Ein paralleler und hochgenauer $\mathcal{O}(n^2)$ Algorithmus für die bidiagonale Singulärwertzerlegung

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

am Fachbereich Mathematik der
Bergischen Universität Gesamthochschule Wuppertal
genehmigte

Dissertation

von

Dipl.-Math. Benedikt Großer

aus Wuppertal

Tag der mündlichen Prüfung:	27. März 2001
Referent:	Prof. Dr. A. Frommer
Korreferent:	PD Dr. B. Lang

Vorwort

Eigen- und Singulärwertberechnungen zählen zu den Grundaufgaben vieler wissenschaftlicher Anwendungen. Je nach Aufgabenstellung steht eine Auswahl verschiedener numerischer Lösungsverfahren bereit, die Näherungen an die gesuchten Größen liefern.
In exakter Arithmetik lassen sich elementare Beziehungen zwischen dem symmetrischen Eigenwertproblem und der Singulärwertzerlegung herleiten. Der naheliegende Ansatz, bewährte Eigenwertverfahren als Black-Box entweder auf die Normalengleichungen oder die Jordan-Wielandt-Darstellung anzuwenden, führt aber durch Rundungsfehler bedingt oft nicht zu befriedigenden Ergebnissen. Die Folge ist teilweise hoher zusätzlicher algorithmischer Aufwand oder die Entwicklung speziell auf die Singulärwertzerlegung zugeschnittener Lösungen.

◇

Für das durch eine symmetrisch tridiagonale Matrix gestellte Eigenwertproblem gilt der RRR-Algorithmus (RRR = relativ robuste Repräsentationen) als Standardmethode. Das Hauptziel dieser Arbeit ist die Übertragung dieses Algorithmus auf die Singulärwertzerlegung einer bidiagonalen Matrix. Die dazu angestellten Untersuchungen münden in die Einführung von Kopplungsmechanismen, die die Erfüllung der zusätzlichen Genauigkeitsanforderungen gewährleisten. Daneben ergeben sich einige Verbesserungsvorschläge für den RRR-Algorithmus selbst.
In dieser Arbeit werden theoretische Resultate über die Qualität der Eigenwerte bei separaten und gekoppelten Faktorisierungen hergeleitet. Als Komplement zum RRR-Algorithmus wird eine Methode zur Konstruktion idealer

Systeme vorgeschlagen. Im Anschluß zeigen ausführliche numerische Experimente die grundlegende Überlegenheit der RRR-basierten Ansätze gegenüber konkurrierenden Lösungsverfahren. Darüberhinaus wird die Eignung für parallele Rechnerarchitekturen demonstriert.

◇

Nach Einführung elementarer Notation behandelt das erste Kapitel bisher bekannte Ergebnisse der relativen Störungstheorie. Insbesondere wird hier auf die Eigenschaft einer Matrix eingegangen, eine RRR zu besitzen. Es folgt die Präsentation gängiger Lösungsstrategien für Singulärwertzerlegungen und symmetrische Eigenwertprobleme. Für den wichtigen Fall bidiagonaler bzw. symmetrisch tridiagonaler Matrizen existieren eine Reihe speziell ausgerichteter Verfahren.
Zu diesen gehört auch der RRR-Algorithmus, dessen Beschreibung das zweite Kapitel gewidmet ist. Obwohl in der Grundidee leicht zu verstehen, erfordert die nachfolgend beschriebene Übertragung auf die Singulärwertzerlegung eine teilweise sehr detaillierte Schilderung.
Nach diesen Vorbereitungen werden im dritten Kapitel Probleme bei einer Black-Box-Anwendung auf die Normalengleichungen und die Jordan-Wielandt-Darstellung konkretisiert. Die vorgeschlagene Lösung besteht darin, den RRR-Algorithmus explizit nur auf eine der Normalengleichungen anzuwenden und die entsprechenden Faktorisierungen der zweiten implizit mitzuführen. Die dazu vorgestellten Kopplungsmechanismen erlauben die Herleitung verschiedener störungstheoretischer Resultate. Die Einbettung der Kopplungen in den RRR-Algorithmus kann ohne größere Änderungen des eigentlichen Verfahrens erfolgen. Auch der zusätzlich anfallende Rechenaufwand bleibt beschränkt.
Der soweit entwickelte Kernalgorithmus wird im vierten Kapitel mit einem Präkonditionierungsansatz und der Konstruktion idealer Systeme erweitert zu einem Gesamtverfahren für die Bestimmung der bidiagonalen Singulärwertzerlegung. Es folgt die eingehende Darstellung serieller numerischer Experimente.
Das fünfte Kapitel befaßt sich mit der Parallelisierung des Gesamtverfahrens. Die für den Kernalgorithmus per se gegebenen guten Skalierungseigenschaften werden bei der Implementierung umgesetzt. Bei den umrahmenden Teilaufgaben stehen schnelle, aber serialisierende Methoden gut skalierenden Verfahren gegenüber, die aber auf kleinen Prozessorzahlen noch unterlegen sind. Lösungsvorschlag ist hier eine hybride Strategie, die durch Einstellung von Parametern gesteuert werden kann.

Eine kurze Zusammenfassung der wichtigsten Ergebnisse und einige Vorschläge für zukünftige Aufgaben schließen die Arbeit ab. Als Orientierungshilfe empfiehlt sich der Anhang mit einem Überblick der wichtigsten Fakten in Stichworten.

◇

Die vorliegende Dissertation entstand im Zeitraum vom August 1997 bis Januar 2001 am Fachbereich Mathematik der Bergischen Universität Gesamthochschule Wuppertal.
Mein besonderer Dank gilt meinem Doktorvater Privatdozent Dr. Bruno Lang für die interessante Themenstellung und seine stets ergiebige Diskussionsbereitschaft. Herrn Prof. Dr. Andreas Frommer danke ich für die Aufnahme in seine Arbeitsgruppe „Scientific Computing“ und für seine engagierte Unterstützung bei der Erstellung dieser Dissertation.
Allen Mitgliedern der Arbeitsgruppe danke ich für langjährige gute und anregende Zusammenarbeit.

Wuppertal, im April 2001 Benedikt Großer

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Behandlung von Singulärwertzerlegungen (engl. *Singular Value Decomposition*, kurz **SVD**[1]) oder den nahe verwandten symmetrischen Eigenwertproblemen (engl. *Symmetric Eigenproblem*, kurz **SEP**) ist eine häufig wiederkehrende elementare Aufgabe in einer Vielzahl von Anwendungsbereichen. Da eine analytische Darstellung der Lösungen für das **SEP** und für die **SVD** ab Matrixdimensionen $n > 4$ i.a. nicht mehr möglich ist, versuchen wir mit Hilfe von *numerischen Lösungsverfahren* (auch kurz als *Löser* bezeichnet) Näherungen zu gewinnen. Die Eigenschaften solcher Lösungsverfahren sind im hohen Maße von der zugrunde liegenden *Rechnerarchitektur* abhängig. Die Beurteilung und der Einsatz von Lösungsverfahren und deren *Implementierungen* erfolgen unter Einbeziehung diverser Faktoren, die in Abbildung 1 skizziert sind.
Besonderes Interesse gilt Lösungsverfahren, die mit möglichst geringem Zeitaufwand arbeiten. Die Komplexität hängt neben dem Umfang der zu ermittelnden Ergebnisse vor allem von der Matrixdimension n ab. Für größere Probleme kommt ferner der effizienten Speicherverwaltung eine wichtige Bedeutung zu. Bei der Parallelisierung wird versucht, das Arbeitsvolumen möglichst gleichmäßig auf mehrere Rechner zu verteilen. Nicht zuletzt wird von einem Löser erwartet, daß er den Genauigkeitsanforderungen genügende Resultate liefert.
Nach der Einführung von Notation beschreiben wir in diesem Kapitel die Aufgabenstellung, den Einfluß der Rechnerarchitektur sowie gängige

[1] Wir werden Abkürzungen jeweils für alle auftretenden Deklinationsformen verwenden.

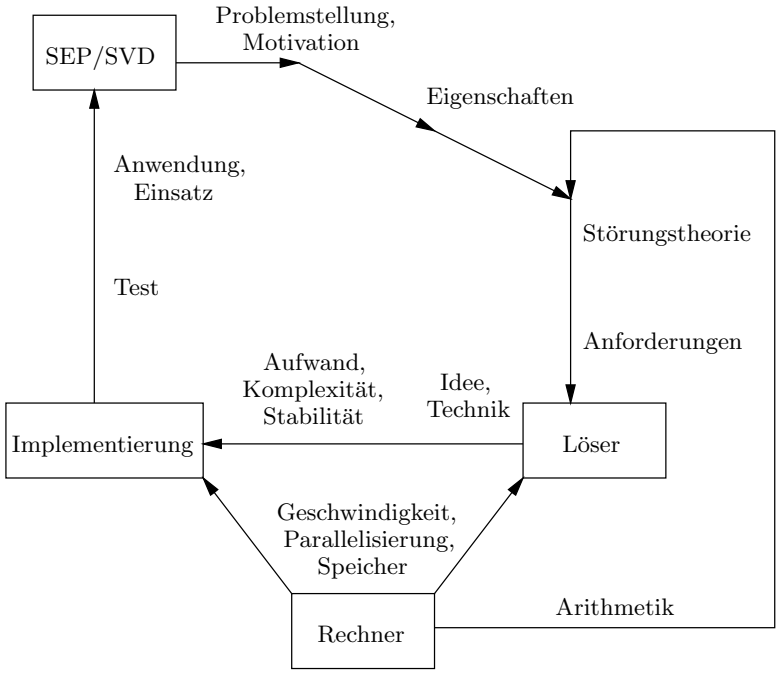


Abbildung 1.1: Numerische Behandlung von **SEP** und **SVD**.

Lösungsverfahren. Zur Vorbereitung der nachfolgenden Kapitel betrachten wir auch die Spezialfälle der bidiagonalen Singulärwertzerlegung (kurz **bSVD**) und des tridiagonalen symmetrischen Eigenwertproblems (kurz **tSEP**).

1.1 Notation

1.1.1 Matrizen, Vektoren, Doppelpunkt-Notation

Wir beschreiben reelle Matrizen durch große und reelle Vektoren durch kleine lateinische Buchstaben. Einträge von Vektoren werden durch Indizes spezifiziert wie z.B. $x = [x_1, \ldots, x_n]$. Der k-te Einheitsvektor wird mit e_k bezeichnet. Einzelne Elemente von Matrizen sind durch die Darstellung $A = (a_{i,j})$ gegeben. Für den Zugriff auf größere Blöcke verwenden wir auch die Doppelpunkt-Notation [61]:

Definition 1.1 (Doppelpunkt-Notation)
Seien $f, l \in \mathbb{N}, s \in \mathbb{Z}$. Ein Vektor $[f, f+s, f+2s\ \ldots, f+ks]$ mit

$$\begin{cases} f+ks & \leq & l & < & f+(k+1)s, & \text{falls } s > 0 \\ f+ks & \geq & l & > & f+(k+1)s, & \text{falls } s < 0 \end{cases}$$

wird durch die *Doppelpunkt-Notation* $f : s : l$ abgekürzt. Für $x \in \mathbb{R}^n$ ist dann $x(f : s : l) := [x_f, x_{f+s}, \ \ldots, x_{f+ks}]$. Für $s = 1$ schreiben wir auch $f : l$. Für $f = s = 1$, $l = n$ notieren wir $1 : n$ oder einfach nur einen Doppelpunkt. Im Falle $f > l$ ist $x(f : l)$ als leeres Tupel zu betrachten.

Mit $x(n : -1 : 1)$ bezeichnen wir also den Vektor, der die Elemente von x in umgekehrter Reihenfolge enthält.
Die Doppelpunkt-Notation auf Matrizen $A \in \mathbb{R}^{m \times n}$ übertragend beschreiben wir beispielsweise einzelne Spalten durch $A(:, j)$, $j \leq n$ und $d+1$ aufeinanderfolgende Spalten durch $A(:, j : j+d)$. Analog können einzelne Zeilen durch $A(i, :)$, $i \leq m$ dargestellt werden. Mit $A(1 : 2 : n, :)$ extrahieren wir folglich die ungeraden Zeilen von A. Gelegentlich stellen wir Matrizen aber auch durch ihre Spaltenvektoren $A = [a_1, a_2, \ \ldots, a_n]$ dar. Die Bedeutung von Schreibweisen wie a_j wird jeweils aus dem Zusammenhang klar.
Die Doppelpunkt-Notation wird auch zur Beschreibung von Zählschleifen bei der Formulierung von Algorithmen verwendet.

Definition 1.2 (Eigenschaften, speziell strukturierte Matrizen)
Eine quadratische Matrix $A \in \mathbb{R}^{n \times n}$ heißt *symmetrisch*, wenn ihre *Transponierte* $A^T = A$ ist, und *orthogonal*, wenn $A^T A = AA^T = I$, wobei I die Einheitsmatrix der Dimension n ist. A heißt *regulär*, wenn ihre *Inverse* A^{-1} existiert, ansonsten *singulär*.
Neben dicht besetzten Matrizen, für die die Doppelpunkt-Notation eine geeignete Beschreibung ist, werden wir häufig Matrizen benutzen, die nur sehr wenige nicht-triviale Einträge haben. Für diese verwenden wir eine andere Art der Darstellung [61]:

- Zu einem gegebenen Vektor $x \in \mathbb{R}^n$ definieren wir $\texttt{diag}(x, k)$ als eine quadratische Matrix der Dimension $n+|k|$ mit den Elementen von x auf der k-ten Nebendiagonalen. Für $k = 0$ entsteht eine Diagonalmatrix. Bei $k > 0$ liegen die Einträge von x oberhalb, bei $k < 0$ unterhalb der Hauptdiagonalen. Abbildung 1.2 zeigt einige Beispiele.

- Die Einträge einer oberen bidiagonalen Matrix sind gegeben durch $B = \texttt{diag}([a_1, \ \ldots, a_n]) + \texttt{diag}([b_1, \ \ldots, b_{n-1}], 1)$

- Obere und untere Bidiagonalmatrizen mit Einsen auf der Diagonalen bezeichnen wir mit $L = I + \texttt{diag}([l_1, \ \ldots, l_{n-1}], -1)$ und $U = I + \texttt{diag}([u_1, \ \ldots, u_{n-1}], 1)$.

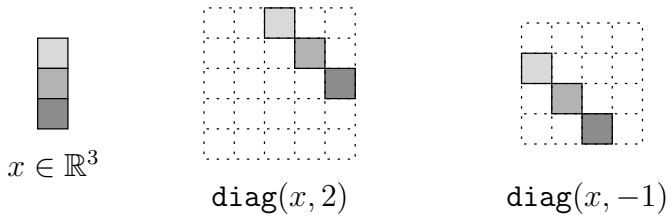


Abbildung 1.2: Beispiele zur `diag`-Notation.

- Eine symmetrisch tridiagonale Matrix notieren wir mit $T = \texttt{diag}([a_1, \ \ldots, a_n]) + \texttt{diag}([b_1, \ \ldots, b_{n-1}], \pm 1)$
- Für $d = [d_1, \ \ldots, d_n]$ beschreiben wir Diagonalmatrizen auch durch $D = \texttt{diag}(d)$. Ist der Zusammenhang klar, verwenden wir D und d synonym.
- Für die Darstellung von Block-Diagonalmatrizen verwenden wir $\texttt{diag}(A_1, \ \ldots, A_m)$ mit quadratischen Matrizen A_k.

Zum bequemeren Lesen von Algorithmen setzen wir außerhalb der Matrix liegende Elemente wie a_0 oder l_n auf Null.

1.1.2 Elementare Operationen auf Matrizen

Definition 1.3 (Translation, LDU-Faktorisierung, Breakdowns)
Für eine Matrix A und einen reellen *Shift* τ bezeichnen wir die Matrix $A - \tau I$ als *Translation* von A.
Für Matrizen und ihre Translationen sind häufig *Faktorisierungen* $A - \tau I = LDU^T$ zu bilden. Dabei ist D eine Diagonalmatrix, während L und U obere bzw. untere Dreiecksmatrizen mit Einsen auf der Diagonalen sind. Die Diagonaleinträge von D bezeichnen wir auch als *Pivotelemente*. LDU-Faktorisierungen sind nicht immer möglich, beispielsweise wenn

$$A - \tau I = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Wir sprechen deshalb von einem *Breakdown*, wenn eine derartige Zerlegung nicht durchführbar ist. Kennzeichen für diese Situation ist, daß ein Pivotelement $d_i = 0$ mit $i < n$ auftritt. Solche Breakdowns sind entweder wie oben durch die Struktur der Matrix bedingt oder entstehen durch Rundungsfehler in numerischen Verfahren.

In manchen Fällen läßt sich aber Abhilfe schaffen. Für eine spezielle Variante der *LDU*-Zerlegung (Stichwort QD-artige Transformationen) gehen wir in Abschnitt 2.3 auf die Behandlung von Breakdowns genauer ein.

Definition 1.4 (QR-Faktorisierung)
Eine *QR-Faktorisierung* einer Matrix $A \in \mathbb{R}^{m\times n}$ ist gegeben durch $A = QR$, wobei $Q \in \mathbb{R}^{m\times m}$ orthogonal und $R \in \mathbb{R}^{m\times n}$ obere Dreiecksmatrix ist.

Definition 1.5 (Minoren)
Für $A \in \mathbb{R}^{n\times n}$ sei $\text{minor}(A, i, j)$ die Matrix, die durch Streichen der i-ten Zeile und j-ten Spalte entsteht. Ist $i = j$, so schreiben wir auch abkürzend $\text{minor}(A, j)$.

Definition 1.6 (Determinante)
Für $A \in \mathbb{R}^{1\times 1}$ ist die *Determinante* gegeben durch $\det(A) = a_{1,1}$. Für $A \in \mathbb{R}^{n\times n}$ definieren wir rekursiv

$$\det(A) = \sum_{j=1}^{n} (-1)^{j+1} a_{1,j} \det(\text{minor}(A, 1, j))$$

1.2 Die Aufgabenstellung: SVD und SEP

1.2.1 Problemstellung

Die (reduzierte) *Singulärwertzerlegung* (**SVD**) einer Matrix $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ ist eine Faktorisierung

$$A = U\Sigma V^T$$

Die Matrix $\Sigma = \texttt{diag}([\sigma_1, \ldots, \sigma_n])$ enthält die *Singulärwerte* $0 \leq \sigma_1 \leq \ldots \leq \sigma_n$ in *aufsteigender* Reihenfolge. Die orthogonalen Matrizen $U \in \mathbb{R}^{m\times n}$ und $V \in \mathbb{R}^{n\times n}$, beschreiben die *linken* bzw. *rechten Singulärvektoren*. Ein Tripel $(\sigma_j, U(:, j), V(:, j))$ heißt j-tes *Singulärtripel* und $(U(:, j), V(:, j))$ heißt j-tes *Singulärvektorpaar*.
Gesucht beim *symmetrischen Eigenwertproblem* (kurz **SEP**) ist die Faktorisierung einer symmetrischen Matrix

$$H = Q\Lambda Q^T$$

in eine Matrix $\Lambda = \texttt{diag}([\lambda_1, \ldots, \lambda_n])$ mit *Eigenwerten* in aufsteigender Reihenfolge und eine orthogonale Matrix Q bestehend aus zugehörigen *Eigenvektoren*. Für symmetrische Matrizen ist bekannt (Spektralsatz), daß eine

solche Faktorisierung existiert, und daß ferner die Eigenwerte reell sind. Ein Tupel $(\lambda_j, Q(:,j))$ heißt j-tes *Eigenpaar*. Für $\lambda_j = \cdots = \lambda_{j+d}$ beschreibt $Q(:, j:j+d)$ den zugehörigen $(d+1)$-dimensionalen invarianten Unterraum. Aus dem Spektralsatz können wir die Existenz einer **SVD** von A folgern, indem wir folgende symmetrische Eigenwertprobleme betrachten:

1. Normalengleichungen: $A^TA = V\Sigma^2V^T$ bzw. $AA^T = U\Sigma^2U^T$
2. Jordan-Wielandt-Darstellung:

$$\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} = \left(\frac{1}{\sqrt{2}} \begin{bmatrix} U & U \\ -V & V \end{bmatrix}\right) \begin{bmatrix} -\Sigma & 0 \\ 0 & \Sigma \end{bmatrix} \left(\frac{1}{\sqrt{2}} \begin{bmatrix} U & U \\ -V & V \end{bmatrix}\right)^T$$

Definition 1.7 (Normen)
Die *p-Norm* eines Vektors $x \in \mathbb{R}^n$ definiert als

$$\|x\|_p := (|x_1|^p + \cdots + |x_n|^p)^{1/p}$$

und die *p-Norm* einer Matrix $A \in \mathbb{R}^{n\times n}$ als

$$\|A\|_p := \max_{x\in\mathbb{R}^n} \frac{\|Ax\|_p}{\|x\|_p}$$

Für $p = 2$ schreiben wir die Norm auch ohne Index. Eine weitere Matrixnorm ist die *Frobenius-Norm*

$$\|A\|_F := \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}^2}$$

Bemerkung 1.8 (Durch SVD bestimmbare Größen)
Folgende Größen lassen sich bei bekannter **SVD** bestimmen [13].

