LAPACK’s test matrix generator DLATMS to construct a bidiagonal matrix with singular values that are distributed as shown in Table 3.2, ranging between 10^{-8} and 100. We were deliberate in ensuring that the resulting bidiagonal matrix is well within numerical range, so that DSTEMR would neither split nor scale either one of the tridiagonal problems $\mathbf{B}\mathbf{B}^*$ or $\mathbf{B}^*\mathbf{B}$.

The results are shown separately for each singular triplet in Figure 3.3. The plots a) and c) convey that the two eigenproblems $\mathbf{B}\mathbf{B}^*$ and $\mathbf{B}^*\mathbf{B}$ are solved to satisfaction. But the residual norms *for the SVD*, shown in plot d), are catastrophic. We observe that the problem seems to become worse for larger singular values, but only those that are computed at deeper levels—in particular note that the residuals for σ_{16} and σ_{20} , which are both large, are just fine. \diamond

The remainder of this section is devoted to provide an in-depth study to explain the problems exhibited by the experiment. Somewhat down the road, Theorem 3.19 in §3.5.2 will also show why singular triplets that are handled at the root are never cause for concern, regardless of their size.

The Fundamental Problem

To understand what is going wrong with Algorithm 3.1 we need to understand MR³ better, in particular with respect to what it promises to do and—even more importantly—what not.

Provided all requirements are fulfilled, MR³ delivers (numerically) orthonormal bases for (nearly) the invariant subspaces belonging to each cluster of eigenvalues; indeed this holds for each node in the tree. This is actually just a combination of Lemma 2.13 and Theorem 2.14, as the first conveys that the computed vectors are indeed “close” to the invariant subspace of the cluster they belong to, and the same argument that was used for proving the latter provides orthogonality.

However, there is no guarantee *at all* how the computed basis relates to the exact basis of the subspace (which is unique up to signs, as all eigenvalues are

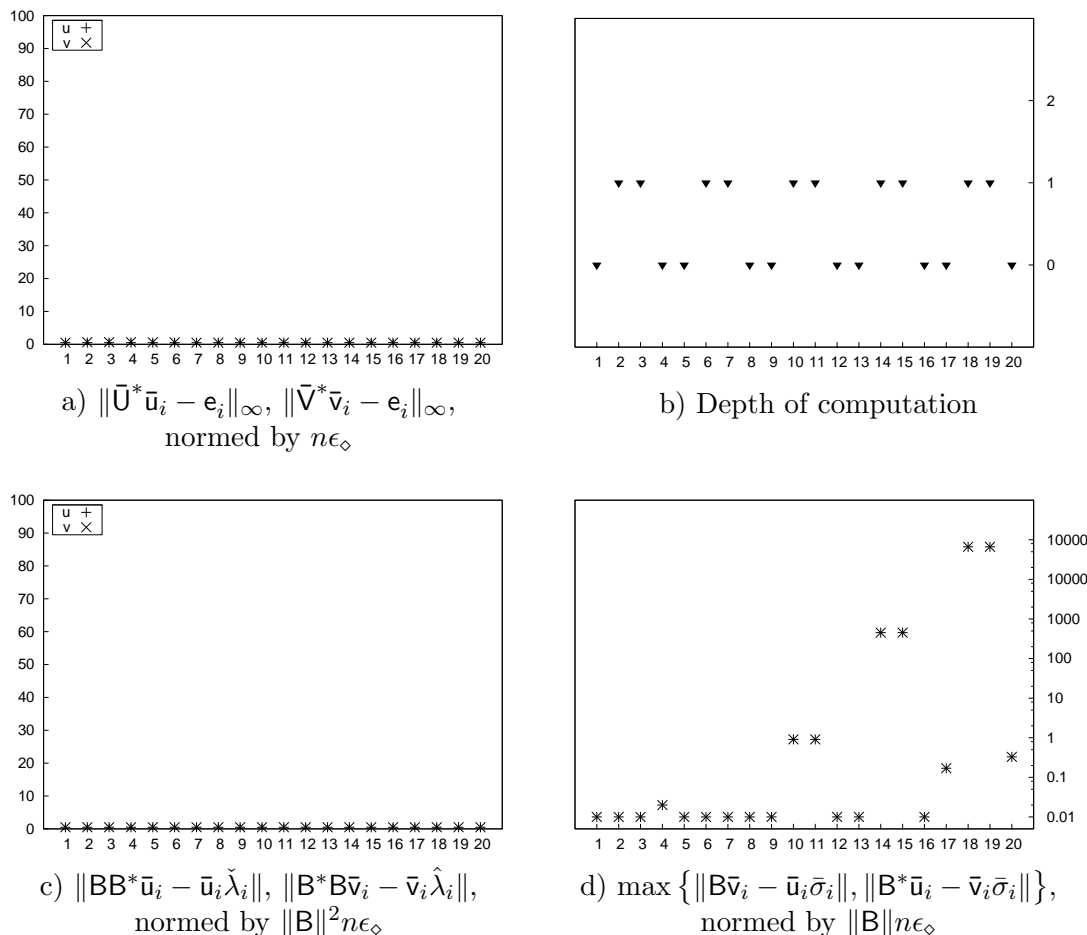


Figure 3.3: Data for Experiment 3.4, on a per-vector basis, $i = 1, \dots, 20$.

distinct). And with regard to Gap Theorem 1.21, it must be clear that no such guarantee could ever be given because the basis of an invariant subspace belonging to a cluster of eigenvalues is not well determined in the first place, with the degree of indetermination being inversely proportional to the (relative) width of the cluster. Note that this is true even if a relatively robust representation is employed. Note also that this in no way contradicts the claim that the computed vectors have small residual norms. In a numerical sense, a very tight cluster of eigenvalues does behave just like a bunch of multiple eigenvalues, with the concept of a unique basis for the associated eigenspace being moot.

To summarize, we can state the following informal property:

For each cluster, MR³ computes an orthonormal basis that is more or less “random”.

Of course it is not truly random; after all, MR³ is deterministic. But the analogy does help in understanding how MR³ behaves. The “randomness” actually

stems directly from the perturbations arising from executing the shifts, and as we generally do not have any control over those, the results do appear random.

The fundamental problem with the normal equations becomes clear: The left and right singular vectors are computed independently and do not necessarily bear any relation to the exact ones; in fact they might be completely “mixed up” for clusters of tight singular values. Thus, there is no reason why they should fit together to produce good residuals for the SVD. In [37, sec. 4.5] Großer & Lang give a very appealing view of this situation based on the geometric interpretation of the SVD using (hyper-)ellipses. We will not reproduce it here, but would instead urge the reader to refer to the source directly.

The local left- and right eigenvalues

The problems involved with Algorithm 3.1 were observed by Großer in his thesis [35]. He also noted that the SVD-residuals tend to become worse for the larger singular values, which we already saw in Experiment 3.4.

As a next step, Großer analyzed how the representation trees of the two runs on the roots $\mathbf{B}\mathbf{B}^*$ and $\mathbf{B}^*\mathbf{B}$ do relate. As those two have identical eigenvalues, in exact arithmetic one could choose identical shifts and the local eigenvalues at each pair of corresponding nodes in the trees would remain identical. However, it turns out that the rounding errors inherently involved in going from father to child do destroy this harmony; Example 4.5 in [37] clearly demonstrates that the eigenvalues of the nodes just one level down from the roots may in principle differ in all significant digits.

The following thought experiment will cast more light on this facet (we gave this in similar form [72], see also Satz 3.1 in [35]). Assume that, on the first level of the representation trees of $\mathbf{B}\mathbf{B}^*$ and $\mathbf{B}^*\mathbf{B}$, a cluster $\sigma_{c,d}^2$ is encountered. Furthermore, in order to proceed to the next level and break up this cluster, let the same shift $\tau \approx \sigma_c^2$ be chosen close by, and let the associated child representations

$$\mathbf{B}\mathbf{B}^* - \tau = \check{\mathbf{M}}, \quad \mathbf{B}^*\mathbf{B} - \tau = \hat{\mathbf{M}} \quad (3.15)$$

form RRRs for their respective eigenpairs $c : d$.

Now, in exact arithmetic, for the local eigenvalues $\hat{\lambda}_i$ of $\hat{\mathbf{M}}$ and $\check{\lambda}_i$ of $\check{\mathbf{M}}$, we would have $\hat{\lambda}_i = \sigma_i^2 - \tau = \check{\lambda}_i$, where $\sigma_i = \sigma_i[\mathbf{B}]$. In practice, however, we only have mixed relative accuracy, meaning the relations (3.15) can only be expected to hold for small erps of the father- and child-representations. As we assume RRRs, the *actual* relationship between $\hat{\lambda}_i$ and $\check{\lambda}_i$ will therefore be of the form

$$\begin{aligned} \check{\lambda}_i(1 + \varepsilon_1) &= \sigma_i^2(1 + \varepsilon_2) - \tau, \\ \hat{\lambda}_i(1 + \varepsilon_3) &= \sigma_i^2(1 + \varepsilon_4) - \tau, \end{aligned}$$

with suitable small constants $\varepsilon_i \equiv \mathcal{O}(n\varepsilon_\circ)$. If the cluster is tight and the shift is chosen close to it ($\sigma_c^2 \approx \sigma_d^2 \approx \tau$), we will in general have $\sigma_i^2 \gg \max\{|\check{\lambda}_i|, |\hat{\lambda}_i|\}$,

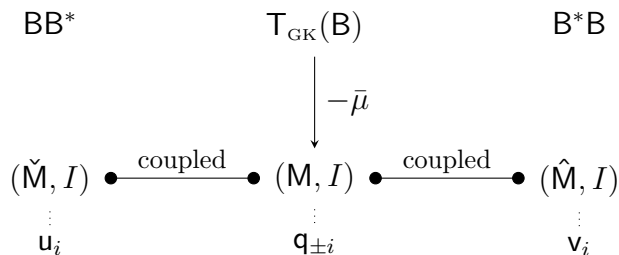


Figure 3.4: Situation for the theoretical analysis as to why computing the left and right singular vectors separately cannot work: \mathbf{M} is exactly shifted from $\mathsf{T}_{\text{GK}}(\mathbf{B})$, its local eigensystem I is coupled to $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$.

for $i = c, \dots, d$. Then it is clear that the *absolute* deviation between the local eigenvalues can become as large as

$$|\check{\lambda}_i - \hat{\lambda}_i| = \mathcal{O}(\sigma_i^2 n \epsilon_\circ). \quad (3.16)$$

In summary, the independent computations of $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$ will cause a slight difference in the implied initial relative perturbations of the σ_i^2 ($\epsilon_2 \neq \epsilon_4$). Then the cancellation ($\sigma_i^2 \approx \tau$) which is implicit—and desired—in shifting close to a cluster magnifies this small deviation up to (3.16).

Clearly, (3.16) shows that, for a tight cluster and a close shift, we cannot expect any kind of relative similarity between $\check{\lambda}_i$ and $\hat{\lambda}_i$. Also it becomes apparent why the problems with the normal equations tend to happen more for the larger singular values.

The problem in detail

The observation that the local eigenvalues might not be close in a relative sense is revealing and hints at what goes wrong. But it does not *explain* the bad residual norms for the SVD. Indeed, even if the runs on $\mathbf{B}\mathbf{B}^*$ and $\mathbf{B}^*\mathbf{B}$ were done with identical shifts and somehow in a way that guarantees close local eigenvalues at all nodes, one would be hard pressed to actually prove that the computed vectors fit together well—ourselves we do not deem it contrivable.

We will now develop a deeper theoretical analysis of the problem to provide more insight into what really goes wrong. It will exhibit that the diverging local eigenvalues are more symptom than cause.

Assume we have representations $\check{\mathbf{M}}$, \mathbf{M} and $\hat{\mathbf{M}}$ such that \mathbf{M} is an exact translate of $\mathsf{T}_{\text{GK}}(\mathbf{B})$ and their partial eigensystems for an index set I are coupled, as shown in Figure 3.4. We do not care at all how the outer representations have been obtained. It could be that they too are exactly shifted from the roots, meaning $\check{\mathbf{M}} = \mathbf{B}\mathbf{B}^* - \bar{\mu}^2$, $\hat{\mathbf{M}} = \mathbf{B}^*\mathbf{B} - \bar{\mu}^2$, but for the following analysis to apply only

the fact that the eigensystems are coupled is relevant. Note that this situation encompasses Algorithm 3.1 for $\bar{\mu} = 0$.

The motivation for allowing $\bar{\mu} \neq 0$ is as follows. In the next section we will see that BSVD can be solved using MR³ on the Golub-Kahan matrix as long as all translates are in some sense “ok”. Thus one could be tempted to work with $T_{\text{GK}}(\mathbf{B})$ until problems arise, and only then use the coupling transformations from §3.4 to switch to the outer layers. The argument in favor of this strategy would be that the local eigenvalues of $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$ have become much smaller due to shifting, so using the normal equations should be ok now. The following analysis will expose also this train of thought to be misguided.

Consider some index $i \in I$ and let $\mathbf{u}_i, \mathbf{v}_i$ be the exact eigenvectors of $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$, respectively, and $\mathbf{q}_{\pm i}$ the ones of \mathbf{M} . Suppose the vectors $\bar{\mathbf{u}}$ and $\bar{\mathbf{v}}$ were computed using MR³ separately on $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$. As was stated before, MR³ can only guarantee that they will be close to the invariant subspaces belonging to I . Thus we can write them as

$$\begin{aligned}\bar{\mathbf{u}} &= \sum_{i \in I} \xi_i \mathbf{u}_i + \mathbf{r}, & \mathbf{r} \perp \mathcal{U}_I, & \quad \|\mathbf{r}\| = \sin \angle(\mathcal{U}_I, \bar{\mathbf{u}}) = \mathcal{O}(n\epsilon_\diamond / \text{relgap}_{\check{\mathbf{M}}}(I)), \\ \bar{\mathbf{v}} &= \sum_{i \in I} \eta_i \mathbf{v}_i + \mathbf{s}, & \mathbf{s} \perp \mathcal{V}_I, & \quad \|\mathbf{s}\| = \sin \angle(\mathcal{V}_I, \bar{\mathbf{v}}) = \mathcal{O}(n\epsilon_\diamond / \text{relgap}_{\hat{\mathbf{M}}}(I)).\end{aligned}$$

Again we may assume exactly normalized vectors, $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$. The root-level SVD-residuals can equally well be measured by combining the computed vectors into $\bar{\mathbf{q}}$ and evaluating that one’s residual for $T_{\text{GK}}(\mathbf{B})$. So we construct

$$\sqrt{2}P_{\text{ps}}^* \bar{\mathbf{q}} = \begin{bmatrix} \sum_{i \in I} \xi_i \mathbf{u}_i + \mathbf{r} \\ \sum_{i \in I} \eta_i \mathbf{v}_i + \mathbf{s} \end{bmatrix}, \quad (3.17)$$

defining $\bar{\mathbf{q}}$. It would have been equally feasible to take $-\bar{\mathbf{v}}$ here, but this makes no difference. Casting $\bar{\mathbf{q}}$ in terms of $T_{\text{GK}}(\mathbf{B})$ ’s exact eigenvectors

$$\sqrt{2}\mathbf{q}_{\pm j} = P_{\text{ps}} \begin{bmatrix} \mathbf{u}_j \\ \pm \mathbf{v}_j \end{bmatrix}, \quad \mathcal{Q}_{\pm I} = \mathcal{Q}_{\pm I}[\mathbf{M}] = \mathcal{Q}_{\pm I}[T_{\text{GK}}(\mathbf{B})] = \text{span}\{\mathbf{q}_j \mid j \in \pm I\},$$

reveals $\sqrt{2}P_{\text{ps}}^* \bar{\mathbf{q}} = \mathbf{w} + \mathbf{t}$, where

$$\begin{aligned}\mathbf{w} &= \sum_{i \in I} \frac{1}{2}(\xi_i - \eta_i)\mathbf{q}_{-i} + \sum_{i \in I} \frac{1}{2}(\xi_i + \eta_i)\mathbf{q}_i \in \mathcal{Q}_{\pm I}, \\ \mathbf{t} &= \begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix} \in \mathcal{Q}_{\pm I}^\perp.\end{aligned}$$

Hence, $P_{\text{ps}}(\mathbf{w} + \mathbf{t})$ is just the unique orthogonal decomposition of $\sqrt{2}\bar{\mathbf{q}}$ along $\mathcal{Q}_{\pm I} = \mathcal{Q}_{-I} \oplus \mathcal{Q}_I$, the direct sum of \mathcal{Q}_{-I} and \mathcal{Q}_I . Thus,

$$\sin^2 \angle(\mathcal{Q}_{\pm I}, \bar{\mathbf{q}}) = \|\mathbf{r}\|^2 + \|\mathbf{s}\|^2 = \sin^2 \angle(\mathcal{U}_I, \bar{\mathbf{u}}) + \sin^2 \angle(\mathcal{V}_I, \bar{\mathbf{v}}).$$

Now, if we had singletons, $|I| = 1$, then we could deduce some more information about $\bar{\mathbf{q}}$. That is exactly what we will do in the proof of Theorem 3.19 down the road. However, for the case $|I| > 1$ we cannot; in particular there is no general way to bound $\sin\angle(\mathcal{Q}_{-I}, \bar{\mathbf{q}})$ or $\sin\angle(\mathcal{Q}_I, \bar{\mathbf{q}})$ separately. And here is the problem, because a small angle between a vector and an invariant subspace does *not* imply small residuals; the following lemma captures in very general terms that the only realistic bound on the residual is the *absolute* spread of the eigenvalues.

Lemma 3.5. *Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be symmetric. Then for any nonempty index set $I \subseteq \{1, \dots, n\}$ one can find a normalized vector $\mathbf{x} \in \mathcal{Q}_I[\mathbf{A}]$ such that*

$$\|\mathbf{A}\mathbf{x} - \rho(\mathbf{x})\mathbf{x}\| = \frac{1}{2}(\lambda_b[\mathbf{A}] - \lambda_a[\mathbf{A}]),$$

where $\rho(\mathbf{x}) = \mathbf{x}^*\mathbf{A}\mathbf{x}$ and $a = \min I$, $b = \max I$.

Proof. $\mathbf{x} := \frac{1}{\sqrt{2}}\mathbf{q}_a[\mathbf{A}] + \frac{1}{\sqrt{2}}\mathbf{q}_b[\mathbf{A}]$. □

Applied to the problem at hand, we see that, if we merely know that the composed vector $\bar{\mathbf{q}}$ has a small angle to the *combined* subspace $\mathcal{Q}_{\pm I}[\mathbf{T}_{\text{GK}}(\mathbf{B})]$, then the best bound for the residual norm that we can give is in terms of the *root* singular values, namely $\max\{|\sigma_i| : i \in I\}$, and this is sharp!

The preceding analysis shows two things. First, as the theoretical worst-case bound for the residuals is the absolute size of the singular values, it explains why the problems with the normal equations become more pronounced for larger singular values. Indeed, we have shown that the approach would be perfectly valid for clusters of tiny singular values ($\approx \text{spdiam}\epsilon_\circ$), but that is not very useful. And second, it demonstrates why the relationship between the \mathbf{u} 's and \mathbf{v} 's is so critical. If those do not fit together, the composed vector \mathbf{q} for the Golub-Kahan matrix can have non-negligible contributions from *both* spaces \mathcal{Q}_{-I} and \mathcal{Q}_I , and exactly this is the cause for bad SVD-residuals.

3.3 MR³ and the Golub-Kahan Matrix

In this section we investigate the approach to use MR³ on the Golub-Kahan matrix to solve the problem BSVD. The standing opinion for a long time has been that there are fundamental problems involved which cannot be overcome, in particular concerning the orthogonality of the extracted left and right singular vectors. The main objective of this section is to refute that notion.

We start our exposition with a numerical experiment to indicate that using MR³ as a pure black-box method on the Golub-Kahan matrix is indeed not a sound idea.

Experiment 3.6. We took the same bidiagonal matrix $\mathbf{B} \in \mathbb{R}^{20 \times 20}$ as in Experiment 3.4 (cf. Table 3.2 for the singular values), formed the symmetric tridiagonal matrix $\mathbf{T}_{\text{GK}}(\mathbf{B}) \in \mathbb{R}^{40 \times 40}$ explicitly, and called the current MR³-implementation DSTEMR from LAPACK 3.2.1 to give us the upper 20 eigenpairs $(\bar{\sigma}_i, \bar{\mathbf{q}}_i)$ of $\mathbf{T}_{\text{GK}}(\mathbf{B})$. The singular vectors were then extracted according to

$$\begin{bmatrix} \bar{\mathbf{u}}_i \\ \bar{\mathbf{v}}_i \end{bmatrix} := \sqrt{2} \mathbf{P}_{\text{ps}}^* \bar{\mathbf{q}}_i. \quad (3.18)$$

The results are shown in Figure 3.5. Note that the residual norms for BSVD, in the sense of (3.4), can be measured in terms of the TSEP-residuals for $\mathbf{T}_{\text{GK}}(\mathbf{B})$, since

$$\mathbf{T}_{\text{GK}}(\mathbf{B}) \bar{\mathbf{q}}_i - \bar{\mathbf{q}}_i \bar{\sigma}_i = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{B} \bar{\mathbf{v}}_i - \bar{\mathbf{u}}_i \bar{\sigma}_i & 0 \\ 0 & \mathbf{B}^* \bar{\mathbf{u}}_i - \bar{\mathbf{v}}_i \bar{\sigma}_i \end{bmatrix}, \quad (3.19)$$

provided the scaling by $\sqrt{2}$ in (3.18) were done exactly.

The plots a) and c) on the left of Figure 3.5 clearly show that DSTEMR does its job of solving the eigenproblem posed by $\mathbf{T}_{\text{GK}}(\mathbf{B})$. But plot b) on the upper right conveys just as clearly that the extracted singular vectors are far from being orthogonal. Note the complementing behavior to Experiment 3.4—now the small singular values are causing trouble. Furthermore, plot d) shows that the \mathbf{u} - and \mathbf{v} -components have somehow lost their property of having equal norm. However, note that their norms are still close enough to one that normalizing them explicitly, instead of the multiplication by $\sqrt{2}$ in (3.18), would not affect the orthogonality levels significantly. \diamond

The previous experiment is not special—similar behaviour can be consistently observed for other test cases with small singular values. There is actually a rather simple explanation for it: MR³ does neither know, nor care, what a Golub-Kahan matrix is. It will start just as always, by first choosing a shift outside the spectrum, say $\tau \lesssim -\sigma_n$, and compute $\mathbf{T}_{\text{GK}}(\mathbf{B}) - \tau = \mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^*$ as positive definite root representation. From there it will then deploy further shifts into the spectrum of $\mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^*$ to isolate the requested eigenpairs.

What happens is that the first shift to the outside smears all small singular values into one cluster, as shown in Figure 3.6. Consider for instance we have $\|\mathbf{B}\| \geq 1$ and are working with the standard $\text{gaptol} = 0.001$. We can even assume the initial shift was done exactly; so let $\lambda_{\pm i}^{(0)} = \sigma_{\pm i} - \tau$ be the eigenvalues of $\mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^*$. Then for all indices i with $\sigma_i \lesssim 0.0005$ the corresponding $\lambda_{\pm i}^{(0)}$ will belong to the same cluster of $\mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^*$, since

$$\text{reldist}(\lambda_{-i}^{(0)}, \lambda_{+i}^{(0)}) = \frac{(\sigma_i - \tau) - (-\sigma_i - \tau)}{\sigma_i - \tau} = \frac{2\sigma_i}{\sigma_i - \tau} < \text{gaptol}.$$

For singular vectors \mathbf{u}_i and \mathbf{v}_i , both of $\mathbf{P}_{\text{ps}} \begin{pmatrix} \mathbf{u}_i \\ \pm \mathbf{v}_i \end{pmatrix}$ will be eigenvectors associated with that cluster. Hence, further (inexact) shifts based on this configuration

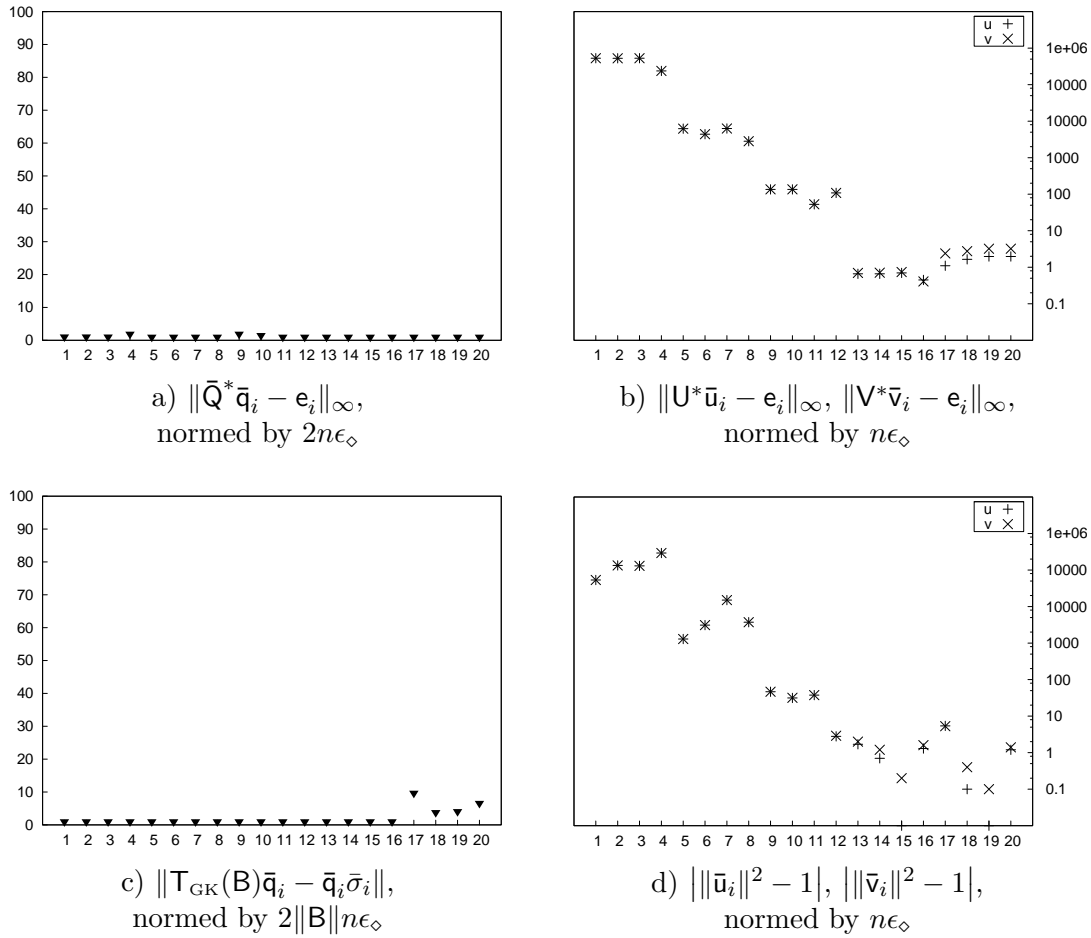


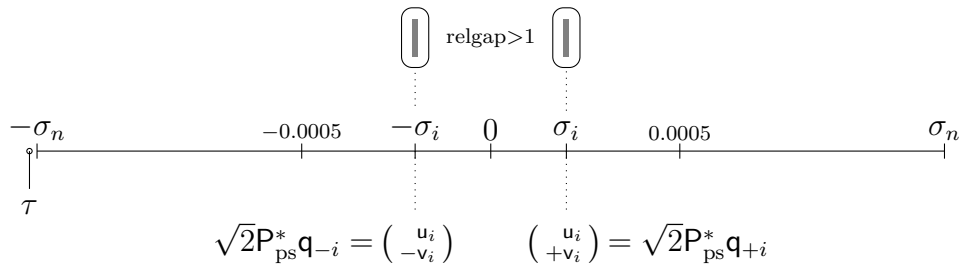
Figure 3.5: Data for Experiment 3.6, on a per-vector basis, $i = 1, \dots, 20$.

cannot guarantee to separate them again cleanly. Consequently, using MR³ as black-box on the Golub-Kahan matrix in this fashion could in principle even produce eigenvectors \mathbf{q} with *identical* u - or v - components.

The problem described is rather easy to overcome. After all we know that the entries of $T_{\text{GK}}(\mathbf{B})$ form an RRR, so the initial outside shift to find a positive definite root representation is completely unnecessary—we can just take $\mathbf{M}_0 := T_{\text{GK}}(\mathbf{B})$ directly as root. For shifting, that is, for computing a child representation $\mathbf{M}^+ = T_{\text{GK}}(\mathbf{B}) - \mu$ on the first level, a special routine exploiting the zero diagonal should be employed; if \mathbf{M}^+ is to be a twisted factorization this is much easier to do than standard `dtwqds`, see [28, 48] and our remarks in §2.4.5. With this setting, small singular values can be handled by a (positive) shift in one step, without danger of spoiling them by unwanted contributions from the negative counterparts. This solution method is sketched in Algorithm 3.2.

Note that we now have heterogeneous representation types in the tree, as the

Spectrum of $T_{\text{GK}}(\mathbf{B})$:



Spectrum of $L_0 D_0 L_0^* = T_{\text{GK}}(\mathbf{B}) - \tau$:

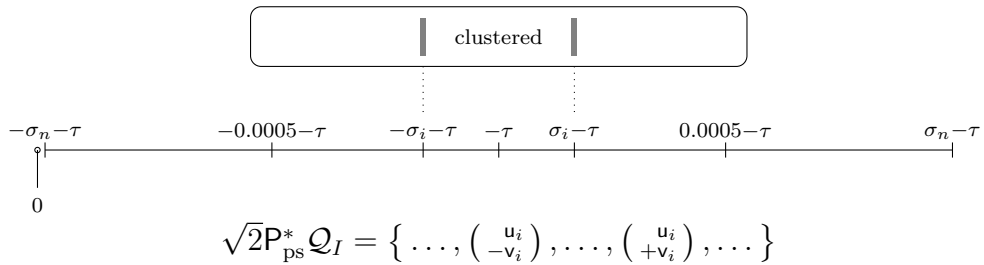


Figure 3.6: Why the naive black-box approach of MR³ on T_{GK} is doomed.

root $T_{\text{GK}}(\mathbf{B})$ is represented by its entries. In any case, our general setup of MR³ and its proof in Chapter 2 can handle this effortlessly.

One can argue that the approach is still flawed on a fundamental level. Großer gives a nice example in [35] which we want to repeat at this point. In fact his argument can be fielded against using *any* TSEP-solver on the Golub-Kahan matrix for BSVD.

Example 3.7 (cf. Beispiel 1.33 in [35]). Assume the exact GK-eigenvectors

$$\mathbf{P}_{\text{ps}}^* \mathbf{q}_i = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{u}_i \\ \mathbf{v}_i \end{bmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ 1 \\ 1 \\ -1 \end{pmatrix}, \quad \mathbf{P}_{\text{ps}}^* \mathbf{q}_j = \frac{1}{\sqrt{2}} \begin{bmatrix} \mathbf{u}_j \\ \mathbf{v}_j \end{bmatrix} = \frac{1}{2} \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix},$$

form (part of) the basis for a cluster. The computed vectors will generally not be exact, but might for instance be $\mathbf{P}_{\text{rot}} \mathbf{P}_{\text{ps}}^* [\mathbf{q}_i | \mathbf{q}_j]$, where \mathbf{P}_{rot} effects a rotation $\begin{pmatrix} c & s \\ -s & c \end{pmatrix}$, $c^2 + s^2 = 1$ within the 2-3 plane (thus influencing only the second entries of the \mathbf{u} 's and the first entries of the \mathbf{v} 's). We end up with computed singular

ALGORITHM 3.2 *MR³ on the Golub-Kahan matrix*

Compute specified singular triplets of bidiagonal \mathbf{B} using the MR³ algorithm on $\mathsf{T}_{\text{GK}}(\mathbf{B})$.

Input: Upper bidiagonal $\mathbf{B} \in \mathbb{R}^{n \times n}$. Index set $I_0 \subseteq \{1, \dots, n\}$.

Output: Singular triplets $(\bar{\sigma}_i, \bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i), i \in I_0$.

- 1: Execute Algorithm 2.1, but take $\mathsf{M}_0 := \mathsf{T}_{\text{GK}}(\mathbf{B})$ as root representation in step 1, using the entries of \mathbf{B} directly.
This gives eigenpairs $(\bar{\sigma}_i, \bar{\mathbf{q}}_i), i \in I_0$.
- 2: Extract the singular vectors via $\begin{bmatrix} \bar{\mathbf{u}}_i \\ \bar{\mathbf{v}}_i \end{bmatrix} := \sqrt{2}\mathsf{P}_{\text{ps}}^* \bar{\mathbf{q}}_i$.

vectors

$$\sqrt{2}\bar{\mathbf{u}}_i = \begin{bmatrix} 1 \\ c+s \end{bmatrix}, \sqrt{2}\bar{\mathbf{u}}_j = \begin{bmatrix} 1 \\ s-c \end{bmatrix}, \sqrt{2}\bar{\mathbf{v}}_i = \begin{bmatrix} c-s \\ -1 \end{bmatrix}, \sqrt{2}\bar{\mathbf{v}}_j = \begin{bmatrix} c+s \\ 1 \end{bmatrix},$$

that have orthogonality levels $|\mathbf{u}_i^* \mathbf{u}_j| = |\mathbf{v}_i^* \mathbf{v}_j| = s^2$. However, this rotation does leave the invariant subspace spanned by \mathbf{q}_i and \mathbf{q}_j (cf. Lemma 3.9 on the next page), so if s^2 is large, the residual norms of $\bar{\mathbf{q}}_i$ and $\bar{\mathbf{q}}_j$ would suffer, too. \diamond

That the extracted singular vectors can be far from orthogonal even if the GK-vectors are fine led Großer to the conclusion that there must be a fundamental problem. Up until recently we believed that as well (cf. p.914 in [72]). However, we will now set out to prove that with just a small additional requirement, Algorithm 3.2 will actually work. This is a new result and shows that there is no *fundamental* problem in using MR³ on the Golub-Kahan matrix. Of particular interest is that the situation in Example 3.7—which, as we mentioned, should apply to all TSEP solvers on T_{GK} —can be avoided if MR³ is deployed as in Algorithm 3.2. Naturally we need more insights into the workings of MR³ to validate this claim, so for the remainder of this section we have to assume that the reader is familiar with §2.2.

The following definition will let us control the danger that the shifts within MR³ lose information about the singular vectors.

Definition 3.8. A subspace \mathcal{S} of $\mathbb{R}^{2n \times 2n}$ with orthonormal basis $(\mathbf{q}_i)_{i \in I}$ is said to have *GK-structure* if the systems $(\mathbf{u}_i)_{i \in I}$ and $(\mathbf{v}_i)_{i \in I}$ of vectors extracted according to

$$\begin{bmatrix} \mathbf{u}_i \\ \mathbf{v}_i \end{bmatrix} := \sqrt{2}\mathsf{P}_{\text{ps}}^* \mathbf{q}_i, \quad i \in I,$$

are orthonormal each. ◇

The special property of a GK-matrix is that all invariant subspaces belonging to (at most) the first or second half of the spectrum have GK-structure. As eigenvectors are invariant under shifting, this property transcends to any matrix that can be written as $T_{\text{GK}}(\mathbf{B}) - \mu$ for suitable \mathbf{B} , which is just any symmetric tridiagonal matrix of even dimension where the diagonal entries are all identical, i.e., matrices with a *constant diagonal*.

The next lemma reveals that the \mathbf{u} - and \mathbf{v} -components of every vector within a subspace with GK-structure have equal norm. Thus the actual choice of the orthonormal system (\mathbf{q}_i) in Definition 3.8 is irrelevant.

Lemma 3.9. *Let the subspace $\mathcal{S} \subseteq \mathbb{R}^{2n \times 2n}$ have GK-structure. Then for each $\mathbf{s} \in \mathcal{S}$,*

$$\sqrt{2}\mathbf{s} = \mathbf{P}_{\text{ps}} \begin{bmatrix} \mathbf{s}_u \\ \mathbf{s}_v \end{bmatrix} \quad \text{with} \quad \|\mathbf{s}_u\| = \|\mathbf{s}_v\|.$$

Proof. As \mathcal{S} has GK-structure, we have an orthonormal basis $(\mathbf{q}_1, \dots, \mathbf{q}_m)$ for \mathcal{S} such that

$$\sqrt{2}\mathbf{P}_{\text{ps}}^* \mathbf{q}_i = \begin{bmatrix} \mathbf{u}_i \\ \mathbf{v}_i \end{bmatrix}, \quad i = 1, \dots, m.$$

Thus each $\mathbf{s} \in \mathcal{S}$ can be written as

$$\begin{aligned} \mathbf{s} &= \alpha_1 \mathbf{q}_1 + \dots + \alpha_m \mathbf{q}_m \\ \implies \sqrt{2}\mathbf{P}_{\text{ps}}^* \mathbf{s} &= \begin{bmatrix} \alpha_1 \mathbf{u}_1 + \dots + \alpha_m \mathbf{u}_m \\ \alpha_1 \mathbf{v}_1 + \dots + \alpha_m \mathbf{v}_m \end{bmatrix} =: \begin{bmatrix} \mathbf{s}_u \\ \mathbf{s}_v \end{bmatrix}. \end{aligned}$$

Since the \mathbf{u}_i 's and \mathbf{v}_j 's are orthonormal we have $\|\mathbf{s}_u\|^2 = \sum \alpha_i^2 = \|\mathbf{s}_v\|^2$. □

Now comes the proof of concrete error bounds for Algorithm 3.2. The additional requirement we need is that the local subspaces are kept “near” to GK-structure. We will discuss how to handle this requirement in practice afterwards.

For simplicity we may assume that the call to MR³ in step 1 of Algorithm 3.2 produces perfectly normalized vectors, $\|\bar{\mathbf{q}}_i\| = 1$, and that the multiplication by $\sqrt{2}$ in step 2 is done exactly.

Theorem 3.10 (Proof of Correctness for Alg. 3.2)

Let Algorithm 3.2 be executed such that the representation tree built by MR³ satisfies all five requirements on page 55.

Furthermore, let each node (\mathbf{M}, I) have the property that a suitable perturbation $\tilde{\mathbf{M}}_{\text{GK}} = \text{erp}(\mathbf{M}, \xi_{\text{GK}})$ can be found such that the subspace $\mathcal{Q}_I[\tilde{\mathbf{M}}_{\text{GK}}]$ has GK-structure.

Finally, let resid_{GK} and orth_{GK} comprise the $\mathcal{O}(n\epsilon_\circ)$ rhs-bounds from Theorems 2.10 and 2.14, respectively, and define

$$A := \text{orth}_{\text{GK}} + C_{\text{vecs}} n \xi_{\text{GK}} / \text{gaptol}.$$

Then the computed singular triplets will satisfy

$$\begin{aligned} \max \{ \cos \angle(\bar{\mathbf{u}}_i, \bar{\mathbf{u}}_j), \cos \angle(\bar{\mathbf{v}}_i, \bar{\mathbf{v}}_j) \} &\leq 2\sqrt{2}A, \quad i \neq j, \\ \max \{ \left| \|\bar{\mathbf{u}}_i\| - 1 \right|, \left| \|\bar{\mathbf{v}}_i\| - 1 \right| \} &\leq \sqrt{2}A + \mathcal{O}(A^2), \\ \max \{ \|\mathbf{B}\bar{\mathbf{v}}_i - \bar{\mathbf{u}}_i\sigma_i\|, \|\mathbf{B}^*\bar{\mathbf{u}}_i - \bar{\mathbf{v}}_i\sigma_i\| \} &\leq \sqrt{2} \text{resid}_{\text{GK}}. \end{aligned}$$

Proof. The fact that all requirements for MR³ are fulfilled means that all the results proven in §2.2 do apply for the computed GK-eigenpairs $(\bar{\sigma}_i, \bar{\mathbf{q}}_i)$, in particular we have Lemma 2.13, Theorems 2.10 and 2.14.

We will first deal with the third bound concerning the residual norms, because that is the easiest. Just invoke the definition of the Golub-Kahan matrix and use Theorem 2.10 to see

$$\|\mathbf{B}\bar{\mathbf{v}}_i - \bar{\mathbf{u}}_i\bar{\sigma}_i\|^2 + \|\mathbf{B}^*\bar{\mathbf{u}}_i - \bar{\mathbf{v}}_i\bar{\sigma}_i\|^2 = 2\|\mathbf{T}_{\text{GK}}(\mathbf{B})\bar{\mathbf{q}}_i - \bar{\mathbf{q}}_i\bar{\sigma}_i\|^2 \leq 2\text{resid}_{\text{GK}}^2.$$

For orthogonality, the approach will be very similar to the one taken during the proof of Theorem 2.14, but we need Lemmas 3.1 and 3.9 to make the semantic step from \mathbf{q} to \mathbf{u} and \mathbf{v} . It suffices to focus on just the \mathbf{u} -components, as the case for the \mathbf{v} -components is (again) analogous.

Consider indices i and j and let (\mathbf{M}, N) be the deepest node in the tree such that $i \in I$ and $j \in J$ for different child index sets $I, J \subseteq N$. Recall that the bound orth_{GK} on the right-hand side in Theorem 2.14 is just the worst-case for the bound from Lemma 2.13, taken over all nodes in the tree. Hence we can state

$$\sin \angle(\mathcal{Q}_I[\mathbf{M}], \bar{\mathbf{q}}_i) \leq \text{orth}_{\text{GK}}.$$

As we have postulated that the representation \mathbf{M} fulfills Requirement RRR, we can link $\bar{\mathbf{q}}_i$ to the nearby matrix $\tilde{\mathbf{M}}_{\text{GK}}$ by

$$\sin \angle(\mathcal{Q}_I[\tilde{\mathbf{M}}_{\text{GK}}], \bar{\mathbf{q}}_i) \leq \text{orth}_{\text{GK}} + C_{\text{vecs}} n \xi_{\text{GK}} / \text{gaptol} = A.$$

This means we can find a unit vector $\mathbf{q} \in \mathcal{Q}_I[\tilde{\mathbf{M}}_{\text{GK}}]$ with $\sin\angle(\mathbf{q}, \bar{\mathbf{q}}_i) \leq A$.

At this point we invoke that $\mathcal{Q}_I[\tilde{\mathbf{M}}_{\text{GK}}] \subseteq \mathcal{Q}_N[\tilde{\mathbf{M}}_{\text{GK}}]$ has GK-structure. By Lemma 3.9 we can therefore partition

$$\sqrt{2}\mathbf{q} = \mathbf{P}_{\text{ps}} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \quad \text{with } \|\mathbf{u}\| = \|\mathbf{v}\| = 1.$$

Let $\mathcal{U}_I[\tilde{\mathbf{M}}_{\text{GK}}]$ denote the invariant subspace spanned by the \mathbf{u} -components of vectors in $\mathcal{Q}_I[\tilde{\mathbf{M}}_{\text{GK}}]$. Thus $\mathbf{u} \in \mathcal{U}_I[\tilde{\mathbf{M}}_{\text{GK}}]$ and Lemma 3.1 gives

$$\sin\angle(\mathcal{U}_I[\tilde{\mathbf{M}}_{\text{GK}}], \bar{\mathbf{u}}_i) \leq \sin\angle(\mathbf{u}, \bar{\mathbf{u}}_i) \leq \sqrt{2}A,$$

as well as the desired property $|\|\bar{\mathbf{u}}_i\| - 1| \leq \sqrt{2}A + \mathcal{O}(A^2)$ for the norms.

Repeat the steps above for j to arrive at

$$\sin\angle(\mathcal{U}_J[\tilde{\mathbf{M}}_{\text{GK}}], \bar{\mathbf{u}}_j) \leq \sqrt{2}A.$$

Since $\mathcal{Q}_N[\tilde{\mathbf{M}}_{\text{GK}}]$ has GK-structure and $I \cap J = \emptyset$, the spaces $\mathcal{U}_I[\tilde{\mathbf{M}}_{\text{GK}}]$ and $\mathcal{U}_J[\tilde{\mathbf{M}}_{\text{GK}}]$ are orthogonal. Then an argumentation just like the one used in the proof of Theorem 2.14 (cf. Figure 2.3) yields $\cos\angle(\bar{\mathbf{u}}_i, \bar{\mathbf{u}}_j) \leq 2\sqrt{2}A$. \square

One technical realization following from Theorem 3.10 is that it really does not matter if we extract the singular vectors as done in step 2 of Algorithm 3.2 by multiplying the \mathbf{q} -subvectors by $\sqrt{2}$, or if we normalize them explicitly.

We need to talk about the new requirement that was introduced in Theorem 3.10. It is stated minimally, namely that the representations \mathbf{M} can be perturbed to yield local invariant subspaces with GK-structure. In this situation we like to say that the subspace of \mathbf{M} “nearly” has GK-structure. There might be a way to specifically test for this property, but at the moment we do not see it. However, we do know that any even-dimensional symmetric tridiagonal matrix with a constant diagonal is just a shifted Golub-Kahan matrix, so trivially each subspace (within one half) has GK-structure. Let us capture this.

Definition 3.11. If for a given representation of symmetric tridiagonal \mathbf{M} there exists a elementwise relative perturbation

$$\tilde{\mathbf{M}} = \text{erp}(\mathbf{M}, \xi) \quad \text{such that} \quad \tilde{\mathbf{M}}(i, i) \equiv c,$$

we say that \mathbf{M} has a *nearly constant diagonal*, in short \mathbf{M} is *ncd*, or, if more detail is to be conveyed, $\mathbf{M} \in \text{ncd}(c)$ or $\mathbf{M} \in \text{ncd}(c, \xi)$. \diamond

Clearly, the precondition for Theorem 3.10 is fulfilled if all representations in the tree are ncd. Note that a representation being ncd does *not* necessarily imply that all diagonal entries are about equal, because there might be large local

element growth. For example, LDL^* can be ncd even if $|d_i| \gg |(\text{LDL}^*)(i, i)|$ for some index i , cf. Example 3.13 below.

The usefulness of identifying the ncd-property is that it can easily and cheaply be verified in practice. Note that the successively shifted descendants of a Golub-Kahan matrix can only violate the ncd-property if there was large local element growth at some diagonal entries on the way.

Remark 3.12. Since Theorem 3.10 requires SHIFTREL anyway, execution of the shifts $\text{T}_{\text{GK}}(\mathbf{B}) - \mu = \mathbf{M}^+$ to get to level one has to be done with mixed relative stability. Therefore, all representations on level one will automatically be ncd and as such always fulfill the extra requirement of having subspaces near to GK-structure, independent of element growth or relative condition numbers. \diamond

The preceding theoretical results will be demonstrated in action by numerical experiments in §3.6. Those will confirm that Algorithm 3.2 is indeed a valid solution strategy for BSVD. However, it will also become apparent that working with a Golub-Kahan matrix as root can sometimes be dangerous; in fact, the coupling approach in the next section will suffer from the same problem to some degree. The reason is that Golub-Kahan matrices are highly vulnerable to element growth when confronted with a tiny shift.

Example 3.13. Consider for $\alpha \ll 1$ the bidiagonal matrix

$$\mathbf{B} = \begin{pmatrix} 1 & 1 \\ & \alpha \end{pmatrix} \quad \text{with } \sigma_1 \approx \alpha, \sigma_2 \approx 2.$$

Think of $\alpha = \epsilon_\diamond$. Shifting $\text{T}_{\text{GK}}(\mathbf{B})$ by α gives

$$\begin{pmatrix} -\alpha & 1 & & \\ 1 & -\alpha & 1 & \\ & 1 & -\alpha & \alpha \\ & & \alpha & -\alpha \end{pmatrix} = \text{LDL}^*$$

with $\mathbf{D} = \text{diag}\left(-\alpha, \frac{1-\alpha^2}{\alpha}, -\alpha\frac{2-\alpha^2}{1-\alpha^2}, \alpha\frac{1+2\alpha^2}{2+\alpha^2}\right)$.

Clearly there is huge local element growth in $\mathbf{D}(2)$. This LDL^* still is ncd, but if we had to shift it again the property would probably be lost completely. \diamond

The thing is that we really have no way to avoid a tiny shift if clusters of tiny singular values are present. In Chapter 4 an alternative to twisted factorizations will be presented that is especially suited for shifting Golub-Kahan matrices and will effectively render the above concerns obsolete.

3.4 Couplings

In the next section we will describe an MR³-based approach for BSVD that is a hybrid of using MR³ on the normal equations or the Golub-Kahan matrix alone. The idea has been refined from pioneering work by Großer & Lang [35–37]. At its heart lies the insight that translates of the normal equations and the Golub-Kahan matrix can be linked using so-called *coupling relations*. Essentially these stem directly from the elementary formula

$$\begin{aligned} \begin{bmatrix} \mathbf{B}\mathbf{B}^* - \mu^2 & 0 \\ 0 & \mathbf{B}^*\mathbf{B} - \mu^2 \end{bmatrix} &= \begin{bmatrix} -\mu & \mathbf{B} \\ \mathbf{B}^* & -\mu \end{bmatrix} \begin{bmatrix} \mu & \mathbf{B} \\ \mathbf{B}^* & \mu \end{bmatrix} \\ &= \mathbf{P}_{\text{ps}}^* (\mathbf{T}_{\text{GK}}(\mathbf{B}) - \mu) (\mathbf{T}_{\text{GK}}(\mathbf{B}) + \mu) \mathbf{P}_{\text{ps}}. \end{aligned} \quad (3.20)$$

The purpose of this section is preparatory groundwork, namely to derive the coupling relations from first principles.

In [35, 36] the coupling relations are presented separately for two cases:

- (1) One level down from the root(s), called “positive definite initial matrices”: Lemma 2.3 in [36], Lemma 3.6 in [35]. These are based on previous work by Fernando [28].
- (2) “indefinite initial matrices” for deeper levels: Satz 3.17 in [35], although this had an omission in the proof, which was corrected in Lemma 3.1 of [36].

The reason for the separate treatment of (1) is that here the auxiliary quantities s_i, p_i from the dqds-transformations can be harnessed to provide for somewhat slicker formulae. Our presentation omits the distinction because these are of no practical relevance (and not used) for the algorithm in §3.5.

We give a completely different proof that is more rudimentary while being (in our opinion) easier to follow. In fact we are not completely happy with how Lemma 3.1 of [36] was set up, because the requirement of a constant diagonal—which is crucial for the coupling relations to hold—was not clearly stated in the formulation there but only appears rather hidden in the proof. This could well have been one cause why an error down the road (Theorem 5.2 in [37]) went undetected; we will elaborate on this in §3.5.

The following presentation will not only strongly pronounce the requirement for a (nearly) constant diagonal but also exhibit the consequences when it is not fulfilled, in the form of the error terms (3.26). Furthermore we extend the relations to allow coupling of arbitrary twisted factorizations stably in practice.

The `IMPLICIT 0` convention from page 67 will be used throughout this section.

We start considering general tridiagonal matrices and only later consider the application to shifted Golub-Kahan matrices. Assume we have scalars μ_1, \dots, μ_m ,

$e_1, \dots, e_{m-1} \in \mathbb{R}$ and

$$\begin{aligned} \mathsf{T}_- &:= \text{diag}_{\pm 1}(e_1, \dots, e_{m-1}) - \text{diag}(\mu_1, \dots, \mu_m), \\ \mathsf{T}_+ &:= \text{diag}_{\pm 1}(e_1, \dots, e_{m-1}) + \text{diag}(\mu_1, \dots, \mu_m). \end{aligned} \quad (3.21)$$

The object of interest is the matrix

$$\mathsf{H} := \mathsf{T}_- \cdot \mathsf{T}_+ \in \mathbb{R}^{m \times m}. \quad (3.22)$$

In particular we want to study how the entries of H can be inferred from representations for T_- and T_+ .

Lemma 3.14. *For T_- , T_+ and H defined as in (3.21) and (3.22), the matrix H is symmetric pentadiagonal with entries*

$$\begin{aligned} \mathsf{H}(i, i) &\equiv e_{i-1}^2 + e_i^2 - \mu_i^2, \\ \mathsf{H}(i, i+1) &\equiv e_i(\mu_i - \mu_{i+1}), \\ \mathsf{H}(i, i+2) &\equiv e_i e_{i+1}. \end{aligned}$$

Proof. A straightforward calculation. \square

Note that because of (3.20), applying this result to a Golub-Kahan matrix ($\mu_i \equiv 0$, $m = 2n$) is just another way of expressing the entries of $\mathsf{B}\mathsf{B}^*$ and $\mathsf{B}^*\mathsf{B}$ in terms of the entries of B , cf. (3.6).

Lemma 3.15. *Let T_- , T_+ and H be defined as in (3.21) and (3.22) and let $\mathsf{T}_- = \mathsf{L}\mathsf{D}\mathsf{L}^*$ be a top-to-bottom bidiagonal factorization of T_- . Then*

$$\begin{aligned} \mathsf{H}(i, i) &\equiv -d_i d_{i+1} - l d_{i-1} l d_{i-2} \\ &\quad + d_i(\mu_i - \mu_{i+1}) + l d_{i-1}(\mu_i - \mu_{i-1}), \\ \mathsf{H}(i, i+1) &\equiv l d_i(\mu_i - \mu_{i+1}), \\ \mathsf{H}(i, i+2) &\equiv l d_i l d_{i+1}. \end{aligned}$$

Proof. With the identity $e_i \equiv l d_i$ the equations for the offdiagonals follow immediately from Lemma 3.14.

For the diagonal elements, Lemma 3.14 gives the further identities

$$\mathsf{H}(i, i) \equiv e_{i-1}^2 + e_i^2 - \mu_i^2. \quad (\star)$$

From $\mathsf{T}_- = \mathsf{L}\mathsf{D}\mathsf{L}^*$ we know

$$-\mu_i \equiv d_i + l d_{i-1} \quad \text{and} \quad e_i^2 \equiv l d_i \cdot d_i, \quad (\star\star)$$

which yield

$$\begin{aligned} e_{i-1}^2 &= -l d_{i-1} \mu_{i-1} - l d_{i-1} l d_{i-2}, \\ e_i^2 &= -d_i \mu_{i+1} - d_i d_{i+1}, \\ -\mu_i^2 &= \mu_i(d_i + l d_{i-1}). \end{aligned}$$

Substitute these into (\star) and rearrange to get the result. \square

Corollary 3.16. *Assume the situation is as in Lemma 3.15 but that we have a bottom-to-top bidiagonal factorization $\mathsf{T}_- = \mathsf{URU}^*$ instead. Then*

$$\begin{aligned} \mathsf{H}(i, i) &\equiv -r_{i-1}r_i - \mathsf{ur}_{i+1}\mathsf{ur}_{i+2} \\ &\quad + r_i(\mu_i - \mu_{i-1}) + \mathsf{ur}_{i+1}(\mu_i - \mu_{i+1}), \\ \mathsf{H}(i, i+1) &\equiv \mathsf{ur}_{i+1}(\mu_i - \mu_{i+1}), \\ \mathsf{H}(i, i+2) &\equiv \mathsf{ur}_{i+1}\mathsf{ur}_{i+2}. \end{aligned}$$

□

The next result combines and extends Lemma 3.15 and Corollary 3.16 to general twisted factorizations of T_- . The drawback for the moment is that we need to have the data for two consecutive twist indices k and $k+1$. We will deal with the ramifications of this requirement at the end of this section.

Lemma 3.17. *Let T_- , T_+ and H be defined as in (3.21) and (3.22) and suppose we have twisted factorizations of T_- for two consecutive twist indices $k, k+1$,*

$$\mathsf{T}_- = \mathsf{N}_k \mathsf{G}_k \mathsf{N}_k^* = \mathsf{N}_{k+1} \mathsf{G}_{k+1} \mathsf{N}_{k+1}^*,$$

that is, we have $n+2$ pivots $d_1, \dots, d_k, \gamma_k, \gamma_{k+1}, r_{k+1}, \dots, r_n$. Define

$$\begin{aligned} \varsigma(i) &:= -\mathsf{lld}_{i-1}\mathsf{lld}_{i-2} + \mathsf{lld}_{i-1}(\mu_i - \mu_{i-1}), & 1 \leq i \leq k+1, \\ \pi(i) &:= -\mathsf{ur}_{i+1}\mathsf{ur}_{i+2} + \mathsf{ur}_{i+1}(\mu_i - \mu_{i+1}), & k \leq i \leq n, \end{aligned}$$

(which imply $\varsigma(1) = \pi(n) = 0$). Then the diagonal elements of H obey

$$\mathsf{H}(i, i) = \begin{cases} -d_i d_{i+1} + d_i(\mu_i - \mu_{i+1}) + \varsigma(i), & i < k \\ \varsigma(i) + \gamma_i \mu_i + \pi(i), & i \in \{k, k+1\} \\ -r_i r_{i-1} + r_i(\mu_i - \mu_{i-1}) + \pi(i), & i > k+1. \end{cases}$$

Proof. We have the data d_1, \dots, d_k and $\ell_1, \dots, \ell_{k-1}$ to describe a complete lower bidiagonal factorization of $(\mathsf{T}_-)_{1:k}$. Thus we can invoke Lemma 3.15 to get equations for $\mathsf{H}(i, i), i < k$. At this point we have to be careful, because Lemma 3.15 would happily offer us a relation for $\mathsf{H}(k, k)$ as well, but this we must reject as it implicitly assumes $\mu_{k+1} = 0$, which we might not have here.

However, for the case $i = k$ we do know from Lemma 3.14 that

$$\mathsf{H}(k, k) = e_{k-1}^2 + e_k^2 - \mu_k^2, \tag{A}$$

and using $\mathsf{T}_- = \mathsf{N}_k \mathsf{G}_k \mathsf{N}_k^*$ we can furthermore deduce

$$d_{k-1} = -\mu_{k-1} - \mathsf{lld}_{k-2}, \tag{B}$$

$$-\mu_k^2 = \mu_k(\mathsf{lld}_{k-1} + \gamma_k + \mathsf{ur}_{k+1}), \tag{C}$$

$$r_{k+1} = -\mu_{k+1} - \mathsf{ur}_{k+2}. \tag{D}$$

Multiply (B) with ℓd_{k-1} , (D) with $u r_{k+1}$ and employ the identities $\ell d_{k-1} d_{k-1} = e_{k-1}^2$, $u r_{k+1} r_{k+1} = e_k^2$ to obtain

$$\begin{aligned} e_{k-1}^2 &= -\ell d_{k-1} \mu_{k-1} - \ell d_{k-1} \ell d_{k-2}, \\ e_k^2 &= -u r_{k+1} \mu_{k+1} - u r_{k+1} u r_{k+2}. \end{aligned}$$

Now substitute these together with (C) into (A) and rearrange to see the result.

The symmetric cases $i = k + 1$ and $i > k + 1$ are shown analogously. \square

The General Coupling Formulae

So far, so good, but still missing is the link between \mathbf{H} and (3.20). We need to decompose \mathbf{H} into two independent twisted factorizations that can in turn be interpreted as shifted normal equations.

Let a symmetric diagonal matrix $\mathbf{T} \in \mathbb{R}^{2n \times 2n}$ be given as

$$\mathbf{T} = \text{diag}_{\pm 1}(e_1, \dots, e_{2n-1}) - \text{diag}(\mu_1, \dots, \mu_{2n}).$$

The idea of course is that \mathbf{T} is the translate of a Golub-Kahan matrix, $\mathbf{T} = \mathbf{T}_{\text{GK}}(\mathbf{B}) - \mu$, but due to rounding errors the constant diagonal might have been violated. Suppose we have twisted factorizations for two twists k and $k + 1$:

$$\mathbf{T} = \mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^* = \mathbf{N}_{k+1} \mathbf{G}_{k+1} \mathbf{N}_{k+1}^*.$$

Now define \check{k}, \hat{k} such that $\{2\check{k}, 2\hat{k} - 1\} = \{k, k + 1\}$, and let $\bar{\mathbf{T}} := \mathbf{T} - 2\text{diag}[\mathbf{T}]$.

Lemma 3.17 gives comprehensive formulae for all elements of the matrix $\mathbf{H} := \mathbf{T}\bar{\mathbf{T}}$.

Recall (3.10) to see that the permutation $\mathbf{P}_{\text{ps}}^* \mathbf{H} \mathbf{P}_{\text{ps}}$ reorders the rows and the columns of \mathbf{H} by taking the even ones first and then the odd ones. With this insight it is actually not hard to deduce

$$\mathbf{P}_{\text{ps}}^* \mathbf{T} \bar{\mathbf{T}} \mathbf{P}_{\text{ps}} = \begin{bmatrix} \check{\mathbf{N}}_{\check{k}} \check{\mathbf{G}}_{\check{k}} \check{\mathbf{N}}_{\check{k}}^* & 0 \\ 0 & \hat{\mathbf{N}}_{\hat{k}} \hat{\mathbf{G}}_{\hat{k}} \hat{\mathbf{N}}_{\hat{k}}^* \end{bmatrix} + \begin{bmatrix} \check{\mathbf{\Omega}} & \mathcal{E} \\ \mathcal{E}^* & \hat{\mathbf{\Omega}} \end{bmatrix}, \quad (3.23)$$

where

$$\check{\mathbf{N}}_{\check{k}} \check{\mathbf{G}}_{\check{k}} \check{\mathbf{N}}_{\check{k}}^* \text{ is defined by } \begin{cases} \check{d}_i = -d_{2i} d_{2i+1}, & i = 1 : \check{k} - 1, \\ \check{\ell}_i = -\ell_{2i} \ell_{2i+1}, & i = 1 : \check{k} - 1, \\ \check{\gamma}_i = \mu_{2\check{k}} \gamma_{2\check{k}}, \\ \check{r}_i = -r_{2i-1} r_{2i}, & i = \check{k} + 1 : n, \\ \check{u}_i = -u_{2i-1} u_{2i}, & i = \check{k} + 1 : n, \end{cases} \quad (3.24)$$

and

$$\hat{\mathbf{N}}_{\hat{k}} \hat{\mathbf{G}}_{\hat{k}} \hat{\mathbf{N}}_{\hat{k}}^* \text{ is defined by } \begin{cases} \hat{d}_i = -d_{2i-1}d_{2i}, & i = 1 : \hat{k} - 1, \\ \hat{\ell}_i = -\ell_{2i-1}\ell_{2i}, & i = 1 : \hat{k} - 1, \\ \hat{\gamma}_{\hat{k}} = \mu_{2\hat{k}-1}\gamma_{2\hat{k}-1}, \\ \hat{r}_i = -r_{2i-2}r_{2i-1}, & i = \hat{k} + 1 : n, \\ \hat{u}_i = -u_{2i-2}u_{2i-1}, & i = \hat{k} + 1 : n. \end{cases} \quad (3.25)$$

For the error terms, $\check{\mathbf{\Omega}}, \hat{\mathbf{\Omega}} \in \mathbb{R}^{n \times n}$ are diagonal matrices with

$$\check{\mathbf{\Omega}}(i) \equiv \begin{cases} d_{2i}(\mu_{2i} - \mu_{2i+1}) + \ell\ell d_{2i-1}(\mu_{2i} - \mu_{2i-1}), & i < \check{k}, \\ \ell\ell d_{2\check{k}-1}(\mu_{2\check{k}} - \mu_{2\check{k}-1}) + u\ur r_{2\check{k}+1}(\mu_{2\check{k}} - \mu_{2\check{k}+1}), & i = \check{k}, \\ r_{2i}(\mu_{2i} - \mu_{2i-1}) + u\ur r_{2i+1}(\mu_{2i} - \mu_{2i+1}), & i > \check{k}, \end{cases} \quad (3.26)$$

$$\hat{\mathbf{\Omega}}(i) \equiv \begin{cases} d_{2i-1}(\mu_{2i-1} - \mu_{2i}) + \ell\ell d_{2i-2}(\mu_{2i-1} - \mu_{2i-2}), & i < \hat{k}, \\ \ell\ell d_{2\hat{k}-2}(\mu_{2\hat{k}-1} - \mu_{2\hat{k}-2}) + u\ur r_{2\hat{k}}(\mu_{2\hat{k}-1} - \mu_{2\hat{k}}), & i = \hat{k}, \\ r_{2i-1}(\mu_{2i-1} - \mu_{2i-2}) + u\ur r_{2i}(\mu_{2i-1} - \mu_{2i}), & i > \hat{k}, \end{cases}$$

and $\mathcal{E} \in \mathbb{R}^{n \times n}$ is upper bidiagonal with characterization

$$\mathsf{T}_{\text{GK}}(\mathcal{E})(i, i+1) \equiv e_{2i-1}(\mu_{2i-1} - \mu_{2i}),$$

that is, $\mathsf{T}_{\text{GK}}(\mathcal{E}) = \text{diag}_{\pm 1}[\mathsf{T}\bar{\mathsf{T}}]$.

The formulae (3.24) and (3.25), are what we call *coupling formulae*, *coupling relations* or also *coupling transformations*.

The Need for a Constant Diagonal

It is obvious that for the coupling relations to be meaningful, the central T must have a constant diagonal, because this is the only way to make the error terms $\check{\mathbf{\Omega}}, \hat{\mathbf{\Omega}}$ and \mathcal{E} vanish. Provided we have that, i.e., $\mathsf{T}(i, i) \equiv -\mu$, (3.23) becomes

$$\mathsf{P}_{\text{ps}}^* \mathsf{T}(\mathsf{T} + 2\mu) \mathsf{P}_{\text{ps}} = \begin{bmatrix} \check{\mathbf{N}}_{\check{k}} \check{\mathbf{G}}_{\check{k}} \check{\mathbf{N}}_{\check{k}}^* & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{N}}_{\hat{k}} \hat{\mathbf{G}}_{\hat{k}} \hat{\mathbf{N}}_{\hat{k}}^* \end{bmatrix}.$$

Thus the representations are perfectly coupled if the coupling relations (3.24) and (3.25) are fulfilled exactly. This gives the desired link between the eigenvectors. The local eigenvalues are then related according to

$$\check{\lambda}_i \equiv \lambda_i(2\mu + \lambda_i) \equiv \hat{\lambda}_i. \quad (3.27)$$

If the constant diagonal is not present, but can be attained by perturbing the data, i.e., if the central representation of T is ncd, then the stability of the

coupling formulae allows to find perturbations of the remaining data to get a perfect coupling. This is what we will usually aim for in practice. In the same manner one can deal with rounding errors that are introduced when executing the coupling transformations in floating-point arithmetic.

However, if the central representation is not ncd, then there is really no conceivable relation between the central and outer eigensystems. One cannot stress this fact enough. First of all we lose that T and $\bar{\mathsf{T}} = \mathsf{T} - 2\text{diag}[\mathsf{T}]$ are connected by a shift. And second, the way from $\mathsf{T}\bar{\mathsf{T}}$ to the outer representations $\check{\mathsf{N}}\check{\mathsf{G}}\check{\mathsf{N}}^*$ and $\hat{\mathsf{N}}\hat{\mathsf{G}}\hat{\mathsf{N}}^*$ is blocked by an absolute perturbation manifested in $\check{\Omega}$, $\hat{\Omega}$ and \mathcal{E} .

Using the Coupling Relations

In principle the coupling relations (3.24) and (3.25) can be used to compute all $8n - 3$ data elements of the three representations as soon as you know $4n - 1$ of them (you need to know μ for coupling from outside to inside).

Now we focus on the task to couple inside-out, that is, given the data for a twisted factorization of a GK-translate, we want to compute the data for the corresponding translates of the normal equations. This is the sole direction we will need later.

The coupling relations are deceptively simple and obviously stable. The essential problem, however, is this: To get coupled eigensystems we need that the coupling relations hold exactly for central matrices $\mathsf{N}_k\mathsf{G}_k\mathsf{N}_k^* = \mathsf{N}_{k+1}\mathsf{G}_{k+1}\mathsf{N}_{k+1}^*$ that *both* have a constant diagonal.

Suppose we have computed multiple twisted factorizations

$$\mathsf{M}_f - \nu = \mathsf{N}_k\mathsf{G}_k\mathsf{N}_k^*, \quad k = 1, \dots, 2n, \quad (3.28)$$

where M_f is the father representation and itself a descendant of a Golub-Kahan matrix. As a side note, if $\mathsf{N}_k\mathsf{G}_k\mathsf{N}_k^* = \mathsf{N}_{k+1}\mathsf{G}_{k+1}\mathsf{N}_{k+1}^*$ were to hold exactly for some k , then by considering the determinants one can easily prove [15, Thm. 3.1.18]

$$\gamma_k r_{k+1} = d_k \gamma_{k+1}. \quad (3.29)$$

It will also be useful to recall the data items that constitute the twisted factorizations $\mathsf{N}_k\mathsf{G}_k\mathsf{N}_k^*$ and $\mathsf{N}_{k+1}\mathsf{G}_{k+1}\mathsf{N}_{k+1}^*$. They have the $2n - 4$ elements

$$d_1, \ell_1, \dots, d_{k-1}, \ell_{k-1} \quad \text{and} \quad r_{k+2}, u_{k+2}, \dots, r_{2n}, u_{2n}$$

in common but differ in the remaining three:

$$\gamma_k, u_{k+1}, r_{k+1} \quad \text{for} \quad \mathsf{N}_k\mathsf{G}_k\mathsf{N}_k^*, \quad d_k, \ell_k, \gamma_{k+1} \quad \text{for} \quad \mathsf{N}_{k+1}\mathsf{G}_{k+1}\mathsf{N}_{k+1}^*.$$

It is tempting to just select a pair $\{k, k + 1\}$ of twists in (3.28) and use the coupling relations as stated on the combined data of $\mathsf{N}_k\mathsf{G}_k\mathsf{N}_k^*$ and $\mathsf{N}_{k+1}\mathsf{G}_{k+1}\mathsf{N}_{k+1}^*$.

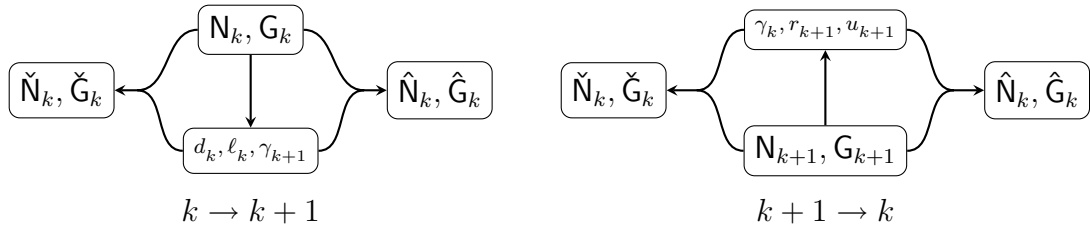


Figure 3.7: Possible ways to couple data from one central twist k to the outside.

However, doing so is a serious mistake. To obtain (3.28) one probably used some variant of `dtwqds`, depending on the representation type of \mathbf{M}_f . Then the shift relations will only hold exactly for perturbed data, and the required source-perturbation of \mathbf{M}_f will generally differ for each k . Thus there is no direct relation between $\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$ and $\mathbf{N}_{k+1} \mathbf{G}_{k+1} \mathbf{N}_{k+1}^*$, and (3.29) will usually not hold at all. This concern is not just academic: When we started experiments with twisted factorizations in the GK-layer using the formulae in this “naive” way, we got horrible results, and precisely this was the cause.

The coupling formulae as given above were stated previously by Großer & Lang in [35–37], and we repeated them in [72], without ever mentioning this problem, simply because it was not known to exist. The reason that this omission caused no trouble up to now is that the couplings were ever only used for plain bidiagonal factorizations ($k = 1$ or $k + 1 = 2n$), where the formulae can be simplified to require only the data for one twist index. We will now derive these special cases and also develop a work-around to facilitate the use of arbitrary twists.

The insight that $\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^* = \mathbf{N}_{k+1} \mathbf{G}_{k+1} \mathbf{N}_{k+1}^*$ will typically not hold for the computed data forces us to fixate on only one of them and explicitly *recompute* the required data for the other.

We must be able to extend the perturbation that establishes a constant diagonal at the source to the computed data for the second central twist, and from there along the coupling relations to the outer representations. In general this is only possible if the data dependency from $\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$ to $\mathbf{N}_{k+1} \mathbf{G}_{k+1} \mathbf{N}_{k+1}^*$ —or the other way around if we take $k + 1$ as source—is well-conditioned.