- die 2-Norm $\|A\|_2 = \sigma_n$ (auch für $m > n$)
- die Frobenius-Norm $\|A\|_F^2 = \sum_{j=1}^{n} \sigma_j^2$
- die *Konditionszahl* $cond(A) = \frac{\sigma_n}{\sigma_1}$
- der *Rang* r als die Anzahl der von Null verschiedenen Singulärwerte
- der *Kern* als der von den zu den Singulärwerten mit $\sigma_j = 0$ gehörenden rechten Singulärvektoren aufgespannte Unterraum

- Für $m = n$ ist A *regulär*, wenn $\sigma_1 > 0$ gilt, sonst *singulär*. Ist A regulär, so gilt für die *Inverse* $A^{-1} = V\Sigma^{-1}U^T$. Dabei ist $\Sigma^{-1}\Sigma = I$.

Bemerkung 1.9 (Invarianzen)
Viele **SEP**- und **SVD**-Löser basieren auf folgenden Eigenschaften:

- Eigenwerte sind invariant unter orthogonalen Ähnlichkeitstransformationen.
- Eigenvektoren sind invariant unter Translation und Inversion und demnach auch unter der *Spektraltransformation* $A_\tau = (A - \tau I)^{-1}$.
- Singulärwerte sind invariant unter orthogonalen Transformationen.

Ein weiteres klassisches Resultat über Invarianzen ist das Trägheitsgesetz von Sylvester.

Definition 1.10 (Signatur, Definitheit)
Eine symmetrische Matrix $H \in \mathbb{R}^{n\times n}$ besitze π positive Eigenwerte, ν negative Eigenwerte und ξ Eigenwerte, die Null sind. Wir bezeichnen das Tripel

$$(\pi, \nu, \xi) = \text{sign}(H)$$

als *Signatur* von H. Für eine skalare Größe h definieren wir dagegen die Signatur als Vorzeichen:

$$\text{sign}(h) = \begin{cases} 1 & \text{falls } h > 0 \\ -1 & \text{falls } h < 0 \\ 0 & \text{falls } h = 0 \end{cases}$$

Wir bezeichnen H als *positiv* bzw. *negativ definit*, wenn $\pi = n$ bzw. $\nu = n$ gilt. Ansonsten ist H *indefinit*.

Satz 1.11 (Das Trägheitsgesetz von Sylvester)
Sei H eine symmetrische und X eine reguläre Matrix. Dann haben H und XHX^T dieselbe Signatur.

Beweis: Siehe [17]. □

Definition 1.12 (Cholesky-Faktorisierung)
Ist H symmetrisch und positiv definit, so existiert eine eindeutig bestimmte untere Dreiecksmatrix L mit positiven Diagonaleinträgen, so daß $H = LL^T$. Wir bezeichnen L als *Cholesky-Faktor* [44].

1.2.2 tSEP und bSVD

In dieser Arbeit untersuchen wir insbesondere das symmetrische Eigenwertproblem für tridiagonale Matrizen (kurz **tSEP**) und die Singulärwertzerlegung von Bidiagonalmatrizen (kurz **bSVD**).
Die Beziehung zwischen beiden Problemen können wir einerseits wieder durch die Normalengleichungen beschreiben. Auf der anderen Seite ergibt eine Perfect-Shuffle-Permutation $P_{ps} = [e_2, e_4, \ \dots\ , e_{2n}, e_1, e_3, \ \dots\ , e_{2n-1}]$ der Jordan-Wielandt-Darstellung die sogennante *Golub-Kahan Matrix* [43]

$$T_{GK} = P_{ps} \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix} P_{ps}^T = \mathtt{diag}([a_1, b_1, a_2, \ \dots\ , b_{n-1}, a_n], \pm 1)$$

Ihre Diagonalelemente sind Null. Die Struktur der von Null verschiedenen Einträge kann der Abbildung 3.6 auf Seite 71 entnommen werden. Aus der Lösung des **tSEP**

$$T_{GK} = Q \Lambda Q^T$$

können wir die linken bzw. rechten Singulärvektoren gewinnen, indem wir die geraden bzw. ungeraden Zeilen von Q betrachten:

$$\sqrt{2} Q = P_{ps} \begin{bmatrix} U & U \\ -V & V \end{bmatrix}$$

Bemerkung 1.13 (Reduziertheit)
Eine Matrix heißt *reduziert*, wenn sie sich schreiben läßt als $A = \mathtt{diag}(A_1, A_2)$. In diesen Fällen können wir die **SVD** oder das **SEP** für A_1 und A_2 getrennt betrachten. Wir nehmen daher o.E. an, daß die vorliegenden Matrizen unreduziert sind. Bei der Untersuchung der **bSVD** und des **tSEP** gehen wir also davon aus, daß die Nebendiagonalelemente von T bzw. Haupt- und Nebendiagonalelemente von B (vgl. Golub-Kahan Matrix) von Null verschieden sind. Ferner sollen alle nicht-trivialen Einträge von B positiv sein. Ansonsten können wir durch Anwenden geeigneter Diagonalmatrizen, deren Elemente vom Betrag Eins sind, diese Eigenschaft herstellen.
Nach [63] sind alle Eigenwerte einer unreduzierten symmetrischen Tridiagonalmatrix paarweise verschieden. Außerdem gilt für alle Eigenvektoren, daß sämtliche Komponenten von Null verschieden sind.
Dennoch kann es vorkommen, daß *Näherungen an Eigenwerte* im Rahmen der verfügbaren Genauigkeit und aufgrund von Rundungsfehlern nicht voneinander zu unterscheiden sind. Siehe Abschnitt 1.3 sowie Beispiele 1.18 und 1.19.

Für *Näherungen an Eigenvektoren* zeigt sich ferner, daß oft viele der Komponenten „vernachlässigbar klein“ sind. Diese Erkenntnis nutzen wir in Abschnitt 4.2 aus, um sowohl die Genauigkeit als auch die Geschwindigkeit des in dieser Arbeit vorgeschlagenen **bSVD**-Lösungsverfahrens zu verbessern (Stichwort Ideale Systeme).

1.2.3 Cluster von Eigen- und Singulärwerten

Neben der absoluten Distanz $|x-y|$ zweier Skalare werden wir häufig relative Distanzen verwenden, um störungstheoretische Ergebnisse oder die Stuktur von Spektren zu beschreiben. Zu Eigenschaften und Entwicklung der Begriffe in diesem Zusammenhang siehe [58].

Definition 1.14 (Relative Distanzen)
Die *p-relativen Distanzen* ρ_p zwischen $x,\ y \in \mathbb{R}$ sind definiert als

$$\rho_p(x,y) := \frac{|x-y|}{\sqrt[p]{|x|^p + |y|^p}} \text{ für } 1 \le p \le \infty$$

In diesem Zusammenhang verwenden wir die Konvention $0/0 = 0$. Für ρ_2 schreiben wir auch ρ.

Die p-relativen Distanzen sind Metriken [4]. Für einen gewissen Schwellenwert t und ein geordnetes Zahlentupel (beispielsweise Eigen- oder Singulärwerte) können wir dann den Begriff des Clusters einführen.

Definition 1.15 (Cluster)
Sei $x = [x_1,\ \ldots, x_n] \in \mathbb{R}^n$ aufsteigend sortiert. $x(f:l)$ heißt *Cluster*, wenn $\rho_p(x_j, x_{j+1}) \le t$ für $j = f : l-1$ und $\rho_p(x_{f-1}, x_f) > t$ sowie $\rho_p(x_l, x_{l+1}) > t$ gilt. (Für $f = 1$ und $l = n$ ist je eines der beiden letzten Kriterien hinfällig.) Gehört das Element x_j zu keinem Cluster, nennen wir es *isoliert*.

Bemerkung 1.16 (Relative Distanzen und Translationen)
Relative Distanzen sind *nicht* invariant unter Translation: Seien x und y Zahlen mit $\rho(x,y) < t$. Für einen Shift-Parameter $\tau \approx x$ beobachten wir $\rho(x,y) \ll \rho(x-\tau, y-\tau)$. Beispielsweise gelten für $p = 2$ und $t = 0.01$ die Elemente von $[1.001, 1.002, 1.003]$ als geclustert. Wählen wir jedoch $\tau = 1$, so sind die Elemente von $[1.001-1, 1.002-1, 1.003-1] = [0.001, 0.002, 0.003]$ als isoliert zu betrachten.

Definition 1.17 (Lücken)
Sei $x = [x_1,\ \ldots, x_n] \in \mathbb{R}^n$. Wir definieren für $x_j \neq 0$

$$\text{absgap}(j,x) := \min_{i\neq j}\{|x_i - x_j|\} \quad \text{und} \quad \text{relgap}(j,x) := \min_{i\neq j}\{|x_i - x_j|/|x_j|\}$$

als *absolute* und *relative Lücke* (engl. *gap*).

1.2.4 Testmatrizen für bSVD und tSEP

Aus der Unreduziertheit von symmetrisch tridiagonalen Matrizen folgt, daß ihre Eigenwerte paarweise verschieden sind. Dennoch lassen sich leicht Klassen von Testmatrizen mit sehr dichten Clustern in ihren Spektren erzeugen.

Beispiel 1.18 (Wilkinson-Matrix)
Die $(2n+1)$-dimensionale symmetrisch tridiagonale Matrix

$$W_{2n+1}^{+} = \texttt{diag}([n, n-1, n-2, \ \ldots\ , n-2, n-1, n]) + \texttt{diag}([1, \ \ldots\ , 1], \pm 1)$$

gilt als klassisches Testproblem. Bei moderater Kondition ist sie unreduziert und ihre Eigenwerte sind in exakter Arithmetik damit alle verschieden. Für $n = 10$ gilt jedoch $\rho_\infty(\lambda_{20}, \lambda_{21}) \approx 6.6 \times 10^{-15}$. Erhöhen wir n, so gibt es Paare von Eigenwerten, deren relative Distanz kleiner ist als die Konstante $\epsilon = 2^{-52}$, die sogenannte Maschinengenauigkeit für doppelt genaue Zahlen, vgl. Abschnitt 1.3.2. Es kann also vorkommen, daß Eigenwerte im Rahmen einer vorgegebenen Genauigkeit nicht mehr unterscheidbar sind, obwohl die zugehörige Matrix unreduziert ist.
Wenn wir einen Shift $\tau < \lambda_1$ wählen, können wir durch eine Cholesky-Faktorisierung $B^T B = W_{2n+1}^{+} - \tau I$ eine Testmatrix für die **bSVD** erzeugen.

Beispiel 1.19 (Geleimte Matrizen)
Zu einer gegebenen symmetrisch tridiagonalen *Basismatrix* $T \in \mathbb{R}^{n \times n}$ und einem reellen Parameter g können wir eine sogenannte *k-fach geleimte* symmetrisch tridiagonale Matrix $T(n, k, g) \in \mathbb{R}^{(k+1)n \times (k+1)n}$ erzeugen, indem wir T darin als Blockdiagonale $(k+1)$-mal eintragen und die restlichen Nebendiagonalelemente auf g setzen. In Abhängigkeit von g erzeugen wir damit mehr oder weniger dichte Cluster in der Nähe der Eigenwerte von T. Analog lassen sich bidiagonale Matrizen erzeugen.

Weitere Beispiele für Testmatrizen sind in [23, 46] beschrieben. Wir gehen in Abschnitt 4.3 ausführlicher darauf ein.

1.3 Einfluß von Rechnerarchitekturen

Numerische Lösungsverfahren werden als Programme auf elektronischen Rechenanlagen (Computern) implementiert. Neben theoretischen Beurteilungen spielt also auch der Umgang mit deren Ressourcen eine Rolle.

1.3.1 Über serielle und parallele Rechner

Die Leistungsfähigkeit eines Rechners beschreiben wir mit einigen Kenngrößen. Durch den Prozessor ist die Arithmetik, also die Durchführung von Rechenoperationen, festgelegt. Die Ausführungsgeschwindigkeiten werden bestimmt durch die Anzahl von Operationen, die pro Sekunde durchgeführt werden können (Flops/s), und durch die Effizienz der Speicherzugriffe. Die theoretisch mögliche Spitzenleistung (engl. *Peak-Performance*) wird in Anwendungen i.d.R. jedoch nicht erreicht. Durch den Einsatz von auf Herstellerseite optimierten Compilern und Software-Bibliotheken kann aber oft eine zufriedenstellende Leistung des Rechners erzielt werden. Zur Zwischenablage von Daten sollte ferner Arbeitsspeicher in hinreichendem Umfang zur Verfügung stehen.
Wir sprechen von seriellen Rechnern, wenn ein Programm nur auf einem einzigen Prozessor (auch Knoten) ausgeführt wird. Sind mehrere Knoten durch ein Netzwerk verbunden, so liegt eine parallele Rechnerarchitektur vor. Der Arbeitsspeicher kann dann gemeinsam (*shared memory*) oder verteilt (*distributed memory*) realisiert sein.

1.3.2 Modell der Gleitpunktarithmetik

Neben der Ausführungsgeschwindigkeit und dem verfügbaren Speicherplatz haben alle numerischen Verfahren bei einer Implementierung als Computerprogramm das grundsätzliche Problem, daß moderne Rechnerarchitekturen in der Regel nur mit *endlicher* Genauigkeit arbeiten können.
Die nach dem *IEEE standard for binary arithmetic* [2] codierten Gleitpunktzahlen (engl. *floating point numbers*) bilden nur eine Teilmenge der reellen Zahlen ab. Somit ist jedes Ergebnis $\bar{x}$ einer Rechnung in Gleitpunktarithmetik bedingt durch Rundungsfehler nur als Näherung (Approximation) an die tatsächliche Lösung $x \in \mathbb{R}$ aufzufassen. Wir messen den *relativen Fehler* in x durch $|\frac{x-\bar{x}}{x}| = \rho_\infty(x, \bar{x})$. Für eine elementare Operation $\circ \in \{+, -, *, /\}$ bezeichnen wir mit $\mathrm{fl}(x \circ y)$ das als Gleitpunktzahl dargestellte Ergebnis von $x \circ y$. Wir verwenden in Rundungsfehleranalysen folgendes Modell, um die Rundungsfehler zu beschreiben:

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

Dabei hängen die relativen Fehler δ und η von x, y und $\circ$ sowie der arithmetischen Einheit des Rechners ab. Sie sind betragsmäßig beschränkt durch die Maschinengenauigkeit ϵ (der kleinsten positiven Gleitpunktzahl, für die $\mathrm{fl}(1 + \epsilon) \neq 1$ gilt). Im einfach bzw. doppelt genauen Format der nach *IEEE* codierten Zahlen beträgt $\epsilon = 2^{-23}$ bzw. $\epsilon = 2^{-52}$.

Oft drücken wir $|\frac{x-\bar{x}}{x}| = \kappa \cdot \epsilon$ als Vielfaches der Maschinengenauigkeit aus und sagen dann, daß sich x und $\bar{x}$ in höchstens κ ULP (engl. **U**nits in the **L**ast **P**lace) unterscheiden.
Die *IEEE*-Arithmetik verwendet ferner Symbole $\pm\infty$ and `NaN` (Not a Number), um Operationen zu beschreiben, die außerhalb des darstellbaren Zahlenbereichs liegen, und stellt ferner standardisierte Rechenregeln für solche Ausnahmen bereit [2, 41]. In diesem Zusammenhang ist ferner die Behandlung von Unter- und Überläufen (*Under-* und *Overflow*) zu beachten.

1.3.3 Stabilitätsanalyse

Die Abbildung f beschreibe eine exakte Rechnung, die durch ein numerisches Verfahren vorgeschrieben ist, um aus gegebenen Eingangsdaten x ein Resultat $f(x)$ zu bestimmen.
Bei der *Vorwärtsanalyse* wird untersucht, wie weit das durch Rundungsfehler gestörte Resultat $\bar{f}(x)$ von $f(x)$ abweicht: $x \longrightarrow f(x) \xrightarrow{VA} \bar{f}(x)$
Ziel der *Rückwärtsanalyse* ist, das berechnete Resultat als exaktes Ergebnis von (leicht) gestörten Ausgangsdaten $\bar{x}$ darzustellen: $x \xrightarrow{RA} \bar{x} \longrightarrow f(\bar{x})$
Die *gemischte Stabilitätsanalyse* kombiniert beide Ansätze. Wir kontrollieren zunächst die Störung in x, wenden dann auf $\bar{x}$ die exakte Rechnung f an und versuchen anschließend, das so gewonnene Resultat $f(\bar{x})$ als Störung von $\bar{f}(x)$ zu interpretieren: $x \xrightarrow{RA} \bar{x} \longrightarrow f(\bar{x}) \xrightarrow{VA} \bar{f}(x)$
Wir sind stark daran interessiert, anstelle von additiven Störungen kleine multiplikative Störungen (um wenige ULP) zu verwenden, um Resultate von hoher relativer Genauigkeit zu erzielen. So sind z.B. Verfahren, die nur Multiplikationen, Divisionen, Additionen positiver Zahlen sowie Subtraktionen *exakt darstellbarer* positiver Zahlen (z.B. Eingangsdaten) verwenden, mit dieser Eigenschaft ausgestattet.