The two possible data flows are depicted in Figure 3.7. For the direction $k \rightarrow k + 1$ the missing data for $\mathbf{N}_{k+1} \mathbf{G}_{k+1} \mathbf{N}_{k+1}^*$ is defined by

$$d_k = \gamma_k + \overline{u}ur_{k+1}, \quad \ell_k = \frac{ur_{k+1}}{d_k}, \quad \gamma_{k+1} = \frac{r_{k+1}}{d_k} \gamma_k,$$

where the latter follows from (3.29) and the other two from (2.18) and (2.20). This relation is well-conditioned if there is at most benign cancellation in $\gamma_k + \overline{u}ur_{k+1}$. For the case $k = 1$ we can do better, because if $\tilde{\mathbf{N}}_k \tilde{\mathbf{G}}_k \tilde{\mathbf{N}}_k^*$ has constant diagonal $-\mu$, then $\tilde{\gamma}_1 + \overline{u}ur_2 = -\mu$, so the definition of d_k above can be replaced by $d_1 = -\mu$.

The other direction $k + 1 \rightarrow k$ works analogously with

$$r_{k+1} = \ell d_k + \gamma_{k+1}, \quad u_{k+1} = \frac{\ell d_k}{r_{k+1}}, \quad \gamma_k = \frac{d_k}{r_{k+1}} \gamma_{k+1}.$$

Now applicability hinges on the condition of $\ell d_k + \gamma_{k+1}$. For the case $k + 1 = 2n$ we can again simplify to $r_{2n} = -\mu$.

To summarize, coupling from an arbitrary twisted factorization $\mathbf{N}_k \mathbf{G}_k \mathbf{N}_k^*$ in the GK-layer is only possible if for at least one of $t \in \{k - 1, k + 1\}$ the relation to the missing data of $\mathbf{N}_t \mathbf{G}_t \mathbf{N}_t^*$ is well-conditioned. Thus we will have to filter possible twist locations with this restriction in mind. This is no serious loss, because one can show that if neither one of the directions $k \rightarrow k - 1$ or $k \rightarrow k + 1$ works in this regard, then $|\gamma_k|$ must be fairly large, so twisting at k would not be desirable anyway. Coupling based on $k = 1$ or $k = 2n$ is always possible.

3.5 The Algorithm of Multiple Coupled RRRs

Preparations are done, preliminary analyses are complete, and we can now embark on presenting our version of the MR³-based algorithm for BSVD, which we regard as one of the centerpieces of this thesis.

A short historical overview. The fundamental idea as proposed by Großer & Lang in [35–37] was to run MR³ on the three roots $\mathbf{B}\mathbf{B}^*$, $\mathbf{T}_{\text{GK}}(\mathbf{B})$ and $\mathbf{B}^*\mathbf{B}$ simultaneously, employing the coupling relations to do so in a synchronized fashion. This was done at a time where MR³ had just been baptized. The then current implementation in LAPACK 3.0 was still the first one, which handled interior clusters by calling `xSTEIN`.

In [72] we described some refinements to the method which were mainly based on integrating and adapting to new developments intended for the next LAPACK's MR³ implementation `xSTEGR`. But as we started more intensive testing too many problems cropped up. For a rather long time we had a method that worked not well enough to be called stable, while at the same time being too good to be just coincidence, and we did not really know why.

Großer & Lang had not provided a rigorous error analysis, but they had given a result that worked as motivation for the coupled algorithm's viability (Theorem 5.2 in [37]). Naturally, the occurring problems did in some way contradict that theory. And indeed, after further study we found a very subtle error in the published proof, which we will elaborate on in §3.5.2. Essentially it had been neglected that the coupling relations are meaningful really only in concert with a constant diagonal. The need for a constant diagonal—which must seem trivial in retrospective—arose from that realization. We really had not thought about this before. But it is easily checkable a priori in practice, and doing so quickly confirmed that loss of the property was a main cause for the problems.

Harnessing the constant diagonal (or the related GK-structure) we could develop the better understanding of the black-box approaches that was given in §3.2 and §3.3. Furthermore it turned out to be the missing link to prove that MR^3 on the Golub-Kahan matrix does work in principle and that in turn made a rigorous proof of the coupled algorithm possible.

What we ultimately strive for is a method that combines the good parts of the black-box approaches and shuns the bad ones. This means the orthogonality levels of the singular vectors should fulfill (3.3), the residual norms should fulfill (3.4), and runtime should preferably be close to using MR^3 on just *one* of the tridiagonal problems BB^* or B^*B , and at the utmost twice that. The original method envisioned by Großer & Lang does provide these features in principle, except for the unforeseen reliability problems in practice.

We regard the following points as our main contributions to further the method:

- A simpler and streamlined formulation. In particular we completely removed the special treatment of first-level nodes that was still present in [72]. At the same time the formulation is general enough to encompass many conceivable variations of the main theme (like what kind of representations to use, how to execute the shifts, etc.).
- A rigorous proof of correctness, giving concise error bounds. It builds on the requirements for MR^3 , which means expertise and implementation technology of MR^3 can be maximally reused. There is one new requirement, but that can be checked a priori. A particularly important feature of the proof is that it does *not* require the representations in the GK-layer to be relatively robust, which to guarantee in practice is problematic at best.

3.5.1 Description of the Algorithm

The coupled approach as it was outlined before leads naturally to the fact that we have to build up the three representation trees rooted at BB^* , $\text{T}_{\text{GK}}(\text{B})$ and B^*B in a consistent and synchronized way. In particular they have to be structurally compatible, meaning there must be a one-to-one correspondence between nodes and index sets. But if this is given, we can think of the trees as being fused together at the nodes to become one tree with *3-layered nodes*. Thus, the nodes

$$\begin{aligned} (\check{\text{M}}, I, \bar{\mu}^2) & \text{ from the tree rooted at } \text{BB}^*, \\ (\text{M}, I, \bar{\mu}) & \text{ from the tree rooted at } \text{T}_{\text{GK}}(\text{B}), \\ (\hat{\text{M}}, I, \bar{\mu}^2) & \text{ from the tree rooted at } \text{B}^*\text{B}, \end{aligned}$$

are combined into one single node

$$([\check{\text{M}}, \text{M}, \hat{\text{M}}], I, \bar{\mu}).$$

In other words, we now have three representations attached to each node. The bar in $\bar{\mu}$ denotes that it stands for accumulated shifts from the root $T_{\text{GK}}(\mathbf{B})$.

With the concept of these “super-nodes”, the coupling-based approach can be described in general terms and very “MR³-like” as is shown in Algorithm 3.3. We would like to coin it Algorithm of *Multiple Coupled Relatively Robust Representations*, or MCR³ in short.

One core principle is that all expensive computations, i.e., refining eigenvalues and computing eigenvectors, are done only for the smaller matrices $\tilde{\mathbf{M}}$ and $\hat{\mathbf{M}}$. The representation \mathbf{M} in the central GK-layer is only there to “knit” the other two together.

The algorithm is designed around the assumption that the three representations at each node are perfectly coupled (or at least their local eigensystems should be). The proof later will show that it actually suffices if this can be attained for perturbed data, but assume for now it holds exactly. Then the local eigenvalues obey

$$\tilde{\lambda}_i = \lambda_i(2\bar{\mu} + \lambda_i) = \hat{\lambda}_i. \quad (3.30)$$

To keep this connection when going down one level, the outer and central shifts μ and τ that are selected in steps 13 and 14 need to obey

$$(\bar{\mu} + \mu)^2 = \bar{\mu}^2 + \tau \Rightarrow \mu^2 + 2\bar{\mu}\mu - \tau = 0.$$

Because we have refined the spectrum for the outer layers, it is sensible to choose τ with respect to that refinement first and then compute μ such that the above is fulfilled, i.e., as smaller root of $x^2 + 2\bar{\mu}x - \tau$. The nature of shifting within MR³ implies that one usually has $|\tau| \ll \bar{\mu}^2$, and it is not hard nor overly hindering to restrict the choice of τ to enforce this. Then computing μ in line 13 using

$$\mu := \tau / (\bar{\mu} + \sqrt{\bar{\mu}^2 + \tau})$$

is guaranteed to be stable.

The most critical part is how the shifts are executed in step 14. Doing them separately would amount to MR³ on the normal equations, and we know that does not work. Thus we will have to do just a part of the factorizations explicitly, and set up the remaining data using the coupling transformations from §3.4. Further discussion on how to shift will be postponed until after the upcoming proof, where specific criteria to guide the decision are developed.

3.5.2 Proof of Correctness

We will now set out to prove that Algorithm 3.3 does compute singular triplets satisfying (3.3) and (3.4). With the intent on providing perspective we will first revisit the existing theory from [35–37], for which we unfortunately have to point out an error in the principal result concerning correctness.

ALGORITHM 3.3 *Algorithm MCR³*

Compute specified singular triplets of bidiagonal \mathbf{B} using couplings to effectively run MR³ on $\mathbf{B}\mathbf{B}^*$, $\mathbf{T}_{\text{GK}}(\mathbf{B})$ and $\mathbf{B}^*\mathbf{B}$ simultaneously.

Input: Upper bidiagonal $\mathbf{B} \in \mathbb{R}^{n \times n}$. Index set $I_0 \subseteq \{1, \dots, n\}$.

Output: Singular triplets $(\bar{\sigma}_i, \bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i), i \in I_0$.

Params: *gaptol* : The Gap Tolerance.

```

1:   $\mathcal{S} := \{([\mathbf{B}\mathbf{B}^*, \mathbf{T}_{\text{GK}}(\mathbf{B}), \mathbf{B}^*\mathbf{B}], I_0, \bar{\mu} = 0)\}$ 
2:  while  $\mathcal{S} \neq \emptyset$  do
3:      Remove one node  $([\check{\mathbf{M}}, \mathbf{M}, \hat{\mathbf{M}}], I, \bar{\mu})$  from  $\mathcal{S}$ .
4:      Approximate eigenvalues  $[\check{\lambda}_i^{\text{loc}}], i \in I$  of  $\check{\mathbf{M}}$  such that they can be classified into singletons and clusters according to gaptol; this gives a partition  $I = I_1 \cup \dots \cup I_m$ .
5:      for  $r = 1$  to  $m$  do
6:          if  $I_r = \{i\}$  then                                     // singleton
7:              Refine the eigenvalue approximation  $[\check{\lambda}_i^{\text{loc}}]$  if necessary and use it to compute eigenvector  $\check{\mathbf{u}}_i$  of  $\check{\mathbf{M}}$ .
8:              Take  $[\hat{\lambda}_i^{\text{loc}}] := [\check{\lambda}_i^{\text{loc}}]$  as initial approximation, adjust/refine if necessary and use it to compute eigenvector  $\hat{\mathbf{v}}_i$  of  $\hat{\mathbf{M}}$ .
9:              Compute  $\bar{\sigma}_i := \check{\mathbf{u}}_i^* \mathbf{B} \bar{\mathbf{v}}_i$ . If this results in  $\bar{\sigma} < 0$ , negate  $\bar{\sigma}_i$  and  $\bar{\mathbf{v}}_i$ .
10:         else                                               // cluster
11:             Refine the eigenvalue approximations at the borders (and/or inside) of the cluster if desired for more accurate shift-selection.
12:             Choose a suitable (outside-layer) shift  $\tau$  near the cluster.
13:             Transform to an inside shift  $\mu := \tau / (\bar{\mu} + \sqrt{\bar{\mu}^2 + \tau})$ .
14:             Compute representations  $\check{\mathbf{M}}^+ = \check{\mathbf{M}} - \tau$ ,  $\mathbf{M}^+ = \mathbf{M} - \mu$  and  $\hat{\mathbf{M}}^+ = \hat{\mathbf{M}} - \tau$  in a synchronized way.
15:             Add new node  $([\check{\mathbf{M}}^+, \mathbf{M}^+, \hat{\mathbf{M}}^+], I_r, \bar{\mu} + \mu)$  to  $\mathcal{S}$ .
16:         endif
17:     endfor
18: endwhile

```

An Error in Previous Work

Großer & Lang did not provide a rigorous proof or error bounds for the coupling approach, but they gave a strong principal argument to indicate validity of the method.

The first part of their argument is based on the experimental studies with the black-box approach on the normal equations that led to the following

Conjecture: If the local eigenvalues of the outer representations at all nodes are close in a relative sense, then the computed singular vectors will fit together well, i.e., have small residual norms.

A geometric argument was used as support [36, §4.3]. We do not see a conceivable way to prove the conjecture rigorously, especially taking the new insights from §3.2 into account. Rather it seems that the eigenvalues being close is a necessary condition for good results, but not a sufficient one. Nevertheless the principal sentiment remains intuitively convincing.

The second part of the argument comes in form of a theorem. We have adjusted the formulation to our notation, in particular we use different accents.

Theorem 5.2 in [37]: Assume \mathbf{M} is the representation in the GK-layer, and $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$ are computed from \mathbf{M} by executing the coupling transformations in floating-point arithmetic. If \mathbf{M} is an RRR for its local eigenvalues in I , then so are $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$, and their local eigenvalues will be relatively close.

To combine these two pillars, they assume the representations in the GK-layer are RRRs of sufficient quality. Provided this is so, basic correctness of the method would indeed follow.

Unfortunately, the argument is flawed because of a subtle error in the proof of Theorem 5.2 in [37]. For demonstration we will redo the main steps of the proof in a compressed version and then highlight where reasoning breaks down.

The proof starts by exploiting the stability of the coupling transformations. Clearly one can find a perturbation $\mathbf{M} \rightsquigarrow \bar{\mathbf{M}}$ of the central representation such that $\check{\mathbf{M}}$, $\bar{\mathbf{M}}$ and $\hat{\mathbf{M}}$ fulfill the coupling relations (3.24) and (3.25) exactly. For this one just has to account for the rounding errors due to executing the coupling transformations in floating-point arithmetic. An analogous argument actually allows to choose any elementwise relative perturbations $\check{\mathbf{M}} \rightsquigarrow \mathbf{M}$, $\hat{\mathbf{M}} \rightsquigarrow \bar{\mathbf{M}}$ of the outer representations and still find a suitable perturbation of $\mathbf{M} \rightsquigarrow \bar{\mathbf{M}}$, such that $\check{\mathbf{M}}$, $\bar{\mathbf{M}}$ and $\hat{\mathbf{M}}$ fulfill the coupling transformations exactly, too.

Then they argue that because the exact coupling relations hold, the local eigenvalues between the layers are related as in (3.30), that is

$$\check{\lambda}_i = \bar{\lambda}_i(2\bar{\mu} + \bar{\lambda}_i) = \hat{\lambda}_i, \quad \text{and} \quad \check{\lambda}_i = \tilde{\lambda}_i(2\bar{\mu} + \tilde{\lambda}_i) = \hat{\lambda}_i, \quad (\star)$$

where the (six kinds of) accents on λ may convey the matrix it is supposed to be an eigenvalue of. Now use that \mathbf{M} is an RRR to see $\bar{\lambda}_i \approx \tilde{\lambda}_i$, in a relative sense. Using $|\bar{\lambda}_i| \ll |\bar{\mu}|$ one can then show $\check{\lambda}_i \approx \tilde{\lambda}_i$ and $\hat{\lambda}_i \approx \tilde{\lambda}_i$. Because this reasoning should hold for any perturbations $\check{\mathbf{M}} \rightsquigarrow \mathbf{M}$, $\hat{\mathbf{M}} \rightsquigarrow \mathbf{M}$, one can conclude that $\check{\mathbf{M}}$ and $\hat{\mathbf{M}}$ are indeed RRRs, as claimed.

The flaw of the argument is (\star) . The implication that exact fulfillment of the coupling relations gives the eigenvalue relation (3.30) is only true if the central matrix has a constant diagonal, see our remarks on page 147. But there is a perturbation between $\bar{\mathbf{M}}$ and $\tilde{\mathbf{M}}$, so in general at most one of them will have its diagonal being constant. Hence, one of the two equalities postulated in (\star) will in general not hold.

There is a variant of the argumentation pertaining to an outside-in coupling strategy for level-one nodes in Theorem 5.4 of [37], but its proof redirects to the one above and as such the claim is cast in doubt as well.

Actually we cannot disprove the claim of Theorem 5.2 in [37] either. However, mainly due to practical experience we doubt that it holds, at least in this form. On the other hand, the coupling transformations clearly indicate that having no or only moderate element growth in the central representation carries over to some extent to the outer representations, so at least the first part claiming the outer representations to be RRRs might be true.

A New and Rigorous Proof

For proving Algorithm MCR³ to be correct we will use the same basic, by now well worn-in approach that we used for MR³ and Algorithm 3.2. Again we identify reasonable requirements (two this time) that have to be fulfilled and upon which the formal argumentation can rest.

Recall the terms *coupled representations* and *coupled eigensystems* from Definitions 3.2 and 3.3.

Requirement TREE (Outer Representation Trees)

The “outer” representation trees with roots \mathbf{BB}^* and $\mathbf{B}^*\mathbf{B}$ fulfill all five MR³-requirements on page 55, except maybe for ELG.

◇

Requirement CPL (Nearly Coupled Representations)

There exist constants η_{cpl} , γ_{cpl} and β_{cpl} such that for each node $([\check{M}, M, \hat{M}], I, \bar{\mu})$ in the 3-layered tree one can find perturbations

$$\check{M} = \text{erp}(\check{M}, \eta_{\text{cpl}}), \quad \tilde{M} = \text{erp}(M, \gamma_{\text{cpl}}), \quad \hat{M} = \text{erp}(\hat{M}, \eta_{\text{cpl}}),$$

with the property that \check{M} , \tilde{M} and \hat{M} are perfectly coupled and

$$\tilde{M} = T_{\text{GK}}(\tilde{B}) - \bar{\mu}, \quad \text{where } \tilde{B} = \text{erp}(B, \beta_{\text{cpl}}).$$

◇

Before we begin the proof, some words about the ramifications of these requirements.

TREE is just what we would like to have when using MR³ on the normal equations separately. Concerning steps 12–14 we can employ all tests and heuristics from MR³ to evaluate good shift candidates, but have to apply them to both child representations \check{M}^+ and \hat{M}^+ .

Requirement CPL is basically fulfilled when the representations in the GK-layer are ncd and we can perturb the three representations at each node suitably such that the coupling relations hold. One could modify the following proof so that it works if the need for a nearly constant diagonal is relaxed to requiring only that the local subspaces are coupled.

As always we may assume the computed vectors to be perfectly normalized, that is, $\|\bar{u}_i\| \equiv \|\bar{v}_i\| \equiv 1$.

Theorem 3.18 (Orthogonality for Algorithm MCR³)

Let Algorithm 3.3 be executed such that requirement TREE is fulfilled. Then the computed left and right singular vectors will obey

$$\frac{1}{2} \max \{ \bar{u}_i^* \bar{u}_j, \bar{v}_i^* \bar{v}_j \} \leq \text{orth}, \quad i \neq j \in I_0,$$

where *orth* comprises the $\mathcal{O}(n\epsilon_\diamond)$ rhs-bound from Theorem 2.14.

Proof. This is just Theorem 2.14 applied twice. Refer to Figure 2.1 to verify that Requirement ELG is not needed for orthogonality. □

Establishing a bound for the residual norms will be harder, because we have to combine vectors that were computed from different matrices. The proof will mimic the analysis from §3.2 up to a point. To obtain the final result we will have to invoke the powerful lower bound from the Gap Theorem 1.22.

Theorem 3.19 (Residual norms for Algorithm MCR³)

Let Alg. 3.3 be executed such that requirements TREE and CPL are fulfilled. Then the computed singular triplets $(\bar{\sigma}_i, \bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i)$ will satisfy, to first order,

$$\max \left\{ \|\mathbf{B}\bar{\mathbf{v}}_i - \bar{\sigma}_i \bar{\mathbf{u}}_i\|, \|\mathbf{B}^* \bar{\mathbf{u}}_i - \bar{\sigma}_i \bar{\mathbf{v}}_i\| \right\} \stackrel{\text{1st}}{\leq} 2\|\mathbf{B}\|(\Delta + n(2\beta_{\text{cpl}} + \epsilon_\diamond)),$$

where $\Delta = \mathcal{R}n\epsilon_\diamond + C_{\text{vecs}}n\eta_{\text{cpl}}/\text{gaptol}$ and with \mathcal{R} as defined in Lemma 2.11.

Proof. Consider one particular index i and let $([\check{\mathbf{M}}, \mathbf{M}, \hat{\mathbf{M}}], I, \bar{\mu})$ be the node where it is computed, i.e., where $\{i\} \subseteq I$ is a singleton child index set of I . We will drop the index and just write $\bar{\mathbf{u}}, \bar{\mathbf{v}}$ and $\bar{\sigma}$ instead of $\bar{\mathbf{u}}_i, \bar{\mathbf{v}}_i$ and $\bar{\sigma}_i$. Because of requirement CPL we can find perturbed matrices

$$\check{\mathbf{M}} \rightsquigarrow \tilde{\mathbf{M}}, \quad \mathbf{M} \rightsquigarrow \tilde{\mathbf{M}}, \quad \hat{\mathbf{M}} \rightsquigarrow \tilde{\mathbf{M}},$$

that are perfectly coupled. Let $\tilde{\mathbf{u}}, \tilde{\mathbf{q}}_\pm$ and $\tilde{\mathbf{v}}$ denote their respective eigenvectors for index i (or $\pm i$ in the case $\tilde{\mathbf{M}}$). Figure 3.8 summarizes these entities and sketches their interplay for the proof.

The computations of $\bar{\mathbf{u}}$ at $\check{\mathbf{M}}$ and of $\bar{\mathbf{v}}$ at $\hat{\mathbf{M}}$ fulfill the conditions of Lemma 2.11, and this is not affected by a possible negation of $\bar{\mathbf{v}}$ in step 9. Furthermore, both outer representations fulfill Requirements RRR and RELGAPS; thus their i th eigenpair has relative separation exceeding gaptol and is determined to high relative accuracy. Thence we know

$$\begin{aligned} \bar{\mathbf{u}} &= \xi \tilde{\mathbf{u}} + \mathbf{r}, & \tilde{\mathbf{u}} &\perp \mathbf{r}, & |\xi|^2 + \|\mathbf{r}\|^2 &= 1, & \|\mathbf{r}\| &\leq \Delta \ll 1, \\ \bar{\mathbf{v}} &= \nu \tilde{\mathbf{v}} + \mathbf{s}, & \tilde{\mathbf{v}} &\perp \mathbf{s}, & |\nu|^2 + \|\mathbf{s}\|^2 &= 1, & \|\mathbf{s}\| &\leq \Delta \ll 1. \end{aligned} \quad (\star)$$

The only link between them is the central layer's representation \mathbf{M} , or more precisely, its perturbed version $\tilde{\mathbf{M}}$. To get there, we combine the computed vectors to form

$$\sqrt{2}\mathbf{P}_{\text{ps}}^* \bar{\mathbf{q}} := \begin{bmatrix} \bar{\mathbf{u}} \\ \bar{\mathbf{v}} \end{bmatrix} = \begin{bmatrix} \xi \tilde{\mathbf{u}} + \mathbf{r} \\ \nu \tilde{\mathbf{v}} + \mathbf{s} \end{bmatrix}.$$

Using (\star) we can write $\bar{\mathbf{q}}$ in terms of $\tilde{\mathbf{M}}$'s eigenvectors $\sqrt{2}\mathbf{P}_{\text{ps}}^* \tilde{\mathbf{q}}_\pm = \begin{bmatrix} \tilde{\mathbf{u}} \\ \pm \tilde{\mathbf{v}} \end{bmatrix}$ as

$$\bar{\mathbf{q}} = \frac{1}{2}(\xi - \nu)\tilde{\mathbf{q}}_- + \frac{1}{2}(\xi + \nu)\tilde{\mathbf{q}}_+ + \frac{1}{\sqrt{2}}\mathbf{P}_{\text{ps}} \begin{bmatrix} \mathbf{r} \\ \mathbf{s} \end{bmatrix}. \quad (\star\star)$$

So far we have followed the same path of reasoning as in §3.2 and one might expect to run into the same problems. But here we are dealing singletons and

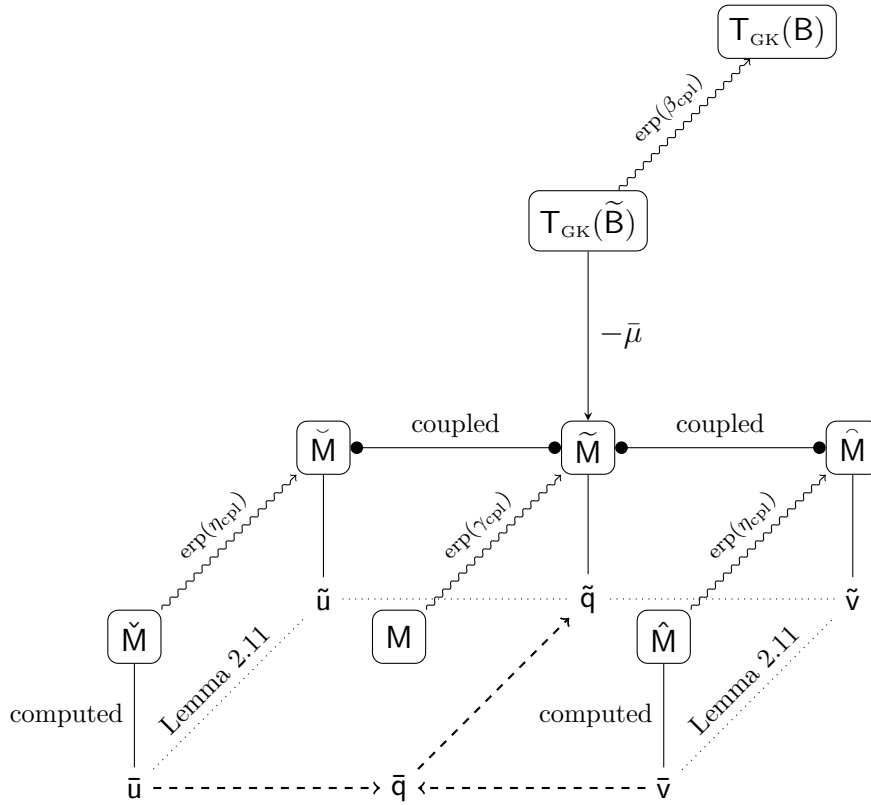


Figure 3.8: Outline of the proof of Theorem 3.19. Dashed lines indicate how the computed vectors are connected to exact eigenvectors of a perturbed GK-root.

one-dimensional eigenspaces; in particular we only have one weight ξ and ν on each side. This fact will now let us deduce that exactly one of the factors $\xi - \nu$ and $\xi + \nu$ in $(\star\star)$ is negligible. From (\star) we see

$$\begin{aligned} 1 - \Delta^2 &\leq 1 - \|r\|^2 = \xi^2 \leq 1, \\ 1 - \Delta^2 &\leq 1 - \|s\|^2 = \nu^2 \leq 1. \end{aligned}$$

Hence $|\xi^2 - \nu^2| = |\xi - \nu||\xi + \nu| \leq \Delta^2$. Because $\Delta \ll 1$, one of the factors $|\xi - \nu|$ and $|\xi + \nu|$ must lie in the interval $[1, 2]$, very close to 2, whereas the other is tiny, that is,

$$\begin{aligned} \max \{|\xi - \nu|, |\xi + \nu|\} &\geq 2\sqrt{1 - \Delta^2} \\ \min \{|\xi - \nu|, |\xi + \nu|\} &\leq \Delta^2. \end{aligned}$$

The roles of the factors are determined by the signs of ξ and ν ; it is not important which is which. They depend on how \bar{v} was signed as computed eigenvector of \hat{M} in line 8 (a detail we have no direct influence on), as well as the explicit sign

choice in line 9 to get $\bar{\sigma}$ nonnegative. In fact, one could show that the latter makes $|\xi - \nu|$ the small one, but we do not need that.

In any case, using $(\star\star)$ we conclude that $\bar{\mathbf{q}}$ must have a small angle to *exactly one* of $\tilde{\mathbf{q}}_-$ or $\tilde{\mathbf{q}}_+$, that is,

$$\begin{aligned} \min \{ \sin \angle(\tilde{\mathbf{q}}_-, \bar{\mathbf{q}}), \sin \angle(\tilde{\mathbf{q}}_+, \bar{\mathbf{q}}) \} \\ \leq \frac{1}{\sqrt{2}} \left\| \begin{bmatrix} r \\ s \end{bmatrix} \right\| + \frac{1}{2} \min \{ |\xi - \nu|, |\xi + \nu| \} \leq \Delta + \frac{1}{2} \Delta^2. \end{aligned}$$

To close the argument, we need to employ the second property given by Requirement CPL, namely that $\tilde{\mathbf{M}}$ is an exact translate of a perturbed GK-root. Thus we have $\tilde{\mathbf{M}} = \tilde{\mathbf{T}} - \tilde{\mu}$, where

$$\tilde{\mathbf{T}} := \mathbf{T}_{\text{GK}}(\tilde{\mathbf{B}}), \quad \tilde{\mathbf{B}} := \text{erp}(\mathbf{B}, \beta_{\text{cpl}}).$$

Hence, $\tilde{\mathbf{q}}_{\pm}$ are exact eigenvectors of $\tilde{\mathbf{T}}$, and the computed $\bar{\mathbf{q}}$ is close to one of them. That is it, actually, the rest is just cleanup.

The lower bound that is provided by the Gap Theorem 1.22 yields

$$\|\tilde{\mathbf{T}}\bar{\mathbf{q}} - (\bar{\mathbf{q}}^* \tilde{\mathbf{T}} \bar{\mathbf{q}}) \bar{\mathbf{q}}\| \leq \text{spdiam}[\tilde{\mathbf{T}}](\Delta + \frac{1}{2} \Delta^2).$$

The transition $\mathbf{B} \rightsquigarrow \tilde{\mathbf{B}}$ is harmless in the sense $\|\tilde{\mathbf{T}}\| = \|\tilde{\mathbf{B}}\| \leq \|\mathbf{B}\|(1 + 2n\beta_{\text{cpl}})$. With $\mathbf{T} := \mathbf{T}_{\text{GK}}(\mathbf{B})$ we get, to first order,

$$\|\mathbf{T}\bar{\mathbf{q}} - (\bar{\mathbf{q}}^* \mathbf{T} \bar{\mathbf{q}}) \bar{\mathbf{q}}\| \stackrel{1\text{st}}{\leq} 2\|\mathbf{B}\|(\Delta + 2n\beta_{\text{cpl}}).$$

Note that $\bar{\mathbf{q}}^* \mathbf{T} \bar{\mathbf{q}} = \bar{\mathbf{u}}^* \mathbf{B} \bar{\mathbf{v}}$. So, padding the bound by a further $2\|\mathbf{B}\|n\epsilon_{\diamond}$ to account for the computation of $\bar{\sigma}_i$ in floating-point arithmetic gives the result. \square

3.5.3 Shifting coupled nodes

So far we left out completely how the shifting of the 3-layered nodes in step 14 of Algorithm 3.3 is to be done. From the previous theory we know that two conditions are to be met:

- SHIFTRREL must hold for the outer trees, that is, the transitions $\check{\mathbf{M}}^+ = \check{\mathbf{M}} - \tau$ and $\hat{\mathbf{M}}^+ = \hat{\mathbf{M}} - \tau$ should behave as if they were done by a mixed relatively stable method.
- CPL must hold in the central layer, that is, the data of $\check{\mathbf{M}}$, $\hat{\mathbf{M}}$ and \mathbf{M} must be related such that we can find perturbed versions of them that are perfectly coupled ($\check{\mathbf{M}}$ has constant diagonal and the coupling relations are satisfied exactly).

There is no obvious way to consistently achieve both goals at the same time in practice. The main avenues one can pursue are:

central Do the shifting purely in the GK-layer and couple to the outside representations:

- (1) Compute $M^+ = M - \mu$ using a stable method like `dtwqds` or some variant. This should be done carefully to keep the ncd-property whenever possible.
- (2) Employ the coupling transformations (3.24) and (3.25) to set up the data for \check{M}^+ and \hat{M}^+ .

outside Do the shifting with the outside data and couple to obtain the central representation:

- (1) Factorize one of $\check{M} - \tau = \check{M}^+$ or $\hat{M} - \tau = \hat{M}^+$ explicitly.
- (2) Use the coupling relations, together with the fact that M^+ should be $\text{ncd}(-\bar{\mu} - \mu)$, to set up the data for M^+ and the other one of \check{M} , \hat{M} .

hybrid Some combination of **central** and **outside**, using the data of all three representations \check{M} , M and \hat{M} at once and maybe switching between factorizing on the outside and on the inside dynamically on an index-by-index basis, depending on where the problems arise.

The approaches **central** and **outside** were already suggested by Großer & Lang in [37, Sec. 5.4], [36, Alg. 3.2]. In [72] we also used a **central** strategy with special treatment for the transition to the first level.

During the work on this thesis we have invested a long time of fruitless research trying to find some kind of magic factorization method for shifting 3-layered nodes that guarantees both properties in all cases. For that task we failed utterly. The work was not completely wasted, though, because much of the gained experience with qd-like factorization algorithms and their mixed relative error analysis led in turn to the improvements in §2.4, §3.4 and Chapter 4.

On the previous pages we presented what we heralded as proof of correctness for Algorithm MCR³. Now one could argue that this proof is worthless if we cannot attain its preconditions in practice. But that is not so, because the proof provides understanding and concrete criteria to steer a computation. The situation is similar to like Requirement RRR for MR³, which cannot be guaranteed a priori either.

The new insight from §3.3 that MR³ on the Golub-Kahan matrix is not that fundamentally doomed as it was long thought to be has caused us to shed the fears of working with GK-translates explicitly. Therefore the strategy we will use henceforth is exclusively **central**. There is a series of strong arguments to be fielded in support of this decision:

- A) Violation of SHIFTRREL is usually detectable in practice. This is because one uses the local eigenvalue approximations to get initial approximations $[\check{\lambda}_i^{\text{loc}}] - \tau$ and $[\hat{\lambda}_i^{\text{loc}}] - \tau$ for the children (after inflating them a bit), cf. Remark 2.6. If SHIFTRREL does not hold then these are highly unlikely to be consistent, and bisection can detect this.
- B) If the representations in the GK-layer are RRRs, we can give an alternative proof that the results will be fine—without requiring SHIFTRREL—by combining the proofs of Theorems 3.10 and 3.19. Thus we have a backup plan to success, should SHIFTRREL fail to hold.
- C) In our experience the ncd-property in the GK-translates is crucial. Doing the factorizations there explicitly provides more control, in particular because we can guarantee a nearly constant diagonal as long as local element growth is kept in check.
- D) In Chapter 4 we will develop new techniques to work with twisted factorizations that allow for 2×2 -blocks as pivots. These are especially well suited to compute successive translates of a Golub-Kahan matrix and provide much finer-grained control over local element growth. As such they drastically reduce the problems with both the ncd-property and relative robustness in the GK-layer.

3.6 Numerical Results

In this chapter we have intensively studied the theoretical aspects of two methods, Algorithms 3.2 (MR^3 on T_{GK}) and 3.3 (MCR^3), for computing the singular value decomposition of a bidiagonal matrix. Now is the time to demonstrate how well they perform in practice. To do so we use the same sets **Pract** and **Synth** of bidiagonal test matrices that were introduced in §2.5.

We have implemented both methods within the same software framework that accommodated our MR^3 -implementation **XMR**. In fact, due to the object-oriented design most of the code could be reused from **XMR**. This includes the optimized configurations of bisection and RQI. Let us denote the implementations of Algorithms 3.2 and 3.3 as **XMR-TGK** and **XMR-CPL**, respectively.

XMR-TGK is basically **XMR**, adapted to use $\text{T}_{\text{GK}}(\mathbf{B})$ as root representation. We switched to using Z -representations for the children to cushion the effect of moderate element growth on the diagonal. A shift candidate has to be $\text{ncd}(-\bar{\mu}, 32n\epsilon_\circ)$, additionally to conditions (2.81) and (2.82), in order to be considered acceptable a priori.

The implementation **XMR-CPL** of Algorithm MCR^3 is a natural mixture of **XMR** and **XMR-TGK**, based on a **central** coupling strategy. Just like **XMR-TGK** we use Z -representations in the central layer, and the representations there have to fulfill

the same ncd-condition, but the conditions (2.81) and (2.82) are only checked for the outer representations. Eigenvalue refinements are done on the side that gives the better a priori bound for relative condition. To counter the fact that for the coupled approach we cannot prove that SHIFTRREL holds always, the Sturm counts to test the initial bounds (2.83) for the child eigenvalues for consistency are done for *both* outer representations. We also do similar “coupling checks” to verify the closeness of the local eigenvalues after they were classified and after they were refined on one side.

Just for the sake of completeness we mention how XMR-TGK and XMR-CPL perform on the test case that cause so much havoc in Experiments 3.4 and 3.6. The computed vectors differ, but interestingly the results are identical: both methods crack the problem with worst orthogonality levels of $1.15n\epsilon_\diamond$ and BSVD-residual norms $0.68\|B\|n\epsilon_\diamond$.

Tables 3.3 and 3.4 hold the orthogonality levels and residual norms of XMR-TGK and XMR-CPL on the testsets.

XMR-TGK works amazingly well. Indeed, comparing the results with Table 2.6 we see that the extracted vectors have better orthogonality than what DSTEMR provides for B^*B alone, and they are not that much worse than those delivered by XMR.

The coupled approach works also well on Pract, but has some undeniable problems with Synth. Indeed, not shown in the tables is that for 24 of the cases in Synth, XMR-CPL failed to produce up to 2.04% of the singular triplets. The reason is that for those cases there were clusters where none of the tried shift candidates satisfied the aforementioned consistency checks for the child eigenvalue bounds to replace the missing SHIFTRREL. Note that these failures are not errors, since the code did flag the triplets as not computed.

The accuracy results would mean that the coupled approach is clearly out-classed by using MR³ on the Golub-Kahan matrix in the fashion of Algorithm 3.2, if it were not for efficiency. Analogously to how it was done in §2.5 we have compiled profiling information in Table 3.5.

The *numbers* of steps of the three kinds are basically comparable between the methods, except for the additional coupling checks (bisection) and the fact that XMR-CPL always has to do at least one more RQI step to compute the second vector. But for XMR-CPL, each bisection and RQI step is done with matrices of dimension n , whereas everything XMR-TGK does involves a matrix of dimension $2n$. This fact is apparent in the estimated cost, which indicates that XMR-CPL could be expected to perform about 20 – 30% faster than XMR-TGK.

These results give in fact rise to a third method for BSVD, namely a combination of the first two: Use Algorithm MCR³, but after singleton eigenvalues have been refined to full accuracy for the outer layer, transform them via (3.30) and compute the final eigenvector with the central layer’s translate of $T_{\text{GK}}(B)$. Stated equivalently but the other way around, use MR³ on the Golub-Kahan matrix like

Pract 75 cases		ORTH	Synth 19240 cases	
XMR-TGK	XMR-CPL		XMR-TGK	XMR-CPL
5.35	10.71	AVG	5.34	6.33
2.71	2.44	MED	1.38	1.01
48.40	154	MAX	3095	27729
81.33 %	82.67 %	0 ... 10	92.59 %	91.04 %
18.67 %	14.67 %	10 ... 100	7.04 %	8.61 %
	2.67 %	100 ... 200	0.12 %	0.21 %
		200 ... 500	0.11 %	0.10 %
		500 ... 10 ³	0.07 %	0.02 %
		10 ³ ... 10 ⁶	0.06 %	0.03 %

Table 3.3: Orthogonality levels $\max\{|\mathbf{U}^*\mathbf{U} - \mathbf{I}|, |\mathbf{V}^*\mathbf{V} - \mathbf{I}|\}$ of XMR-TGK compared to XMR-CPL, measured as multiple of $n\epsilon_\diamond$.

Pract 75 cases		RESID	Synth 19240 cases	
XMR-TGK	XMR-CPL		XMR-TGK	XMR-CPL
0.35	15.78	AVG	0.45	3.14
0.07	1.37	MED	0.13	0.72
4.19	453	MAX	118	6873
92.00 %	34.67 %	0 ... 1	84.96 %	57.45 %
8.00 %	50.67 %	1 ... 10	15.03 %	35.50 %
	8.00 %	10 ... 100		7.00 %
	6.67 %	> 100	0.01 %	0.06 %

Table 3.4: Residual norms $\max_i \{ \|\mathbf{B}\bar{\mathbf{v}}_i - \bar{\mathbf{u}}_i\bar{\sigma}_i\|, \|\mathbf{B}^*\bar{\mathbf{u}}_i - \bar{\mathbf{v}}_i\bar{\sigma}_i\| \}$ of XMR-TGK compared to XMR-CPL, measured as multiples of $\|\mathbf{B}\|n\epsilon_\diamond$.

	Pract			Synth	
	XMR-TGK	XMR-CPL		XMR-TGK	XMR-CPL
<i># bisection steps</i>	8.10 <i>n</i>	10.55 <i>n</i>	AVG	14.74 <i>n</i>	17.93 <i>n</i>
	4.57 <i>n</i>	6.39 <i>n</i>	MED	17.44 <i>n</i>	20.46 <i>n</i>
	25.02 <i>n</i>	33.16 <i>n</i>	MAX	40.19 <i>n</i>	48.82 <i>n</i>
<i># RQI steps (no fallback bis.)</i>	2.01 <i>n</i>	3.24 <i>n</i>	AVG	2.12 <i>n</i>	3.40 <i>n</i>
	1.98 <i>n</i>	3.09 <i>n</i>	MED	2.14 <i>n</i>	3.39 <i>n</i>
	3.00 <i>n</i>	5.00 <i>n</i>	MAX	4.50 <i>n</i>	6.00 <i>n</i>
<i># tries for shifts</i>	0.62 <i>n</i>	0.56 <i>n</i>	AVG	0.92 <i>n</i>	0.79 <i>n</i>
	0.52 <i>n</i>	0.49 <i>n</i>	MED	1.00 <i>n</i>	0.75 <i>n</i>
	2.41 <i>n</i>	2.34 <i>n</i>	MAX	2.64 <i>n</i>	2.41 <i>n</i>
<i>est. cost (as plain Sturm counts)</i>	36.49 <i>n</i>	28.21 <i>n</i>	AVG	52.30 <i>n</i>	38.77 <i>n</i>
	34.45 <i>n</i>	27.47 <i>n</i>	MED	59.45 <i>n</i>	42.44 <i>n</i>
	93.69 <i>n</i>	71.68 <i>n</i>	MAX	113.93 <i>n</i>	86.29 <i>n</i>
<i>... bisection to classify & refine clusters</i>	33%	28%	AVG	43%	36%
	36%	28%	MED	57%	47%
	61%	53%	MAX	72%	62%
<i>... to refine singleton eigs & compute vectors</i>	48%	50%	AVG	42%	42%
	36%	36%	MED	28%	28%
	100%	100%	MAX	100%	100%
<i>... to determine en- velopes & find shifts</i>	19%	22%	AVG	15%	21%
	15%	22%	MED	17%	25%
	50%	48%	MAX	49%	55%

Table 3.5: Efficiency of XMR-TGK compared to XMR-CPL.

in Algorithm 3.2, but employ the coupling relations to outsource the expensive eigenvalue refinements to smaller matrices of half the size.

This approach would retain the increased accuracy of XMR-TGK at reduced cost, without the need for coupling checks. The catch is that we still need the central layer to be robust for XMR-TGK, but to do the eigenvalue computations with one outer layer the representation there has to be robust as well. A pity that we have not yet been able to repair the proof of Theorem 5.2 in [37], since then this additional condition would be implicit.

The described method could actually be regarded as new and sounds promising. In the upcoming Chapter 4 we will see that the further ingredient of using *block factorizations* for GK-translates lead again to more accuracy that make the combined method even more tasty.

Chapter 4

Block Factorizations

And he that breaks a thing to find out
what it is has left the path of wisdom.
— JOHN R.R. TOLKIEN, *The Lord of the Rings* (1954)

The sole possible cause of trouble with Algorithm MR³ is that relatively robust representations cannot always be guaranteed a priori in a practicable way. We have seen that robustness is intimately linked to element growth when forming the bidiagonal (or twisted) factorizations. The problem was magnified in the context of problem BSVD, since there local element growth in successive translates of a Golub-Kahan matrix can cause violation of the nearly constant diagonal and thus loss of the essential GK-structure of the local invariant subspaces.

This chapter is devoted to a technique that just might turn out to be a solution for all problems involving element growth. The approach is simply to allow 2×2 -pivots, or *blocks*, leading to a *block factorization* (BF).

For factorizing a symmetric tridiagonal matrix T given by its entries as $T = LDL^*$, block factorizations have been employed to great success [25, 41]. Higham shows in [42] how 2×2 -pivots can be used to solve a symmetric tridiagonal linear equation system in a normwise backward stable way.

The idea to employ block factorizations as representations within MR³ and its derivatives is not new. However, up to now it was not known how to do so. Recall that one essential requirement (SHIFTREL) is that representations can be shifted in a componentwise mixed relatively stable way, that is, we need to compute

$$L^+D^+(L^+)^* := LDL^* - \tau \tag{4.1}$$

such that small relative changes to the inputs (the data defining LDL^*) and outputs (the data defining $L^+D^+(L^+)^*$) give an exact relation. For diagonal matrices D and D^+ this was delivered by `dstqds`. But the situation becomes rather more

intricate if D and D^+ are block-diagonal of bandwidth one, maybe even with non-conforming structure.

We have devised a new algorithm to compute (4.1), with the feature to change the block-structure from D to D^+ on-the-fly. One could call it blocked `dstqds`. After studying some general structural properties of block factorizations with 2×2 -pivots in §4.1, we will present the algorithm in §4.2 and provide a complete mixed relative error analysis. Finally, §4.3 contains numerical experiments to show how the MR^3 -based methods for TSEP and BSVD from the previous chapters can benefit from using block factorizations.

We do also know how to do a progressive transformation (`dqds`) with blocks, but with the intention of keeping this thesis within (somewhat) reasonable bounds we decided to leave the full error analysis for that case in the hand-written stage it is right now and focus on presenting the stationary transformation.

To the best of our knowledge, the contents of this chapter are new. Many details have been inspired by private communications with Beresford Parlett. The only previous work pertaining to qd-like algorithms for block factorizations that we know of is unpublished work by Carla Ferreira and Lisa Miranian [31]. We took the motivation for the conditions when to choose a 2×2 -pivot (Block-Criterion I on page 186) from there, but except for that, the approach we take is different.

4.1 Basics

4.1.1 A note on 2-by-2 matrices

Let us compile the basic properties of a simple unreduced symmetric 2×2 matrix

$$D = \begin{bmatrix} d & e \\ e & c \end{bmatrix}, \quad e \neq 0.$$

The decision to name the diagonal entries d and c was deliberate and will become clear soon. Elementary properties of matrix norms reveal

$$\|D\|_1 = |e| + \max\{|d|, |c|\} = \|D\|_\infty, \quad \frac{\sqrt{2}}{2}\|D\|_1 \leq \|D\|_2 \leq \sqrt{2}\|D\|_1. \quad (4.2)$$

Provided that $\Delta := \det(D) = dc - e^2$ is nonzero the inverse is

$$D^{-1} = \frac{1}{\Delta} \begin{bmatrix} c & -e \\ -e & d \end{bmatrix}. \quad (4.3)$$

The eigendecomposition of D is $Dx_\pm = \lambda_\pm x_\pm$ where the eigenvalues λ_- and λ_+ as the (two, real) roots of $\det(D - \lambda)$ satisfy

$$2\lambda_\pm = (d + c) \pm \sqrt{(d + c)^2 - 4\Delta} = (d + c) \pm \sqrt{(d - c)^2 + 4e^2} \quad (4.4)$$

and

$$\lambda_- + \lambda_+ = d + c, \quad \lambda_- \lambda_+ = \Delta. \quad (4.5)$$

Remark 2.1 shows that the prerequisite $e \neq 0$ implies the Cauchy Interlace property (Thm. 1.23) holding strictly, i.e.,

$$\lambda_- < \min\{d, c\} \leq \max\{d, c\} < \lambda_+. \quad (4.6)$$

For the corresponding eigenvectors we obtain

$$\text{span}\{\mathbf{x}_\pm\} = \text{span}\left\{\begin{pmatrix} e \\ \lambda_\pm - d \end{pmatrix}\right\} = \text{span}\left\{\begin{pmatrix} \lambda_\pm - c \\ e \end{pmatrix}\right\}. \quad (4.7)$$

4.1.2 Structural implications of 2-by-2 pivots

For a given symmetric tridiagonal matrix \mathbf{T} , let us denote the diagonal and off-diagonal entries as c_i and e_i , respectively, that is,

$$\mathbf{T} = \begin{pmatrix} c_1 & e_1 & & & & \\ e_1 & c_2 & \ddots & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & c_{n-1} & e_{n-1} & \\ & & & e_{n-1} & c_n & \end{pmatrix} \in \mathbb{R}^{n \times n},$$

or shorter,

$$\mathbf{T} = \text{diag}(c_1, \dots, c_n) + \text{diag}_{\pm 1}(e_1, \dots, e_{n-1}).$$

We will only consider unreduced matrices, that is, $e_i \neq 0$ for $i = 1, \dots, n-1$.

Suppose we have a decomposition $\mathbf{T} = \mathbf{L}\mathbf{D}\mathbf{L}^*$ with unit lower *triangular* \mathbf{L} and *block*-diagonal \mathbf{D} with blocks of size one or two, that is,

$$\mathbf{D} = \text{diag}(\mathbf{D}_1, \dots, \mathbf{D}_N),$$

where

$$\mathbf{D}_j \in \mathbb{R}^{\text{size}(j) \times \text{size}(j)}, \quad \text{size}(j) \in \{1, 2\}, \quad j = 1, \dots, N.$$

The blocks in \mathbf{D} induce a partition of $\{1, \dots, n\}$. Let us structure the matrices \mathbf{L} and \mathbf{T} conformably and write $\mathbf{T}_{k,j}$ and $\mathbf{L}_{k,j}$ to refer to individual blocks. We will emphasize the lower triangles ($k \geq j$) and use $\mathbf{L}_{k,j}^* := (\mathbf{L}_{k,j})^* = (\mathbf{L}^*)_{j,k}$, which is notably different from $(\mathbf{L}^*)_{k,j}$.

As \mathbf{L} is unit triangular, the blocks $\mathbf{L}_{j,j}$ must be nonsingular. Multiply the j th column of \mathbf{L} by $\mathbf{L}_{j,j}^{-1}$ and take $\mathbf{L}_{j,j}\mathbf{D}_j\mathbf{L}_{j,j}^*$ instead of \mathbf{D}_j to see that, without loss of generality, we can assume them to be just copies of the identity, i.e.,

$$\mathbf{L}_{j,j} = \mathbf{I}_{\text{size}(j)}, \quad j = 1, \dots, N.$$

Using the triangular structure of L , the relation $T = LDL^*$ read in block notation becomes

$$T_{k,j} = \sum_{i=1}^{\min\{j,k\}} L_{k,i} D_i L_{j,i}^*, \quad 1 \leq j, k \leq N. \quad (4.8)$$

The following lemma summarizes some essential properties of block factorizations, in particular that they can only exist if the diagonal blocks D_j are nonsingular, except possibly for D_N . In this regard it generalizes Lemma 2.15.

Lemma 4.1. *Let unreduced tridiagonal T have a block factorization $T = LDL^*$ as outlined above. Then $T_{1,1} = D_1$ and for $j = 1, \dots, N-1$ the following holds:*

- (i) $T_{j+1,j} = L_{j+1,j} D_j$.
- (ii) D_j is nonsingular.
- (iii) $L_{k,j} = 0$ for $k > j+1$.
- (iv) $T_{j+1,j+1} = D_{j+1} + L_{j+1,j} D_j L_{j+1,j}^*$.
- (v) $T_{j+1,j+1} = D_{j+1} + s e_1 e_1^*$, with $s \in \mathbb{R}$, $e_1 = I_{\text{size}(j+1)}(:, 1)$.

Proof. We will proceed by induction on j . Assume the claims hold for all $j' < j$; this includes $j = 1$.

Invoke equation (4.8). For $j = 1$, it is just $T_{1,1} = D_1$; for $j > 1$ and (iii) holding for all $j' < j$, it folds down to give us (i), (iv) and $T_{k,j} = L_{k,j} D_j = 0$ for $k > j+1$, as T is tridiagonal. Clearly then, (iii) will fall into place as soon as we establish nonsingularity of D_j .

Using the induction hypothesis on (v) for $j-1$ yields

$$T_{j,j} = D_j + s e_1 e_1^*. \quad (\star)$$

This will allow to prove nonsingularity of D_j . From (i) and T being unreduced, we have

$$T_{j+1,j} = L_{j+1,j} D_j \neq 0, \quad (\star\star)$$

and therefore $D_j \neq 0$. Hence, $\det(D_j) = 0$ is only possible if $\text{size}(j) = 2$ and D_j has rank one. Use (\star) to see that D_j has nonzero offdiagonal entries, so with $\det(D_j) = 0$ we get that D_j cannot have zero entries at all. In particular, there is no vector of the form $(0, \alpha)$, $\alpha \neq 0$ in the row space of D_j . On the other hand, T is tridiagonal, so if $\text{size}(j) = 2$ then the block $T_{j+1,j}$ has only zeroes in its first (of two) columns. But this contradicts $(\star\star)$. So D_j must be nonsingular after all.

We have already established (iv). Thus, for (v) there is only something to prove if $\text{size}(j) = 2$. Denote the one nonzero entry in $T_{j+1,j}$ by e_i . Then

$$\begin{aligned} L_{j+1,j} D_j L_{j+1,j}^* &= L_{j+1,j} D_j D_j^{-1} D_j L_{j+1,j}^*, && \text{as } D_j \text{ is nonsingular,} \\ &= T_{j+1,j} D_j^{-1} T_{j+1,j}^*, && \text{by (i),} \\ &= \underbrace{e_i^2 D_j^{-1}(2, 2)}_{:=s} e_1 e_1^* \end{aligned}$$

gives us (v). □

Thus, \mathbf{L} has in fact only $N - 1$ nontrivial blocks, making double indices superfluous. With

$$\mathbf{L}_j := \mathbf{L}_{j+1,j} \in \mathbb{R}^{\text{size}(j+1) \times \text{size}(j)}, \quad j = 1, \dots, N - 1$$

and $\mathbf{l}_j := \mathbf{l}_{\text{size}(j)}$ we can summarize the situation as

$$\mathbf{T} = \begin{bmatrix} \mathbf{l}_1 & & & & & \\ \mathbf{L}_1 & \mathbf{l}_2 & & & & \\ & \ddots & \ddots & & & \\ & & & \mathbf{L}_{N-1} & \mathbf{l}_N & \\ & & & & & \end{bmatrix} \begin{bmatrix} \mathbf{D}_1 & & & & & \\ & \mathbf{D}_2 & & & & \\ & & \ddots & & & \\ & & & \mathbf{D}_N & & \end{bmatrix} \begin{bmatrix} \mathbf{l}_1 & \mathbf{L}_1 & & & & \\ & \mathbf{l}_2 & \ddots & & & \\ & & \ddots & \mathbf{L}_{N-1} & & \\ & & & & \mathbf{l}_N & \end{bmatrix}.$$

Glancing at this formula one could think that \mathbf{L} has bandwidth three. That would be true, but rather imprecise. Combine (i) and (ii) from Lemma 4.1 to see that, independent of $\text{size}(j)$ and $\text{size}(j + 1)$, \mathbf{L}_j must have rank one; in fact, for $\text{size}(j + 1) = 2$ we get

$$\mathbf{e}_2^* \mathbf{L}_j = \mathbf{o} \in \mathbb{R}^{1 \times \text{size}(j)}.$$

This reveals the rather special structure of \mathbf{L} : a bandwidth bounded by two but nevertheless only $n - \text{size}(N) \leq n - 1$ nontrivial (meaning nonzero and offdiagonal) entries, at most two in each block.

Now we will look more closely at the block entries and how they relate to the entries of \mathbf{T} . Independent of $\text{size}(j)$ we can say that \mathbf{D}_j covers the entries (i, i) , (i, i') , (i', i) , and (i', i') of \mathbf{D} , where

$$i = i(j) = 1 + \sum_{k=1}^{j-1} \text{size}(k) \quad \text{and} \quad i' = i + \text{size}(j) - 1.$$

Fix this relation for concrete j and i for a moment. Recall that property (v) of Lemma 4.1 revealed that \mathbf{D}_j has at most the top left entry not being also an entry of \mathbf{T} . We will denote this qualifying feature of \mathbf{D}_j as d_i . Then we have

$$\mathbf{D}_j = \begin{cases} d_i, & \text{size}(j) = 1, \\ \begin{bmatrix} d_i & e_i \\ e_i & c_{i+1} \end{bmatrix}, & \text{size}(j) = 2. \end{cases} \quad (4.9)$$

It was already stated that L has at most $n - 1$ nontrivial entries. Depending on the structure, we will use letters k and ℓ to refer to them, in the sense

$$L_j = \begin{cases} \ell_i, & \text{size}(j) = \text{size}(j + 1) = 1, \\ [\ell_i \ 0]^*, & \text{size}(j) = 1, \text{size}(j + 1) = 2, \\ [k_i \ \ell_{i+1}], & \text{size}(j) = 2, \text{size}(j + 1) = 1, \\ \begin{bmatrix} k_i & \ell_{i+1} \\ 0 & 0 \end{bmatrix}, & \text{size}(j) = \text{size}(j + 1) = 2. \end{cases} \quad (4.10)$$

Note that with our definition there exists for each index i either a d_i or a c_i , and either a k_i or an ℓ_i , but never both. The natural question arises why we differentiate at all between d 's and c 's for D , k 's and ℓ 's for L , that is, why not use simply $d_i = D(i, i)$ for the diagonal of D and $\ell_1, \dots, \ell_{n-\text{size}(N)}$ for the nontrivial entries in L (like in [31])? We have two reasons to refrain from doing so. The first is that the respective two quantities have very differing semantics, e.g., a c_i is also a diagonal entry of T , but a d_i is not. This distinction will become only more pronounced as we go on. The second reason is clarity of presentation. Employing block factorizations inevitably involves case distinctions to deal with the block structure, which can become confusing at points. Separating $\text{diag}[D]$ into d_i 's and c_i 's, and L into k_i 's and ℓ_i 's lets formulae carry the block structure implicitly, whereas using just d_i and ℓ_i does not.

Henceforward our treatment is based on entries (indices i) instead of whole blocks D_j ; in fact we can all but forget about the block indices j . We will make a slight but crucial adjustment in terminology in denoting the D_j 's as *pivots* from now on and use *block* synonymously to 2×2 -pivot or 2×2 -block; a D_j with $\text{size}(j) = 1$ is just a 1×1 - or *single* pivot (but *not* a block anymore). Thus we can categorize the role of an index $i \in \{1, \dots, n\}$ into exactly one of either

- being *single*, with data d_i and ℓ_i , if $i < n$;
- *starting a block*, with data d_i and k_i , if $i < n - 1$;
- *ending a block*, with data c_i and ℓ_i , if $i < n$.

The determinants of 2×2 -pivots will play an important role, so let us ease working with them by introducing

$$\boxed{\Delta_i := d_i c_{i+1} - e_i^2} \quad (4.11)$$

for each i that starts a block. Based on these we define for $i = 1, \dots, n - 1$ the revealing quantity

$$\text{inv}_D(i) := \begin{cases} 1/d_i, & \text{if } i \text{ is single,} \\ 0, & \text{if } i \text{ starts a block,} \\ d_{i-1}/\Delta_{i-1}, & \text{if } i \text{ ends a block.} \end{cases} \quad (4.12)$$

Note that this definition is always proper if the factorization exists. Based on our implicit-zero convention and using Lemma 4.1, the relation between the diagonals of \mathbf{D} and \mathbf{T} can now be stated compactly as

$$\mathbf{T}(i, i) \equiv \mathbf{D}(i, i) + e_{i-1}^2 \text{inv}_{\mathbf{D}}(i-1), \quad (4.13)$$

which encompasses the special case $\mathbf{T}(1, 1) = \mathbf{D}(1, 1) = d_1$. Concerning \mathbf{L} , point (i) in Lemma 4.1 gives the characterization

$$\begin{cases} k_i = -e_{i+1}e_i/\Delta_i, & \text{if } i \text{ starts a block and } i < n-1, \\ \ell_i = e_i \text{inv}_{\mathbf{D}}(i), & \text{if } i \text{ does not start a block.} \end{cases} \quad (4.14)$$

We close our introduction to 2×2 -pivots with some examples.

Example 4.2. A Golub-Kahan matrix does not admit a bidiagonal (or twisted) factorization, because the first pivot is zero regardless of where you start. The natural way to factorize a Golub-Kahan matrix is using 2×2 -pivots all the way:

$$\begin{pmatrix} 0 & a_1 & & \\ a_1 & 0 & b_1 & \\ & b_1 & 0 & a_2 \\ & & a_2 & 0 \end{pmatrix} = \mathbf{LDL}^*,$$

with $\mathbf{L} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ b_1/a_1 & & 1 & \\ & & & 1 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} 0 & a_1 & & \\ a_1 & 0 & & \\ & & 0 & a_2 \\ & & a_2 & 0 \end{pmatrix}.$

Note that there was at no time a choice to take a single pivot; the block structure is indeed unique. \diamond

Example 4.3. Consider Example 3.13 again, which exhibited that applying a tiny shift to a Golub-Kahan matrix will almost inevitably incur huge element growth. Using 2×2 -pivots, the same matrix can be factorized as follows:

$$\begin{pmatrix} -\alpha & 1 & & \\ 1 & -\alpha & 1 & \\ & 1 & -\alpha & \alpha \\ & & \alpha & -\alpha \end{pmatrix} = \mathbf{LDL}^*$$

where

$$\mathbf{L} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ k_1 & \ell_2 & 1 & \\ & & \ell_3 & 1 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} d_1 & 1 & & \\ 1 & c_2 & & \\ & & d_3 & \\ & & & d_4 \end{pmatrix},$$

$$\begin{cases} k_1 = \frac{1}{1-\alpha^2}, \\ \ell_2 = \frac{\alpha}{1-\alpha^2}, \\ \ell_3 = -\frac{2-\alpha^2}{1-\alpha^2}, \end{cases} \quad \begin{cases} d_1 = -\alpha, \\ c_2 = -\alpha, \\ d_3 = -\alpha \frac{2-\alpha^2}{1-\alpha^2}, \\ d_4 = \alpha \frac{1+2\alpha^2}{2+\alpha^2}. \end{cases}$$

Note that for this case the block structure is not unique, but the chosen one makes the most sense, showing no discernible element growth at all. \diamond

Example 4.4. A block factorization \mathbf{LDL}^* with factors \mathbf{L} and \mathbf{D} that have the structural properties we saw in this section is not necessarily tridiagonal. Consider the factors from the previous example and change \mathbf{L} to $\tilde{\mathbf{L}}$ just by replacing ℓ_2 with $\ell_2(1 + \epsilon_\diamond)$. This results in

$$(\tilde{\mathbf{L}}\tilde{\mathbf{D}}\tilde{\mathbf{L}}^*)(3, 1) = \ell_2\epsilon_\diamond \neq 0.$$

Hence, relative perturbations to the nontrivial entries of \mathbf{L} can destroy tridiagonal structure. This happens only if the perturbations are uncorrelated. For the case above, we should have perturbed k_1 to $k_1(1 + \epsilon_\diamond/d_1)$ at the same time to retain the structure. \diamond

Example 4.5. We mentioned that the factor \mathbf{L} can have less than $n - 1$ nontrivial entries, if (and only if) a block ends at n . Consider the following 3×3 toy-problem to see this behavior in action:

$$\begin{pmatrix} 1 & 2 & \\ 2 & 3 & 5 \\ & 5 & 4 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & & \\ & -1 & 5 \\ & 5 & 4 \end{pmatrix} \begin{pmatrix} 1 & 2 & \\ & 1 & \\ & & 1 \end{pmatrix}$$

$\mathbf{L} \qquad \mathbf{D} \qquad \mathbf{L}^*$

\diamond

A representation for block factorizations

To define a specific block factorization, we need first of all a means to capture the block structure, i.e., the “kind” of each index. Minimality of information will prove to be a boon here. Thus we just keep track of the indices where a 2×2 -pivot ends, in a set

$$\Omega \subseteq \{2, \dots, n\}.$$

Due to its nature, this set will always satisfy $i \in \Omega \Rightarrow \{i-1, i+1\} \cap \Omega = \emptyset$ and can therefore never contain more than $\lfloor n/2 \rfloor$ elements. No other information is required to capture the block structure.

A block factorization of a symmetric tridiagonal matrix is then completely described by the following data:

STANDARD REPRESENTATION FOR A TOP-DOWN BF

Ω	Block structure: Indices where a 2×2 -pivot ends.
e_1, \dots, e_{n-1}	The offdiagonal elements.
$D(1, 1), \dots, D(n, n)$	Diagonal entries of D.

Note that this is in fact a generalization of the e -representation for non-blocked decompositions, as for the case that only single pivots are employed ($\Omega = \emptyset$), the same data items are kept. It has the same benefits as well, as the offdiagonal elements e_i can be reused, which is good for efficiency and makes the mixed relative error analysis for shifting easier.

Using the set Ω one can tell if a $D(i, i)$ is actually a d_i or a c_i , through

$$D(i, i) = \begin{cases} d_i, & \text{if } i \notin \Omega, \\ c_i, & \text{if } i \in \Omega. \end{cases}$$

Recall from § 2.4.1 our distinction between *primary* and *secondary* data items. With respect to the standard representation for block factorizations above, the quantities Δ_i , $\text{inv}_D(i)$, as well as the entries k_i and ℓ_i of L , are secondary data and can be derived from (4.11), (4.12) and (4.14), respectively.

The reader might now ask why we do not employ L in the representation at all. Indeed, why not represent a block factorization using the nontrivial entries of D and L , similar to the N -representation we used for standard bidiagonal decompositions? The answer is quite subtle. Such a representation would effectively contain *five* numbers for each 2×2 -pivot (except the N 'th), namely d_i, c_{i+1}, e_i, k_i and ℓ_{i+1} . But these quantities are not independent, since they have to obey (4.14). Thus, basing a componentwise error analysis on such a representation would be fatal, as uncorrelated perturbations to all five data items at once will in general cause (4.14) not to be satisfied any more. The effect was already exhibited in Example 4.4: loss of tridiagonal structure, due to fill-in in $L_j D_j$.

One sensible alternative representation to our standard one would be to use all entries from D but only the ℓ_i 's of L , i.e., the data

$$\{d_i, \ell_{i-1} \mid i \notin \Omega\} \quad \text{and} \quad \{e_{i-1}, c_i \mid i \in \Omega\}.$$

The main reason why we stick to our standard representation is laziness. Maximal use of the offdiagonal elements inherits from an e -representation the advantage that, for a mixed relative error analysis of a shifting algorithm, we have $n-1$ less perturbations to specify, since the offdiagonal entries can be reused. This will save us a lot of work on the following pages.

The connection between blocked and unblocked

Allowing 2×2 -pivots in D forfeits uniqueness, in the sense that multiple block structures may be possible for the same symmetric tridiagonal matrix, including for example using no blocks at all. Let T have a block factorization $T = LDL^*$ as before but also a “normal” lower bidiagonal factorization

$$T = \hat{L}\hat{D}\hat{L}^*$$

with diagonal \hat{D} . Using our results so far, in particular (4.14), the following relations between the elements are easily derived:

$$\hat{d}_i \equiv \begin{cases} \Delta_{i-1}/d_{i-1}, & \text{if } i \text{ ends a block in } D, \\ d_i, & \text{otherwise,} \end{cases}$$

$$\hat{\ell}_i \equiv \begin{cases} -k_i/\ell_{i+1}, & \text{if } i \neq n-1 \text{ starts a block in } D, \\ e_{n-1}/d_{n-1}, & \text{if } i = n-1 \text{ starts a block in } D, \\ \ell_i, & \text{otherwise.} \end{cases} \quad (4.15)$$

Clearly, the only case where a blocked $T = LDL^*$ exists but non-blocked $T = \hat{L}\hat{D}\hat{L}^*$ does not occur when a block-starting d_i from D is zero.

As a remark, these relations were one more reason for us to keep c_i and k_i separate from d_i and ℓ_i , as only the latter are identical to the corresponding data in a non-blocked factorization of the same matrix.

4.1.3 When not to choose a 2-by-2 pivot

For factorizing $T = LDL^*$ —with or without shift and not regarding how T is represented—the basic motivation for allowing a 2×2 -pivot in D covering indices i and $i + 1$ is to avoid what would otherwise become a very “large” single pivot d_{i+1} . How to gauge what is “large” depends on the application. In the past, a variety of schemes have been devised to evaluate when selecting a 2×2 -pivot, see for example [25, 42]. All of those strategies essentially try to avoid global element growth, that is, they would compare d_{i+1} to $\|T\|$, $\text{spdiam}[T]$ or some variant thereof.

Alternatively, one can evaluate the *local* element growth caused by the potential single pivot d_{i+1} , by comparing it directly to the concerned diagonal entry $T(i + 1, i + 1)$. This would become the lower right entry c_{i+1} of a block, should one be chosen. If not, the single pivot d_{i+1} is given by

$$T(i + 1, i + 1) = d_{i+1} + e_i^2/d_i.$$

Hence, if d_{i+1} exceeds $T(i + 1, i + 1)$ in magnitude by far, there has to be cancellation between d_{i+1} and e_i^2/d_i , so e_i^2/d_i must also exceed $T(i + 1, i + 1)$ in magnitude. The latter is preferable for a test, since it does not require to compute d_{i+1} .

All in all, this motivates that each 2×2 -pivot $\begin{bmatrix} d_i & e_i \\ e_i & c_{i+1} \end{bmatrix}$ in D should fulfill

$$\boxed{|d_i c_{i+1}| < K_{\square} e_i^2} \quad (4.16)$$

for a constant $K_{\square} \in (0, 1)$.

None of the pivoting strategies aiming at global element growth that were mentioned above would ever choose a 2×2 -pivot violating the condition (4.16). The reason is, basically, that avoiding local element growth requires the most 2×2 -pivots, and any more make really no sense in whatever application (that we know of). Since the error analysis in this chapter does only require that each selected 2×2 -pivot obeys (4.16), it remains fully applicable if other (more lax) pivoting schemes are employed.

This condition is linked closely to the block determinants. Obviously, fulfillment of (4.16) implies $\Delta_i < 0$, which is in fact a consequence of the stronger property

$$-(1 + K_{\square})e_i^2 < \Delta_i < (K_{\square} - 1)e_i^2 < 0.$$

However, a far more important purpose of requiring (4.16) to be fulfilled is to ensure that computing the block determinants from given representation data is well-conditioned. Recall (1.11) to obtain

$$\boxed{\kappa_{-(d_i c_{i+1}, e_i^2)} < \frac{1 + K_{\square}}{1 - K_{\square}} =: \kappa_{\Delta}} \quad (4.17)$$

as worst-case bound for the condition number of a block determinant. Even the lax choice $K_{\square} = 1/4$ results in a benign bound of $\kappa_{\Delta} = 5/3$. As about every use of a block factorization will have to refer to the block determinants at some point, being able to compute them stably is crucial.

The meanings of K_{\square} and κ_{Δ} will remain in effect throughout this chapter.

A note on parameter-based tests and rounding errors. Usually we design tests to be executed in a floating-point context, where a successful outcome will merely indicate that the condition is fulfilled for slightly perturbed data. For example, a positive evaluation of (4.16) would mean that the data on the machine obeys

$$|d_i c_{i+1}|(1 + \alpha_*) < K_{\square} e_i^2 (1 + \beta_*)(1 + \alpha_{\text{sq}}), \quad |\alpha_*|, |\beta_*|, |\alpha_{\text{sq}}| \leq \epsilon_{\diamond}.$$

So, strictly speaking, (4.16) would not be satisfied as such, but only for a modified parameter K_{\square} .

It will often be necessary to rely on (4.16) or (4.17) for proving a result. Then these inaccuracies—although of no real practical importance whatsoever—would cause loss of mathematical correctness in the stated results if ignored. There is

a simple remedy: execute the test with a parameter that has been modified to anticipate any effects due to floating-point operations (or perturbations of the data).

Obviously, a relative modification of the parameter is sufficient, and there is really no sense to limit this modification to just a couple of ulps. If all tests were executed with, say,

$$\boxed{K_{\square}^* := K_{\square} - \max\{n^2, \epsilon_{\diamond}^{-1/2}\} \text{ulp}(K_{\square})} \quad (4.18)$$

instead of K_{\square} , we would be on the safe side and need never again worry about if the condition will hold for perturbed data as well. This definition is just a suggestion, in fact, for our applications $K_{\square}^* \lesssim K_{\square}(1 - \epsilon_{\diamond})^3$ would already suffice.