1.3.4 Störungstheorie für SEP und SVD

Wir stellen nun wichtige Aussagen über die Empfindlichkeit von Eigenpaaren und Singulärtripeln auf Störungen in der Eingangsmatrix zusammen (Rückwärtsanalyse). Für Details siehe [17].
Wir betrachten jeweils für A, H und leicht gestörte Matrizen $\bar{A}, \bar{H}$

- **SEP** $H = Q\Lambda Q^T$ und $\bar{H} = \bar{Q}\bar{\Lambda}\bar{Q}^T$
- **SVD** $A = U\Sigma V^T$ und $\bar{A} = \bar{U}\bar{\Sigma}\bar{V}^T$

Ferner seien jeweils die spitzen Winkel zwischen Vektoren gegeben:

$$\begin{aligned}\theta_j^Q &= \angle(Q(:,j), \bar{Q}(:,j)) \\ \theta_j^U &= \angle(U(:,j), \bar{U}(:,j)) \\ \theta_j^V &= \angle(V(:,j), \bar{V}(:,j))\end{aligned}$$

Satz 1.20 (Weyl, Variation des $\sin\theta$-Theorems, absolute Fehler)
Sei $\bar{H} = H + E$. Dann gilt für $j = 1 : n$

$$|\lambda_j - \bar{\lambda}_j| \leq \|E\|$$

sowie

$$\frac{1}{2}\sin(2\theta_j^Q) \leq \frac{\|E\|}{\text{absgap}(j, \Lambda)}$$

und

$$\frac{1}{2}\sin(2\theta_j^Q) \leq \frac{\|E\|}{\text{absgap}(j, \bar{\Lambda})}$$

Folgerung 1.21 (Übertragung auf SVD)
Sei $\bar{A} = A + F$. Dann gilt für $j = 1 : n$

$$|\sigma_j - \bar{\sigma}_j| \leq \|F\|$$

sowie

$$\frac{1}{\sqrt{2}}\max\{\sin(2\theta_j^U), \sin(2\theta_j^V)\} \leq \frac{\|F\|}{\text{absgap}(j, \Sigma)}$$

und

$$\frac{1}{\sqrt{2}}\max\{\sin(2\theta_j^U), \sin(2\theta_j^V)\} \leq \frac{\|F\|}{\text{absgap}(j, \bar{\Sigma})}$$

Beweis: Klar durch Anwenden des vorigen Satzes auf die Jordan-Wielandt-Darstellungen von A und $\bar{A}$. □

Die bisherigen Abschätzungen gehen von additiven Störungen aus. Vor allem für die kleinsten Eigen- und Singulärwerte kann dies bedeuten, daß die exakten und die gestörten Resultate nur auf wenigen Nachkommastellen übereinstimmen. (Klein bedeutet in diesem Zusammenhang $\lambda_j \approx \epsilon\lambda_n$.) Bei additiven Störungen hängt ferner die Empfindlichkeit der Vektoren gegenüber Störungen stark von den *absoluten* Lücken im Spektrum ab. Diese Problematik wird beispielsweise bei der *Inversen Iteration* deutlich, vgl. Abschnitt 1.5.5.

Beispiel 1.22

- Wir betrachten $H = \frac{1}{2}\begin{bmatrix} 1+\epsilon & 1-\epsilon \\ 1-\epsilon & 1+\epsilon \end{bmatrix}$ und $E = \frac{\epsilon}{2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$ als additive Störung mit $\|E\| = \epsilon$. Der kleinere Eigenwert von H ist $\lambda_1 = \epsilon$. Für den kleineren Eigenwert der gestörten Matrix $\bar{H} = H + E$ gilt aber $\bar{\lambda}_1 = 0$. Im Extremfall stimmen also keine signifikanten Stellen der Eigenwerte mehr überein.
- Siehe [63]. Wir betrachten $H(t) = \begin{bmatrix} 1 + t\cos\frac{2}{t} & t\sin\frac{2}{t} \\ t\sin\frac{2}{t} & 1 - t\cos\frac{2}{t} \end{bmatrix}$ mit Eigenwerten $1-t$ und $1+t$ und $Q(t) = \begin{bmatrix} \cos\frac{1}{t} & \sin\frac{1}{t} \\ \sin\frac{1}{t} & -\cos\frac{1}{t} \end{bmatrix}$. Die Untersuchung der die Eigenvektoren beschreibenden Matrix $Q(t)$ zeigt für $t \to 0$, daß bei kleinen Variationen δ_t beliebig große Winkel zwischen $Q(t)$ und $Q(t+\delta_t)$ auftreten können, während die Störung $E = H(t+\delta_t) - H(t)$ klein bleibt.

Im folgenden Satz stellen wir einen multiplikativen Störungsansatz vor. Dieser erlaubt es, die Empfindlichkeit der Vektoren in Abhängigkeit der *relativen* Lücken zu beschreiben sowie die *relativen* Fehler der Eigen- und Singulärwerte zu beschränken [17, 59].

Satz 1.23 (Variation des $\sin\theta$-Theorems, relative Fehler)
Sei $\bar{H} = X^T H X$ und $\epsilon_1 = \|I - (XX^T)^{-1}\|$. Dann gilt für $j = 1:n$

$$|\lambda_j - \bar{\lambda}_j| \leq |\lambda_j| \|X^T X - I\|$$

sowie

$$\frac{1}{2}\sin(2\theta_j^Q) \leq \frac{\epsilon_1}{1-\epsilon_1} \cdot \frac{1}{\operatorname{relgap}(j,\Lambda)} + \|X - I\|$$

und

$$\frac{1}{2}\sin(2\theta_j^Q) \leq \frac{\epsilon_1}{1-\epsilon_1} \cdot \frac{1}{\operatorname{relgap}(j,\bar{\Lambda})} + \|X - I\|$$

Folgerung 1.24 (Übertragung auf SVD)
Sei $\bar{A} = Y^T H Z$ sowie $\epsilon_1 = \max\{\|I - (YY^T)^{-1}\|, \|I - (ZZ^T)^{-1}\|\}$ und $\epsilon_2 = \max\{\|X - I\|, \|Y - I\|\}$. Dann gilt für $j = 1:n$

$$|\sigma_j - \bar{\sigma}_j| \leq |\sigma_j| \max\{\|Y^T Y - I\|, \|Z^T Z - I\|\}$$

sowie

$$\frac{1}{\sqrt{2}}\max\{\sin(2\theta_j^U), \sin(2\theta_j^V)\} \leq \frac{\epsilon_1}{1-\epsilon_1} \cdot \frac{1}{\operatorname{relgap}(j,\Sigma)} + \epsilon_2$$

und

$$\frac{1}{\sqrt{2}}\max\{\sin(2\theta_j^U), \sin(2\theta_j^V)\} \leq \frac{\epsilon_1}{1-\epsilon_1} \cdot \frac{1}{\operatorname{relgap}(j,\bar{\Sigma})} + \epsilon_2$$

Beweis: Verwende den vorigen Satz mit $X = \texttt{diag}(Y, Z)$. □

Verallgemeinerungen auf höherdimensionale invariante Unterräume gelten sinngemäß (siehe [59] sowie die dort aufgeführten Referenzen).
Lösungsverfahren für **SEP** und **SVD**, die eine Rückwärtsanalyse mit einem multiplikativen Störungsansatz erlauben, bestimmen demnach die gesuchten Größen auf höhere Genauigkeit. So können beispielsweise auch die kleinsten Singulärwerte (oder Eigenwerte) mit einer Abweichung von wenigen ULP angenähert werden. Insbesondere wird die Null immer exakt bestimmt.
Es stellt sich nun die Frage, für welche Klassen von Matrizen eine relative Störungstheorie wie in Satz 1.23 möglich ist und ob darauf aufbauend Lösungsverfahren entwickelt werden können. Es ist also nach einer geeigneten Darstellung (Repräsentation) der Matrix gesucht [27, 29]:

Definition 1.25 (Relativ robuste Repräsentation (RRR))
Eine *relativ robuste Repräsentation* (engl. *relatively robust representation*, kurz *RRR*) ist eine Darstellung einer Matrix A durch eine Menge von Zahlen $\mathcal{A} = \{\alpha_i\}$ in einer Weise, daß kleine multiplikative Störungen $\alpha_i(1 + \kappa_i\epsilon)$ nur zu kleinen multiplikativen Störungen der gesuchten Elemente der Spektralzerlegung oder **SVD** von A führen.

Beispiel 1.26

1. Die Einträge einer Bidiagonalmatrix B, also die Darstellung $\mathcal{A} = \{a_1, \ \ldots \ , a_n, b_1, \ \ldots \ , b_{n-1}\}$, bilden eine RRR für alle Singulärwerte und -vektoren [21]. Lösungsansätze wie das QR-Verfahren, das QD-Verfahren [38] und das Bisektionsverfahren [37] nutzen dies aus. Die beiden letztgenannten Verfahren bestimmen nur Singulärwerte.

2. Ist die das **tSEP** definierende Matrix T als B^TB, BB^T oder durch die Golub-Kahan Matrix gegeben, so bestimmt die o.a. Repräsentation $\mathcal{A}$ naturgemäß alle Eigenwerte (und entsprechend auch Eigenvektoren) auf hohe relative Genauigkeit.

3. Ist T jedoch explizit durch ihre Einträge auf der Haupt- und Nebendiagonalen gegeben, so bilden diese Einträge im allgemeinen *keine* RRR. Siehe Beispiel 1.22.

4. Eine Matrix muß nicht unbedingt Bi- oder Tridiagonalgestalt aufweisen, um eine RRR zu besitzen. Siehe auch [19, 22, 24].

5. Manchmal ist es hinreichend, daß $\mathcal{A}$ nicht das gesamte Spektrum, sondern nur Teile davon auf hohe relative Genauigkeit bestimmt.

Der RRR-Algorithmus macht starken Gebrauch von solchen *partiellen RRR*. Es gibt *a priori* und *a posteriori* Kriterien, die wir in Abschnitt 2.4 diskutieren. Siehe auch [27, 29].

Es erhebt sich die Frage nach *Abschätzungen* für die Fehler der Eigen- oder Singulärwerte, wenn die Menge $\mathcal{A}$ durch kleine multiplikative Terme modifiziert wird. Einerseits hängen diese Fehler naturgemäß von den Störungen der Menge $\mathcal{A}$ ab. Für konkrete Schranken benötigen wir aber zusätzliche Eigenschaften der jeweiligen Matrizen. Beispielsweise wird in [21] folgende Aussage gezeigt:

Lemma 1.27 (Fehlerschranken für die Singulärwerte von B)
Sei mit B eine Bidiagonalmatrix und eine gestörte Variante mit

$$\begin{aligned}\widehat{B} &= \texttt{diag}([a_1(1+\kappa_1\epsilon),\ \ldots\ , a_n(1+\kappa_n\epsilon)]) + \\ &\quad\ \texttt{diag}([b_1(1+\kappa_{n+1}\epsilon),\ \ldots\ , b_{n-1}(1+\kappa_{2n-1}\epsilon)], 1)\end{aligned}$$

gegeben. Sei $\bar{\kappa} := \prod_{i=1}^{2n-1} \max\{|(1+\kappa_i\epsilon)|, |(1+\kappa_i\epsilon)|^{-1}\}$. Für die Singulärwerte $\hat{\sigma}_j$ von $\widehat{B}$ gilt dann

$$\frac{\sigma_j}{\bar{\kappa}} \leq \hat{\sigma}_j \leq \sigma_j \bar{\kappa}$$

Sind also z.B. alle $|\kappa_i| \leq 1$, kann demnach kein Singulärwert um mehr als einen Faktor $\bar{\kappa} = (1-\epsilon)^{1-2n}$ gestört werden.

Die Eigenschaft einer Matrix, eine RRR zu besitzen, kann sowohl bei störungstheoretischen Überlegungen (z.B. Sätze 3.1, 3.14 und 3.21) als auch bei der praktischen Entwicklung von Algorithmen vorteilhaft ausgenutzt werden. Einen Überblick zur relativen Störungstheorie finden wir ferner in [53].

Bemerkung 1.28 (Partielle RRR für indefinite Matrizen)
Im Rahmen dieser Arbeit werden wir für die o.g. Sätze *voraussetzen*, daß gewisse indefinite, als LDL^T-Faktorisierung gegebene Matrizen eine partielle RRR bilden. Wir verzichten dabei darauf, genaue Abschätzungen für die Fehler der Eigenwerte anzugeben. Wichtig ist nur, daß bei einem Übergang

$$D, L \xrightarrow{\text{Störung}} \widehat{D}, \widehat{L}$$

der Fehler kontrollierbar bleibt, d.h. $\lambda_j = \widehat{\lambda}_j(1+\kappa\epsilon)$ für $j = f:l$ mit einem *moderaten* Wert für κ.

1.3.5 Anforderungen an die Genauigkeit

Bei Implementierungen von numerischen Lösungsverfahren in der Gleitpunktarithmetik akzeptieren wir Approximationen als hinreichend genau, wenn folgende Kriterien erfüllt sind:

SEP

- Numerische Orthogonalität, d.h. $\bar{Q}(:,i)^T\bar{Q}(:,j) = \mathcal{O}(n\epsilon)$ für $i \neq j$
- Kleine Residuen bzgl. H, d.h. $\|H\bar{Q}(:,j) - \bar{\lambda}_j\bar{Q}(:,j)\| = \mathcal{O}(\|H\|n\epsilon)$

SVD

- Numerische Orthogonalität, d.h. $\bar{U}(:,i)^T\bar{U}(:,j) = \mathcal{O}(n\epsilon)$ und $\bar{V}(:,i)^T\bar{V}(:,j) = \mathcal{O}(n\epsilon)$ für $i \neq j$
- Kleine Residuen bzgl. A, d.h. $\|A\bar{V}(:,j) - \bar{\sigma}_j\bar{U}(:,j)\| = \mathcal{O}(\|A\|n\epsilon)$. Wir bezeichnen Näherungen an Singulärvektorpaare in Abhängigkeit von diesem Maß als gut oder schlecht *gekoppelt*.

Diese Forderungen sind i.A. bestmöglich. Typischerweise treten bereits beim Abspeichern der Komponenten exakter Eigenvektoren als Gleitpunktzahlen durch Rundungsfehler bedingt die beschriebenen Abweichungen von der Orthogonalität auf. Auch die Auswertung des Skalarprodukts in Gleitpunktarithmetik kann nur mit dieser Genauigkeit gestaltet werden. Bei den Residuen begrenzen die Matrix-Normen die Skalierungen mit den Eigen- bzw. Singulärwerten.

Bemerkung 1.29 ($\mathcal{O}$-Notation)
Zu einer gegebenen Funktion f steht $\mathcal{O}(f)$ für eine gewisse Funktion g mit $|g| \leq c|f|$, wobei c eine (möglichst kleine) Konstante ist. Vernachlässigbare Terme werden also hinter der $\mathcal{O}$-Notation versteckt, um Rechnungen zu vereinfachen und das Augenmerk auf die wesentlichen Bestandteile eines Sachverhalts zu lenken. Wir machen von dieser Notation Gebrauch, um die Abweichung von Näherungen an exakte Werte zu beschreiben und um Aussagen über die Komplexität eines Verfahrens zu machen.

Ein naives Vorgehen

Aufgrund der nahen mathematischen Verwandtschaft zum **SEP** drängt sich die nachfolgend beschriebene Strategie zur Bestimmung der **SVD** auf:

- Berechne $H = A^T A$.
- Verwende ein Lösungsverfahren für die Zerlegung $H = V \Lambda V^T$.
- Bestimme Σ aus den Quadratwurzeln der Diagonalelemente von Λ.
- Bestimme U durch Lösen von $U\Sigma = AV$.

Dieses Vorgehen erweist sich aber als numerisch nicht praktikabel, da allein durch die explizite Bildung von $H = A^T A$ entscheidende Informationen über A verloren gehen können:

Beispiel 1.30
Für $\eta = \epsilon/2$ ist $A = \begin{bmatrix} 1 & 1 \\ 0 & \eta \end{bmatrix}$ regulär, jedoch $\mathrm{fl}(A^T A) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ — also das in Gleitpunktarithmetik berechnete Produkt — singulär. Somit kann keine hohe relative Genauigkeit für den kleineren Singulärwert erzielt werden.

Für die weitere Diskussion des o.g. Vorgehens nehmen wir an, daß die Auswertung in Gleitpunktarithmetik durch $\mathrm{fl}(A^T A) = \bar{A}^T \bar{A}$ mit $\bar{A} = (A + E)$ beschrieben werden kann. Die Symmetrie ausnutzend berechnen wir beispielsweise nur die Elemente des oberen Dreiecks von $A^T A$. Dann ergibt sich für die Eigenwerte

$$|\lambda_j - \bar{\lambda}_j| \leq \|\mathrm{fl}(A^T A) - A^T A\| = \|A^T E + E^T A + E^T E\| = \mathcal{O}(\|A\|\|E\|)$$

und damit für die Singulärwerte

$$|\sigma_j - \bar{\sigma}_j| \approx \left| \frac{\lambda_j - \bar{\lambda}_j}{\sigma_j} \right| = \mathcal{O}(\|A\|\|E\|/\sigma_j)\,,$$

also eine Näherung, die im Vergleich zu Folgerung 1.21 um einen Faktor $\|A\|/\sigma_j$ schlechter sein kann. Hiervon sind vor allem die kleinsten Singulärwerte stark betroffen. Zur Bestimmung der **SVD** sollte also immer mit der Eingangsmatrix A und nie mit explizit ausmultiplizierten Produkten $A^T A$ oder AA^T gearbeitet werden. Daher ist auch nicht jedes Lösungsverfahren für **SEP** ohne eingehende Modifikationen geeignet, um gute Näherungen an eine **SVD** zu erhalten.
Bei der Bestimmung der linken Singulärvektoren kann ferner das Lösen von $U\Sigma = AV$ in der Gleitpunktarithmetik zu großen Fehlern führen, wenn die Konditionszahl von A hoch ist. Insbesondere kann die Näherung an U sehr weit von numerischer Orthogonalität entfernt sein.
Im nächsten Abschnitt stellen wir weitere Varianten des oben skizzierten Ansatzes vor, zeigen aber gleichzeitig mögliche Problemfälle auf.

1.3.6 Probleme für SEP-Löser bei Normalengleichungen und der Jordan-Wielandt-Darstellung

Ein wichtiges Merkmal von numerischen Lösungsverfahren für **SEP** und **SVD** bezieht sich darauf, wie Näherungen an Unterräume zu geclusterten Eigen- und Singulärwerten aufgestellt werden. Aus der Störungstheorie wissen wir, daß diese Unterräume selbst gut bestimmbar sind [59]. Die Herausforderung für Lösungsverfahren liegt jetzt darin, möglichst effizient *eine* Orthonormalbasis zu diesen Unterräumen anzunähern. Sei im Folgenden eine Rotation R gegeben durch

$$R = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \quad \text{mit } c^2 + s^2 = 1$$

Beispiel 1.31 (Bisektoren)
Die Matrix H besitze Eigenwerte $\lambda_{j+1} = \lambda_j(1+\epsilon)$. Sei $\mathfrak{Q}$ der zugehörige invariante Unterraum. Eine gute Näherung $\{\bar{Q}(:,j), \bar{Q}(:,j+1)\}$ an eine *beliebige* Orthonormalbasis von $\mathfrak{Q}$ erfüllt dann bereits die Genauigkeitsanforderungen bzgl. des Residuums. Ebenso gut könnten wir aber auch die aus den zugehörigen *Bisektoren*

$$\frac{\bar{Q}(:,j) \pm \bar{Q}(:,j+1)}{\sqrt{2}}$$

gebildete Näherung an $\mathfrak{Q}$ verwenden, beispielsweise um die Anforderungen an die numerische Orthogonalität zu verbessern [64].

Beispiel 1.32 (Probleme bei Normalengleichungen)
Wir nehmen an, daß ein **SEP**-Lösungsverfahren Näherungen bestimmen kann, ohne die Produkte aus A und A^T explizit zu berechnen (z.B. Iterative Methoden, Inverse Iteration, RRR-Algorithmus):

- Verwende ein Lösungsverfahren für die Zerlegung $A^TA = V\Lambda V^T$.
- Verwende ein Lösungsverfahren für die Zerlegung $AA^T = U\Lambda U^T$.
- Bestimme Σ aus den Quadratwurzeln der Diagonalelemente von Λ.