This approach makes K_{\square} our pristine tool for theoretical arguments. As an application, (4.16) executed with K_{\square}^* gives us $|\text{fl}(d_i c_{i+1})| < \text{fl}(K_{\square}^* e_i^2)$. Because of the safeguard function of K_{\square}^* , we could then without hesitation conclude

$$|\text{fl}(d_i c_{i+1})| < K_{\square} \text{fl}(e_i^2) \quad (4.19)$$

and also strengthen (4.17) to

$$\kappa_{-}(\text{fl}(d_i c_{i+1}), \text{fl}(e_i^2)) < \kappa_{\Delta}. \quad (4.20)$$

The latter in particular will prove to be very handy for relating computed block determinants to their perturbed counterparts.

4.2 Stationary Block Factorization

Purpose of this section is to develop an analogon to `dstqds` for block factorizations, to compute

$$\mathbb{T} - \tau = \mathbb{T}^+, \quad \text{where } \mathbb{T} = \text{LDL}^*, \mathbb{T}^+ = \text{L}^+ \text{D}^+ (\text{L}^+)^* \in \mathbb{R}^{n \times n} \quad (4.21)$$

and D , D^+ are block-diagonal (with bandwidth one) each. We call LDL^* the *source* and $\text{L}^+ \text{D}^+ (\text{L}^+)^*$ the *target* of the process. Our treatment is based on the standard representation, that is, we assume to get the data $(\Omega, \{d_i\}, \{c_j\}, \{e_k\})$ as input for LDL^* and the outputs $(\Omega^+, \{d_i^+\}, \{c_j^+\}, \{e_k\})$ will define $\text{L}^+ \text{D}^+ (\text{L}^+)^*$.

The ultimate goal is an algorithm that allows a componentwise mixed relative error analysis. Hence we need to find suitable perturbations of the input and output data items, of the form

$$\begin{cases} d_i \rightsquigarrow \tilde{d}_i, & i \notin \Omega, & d_i^+ \rightsquigarrow \tilde{d}_i^+, & i \notin \Omega^+, \\ c_i \rightsquigarrow \tilde{c}_i, & i \in \Omega, & c_i^+ \rightsquigarrow \tilde{c}_i^+, & i \in \Omega^+, \\ e_i \rightsquigarrow \tilde{e}_i, & i = 1, \dots, n-1, & & \end{cases}$$

such that the thusly perturbed matrices satisfy $\widetilde{\mathbf{L}}\widetilde{\mathbf{D}}\widetilde{\mathbf{L}}^* - \tau = \widetilde{\mathbf{L}}^+\widetilde{\mathbf{D}}^+(\widetilde{\mathbf{L}}^+)^*$. Note that the perturbations to d_i, c_i, d_i^+, c_i^+ can be compactly stated as

$$\mathbf{D}(i, i) \rightsquigarrow \widetilde{\mathbf{D}}(i, i), \quad \mathbf{D}^+(i, i) \rightsquigarrow \widetilde{\mathbf{D}}^+(i, i), \quad i = 1, \dots, n.$$

Just as in standard `dstqds` we introduce auxiliary *adjustment* quantities

$$s_i := \mathbf{D}_+(i, i) - \mathbf{D}(i, i) + \tau, \quad i = 1, \dots, n. \tag{4.22}$$

However, for block factorizations these do not allow for a recursive formulation of the factorization process like in Remark 2.22, except if the block structures Ω and Ω^+ are identical. Furthermore, the way to compute s_{i+1} is not unique anymore, but depends on the local structure at hand, meaning the four truth values of $i \in \Omega, i \in \Omega^+, i + 1 \in \Omega, i + 1 \in \Omega^+$. With (4.13) we have

$$s_{i+1} = e_i^2(\text{inv}_{\mathbf{D}}(i) - \text{inv}_{\mathbf{D}^+}(i)), \quad i = 1, \dots, n - 1.$$

Just spelling out the definition (4.12) of $\text{inv}_{\mathbf{D}}(i)$ and $\text{inv}_{\mathbf{D}^+}(i)$ yields nine possible cases that are to be considered¹; they are compiled in Table 4.1.

The cases S1, S3, S6 and S7 are special because they do not allow to compute s_{i+1} using just multiplications and divisions. It is possible to rewrite their definitions such that previously computed auxiliaries can be utilized for computing s_{i+1} ; the resulting alternative definitions are collected in Table 4.2. To see how these are derived, let us exhibit case S3:

$$\begin{aligned} s_{i+1} &= e_i^2 \left(\frac{d_{i-1}}{\Delta_{i-1}} - \frac{d_{i-1}^+}{\Delta_{i-1}^+} \right) && \text{from Table 4.1,} \\ &= e_i^2 \frac{d_{i-1}(d_{i-1}^+c_i^+ - e_{i-1}^2) - d_{i-1}^+(d_{i-1}c_i - e_{i-1}^2)}{\Delta_{i-1}\Delta_{i-1}^+} && \text{by (4.11),} \\ &= e_i^2 \frac{e_{i-1}^2(d_{i-1}^+ - d_{i-1}) - d_{i-1}d_{i-1}^+\tau}{\Delta_{i-1}\Delta_{i-1}^+} && \text{as } c_i^+ = c_i - \tau, \\ &= e_i^2 \frac{e_{i-1}^2(s_{i-1} - \tau) - d_{i-1}d_{i-1}^+\tau}{\Delta_{i-1}\Delta_{i-1}^+} && \text{by (4.22).} \end{aligned}$$

Note that standard `dstqds` is completely subsumed in case S1 alone and the respective (standard and alternative) formulae are identical.

The error analysis to come is more or less based on considering these cases separately but we will have to jump between them occasionally. With the intent of making the presentation more accessible we will continue to use the pictograms shown in the tables. To explain how they are intended to work, consider case S6 as example. Formally, this case is characterized by

$$i \in \Omega, \quad i \notin \Omega^+, \quad i + 1 \notin \Omega^+.$$

¹Not sixteen, because neither of Ω and Ω^+ does contain two consecutive indices.

Case	Description	i ↓	s_{i+1}
S1	$i, i+1 \notin \Omega$ $i, i+1 \notin \Omega^+$	$\bullet \bullet$ $\bullet \bullet$	$e_i^2/d_i - e_i^2/d_i^+$
S2	$i+1 \in \Omega$ $i+1 \in \Omega^+$	$\bullet \text{---} \circ$ $\bullet \text{---} \circ$	0
S3	$i \in \Omega$ $i \in \Omega^+$	$\bullet \text{---} \circ \bullet$ $\bullet \text{---} \circ \bullet$	$e_i^2 d_{i-1} / \Delta_{i-1} - e_i^2 d_{i-1}^+ / \Delta_{i-1}^+$
S4	$i+1 \in \Omega$ $i, i+1 \notin \Omega^+$	$\bullet \text{---} \circ$ $\bullet \bullet$	$-e_i^2 / d_i^+$
S5	$i, i+1 \notin \Omega$ $i+1 \in \Omega^+$	$\bullet \bullet$ $\bullet \text{---} \circ$	e_i^2 / d_i
S6	$i \in \Omega$ $i, i+1 \notin \Omega^+$	$\bullet \text{---} \circ \bullet$ $\bullet \bullet$	$e_i^2 d_{i-1} / \Delta_{i-1} - e_i^2 / d_i^+$
S7	$i, i+1 \notin \Omega$ $i \in \Omega^+$	$\bullet \bullet$ $\bullet \text{---} \circ \bullet$	$e_i^2 / d_i - e_i^2 d_{i-1}^+ / \Delta_{i-1}^+$
S8	$i \in \Omega$ $i+1 \in \Omega^+$	$\bullet \text{---} \circ \bullet$ $\bullet \text{---} \circ$	$e_i^2 d_{i-1} / \Delta_{i-1}$
S9	$i+1 \in \Omega$ $i \in \Omega^+$	$\bullet \text{---} \circ$ $\bullet \text{---} \circ \bullet$	$-e_i^2 d_{i-1}^+ / \Delta_{i-1}^+$

Table 4.1: Standard formulae for the next adjustment s_{i+1} .

Case	i ↓	$s_{i+1} =$
S1	$\begin{matrix} \bullet & \bullet \\ \bullet & \bullet \end{matrix}$	$e_i^2(s_i - \tau) / (d_i d_i^+)$
S3	$\begin{matrix} \bullet \text{---} \circ & \bullet \\ \bullet \text{---} \circ & \bullet \end{matrix}$	$e_i^2 [e_{i-1}^2(s_{i-1} - \tau) - d_{i-1} d_{i-1}^+ \tau] / (\Delta_{i-1} \Delta_{i-1}^+)$
S6	$\begin{matrix} \bullet \text{---} \circ & \bullet \\ \bullet & \bullet \end{matrix}$	$e_i^2 [d_{i-1}(s_i - \tau) + e_{i-1}^2] / (\Delta_{i-1} d_i^+)$
S7	$\begin{matrix} \bullet & \bullet \\ \bullet \text{---} \circ & \bullet \end{matrix}$	$e_i^2 [d_{i-1}^+(s_i - \tau) - e_{i-1}^2] / (d_i \Delta_{i-1}^+)$

Table 4.2: Alternative formulae for s_{i+1} , utilizing previous auxiliaries in the form of already computed quantities $s_j - \tau, j \leq i$.

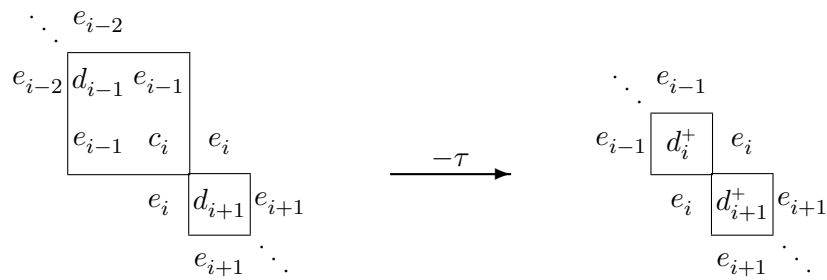


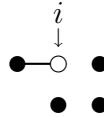
Figure 4.1: Detailed situation for case S6

Although being short, it quickly becomes cumbersome and error-prone to differentiate the cases based solely on this purely mathematical definition. A more graphical representation for the situation results in something like Figure 4.1.

Now that is too much information. Essential is just the structure of Ω and Ω^+ and the pattern they induce for the distribution of the diagonal entries of D and D^+ into d_i 's and c_j 's; for case S6 this is

$$\begin{array}{cccc} \cdots & \boxed{d_{i-1} & c_i} & d_{i+1} & \cdots & \text{as relevant structure in } D, \text{ and} \\ & \cdots & d_i^+ & d_{i+1}^+ & \cdots & \text{as relevant structure in } D^+. \end{array}$$

The pictograms we use capture just that, using \bullet to represent a d_i and \circ for a c_j , with a connecting line to further pronounce a block:



The top line represents the structure in D and the bottom line the structure in D^+ . We find these little pictures to be of great help in keeping track of the situation at hand. Two things should be pointed out. Firstly, the pictogram for case S6 has nothing in its lower left corner. The reason is that this case does not specify if $i - 1$ belongs to Ω^+ or not, because it has no impact on the definition of s_{i+1} . And secondly, keep in mind that a \bullet at the right end may well stand for the start of a block, but this, too, has no effect on how s_{i+1} is defined.

The task to compute a block factorization will prove to be much harder than standard `dstqds`. To make the problem manageable, we allow for two simplifications:

- The computed auxiliaries s_i need not fulfill the respective relation from Table 4.1 exactly for the perturbed data. Instead, we will be content if a small relative perturbation of s_i has this property.

This relaxation makes it meaningful to see the auxiliaries as secondary data and denote by \tilde{s}_i the correct value for the perturbed data; you get this by replacing everything in any definition by its perturbed counterpart, e.g.,

$$\tilde{s}_{i+1} = \tilde{e}_i^2 (\tilde{d}_{i-1} / \tilde{\Delta}_{i-1} - 1 / \tilde{d}_i^+) = \tilde{e}_i^2 (\tilde{d}_{i-1} / (\tilde{d}_{i-1} \tilde{c}_i - \tilde{e}_{i-1}^2) - 1 / \tilde{d}_i^+)$$

for case S6.

- The effective shift for any index i may be perturbed. What we mean by this is that instead for (4.21) above we actually strive for

$$\tilde{L} \tilde{D} \tilde{L}^* - \text{diag}(\tilde{\tau}_i) = \tilde{L}^+ \tilde{D}^+ (\tilde{L}^+)^*.$$

Recall from (2.58) that this is harmless as it just states that an additional outer perturbation is required to obtain (4.21).

With those in place, the relations to be fulfilled by the perturbation are just

$$\tilde{D}^+(i, i) \stackrel{!}{=} \tilde{D}(i, i) + \tilde{s}_i - \tilde{\tau}_i, \quad i = 1, \dots, n, \quad (4.23)$$

as our standard representation allows to reuse the offdiagonals e_i for the target.

Based on the auxiliaries s_i and assuming the block structure Ω^+ is known, the computation of $L^+D^+(L^+)^*$ can proceed in a manner similar to standard `dstqds`, using the following computational sequence as algorithmic template:

CS 4.1: *Template for the stationary block factorization*

```

1:    $s_1 := 0$ 
2:   for  $i = 1$  to  $n - 1$  do
3:      $D^+(i, i) := D(i, i) + (s_i - \tau)$ 
4:     // Compute block determinants
5:     if  $i \in \Omega$  then
6:        $\Delta_{i-1} := d_{i-1}c_i - e_{i-1}^2$ 
7:     endif
8:     if  $i \in \Omega^+$  then
9:        $\Delta_{i-1}^+ := d_{i-1}^+c_i^+ - e_{i-1}^2$ 
10:    endif
11:    // Compute next auxiliary  $s_{i+1}$ 
12:     $s_{i+1} := \dots$ 
13:  endfor
14:   $D^+(n, n) := D(n, n) + (s_n - \tau)$ 

```

One application where the block structure Ω^+ would be known beforehand is for example when a non-blocked factorization $L^+D^+(L^+)^*$ is desired, i.e., with D^+ being diagonal or, equivalently stated, $\Omega^+ = \emptyset$. In §4.2.2 we will present a customized algorithm just for that purpose.

Generally, however, we want to determine a suitable block structure on the fly, for example with the intent to minimize (local) element growth. Then CS 4.1 needs to be augmented with suitable tests to build up Ω^+ . The most convenient position for such a test is right after a d_i^+ has been computed in line 3 to decide if this should start a new 2×2 -pivot in the target factorization, that is, if $i + 1$ should be added to Ω^+ . The concrete shape and form of the test has to depend on the circumstances, which is to say on the block structure Ω in the source. We will develop and discuss a series of usable situation-specific tests on the following pages. But regardless of how they are implemented, we should keep in mind that a 2×2 -pivot may only be chosen if (4.16) is fulfilled for T^+ , that is

$$|d_i^+ c_{i+1}^+| < K_{\square} e_i^2. \quad (4.24)$$

Except for the composition of Ω^+ , the prominent point left open in CS 4.1 is the computation of s_{i+1} in line 12. We did already indicate that this has to depend on the actual case at hand as determined by Ω and Ω^+ and that there are different approaches for some. Doing it wrong is the easiest way of making componentwise mixed relative stability impossible to achieve.

We will tackle the nine cases for computing s_{i+1} from Table 4.1 by grouping them into pieces which can then be considered one at a time:

- (i) Case S1 has already been dealt with for standard `dstqds` based on a e -representation. Accordingly, the reformulation from Table 4.2 can be used and the error analysis from Theorem 2.24 does apply.
- (ii) Cases S2 and S3 state that a block in D corresponds to one in D^+ . This will be our first challenge in §4.2.1.
- (iii) In §4.2.2 we will deal extensively with cases S4 and S6. Those constitute what we call *breaking a block*: single pivots in D^+ where D has a block.
- (iv) The largest chunk will be to tackle S5–S8 in §4.2.3. They have in common that a block in D^+ is (or has just been) introduced where D does not have one—we call this *creating a block*. A special role will fall to S9 and S8, where blocks in D and D^+ do *overlap*, because once these two cases start to alternate and form an *overlap sequence*, the worst-case relative perturbation bounds will depend on the length of the sequence. We will not be able to overcome this problem completely, but it can be controlled in a practicable way.

Our exposition is designed around the principle that we will present, for each of (ii)–(iv), a computational sequence tailored just for the computation of the s_i concerned. These are intended to be used as plugin for the template above and will be accompanied by a complete relative error analysis covering all data involved. The final algorithm and its error analysis in §4.2.4 can then be built by composition.

Error Analysis: Preparations

We assume the reader has built up some skill with mixed relative perturbation analysis up to this point, in particular with the notation we use, so we will skip some elementary steps.

The whole error analysis to follow in this and the subsequent sections assumes that Axiom FP is upheld by the floating-point environment and that no underflow or overflow occurs.

For any quantity a we will normally use \tilde{a} for its perturbed counterpart. It will be beneficial to be able to refer to the individual relative perturbation factors

associated with perturbed quantities; for this purpose we reserve the letter ϱ .

$$\boxed{\tilde{a} = a\varrho(a)}$$

So we use the \sim -notation as map between unperturbed and perturbed data items, whereas ϱ maps data to the factor separating it from the perturbed data. Note that we deliberately let the use of ϱ on a perturbed quantity, like $\varrho(\tilde{a})$, undefined.

Recall our use of the term secondary data for anything (meaningful) which can be derived from a representation's primary data; so far we have already introduced the block determinants $\Delta_i, i + 1 \in \Omega$ as such. Secondary data also has a natural counterpart under the influence of a perturbation, namely the value one obtains if every primary data occurrence in a definition is replaced by the perturbed version. We will extend the \sim -notation to refer to perturbed secondary data as well. Hence, the determinants for 2×2 blocks in \tilde{D} and \tilde{D}^+ are

$$\tilde{\Delta}_i = \tilde{d}_i \tilde{c}_{i+1} - \tilde{e}_i^2, \quad i + 1 \in \Omega, \quad \tilde{\Delta}_i^+ = \tilde{d}_i^+ \tilde{c}_{i+1}^+ - \tilde{e}_i^2, \quad i + 1 \in \Omega^+.$$

Note that, although our lax use of the \sim -notation might suggest otherwise, there still remains the subtle point that we can choose primary perturbations like $d_i \rightsquigarrow \tilde{d}_i$ freely, whereas $\Delta_i \rightsquigarrow \tilde{\Delta}_i$ is an immediate *consequence* once all perturbations to the primary data are fixed.

Together with the \sim -notation, the use of ϱ will also extend in a straightforward manner to secondary data, thus giving meaning to

$$\begin{aligned} \tilde{\Delta}_i &= \Delta_i \varrho(\Delta_i), & i + 1 \in \Omega, & \quad \tilde{\Delta}_i^+ &= \Delta_i^+ \varrho(\Delta_i^+), & i + 1 \in \Omega^+, \\ \tilde{s}_i &= s_i \varrho(s_i), & i = 1, \dots, n, & \quad \tilde{\tau}_i &= \tau \varrho(\tau_i), & i = 1, \dots, n. \end{aligned}$$

Concerning the offdiagonal elements e_i , for a shifted factorization based on our standard representation only their squares e_i^2 will ever be needed, so assume we have them as

$$\text{fl}(e_i^2) = e_i^2(1 + \varepsilon_i), \quad |\varepsilon_i| \leq \epsilon_\diamond, \quad i = 1, \dots, n - 1. \quad (4.25)$$

It will be necessary to relate the block determinants Δ_i and Δ_i^+ as computed in lines 6 and 9 to the exact ones $\tilde{\Delta}_i$ and $\tilde{\Delta}_i^+$ for the perturbed matrices. Based on Axiom FP and (4.25), we can state

$$\begin{aligned} \Delta_i(1 + \beta_\Delta) &= d_i c_{i+1}(1 + \alpha_\Delta) - e_i^2(1 + \varepsilon_i), & \text{for } i + 1 \in \Omega, \\ \Delta_i^+(1 + \beta_\Delta^+) &= d_i^+ c_{i+1}^+(1 + \alpha_\Delta^+) - e_i^2(1 + \varepsilon_i), & \text{for } i + 1 \in \Omega^+, \end{aligned} \quad (4.26)$$

with suitable perturbations $|\alpha_\Delta|, |\alpha_\Delta^+|, |\beta_\Delta|, |\beta_\Delta^+| \leq \epsilon_\diamond$. Those will of course depend on i , but it is not necessary to make this dependency explicit. We will be deliberate to ensure that for all 2×2 -pivots, the condition of computing its determinant is bounded by κ_Δ from (4.20). Then we can invoke Lemma 1.4 to

obtain from (4.26) a connection between the computed value Δ_i and the exact one $\tilde{\Delta}_i$ for perturbed (primary) data as follows:

$$\begin{aligned}\tilde{\Delta}_i &= \Delta_i \varrho(\Delta_i) = d_{i-1} \varrho(d_{i-1}) \cdot c_i \varrho(c_i) - e_i^2 \varrho(e_i^2) \\ &= d_{i-1} c_i (1 + \alpha_\Delta) \underbrace{\frac{\varrho(d_{i-1}) \varrho(c_i)}{(1 + \alpha_\Delta)}}_A - e_i^2 (1 + \varepsilon_i) \underbrace{\frac{\varrho(e_i^2)}{(1 + \varepsilon_i)}}_B \\ &= \Delta_i (1 + \beta_\Delta) (1 + \gamma),\end{aligned}$$

with $|\gamma| \leq \kappa_\Delta \max\{|A-1|, |B-1|\}$. The same technique works for Δ_i^+ and $\tilde{\Delta}_i^+$. To get sharper error bounds one can take common factors of $\varrho(e_i^2)$ and $\varrho(d_i) \varrho(c_{i+1})$ or $\varrho(d_i^+) \varrho(c_{i+1}^+)$ out from under the control of κ_Δ .

Thus, extract any nonzero factors F, F^+ and repeat the steps above to derive two useful formulae that we state for future reference:

$$\begin{aligned}\varrho(\Delta_i) &= (1 + \beta_\Delta) F (1 + \gamma) \quad \text{for } i+1 \in \Omega, \quad \text{where} \\ |\gamma| &\leq \kappa_\Delta \cdot \max\left\{ \left| \frac{\varrho(d_i) \varrho(c_{i+1})}{(1 + \alpha_\Delta) F} - 1 \right|, \left| \frac{\varrho(e_i^2)}{(1 + \varepsilon_i) F} - 1 \right| \right\},\end{aligned}\tag{4.27}$$

$$\begin{aligned}\varrho(\Delta_i^+) &= (1 + \beta_\Delta^+) F^+ (1 + \gamma^+) \quad \text{for } i+1 \in \Omega^+, \quad \text{where} \\ |\gamma^+| &\leq \kappa_\Delta \cdot \max\left\{ \left| \frac{\varrho(d_i^+) \varrho(c_{i+1}^+)}{(1 + \alpha_\Delta^+) F^+} - 1 \right|, \left| \frac{\varrho(e_i^2)}{(1 + \varepsilon_i) F^+} - 1 \right| \right\}.\end{aligned}$$

As a side remark, these clearly reveal the secondary nature of $\varrho(\Delta_i)$ and $\varrho(\Delta_i^+)$.

Except for few special cases, we will perturb the data influencing s_i , and maybe also the shift $\tilde{\tau}_i$, just so that

$$s_i - \tau = \tilde{s}_i - \tilde{\tau}_i\tag{4.28}$$

holds. This means the exact difference of the quantities s_i and τ (which are floating-point numbers stored in the machine) equals the exact difference of the perturbed data \tilde{s}_i and $\tilde{\tau}_i$ (which will in general not be representable as floating-point numbers).

Provided the relation (4.28) holds, there is an obvious way to perturb the diagonal data of D and D^+ such that (4.23) is achieved. Assuming the computation in line 3 obeys

$$D^+(i, i)(1 + \delta_i^+) = D(i, i) + (s_i - \tau)/(1 + \sigma_i), \quad |\delta_i^+|, |\sigma_i| \leq \epsilon_\diamond,\tag{4.29}$$

the way to go is

$$\varrho(D(i, i)) := (1 + \sigma_i), \quad \varrho(D^+(i, i)) := (1 + \delta_i^+)(1 + \sigma_i).\tag{4.30}$$

These will serve as our default perturbation.

To attain the relation $s_i - \tau = \tilde{s}_i - \tilde{\tau}_i$ in the first place, there are basically two ways. The obvious one is to choose $e_i \rightsquigarrow \tilde{e}_i$ just so that the computed s_i becomes exact, i.e., $s_i = \tilde{s}_i$. Then there is even no need to touch the shift, as $\tilde{\tau}_i := \tau$ will do the trick. An alternative is made possible if s_i is not too large in magnitude compared to the shift, e.g.,

$$|s_i| \leq R|\tau|$$

for some parameter R . Then we can achieve $s_i - \tau = \tilde{s}_i - \tilde{\tau}_i$ for *every* choice of $e_i \rightsquigarrow \tilde{e}_i$ by moving any “excess” from \tilde{s}_i to $\tilde{\tau}_i$, in the form

$$\tilde{\tau}_i - \tau = \tilde{s}_i - s_i \quad \implies \quad |\varrho(\tilde{\tau}_i) - 1| \leq R|\varrho(s_i) - 1|, \quad (4.31)$$

defining $\tilde{\tau}_i$. This provides us with one additional degree of freedom in the choice of \tilde{e}_i , which can be used to fix some other critical computation. Note that, effectively, we did cast a relative perturbation from one quantity (\tilde{s}_i) as an absolute one and then wrote it again as a relative one, but for a different quantity (τ). This technique will prove to be a crucial ingredient for the error analysis to succeed.

This closes the general preparations. Note that, whenever (4.30) can be used, all that remains to be done is to specify fitting perturbations $e_i \rightsquigarrow \tilde{e}_i$ for the offdiagonal data.

4.2.1 Keep a block

Block factorizations are mostly harmless as long as the block structure is not changed. With respect to Table 4.1 this comprises the cases S1, S2 and S3. For S1 a single pivot in the source corresponds to a single pivot in the target—that is just like standard `dstqds` and the error analysis from Theorem 2.24 is fully applicable. In this section we will deal with the cases S2 and S3; together they constitute that a block in \mathbf{D} is reproduced in \mathbf{D}^+ , that is, we *keep* the block.

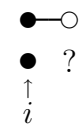
We begin by devising a criterion to determine when a block should be kept. Assume we have a block in the source covering indices i and $i+1$, that is, $i+1 \in \Omega$. The option of keeping the block does only present itself if we did not already choose a block in the target at $i-1$, so assume $i \notin \Omega^+$. Considering again Table 4.1, the choice between keeping the block or not corresponds to the choice between cases S2 or S4.

It is desirable to keep the structure, but that is not always possible. At least, we have to ensure that each block in \mathbf{D}^+ satisfies condition (4.24). In this situation the data $\mathbb{T}(i+1, i+1) = c_{i+1}$ is readily available in our standard representation, so just one extra addition gives $\mathbb{T}^+(i+i, i+1) = c_{i+1} - \tau$. This yields the following test.

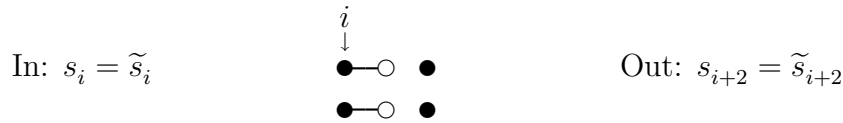
Block-Criterion I.

Fix a parameter $K_1 \leq K_{\square}^*$. Keep the block if

$$|d_i^+(c_{i+1} - \tau)| < K_1 e_i^2.$$



Now focus on the case that a block $i + 1 \in \Omega$ is indeed kept, that is, we assume the following situation:



As depicted, we require s_i as input and assume it to be exact with respect to the perturbed data; the produced output s_{i+2} shall have the same property. The computation will go through case S2 followed by case S3, but to ensure stability we need to take the reformulation from Table 4.2 for the latter. This leads to the following computational sequence, to be integrated with our algorithmic template CS 4.1.

CS 4.2: <i>Keep a block</i>
1: $s_{i+1} := 0$
2: $x := e_i^2(s_i - \tau) - d_i d_i^+ \tau$
3: $s_{i+2} := e_{i+1}^2 x / (\Delta_i \Delta_i^+)$

Recall our goal: We need to find perturbations to the primary data involved, namely $d_i, d_i^+, c_{i+1}, c_{i+1}^+, e_i$ and e_{i+1} , and optionally also the shifts τ_i, τ_{i+1} , such that the relation(4.23) is fulfilled for i and $i + 1$, with respect to the exact adjustments $\tilde{s}_i, \tilde{s}_{i+1}$ for the perturbed data. Combined with the In/Out-specification above, this boils down to achieving

$$\begin{aligned} \tilde{d}_i^+ &\stackrel{!}{=} \tilde{d}_i + (s_i - \tilde{\tau}_i), && \text{since } s_i = \tilde{s}_i, \\ \tilde{c}_{i+1}^+ &\stackrel{!}{=} \tilde{c}_{i+1} - \tilde{\tau}_{i+1}, && \text{as } s_{i+1} = 0, \\ \tilde{s}_{i+2} &\stackrel{!}{=} s_{i+2}. \end{aligned}$$

Concerning the intermediate x and s_{i+2} , under the assumption that the execution environment may obey Axiom FP we will have something like

$$x(1 + \beta_x) = e_i^2(s_i - \tau) \frac{(1 + \varepsilon_i)}{(1 + \sigma_i)(1 + \alpha_1)} - d_i d_i^+ \tau(1 + \alpha_2), \tag{4.32}$$

$$s_{i+2}(1 + \beta_s) = e_{i+1}^2(1 + \varepsilon_{i+1}) x / (\Delta_i \Delta_i^+), \tag{4.33}$$

where $|\alpha_1|, |\beta_x| \leq \epsilon_\circ$, $\alpha_2 \doteq \epsilon(2)$, $\beta_s \doteq \epsilon(3)$ and $\varepsilon_i, \varepsilon_{i+1}, \sigma_i$ stem from (4.25) and (4.29).

The shift need not be perturbed: $\tilde{\tau}_i = \tilde{\tau}_{i+1} = \tau$. Then, as per requirement $s_i = \tilde{s}_i$ and trivially also $s_{i+1} = 0 = \tilde{s}_{i+1}$, the default perturbations defined in (4.30) can be employed for d_i, d_i^+, c_{i+1} and c_{i+1}^+ , to the effect

$$\begin{aligned} \tilde{d}_i &:= d_i(1 + \sigma_i), & \tilde{c}_{i+1} &:= c_{i+1}(1 + \sigma_{i+1}), \\ \tilde{d}_i^+ &:= d_i^+(1 + \delta_i^+)(1 + \sigma_i), & \tilde{c}_{i+1}^+ &:= c_{i+1}^+(1 + \delta_{i+1}^+)(1 + \sigma_{i+1}). \end{aligned} \quad (4.34)$$

So far this gives us the desired relation (4.23) for i and $i + 1$. The sole remaining task now is to assure that the computed s_{i+2} is again exact for the perturbed data, i.e., $s_{i+2} = \tilde{s}_{i+2}$. We still have not touched either e_i or e_{i+1} and can perturb them freely to achieve this goal.

As a first step, perturb $e_i \rightsquigarrow \tilde{e}_i$ to control the subtraction involved in computing x . The goal is to get the exact intermediate for perturbed data, \tilde{x} , to be a small relative perturbation of the computed value, x . Based on (4.32) we set

$$\tilde{e}_i := e_i \cdot \sqrt{(1 + \delta_i^+) \frac{(1 + \varepsilon_i)(1 + \sigma_i)}{(1 + \alpha_1)(1 + \alpha_2)}} \implies |\varrho(e_i) - 1| \leq \epsilon^{[4]}(3) \quad (4.35)$$

and use $s_i - \tau = \tilde{s}_i - \tilde{\tau}_i$ to obtain

$$\begin{aligned} \tilde{x} &= \tilde{e}_i^2(s_i - \tau) - \tilde{d}_i \tilde{d}_i^+ \tau \\ &= x(1 + \beta_x)(1 + \sigma_i)^2(1 + \delta_i^+) / (1 + \alpha_2) \\ &= x\varrho(x), \end{aligned}$$

defining $\varrho(x)$.

The second (and last) step is to perturb $e_{i+1} \rightsquigarrow \tilde{e}_{i+1}$ to get $\tilde{s}_{i+2} = s_{i+2}$. As the computation of s_{i+2} involves the block determinants Δ_i and Δ_i^+ , we have to control the perturbation's effect on them. Just for this purpose we did craft the tool (4.27), which comes to its first use now.

Straightforward application of the formula (4.27), with $F := (1 + \sigma_i)/(1 + \alpha_2)$, yields

$$\tilde{\Delta}_i = \Delta_i(1 + \beta_\Delta)(1 + \sigma_i)(1 + \gamma) / (1 + \alpha_2), \quad (4.36)$$

with

$$|\gamma| \leq \kappa_\Delta \cdot \max \left\{ \left| \frac{\varrho(d_i)\varrho(c_{i+1})(1 + \alpha_2)}{(1 + \alpha_\Delta)(1 + \sigma_i)} - 1 \right|, \left| \frac{\varrho(e_i^2)(1 + \alpha_2)}{(1 + \varepsilon_i)(1 + \sigma_i)} - 1 \right| \right\}. \quad (4.37)$$

By (4.34) and (4.35) we can write this as

$$|\gamma| \leq \kappa_\Delta \cdot \max \left\{ \left| \frac{(1 + \sigma_{i+1})(1 + \alpha_2)}{(1 + \alpha_\Delta)} - 1 \right|, \left| \frac{(1 + \delta_i^+)}{(1 + \sigma_i)(1 + \alpha_1)} - 1 \right| \right\}. \quad (4.38)$$

Finally, Corollary 1.12 lets us simplify to

$$|\gamma| \doteq \kappa_\Delta \epsilon(4). \quad (4.39)$$

In analogous fashion we obtain a bound for $\varrho(\Delta_i^+)$, using the lower formula from (4.27), with $F^+ := (1 + \sigma_i)(1 + \delta_i^+)$, for

$$\tilde{\Delta}_i^+ = \Delta_i^+(1 + \beta_\Delta^+)(1 + \sigma_i)(1 + \delta_i^+)(1 + \gamma^+), \quad \gamma^+ \doteq \kappa_\Delta \epsilon(3). \quad (4.40)$$

The purpose for factoring out $(1 + \alpha_2)$ from $\tilde{\Delta}_i$ was to cancel out with the one from $\varrho(x)$ in

$$\begin{aligned} \tilde{e}_{i+1} &:= e_{i+1} \sqrt{\frac{(1 + \varepsilon_{i+1})}{(1 + \beta_s)} \varrho(\Delta_i) \varrho(\Delta_i^+) / \varrho(x)} \\ &= e_{i+1} \sqrt{\frac{(1 + \beta_\Delta)(1 + \beta_\Delta^+)}{(1 + \beta_x)(1 + \beta_s)} (1 + \varepsilon_{i+1})(1 + \gamma)(1 + \gamma^+)} \\ &\doteq e_{i+1} \sqrt{(1 + \epsilon(7))(1 + \epsilon(7\kappa_\Delta))} \\ &\implies |\varrho(e_{i+1}) - 1| \leq \epsilon^{[4]}(\frac{7}{2} + \frac{7}{2}\kappa_\Delta). \end{aligned} \quad (4.41)$$

Taking everything together, we obtain the desired relation

$$s_{i+2} = \tilde{e}_{i+1}^2 (\tilde{e}_i^2 (s_i - \tau) - \tilde{d}_i \tilde{d}_i^+ \tau) / (\tilde{\Delta}_i \tilde{\Delta}_i^+) = \tilde{s}_{i+2}.$$

The following result summarizes the preceding error analysis for keeping a block.

Lemma 4.6. *For the case that a block in the source is reproduced in the target, $i + 1 \in \Omega \cap \Omega^+$, let in CS 4.1 the auxiliaries s_{i+1} and s_{i+2} be computed as in CS 4.2. Then we can find perturbations*

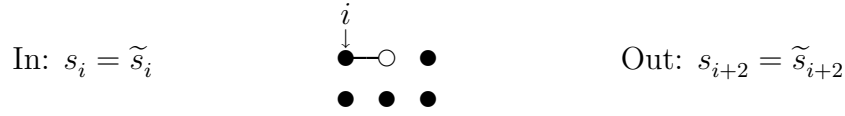
$$\begin{cases} d_i \rightsquigarrow \tilde{d}_i & \doteq \epsilon(1), & d_i^+ \rightsquigarrow \tilde{d}_i^+ & \doteq \epsilon(2), \\ c_{i+1} \rightsquigarrow \tilde{c}_{i+1} & \doteq \epsilon(1), & c_{i+1}^+ \rightsquigarrow \tilde{c}_{i+1}^+ & \doteq \epsilon(2), \\ e_i \rightsquigarrow \tilde{e}_i & \doteq \epsilon^{[4]}(3), & e_{i+1} \rightsquigarrow \tilde{e}_{i+1} & \doteq \epsilon^{[4]}(\frac{7}{2} + \frac{7}{2}\kappa_\Delta), \end{cases}$$

such that

$$\tilde{d}_i^+ = \tilde{d}_i + s_i - \tau, \quad \tilde{c}_{i+1}^+ = \tilde{c}_{i+1} - \tau, \quad \text{and} \quad s_{i+2} = \tilde{s}_{i+2}.$$

4.2.2 Break a block

In the previous section we dealt with the situation that a block in the source D is kept for the target D^+ . Now we will consider how to break a block without any overlap, that is, without creating a new block ending within $\{i, i + 1, i + 2\}$:



As depicted, we require s_i as input and assume it to be exact with respect to the perturbed data, and we want to deliver s_{i+2} enjoying the same property. We will not assume that the decision to refrain from adding either one of i , $i + 1$ or $i + 2$ to Ω^+ was supported by BC-I, but nevertheless we require

$$d_i^+ \neq 0 \quad \text{and} \quad d_{i+1}^+ \neq 0,$$

as otherwise the factorization would not be possible.

With respect to Table 4.1, the computation will go through S4 followed by S6. There are different ways to compute s_{i+2} . From case S6 in Table 4.1, applied to s_{i+2} , we get the formula

$$s_{i+2} = \frac{e_{i+1}^2}{\Delta_i d_{i+1}^+} \underbrace{[d_i d_{i+1}^+ - \Delta_i]}_{=: x}, \tag{4.42}$$

revealing the intermediate quantity x whose stable computation is a critical ingredient for breaking a block. This meaning of x will remain in effect throughout this section. The reformulation from Table 4.2 for case S6 shows that x can also be written as

$$x = d_i(s_{i+1} - \tau) + e_i^2. \tag{4.43}$$

Two points should be clear:

- For having the slightest chance of finding a perturbation for e_{i+1} such that s_{i+2} becomes exact, we must compute x stably, meaning the computed x should be only a small relative perturbation away from the exact \tilde{x} for perturbed data.
- Neither one of the two ways for computing x introduced above will be stable in general. This remains true even if one would assume that Block-Criterion I was not fulfilled.

These points make the task of breaking a block special, because they effectively force us to include a branch in the computation.

Computing x as in (4.43) is advantageous, because it reuses the intermediate $s_{i+1} - \tau$, which is required to compute d_{i+1}^+ anyway (line 3 of CS 4.1). But for a relative error analysis, this approach is problematic because it uses e_i explicitly again, although s_{i+1} does already depend on e_i . An alternative formulation for x is

$$x = \frac{e_i^2}{d_i^+}(s_i - \tau) - d_i \tau = -s_{i+1}(s_i - \tau) - d_i \tau. \tag{4.44}$$

This one is easily derived from (4.43) if one puts the identities $s_i - \tau = d_i^+ - d_i$ and $s_{i+1} = -e_i^2/d_i^+$ to good use. Again it should be clear that the outermost subtraction involved cannot bode well for all configurations, but at least the double dependency of e_i is removed.

Both ways to compute x that were just introduced have their uses. So let us combine them to the following prototype computational sequence for breaking a block. Again we state only the parts relevant for computing the auxiliaries s_{i+1} and s_{i+2} and assume the rest is done according to CS 4.1. The keyword **branch** is used just as it was in §2.4, cf. the remarks on page 85.

CS 4.3: *Branches to break a block*

1:	$s_{i+1} := -e_i^2/d_i^+$	
	branch I :	
2:	$x := d_i(s_{i+1} - \tau) + e_i^2$	
	branch II :	
2:	$x := -s_{i+1}(s_i - \tau) - d_i\tau$	
	endbranch	
3:	$s_{i+2} := e_{i+1}^2 x / (\Delta_i d_{i+1}^+)$	

Let us without delay identify the effects of executing these steps using floating-point arithmetic:

$$\begin{aligned}
 s_{i+1}(1 + \beta_s) &= -e_i^2(1 + \varepsilon_i)/d_i^+, \\
 x(1 + \beta_x) &= \begin{cases} \frac{d_i(s_{i+1} - \tau)}{(1 + \alpha_1)(1 + \sigma_{i+1})} + e_i^2(1 + \varepsilon_i), & \text{for branch I,} \\ -\frac{s_{i+1}(s_i - \tau)}{(1 + \alpha_2)(1 + \sigma_i)} - d_i\tau(1 + \alpha_3), & \text{for branch II,} \end{cases} \\
 s_{i+2}(1 + \beta'_s) &= e_{i+1}^2(1 + \varepsilon_{i+1})x / (\Delta_i d_{i+1}^+),
 \end{aligned} \tag{4.45}$$

where $|\alpha_1|, |\alpha_2|, |\alpha_3|, |\beta_x|, |\beta_s| \leq \epsilon_\circ$, $\beta'_s \doteq \epsilon(3)$, and $\varepsilon_i, \varepsilon_{i+1}, \sigma_i, \sigma_{i+1}$ stem from (4.25) and (4.29).

We would like to note at this point that we did investigate even other ways to compute s_{i+2} . But, as was already indicated, we do not see a way to lead a mixed relative error analysis to completion if only one computational branch (I, II or other ones) is used for all cases. However, what we can (and will) do is to show that the combination of branches I and II, together with the right test to choose between them, leads to success. To this end, we highlight the following conditions under which the computation according to each branch will be stable:

- (a) Branch I works if the computation of x in line 2 is well-conditioned. Then we are free to fix $\varrho(e_i)$ for any choice of $\varrho(d_i)$ such that $s_{i+1} = \tilde{s}_{i+1}$ holds, while still being able to control the effect on x . We will explore this condition in Lemma 4.7.

- (b) Branch I is also fine if s_{i+1} is not much larger than τ in magnitude. Then we can employ the technique surmised in (4.31) to modify the shift, opening up the freedom to perturb e_i to control the subtraction involved in computing x in line 2. Lemma 4.8 will deal with this case.
- (c) Branch II works if s_{i+1} and τ are not about equal in magnitude, i.e., there would be no cancellation in $s_{i+1} - \tau$ (which is nevertheless not computed for Branch II). This claim will be established in Lemma 4.9.

These are just a selection out of a variety of conditions we studied while investigating how to break a block. For example, similarly to (a) one can show that branch II is fine if the computation of x there (line 2) is well-conditioned. However, just the three conditions stated above will contribute to our algorithm to break a block, so we will present only them in detail. Indeed, as $|s_{i+1}| \gg |\tau|$ excludes the possibility of harmful cancellation between s_{i+1} and τ , just (b) and (c) alone would suffice to cover all input configurations, so one could even skip (a).

The error analyses in the three following lemmas have in common that the default perturbation (4.30) is deployed for d_i, d_i^+ , i.e., we set

$$\tilde{d}_i := d_i(1 + \sigma_i), \quad \tilde{d}_i^+ := d_i^+(1 + \delta_i^+)(1 + \sigma_i). \quad (4.46)$$

Furthermore, the shift for index i is left unperturbed, $\tilde{\tau}_i := \tau$. One main goal will always be to prove that the computed x has a small relative distance to the exact one for the perturbed data,

$$\tilde{x} = x\varrho(x) = \tilde{d}_i(\tilde{s}_{i+1} - \tilde{\tau}_{i+1}) + \tilde{e}_i^2 = -\tilde{s}_{i+1}(\tilde{s}_i - \tau) - \tilde{d}_i\tilde{\tau}_{i+1}.$$

Provided we have an acceptable bound on $|\varrho(x) - 1|$, based on (4.45) we can then attain $\tilde{s}_{i+2} = s_{i+2}$ by defining \tilde{e}_i according to

$$\begin{aligned} \tilde{e}_{i+1} &:= e_{i+1} \sqrt{\frac{(1 + \varepsilon_{i+1})}{(1 + \beta'_s)} \varrho(\Delta_i) \varrho(d_{i+1}^+) / \varrho(x)} \\ \implies \varrho(e_{i+1}) &\doteq (1 + \epsilon^{[6]}(2)) \sqrt{\varrho(\Delta_i) \varrho(d_{i+1}^+) / \varrho(x)}. \end{aligned} \quad (4.47)$$

So, what is left to do is to perturb three data items and the shift for index $i + 1$, namely

$$c_{i+1} \rightsquigarrow \tilde{c}_{i+1}, \quad d_{i+1}^+ \rightsquigarrow \tilde{d}_{i+1}^+, \quad e_i \rightsquigarrow \tilde{e}_i, \quad \tau \rightsquigarrow \tilde{\tau}_{i+1}, \quad (4.48)$$

in order to ensure

$$\tilde{d}_{i+1}^+ \stackrel{!}{=} \tilde{c}_{i+1} + \tilde{s}_{i+1} - \tilde{\tau}_{i+1} \quad \text{and} \quad \tilde{s}_{i+2} \stackrel{!}{=} s_{i+2}. \quad (4.49)$$

With these preparations done, the following three lemmas can be considered separately, as they do not depend on each other.

Lemma 4.7. *For branch I, let*

$$\kappa[x] := \kappa_+(\text{fl}(d_i(s_{i+1} - \tau)), \text{fl}(e_i^2))$$

be benign. Then we can find a perturbation

$$\begin{cases} c_{i+1} \rightsquigarrow \tilde{c}_{i+1} & \doteq \epsilon(1), & e_i \rightsquigarrow \tilde{e}_i & \doteq \epsilon^{[4]}(2), \\ d_{i+1}^+ \rightsquigarrow \tilde{d}_{i+1}^+ & \doteq \epsilon(2), & e_{i+1} \rightsquigarrow \tilde{e}_{i+1} & \doteq \epsilon^{[6]}(4 + \kappa[x] + \kappa_\Delta), \end{cases}$$

such that (4.49) is fulfilled with $\tilde{\tau}_{i+1} = \tau$. The computed adjustment s_{i+1} will be exact, that is, $s_{i+1} = \tilde{s}_{i+1}$.

Proof. Based on (4.45) we attain $\tilde{s}_{i+1} = s_{i+1}$ by setting

$$\begin{aligned} \tilde{e}_i &:= e_i \sqrt{\varrho(d_i^+)(1 + \varepsilon_i)/(1 + \beta_s)} \\ &= e_i \sqrt{(1 + \varepsilon_i)(1 + \sigma_i)(1 + \delta_i^+)/(1 + \beta_s)} \\ &\implies \varrho(e_i) \doteq 1 + \epsilon^{[4]}(2). \end{aligned} \tag{4.50}$$

In cooperation with the default perturbations (4.30) for c_{i+1} , d_{i+1}^+ this gives us

$$\tilde{d}_{i+1}^+ = \tilde{c}_{i+1} + s_{i+1} - \tau.$$

For the intermediate x we have

$$\begin{aligned} \tilde{x} &= \tilde{d}_i(s_{i+1} - \tau) + \tilde{e}_i^2 && \text{since } s_{i+1} = \tilde{s}_{i+1}, \tilde{\tau}_i = \tau, \\ &= d_i(s_{i+1} - \tau)(1 + \sigma_i) + e_i^2(1 + \varepsilon_i) && \text{by (4.46) and (4.50).} \end{aligned}$$

Now we cast this in terms of (4.45) and invoke Lemma 1.4. Using the prerequisite on $\kappa[x]$ we see that the perturbation's effect on x can be controlled as

$$\varrho(x) = (1 + \beta_x)(1 + \sigma_i)(1 + \xi), \quad \xi \doteq \epsilon(2\kappa[x]).$$

Since $\varrho(\tilde{d}_i)$ and $\varrho(e_i^2)$ have the common factor $F := (1 + \sigma_i)$, (4.27) lets us conclude that the computed $\tilde{\Delta}_i$ will relate to the exact block determinant for perturbed data according to $\tilde{\Delta}_i = \Delta_i \varrho(\Delta_i)$, where

$$\varrho(\Delta_i) = (1 + \beta_\Delta)(1 + \sigma_i)(1 + \gamma), \quad \gamma \doteq \epsilon(2\kappa_\Delta).$$

Finally, plug the obtained $\varrho(\tilde{d}_i)$, $\varrho(x)$ and $\varrho(\Delta_i)$ into (4.47) and cancel terms to determine \tilde{e}_{i+1} such that $\tilde{s}_{i+2} = s_{i+2}$ holds. \square

Lemma 4.8. *For branch I, let*

$$|s_{i+1}| \leq R|\tau|$$

for a parameter $R > 1$. Then there is a perturbation

$$\begin{cases} c_{i+1} \rightsquigarrow \tilde{c}_{i+1} & \doteq \epsilon(1), & e_i \rightsquigarrow \tilde{e}_i & \doteq \epsilon^{[4]}(2), \\ d_{i+1}^+ \rightsquigarrow \tilde{d}_{i+1}^+ & \doteq \epsilon(2), & e_{i+1} \rightsquigarrow \tilde{e}_{i+1} & \doteq \epsilon^{[4]}(\frac{9}{2} + \frac{1}{2}\kappa_\Delta), \\ \tau \rightsquigarrow \tilde{\tau}_{i+1} & \doteq \epsilon(4R), & & \end{cases}$$

such that (4.49) is fulfilled. The computed adjustment s_{i+1} will satisfy

$$\tilde{s}_{i+1} \doteq s_{i+1}(1 + \epsilon(4)).$$

Proof. We have to assume that the computation of x is ill-conditioned. Therefore we choose the perturbation $e_i \rightsquigarrow \tilde{e}_i$ specifically to safeguard the subtraction involved in computing x , namely as

$$e_i := e_i \cdot \sqrt{(1 + \varepsilon_i)(1 + \sigma_i)(1 + \sigma_{i+1})(1 + \alpha_1)}.$$

This gives us

$$x(1 + \beta_x)(1 + \sigma_i)(1 + \sigma_{i+1})(1 + \alpha_1) = \tilde{d}_i(s_{i+1} - \tau) + \tilde{e}_i^2,$$

as well as

$$\begin{aligned} s_{i+1} &= -\frac{\tilde{e}_i^2}{\tilde{d}_i^+}(1 + \delta_i^+)[(1 + \sigma_{i+1})(1 + \alpha_1)(1 + \beta_s)]^{-1} \\ &\implies \varrho(s_{i+1}) \doteq (1 + \epsilon(4)). \end{aligned}$$

Now employ the precondition and invoke (4.31) to define $\tilde{\tau}_{i+1}$ satisfying

$$\varrho(\tau_{i+1}) \doteq 1 + \epsilon(4R),$$

such that $s_{i+1} - \tau = \tilde{s}_{i+1} - \tilde{\tau}_{i+1}$. Together with the default perturbations (4.30) for c_{i+1} , d_{i+1}^+ we get

$$\tilde{d}_{i+1}^+ = \tilde{c}_{i+1} + \tilde{s}_{i+1} - \tilde{\tau}_{i+1}.$$

Concerning the block determinant Δ_i , note that $\varrho(d_i)\varrho(c_{i+1})$ and $\varrho(e_i^2)$ have the common factor $F = (1 + \sigma_i)(1 + \sigma_{i+1})$, so our tool (4.27) gives

$$\varrho(\tilde{\Delta}_i) \doteq (1 + \beta_\Delta)(1 + \sigma_i)(1 + \sigma_{i+1})(1 + \epsilon(\kappa_\Delta)).$$

Take everything together, invoke (4.47) and cancel terms to find the right perturbation for e_{i+1} . \square

Lemma 4.9. For branch II, let

$$|s_{i+1}| > R|\tau|$$

for a parameter $R > 1$ and define $R^* := (R - 1)^{-1}$. Then there is a perturbation

$$\begin{cases} c_{i+1} \rightsquigarrow \tilde{c}_{i+1} \doteq \epsilon(3 + 2R^*), & e_i \rightsquigarrow \tilde{e}_i \doteq \epsilon^{[4]}(3), \\ d_{i+1}^+ \rightsquigarrow \tilde{d}_{i+1}^+ \doteq \epsilon(4 + 2R^*), & e_{i+1} \rightsquigarrow \tilde{e}_{i+1} \doteq \epsilon^{[6]}(\frac{11}{2} + 2\kappa_\Delta + (\kappa_\Delta + 1)R^*), \end{cases}$$

such that (4.49) is fulfilled with $\tilde{\tau}_i = \tau$. The computed adjustment s_{i+1} will satisfy

$$\tilde{s}_{i+1} \doteq s_{i+1}(1 + \epsilon(2)).$$

Proof. As we cannot assume the computation of x to be well-conditioned, our first action is to perturb e_i to safeguard it. Setting

$$\tilde{e}_i := e_i \sqrt{\frac{(1 + \varepsilon_i)(1 + \sigma_i)(1 + \delta_i^+)}{(1 + \alpha_2)(1 + \alpha_3)(1 + \beta_s)}}$$

we reach that goal, because in concert with (4.45) it gives us

$$s_{i+1} = -\frac{\tilde{e}_i^2}{\tilde{d}_i^+}(1 + \alpha_2)(1 + \alpha_3) \implies \varrho(s_{i+1}) \doteq (1 + \epsilon(2)),$$

as well as

$$x(1 + \beta_x)(1 + \sigma_i)/(1 + \alpha_3) = \frac{\tilde{e}_i^2}{\tilde{d}_i^+}(s_i - \tau) - \tilde{d}_i^+ \tau.$$

The precondition implies $|s_{i+1}| < (1 + R^*)|s_{i+1} - \tau|$. Hence

$$\begin{aligned} \tilde{s}_{i+1} - \tau &= s_{i+1}(1 + \alpha_2)^{-1}(1 + \alpha_3)^{-1} - \tau \\ &= (s_{i+1} - \tau)(1 + \zeta), \quad \text{with } \zeta \doteq \epsilon(2 + 2R^*). \end{aligned}$$

Thus, together with (4.29) we can achieve $\tilde{d}_{i+1}^+ = \tilde{c}_{i+1} + \tilde{s}_{i+1} - \tau$ through

$$\begin{aligned} \tilde{c}_{i+1} &:= c_{i+1} \cdot (1 + \sigma_{i+1})(1 + \zeta), \\ \tilde{d}_{i+1}^+ &:= d_{i+1}^+ \cdot (1 + \delta_{i+1}^+)(1 + \sigma_{i+1})(1 + \zeta). \end{aligned}$$

Concerning the block determinant, invoke (4.27) with $F := (1 + \sigma_i)$ for

$$\varrho(\Delta_i) = (1 + \beta_\Delta)(1 + \sigma_i)(1 + \gamma), \quad \gamma \doteq \kappa_\Delta \epsilon(4 + 2R^*).$$

Take everything together and use (4.47) for the definition of \tilde{e}_{i+1} . \square

Putting everything together

We propose the following computational sequence for breaking a block. It is designed to exploit that the preconditions for Lemmas 4.8 and 4.9 mesh perfectly.

CS 4.4: *Break a block*

```

1:    $s_{i+1} := -e_i^2/d_i^+$ 
2:   if  $|s_{i+1}| \leq R|\tau|$  or  $\text{sign}(d_i) \neq \text{sign}(d_i^+)$  then
3:     // branch I
4:      $x := d_i(s_{i+1} - \tau) + e_i^2$ 
5:   else
6:     // branch II
7:      $x := -s_{i+1}(s_i - \tau) - d_i\tau$ 
8:   endif
9:    $s_{i+2} := e_{i+1}^2 x / (\Delta_i d_{i+1}^+)$ 

```

A complete error analysis for this sequence is now a direct reward from Lemmas 4.7–4.9. In fact, Lemma 4.7 is not strictly required to break a block stably. We use it in a supporting role to alleviate some of the rather largish error bounds from Lemma 4.9 in practice. However, we do not want to test for $\kappa[x]$ directly, as such a test would be rather expensive. It turns out that the cheap test $\text{sign}(d_i) \neq \text{sign}(d_i^+)$ leads all cases with $\kappa[x] = 1$ into branch I, and hence into the purview of Lemma 4.7.

Corollary 4.10. *For the case that a block in \mathbf{D} is broken without overlap,*

$$i + 1 \in \Omega \quad \text{and} \quad \Omega^+ \cap \{i - 1, i, i + 1\} = \emptyset,$$

let in CS 4.1 the auxiliaries s_{i+1} and s_{i+2} be computed as in CS 4.4 above, with a parameter $R > 1$. Let $R^* := (R - 1)^{-1}$. Then there is a perturbation

$$\left\{ \begin{array}{ll} d_i \rightsquigarrow \tilde{d}_i \doteq \epsilon(1), & d_i^+ \rightsquigarrow \tilde{d}_i^+ \doteq \epsilon(2), \\ c_{i+1} \rightsquigarrow \tilde{c}_{i+1} \doteq \epsilon(3 + 2R^*), & d_{i+1}^+ \rightsquigarrow \tilde{d}_{i+1}^+ \doteq \epsilon(4 + 2R^*), \\ e_i \rightsquigarrow \tilde{e}_i \doteq \epsilon^{[4]}(3), & e_{i+1} \rightsquigarrow \tilde{e}_{i+1} \doteq \epsilon^{[6]}(\frac{11}{2} + 2\kappa_\Delta + (\kappa_\Delta + 1)R^*), \\ \tau \rightsquigarrow \tilde{\tau}_{i+1} \doteq \epsilon(4R), & \end{array} \right.$$

such that

$$\tilde{d}_i^+ = \tilde{d}_i + s_i - \tau, \quad \tilde{d}_{i+1}^+ = \tilde{c}_{i+1} + \tilde{s}_{i+1} - \tilde{\tau}_{i+1}, \quad \text{and} \quad s_{i+2} = \tilde{s}_{i+2}.$$

The computed adjustment s_{i+1} will satisfy $\tilde{s}_{i+1} \doteq s_{i+1}(1 + \epsilon(4))$.

Proof. The stated bounds are the worst of Lemmas 4.7–4.9, we have just to demonstrate that one of them is always applicable.

Clearly, if branch II is taken then Lemma 4.9 can be used and Lemma 4.8 applies should the computation follow branch I when $|s_{i+1}| \leq R|\tau|$.

What remains to be considered is that branch I is taken with $|s_{i+1}| > R|\tau|$ and $\text{sign}(d_i) \neq \text{sign}(d_i^+)$. We have

$$\begin{aligned} \text{sign}(d_i(s_{i+1} - \tau)) &= \text{sign } d_i \cdot \text{sign } s_{i+1}, & R > 1 \Rightarrow |s_{i+1}| > |\tau|, \\ &= \text{sign } d_i \cdot \text{sign } -d_i^+, & s_{i+1} &= -e_i^2/d_i^+, \\ &\in \{0, 1\}. & \text{sign}(d_i) &\neq \text{sign}(d_i^+). \end{aligned}$$

This gives $\kappa[x] = 1$ for Lemma 4.7. \square

An important application of breaking blocks is when we desire a non-blocked target factorization $L^+D^+(L^+)^*$ with D^+ being diagonal, i.e., $\Omega^+ = \emptyset$. Computation of a general (blocked to blocked) factorization is expensive due to the conditionals involved. For an MR³-algorithm based on block factorizations as representations for inner nodes, there is really no need to employ blocks during bisection or computing eigenvectors, as there element growth in the target has no effect on accuracy. Algorithm 4.5 puts the content of the previous pages to good use by providing a non-blocked factorization.

Theorem 4.11 (Error Analysis for blocked to non-blocked `dstqds`)

Let Algorithm 4.5 be executed without underflow or overflow in an environment that satisfies Axiom FP, and let all blocks in D satisfy (4.17).

Then the diagram in Figure 4.2 commutes, that is, there are perturbations to the inputs and outputs such that

$$LDL^* - \text{diag}(\tilde{\tau}_i) = L^+D^+(L^+)^*$$

holds exactly. The perturbations can be bounded depending on the parameters R and K_\square according to Corollary 4.10; for specific choices the resulting bounds are compiled in Table 4.3.

Proof. This is just Corollary 4.10 combined with the error analysis of non-blocked `dstqds` (e -representation) from Theorem 2.24. The parameters $R = 3$, $K_\square = 1/8$ imply $R^* = 0.5$ and $\kappa_\Delta = 9/7$; then the bounds in Table 4.3 are immediate. \square

Remark 4.12 (Breakdowns in Algorithm 4.5). Since no 2×2 -pivots are allowed in the target, the factorization may break down if a d_i^+ becomes zero. This can be handled analogously to standard `dstqds`, cf. §2.4.5. \diamond

ALGORITHM 4.5 *Factorize blocked to non-blocked*

Given block factorization $T = LDL^* \in \mathbb{R}^{n \times n}$ in standard representation, compute data for non-blocked $L^+D^+(L^+)^* = T^+$ such that

$$L^+D^+(L^+)^* = LDL^* - \tau.$$

Notes:

- The offdiagonal elements e_i are reused to represent T^+ .
- Computing the block determinants $\Delta_i, i + 1 \in \Omega$ is not shown; these should have been cached once beforehand.

Input: Ω , shift τ , $\{D(i, i)\} = \{d_i \mid i \notin \Omega\} \cup \{c_i \mid i \in \Omega\}$, $\{e_1, \dots, e_{n-1}\}$

Output: $\{d_1^+, \dots, d_n^+\}$

Param: $R > 1$

```

1:   $s_1 := 0$ 
2:  for  $i = 1$  to  $n - 1$  do
3:     $d_i^+ := D(i, i) + (s_i - \tau)$ 
4:    if  $i + 1 \in \Omega$  then                                     // initiate breaking the block
5:       $s_{i+1} := -e_i^2/d_i^+$                                      // S4
6:    elseif  $i \notin \Omega$  then                                 // standard dstqds
7:       $s_{i+1} := e_i^2(s_i - \tau)/(d_i d_i^+)$                    // S1
8:    else                                                       // finish breaking the block
9:      if  $|s_i| \leq R|\tau|$  or  $\text{sign}(d_{i-1}) \neq \text{sign}(d_{i-1}^+)$  then
10:         $x := d_{i-1}(s_i - \tau) + e_{i-1}^2$                        // branch I
11:      else
12:         $x := -s_i(s_{i-1} - \tau) - d_{i-1}\tau$                    // branch II
13:      endif
14:       $s_{i+1} := e_i^2 x / (\Delta_{i-1} d_i^+)$                    // S6
15:    endif
16:  endfor
17:   $d_n^+ := D(n, n) + (s_n - \tau)$ 

```

LDL*		L ⁺ D ⁺ (L ⁺)*	$i \notin \Omega$	$i \in \Omega$
$d_i \rightsquigarrow \tilde{d}_i$	1	$d_i^+ \rightsquigarrow \tilde{d}_i^+$	2	5
$c_i \rightsquigarrow \tilde{c}_i$	4			
		$i \notin \Omega$ $i \in \Omega$		
		$e_i \rightsquigarrow \tilde{e}_i$	3	10
		$\tau \rightsquigarrow \tilde{\tau}_i$	0	12

Table 4.3: Error bounds to achieve mixed relative stability for Algorithm 4.5, for the concrete parameters $R = 3, K_{\square} = 1/8$, cf. Theorem 4.11 and Figure 4.2. Only first-order bounds are shown, i.e., an entry p stands for a bound $p\epsilon_{\diamond} + \mathcal{O}(\epsilon_{\diamond}^2)$.

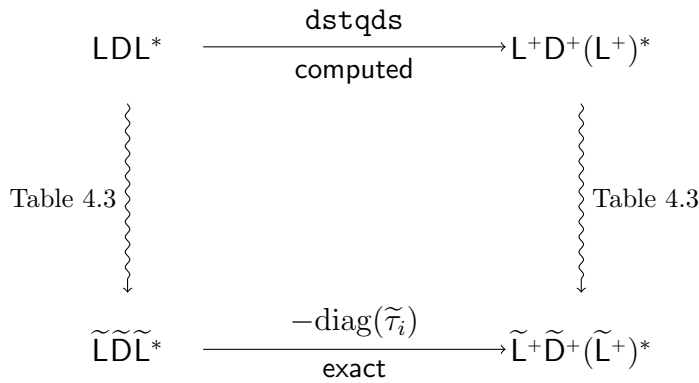


Figure 4.2: Mixed relative error analysis for Algorithm 4.5.

Remark 4.13 (Optimizing Algorithm 4.5). We formulated Algorithm 4.5 with the intention to maximize clarity, but in this form it is quite inefficient, due to the many conditionals involved. An alternative design could use a nested loop structure:


```

1:  while  $i < n$  do
2:    while  $i + 1 \notin \Omega$  do
3:      ... // do normal dstqds
4:    endwhile
5:    // break the block
6:    ...
7:     $i := i + 2$ 
8:  endwhile

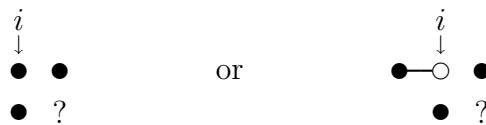
```

This would reduce the number of extra conditionals to one per block in the source, which is minimal, since we cannot avoid having to select between branches I and II for stable computation. \diamond

4.2.3 Creating blocks & handling overlap

In this section we will analyze how to *create* new blocks at a position where the source does not have one. We will start with discussing criteria to decide when this is sensible.

Assume the factorization process did just compute d_i^+ at an index i where no block starts in the source, that is, $i + 1 \notin \Omega$. This leaves two cases to consider, since D might still have a block ending at index i :



One could try to test for (4.16) directly, by tentatively computing the diagonal element $T^+(i+1, i+1) = c_{i+1}^+$. This would require to first determine s_{i+1} according to either case S5 or S9 in Table 4.1, and then check if

$$|d_i^+(d_{i+1} + (s_{i+1} - \tau))| < K_{\square} e_i^2 \quad (4.51)$$

holds. This approach has the advantage of being to-the-point. A 2×2 -pivot is chosen aggressively in every situation where it makes sense according to the basic condition (4.16). However, there are two drawbacks. If the test should indicate *not* to choose a 2×2 -pivot, we end up in one of the cases S1 or S6 instead. Then the computed s_{i+1} becomes invalid; its computation—including at least one multiplication and one division—as well as the two additions to get c_{i+1}^+ , would have been wasted. The second drawback is indeed more serious: we need to make some additional assumptions about created blocks for the following error analysis to succeed, and the direct test (4.51) does not provide them.

Due to these reasons, we use the following stronger criterion for determining if a block should be introduced where the source does not have one.

Block-Criterion II

Fix parameter $K_2 \leq K_{\square}^*/3$. Choose a 2×2 -pivot if

$$|d_i^+| \cdot \max \{|\tau|, |d_{i+1}|\} < K_2 e_i^2$$

and

$$\begin{cases} |d_i^+| < K_2 |d_i| & \text{if } i \notin \Omega, \\ |d_i^+ d_{i-1}| < K_2 |\Delta_{i-1}| & \text{if } i \in \Omega. \end{cases}$$

Should a created block satisfy this criterion in exact arithmetic, then the expressions for s_{i+1} in cases S5 or S8 from Table 4.1 reveal

$$c_{i+1}^+ = \begin{cases} d_{i+1} + e_i^2/d_i - \tau, & \text{if } i \notin \Omega, \\ d_{i+1} + e_i^2 d_{i-1}/\Delta_{i-1} - \tau, & \text{otherwise,} \end{cases} \implies |d_i^+ c_{i+1}^+| < K_{\square} e_i^2.$$

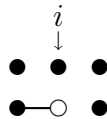
The adjusted choice of K_2 based on K_{\square}^* ensures that this will hold for the perturbed quantities as well, so indeed the computed block data will even have the stronger properties (4.19) and (4.20) again.

Deploying BC-II instead of (4.51) has the advantage that

$$|\Delta_i^+| > (1 - K_{\square})e_i^2 > \frac{3(1 - K_{\square})}{K_{\square}} |d_i^+ d_{i+1}| \tag{4.52}$$

will hold for any chosen block. For example, with $K_{\square} = .25$ this means the “hidden” pivot Δ_i^+/d_i^+ would have been at least nine times larger in magnitude than d_{i+1} (or infinite if $d_i^+ = 0$); so the choice to create a block was well-founded. This particular property will be crucial for the coming error analysis.

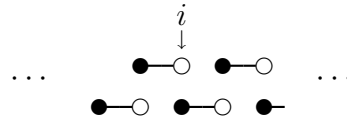
A newly created block in the target can *overlap* with blocks in the source if $i - 1 \in \Omega$ or $i + 1 \in \Omega$. There is only one situation where this does not happen:



One could call this the “clean” or “pristine” creation of a new block. It is symmetrical to breaking a block. Indeed, this can be realized based on the less restrictive test (4.51) in a way completely analogous to breaking a block in §4.2.2, including a necessary branch in the computation. Our implementation does indeed include this special treatment, but we have chosen not to present its error analysis here, since it does not convey anything new compared to breaking a block. Furthermore, as it turns out, if the more restrictive Block-Criterion II is employed, the computational branch is not necessary.

There is a fundamental problem involved with overlapping blocks. It arises when D and D^+ each have a sequence of consecutive blocks that are out-of-sync,

in the sense $i \in \Omega \Leftrightarrow i \notin \Omega^+$. With respect to Table 4.1, this means alternating between S8 and S9. We call this phenomenon an *overlap sequence*. The essential problem with it is that we cannot formulate the factorization as a recursive process like we could for standard `dstqds`, cf. Remark 2.22. As a consequence the perturbation bounds to attain mixed relative stability will grow with the length of the sequence, at least in general. To explain why this is so, consider the following excerpt of an overlap sequence, around $i \in \Omega$:



Any perturbation strategy has to fix \tilde{e}_i to control \tilde{s}_{i+1} such that (4.23) is fulfilled. As $i \in \Omega$, this makes $\varrho(e_i)$ depend on $\varrho(\Delta_{i-1})$. Now e_i contributes to Δ_i^+ , so $\varrho(\Delta_i^+)$ will depend on $\varrho(e_i)$ and therefore, $\varrho(e_{i+1})$ will have to depend on $\varrho(e_i)$, too, forming a cycle.

Because of this developing interdependency, we can deal with overlap sequences only by considering them en bloc, starting from the last index j with $j \notin \Omega \cup \Omega^+$ and up to the next k with $k + 1 \notin \Omega \cup \Omega^+$. Then each sequence can start and end by either creating a block or breaking one. This leaves four basic kinds of overlap sequences; they are depicted in Figure 4.3. The constraints shown on j and k stem from requiring that at least one new block from D^+ be contained. Note that type $C-C$ includes $k = j + 1$, the creation of a new block without any overlap, which was introduced already. We kept this case in as the error analysis to follow will cover it seamlessly.

The meaning of the indices j and k to denote the beginning and end of the overlap sequence will remain in effect during this section, thus freeing i to be used as running index.

The computation of the adjustments s_{j+1}, \dots, s_{k+1} for any of the four kinds can proceed as summarized in CS 4.6 on page 203. It uses the formulae from Table 4.1 for up to s_k and the alternative formulations from Table 4.2 for s_{k+1} .