A besitze einen Cluster $\sigma_f \approx \ldots \approx \sigma_l$. Wir nehmen ferner an, daß $\bar{U} = U$ korrekt approximiert wird, während $\bar{V}$ von V in den Spalten f bis l abweicht: $\bar{V}(:, f:l) = V(:, f:l) \cdot R'$, wobei R' eine orthogonale Matrix ist, die beispielsweise durch das Akkumulieren einzelner Rotationen oder durch den Einsatz von Bisektoren entstanden ist. Während die Genauigkeitsanforderungen an beide Eigenwertprobleme erfüllt sind, ist die Interpretation der Näherungen für die **SVD** unbefriedigend: $U^TA\bar{V} = \Sigma \cdot \texttt{diag}(I_{f-1}, R', I_{n-l})$. Abbildung 1.3 versinnbildlicht die Problematik bei mehreren Clustern.

Abbildung 1.3: Typisches Resultat bei getrennter Anwendung eines **SEP**-Lösers auf die Normalengleichungen. Für eine Matrix A, die drei enge Cluster mit je vier Singulärwerten aufweist, wäre das erwünschte Ergebnis für $\bar{U}^T A \bar{V}$ eine Diagonalmatrix ($\approx \Sigma$, links). Da die Basen aber ohne direkten Bezug zueinander aufgestellt werden, sind sie oft schlecht gekoppelt (rechts).

Beispiel 1.33 (Probleme bei der Jordan-Wielandt-Darstellung)
Dieses Beispiel zeigt, daß auch eine hinreichend genaue Näherungslösung für das **SEP** einer Jordan-Wielandt-Darstellung u.U. keine befriedigende Approximation für die entsprechende **SVD** ergeben muß.
Wir suchen eine Näherungslösung zur **SVD** $A = U\Sigma V^T$, indem wir das **SEP** $\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} = Q\Lambda Q^T$ betrachten. (Die Matrix Q setzt sich dabei aus U und V zusammen.) Mit $\lambda_{j+1} = \lambda_j(1+\epsilon)$ sei ein eng geclustertes Paar von Eigenwerten gegeben. Eine Näherung $\bar{Q}(:, j:j+1) = R' \cdot Q(:, j:j+1)$ sei beschrieben durch eine Rotation R', die lediglich auf die k_u-te und k_v-te Zeile wirkt. Dabei seien $k_u \leq n$ und $k_v > n$ so gewählt, daß jeweils eine Zeile von U und V betroffen ist.

$$\begin{pmatrix} \bar{q}_{k_u,k} \\ \bar{q}_{k_v,k} \end{pmatrix} = R \cdot \begin{pmatrix} q_{k_u,k} \\ q_{k_v,k} \end{pmatrix} \quad \text{für } k = j : j+1$$

Während $\bar{Q}$ die Genauigkeitsanforderungen für das **SEP** gut erfüllt, stellen wir fest, daß für die extrahierten Lösungen $\bar{U}$ und $\bar{V}$ zwar immer noch kleine Residuen bzgl. A vorliegen, jedoch starke Abweichungen von der numerischen Orthogonalität auftreten können.
Ein Zahlenbeispiel:

$$Q(:, j:j+1) = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 1 \\ -1 & 1 \end{bmatrix} \quad R' = \begin{bmatrix} 1 & & & \\ & c & s & \\ & -s & c & \\ & & & 1 \end{bmatrix}$$

ergibt

$$\bar{U} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ c+s & s-c \end{bmatrix} \quad \bar{V} = \frac{1}{\sqrt{2}}\begin{bmatrix} c-s & s+c \\ -1 & 1 \end{bmatrix}$$

mit $\bar{U}^T\bar{U} = \begin{bmatrix} 1+cs & s^2 \\ s^2 & 1-cs \end{bmatrix}$ und $\bar{V}^T\bar{V} = \begin{bmatrix} 1-cs & -s^2 \\ -s^2 & 1+cs \end{bmatrix}$.

Wir halten als Ergebnis fest:

1. Bei Normalengleichungen kann sich das *Residuum* unserer Kontrolle entziehen.
2. Bei der Jordan-Wielandt-Darstellung ist die numerische *Orthogonalität* gefährdet.

1.4 Lösungsverfahren

Nach der allgemeinen Beschreibung der Aufgabenstellung **SVD** und **SEP** sowie der Darstellung der durch Rechnerarchitekturen gegebenen Rahmenbedingungen sollen nun die gängigen numerischen Lösungsverfahren kurz skizziert werden. Es gibt eine große Auswahl von verschiedenen Ansätzen. Welche Methode am besten geeignet ist, hängt von der Anforderungsschnittstelle („Was ist gegeben?“ – „Was ist gesucht?“) ab.

- Eingangsdaten: Haben wir aus der Problemstellung heraus Informationen über das Spektrum? Können wir die Struktur der Matrix (vollbesetzt, dünn besetzt, Bandstruktur) ausnutzen?
- Quantität der Ergebnisse: Ist das gesamte Spektrum gefragt oder nur Teile davon? Benötigen wir lediglich Approximationen an Eigen- oder Singulärwerte oder sind auch Näherungen an die zugehörigen Vektoren zu bestimmen?
- Qualität der Ergebnisse: Genauigkeit, Aufwand, d.h. Geschwindigkeit und Speicherbedarf (→ Parallelisierbarkeit?)
- Verfügbarkeit, Zuverlässigkeit, Wartbarkeit und einfache Benutzung von Software-Implementierungen

Aus Anwendersicht kommt dem letzten Punkt besondere Bedeutung zu, da die Aufgaben **SVD** und **SEP** oft als Teilprobleme größerer Anwendungen auftreten.
Im Folgenden geben wir einen kurzen Überblick der gängigen Verfahren. Wir skizzieren dabei jeweils die Idee und versuchen eine Beurteilung bzgl. der oben genannten Kriterien, soweit dies im Rahmen dieser Arbeit möglich ist. Ausführlicher werden wir auf die Verfahren mit Vorverarbeitung eingehen, vgl. Abschnitte 1.4.4 und 1.5.

Viele der hier vorgestellten Methoden zielen in erster Linie auf die Lösung des **SEP** ab. Wie in Abschnitt 1.3.6 erwähnt, erfordert eine Übertragung auf die **SVD** oft zusätzlichen algorithmischen Aufwand. Dieser führt teilweise zu Varianten, die sich vom **SEP**-Löser substantiell unterscheiden.

1.4.1 Iterative Methoden

Obwohl alle Lösungsverfahren grundsätzlich iterativer Natur sind, wird in der Literatur der Begriff *Iterative Methoden* verwendet, wenn die nachgefragte Information über das Spektrum ermittelt wird, ohne die Matrizen A oder H explizit aufzustellen und abzuspeichern. Stattdessen ist in der Regel nur die Wirkung der Matrix H auf einen beliebigen Vektor x, also $y = Hx$ bekannt. Software-Implementierungen arbeiten meistens mit einem sogenannten *Reverse Communication Interface*, bei dem der Benutzer die Wirkung $y = Hx$ spezifizieren muß. Iterative Methoden eignen sich, wenn bei sehr großer Dimension n die Matrix mit dünner Struktur besetzt ist und nur Teile des Spektrums angenähert werden sollen.
Eine Vorgehensweise für das **SEP** ist der Lanczos-Algorithmus. Durch Aufbauen von geeigneten Krylov-Unterräumen wird in k Schritten eine partielle Reduktion auf symmetrisch tridiagonale Form erzeugt:

$$T_{\text{part}} = Q^T H Q, \quad Q = [Q_k, Q_{n-k}]$$

Hier ist $T = Q_k^T H Q_k$ symmetrisch tridiagonal; die übrigen Blöcke von T_{part} sind i.A. beliebig besetzt. Die Lösung $[\Lambda, V]$ des **tSEP** $T = V \Lambda V^T$ mit den sogenannten *Ritz-Werten* Λ liefert zusammen mit den *Ritz-Vektoren* $Q_k V$ Näherungen an Eigenpaare von H. Als Probleme treten die Notwendigkeit expliziter Reorthogonalisierung sowie das Phänomen der „Fehlkonvergenz“ auf [17].
In der Regel müssen der Lanczos-Algorithmus oder andere Standard-Verfahren, die z.B. in der ARPACK-Bibliothek [3] implementiert vorliegen, beim Auftreten von Problemen durch Einsatz von Präkonditionierungstechniken oder durch Einbetten von aus der Problemstellung heraus bekannten Zusatzinformationen angepaßt und optimiert werden.
Die Übertragung auf die **SVD** stützt sich meistens auf das Lösen der entsprechenden Normalengleichungen oder der Jordan-Wielandt-Darstellung. So wird im einfachsten Fall bei der Spezifikation für das Reverse Communication Interface als Wirkung entweder $y = A^T x'$ mit $x' = Ax$ oder $y = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} x$ verwendet. Bei den Normalengleichungen erhöht sich also wieder die Konditionszahl.

Obwohl hier nur sehr oberflächlich behandelt, kommt den iterativen Methoden eine hohe Bedeutung zu, da sie bei den nachgefragten Problemgrößen und der verfügbaren Rechnerkapazität oft die einzig praktikable Möglichkeit sind, überhaupt Informationen über das Spektrum zu erhalten. Für eine gegebene Anwendung können häufig sehr effiziente Varianten der Standard-Lösungsverfahren entwickelt werden.

1.4.2 Jacobi-Verfahren

Liegt die Matrix H oder A explizit im Speicher vor, unterscheiden wir zwischen Verfahren, die den iterativen Prozeß sofort auf der dicht besetzten Eingangsmatrix aufsetzen und Ansätzen, die in einer Vorverarbeitungsphase zunächst eine Reduktion auf spezielle Gestalt (Tridiagonal- oder Bidiagonalmatrix) durchführen und danach mit der Diagonalisierungsphase beginnen.

Zu den erstgenannten gehört das Jacobi-Verfahren. Wir beschreiben zunächst die Vorgehensweise für das **SEP**. Mit dem Ziel, die Eingangsmatrix zu diagonalisieren — also der Bestimmung des vollständigen Spektrums — werden gewisse links- und rechtsseitige Rotationen [44] ausgeführt. Solche orthogonalen Transformationen — auch Jacobi-Rotationen genannt — wirken jeweils nur auf vier Elemente $h_{i,i}, h_{i,j}, h_{j,i}$ und $h_{j,j}$ der Matrix. Sie sind derart gewählt, daß sie das **SEP** für die so gebildete 2×2-Submatrix lösen. Durch wiederholte Anwendung solcher Rotationen auf geeignete Teilmatrizen erfolgt Konvergenz gegen eine Diagonalmatrix.

Zur Bestimmung der **SVD** lassen sich zwei Strategien verfolgen: *Zweiseitige* Jacobi-Verfahren diagonalisieren die Eingangsmatrix A, indem wechselseitig Rotationen bestimmt werden, die eine **SVD** einer ausgewählten 2×2-Submatrix bewirken. Der alternative Ansatz verwendet wieder Normalengleichungen, bei denen Jacobi-Rotationen nur *einseitig* auf A ausgeführt werden.

In Rahmen der Untersuchungen, welche Verfahren Näherungslösungen von hoher relativer Genauigkeit erzeugen, kann für gewisse Klassen von Testproblemen nachgewiesen werden, daß das Jacobi-Verfahren solche Kriterien erfüllt [24]. Diese Testprobleme sind also weitere Beispielklassen für relativ robuste Repräsentationen, vgl. Definition 1.25. Um der aufgrund des Konvergenzverhaltens vergleichsweise hohen Komplexität und den damit verbundenen großen Ausführungszeiten zu begegnen, können Blockungsstrategien und effiziente parallele Implementierungen [5, 6] verwendet werden.

1.4.3 Spectral Divide and Conquer

Bei diesem Ansatz wird versucht, eine Zerlegung

$$[Q_1, Q_2]^T H [Q_1, Q_2] = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix}$$

einer dicht besetzten Matrix zu generieren und dann rekursiv die so gewonnenen Teilprobleme H_1 und H_2 zu betrachten. Eine wie oben skizzierte Zerlegung erfolgt (abstrahierend von Detailfragen) i.d.R. nach folgendem Schema [7]:

Skalierung: Bestimme $X = \alpha H + \beta I$, so daß die Eigenwerte von X im Intervall $[0, 1]$ liegen.

Glättung: Bestimme eine Folge analytischer Funktionen $f_i(x)$, $i = 1, 2, \ldots$ (typischerweise Polynome), so daß

$$\lim_{i \to \infty} (f_i \circ f_{i-1} \circ \ldots \circ f_1)([0, 1]) = \{0, 1\}$$

d.h. daß im Grenzwert alle Werte aus dem Intervall $[0, 1]$ entweder auf 0 oder auf 1 abgebildet werden. Führe die Iteration

$$Y_0 = X, \quad Y_i = f_i(Y_{i-1})$$

aus, bis alle Eigenwerte von Y_k „nahe bei“ 0 oder 1 liegen.

Berechnung invarianter Unterräume: Nachdem das Spektrum von Y_k in Eigenwerte „nahe bei“ 0 und 1 aufgeteilt wurde, bestimme orthogonale Basen Q_1 und Q_2 für die jeweiligen invarianten Unterräume: $\operatorname{Bild} Y_k Q_1 = \operatorname{Bild} Q_1$ und $Q_1^T Q_2 = 0$.

Zerlegung: Wir können nun ausnutzen, daß die Eigenvektoren invariant unter der Transformation $H \to Y_k$ bleiben. Mit Hilfe der Basen Q_1 und Q_2 läßt sich also die angestrebte Zerlegung von H erreichen.

Bei der Konkretisierung dieses Schemas lassen sich nun verschiedene Strategien zur Lösung der aufgelisteten Teilaufgaben einsetzen und kombinieren. Durch geschickte Auswahl der f_i können wir beispielsweise gezielt nach Teilen des Spektrums suchen. Neben den f_i beeinflußt auch das Konvergenzkriterium die Genauigkeit der Lösungen sowie die Ausführungszeiten. Ein einzelner Iterationsschritt wird dominiert durch den Aufwand für dicht besetzte Matrix-Matrix-Multiplikationen, die aber unter Einsatz hochoptimierter Bibliotheksroutinen (Basic Linear Algebra Subprograms, kurz

BLAS [33]) sehr effizient ausgeführt werden können. Parallelisierbarkeit ist sowohl durch den Divide-and-Conquer Ansatz (die Teilprobleme H_1 und H_2 können unabhängig auf verschiedenen Prozessoren bearbeitet werden) als auch durch effiziente Verteilbarkeit der Matrixoperationen (Parallel BLAS, kurz PBLAS [11]) gegeben.
Verallgemeinerungen auf das unsymmetrische Eigenwertproblem sind in [20] und den dort zitierten Referenzen zusammengefaßt. Eine Übertragung auf die **SVD**, die eine Alternative zum Lösen der Normalengleichungen bzw. der Jordan-Wielandt-Darstellung bietet, ist noch offen.

1.4.4 Verfahren mit Vorverarbeitung

Während Jacobi-Verfahren und Spectral Divide-and-Conquer Ansätze den iterativen Prozeß auf der vollbesetzten Eingangsmatrix beginnen, verfolgen die Verfahren mit Vorverarbeitung zunächst die Strategie, die gegebene Matrix auf symmetrisch tridiagonale bzw. bidiagonale Form zu reduzieren:

$$H = Q_1 T Q_1^T \quad \text{bzw.} \quad A = U_1 B V_1^T$$

Dies geschieht mit Hilfe direkter Methoden durch Anwenden geeigneter Orthogonaltransformationen. Direkte Methoden haben unabhängig von den Daten der Eingangsmatrizen einen Aufwand von $\mathcal{O}(n^2 b)$, wobei b die Bandbreite der zu reduzierenden Matrix ist. Anschließend wenden wir iterative Lösungsverfahren für das **tSEP** und die **bSVD** an

$$T = Q_2 \Lambda Q_2^T \quad \text{bzw.} \quad B = U_2 \Sigma V_2^T$$

und erhalten somit Näherungen für das **SEP** und die **SVD** durch Rück-Akkumulation

$$H = Q_1 Q_2 \Lambda (Q_1 Q_2)^T \quad \text{bzw.} \quad A = U_1 U_2 \Sigma (V_1 V_2)^T$$

Wir reduzieren also in der Vorverarbeitungsphase die Information von $\mathcal{O}(nb)$ Daten auf Matrizen, die nur noch $\mathcal{O}(n)$ nicht-triviale Einträge haben. Verwenden wir diese kompakten Darstellungen in der Diagonalisierungsphase, so ergeben sich einerseits Vorteile bzgl. Geschwindigkeit und Speicherbedarf. Außerdem gibt es Verfahren, die ausgehend von Bidiagonal- und speziellen Tridiagonalmatrizen Approximationen von hoher relativer Genauigkeit erzeugen können (Stichwort RRR). Zudem liegen mit der LAPACK-Bibliothek (Linear Algebra PACKage [1, 57]) für serielle und der ScaLAPACK-Bibliothek [10] für parallele Rechner sehr effiziente und stabile Software-Implementierungen vor.

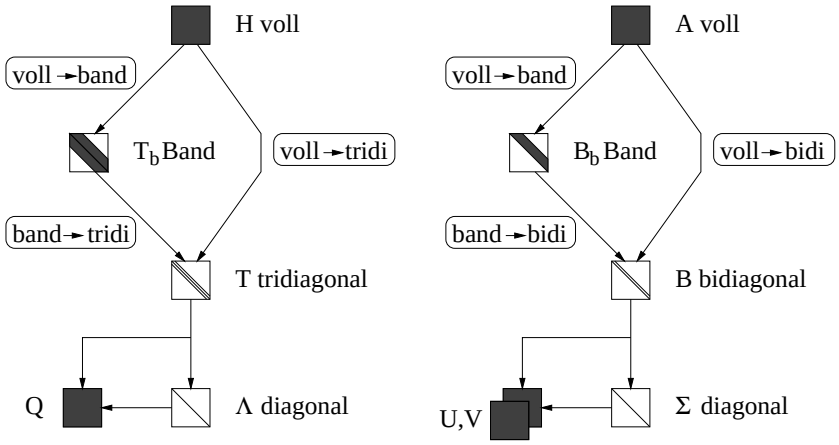


Abbildung 1.4: Schematischer Überblick der Reduktionsphasen.

Abbildung 1.4 skizziert die Vorgehensweise: Es ist möglich, die vollbesetzte Eingangsmatrix direkt auf Tridiagonal- oder Bidiagonalgestalt zu bringen [12]. Diese Verfahren sind als LAPACK-Routinen `DSYTRD` und `DGEBRD` umgesetzt. Alternativ können wir die Reduktionsphase noch weiter unterteilen, indem wir erst auf Bandgestalt reduzieren und von dieser Zwischenstufe weiter ausgehend B oder T ermitteln. Obwohl dies auf den ersten Blick wie ein Umweg aussieht, läßt sich durch Verwendung effizient optimierter BLAS-Routinen höhere Ausführungsgeschwindigkeit erzielen [8, 48, 56]. Selbstverständlich können die Verfahren $band \rightarrow \{tridi, bidi\}$ auch eingesetzt werden, wenn die Eingangsmatrix bereits in Bandgestalt vorliegt.
Für die Diagonalisierungsphase stehen eine Reihe von Verfahren zur Behandlung von **tSEP** und **bSVD** zur Verfügung, die im folgenden Abschnitt behandelt werden.