We can summarize the effects of floating-point arithmetic by stating that, for $i = j, \dots, k$,

$$s_{i+1} = (1 + \varepsilon_i)(1 + \alpha_i) \cdot \begin{cases} e_i^2 / (\dots) & \text{with } \alpha_i \doteq \epsilon(1), & \text{if } i = j, \\ e_i^2 \cdot (\dots) & \text{with } \alpha_i \doteq \epsilon(2), & \text{if } j < i < k, \\ e_k^2 x / (\dots) & \text{with } \alpha_i \doteq \epsilon(4), & \text{if } i = k, \end{cases} \quad (4.53)$$

introducing the intermediate

$$x = \begin{cases} (s_k - \tau)d_{k-1}(1 + \beta_k) / (1 + \sigma_k) + e_{k-1}^2(1 + \varepsilon_{k-1}), & \text{if } k \in \Omega, \\ (s_k - \tau)d_{k-1}^+(1 + \beta_k) / (1 + \sigma_k) - e_{k-1}^2(1 + \varepsilon_{k-1}), & \text{if } k \in \Omega^+, \end{cases} \quad (4.54)$$

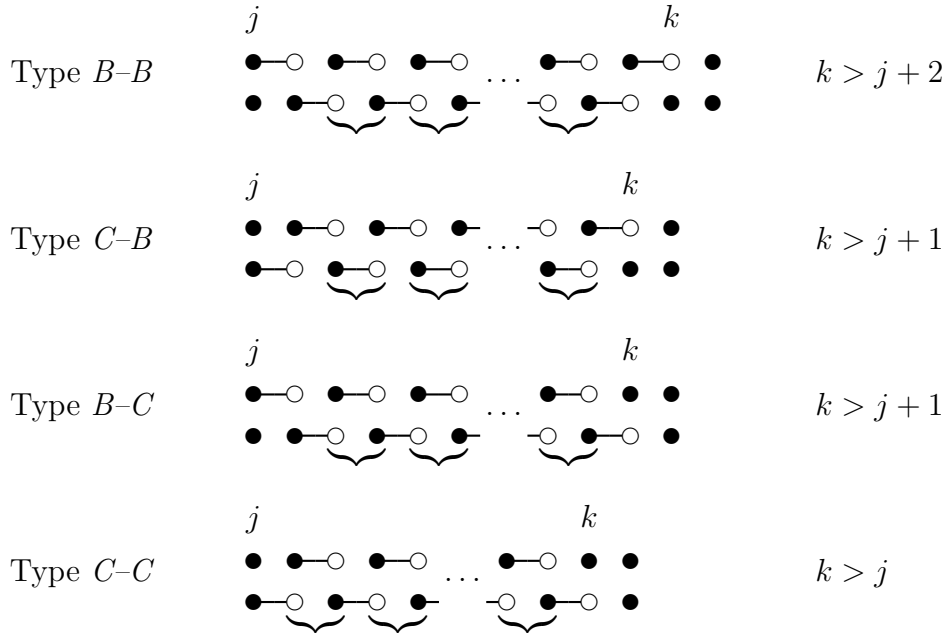


Figure 4.3: The four kinds of overlap sequences, classified as to whether they start/end by breaking a block (B) or creating a new one (C). The braces indicate the repeating block structure pattern.

where $|\beta_k| \leq \epsilon_\diamond$ and ε_i, σ_k are the ones from (4.25) and (4.29), respectively. Note that the rounding error from the outermost addition in computing x contributes to α_k and that we use the same x and β_k for both types of endings.

The error analysis for CS 4.6 will proceed as follows. We will first focus on the start and the repeating middle part up to s_k . That is the hardest part, as it involves dealing with perturbation bounds that depend on the length of the sequence (i.e., the distance to j). Once that is covered, we can deal with the two types of endings and wrap up.

As was already hinted at, the main challenge is that the required perturbations of e_i and Δ_i depend on e_{i-1} and Δ_{i-1} , respectively. With the intent of handling these interdependencies more fluently, let us define numbers p_i and q_i to be minimal such that

$$\varrho(e_i^2)/(1 + \varepsilon_i) \doteq 1 + \epsilon(p_i), \quad i = j, \dots, k, \tag{4.55}$$

as well as

$$\begin{cases} \varrho(\Delta_i) \doteq 1 + \epsilon(q_i), & \text{for } i + 1 \in \Omega, \\ \varrho(\Delta_i^\dagger) \doteq 1 + \epsilon(q_i), & \text{for } i + 1 \in \Omega^+. \end{cases} \quad i = j, \dots, k - 1, \tag{4.56}$$

are fulfilled.

CS 4.6: *Compute adjustments for an overlap sequence*

```

1:  if  $j + 1 \in \Omega$  then
2:      // begin by breaking a block
3:       $s_{j+1} := -e_j^2/d_j^+$ 
4:  else
5:      // begin by creating a block
6:       $s_{j+1} := e_j^2/d_j$ 
7:  endif
8:  // the middle part
9:  for  $i = j + 1$  to  $k - 1$  do
10:     if  $i \in \Omega$  then
11:          $s_{i+1} := e_i^2 d_{i-1} / \Delta_{i-1}$ 
12:     else
13:          $s_{i+1} := -e_i^2 d_{i-1}^+ / \Delta_{i-1}^+$ 
14:     endif
15: endfor
16: if  $k \in \Omega$  then
17:     // end with breaking a block
18:      $s_{k+1} := e_k^2 [(s_k - \tau) d_{k-1} + e_{k-1}^2] / (\Delta_{k-1} d_k^+)$ 
19: else
20:     // end with creating a block
21:      $s_{k+1} := e_k^2 [(s_k - \tau) d_{k-1}^+ - e_{k-1}^2] / (d_k \Delta_{k-1}^+)$ 
22: endif

```

For the diagonal data of D and D^+ we deploy the default perturbations (4.30). Furthermore, we will have to use (4.27) so often that we will not always state the reference.

Two Beginnings. The “right” perturbation $e_j \rightsquigarrow \tilde{e}_j$ to get s_{j+1} to be exact for the perturbed data is captured by

$$\varrho(e_j^2) := (1 + \varepsilon_j)(1 + \alpha_j) \cdot \begin{cases} \varrho(d_j^+), & \text{if } j + 1 \in \Omega, \\ \varrho(d_j), & \text{if } j + 1 \in \Omega^+, \end{cases} \implies p_j \leq 3. \quad (4.57)$$

The default perturbations (4.30) for this case are

$$\begin{aligned} \tilde{d}_j &= d_j(1 + \sigma_j), & \varrho(D) &= (1 + \sigma_{j+1}), \\ \tilde{d}_j^+ &= d_j^+(1 + \delta_j^+)(1 + \sigma_j), & \varrho(D^+) &= (1 + \delta_{j+1}^+)(1 + \sigma_{j+1}), \end{aligned}$$

where either $D = c_{j+1}$, $D^+ = d_{j+1}^+$ if $j + 1 \in \Omega$, or $D = d_{j+1}$, $D^+ = c_{j+1}^+$ if $j + 1 \in \Omega^+$. Hence, depending on the situation we can invoke the upper formula from (4.27) with $F := (1 + \sigma_{j+1})$, or the lower one with $F^+ := (1 + \sigma_{j+1})$ to reveal the perturbation's effect on the first block determinant of the sequence to be

$$\begin{cases} \varrho(\Delta_j) \doteq 1 + \epsilon(2 + 2\kappa_\Delta), & \text{if } j + 1 \in \Omega, \\ \varrho(\Delta_j^+) \doteq 1 + \epsilon(2 + 4\kappa_\Delta), & \text{if } j + 1 \in \Omega^+, \end{cases} \implies q_j \leq 2 + 4\kappa_\Delta. \quad (4.58)$$

The Middle Part. Here, that is, for $i = j + 1 : k - 1$, the perturbation $e_i \rightsquigarrow \tilde{e}_i$ to get $s_{i+1} = \tilde{s}_{i+1}$ is

$$\begin{aligned} \varrho(e_i^2) &:= (1 + \varepsilon_i)(1 + \alpha_i) \cdot \begin{cases} \varrho(\Delta_{i-1})/\varrho(d_{i-1}), & i \in \Omega, \\ \varrho(\Delta_{i-1}^+)/\varrho(d_{i-1}^+), & i \in \Omega^+, \end{cases} \\ &\implies p_i \leq q_{i-1} + 4. \end{aligned} \quad (4.59)$$

Concerning the block determinants, it is not hard to realize that the maxima (4.27) are then attained with the e_i^2 -terms. Hence, if those are perturbed as just specified, we will have

$$q_i \leq 1 + \kappa_\Delta p_i. \quad (4.60)$$

The perturbations as specified so far depend on the meshed recurrence between the p_i 's and q_i 's. To get a grip on it we employ the following elementary tool.

Fact & Definition. Define for $m \in \mathbb{N}_0$ and $z \in \mathbb{R}$

$$\phi_m(z) := \sum_{i=0}^m z^i = \begin{cases} m + 1, & \text{if } z = 1, \\ (z^{m+1} - 1)/(z - 1), & \text{otherwise.} \end{cases}$$

Let for given scalars $b, r \in \mathbb{R}$ the sequence $(a_k)_{k \geq 0}$ of reals satisfy

$$a_0 > 0, \quad a_k \leq b + r a_{k-1} \text{ for } k > 0.$$

Then, for all $k \in \mathbb{N}_0$,

$$a_k \leq \max\{b, a_0\} \phi_k(r).$$

Combine (4.59) with (4.60) to get $q_i \leq (1 + 4\kappa_\Delta) + \kappa_\Delta q_{i-1}$ for $i > j$. With the starting value from (4.58) we can then model the recurrence ($a_0 = q_j = 2 + 4\kappa_\Delta, b = 1 + 4\kappa_\Delta, r = \kappa_\Delta$) to obtain

$$q_i \leq (2 + 4\kappa_\Delta) \cdot \phi_{i-j}(\kappa_\Delta) \quad \text{and} \quad p_i \leq q_{i-1} + 4 \quad \text{for } i > j. \quad (4.61)$$

The dependence on the length of the overlap sequence is far too strong to make this result of much practical use. For example, with the moderate bound

$\kappa_\Delta = 3/2$ (which we get with $K_\square = 1/5$) and a sequence of length $k - j = 6$, the perturbation to e_{k-1}^2 alone would already exceed $100\epsilon_\diamond$.

But there is light at the horizon, in the form of choice. After all, the bound above can be monitored during a computation. Should it grow too large, we can simply choose not to take a 2×2 -pivot, thus capping the sequence. The only situation where we absolutely *must* take a 2×2 -pivot is if d_i^+ becomes zero. But that is actually a favorable situation, because it causes s_{i+2} to be zero as well (cf. line 13 in CS 4.6 for $i + 1$). This effectively lifts any restriction on \tilde{e}_{i+1} and therefore cuts the dependency-chain. So far our analysis does not exploit this. Indeed, we can improve the analysis not only for $d_i^+ = 0 \Rightarrow s_{i+2} = 0$, but more generally for any situation where an $|s_i|$ becomes “small”. The idea here was used before, namely that if, for any i , the adjustment s_i is not much larger in magnitude than the shift τ , then the latter can be perturbed to provide freedom in the choice for \tilde{e}_{i-1} .

We will limit this optimization to the middle part. So, assume we have for an i with $j < i < k$ that $|s_{i+1}| \leq R|\tau|$ holds for a parameter $R > 0$. Then perturb $e_i \rightsquigarrow \tilde{e}_i$ instead of (4.59) simply as

$$\tilde{e}_i^2 := e_i^2(1 + \epsilon_i) \tag{4.62}$$

and employ (4.31) to define $\tilde{\tau}_{i+1}$ such that $s_{i+1} - \tau = \tilde{s}_{i+1} - \tilde{\tau}_{i+1}$ is upheld. Look at the middle line of (4.53) to see that the above choice for \tilde{e}_i implies

$$\varrho(s_{i+1}) = (1 + \alpha_i)^{-1} \cdot \begin{cases} \varrho(d_{i-1})/\varrho(\Delta_{i-1}), & i \in \Omega, \\ \varrho(d_{i-1}^+)/\varrho(\Delta_{i-1}^+), & i \in \Omega^+. \end{cases}$$

Hence, the necessary perturbation $\tilde{\tau}_{i+1} = \tau\varrho(\tau_i)$ to the shift from (4.31) can be bounded as

$$|\varrho(\tau_{i+1}) - 1| \leq R|\varrho(s_{i+1}) - 1| \leq R\epsilon^{[2]}(4 + q_{i-1}).$$

The benefit of doing so is that it effectively resets the recurrence, since (4.62) gives $p_i = 0$ and $q_i \leq 1 + 5\kappa_\Delta$. The parameter R is yet unspecified. Indeed, we can choose it freely; its sole purpose is to control $\varrho(\tau_i)$.

Let us summarize where we stand.

Lemma 4.14. *Let in CS 4.1 the adjustments s_{j+1}, \dots, s_k for an overlap sequence be computed as in CS 4.6. Fix a parameter $R > 0$ and define for $i = j, \dots, k - 1$*

$$h(i) := i - \max \{m \mid m = j \text{ or } j < m \leq i \text{ and } |s_{m+1}| \leq R|\tau|\}.$$

Then there is a perturbation such that (4.23) is fulfilled for $i = j, \dots, k - 1$. With suitable numbers

$$q_i \leq (2 + 4\kappa_\Delta)\phi_{h(i)}(\kappa_\Delta), \quad j \leq i < k,$$

the individual perturbations can be bounded as follows:

$$\begin{aligned} D(i, i) \rightsquigarrow \tilde{D}(i, i) &\doteq \epsilon(1), & D^+(i, i) \rightsquigarrow \tilde{D}^+(i, i) &\doteq \epsilon(2), \\ e_j \rightsquigarrow \tilde{e}_j &\doteq \epsilon^{[4]}(2), & e_i \rightsquigarrow \tilde{e}_i &\doteq \epsilon^{[4]}(\frac{5}{2} + \frac{1}{2}q_{i-1}), \quad j < i < k, \\ \tilde{\tau}_j = \tau, & \tilde{\tau}_{j+1} = \tau, & \tau \rightsquigarrow \tilde{\tau}_i &\doteq R\epsilon^{[2]}(4 + q_{i-2}), \quad j + 1 < i \leq k. \end{aligned}$$

The implied perturbations for the secondary quantities will obey

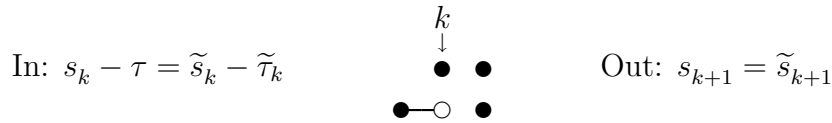
$$\begin{aligned} \left. \begin{aligned} i + 1 \in \Omega : & \quad |\varrho(\Delta_i) - 1| \\ i + 1 \in \Omega^+ : & \quad |\varrho(\Delta_i^+) - 1| \end{aligned} \right\} \leq \epsilon(q_i), \quad j \leq i < k, \\ \tilde{s}_{j+1} = s_{j+1}, & \quad |\varrho(s_{i+1}) - 1| \leq \epsilon^{[2]}(4 + q_{i-1}), \quad j < i < k. \end{aligned}$$

□

Recall that the main purpose of the preceding analysis was not to provide sharp worst-case perturbation bounds, but also to give us the means to control the computation of an overlap sequence. Whenever we absolutely need to choose a 2×2-pivot due to a tiny d_i^+ , we can do so and because s_{i+2} will then be tiny, too, the bounds are controlled. Otherwise, we can keep track of the current length of the sequence (better: the current $h(i)$) and if this exceeds some threshold just choose a single pivot, thus ending the sequence and capping the bounds.

The challenging part is done, what remains is just a little tedious. We need to deal with the two possible ways for the sequence to end. For the following we have to rely on the stronger Block-Criterion II, and in particular (4.52). Furthermore we require $K_\square < 1/3$.

An End by Creation: $k \in \Omega^+$. We assume $s_k - \tau$ is exact, i.e., $s_k - \tau = \tilde{s}_k - \tilde{\tau}_k$. Lemma 4.14 does deliver this. What remains to be done is just to define $e_k \rightsquigarrow \tilde{e}_k$ such that s_{k+1} becomes exact, $s_{k+1} = \tilde{s}_{k+1}$. The following picture summarizes the situation:



Note that we may have $k = j + 1$.

One could fear the task to be impossible should the computation leading to the intermediate x in line 21 of CS 4.6 involve cancellation. Indeed, this would cause problems in general, but we will show that this computation cannot be too badly conditioned if the choice for a 2×2-pivot at $k - 1$ was based on Block-Criterion II.

Depending on the situation at hand, s_k is defined in one of two ways:

$$s_k = \begin{cases} e_{k-1}^2/d_{k-1}, & \text{if } k - 1 \notin \Omega, \\ e_{k-1}^2 d_{k-2}/\Delta_{k-2} & \text{if } k - 1 \in \Omega. \end{cases}$$

Thus fulfillment of the second condition from Block-Criterion II implies $|s_k| < (K_\square^*/3)e_{k-1}^2/|d_{k-1}^+|$, which yields

$$|s_k - \tau||d_{k-1}^+| \leq 2 \max\{|s_k|, |\tau|\}|d_{k-1}^+| < \frac{2}{3}K_\square^*e_{k-1}^2.$$

Because of the safeguard $K_\square^* < K_\square$ to anticipate rounding errors, we can safely conclude that the computation of x in line 21, is governed by

$$\kappa_-(\text{fl}((s_k - \tau)d_{k-1}^+), \text{fl}(e_{k-1}^2)) < \frac{3 + 2K_\square}{3 - 2K_\square} =: \kappa[x]. \quad (4.63)$$

Hence we can control the rounding errors expressed in the lower line of (4.54) as

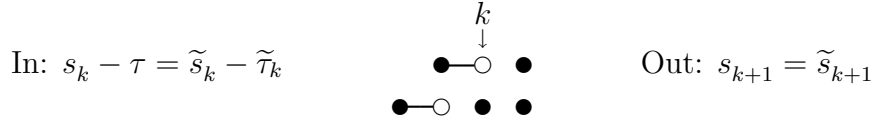
$$\begin{aligned} \tilde{x} &= (s_k - \tau)\tilde{d}_{k-1}^+ - \tilde{e}_{k-1}^2 = x\varrho(x), \\ \text{where } |\varrho(x) - 1| &\leq \kappa[x]\epsilon^{[2]}(\max\{4, p_{k-1}\}). \end{aligned} \quad (4.64)$$

Finally,

$$\varrho(e_k^2) := (1 + \varepsilon_k)\varrho(\Delta_{k-1}^+)\varrho(d_k)(1 + \alpha_k)/\varrho(x) \quad (4.65)$$

makes s_{k+1} exact, as desired.

An End by Breaking: $k \in \Omega$. The situation at hand is



This is very similar to breaking a block from the §4.2.2. Indeed, the way s_{k+1} is computed in line 18 of CS 4.6 corresponds to branch I from CS 4.3. The difference is that, here, we have overlap at the left end: $k - 1 \in \Omega^+$. This will prove to be beneficial, because here we can assume that the created block covering indices $\{k - 2, k - 1\}$ satisfies Block-Criterion II, which will allow us to show that the single computational branch does work for all configurations. Note that we must have $k > j + 1$, so that s_k is defined as

$$s_k = e_{k-1}^2 d_{k-2}^+ / \Delta_{k-2}^+.$$

For the analysis, we consider two cases, depending on the parameter

$$\bar{R} := (1 - 2K_\square)^{-1},$$

which is larger than one only if $K_\square < 1/2$ holds, and we assume it does.

CASE: $|s_k| \leq \bar{R}|\tau|$. We employ the perturbations from Lemma 4.14 but throw away those for e_{k-1} and τ_k . We will redefine them manually to still retain $s_k - \tau = \tilde{s}_k - \tilde{\tau}_k$. With the intent of controlling x , we perturb e_{k-1} instead as

$$\varrho(e_{k-1}^2) := (1 + \varepsilon_{k-1})\varrho(d_{k-1})(1 + \sigma_k)/(1 + \beta_k),$$

because according to the upper line in (4.54) this results in

$$\begin{aligned}\tilde{x} &= (s_k - \tau)\tilde{d}_{k-1} + \tilde{e}_{k-1}^2 = x\varrho(d_{k-1})(1 + \sigma_k)/(1 + \beta_k) \\ &\implies |\varrho(x) - 1| \leq \epsilon^{[2]}(3).\end{aligned}\quad (4.66)$$

The effect of the above choice for \tilde{e}_{k-1} on s_k is

$$\tilde{s}_k = s_k\varrho(s_k) = s_k\varrho(d_{k-2}^+)\varrho(d_{k-1})[\varrho(\Delta_{k-2}^+)(1 + \alpha_{k-1})(1 + \beta_k)]^{-1}(1 + \sigma_k).$$

This looks a bit unwieldy, but nevertheless we can invoke (4.31) to move all the relative perturbations onto the shift. This defines $\tau \rightsquigarrow \tilde{\tau}_k$ such that

$$|\varrho(\tau_k) - 1| \leq \bar{R} \cdot |\varrho(s_k) - 1| \leq \bar{R}\epsilon^{[2]}(7 + q_{k-2}),$$

and gives the desired $s_k - \tau = \tilde{s}_k - \tilde{\tau}_k$. \diamond

CASE: $|s_k| > \bar{R}|\tau|$. If Block-Criterion II is fulfilled for the created block ending at $k - 1 \in \Omega^+$, then (4.52) and our definition of \bar{R} gives

$$|s_k - \tau||d_{k-1}| < 2(1 - K_\square)|s_k||d_{k-1}| < \frac{2}{3}K_\square e_{k-1}^2.$$

In fact, the safeguard $K_\square^* < K_\square$ allows us to relax this relation by some ulp on each side, so that we may assume the computation of x to be controlled by

$$\kappa_+(\text{fl}((s_k - \tau)d_{k-1}), \text{fl}(e_{k-1}^2)) < \frac{3 + 2K_\square}{3 - 2K_\square} = \kappa[x],$$

with the same $\kappa[x]$ as in (4.63). In fact, attaining that was the motivation behind our definition of \bar{R} . Thus,

$$\begin{aligned}\tilde{x} &= (s_k - \tau)\tilde{d}_{k-1} + \tilde{e}_{k-1}^2 = x\varrho(x), \\ \text{where } |\varrho(x) - 1| &\leq \kappa[x]\epsilon^{[2]}(\max\{3, p_{k-1}\}).\end{aligned}\quad (4.67)$$

\diamond

Finally, for both cases the analysis achieved that the computed x is close to the exact value for computed data, meaning we have a bound on $\varrho(x)$. Thus we can perturb $e_k \rightsquigarrow \tilde{e}_k$ according to

$$\tilde{e}_k^2 := e_k^2(1 + \varepsilon_k)\varrho(\Delta_{k-1})\varrho(d_k^+)(1 + \alpha_k)/\varrho(x) \quad (4.68)$$

to obtain $s_{k+1} = \tilde{s}_{k+2}$.

Lemma 4.15. *Let in CS 4.1 the adjustments s_{j+1}, \dots, s_{k+1} for an overlap sequence be computed as in CS 4.6. Furthermore, let all 2×2 -pivots in D^+ satisfy Block-Criterion II with a parameter $K_\square < 1/3$.*

Then there is a perturbation such that (4.23) holds for $j \leq i \leq k$ and the computed s_{k+1} is exact. The componentwise bounds from Lemma 4.14 do apply, with the following adjustments and extensions:

$$\begin{aligned} D(k, k) \rightsquigarrow \tilde{D}(k, k) &\doteq \epsilon(1), & D^+(k, k) \rightsquigarrow \tilde{D}^+(k, k) &\doteq \epsilon(2). \\ e_k \rightsquigarrow \tilde{e}_k &\doteq \epsilon^{[6]}(m), & m &\leq \begin{cases} 6 + \frac{1}{2}q_{k-1}, & k = j + 1, \\ 7 + \frac{3}{2}q_{k-1}, & k > j + 1, \end{cases} \\ \tilde{\tau}_k = \tau &\text{ if } k = j + 1, & \tau \rightsquigarrow \tilde{\tau}_k &\doteq \epsilon^{[2]}(21 + 3q_{k-2}), \text{ if } k > j + 1, \end{aligned}$$

and

$$|\varrho(s_k) - 1| \leq \epsilon^{[2]}(6 + q_{k-1}).$$

Proof. The subtle point is that we might have redefined \tilde{e}_{k-1} if ending with breaking a block. But it is easy to verify that the bounds to $\varrho(e_{k-1})$ and $\varrho(\Delta_{k-1})$ from Lemma 4.14 do still apply, just $\varrho(s_k)$ can increase.

For the final $e_k \rightsquigarrow \tilde{e}_k$, note that regardless of the type of ending, the perturbation’s effect on the intermediate x as stated in (4.64), (4.66) and (4.67) allow the uniform bound $|\varrho(x) - 1| \leq \kappa[x] \epsilon^{[2]}(\max\{4, p_{k-1}\})$ with $\kappa[x] = (3 + 2K_\square)/(3 - 2K_\square)$. The prerequisite $K_\square < 1/3$ gives $\kappa[x] < 2$ and $\bar{R} < 3$. The latter determines the bound on $\varrho(\tau_k)$, and (4.65) and (4.68) lead to the stated bound for $\varrho(e_k)$; recall that the analysis leading to Lemma 4.14 revealed $p_j \leq 3$ and $p_i \leq q_{i-1} + 4, i > j$. \square

4.2.4 The complete algorithm

The work on the previous pages culminates in our general algorithm for stationary block factorizations, with support for changing the block structure, shown in Algorithm 4.7 on pages 210-211.

The basic structure remains identical to our template CS 4.1, except that we have omitted statements to compute the block determinants Δ_i and Δ_i^+ . Essentially it is the straightforward combination of standard `dstqds` (for an e -representation, Alg. 2.7) and the computational sequences that we have considered for keeping (CS 4.2), breaking (CS 4.4 with parameter $R = R_{\text{brk}}$) or creating (CS 4.6) blocks. However, this fact is obscured somewhat due to the integration of a control mechanism for overlap sequences. That is controlled by a parameter R_{osq} which is the R from Lemma 4.14.

The outermost loop is composed of two parts. The “first” half, lines 3–25, handles the factorization as long as the block structure is not changed. With respect to Table 4.1 this encompasses cases S1–S5. There is nothing new here.

The “second” half, lines 26–56, is essentially one inner loop to handle any block structure changes from source to target. As such it concerns cases S6–S9

ALGORITHM 4.7 *blocked dstqds*

Compute $L^+D^+(L^+)^* = LDL^* - \tau$ for block factorizations.

Input: Ω , shift τ , $\{d_i \mid i \notin \Omega\}$, $\{c_i \mid i \in \Omega\}$, $\{e_1, \dots, e_{n-1}\}$

Output: Ω^+ , $\{d_i^+ \mid i \notin \Omega^+\}$, $\{c_i^+ \mid i \in \Omega^+\}$

Params: $R_{\text{brk}} > 1$, $R_{\text{osq}} > 0$, $k_{\text{max}} \geq 0$, $K_{\square} < 1/3$ as in (4.16),
 $K_1 \leq K_{\square}^*$, $K_2 \leq K_{\square}^*/3$ for BC-I (p. 186) and BC-II (p. 200)

```

1:   $\Omega^+ := \emptyset$ ,  $i := 1$ ,  $s_1 := 0$ 
2:  while  $i < n$  do
3:     $d_i^+ := d_i + (s_i - \tau)$ 
4:    if  $i + 1 \in \Omega$  then
5:      if BC-I is fulfilled then                                // Keep the block
6:         $\Omega^+ := \Omega^+ \cup \{i + 1\}$ 
7:         $s_{i+1} := 0$                                            // S2
8:        if  $i < n - 1$  then
9:           $c_{i+1}^+ := c_{i+1} - \tau$ 
10:          $x := e_i^2(s_i - \tau) - d_i d_i^+ \tau$ 
11:          $s_{i+2} := e_{i+1}^2 x / (\Delta_i \Delta_i^+)$                 // S3
12:          $i := i + 1$ 
13:       endif
14:     else                                                       // initiate breaking the block
15:        $s_{i+1} := -e_i^2 / d_i^+$                                    // S4
16:     endif
17:     else
18:       if BC-II is fulfilled then                               // initiate creating a new block
19:          $\Omega^+ := \Omega^+ \cup \{i + 1\}$ 
20:          $s_{i+1} := e_i^2 / d_i$                                    // S5
21:       else                                                       // standard dstqds
22:          $s_{i+1} := e_i^2(s_i - \tau) / (d_i d_i^+)$               // S1
23:       endif
24:     endif
25:      $i := i + 1$ 
26:      $k := 0$  // counts number of created blocks in an overlap-sequence
27:     ...

```

ALGORITHM 4.7 *blocked dstqds (continued)*

```

27:   while  $i < n$  and  $(i \in \Omega) \not\Leftarrow (i \notin \Omega^+)$  do
28:        $D^+(i) := D(i) + (s_i - \tau)$ 
29:       if  $|s_i| \leq R_{\text{osq}}|\tau|$  then
30:            $k := 0$ 
31:       endif
32:       if  $i \in \Omega^+$  then
33:           if  $i + 1 \in \Omega$  then // continue sequence
34:                $s_{i+1} := -e_i^2 d_{i-1}^+ / \Delta_{i-1}^+$  // S9
35:           else // end by create
36:                $s_{i+1} := e_i^2 (d_{i-1}^+(s_i - \tau) - e_{i-1}^2) / (\Delta_{i-1}^+ d_i)$  // S7
37:           endif
38:       else
39:           if BC-II is fulfilled and
40:                $(k < k_{\text{max}} \text{ or } i = n - 1 \text{ or } |d_i^+|e_{i+1}^2 \leq (1 - K_{\square}^*)R_{\text{osq}}|\tau|e_i^2)$ 
41:           then // create next block in the sequence
42:                $\Omega^+ := \Omega^+ \cup \{i + 1\}$ 
43:                $k := k + 1$ 
44:                $s_{i+1} := e_i^2 d_{i-1} / \Delta_{i-1}$  // S8
45:           else // end by break or clean break
46:               if  $i - 1 \in \Omega^+$  or  $|s_i| \leq R_{\text{brk}}|\tau|$  or  $\text{sign}(d_{i-1}) \neq \text{sign}(d_{i-1}^+)$ 
47:               then
48:                    $x := d_{i-1}(s_i - \tau) + e_{i-1}^2$ 
49:               else
50:                    $x := -s_i(s_{i-1} - \tau) - d_{i-1}\tau$ 
51:               endif
52:                $s_{i+1} := e_i^2 x / (\Delta_{i-1} d_i^+)$  // S6
53:           endif
54:       endif
55:        $i := i + 1$ 
56:   endwhile
57: endwhile
58:  $D^+(n) := D(n) + (s_n - \tau)$ 

```

from Table 4.1. This loop will only be entered if either one of the statements in line 15 or 20 were executed to initiate a structure change, i.e., breaking or creating a block, respectively. The former one of those, together with lines 46–52 incorporates breaking a block from CS 4.4 as well as end-by-break for an overlap sequence (line 18 from CS 4.6), due to the extra test $i - 1 \in \Omega^+$ in line 45.

The noteworthy new ingredient is the counter k , which is increased by one for each created block in an overlap sequence. It is reset to zero in line 30 whenever the current adjustment s_i is not much larger than the shift τ , thus integrating the optimized test from Lemma 4.14.

A new block may only be created if the complex test in lines 39–40 is passed. Besides checking for BC–II, creating a block in the target is only allowed if we can control the error bounds for the overlap sequence. To this end, one of three conditions has to be met:

- (1) $k < k_{\max}$: The length of the sequence is still deemed acceptable.
- (2) $i = n - 1$: Our error analysis did not dwell on this fact, but for a block that is created at the end it is easy to give very benign perturbation bounds to attain mixed relative stability, since no s_{i+2} has to be computed.
- (3) Because we require BC–II, the stronger property (4.52) will hold. Then fulfillment of the test $|d_i^+|e_{i+1}^2 \leq (1 - K_{\square}^*)R_{\text{osq}}|\tau|e_i^2$ implies $s_{i+2} \leq R_{\text{osq}}|\tau|$. Thus the error bound optimization for the middle part of an overlap sequence from Lemma 4.14 applies, and we can safely take the 2×2 -pivot. The test in line 29 will subsequently cause the counter to be reset.

Note that condition (3) permits the choice of a 2×2 -pivot whenever a tiny d_i^+ is encountered; in particular the condition is always fulfilled if $d_i^+ = 0$. Hence, the factorization cannot break down, even for $k_{\max} = 0$.

The following result summarizes the componentwise mixed relative error analysis for Algorithm 4.7.

Theorem 4.16 (Error Analysis for blocked `dstqds`)

Let Algorithm 4.7 be executed without underflow or overflow in an environment that satisfies Axiom FP.

Then the diagram in Figure 4.4 commutes, that is, there are perturbations to the inputs and outputs such that

$$\mathbf{LDL}^* - \text{diag}(\tilde{\tau}_i) = \mathbf{L}^+\mathbf{D}^+(\mathbf{L}^+)^*$$

holds exactly. The perturbations can be bounded depending on the Parameters according to Lemma 4.6, Corollary 4.10 and Lemma 4.15; for one set of parameters the resulting bound are given in Table 4.4.

Proof. Guideline to obtain the bounds in Table 4.4: $K_{\square}^* = 1/8$ implies $\kappa_{\Delta} = 9/7$, $R_{\text{brk}} = 3$ gives $R^* = 1/2$ for Corollary 4.10 and $R_{\text{osq}} = 1$ is the R in Lemma 4.15. Finally, $k_{\text{max}} = 1$ means overlap sequences j, \dots, k are limited to $k - j \leq 3$, hence we get $q_{k-1} \leq 24$ and $q_{k-2} \leq 10$ for Lemma 4.15. \square

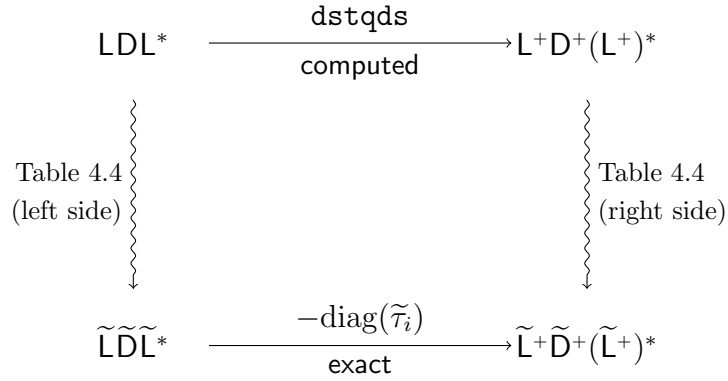


Figure 4.4: Mixed relative error analysis for blocked `dstqds` (Alg. 4.7).

LDL*	$i \in \Omega^+ \quad i \notin \Omega^+$		L ⁺ D ⁺ (L ⁺) [*]	$i \in \Omega \quad i \notin \Omega$	
$d_i \rightsquigarrow \tilde{d}_i$	1	1	$d_i^+ \rightsquigarrow \tilde{d}_i^+$	5	2
$c_i \rightsquigarrow \tilde{c}_i$	1	4	$c_i^+ \rightsquigarrow \tilde{c}_i^+$	2	2

	$i \notin \Omega$	$i \in \Omega$	$i \notin \Omega$	$i \in \Omega$
	$i \notin \Omega^+$	$i \notin \Omega^+$	$i \in \Omega^+$	$i \in \Omega^+$
$e_i \rightsquigarrow \tilde{e}_i$	3	43	43	8
$\tau \rightsquigarrow \tilde{\tau}_i$	0	51	51	0

Table 4.4: Error bounds to achieve mixed relative stability for Algorithm 4.7, for the concrete parameters $R_{\text{brk}} = 3, R_{\text{osq}} = 1, K_{\square}^* = 1/8, k_{\text{max}} = 1$, cf. Theorem 4.16 and Figure 4.4. Only first-order bounds are shown, i.e., an entry p stands for a bound $p\epsilon_{\diamond} + \mathcal{O}(\epsilon_{\diamond}^2)$.

Recall that our main objective was not to provide sharp error bounds, but to devise an algorithm for stationary block factorizations for which we can give componentwise relative error bounds in the first place.

However, there is no denying the fact that the bounds are quite large compared to standard `dstqds`. This is mainly due to the problems with overlap sequences

(for $k_{\max} = 0$ we could bound $|\rho(e_i) - 1|$ by $10\epsilon_\diamond$). One could fear that such large componentwise errors would overshadow the benefits of using 2×2 -pivots regarding control of local element growth. But keep in mind that the bounds are of a worst-case nature only. The experiments in the next section will show that the accuracy of the delivered results is far better than what these bounds would suggest.

Nevertheless, one future goal of research should be to simplify the treatment and analysis of overlap.

4.3 Numerical Results

Block factorizations can be used as components for the tridiagonal MR^3 as well as the adapted solution strategies for BSVD. This section will evaluate the potential gain from doing so.