1.5 Verfahren für tSEP und bSVD

Bei den bisher vorgestellten Lösungsverfahren wird meist primär das **SEP** betrachtet und die Anwendung auf die **SVD** als „Nebenprodukt“ behandelt. Da jedoch die bidiagonale Singulärwertzerlegung (**bSVD**) eine relative Störungstheorie erlaubt (die Einträge von B sind eine RRR), während dies für das allgemeine symmetrisch tridiagonale Eigenwertproblem (**tSEP**) nicht durchgängig möglich ist, gibt es einige speziell auf die **bSVD** zugeschnittene Varianten. Diese lassen sich wiederum in der Umkehrung auf spezielle abgeleitete **tSEP** (positiv definit, Golub-Kahan Matrix) anwenden. Um die in Tabelle 1.1 aufgeführten Ansätze grob einzuteilen, können wir die Frage

	D&C	**QR**	**QD**	**Bis.**	**Inv. It.**	**RRR**
Technik	Basis	Basis	Werte	Werte	unabh.	unabh.
Fehler	abs.	rel.	rel.	rel.		
adaptiv	nein	nein	nein	ja	ja	ja
parallel	shared vs. distributed	Aufdat. Vektoren	nein	ja	Problem bei MGS	ja
Aufwand	b.c. $\mathcal{O}(n^2)$ w.c. $\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	$\mathcal{O}(kn)$	b.c. $\mathcal{O}(kn)$ w.c. $\mathcal{O}(kn^2)$	$\mathcal{O}(kn)$
$\exists$ *Impl.*	ja	ja	ja	ja	ja(?)	nein

Tabelle 1.1: Überblick der **tSEP**- und **bSVD**-Lösungsverfahren. Die Abkürzungen b.c. und w.c. stehen für *best case* und *worst case*.

nach der zugrunde liegenden *Technik* stellen.

Zunächst gibt es Verfahren (QD, Bisektion, auch QR), die eingesetzt werden, wenn lediglich Eigen- und Singulär*werte* gefragt sind. Für die **bSVD** (und abgeleitete **tSEP**) erlauben solche Verfahren entweder eine *absolute* oder *relative Fehleranalyse*.

Sind die zugehörigen Vektoren gesucht, können wir weiter zwischen zwei verschiedenen Techniken unterscheiden:

Auf der einen Seite versuchen Methoden wie QR oder Divide-and-Conquer während des gesamten Verfahrens $(T, I) \to (\Lambda, Q)$ eine orthogonale *Basis* aufrecht zu erhalten. Alle zur Diagonalisierung von T verwendeten Operationen werden durch sukzessives Aufdatieren von I in einer Matrix Q akkumuliert. Die Näherungen zeichnen sich daher durch gute numerische Orthogonalität aus, die jedoch im *worst case* mit einem hohen Aufwand bezahlt wird. Beide Verfahren liefern sowohl Eigen- bzw. Singulärwerte als auch Vektoren (siehe auch Abbildung 1.4).

Auf der anderen Seite stellen Verfahren wie die Inverse Iteration oder der RRR-Algorithmus die Vektoren voneinander *unabhängig* auf. Zu einer gegebenen Näherung an einen Eigen- oder Singulärwert wird durch Lösen geeigneter Gleichungssysteme eine Approximation an den zugehörigen Vektor bestimmt. Bei Clustern müssen jedoch Vorkehrungen getroffen werden, damit die numerische Orthogonalität nicht verloren geht. Während bei der Inversen Iteration oft nur eine explizite Reorthogonalisierung möglich ist, garantiert der RRR-Algorithmus durch geschickte Auswahl der zu lösenden Gleichungssysteme numerische Orthogonalität. Das unabhängige Aufstellen der Basisvektoren macht die Verfahren *adaptiv*, d.h. daß für k gesuchte Elemente des Spektrums nur ein entsprechend reduzierter Aufwand betrieben

werden muß. Adaptive Verfahren sind grundsätzlich einfach *parallelisierbar*. Zu den Beurteilungskriterien gehört auch hier noch die Verfügbarkeit von stabilen und geschwindigkeitsoptimierten *Implementierungen*. Fast alle beschriebenen Verfahren liegen als Routinen der LAPACK-Bibliothek vor.
Wir werden auf die in Tabelle 1.1 aufgeführten Methoden in den nächsten Abschnitten detailliert eingehen. Da wir in den folgenden Kapiteln ein Verfahren vorschlagen werden, das alternativ zu den übrigen **bSVD**-Lösern verwendet werden kann, nutzen wir den Rest dieses Kapitels auch, um unsere Anknüpfungspunkte deutlich zu machen.

1.5.1 Divide and Conquer

Im Unterschied zu der in Abschnitt 1.4.3 erwähnten Methode (*Spectral* Divide-and-Conquer) zielt dieser Ansatz auf die Lösung des **tSEP** ab. Die Idee ist, die Matrix T als Summe einer reduzierten Matrix und einer kontrollierbaren Störung (z.B. Rang-1-Modifikation mit $x \in \mathbb{R}^n$) darzustellen. Aus den rekursiv zu bearbeitenden Teilsystemen wird dann eine Lösung des ursprünglichen Problems erzeugt [9, 14, 42].

Divide:
Zerlege $T = \texttt{diag}(T_1, T_2) + xx^T$.

Teilprobleme:
Bearbeite rekursiv die **tSEP** $T_1 = Q_1 \Lambda_1 Q_1^T$ und $T_2 = Q_2 \Lambda_2 Q_2^T$.

Conquer:
Ermittle aus $(\Lambda_1, \Lambda_2, Q_1, Q_2)$ eine Lösung des **tSEP** $T = Q \Lambda Q^T$.

Dieses rekursive Vorgehen ermittelt alle Eigenwerte und datiert wahlweise die Basis aus Eigenvektoren auf.
Der Ansatz gilt als sehr attraktiv, da sich bereits serielle Implementierungen als sehr schnell gegenüber den bis dahin eingesetzten Standardverfahren (QR) erweisen. Bei Clustern von Eigenwerten kann durch *Deflation* die Komplexität von $\mathcal{O}(n^3)$ (w.c.) auf $\mathcal{O}(n^2)$ (b.c.) verringert werden. Das Aufdatieren der Eigenvektorbasis erfolgt mit Hilfe von Matrix-Matrix-Multiplikationen. Für diese stehen optimierte datenparallele Bibliotheken (PBLAS) zur Verfügung. Parallelisierbarkeit ist ferner gegeben durch die Bildung von Teilproblemen, die unabhängig auf verschiedenen Prozessoren bearbeitet werden können.
Auftretende Probleme bei der Entwicklung einer numerisch stabilen Implementierung — insbesondere der Lösung der sogenannten *secular equation* — sind weitgehend gelöst [50, 52]. Eine serielle Version ist mit der LAPACK-Routine `DSTEDC` verfügbar [66]. Es stellt sich heraus, daß eine Parallelisierung

für Shared-Memory-Rechner [30] leichter zu erzielen ist als für Distributed-Memory-Architekturen [31].
Der relativ hohe Bedarf an zusätzlichem Arbeitsspeicher kann reduziert werden, indem die Information über die aufdatierte Basis implizit abgespeichert wird und daraus explizite Darstellungen von Eigenvektoren erst bei Bedarf ermittelt werden.
Gezielte Modifikationen des Verfahrens erlauben den Entwurf eines Divide-and-Conquer Ansatzes für die **bSVD** [51]. Auch hier steht mit der Routine `DBDSDC` eine Implementierung mit Bibliotheksstandard bereit.
Als Gegenstand intensiver Forschungsarbeiten gilt der Divide-and-Conquer Ansatz als gut verstanden, wird aber möglicherweise seine Position als schnellstes Verfahren bald an den später beschriebenen RRR-Algorithmus abtreten müssen.

1.5.2 Das QR-Verfahren

Auch dieses Verfahren ([44] und die dort aufgeführten Referenzen) erzeugt — vergleichbar mit dem Divide-and-Conquer Ansatz — eine Basis aus Eigenvektoren durch sukzessives Aufdatieren, nun aber mit orthogonalen Ähnlichkeitstransformationen:

$$T_0 = T, \quad T_i - \tau_i I \stackrel{QR}{=} Q_i R_i, \quad T_{i+1} = R_i Q_i + \tau_i I, \quad i = 0, 1, \ \ldots$$

Mit geeignet gewählten Shift-Parametern τ_i konvergiert die Folge T_i global und kubisch gegen die gesuchte diagonale Matrix Λ aus Eigenwerten. Das Produkt $Q_0 Q_1 \cdots Q_i$ nähert eine Basis Q aus Eigenvektoren an. Die Transformationen Q_i lassen sich als Akkumulation geeigneter Givensrotationen so wählen, daß die Tridiagonalgestalt von T_i erhalten bleibt. Bis zum Aufkommen des Divide-and-Conquer Ansatzes als Standardlösung geltend, wenn alle Eigenwerte und -vektoren zu bestimmen sind, liegt dieses Verfahren als LAPACK-Routine `DSTEQR` implementiert vor. Der QR-Algorithmus ist inhärent seriell. Wir können aber das Aufdatieren der Ähnlichkeitstransformationen parallelisieren, indem wir jedem Prozessor eine möglichst gleich verteilte Anzahl von Eigenvektoren zuordnen.
Sind wir nicht an den Eigenvektoren, sondern nur an den Eigenwerten interessiert, so bietet sich die Variante von Pal-Walker-Kahan [63] an, die als wurzelfreies QR-Verfahren interpretiert werden kann und deutlich niedrigere Ausführungszeiten erlaubt, jedoch in manchen Fällen schlechteres Konvergenzverhalten zeigt. Die zugehörige LAPACK-Routine heißt `DSTERF` [45].
Die Übertragung des geshifteten QR-Verfahrens vom **tSEP** auf die **bSVD**

führt zu einer Folge

$$B_0 = B, \quad B_{i+1} = U_i^T B_i V_i, \quad i = 0, 1, \ \ldots \tag{1.1}$$

von Bidiagonalmatrizen, die gegen eine Diagonalmatrix Σ konvergiert [44]. Wir können die V_i analog zu den Q_i wählen. Die Matrizen U_i sind ebenfalls als Akkumulation von geeigneten Givensrotationen zu interpretieren, die für die Erhaltung der bidiagonalen Gestalt sorgen.
Bei der Bestimmung der **bSVD** sind entscheidende Verbesserungen des QR-Verfahrens möglich, die eine relative Störungstheorie erlauben [21]. Diese werden erreicht durch geeignete Auswahl der Shifts sowie durch Verwendung von Kriterien, die es ermöglichen, ein Nebendiagonalelement auf Null zu setzen, ohne die relative Genauigkeit der Approximationen an die Singulärwerte zu beeinträchtigen. Eine Implementierung liegt als LAPACK-Routine `DBDSQR` vor.
Für unser Ziel, ein schnelles und stabiles Lösungsverfahren für die **bSVD** zu entwerfen, wenden wir einige Schritte des QR-Verfahrens als *Präkonditionierung der Eingangsmatrix* B an. Unser Gesamtalgorithmus sieht als Option vor, anstelle von $B = B_0$ die Matrix B_k mit einem gewissen $k \ll n$ als Eingangsmatrix für den Kernalgorithmus zu verwenden. Für Details siehe Abschnitt 4.1.2.

1.5.3 Das QD-Verfahren

Anstelle von QR-Zerlegungen lassen sich auch LU- oder Cholesky-Faktorisierungen im iterativen Diagonalisierungsprozeß anwenden [34, 38, 65]:

$$B_0 = B, \quad B_{i+1}^T B_{i+1} \overset{Chol.}{=} B_i B_i^T - \tau_i^2 I, \quad i = 0, 1, \ \ldots$$

Ist i gerade und sind $\tau_i = \tau_{i+1} = 0$, so lassen sich zwei aufeinander folgende Iterationsschritte als ein Iterationsschritt des (ungeshifteten) QR-Verfahrens interpretieren:

$$B_{i+1} = B_i^T U_{i/2} \quad B_{i+2} = B_{i+1}^T V_{i/2}$$

Wir bezeichnen von nun an zwei aufeinander folgende Iterierte mit $B = B_i$ und $B^+ = B_{i+1}$. Die Auswertung der einzelnen Einträge der Gleichung $(B^+)^T B^+ = BB^T - \tau^2 I$ ergibt („Rhombus-Regel“)

$$(a_{j+1}^+)^2 + (b_j^+)^2 = a_{j+1}^2 + b_{j+1}^2 - \tau^2 \quad \text{und} \quad (a_j^+)^2 (b_j^+)^2 = a_{j+1}^2 b_j^2$$

Mit Einführung der Größen $q_j = a_j^2$ und $e_j = b_j^2$ kann die Iteration ohne Verwendung von Wurzeln ausgeführt werden. Der Übergang $B \to B^+$ wird

nun mit einer *Quotienten-Differenzen*-Transformation (QD) gestaltet. Mit den Hilfsvariablen

$$\begin{aligned} d_{j+1} &:= q_{j+1}^+ - e_{j+1} \\ &= q_{j+1} - e_j^+ - \tau^2 \\ &= \frac{q_{j+1}}{q_j^+}\left(q_j^+ - \frac{q_j^+}{q_{j+1}} e_j^+\right) - \tau^2 \\ &= \frac{q_{j+1}}{q_j^+}\left(q_j^+ - e_j\right) - \tau^2 \\ &= \frac{q_{j+1}}{q_j^+} d_j - \tau^2 \end{aligned} \tag{1.2}$$

können wir den Übergang in Algorithmus 1.1 formulieren. Die Abkürzung **dqds** steht für „**d**ifferential **q**uotient **d**ifference with **s**hift“.

Algorithmus 1.1 Die **dqds**-Transformation (klassisch).

1: $d_1 = q_1 - \tau^2$
2: **for** $j = 1 : n-1$ **do**
3: $\quad q_j^+ = d_j + e_j$
4: $\quad t = \frac{q_{j+1}}{q_j^+}$
5: $\quad e_j^+ = t e_j$
6: $\quad d_{j+1} = t d_j - \tau^2$
7: **end for**
8: $q_n^+ = d_n$

Wir werden auf dieser Basis weitere QD-artige Transformationen entwickeln. Für $\tau = 0$ ist dieser Algorithmus rückwärtsstabil und erlaubt ansonsten eine gemischte Stabilitätsanalyse mit relativen Störungen. Solche Analysen werden in Abschnitt 2.3 eingehend dargestellt.

Eine effiziente Implementierung der LAPACK-Routine `DLASQ1` für das QD-Verfahren führt zu deutlich besseren Ausführungszeiten als der QR-Algorithmus, wenn nur Singulärwerte gefragt sind [60]. Sind auch Singulärvektoren gesucht, so ist letzterer vorzuziehen. Auch das QD-Verfahren ist inhärent seriell, jedoch deutlich schneller als das nachfolgend vorgestellte Bisektionsverfahren. Die Übertragung auf durch Normalengleichungen oder Jordan-Wielandt-Darstellungen gestellte **tSEP** ist ohne größeren zusätzlichen Aufwand möglich.

1.5.4 Das Bisektionsverfahren

Zu einer symmetrisch tridiagonalen Matrix T und einem reellen Shift τ existiere eine Zerlegung

$$T-\tau I = LDL^T, \quad D = \texttt{diag}([d_1, \ldots, d_n]), \quad L = I+\texttt{diag}([l_1, \ldots, l_{n-1}], -1)$$

Diese läßt sich beispielsweise mit Varianten des Gaußverfahrens aufstellen. Nach dem Trägheitsgesetz von Sylvester (Satz 1.11) gilt

$$\text{sign}(T - \tau I) = \text{sign}(D)$$

Sei $n_\tau := \#\{d_i | d_i < 0\}$ die Anzahl der negativen Pivotelemente. Der Wert von n_τ entspricht der Anzahl der Eigenwerte von T, die kleiner als τ sind, also

$$\lambda_1 < \ldots < \lambda_{n_\tau} < \tau \leq \lambda_{n_\tau+1} < \ldots < \lambda_n$$

Entsprechend ist die Anzahl der Eigenwerte in dem halboffenen Intervall $[\alpha, \beta[$ beschrieben durch $n_\beta - n_\alpha$. Dieses Intervall kann feiner unterteilt werden durch die Bestimmung von n_τ mit $\alpha < \tau < \beta$. Daraus lassen sich Algorithmen wie das Bisektionsverfahren entwickeln, die Intervall-Einschließungen für ausgewählte Eigenwerte bestimmen. Unter Beachtung der zugrunde liegenden Gleitpunktarithmetik gibt es numerisch stabile serielle (`DSTEBZ`) und parallele (`PDSTEBZ`) Computer-Implementierungen [18, 54]. Ihr Vorteil gegenüber dem QD-Verfahren oder der Methode von Pal, Walker und Kahan ist, daß Teilspektren mit entsprechend reduziertem Aufwand bestimmt werden können (Adaptivität). Zur Berechnung aller Eigenwertapproximationen benötigen sie allerdings deutlich mehr Rechenzeit.
Die Übertragung auf das Zählen von Singulärwerten einer bidiagonalen Matrix B ist in [37] beschrieben. Hier wird ausgenutzt, daß die zu bearbeitende Tridiagonalmatrix mit B^TB, BB^T oder T_{GK} bereits als RRR vorliegt. Ferner wird beschrieben, wie die LDL^T-Faktorisierungen dieser drei Darstellungen untereinander in Beziehung gesetzt werden können. In Kapitel 3 werden wir dieses Konzept wesentlich erweitern, um gut gekoppelte Singulärvektorpaare zu bestimmen.

1.5.5 Inverse Iteration

Die Eigenwerte aus dem **SEP** für $(T - \bar{\lambda} I)^{-1}$ können ausgedrückt werden durch $1/(\lambda_j - \bar{\lambda})$, während die Eigenvektoren im Vergleich zu dem durch die Matrix T gestellten **tSEP** unverändert bleiben, vgl. Bemerkung 1.9. Das Verfahren der Inversen Iteration erlaubt es damit, zu einer gegebenen (guten) Näherung $\bar{\lambda} = \bar{\lambda}^{(0)}$ an einen Eigenwert und einem Startvektor $\bar{q}^{(0)}$ eine

Approximation für den zugehörigen Eigenvektor zu bestimmen:

$$\text{Löse} \quad (T - \bar{\lambda}^{(i-1)} I) \cdot \bar{q}^{(i)} = \zeta^{(i)} \bar{q}^{(i-1)}, \quad i = 1, 2, \ \ldots \tag{1.3}$$

Dabei ist in jedem Schritt ein lineares Gleichungssystem (LGS) zu lösen und die neue Iterierte zu normieren (angedeutet durch die Skalierungsfaktoren $\zeta^{(i)}$). Die Shift-Parameter $\bar{\lambda}^{(i-1)}$ können variiert werden, um weiter verbesserte Näherungen an den entsprechenden Eigenwert zu erhalten.
Ist $\bar{\lambda}$ eine Näherung an den j-ten Eigenwert von T und außerdem $|\lambda_j - \bar{\lambda}| \ll \min_{l \neq j}\{|\lambda_l - \bar{\lambda}|\}$, so konvergieren die $\bar{q}^{(i)}$ der Richtung nach gegen den j-ten Eigenvektor [63]. (Eventuell alternieren die Vorzeichen der Iterierten $\bar{q}^{(i)}$.) Während dieses Verfahren für isolierte Eigenwerte eine sehr hohe Konvergenzgeschwindigkeit besitzt, müssen bei Clustern die ermittelten Basisvektoren oft explizit reorthogonalisiert werden. Sind k Vektoren mit einem Verfahren wie der Modifizierten Gram-Schmidt Orthogonalisierung (MGS, vgl. Algorithmus 1.2) nachzubearbeiten, so ergibt sich ein zusätzlicher Aufwand von $\mathcal{O}(k^2 n)$. Hat k die Größenordnung von n, verliert die Inverse Iteration ihre Eigenschaft eines ursprünglich einfach zu parallelisierenden Verfahrens mit quadratischem Aufwand.

Algorithmus 1.2 Modifizierte Gram-Schmidt Orthogonalisierung (MGS).

Gegeben: Eine zu orthogonalisierende Matrix $W \in \mathbb{R}^{n \times k}$.
Gesucht: Eine Zerlegung $W = Q \cdot R$ mit orthogonalem $Q \in \mathbb{R}^{n \times k}$ und oberer Dreiecksmatrix $R \in \mathbb{R}^{k \times k}$.

1: **for** $j = 1 : k$ **do**
2: $\quad Q(:, j) = W(:, j)$
3: $\quad$ **for** $i = 1 : j - 1$ **do**
4: $\quad\quad r_{ij} = Q(:, i)^T Q(:, j)$
5: $\quad\quad Q(:, j) = Q(:, j) - r_{ij} Q(:, i)$
6: $\quad$ **end for**
7: $\quad r_{jj} = \|Q(:, j)\|$
8: $\quad Q(:, j) = Q(:, j)/r_{jj}$
9: **end for**

Das Kernproblem für numerisch stabile Implementierungen bleibt die Sensitivität der Vektoren bei kleinen *absoluten* Lücken in der Verteilung der Eigenwerte. Es gibt gezielt konstruierte Matrizen, bei denen die aktuelle Version der LAPACK-Routine `DSTEIN` zu fehlerhaften Ergebnissen führen kann [25]. Für viele andere Problemklassen (mit weitgehend isolierten Eigenwerten) zählt die Inverse Iteration aber zu den schnellsten Verfahren mit zufriedenstellender Genauigkeit.

Wir weisen darauf hin, daß gerade die Inverse Iteration bei engen Clustern oft eine beliebige Basis des betreffenden Unterraumes erzeugt (siehe auch Abschnitt 1.3.6). Für die Bestimmung der **bSVD** ist deshalb zu gewährleisten, daß die Iterationsschritte für linke und rechte Singulärvektoren aufeinander abgestimmt werden. Insbesondere müssen Kopplungen bei eventuellen Reorthogonalisierungsmaßnahmen erhalten bleiben.

Kapitel 2

Der RRR-Algorithmus

Ein sehr effizientes **tSEP**-Lösungsverfahren ist der RRR-Algorithmus (RRR = relativ robuste Repräsentationen, vgl. Definition 1.25). Er setzt sich aus der Anwendung verschiedener Teilkonzepte zusammen. Diese werden im Anschluß an einen Überblick der grundsätzlichen Arbeitsweise genauer dargestellt. Danach geben wir eine vorläufige Beurteilung und schließen damit die Diskussion aller gängigen Lösungsverfahren für das **tSEP** aus dem vorigen Kapitel ab. Die Kombination der einzelnen Teilkonzepte zum RRR-Algorithmus findet sich erstmalig in [29].
Bei der Beschreibung nutzen wir wieder die Gelegenheit, Ansatzpunkte für Verbesserungsvorschläge herauszustellen. Wir befassen uns ausführlich mit der Übertragung dieses Lösungsverfahrens auf die **bSVD** in Kapitel 3. Diese mündet in die Einführung des sogenannten *Kernalgorithmus*. Ein Vorschlag für ein umfassendes **bSVD**-Lösungsverfahren und serielle numerische Experimente erfolgen in Kapitel 4.

2.1 Einführung

Der RRR-Algorithmus läßt sich in einem Satz beschreiben als eine Variante der Inversen Iteration, die auch bei Clustern ohne explizite Reorthogonalisierung der Eigenvektoren auskommt. Diese Variante erfüllt die wichtigsten Eigenschaften, die bei der Bearbeitung des **tSEP** nachgefragt werden (Aufwand, Geschwindigkeit, Genauigkeit, Adaptivität, Parallelisierbarkeit) sehr gut durch das Verwenden der folgenden Strategien:

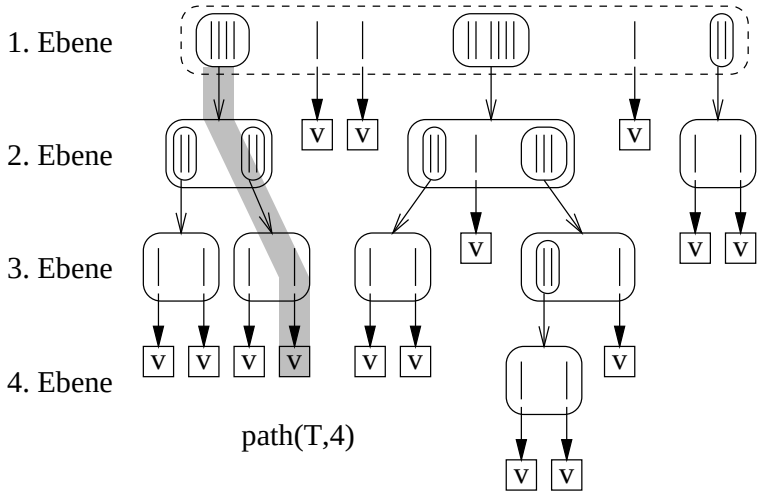


Abbildung 2.1: Ein Darstellungbaum. Auf der ersten Ebene (Stufe) ist das Spektrum von T angedeutet. Bei isolierten Eigenwerten kann mit Hilfe von BABE-Faktorisierungen direkt eine Eigenvektorapproximation bestimmt werden (gefüllte Pfeile). Sind die Eigenwerte geclustert, so wählen wir einen geeigneten Shift-Parameter (offene Pfeile) und arbeiten mit der Translation weiter, bis alle Eigenwerte isoliert sind. Für den Pfad math$(T, 4)$ sind zwei Shift-Parameter notwendig, um eine Translation von T zu finden, bzgl. derer der vierte Eigenwert isoliert ist. Der dritte Shift-Parameter wird dann benutzt, um ein lineares Gleichungssystem mit einer BABE-Faktorisierung aufzustellen.

1. Darstellung von T und ihren Translationen als Produkte von diagonalen und bidiagonalen Matrizen (z.B. $T = L^{(0)}D^{(0)}(L^{(0)})^T$) mit dem Ziel, relativ robuste Repräsentationen zu erzeugen (siehe Abschnitt 1.3.4 über Störungstheorie und insbesondere Definition 1.25).

2. Faktorisierungen von Translationen wie

$$LDL^T - \nu I = L^+D^+(L^+)^T$$

   werden berechnet mit QD-artigen Transformationen (Varianten von Algorithmus 1.1). Diese beschreiben wir in Abschnitt 2.3.

3. Bestimmung von guten Näherungen an Eigenvektoren zu isolierten Eigenwerten mit einer auf sogenannten BABE-Faktorisierungen basierenden Technik. Für Details siehe Abschnitt 2.2.

4. Der Begriff von *relativen* Distanzen erlaubt es, eine Strategie zu entwickeln, um mit Clustern von Eigenwerten umzugehen. Dabei nutzen wir folgende Beobachtungen aus:

   - Eigenvektoren sind invariant unter Translation (Bemerkung 1.9).
   - Relative Distanzen sind nicht invariant unter Translation (Bemerkung 1.16).

   Sind Eigenwerte einer Matrix LDL^T bzgl. ρ_p als geclustert anzusehen, wählen wir einen Shift ν „nahe bei“ diesem Cluster und setzen das Verfahren mit einer Faktorisierung der Translation $LDL^T - \nu I = L^+D^+(L^+)^T$ fort. Siehe auch Beispiel 2.1 im Abschnitt über BABE-Faktorisierungen.

5. Die Durchführung des RRR-Algorithmus auf dem Spektrum kann interpretiert werden als das Abarbeiten eines Darstellungsbaumes (engl. *representation tree*, vgl. Abbildung 2.1). Die Konstruktion einer Eigenvektorapproximation zu einem einzelnen Eigenwert λ_j können wir als das Beschreiten eines speziellen Pfades von der Wurzel zu einem Blatt dieses Baumes auffassen. Ein Pfad kann beschrieben werden durch die Auswahl der Shift-Parameter $\mathrm{math}(T, j) := (\nu_j^{(1)}, \ldots, \nu_j^{(\mathrm{rec}_j)})$.

Mit der Routine `DSTEGR` existiert ab der LAPACK-Version 3.0 eine vorläufige Implementierung des RRR-basierten Ansatzes.

2.2 Bestimmung eines Eigenvektors

Eine Kernidee des RRR-Algorithmus ist das Anwenden eines bestimmten Verfahrens zur Bestimmung von guten Näherungen an Eigenvektoren, sofern die zugehörigen Eigenwerte gut approximiert sind. Dieses Verfahren kann sowohl als Modifikation des Godunov-Schemas [39, 40] interpretiert werden als auch als Lösung eines linearen Gleichungssystems, dessen Matrix dargestellt wird als sogenannte BABE-Faktorisierung[1] [35, 36]. Die Beziehung zwischen beiden Interpretationen und die Einbettung in den RRR-Algorithmus sind in [26, 28] dargestellt.
Ist $\lambda = \lambda_j$ ein Eigenwert von T, so erhalten wir den zugehörigen Eigenvektor $q = Q(:, j)$ durch Lösen der Gleichung

$$(T - \lambda I) \cdot q = 0$$

[1] (engl. BABE = *Burn At Both Ends*)

In der Gleitpunktarithmetik steht i.d.R. nur eine Näherung $\bar{\lambda}$ an λ zur Verfügung, so daß wir auf die Lösung von

$$(T - \bar{\lambda} I) \cdot \bar{q} = r \tag{2.1}$$

ausweichen müssen, wobei r hier für ein (kleines) Residuum steht. Konkret betrachten wir nun

$$(T - \bar{\lambda} I) \cdot \bar{q}^{(k)} = \gamma_k^+ e_k \quad \text{mit} \quad \bar{q}_k^{(k)} = 1 \tag{2.2}$$

Wir nehmen an, daß folgende Faktorisierungen existieren:

$$T - \bar{\lambda} I = L^+ D^+ (L^+)^T = U^+ R^+ (U^+)^T$$

Dabei seien D^+ und R^+ Diagonalmatrizen und L^+ bzw. U^+ untere bzw. obere Einheits-Bidiagonalmatrizen, also z.B.

$$D^+ = \mathtt{diag}([d_1^+, \ldots, d_n^+]) \quad \text{und} \quad L^+ = I + \mathtt{diag}([l_1^+, \ldots, l_{n-1}^+], -1)$$

Wir können dann $\bar{q}^{(k)}$ bestimmen durch

$$\begin{array}{rcll} \bar{q}_k^{(k)} & = & 1 & \\ \bar{q}_i^{(k)} & = & -l_i^+ \bar{q}_{i+1}^{(k)} & \text{für } i = k-1 : -1 : 1 \\ \bar{q}_{i+1}^{(k)} & = & -u_i^+ \bar{q}_i^{(k)} & \text{für } i = k : n-1 \end{array} \tag{2.3}$$

Formal lösen wir damit ein LGS

$$(T - \bar{\lambda} I) \cdot \bar{q}^{(k)} = (N_k^+ G_k^+ (N_k^+)^T) \cdot \bar{q}^{(k)} = \gamma_k^+ e_k \tag{2.4}$$

dessen Matrix gegeben ist durch eine BABE-Faktorisierung $N_k^+ G_k^+ (N_k^+)^T$ mit $G_k^+ = \mathtt{diag}([d_1^+, \ldots, d_{k-1}^+, \gamma_k^+, r_{k+1}^+, \ldots, r_n^+])$ und

$$\begin{array}{rcll} N_k^+ & = I & + & \mathtt{diag}([l_1^+, \ldots, l_{k-1}^+, 0, \ldots, 0], -1) \\ & & + & \mathtt{diag}([0, \ldots, 0, u_k^+, \ldots, u_{n-1}^+], 1) \end{array}$$

Siehe dazu Abbildung 2.2. Durch Auswerten von

$$e_k^T (T - \bar{\lambda} I) e_k = e_k^T (N_k^+ G_k^+ (N_k^+)^T) e_k$$

erhalten wir $\Gamma^+ = [\gamma_1^+, \ \ldots, \gamma_n^+]$ mit

$$\gamma_k^+ = d_k^+ + r_k^+ - (T_{k,k} - \bar{\lambda}), \quad k = 1 : n \tag{2.5}$$

Wegen der Gleitpunktarithmetik werden wir jedoch numerisch stabilere Formeln verwenden, siehe Gleichung (3.3) auf Seite 73. Die Werte γ_k^+ bezeichnen wir auch als *Twist-Elemente*, die zugehörigen Indizes als *Twist-Positionen*. Anhand von vier Leitfragen erfolgt nun eine Diskussion der wichtigsten Eigenschaften von Γ^+. Für Details siehe [29].

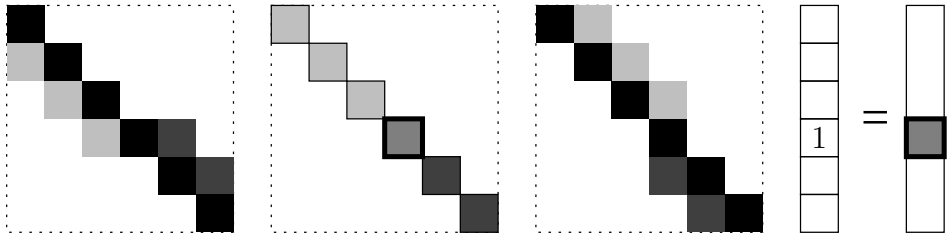


Abbildung 2.2: Lösung von Gleichung (2.4) mit einer BABE-Faktorisierung. N_k^+ kann auch als verdrehte (engl. *twisted*) Matrix angesehen werden. Der Index k entspricht dann der Twist-Position.

1. Welche Wahl von k führt zu einer guten Näherung an den Eigenvektor?

Es liegt nahe, das betragskleinste Element von Γ^+ zu wählen, um das Residuum $\|(T-\bar{\lambda}I)\cdot\bar{q}^{(k)}\| = |\gamma_k^+| = \min_i\{|\gamma_i^+|\}$ zu minimieren. Prinzipiell können wir alle Twist-Elemente mit $|\gamma_i^+| = \mathcal{O}(|\gamma_k^+|)$ in Betracht ziehen.
Unter Beachtung der im Satz von Davis und Kahan [15, 16, 29, 63] beschriebenen Abschätzung für die Winkel

$$\left|\sin\left(\angle(q,\bar{q}^{(i)})\right)\right| \leq \frac{\|(T-\bar{\lambda}I)\cdot\bar{q}^{(i)}\|}{\min_{l\neq j}\{|\lambda_l-\bar{\lambda}|\}\cdot\|\bar{q}^{(i)}\|} = \frac{1}{\min_{l\neq j}\{|\lambda_l-\bar{\lambda}|\}}\cdot\frac{|\gamma_i^+|}{\|\bar{q}^{(i)}\|}$$

können wir auch nach $|\gamma_i^+|/\|\bar{q}^{(i)}\|$ minimieren, indem wir

$$\left\|\bar{q}^{(i+1)}\right\|^2 = l_i^2\left\|\bar{q}^{(i)}(1:i)\right\|^2 + \frac{1}{u_i^2}\left\|\bar{q}^{(i)}(i+1:n)\right\|^2$$

ausnutzen, um die 2-Normen der $\bar{q}^{(i)}$ effizient zu berechnen [26].

2. In welcher Beziehung steht Γ^+ zu $Q(:,j)$?

Wir betrachten zunächst die Spektralzerlegung

$$(T-\bar{\lambda}I)^{-1} = Q(\Lambda-\bar{\lambda}I)^{-1}Q^T = \sum_{i=1}^{n} Q(:,i)Q(:,i)^T(\lambda_i-\bar{\lambda})^{-1}$$

Für $\bar{\lambda}\to\lambda_j$ bilden wir den Grenzwert

$$(\lambda_j-\bar{\lambda})(T-\bar{\lambda}I)^{-1} = \sum_{i=1}^{n} Q(:,i)Q(:,i)^T\frac{\lambda_j-\bar{\lambda}}{\lambda_i-\bar{\lambda}} \to Q(:,j)Q(:,j)^T$$

und betrachten die (k,k)-te Komponente dieser Gleichung

$$(\lambda_j - \bar{\lambda})\frac{1}{\gamma_k^+} = (\lambda_j - \bar{\lambda})\frac{1}{\gamma_k^+(\bar{\lambda})} \to Q(k,j)^2 \quad \text{für } \bar{\lambda} \to \lambda_j$$

Im Grenzwert ist also $\frac{1}{\gamma_k^+}$ proportional zum Quadrat der k-ten Komponente des zu approximierenden j-ten Eigenvektors. Der betragskleinste Eintrag von Γ^+ korreliert also mit dem betragsgrößten Eintrag von $Q(:,j)$.

3. Wann ist $|\gamma_k^+|$ hinreichend klein?

In [29] wird gezeigt, daß für gute Approximationen $\bar{\lambda}$ mit

$$\frac{|\lambda_j - \bar{\lambda}|}{\min_{l\neq j}\{|\lambda_l - \bar{\lambda}|\}} \leq \frac{1}{M}\cdot\frac{1}{n-1} \quad \text{wobei } M > 1$$

und $|Q(k,j)| \geq n^{-1/2}$ die Abschätzung

$$|\gamma_k^+| \leq \frac{|\lambda_j - \bar{\lambda}|}{Q(k,j)^2}\cdot\frac{M}{M-1} \leq n|\lambda_j - \bar{\lambda}|\cdot\frac{M}{M-1}$$

gilt. Wir halten als Zwischenergebnis fest, daß die (normierte) Lösung von $(T-\bar{\lambda}I)\cdot\bar{q}^{(k)} = \gamma_k^+ e_k$ eine gute Eigenvektorapproximation ist, wenn

$$|\gamma_k^+| = \min_i\{|\gamma_i^+|\} \tag{2.6}$$

$$\frac{|\lambda_j - \bar{\lambda}|}{\min_{l\neq j}\{|\lambda_l - \bar{\lambda}|\}} \ll 1 \tag{2.7}$$

Um die zweite Bedingung zu erreichen, greifen wir auf das Konzept von *relativen Distanzen* der Eigenwerte zurück:

Beispiel 2.1 (Bessere Näherungen durch Translationen)
Wir nehmen an, daß λ_j und $\lambda_{j+1} = (1+\epsilon)\lambda_j$ einen Eigenwertcluster von T bilden. In der Gleitpunktarithmetik können wir bestenfalls eine Näherung $\bar{\lambda}$ an λ_j mit $|\bar{\lambda} - \lambda_j| = \mathcal{O}(\epsilon|\lambda_j|)$ bestimmen und wegen $|\lambda_j - \bar{\lambda}|/|\lambda_{j+1} - \bar{\lambda}| = \mathcal{O}(1)$ und (2.7) keine gute Abschätzung für ein entsprechendes $|\gamma_k^+|$ einer BABE-Faktorisierung erwarten.
Zur Lösung dieses Problems suchen wir nach einem Shift-Parameter $\tau \approx \lambda_j$, der die Erzeugung einer RRR für $T - \tau I$ erlaubt. Mit einer guten Näherung $\bar{\lambda}^+$ an den j-ten Eigenwert von $T - \tau I$ ist die Bedingung (2.7) deutlich besser erfüllt. Wir können nun anstelle von T mit $T - \tau I$ arbeiten, um Approximationen an die Eigenvektoren zu bestimmen. Details zur Erzeugung einer RRR und zur Bestimmung hinreichend genauer Eigenwertnäherungen sind in [28, 29] sowie in den Abschnitten 2.4 und 3.4.3 beschrieben.