We took our three implementations **XMR** (standard MR^3), **XMR-TGK** (MR^3 on T_{GK}) and **XMR-CPL** (MCR^3) and introduced plain LDL^* block factorizations as follows:

- For **XMR** and **XMR-TGK** they completely replace standard twisted factorizations as representations at the nodes.
- For **XMR-CPL** block factorizations take over the role of representations in the central layer, but for the outer layers, e -representation of standard twisted factorizations are used. The reason is that the coupling relations (3.24) and (3.25) cannot easily be extended to yield block factorizations for the outer layers.

Sturm counts for bisection are done using the customized blocked to non-blocked factorization from Algorithm 4.5 with parameter $R = 8$. To construct the (non-blocked) factorizations that are needed for computing accurate eigenvectors, the same instantiation of Algorithm 4.5 was employed, together with a progressive analogon (which we have not presented).

For the execution of shifts, that is, to construct BFs for the representations at child nodes, Algorithm 4.7 is deployed with parameters

$$R_{\text{brk}} = 5, R_{\text{osq}} = n/4, k_{\max} = 4, \\ K_{\square} = 1/8, K_1 = K_{\square}^*, K_2 = K_{\square}^*/3.01.$$

Together this yields three new routines which we will denote as **BMR**, **BMR-TGK** and **BMR-CPL**, respectively. Those will now be compared to the originals on the already introduced testsets **Pract** and **Synth**.

Block Factorizations for standard MR³

The results are shown in Tables 4.5 and 4.6 on the next page. We see that for the set **Pract**, orthogonality is somewhat better with blocks, but residual norms go minimally up compared to **XMR**. The residual norms for **Synth** are significantly better with blocks, however, but for that set the average orthogonality level increases minimally.

An interesting facet is that both methods have the same worst orthogonality levels of $608n\epsilon_\diamond$ over **Synth**. Those stem in fact from the same test case, which has two singleton eigenvalues at root level with relative separation ≈ 0.0016 . Hence the quality of the result is within expected bounds. Since both methods start with the same root representation, and no blocks are used during the computation of eigenvectors, the produced vectors are exactly identical.

Block Factorizations for MR³ on the Golub-Kahan matrix

Tables 4.7 and 4.8 on on page 217 show clearly that block factorizations are the right way to handle translates of Golub-Kahan matrices. This fact was already hinted at in Examples 4.2 and 4.3.

The worst delivered orthogonality and residual norms are significantly better than with **XMR-TGK**. The averages improve as well, if only slightly. Interestingly, the median orthogonality goes up a little. Combined with the decreasing average this means there are fewer cases with orthogonality of about $n\epsilon_\diamond$ or smaller. We interpret this as effect of the larger componentwise error bounds for BFs.

Block Factorizations for the coupled approach

The results of using blocks in the coupled approach are compiled in Tables 4.9 and 4.10 on on page 218. We observe that orthogonality actually worsens a bit with blocks, but is still within acceptable limits.

The prominent conclusion, however, is that the bad cases for **XMR-CPL** in the testset **Synth** are evidently smoothed out using blocks. As such the results are actually rather satisfactory, but still noticeably worse than what **BMR-TGK** can deliver.

Nevertheless we must mention that there are still cases in **Synth** where **BMR-CPL** refuses to compute some singular triplets, because at some point no child representation can be found that withstands the coupling checks (lack of **SHIFTREL** again). The good news is that the number of cases in **Synth** where this happens goes down, from 24 for **XMR-CPL** to 4 with **BMR-CPL**.

Conclusion

There is really no question that block factorizations are the way to go to achieve ultimate accuracy and reliability. Note that the previous tests did compare **LDL***

Pract 75 cases		ORTH	Synth 19240 cases	
XMR	BMR		XMR	BMR
5.28	3.10	AVG	3.09	3.20
1.78	1.31	MED	0.91	0.96
91	41	MAX	608	608
89.33 %	94.67 %	0 ... 10	94.39 %	93.81 %
10.67 %	5.33 %	10 ... 100	5.43 %	5.99 %
		100 ... 200	0.12 %	0.16 %
		200 ... 500	0.05 %	0.04 %
		500 ... 10 ³	0.01 %	0.01 %

Table 4.5: Orthogonality levels $|Q^*Q - I|$ of XMR compared to BMR, as multiples of $n\epsilon_\diamond$. The data for XMR is identical to Table 2.6.

Pract 75 cases		RESID	Synth 19240 cases	
XMR	BMR		XMR	BMR
0.21	0.22	AVG	0.37	0.37
0.05	0.05	MED	0.09	0.09
3.10	3.52	MAX	56.5	3.62
97.33 %	97.33 %	0 ... 1	88.24 %	88.64 %
2.67 %	2.67 %	1 ... 10	11.75 %	11.36 %
		10 ... 100	0.01 %	

Table 4.6: Residual norms $\|B^*Bq - q\lambda\|$ of XMR compared to BMR, as multiples of $\|B^*B\|n\epsilon_\diamond$. The data for XMR is identical to Table 2.7.

Pract 75 cases		ORTH	Synth 19240 cases	
XMR-TGK	BMR-TGK		XMR-TGK	BMR-TGK
5.35	4.98	AVG	5.34	4.41
2.71	2.47	MED	1.38	1.69
48.40	39.65	MAX	3095	788
81.33 %	89.33 %	0 ... 10	92.59 %	91.04 %
18.67 %	10.67 %	10 ... 100	7.04 %	0.12 %
		100 ... 200	0.12 %	0.08 %
		200 ... 500	0.11 %	0.03 %
		500 ... 10 ³	0.07 %	
		10 ³ ... 10 ⁶	0.06 %	

Table 4.7: Orthogonality $\max\{|\mathbf{U}^*\mathbf{U} - \mathbf{I}|, |\mathbf{V}^*\mathbf{V} - \mathbf{I}|\}$ of XMR-TGK compared to BMR-TGK, as multiples of $n\epsilon_\diamond$. The data for XMR-TGK is the same as in Table 3.3.

Pract 75 cases		RESID	Synth 19240 cases	
XMR-TGK	BMR-TGK		XMR-TGK	BMR-TGK
0.35	0.32	AVG	0.45	0.44
0.07	0.06	MED	0.13	0.13
4.19	3.20	MAX	118	4.67
92.00 %	93.33 %	0 ... 1	84.96 %	85.11 %
8.00 %	6.67 %	1 ... 10	15.03 %	14.89 %
		10 ... 100		
		> 100	0.01 %	

Table 4.8: Residual norms $\max_i \{ \|\mathbf{B}\bar{\mathbf{v}}_i - \bar{\mathbf{u}}_i\bar{\sigma}_i\|, \|\mathbf{B}^*\bar{\mathbf{u}}_i - \bar{\mathbf{v}}_i\bar{\sigma}_i\| \}$ of XMR-TGK compared to BMR-TGK, measured as multiples of $\|\mathbf{B}\|n\epsilon_\diamond$. The data for XMR-TGK is the same as in Table 3.4.

Pract 75 cases		ORTH	Synth 19240 cases	
XMR-CPL	BMR-CPL		XMR-CPL	BMR-CPL
10.71	14.09	AVG	6.33	3.90
2.44	2.12	MED	1.01	1.03
154	424	MAX	27730	1536
82.67 %	80.00 %	0 ... 10	91.03 %	92.14 %
14.67 %	17.33 %	10 ... 100	8.61 %	7.64 %
2.67 %	1.33 %	100 ... 200	0.21 %	0.12 %
	1.33 %	200 ... 500	0.10 %	0.07 %
		500 ... 10 ³	0.02 %	0.02 %
		10 ³ ... 10 ⁶	0.03 %	0.01 %

Table 4.9: Orthogonality $\max\{|\mathbf{U}^*\mathbf{U} - \mathbf{I}|, |\mathbf{V}^*\mathbf{V} - \mathbf{I}|\}$ of XMR-CPL compared to BMR-CPL, as multiples of $n\epsilon_\diamond$. The data for XMR-CPL is the same as in Table 3.3.

Pract 75 cases		RESID	Synth 19240 cases	
XMR-CPL	BMR-CPL		XMR-CPL	BMR-CPL
15.78	7.56	AVG	3.14	1.76
1.37	0.77	MED	0.72	0.45
453	244	MAX	6873	2414
34.67 %	58.67 %	0 ... 1	57.44 %	71.36 %
50.67 %	30.67 %	1 ... 10	35.50 %	25.45 %
8.00 %	9.33 %	10 ... 100	7.00 %	3.15 %
6.67 %	1.33 %	> 100	0.06 %	0.03 %

Table 4.10: Residual norms $\max_i \{\|\mathbf{B}\bar{\mathbf{v}}_i - \bar{\mathbf{u}}_i\bar{\sigma}_i\|, \|\mathbf{B}^*\bar{\mathbf{u}}_i - \bar{\mathbf{v}}_i\bar{\sigma}_i\|\}$ of XMR-CPL compared to BMR-CPL, measured as multiples of $\|\mathbf{B}\|n\epsilon_\diamond$. The data for XMR-CPL is the same as in Table 3.4.

block factorizations with optimally twisted standard factorizations, so the results can probably still be improved once we start using twisted block factorizations.

Without doubt using blocks will cause a performance hit. We have not shown the counts of bisection-, RQI-, and shift-steps because they do not change much from the non-blocked algorithms. However, we cannot evaluate the impact on efficiency at this moment, since there is really no conceivable abstract cost measure to compare Algorithm 4.7 with standard `dstqds`. We mentioned in Remark 4.13 that Algorithm 4.5 for factorizing blocked to non-blocked can be optimized to become not that much more expensive than standard `dstqds`. Furthermore, the step counts in Tables 2.8 and 3.5 convey that the expensive general factorization in Algorithm 4.7 will not be called that often compared to normal bisection of RQI steps. Together this means the overall decrease in performance might turn out not to be that bad after all, but a satisfactory answer to this issue can only be given on the basis of optimized implementations.

Summary

Habe nun, ach! Philosophie,
Juristerei und Medizin,
Und leider auch Theologie!
Durchaus studiert, mit heißem Bemühn.
Da steh ich nun, ich armer Tor!
Und bin so klug als wie zuvor.
— JOHANN W. VON GOETHE, *Faust I* (1808)

The previous pages took us on quite a long journey. We have studied this algorithm with the strange name MR^3 in most of its intricacies (some were still glossed over), have gained understanding in what approaches work (and what do not) for using MR^3 to solve BSVD, and learned to either love or hate mixed relative error analysis by the way of block factorizations.



Scattered along the way through Chapters 2–4 we feel to have made some minor and a few major contributions to the field. Since this is a doctoral thesis, it is customary to summarize them explicitly, but we will limit this to what we, personally, regard as most important.

For the problem BSVD, we have built a robust body of theory that supports in fact two valid solution strategies— MR^3 on the Golub-Kahan matrix and the coupled approach, Algorithm MCR^3 . Both are realized in software and were shown to be not perfect, but predictably effective and reasonably efficient. The tradeoff between them is the old choice between accuracy and speed.

During the work on BSVD we gained quite a lot of theoretical expertise and practical experience with MR^3 . This allowed us to devise and incorporate some new techniques to craft our own implementation of MR^3 . Based on numerical

experiments we have shown it to be significantly more reliable, and argued it to be more efficient, than the current implementation `DSTEMR` of MR^3 in `LAPACK`.

Finally, a new algorithm was devised to compute translates of symmetric tridiagonal matrices given in block-factored form, and componentwise mixed relative stability of the method was rigorously established. Concrete experiments validated that this technology can greatly improve accuracy and reliability of MR^3 -based solution methods for `TSEP` and `BSVD`.



Research is never complete, since answers more often than not lead to better questions. Hence, there remains always something to do.

For us, one of the next steps must be to cast our MR^3 -prototype into optimized form, so that the ambitious claims we made about beating `DSTEMR` in speed can be supported by fact. Then we could also evaluate if the further accuracy gained by including block factorizations for the standard MR^3 is worthwhile, compared to the probable loss in efficiency.

Concerning problem `BSVD`, in our opinion there is really no arguing the fact that Golub-Kahan matrices and their translates should be factorized with blocks. We remarked at the end of Chapter 3 that, actually, a hybrid of using MR^3 solely on the Golub-Kahan matrix and the coupled approach seems to be more promising than either one alone. This needs to be investigated further, because if it turns out to be true, we can enjoy the best properties of both: being able to compute the singular value decomposition of \mathbf{B} stably, and in just little more time than MR^3 requires to solve *one* of the normal equations, $\mathbf{B}\mathbf{B}^*$ or $\mathbf{B}^*\mathbf{B}$.

For block factorizations, we should try to simplify the analysis. The theoretical worst-case error bounds are quite large, but not really felt in the experiments. This indicates that the bounds might be sharpened. The computation of parts of the factorization that constitute ongoing changes in the block structure (“overlap sequences”) was actually simpler than for just breaking a block. But we had to cast a rather coarse net to encompass the possible errors, since we could not get a grip on the inherent interdependency. Further research should be directed to try different computation schemes that might admit a sharper analysis.

We can only see a short distance ahead, but we can see plenty there that needs to be done.

— ALAN M. TURING, *Computing Machinery and Intelligence* (1950)

Everything in the universe denies nothing; to suggest an ending is the one absurdity.

— STEPHEN KING, *The Dark Tower I: The Gunslinger* (1982)

Logic, logic, logic. Logic is the beginning of wisdom, Valeris, not the end.

— SPOCK, in *Star Trek VI: The Undiscovered Country* (1991)

List of Figures

1.1	Functions and placeholders in the notation for error analysis	17
1.2	Geometrical interpretation of an angle between a vector (line) and a 2-dimensional subspace (plane)	22
1.3	The Rayleigh Quotient and its associated residual	30
1.4	The Cauchy Interlace Theorem by the way of an example	35
2.1	MR ³ : Logical structure of the proof of correctness	58
2.2	MR ³ : How the computed eigenpairs relate to the root	58
2.3	MR ³ : Principal situation during the proof of orthogonality	63
2.4	Mixed relative error analysis for <code>dstqds</code>	86
2.5	Mixed relative error analysis for <code>dqds</code>	94
2.6	Mixed relative error analysis for <code>dtwqds</code>	101
3.1	Situation for the proof of Lemma 3.1	122
3.2	Coupling relations for LDL*-factorizations	125
3.3	Data for Experiment 3.4, on a per-vector basis	130
3.4	Principal situation for the analysis of using MR ³ on the normal equations as black box	132
3.5	Data for Experiment 3.6, on a per-vector basis	136
3.6	Why the naive black-box approach of MR ³ on \mathbb{T}_{GK} is doomed . . .	137
3.7	Possible ways to couple data from one central twist to the outside	149
3.8	Outline and logical structure of the proof for Algorithm MCR ³ . .	158
4.1	Detailed situation for breaking a block	179
4.2	Mixed relative error analysis for blocked to non-blocked <code>dstqds</code> .	198
4.3	The four kinds of overlap sequences	202
4.4	Mixed relative error analysis for blocked <code>dstqds</code>	213

List of Tables

1.1	Formats in the IEEE 754 standard	12
2.1	Parameters in the requirements for MR ³	54
2.2	Secondary/derived data for representations based on bidiagonal factorizations	83
2.3	Error bounds for <code>dstqds</code>	86
2.4	Error bounds for <code>dqds</code>	94
2.5	Error bounds for <code>dtwqds</code>	99
2.6	Orthogonality of DSTEMR compared to XMR	113
2.7	Residual norms of DSTEMR compared to XMR	114
2.8	Efficiency of DSTEMR compared to XMR	116
2.9	Some additional statistics comparing DSTEMR and XMR	116
3.1	Notational adjustments to handle representations associated with the three root matrices.	126
3.2	Singular value distribution of test matrix in Experiment 3.4	129
3.3	Orthogonality of XMR-TGK compared to XMR-CPL	163
3.4	Residual norms of XMR-TGK compared to XMR-CPL	163
3.5	Efficiency of XMR-TGK compared to XMR-CPL	164
4.1	Standard formulae for the next adjustment s_{i+1}	178
4.2	Alternative formulae for s_{i+1}	179
4.3	Error bounds for blocked to non-blocked <code>dstqds</code>	198
4.4	Error bounds for blocked <code>dstqds</code>	213
4.5	Orthogonality levels of XMR compared to BMR	216
4.6	Residual norms of XMR compared to BMR	216
4.7	Orthogonality of XMR-TGK compared to BMR-TGK	217
4.8	Residual norms of XMR-TGK compared to BMR-TGK	217
4.9	Orthogonality of XMR-CPL compared to BMR-CPL	218

4.10 Residual norms of XMR-CPL compared to BMR-CPL	218
--	-----

List of Algorithms

2.1	The MR ³ Algorithm.....	51
2.2	Factorize $T = LDL^*$	65
2.3	Factorize $T = URU^*$	66
2.4	Solve for eigenvector	72
2.5	RQI with twisted factorizations	75
2.6	Stationary qd-transform with shift	84
2.7	<code>dstqds</code> —template for all representations	87
2.8	Progressive qd-transform with shift.....	91
2.9	<code>dqds</code> —template for all representations.....	93
2.10	<code>dtwqds</code> —template for all representations	100
2.11	Optimized bisection, all representation-types	103
3.1	MR ³ as black box on the normal equations.....	128
3.2	MR ³ on the Golub-Kahan matrix	138
3.3	Algorithm MCR ³	153
4.1	Template for the stationary block factorization	181
4.2	Keep a block.....	186
4.3	Branches to break a block.....	190
4.4	Break a block	195
4.5	Factorize blocked to non-blocked	197
4.6	Compute adjustments for an overlap sequence.....	203
4.7	blocked <code>dstqds</code>	210
4.7	blocked <code>dstqds</code> (continued)	211

Bibliography

- [1] E. Anderson, Z. Bai, C. H. Bischof, L. S. Blackford, J. W. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' guide (third ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.
- [2] <http://www.netlib.org/lapack>.
- [3] Dominic Antonelli and Christof Vömel. PDSYEV. SCALAPACK's parallel MRRR algorithm for the symmetric eigenvalue problem. Technical Report UCB//CSD-05-1399, University of California, Berkeley, 2005. (Also available as LAPACK Working Note #168).
- [4] Paolo Bientinesi, Inderjit S. Dhillon, and Robert A. van de Geijn. A parallel eigensolver for dense symmetric matrices based on multiple relatively robust representations. *SIAM Journal on Scientific Computing*, 27(1):43–66, 2005.
- [5] Christian H. Bischof, Bruno Lang, and Xiaobai Sun. Algorithm 807: The SBR-toolbox-software. *ACM Transactions on Mathematical Software (TOMS)*, 26(4):602–616, 2000.
- [6] Christian H. Bischof, Bruno Lang, and Xiaobai Sun. A framework for symmetric band reduction. *ACM Transactions on Mathematical Software (TOMS)*, 26(4):581–601, 2000.
- [7] Adam W. Bojanczyk, Ruth Onn, and Allan O. Steinhardt. Existence of the hyperbolic singular value decomposition. *Linear Algebra and its Applications*, 185:21–30, 1993.
- [8] J.J.M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numerische Mathematik*, 36:177–195, 1981.

- [9] Chandler Davis and William Kahan. The rotation of eigenvectors by a perturbation. III. *SIAM Journal on Numerical Analysis*, 7(1):1–47, 1970.
- [10] James W. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.
- [11] James W. Demmel and William Gragg. On computing accurate singular values and eigenvalues of matrices with acyclic graphs. *Linear Algebra and its Applications*, 185:203–217, 1993.
- [12] James W. Demmel and William Kahan. Accurate singular values of bidiagonal matrices. *SIAM Journal on Scientific Computing*, 11(5):873–912, 1990.
- [13] James W. Demmel, Osni A. Marques, Beresford N. Parlett, and Christof Vömel. Performance and Accuracy of LAPACK’s Symmetric Tridiagonal Eigensolvers. *SIAM Journal on Scientific Computing*, 30(3):1508–1526, 2008.
- [14] James W. Demmel, Osni A. Marques, Beresford N. Parlett, and Christof Vömel. A testing infrastructure for LAPACK’s symmetric tridiagonal eigensolvers. *ACM Transactions on Mathematical Software (TOMS)*, 35(1):1–13, 2008. (Also available as LAPACK Working Note #182).
- [15] Inderjit S. Dhillon. *A new $O(n^2)$ algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem*. PhD thesis, University of California, Berkeley, 1997.
- [16] Inderjit S. Dhillon. Reliable computation of the condition number of a tridiagonal matrix in $O(n)$ time. *SIAM Journal on Matrix Analysis and Applications*, 19(3):776–796, July 1998.
- [17] Inderjit S. Dhillon and Beresford N. Parlett. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra and its Applications*, 387:1–28, August 2004.
- [18] Inderjit S. Dhillon and Beresford N. Parlett. Orthogonal eigenvectors and relative gaps. *SIAM Journal on Matrix Analysis and Applications*, 25(3):858–899, August 2004.
- [19] Inderjit S. Dhillon, Beresford N. Parlett, and Christof Vömel. Glued matrices and the MRRR algorithm. *SIAM Journal on Scientific Computing*, 27(2):496–510, 2005. Revised version of LAPACK Working Note #163.
- [20] Inderjit S. Dhillon, Beresford N. Parlett, and Christof Vömel. The design and implementation of the MRRR algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 32(4):533–560, 2006.

- [21] Zlatko Drmač and Krešimir Veselić. New fast and accurate Jacobi SVD algorithm: I. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1322–1342, 2007. (Also available as LAPACK Working Note #169).
- [22] Zlatko Drmač and Krešimir Veselić. New fast and accurate Jacobi SVD algorithm: II. *SIAM Journal on Matrix Analysis and Applications*, 29(4):1343–1362, 2007. (Also available as LAPACK Working Note #170).
- [23] Stanley C. Eisenstat and Ilse C. F. Ipsen. Relative perturbation techniques for singular value problems. *SIAM Journal on Numerical Analysis*, 32(6):1972–1988, December 1995.
- [24] Stanley C. Eisenstat and Ilse C. F. Ipsen. Relative perturbation results for eigenvalues and eigenvectors of diagonalisable matrices. *BIT Numerical Mathematics*, 38(3):502–509, 1998.
- [25] Haw-Ren Fang and Dianne P. O’Leary. Stable factorizations of symmetric tridiagonal and triadic matrices. *SIAM Journal on Matrix Analysis and Applications*, 28(2):576–595, 2007.
- [26] K. V. Fernando. Accurate BABE factorisation of tridiagonal matrices for eigenproblems. Technical Report TR5/95, Numerical Algorithms Group Ltd, Wilkinson House, Jordan Hill, Oxford, 1985.
- [27] K. V. Fernando. On computing an eigenvector of a tridiagonal matrix. Part I: Basic results. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1013–1034, October 1997.
- [28] K. V. Fernando. Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices. *SIAM Journal on Matrix Analysis and Applications*, 20(2):373–399, 1998.
- [29] K. V. Fernando. Computation of exact inertia and inclusions of eigenvalues (singular values) of tridiagonal (bidiagonal) matrices. *Linear Algebra and its Applications*, 422(1):77–99, 2007.
- [30] K. V. Fernando and Beresford N. Parlett. Accurate singular values and differential qd algorithms. *Numerische Mathematik*, 67:191–229, 1994.
- [31] Carla M. A. Ferreira and Lisa Miranian. (no title). Notes on qd-algorithms using blocks, 2005.
- [32] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

-
- [33] Gene H. Golub and William Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM Journal on Numerical Analysis (Series B)*, 2(2):205:224, 1965.
- [34] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 3rd edition, 1996.
- [35] Benedikt Großer. *Ein paralleler und hochgenauer $O(n^2)$ Algorithmus für die bidiagonale Singulärwertzerlegung*. PhD thesis, Bergische Universität Gesamthochschule Wuppertal, Fachbereich Mathematik, Wuppertal, Germany, 2001. In German.
- [36] Benedikt Großer and Bruno Lang. An $O(n^2)$ algorithm for the bidiagonal SVD. *Linear Algebra and its Applications*, 358:45–70, 2003.
- [37] Benedikt Großer and Bruno Lang. On symmetric eigenproblems induced by the bidiagonal SVD. *SIAM Journal on Matrix Analysis and Applications*, 26(3):599–620, 2005.
- [38] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM Journal on Matrix Analysis and Applications*, 16:79–92, 1995.
- [39] Ming Gu and Stanley C. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM Journal on Matrix Analysis and Applications*, 16:172–191, 1995.
- [40] Nicholas J. Higham. A survey of componentwise perturbation theory in numerical linear algebra. In *Mathematics of Computation 1943–1993: A Half Century of Computational Mathematics*, pages 49–77, 1994.
- [41] Nicholas J. Higham. Stability of the diagonal pivoting method with partial pivoting. *SIAM Journal on Matrix Analysis and Applications*, 18(1):52–65, January 1997.
- [42] Nicholas J. Higham. Stability of block LDL^T factorization of a symmetric tridiagonal matrix. *Linear Algebra and its Applications*, 287:181–189, 1999.
- [43] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, 2nd edition, 2002.
- [44] IEEE. *IEEE Standard 754-1985 for binary floating-point arithmetic*, August 1985.
- [45] IEEE. *IEEE Standard 754-2008 for floating-point arithmetic*, August 2008.

- [46] Ilse C. F. Ipsen. Computing an eigenvector with inverse iteration. *SIAM Review*, 39(2):254–291, June 1997.
- [47] Camille Jordan. Mémoires sur les formes bilinéaires. *Journal de Mathématiques Pures et Appliquées*, 19:35–54, 1874.
- [48] William Kahan. Accurate eigenvalues of a symmetric tri-diagonal matrix. Technical Report CS41, Computer Science Department, Stanford University, July 1966.
- [49] William Kahan. Lecture notes on the status of IEEE standard 754 for binary floating point arithmetic, 1995.
- [50] Bruno Lang. Reduction of banded matrices to bidiagonal form. *Zeitschrift für Angewandte Mathematik und Mechanik*, 76:155–158, 1996.
- [51] Ren-Cang Li. Relative perturbation theory: I. Eigenvalue and singular value variations. *SIAM Journal on Matrix Analysis and Applications*, 19(4):956–982, 1998.
- [52] Ren-Cang Li. Relative perturbation theory: II. Eigenspace and singular subspace variations. *SIAM Journal on Matrix Analysis and Applications*, 20(2):471–492, 1998.
- [53] Osni A. Marques, Beresford N. Parlett, and Christof Vömel. Computations of Eigenpair Subsets with the MRRR algorithm. *Numerical Linear Algebra with Applications*, 13(8):643–653, 2006.
- [54] Osni A. Marques, Jason Riedy, and Christof Vömel. Benefits of IEEE-754 Features in Modern Symmetric Tridiagonal Eigensolvers. *SIAM Journal on Scientific Computing*, 28(5):1613–1633, 2006.
- [55] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*. SIAM, Philadelphia, PA, 2000.
- [56] Beresford N. Parlett. *The symmetric eigenvalue problem*. Prentice Hall, Englewood Cliffs, NJ, 1980.
- [57] Beresford N. Parlett. The new qd algorithms. In *Acta Numerica*, pages 459–491. Cambridge University Press, 1995.
- [58] Beresford N. Parlett. Invariant subspaces for tightly clustered eigenvalues of tridiagonals. *BIT Numerical Mathematics*, 36(3):542–562, 1996.
- [59] Beresford N. Parlett. Spectral sensitivity of products of bidiagonals. *Linear Algebra and its Applications*, 275-276:417–431, 1998.

- [60] Beresford N. Parlett. Perturbation of eigenpairs of factored symmetric tridiagonal matrices. *Foundations of Computational Mathematics*, 3(2):207–223, 2003.
- [61] Beresford N. Parlett. A bidiagonal matrix determines its hyperbolic SVD to varied relative accuracy. *SIAM Journal on Matrix Analysis and Applications*, 26(4):1022–1057, 2005.
- [62] Beresford N. Parlett and Inderjit S. Dhillon. Fernando’s solution to Wilkinson’s problem: an application of double factorization. *Linear Algebra and its Applications*, 267:247–279, 1997.
- [63] Beresford N. Parlett and Inderjit S. Dhillon. Relatively robust representations of symmetric tridiagonals. *Linear Algebra and its Applications*, 309:121–151, 2000.
- [64] Beresford N. Parlett and Osni A. Marques. An implementation of the dqds algorithm (positive case). *Linear Algebra and its Applications*, 309:217–259, 2000.
- [65] Beresford N. Parlett and Christof Vömel. Detecting localization in an invariant subspace. in preparation, 2009.
- [66] Beresford N. Parlett and Christof Vömel. The spectrum of a glued matrix. *SIAM Journal on Matrix Analysis and Applications*, 31:114–132, 2009.
- [67] Heinz Rutishauer. Der Quotienten-Differenzen-Algorithmus. *Zeitschrift für angewandte Mathematik und Physik*, 5(3):233–251, 1954.
- [68] G. W. Stewart. On the early history of the singular value decomposition. *sirev*, 35(4):551–566, 1993.
- [69] Lloyd N. Trefethen and David Bau III. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.
- [70] Christof Vömel. ScaLAPACK’s MRRR algorithm. *ACM Transactions on Mathematical Software (TOMS)*, 37(1), 2009. (Also available as LAPACK Working Note #195).
- [71] James H. Wilkinson. *The Algebraic Eigenvalue Problem*. Oxford University Press, 1965. ISBN 0-19-853418-3.
- [72] Paul R. Willems, Bruno Lang, and Christof Vömel. Computing the bidiagonal SVD using multiple relatively robust representations. *SIAM Journal on Matrix Analysis and Applications*, 28(4):907–926, 2007.

Summary of Notation

VECTOR SPACES

\mathbb{R}^n	n -dimensional Euclidean Space over \mathbb{R}
$\mathcal{U}, \mathcal{V}, \mathcal{S}, \dots$	Subspaces of \mathbb{R}^n
$\text{span}\{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$	Subspace spanned by the given vectors
$\dim \mathcal{U}$	Dimension of \mathcal{U}
\mathcal{U}^\perp	Orthogonal complement of \mathcal{U}
$\mathcal{U} \perp \mathcal{V}$	True iff $\mathbf{u}^* \mathbf{v} = 0$ for all $\mathbf{u} \in \mathcal{U}, \mathbf{v} \in \mathcal{V}$.
$\angle(\mathcal{U}, \mathcal{V})$	Largest (acute) principal angle between \mathcal{U} and \mathcal{V} , see §1.3.

VECTORS

$\mathbf{x}, \mathbf{y}, \dots$	(Column-) Vectors $\in \mathbb{R}^n$ (lowercase sans-serif roman letters)
\mathbf{x}^*	Transpose of \mathbf{x} , i.e., a row-vector in $\mathbb{R}^{1 \times n}$
$\ \mathbf{x}\ $	$= \ \mathbf{x}\ _2 = \sqrt{\mathbf{x}^* \mathbf{x}}$, Euclidean norm of \mathbf{x}
$\angle(\mathbf{x}, \mathbf{y})$	Acute angle between $\text{span}\{\mathbf{x}\}$ and $\text{span}\{\mathbf{y}\}$, see §1.3.
dimension	n unless stated otherwise
indexing	Always starts with 1
$\mathbf{x}(i)$	Entry i of the vector \mathbf{x}
$\mathbf{x}(I)$ or \mathbf{x}_I	Subvector of \mathbf{x} consisting of entries with index in I

SPECIAL ENTITIES:

\mathbf{o}	The zero vector, dimension from context
\mathbf{e}_j	$= \mathbf{I}(:, j)$, the j th column of the identity, dimension from context

 MATRICES

A, B, \dots	Matrices (Uppercase sans-serif roman letters)
dimension	$n \times n$ unless stated otherwise
indexing	Always starts with 1 for rows and columns
$A(i, j)$ or $A_{i,j}$	Entry at row i , column j of the matrix A
$A(I, J)$ or $A_{I,J}$	Submatrix of A containing all elements with row-indices in I and column-indices in J , i.e., with dimension $ I \times J $
A_I	$= A_{I,I}$, principal submatrix
$\ A\ $	Spectral norm of A , operator norm induced by Euclidean vector norm
$ A $	Can mean the matrix of absolutes or the largest absolute entry of A , depending on context.
A^{-1}	Inverse of A
A^*	Transpose of A
A^{-*}	$= (A^{-1})^* = (A^*)^{-1}$
$\text{null}(A)$	Nullspace of A
$\text{range}(A)$	Space spanned by the columns of A
$\det A$	Determinant of A
$\text{diag}_k[A]$	Matrix of same dimensions as A with all entries zero except for the k th band, which is retained from A ; $k = 0$ if omitted, $\pm k$ for symmetric bands, allow index set I instead of k to grab multiple bands at once.
$\text{diag}_k(a_1, \dots, a_{n-k})$	Matrix of dimension $n \times n$, zero except for the entries a_i in its k th band; $k = 0$ if omitted, write $\pm k$ to get a symmetric matrix.
$A - \sigma$	Shifted matrix $A - \sigma I$ (we omit I)
$\rho_A(x)$	Rayleigh Quotient $x^*Ax/\ x\ ^2$, implies $x \neq 0$, see §1.4.1.

SPECIAL ENTITIES:

0	The zero matrix, dimension from context
I	The identity matrix, dimension from context

EIGENSYSTEMS

$\lambda_i[\mathbf{A}]$	i th smallest eigenvalue of \mathbf{A} (for $i > 0$)
$\lambda_{-i}[\mathbf{A}]$	i th largest eigenvalue of \mathbf{A} (for $i > 0$)
$\mathbf{q}_{\pm i}[\mathbf{A}]$	Normalized eigenvector to $\lambda_{\pm i}[\mathbf{A}]$, chosen so that $\{\mathbf{q}_i[\mathbf{A}] : 1 \leq i \leq n\}$ forms an orthonormal basis of \mathbb{R}^n .
$\mathcal{Q}_I[\mathbf{A}]$	$= \text{span} \{\mathbf{q}_i[\mathbf{A}] : i \in I\}$
$\text{spec}(\mathbf{A})$	Set of all eigenvalues of \mathbf{A}
$\text{spdiam}[\mathbf{A}]$	$= \lambda_n[\mathbf{A}] - \lambda_1[\mathbf{A}]$, spectral diameter
$\text{gap}_{\mathbf{A}}(I)$	$\min \{ \lambda_j - \lambda_i : i \in I, j \notin I\}$, undefined if $I = \{1, \dots, n\}$
$\text{gap}_{\mathbf{A}}(\mu)$	$\min \{ \mu - \lambda : \mu \neq \lambda \in \text{spec}(\mathbf{A})\}$
$\text{relgap}_{\mathbf{A}}(I)$	$\min \{ \lambda_j - \lambda_i / \lambda_i : i \in I, j \notin I\}$, undefined if $I = \{1, \dots, n\}$
$\text{relgap}_{\mathbf{A}}(\mu)$	$\min \{ \mu - \lambda / \mu : \mu \neq \lambda \in \text{spec}(\mathbf{A})\}$

OTHER
NUMBERS:

\mathbb{N}	$= \{1, 2, \dots\}$
\mathbb{N}_0	$= \mathbb{N} \cup \{0\}$
$\text{sign}(x)$	$\in \{-1, 0, +1\}$

FLOATING-POINT ARITHMETIC:

\mathbb{F}	Set of floating-point numbers
ϵ_{\diamond}	Machine epsilon; may be used instead of $\epsilon^{[0]}(1)$ if in placeholder context (right side of a \doteq).
u_{\diamond}	Underflow threshold
o_{\diamond}	Overflow threshold
$\text{fl}(z)$	Nearest floating-point number to $z \in \mathbb{R}$. Can also be used with a term, like $\text{fl}(a + b)$ to indicate the computed result.
\bar{x}	(Overbar) Indicates a computed quantity

ERROR ANALYSIS:

\tilde{x}	Indicates a perturbed something
$x \rightsquigarrow \tilde{x}$	Means x is perturbed to \tilde{x} ; for scalar x also identified with $(\tilde{x} - x)/x$.
$\epsilon^{[p]}(m)$	Placeholder for unspecified quantity bounded by $m\epsilon_{\diamond}/(1 - pm\epsilon_{\diamond})$, defined only for $p, m \geq 0$ and $0 \leq pm\epsilon_{\diamond} < 1$.
$\epsilon(m)$	$= \epsilon^{[1]}(m)$
\doteq	Indicates that an equality involves placeholders; should be interpreted from left to right to avoid ambiguity.