4. Welchen Einfluß haben Rundungsfehler auf die numerische Qualität der Eigenvektornäherungen?

Wir untersuchen nun die Näherungslösungen $\bar{\Lambda}$ und $\bar{Q}$ für das durch T gegebene **tSEP**. Die Eigenwertapproximationen seien dabei auf hohe relative Genauigkeit bestimmt. Die Berechnung von $\bar{Q}$ erfolge für $j = 1 : n$ durch das Lösen eines LGS

$$(T - \bar{\lambda}_j I) \cdot \bar{Q}(:, j) = (N_k^+ G_k^+ (N_k^+)^T) \cdot \bar{Q}(:, j) = \gamma_k^+ e_k$$

mit dem jeweils betragskleinsten Twist-Element γ_k^+. Zusätzlich fordern wir bei der Implementierung in der Gleitpunktarithmetik, daß die Matrizen N_k^+ und G_k^+ mit QD-artigen Transformationen (Varianten von Algorithmus 1.1, vgl. Abschnitt 2.3) bestimmt werden. Nach [29] gilt dann

$$\|(T - \bar{\lambda}_j I)\bar{Q}(:, j)\| = \mathcal{O}(n\epsilon\|T\|) \tag{2.8}$$

$$\left|\bar{Q}(:, j)^T \bar{Q}(:, j+1)\right| = \frac{\mathcal{O}(n\epsilon)}{\rho(\lambda_j, \lambda_{j+1})} \tag{2.9}$$

Die numerische Orthogonalität der Eigenvektornäherungen hängt also von den *relativen* Distanzen der Eigenwerte ab. Für Cluster verwenden wir die in Beispiel 2.1 skizzierte Shift-Strategie. In numerischen Experimenten ([29], siehe auch Abschnitt 4.3) bewährt sich eine Wahl $t = \min\{0.01, 1/n\}$ für die Klassifizierung des Spektrums in Cluster und isolierte Eigenwerte, vgl. Definition 1.15.
Im nächsten Abschnitt präsentieren wir die bereits öfter erwähnten QD-artigen Transformationen.

2.3 QD-artige Transformationen

Wir betrachten zunächst einige Quotienten-Differenzen-Transformationen:

- In Algorithmus 2.1 (links) bestimmen wir $BB^T - \mu^2 I = \hat{L}^+ \hat{D}^+ (\hat{L}^+)^T$ mit Hilfe von $\hat{s}_{i+1} = \hat{d}_{i+1}^+ - a_{i+1}^2 = \frac{b_i \hat{l}_i^+}{a_i} \hat{s}_i - \mu^2$.
- In Algorithmus 2.1 (rechts) bestimmen wir $B^T B - \mu^2 I = \check{L}^+ \check{D}^+ (\check{L}^+)^T$ mit Hilfe von $\check{p}_{i+1} = \check{d}_{i+1}^+ - b_{i+1}^2 = \frac{a_{i+1} \check{l}_i^+}{b_i} \check{p}_i - \mu^2$.
- In Algorithmus 2.2 bestimmen wir $T_{GK} - \mu I = \tilde{L}^+ \tilde{D}^+ (\tilde{L}^+)^T$.
- In Algorithmus 2.3 bestimmen wir $LDL^T - \tau I = L^+ D^+ (L^+)^T$ mit Hilfe von $s_{i+1} = d_{i+1}^+ - d_{i+1} = l_i^+ l_i s_i - \tau$.

Algorithmus 2.1 Faktorisiere $B^TB - \mu^2 I$ (links) und $BB^T - \mu^2 I$ (rechts).

Gegeben: B, μ
Gesucht: $\hat{D}^+, \hat{L}^+, \hat{S}$

1: $\hat{s}_1 = -\mu^2$
2: **for** $i = 1 : n-1$ **do**
3: $\quad \hat{d}_i^+ = \hat{s}_i + a_i^2$
4: $\quad \hat{l}_i^+ = \frac{a_i b_i}{\hat{d}_i^+}$
5: $\quad \hat{s}_{i+1} = \frac{b_i \hat{l}_i^+}{a_i}\hat{s}_i - \mu^2$
6: **end for**
7: $\hat{d}_n^+ = \hat{s}_n + a_n^2$

Gegeben: B, μ
Gesucht: $\check{D}^+, \check{L}^+, \check{P}$

1: $\check{p}_1 = a_1^2 - \mu^2$
2: **for** $i = 1 : n-1$ **do**
3: $\quad \check{d}_i^+ = \check{p}_i + b_i^2$
4: $\quad \check{l}_i^+ = \frac{a_{i+1} b_i}{\check{d}_i^+}$
5: $\quad \check{p}_{i+1} = \frac{a_{i+1}\check{l}_i^+}{b_i}\check{p}_i - \mu^2$
6: **end for**
7: $\check{d}_n^+ = \check{p}_n$

Algorithmus 2.2 Faktorisiere $T_{GK} - \mu I = \tilde{L}^+ \tilde{D}^+ (\tilde{L}^+)^T$.

Gegeben: $c := [a_1, b_1, a_2, \ldots, b_{n-1}, a_n]$, μ
Gesucht: $\tilde{D}^+, \tilde{L}^+$

1: $\tilde{d}_1^+ = -\mu$
2: **for** $i = 1 : 2n-1$ **do**
3: $\quad \tilde{l}_i^+ = \frac{c_i}{\tilde{d}_i^+}$
4: $\quad \tilde{d}_{i+1}^+ = -\frac{c_i^2}{\tilde{d}_i^+} - \mu \quad \{= -c_i \tilde{l}_i^+ - \mu\}$
5: **end for**

Algorithmus 2.3 Faktorisiere $LDL^T - \tau I = L^+ D^+ (L^+)^T$.

Gegeben: D, L, τ
Gesucht: D^+, L^+, S

1: $s_1 = -\tau$
2: **for** $i = 1 : n-1$ **do**
3: $\quad d_i^+ = d_i + s_i$
4: $\quad l_i^+ = \frac{d_i l_i}{d_i^+}$
5: $\quad s_{i+1} = l_i^+ l_i s_i - \tau$
6: **end for**
7: $d_n^+ = d_n + s_n$

Diese Alternativen zur klassischen Gauss-Elimination werden in [37] (Algorithmen 2.1 und 2.2) bzw. in [29] (Algorithmus 2.3) vorgeschlagen. Sie bauen auf der in Abschnitt 1.5.3 dargestellten **dqds**-Transformation auf.
Die Verallgemeinerung von LDL^T-Faktorisierungen auf BABE-Faktorisierungen beschreiben wir in Abschnitt 3.2.1, Algorithmus 3.1 und Abschnitt 3.3.1, Algorithmus 3.5. Zunächst erfolgt eine Diskussion der numerischen Stabilität dieser Faktorisierungen. Wir gehen anschließend auf mögliche Breakdowns bei der Durchführung in der Gleitpunktarithmetik ein.

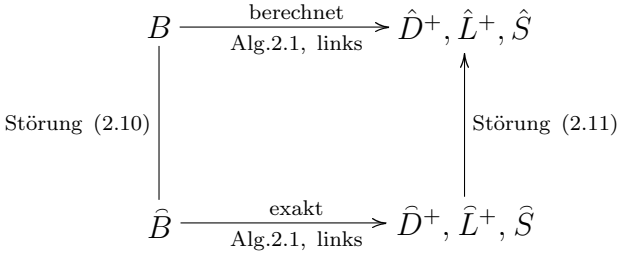
Gemischte relative Stabilitätsanalyse

Bei Formulierungen wie

Faktorisiere $B^TB - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$ mit Hilfe von Algorithmus 2.1.

müssen wir uns in der Gleitpunktarithmetik normalerweise mit Ergebnissen $[\hat{D}^+, \hat{L}^+, \hat{S}]$ zufrieden geben, die durch Rundungsfehler bedingt nur eine Näherung der Zerlegung sind. Wir diskutieren nun die Qualität der mit den Algorithmen 2.1, 2.2 und 2.3 generierten Daten. Dazu geben wir repräsentativ eine gemischte Stabilitätsanalyse für die Faktorisierung $B^TB - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$ an. Wir betrachten also eine leichte multiplikative Störung $\widehat{B}$ der Eingangsmatrix B und führen darauf Algorithmus 2.1 *in exakter Arithmetik* aus. Die so gewonnenen Ergebnisse $\widehat{D}^+, \widehat{L}^+$ und $\widehat{S}$ werden wiederum komponentenweise gestört und lassen sich dann als die durch Algorithmus 2.1 *in Gleitpunktarithmetik* berechnete Größen $\hat{D}^+, \hat{L}^+$ und $\hat{S}$ interpretieren. Das Bemerkenswerte am folgendem Satz und Beweis ist, daß die Aussagen weitgehend unabhängig vom Wert des gewählten Shift-Parameters μ sind. Es ist lediglich gefordert, daß kein Unter- oder Überlauf und damit auch kein Breakdown auftritt.

Satz 2.2 (Stabilitätsanalyse für $B^TB - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$)
Wir nehmen an, daß die linke Seite von Algorithmus 2.1 ohne Unter- und Überlauf durchgeführt werden kann. Dann kommutiert das nachfolgende Diagramm.



Die multiplikativen Störungen sind dabei wie folgt festgelegt:

$$\begin{array}{rclcl} \widehat{a}_i & = & a_i(1+\kappa_a^{(i)}\epsilon), & |\kappa_a^{(i)}| & \leq & 1/2+\mathcal{O}(\epsilon) \\ \widehat{b}_i & = & b_i(1+\kappa_b^{(i)}\epsilon), & |\kappa_b^{(i)}| & \leq & 3+\mathcal{O}(\epsilon) \end{array} \tag{2.10}$$

$$\begin{array}{rclcl} \widehat{d_i^+} & = & \hat{d}_i^+(1+\kappa_{d^+}^{(i)}\epsilon), & |\kappa_{d^+}^{(i)}| & \leq & 3+\mathcal{O}(\epsilon) \\ \widehat{l_i^+} & = & \hat{l}_i^+(1+\kappa_{l^+}^{(i)}\epsilon), & |\kappa_{l^+}^{(i)}| & \leq & 5+\mathcal{O}(\epsilon) \\ \widehat{s}_i & = & s_i(1+\kappa_s^{(i)}\epsilon), & |\kappa_s^{(i)}| & \leq & 1+\mathcal{O}(\epsilon) \end{array} \tag{2.11}$$

Beweis: Der Satz ist eine Variante einschlägiger Theoreme aus [29, 37, 38]. Wir betrachten zunächst die Rechnung in exakter Arithmetik:

$$\begin{array}{rcll} \widehat{d_i^+} & = & \widehat{s}_i + \widehat{a}_i^2 & (A1) \\ \widehat{l_i^+} & = & \frac{\widehat{a}_i\widehat{b}_i}{\widehat{s}_i+\widehat{a}_i^2} & (B1) \\ \widehat{s}_{i+1} & = & \frac{\widehat{s}_i}{\widehat{s}_i+\widehat{a}_i^2}\widehat{b}_i^2 - \mu^2 & (C1) \end{array}$$

Unter Verwendung des Modells der Gleitpunktarithmetik (1.3.2) erhalten wir weiter

$$\begin{array}{rcll} \hat{d}_i^+ & = & (\hat{s}_i + a_i^2)/((1+\epsilon_+)(1+\epsilon_{a^2})) & (A2) \\ \hat{l}_i^+ & = & \frac{a_ib_i}{\hat{s}_i+a_i^2}(1+\epsilon_+)(1+\epsilon_{a^2})(1+\epsilon_/)(1+\epsilon_{ab}) & (B2) \end{array}$$

Die Fehleranalyse

$$\hat{s}_{i+1} = \left(\frac{\hat{s}_i}{\hat{d}_i^+} \cdot b_i^2(1+\epsilon_{//})(1+\epsilon_*)(1+\epsilon_{b^2}) - \mu^2 \right) /(1+\epsilon_{\mu^2}^{(i+1)})$$

schreiben wir um zu

$$(1+\epsilon_{\mu^2}^{(i+1)})\hat{s}_{i+1} = \frac{\hat{s}_i}{\hat{s}_i+a_i^2}b_i^2(1+\delta_b) - \mu^2 \quad (C2)$$

mit $(1+\delta_b) = (1+\epsilon_+)(1+\epsilon_{a^2})(1+\epsilon_{//})(1+\epsilon_*)(1+\epsilon_{b^2})$. Zur Vereinfachung lassen wir dabei mit Ausnahme von $\epsilon_{\mu^2}^{(i+1)}$ die entsprechenden Laufindizes weg. Mit den Störungsansätzen

$$\begin{array}{rcl} \widehat{a}_i & = & a_i\sqrt{1+\epsilon_{\mu^2}^{(i)}} \\ \widehat{b}_i & = & b_i\sqrt{1+\delta_b} \\ \widehat{s}_i & = & \hat{s}_i(1+\epsilon_{\mu^2}^{(i)}) \\ \widehat{d_i^+} & = & \hat{d}_i^+(1+\epsilon_{\mu^2}^{(i)})(1+\epsilon_+)(1+\epsilon_{a^2}) \\ \widehat{l_i^+} & = & \hat{l}_i^+(1+\delta_{l^+}) \end{array}$$

lassen sich die Äquivalenzen

$$(A1) \Leftrightarrow (A2) \quad \text{und} \quad (C1) \Leftrightarrow (C2)$$

leicht überprüfen. Die Wahl von

$$(1+\delta_{l^+}) = \frac{1}{(1+\epsilon_/)(1+\epsilon_{ab})}\sqrt{\frac{(1+\epsilon_{//})(1+\epsilon_*)(1+\epsilon_{b^2})}{(1+\epsilon_+)(1+\epsilon_{a^2})(1+\epsilon_{\mu^2}^{(i)})}} = 1+5\epsilon+\mathcal{O}(\epsilon^2)$$

ergibt sich durch den Nachweis von $(B1) \Leftrightarrow (B2)$:

$$\begin{aligned}
(B1) \quad &\Leftrightarrow \quad \widehat{l}_i^+ = \frac{\widehat{a}_i \widehat{b}_i}{\widehat{s}_i + \widehat{a}_i^2} \\
&\Leftrightarrow \quad \hat{l}_i^+(1+\delta_{l^+}) = \frac{a_i\sqrt{1+\epsilon_{\mu^2}^{(i)}}\, b_i \sqrt{1+\delta_b}}{\hat{s}_i(1+\epsilon_{\mu^2}^{(i)}) + a_i^2(1+\epsilon_{\mu^2}^{(i)})} \\
&\Leftrightarrow \quad \hat{l}_i^+ = \frac{a_i b_i}{\hat{s}_i + a_i^2}\left(\frac{\sqrt{1+\delta_b}}{\sqrt{1+\epsilon_{\mu^2}^{(i)}}} \cdot \frac{1}{(1+\delta_{l^+})}\right) \\
&\Leftrightarrow \quad (B2)
\end{aligned}$$

Der letzte Schritt wird durch das Nachrechnen von

$$\frac{\sqrt{1+\delta_b}}{\sqrt{1+\epsilon_{\mu^2}^{(i)}}} \cdot \frac{1}{(1+\delta_{l^+})} = (1+\epsilon_+)(1+\epsilon_{a^2})(1+\epsilon_/)(1+\epsilon_{ab})$$

gezeigt. □

Bemerkung 2.3
Auch für die übrigen Faktorisierungsalgorithmen können wir eine gemischte Stabilitätsanalyse mit kleinen multiplikativen Störungsansätzen durchführen. Vergleiche [29, 37, 38]. Im Anhang B präsentieren wir weitere Rundungsfehleranalysen für die allgemeineren BABE-Faktorisierungen.

Breakdowns

Für gewisse Shift-Parameter sind Faktorisierungen der Translationen nicht möglich. Wir untersuchen die Gründe und mögliche Abhilfen nun genauer.

Satz 2.4 (Cauchy Interlacing Theorem)
Sei H symmetrisch und für $r = 1 : n$ die entsprechende Spektralzerlegung $H(1:r) = Q^{(r)}\Lambda^{(r)}(Q^{(r)})^T$ gegeben. Dann gilt für $r = 2 : n$

$$\lambda_1^{(r)} \leq \lambda_1^{(r-1)} \leq \lambda_2^{(r)} \leq \ldots \leq \lambda_{r-1}^{(r)} \leq \lambda_{r-1}^{(r-1)} \leq \lambda_r^{(r)}$$

Beweis: Siehe [44], Seite 396. □

Bemerkung 2.5 (Über Minoren)
Nach Definition 1.5 ist mit $\text{minor}(A, j)$ die Matrix bezeichnet, die durch Streichen der j-ten Zeile und Spalte von A entsteht. Wir fassen einige Eigenschaften von Minoren zusammen:

1. $A(1 : n-1, 1 : n-1) = \text{minor}(A, n)$
2. $A(1 : r, 1 : r)$ mit $r = n-1 : -1 : 1$ entsteht durch $(n-r)$-maliges Bilden von Minoren wie unter 1.
3. Das $(n-r)$-malige Streichen beliebiger Zeilen und Spalten — jeweils mit identischem Index — kann mit Hilfe einer geeigneten Permutationsmatrix P_r dargestellt werden:

   $$A_s = (P_r A P_r^T)(1 : r, 1 : r)$$

   Dabei sei s ein Vektor der Länge $(n-r)$, der die Nummern der gestrichenen Zeilen und Spalten enthält. Die Eigenwerte von A_s heißen *Ritz*-Werte, vgl. Abschnitt 1.4.1.
4. Jeder Minor von T_{GK} ist singulär.

Beweis: Um die letzte Aussage zu zeigen, streichen wir die j-te Zeile und Spalte nicht, sondern setzen die entsprechenden Elemente der Golub-Kahan Matrix auf Null. In der zugehörigen Bidiagonalmatrix B ist dann ein Diagonalelement $a_i = 0$ und damit wegen $0 = \prod_{i=1}^n a_i = \det(B) = \prod_{i=1}^n \sigma_i$ mindestens $\sigma_1 = 0$. Damit sind mindestens zwei Eigenwerte von der modifizierten Golub-Kahan Matrix Null. Entfernen wir nun die j-te Zeile und Spalte, ist nach Satz 2.4 ein Eigenwert des Minors Null. Der Minor der modifizierten Matrix entspricht aber $\text{minor}(T_{GK}, j)$. □

Bemerkung 2.6 (Breakdowns bei Faktorisierungen)
Wenn der Shift τ genau als ein Eigenwert von $T(1:r,1:r)$ (mit $r<n$) gewählt wird, ist wegen $d_r^+ = 0$ eine LDL^T-Faktorisierung nicht möglich.[2] Wir sprechen dann von einem *Breakdown* der Faktorisierung. Gemäß Bemerkung 2.5 tritt diese Situation beispielsweise auf, wenn wir Algorithmus 2.2 mit $\mu = 0$ ausführen wollen.
Liegt τ sehr nahe bei einem Eigenwert von $T(1:r,1:r)$, so kann wegen Rundungsfehlern ebenfalls ein Pivotelement d_r^+ auf Null gesetzt werden.
Wir können aber trotzdem versuchen, noch möglichst viel Information über die Faktorisierung zu erhalten, indem wir die in [2] festgelegten Rechenregeln mit dem Symbol ∞ verwenden. Eine modifizierte Variante der linken Seite von Algorithmus 2.1 stellt beispielsweise folgende Sequenz auf:

$$\hat{d}_r^+ = 0, \quad \hat{l}_r^+ = \hat{s}_{r+1} = \hat{d}_{r+1}^+ = \infty, \quad \hat{l}_{r+1}^+ = 0, \quad \hat{s}_{r+2} = b_{r+1}^2 - \mu^2 \tag{2.12}$$

Ist $\hat{d}_r^+$ das einzige auftretende Null-Pivotelement (mit $r<n$), so gilt

$$L^+D^+(L^+)^T = \begin{bmatrix} & \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \infty \end{bmatrix} & \\ (T-\tau I)_{(1:r)} & & \\ & & \\ [0,\ \dots,0,\infty] & \infty & [0,0,\ \dots,0] \\ & \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} & \\ & & (T-\tau I)_{(r+2:n)} \end{bmatrix} \tag{2.13}$$

Dabei steht $(T-\tau I)_{(1:r)}$ für die ersten r und $(T-\tau I)_{(r+2:n)}$ für die letzten $n-r-1$ Zeilen und Spalten von $T-\tau I$. Bis auf die $(r+1)$-te Zeile und Spalte sind also alle Daten der LDL^T-Faktorisierung mit gültigen Werten belegt. Eine solche Darstellung macht durchaus Sinn, wenn beispielsweise mit korrespondierenden BABE-Faktorisierungen eine Näherung für einen Eigenvektor bestimmt werden soll, siehe Gleichung (2.4). Die ungültige $(r+1)$-te Zeile kann dann geschickt übersprungen werden [29].
Die Faktorisierung aus Gleichung (2.13) kann natürlich nicht verwendet werden, wenn eine weitere Translation erfolgen soll (z.B. zur Bestimmung neuer Eigenwerte, wenn τ in einem Cluster liegt). Es ist offensichtlich, daß sie als Kandidatin für eine RRR *a priori* ausscheidet.

Wir wenden uns nun der Frage zu, unter welchen Umständen eine Faktorisierung eine RRR einer Matrix darstellt.

[2] Zur Vereinfachung sei r dabei kleinstmöglich gewählt, d.h. daß alle vorausgehenden Pivotelemente von Null verschieden sind.

2.4 Bestimmung einer RRR

Der RRR-Algorithmus verlangt Approximationen an die Eigenwerte von T und ihren Translationen mit hoher relativer Genauigkeit. Ein kritisches Problem ist also die Erzeugung von relativ robusten Repräsentationen (RRR).

Eine RRR für die Eingangsmatrix T

Der RRR-Algorithmus ist ein Verfahren zur Näherung von *Eigenvektoren*. Es ist prinzipiell möglich, daß für die Eingangsmatrix T selbst keine RRR gefunden werden kann, um hinreichend genaue Approximationen an Eigenwerte zu erhalten. Da Eigenvektoren aber invariant unter Translation sind, dürfen wir einen Shift μ so wählen, daß eine Cholesky-Zerlegung $T+\mu I = B^T B$ existiert. Die Bestimmung der Singulärwerte von B erfolgt dann durch Anwendung des QD-Verfahrens (siehe Abschnitt 1.5.3) oder des Bisektionsverfahrens (siehe Abschnitt 1.5.4). Sehr präzise Näherungen an Eigenwerte von $T+\mu I$ erhalten wir dann durch Quadrieren. Wir nehmen also o.E. an, daß Approximationen für die Eigenwerte der Eingangsmatrix T mit hoher relativer Genauigkeit bestimmt werden können.

Wann ist $L^+D^+(L^+)^T$ eine RRR?

Sei $\mathcal{T}$ eine Translation von T. Anstelle der üblichen Darstellung $L^+D^+(L^+)^T$ verwenden wir

$$\mathcal{T} = \mathcal{L}\Omega\mathcal{L}^T \quad \text{mit Eigenpaar} \quad (\lambda, q) \tag{2.14}$$

Dabei entsteht $\mathcal{L} = L^+\sqrt{|D^+|}$, indem für $i = 1 : n$ die i-te Spalte von L^+ mit $\sqrt{|d_i^+|}$ multipliziert wird. Die Diagonalmatrix $\Omega = \texttt{diag}([\omega_1, \ \ldots, \omega_n])$ mit $\omega_i = \text{sign}(d_i^+)$ enthält die Information über die Vorzeichen von D^+.
Ein multiplikativer Störungsansatz für $\mathcal{L}$ kann durch zwei diagonale Matrizen $\mathcal{X}_1$ und $\mathcal{X}_2$ beschrieben werden:

$$\bar{\mathcal{T}} = \mathcal{X}_1\mathcal{L}\mathcal{X}_2\Omega\mathcal{X}_2\mathcal{L}^T\mathcal{X}_1 \quad \text{mit Eigenwert} \quad \bar{\lambda}$$

Sei ferner

$$\bar{\bar{\mathcal{T}}} = \mathcal{L}\mathcal{X}_2\Omega\mathcal{X}_2\mathcal{L}^T \quad \text{mit Eigenwert} \quad \bar{\bar{\lambda}}$$

Mit der Abkürzung $\Delta_1 = \mathcal{X}_1^T\mathcal{X}_1 - I$ folgt aus Satz 1.23

$$\left|\frac{\bar{\lambda} - \bar{\bar{\lambda}}}{\bar{\lambda}}\right| \leq \|\Delta_1\| \tag{2.15}$$

Diese Ungleichung können wir umschreiben zu

$$\bar{\lambda}(1 - \text{sign}(\bar{\lambda})\|\Delta_1\|) \leq \bar{\bar{\lambda}} \leq \bar{\lambda}(1 + \text{sign}(\bar{\lambda})\|\Delta_1\|) \tag{2.16}$$

Wie in [27, 29] definieren wir als *relative Konditionszahl*

$$\kappa_{\text{rel}}(\lambda) := \left| \frac{q^T \mathcal{L}\mathcal{L}^T q}{q^T \mathcal{L}\Omega\mathcal{L}^T q} \right| = \left| \frac{q^T L^+ |D^+| (L^+)^T q}{\lambda q^T q} \right| \tag{2.17}$$

und erhalten mit $\Delta_2 = \mathcal{X}_2 - I$ nach [29], Theorem 5.2.1

$$\left| \frac{\bar{\bar{\lambda}} - \lambda}{\lambda} \right| \leq \kappa_{\text{rel}}(\lambda) \cdot 2\|\Delta_2\| + \mathcal{O}((\kappa_{\text{rel}}(\lambda) \cdot 2\|\Delta_2\|)^2) =: h \tag{2.18}$$

Analog zum Übergang von (2.15) nach (2.16) notieren wir (2.18) als

$$\lambda(1 - \text{sign}(\lambda)h) \leq \bar{\bar{\lambda}} \leq \lambda(1 + \text{sign}(\lambda)h) \tag{2.19}$$

Die Kombination der linken Abschätzung aus (2.16) mit der rechten Abschätzung aus (2.19) ergibt

$$\begin{array}{rrcl} & \bar{\lambda}(1 - \text{sign}(\bar{\lambda})\|\Delta_1\|) & \leq & \lambda(1 + \text{sign}(\lambda)h) \\ \Leftrightarrow & (\bar{\lambda} - \lambda)(1 - \text{sign}(\bar{\lambda})\|\Delta_1\|) & \leq & \lambda(\text{sign}(\lambda)h + \text{sign}(\bar{\lambda})\|\Delta_1\|) \\ \Leftrightarrow & (\bar{\lambda} - \lambda)(1 - \text{sign}(\bar{\lambda})\|\Delta_1\|) & \leq & |\lambda|(h + \text{sign}(\bar{\lambda})\text{sign}(\lambda)\|\Delta_1\|) \\ \Leftrightarrow & \frac{\bar{\lambda}-\lambda}{|\lambda|} & \leq & \frac{h+\text{sign}(\bar{\lambda})\text{sign}(\lambda)\|\Delta_1\|}{1-\text{sign}(\bar{\lambda})\|\Delta_1\|} \end{array}$$

Wir betrachten die Beträge der letzten Ungleichung und lösen die Vorzeichen $\text{sign}(\lambda)$ und $\text{sign}(\bar{\lambda})$ auf der rechten Seite durch einfache Abschätzungen nach oben auf. Ersetzen wir h durch die Terme aus (2.18), so erhalten wir schließlich

$$\left| \frac{\bar{\lambda} - \lambda}{\lambda} \right| \leq \frac{\|\Delta_1\| + \kappa_{\text{rel}}(\lambda) \cdot 2\|\Delta_2\| + \mathcal{O}((\kappa_{\text{rel}}(\lambda) \cdot 2\|\Delta_2\|)^2)}{1 - \|\Delta_1\|} \tag{2.20}$$

Zur Vereinfachung nehmen wir an, daß die relativen Störungen der Elemente von $\mathcal{L}$ durch $\|\Delta_1\| \leq \eta$ und $2\|\Delta_2\| \leq \eta$ mit $\eta = \mathcal{O}(\epsilon)$ kontrollierbar bleiben:

$$\left| \frac{\bar{\lambda} - \lambda}{\lambda} \right| \leq \frac{(1 + \kappa_{\text{rel}}(\lambda)) \cdot \eta + \mathcal{O}((\kappa_{\text{rel}}(\lambda) \cdot \eta)^2)}{1 - \eta} \tag{2.21}$$

Die relative Konditionszahl $\kappa_{\text{rel}}(\lambda)$ nimmt also entscheidenden Einfluß auf die Qualität einer Approximation $\bar{\lambda}$ an λ.
Wir beachten dabei, daß $\kappa_{\text{rel}}(\lambda_j)$ für $j = f : l$ unterschiedliche Werte annehmen kann. Liegt sie unterhalb einer zu spezifizierenden Toleranzgrenze κ_{tol}, so betrachten wir $\mathcal{L}\Omega\mathcal{L}^T = L^+ D^+ (L^+)^T$ als partielle RRR für den j-ten Eigenwert. Für $\Omega = \pm I$ nimmt die relative Konditionszahl ihren minimalen Wert 1 an.

Ein Kriterium für eine RRR (*a posteriori*)

In Algorithmus 2.4 skizzieren wir die Vorgehensweise, um zu einer gegebenen symmetrisch tridiagonalen Matrix in der Darstellung LDL^T einen geeigneten Shift τ und eine Faktorisierung $LDL^T - \tau I = L^+D^+(L^+)^T$ zu ermitteln, die folgende Eigenschaften hat:

- $L^+D^+(L^+)^T$ ist eine partielle RRR für alle Eigenwerte, die zu einem gewissen Cluster gehören.
- Durch den Shift τ werden die relativen Distanzen der interessierenden Eigenwerte deutlich vergrößert.

Algorithmus 2.4 Bestimmung einer RRR.

Gegeben: D, L als RRR einer Translation von T, deren Eigenwerte λ_j für $j = f : l$ einen Cluster bilden.
Ein Schwellenwert κ_{tol} (z.B. 100).

Gesucht: Ein Shift τ, so daß D^+, L^+ eine RRR von $LDL^T - \tau I$ beschreiben, deren Eigenwerte besser isoliert sind.

1: **repeat**
2: Wähle τ.
3: Faktorisiere $LDL^T - \tau I = L^+D^+(L^+)^T$.
4: Bestimme Eigenwertapproximationen $\bar{\lambda}_j^+$ für $j = f : l$.
5: Bestimme $\kappa_{max} = \max\{\kappa_{\text{rel}}(\bar{\lambda}_j^+) \,|\, j = f : l\}$.
6: **until** $\kappa_{max} \leq \kappa_{tol}$

An dieser Stelle ist zu betonen, daß die Bestimmung der Eigenwertnäherungen in Zeile 4 der weitaus aufwendigste Teil des Verfahrens ist. Mit den Größen aus Zeile 5 läßt sich also nur *a posteriori* bestimmen, ob der Shift τ für eine RRR geeignet ist oder nicht. Sollte das Abbruchkriterium (Zeile 6) nicht erfüllt sein, ist die zuvor investierte Arbeit größtenteils vergeblich. Für diesen kritischen Teil einer effizienten Implementierung des RRR-Algorithmus sind also Antworten auf verschiedene Fragen gesucht. Wir diskutieren hier nur die erste und verlagern Lösungsvorschläge für die übrigen Probleme auf die nachfolgenden Abschnitte.

1. Können wir *a priori* bestimmen, ob ein Shift τ zu einer RRR führt?
2. Wie bestimmen wir die $\bar{\lambda}_j^+$ schnell und (möglichst) genau?
   – Abschnitt 3.4.3
3. Wie können wir die relativen Konditionszahlen numerisch stabil bestimmen?
   – Abschnitt 3.4.5

Eine Heuristik für eine RRR (*a priori*)

Unser Ziel ist nun die Entwicklung einer Strategie, die mit wenig Aufwand die Bestimmung von geeigneten Shift-Parametern τ zur Bearbeitung eines Clusters $\lambda_f \leq \cdots \leq \lambda_l$ von Eigenwerten erlaubt. Zunächst einmal sind für $f = 1$ bzw. $l = n$ minimale relative Konditionszahlen gegeben, wenn wir $\tau < \lambda_1$ bzw. $\tau > \lambda_n$ wählen, da dann $\Omega = I$ bzw. $\Omega = -I$. Diese einfache Strategie ist in der aktuellen Version der LAPACK-Routine `DSTEGR` implementiert. Für innere Cluster ($f > 1$ bzw. $l < n$) ist es möglich, die relativen Konditionszahlen abzuschätzen:

Satz 2.7 (Schranken für $\kappa_{\mathrm{rel}}(\lambda_j^+)$)
Sei $\lambda_{f-1} \ll \tau < \lambda_f$, also τ nahe am linken Rand des Clusters gewählt, so daß die Zerlegung (2.14) exisitiert und deren Eigenwerte λ_j^+ für $j = f : l$ bekannt sind. Dann gilt:

$$\sum_{j=f:l} \kappa_{\mathrm{rel}}(\lambda_j^+) \leq (l - f + 1) + \frac{2}{\tau - \lambda_{f-1}} \sum_{d_i^+ < 0} \left\| L^+ \sqrt{|D^+|} e_i \right\|^2$$

Eine ähnliche Abschätzung ergibt sich für eine Wahl $\lambda_l < \tau \ll \lambda_{l+1}$ am rechten Rand des Clusters.

Beweis: Siehe [27]. □

Wählen wir also τ nahe bei, aber noch außerhalb des Clusters, so können wir moderate relative Konditionszahlen erwarten, wenn die einfach zu bestimmenden Größen $\tau - \lambda_{f-1}$ bzw. $\lambda_{l+1} - \tau$ und $\left\| L^+ \sqrt{|D^+|} e_i \right\|^2 = |d_i^+|(1 + (l_i^+)^2)$ eine kleine obere Schranke ergeben. Sie wird beeinflußt durch den *absoluten Abstand zum nächstgelegen Eigenwert außerhalb des Clusters* — nachfolgend kurz *absoluter Abstand* genannt — und durch das *Elementwachstum*

$$elg(T, \tau) := \max_i \left\{ |d_i^+|(1 + (l_i^+)^2) \right\} \tag{2.22}$$

Bevor wir diese Schranken diskutieren, weisen wir darauf hin, daß in [27] noch schärfere Abschätzungen angegeben sind. Die dort verwendeten Größen entziehen sich jedoch einer einfachen numerischen Berechnung.

Beispiel 2.8 (Abhängigkeit vom absoluten Abstand)
In diesem Beispiel illustrieren wir die Abhängigkeit der Schranken vom Abstand $\tau - \lambda_{f-1}$ bzw. $\lambda_{l+1} - \tau$. Zur Vereinfachung gehen wir davon aus, daß nur moderates Elementwachstum auftritt. Eine Matrix T besitze Eigenwerte

$$\Lambda = \mathtt{diag}([\epsilon, 1 + \epsilon, 1 + \sqrt{\epsilon}, 1 + \sqrt{\epsilon} + \epsilon, 1 + 2\sqrt{\epsilon}, 2])$$

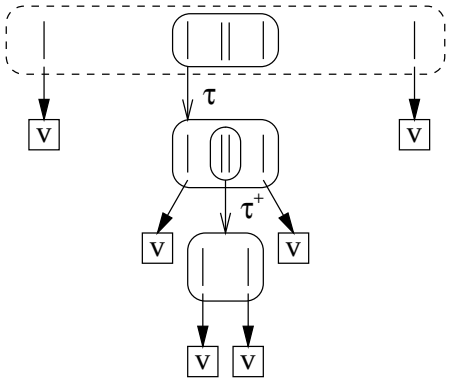


Abbildung 2.3: Cluster $[\epsilon, 1+\epsilon, 1+\sqrt{\epsilon}, 1+\sqrt{\epsilon}+\epsilon, 1+2\sqrt{\epsilon}, 2]$ mit Substruktur.

mit einem Darstellungsbaum wie in Abbildung 2.3. Wählen wir $\tau \approx 1$, so ist der Abstand $\mathcal{O}(1)$. Wenn für die interessierenden Eigenwerte von $T - \tau I$

$$\Lambda^+ = \texttt{diag}([*, \mathcal{O}(\epsilon), \mathcal{O}(\sqrt{\epsilon}), \mathcal{O}(\sqrt{\epsilon}+\epsilon), \mathcal{O}(2\sqrt{\epsilon}), *])$$

ein weiterer Cluster vorliegt, so hat ein Shift-Parameter τ^+, der diesen auflösen soll, nun einen Abstand von $\mathcal{O}(\sqrt{\epsilon})$. Dadurch kann die Abschätzung aus Satz 2.7 erheblich schlechter ausfallen.

Beispiel 2.9 (Golub-Kahan Matrix vs. Normalengleichungen)
In diesem Beispiel wird verdeutlicht, daß auch das Elementwachstum ungünstigen Einfluß auf die Abschätzung haben kann. Eine Bidiagonalmatrix B mit $a_1 = \mathcal{O}(1)$ besitze Singulärwerte

$$[\sqrt{\epsilon}(1+\epsilon), \sqrt{\epsilon}+\epsilon, 2\sqrt{\epsilon}, 2\sqrt{\epsilon}+\epsilon, 1]$$

Abbildung 3.4 auf Seite 68 verdeutlicht die Cluster-Struktur von T_{GK}. Wählen wir als Shift-Parameter $\tau = \sqrt{\epsilon}$, so ist der Abstand zum nächstgelegenen Eigenwert mit $\tau - (-\sqrt{\epsilon}(1+\epsilon)) \approx 2\sqrt{\epsilon}$ bereits sehr klein. Betrachten wir ferner die ersten Schritte einer mit Algorithmus 2.2 durchgeführten Faktorisierung $T_{GK} - \tau I = \tilde{L}^+ \tilde{D}^+ (\tilde{L}^+)^T$, also

$$\tilde{d}_1^+ = -\epsilon^{1/2} < 0, \quad \tilde{l}_1^+ = -a_1\epsilon^{-1/2}, \quad \tilde{d}_2^+ = -a_1^2\epsilon^{-1/2} - \epsilon^{1/2} < 0, \quad \tilde{l}_2^+ = -\frac{b_1\epsilon^{1/2}}{a_1^2 - \epsilon}$$

so ergibt sich

$$\begin{aligned} |\tilde{d}_1^+|(1+(\tilde{l}_1^+)^2) &= \epsilon^{1/2} + a_1^2\epsilon^{-1/2} \\ |\tilde{d}_2^+|(1+(\tilde{l}_2^+)^2) &= (\epsilon^{1/2} + a_1^2\epsilon^{-1/2})\left(1 + \frac{b_1^2\epsilon}{(a_1^2-\epsilon)^2}\right) \end{aligned}$$

und damit ein Elementwachstum

$$elg(T_{GK}, \sqrt{\epsilon}) \geq a_1^2 \epsilon^{-1/2} = \mathcal{O}(\epsilon^{-1/2})$$

Viel günstiger erscheint hier das Arbeiten mit $B^T B$. Mit $\tau = \sqrt{\epsilon}$ ist eine Faktorisierung $B^T B - \tau^2 I = \hat{L}^+ \hat{D}^+ (\hat{L}^+)^T$ positiv definit. Vergleichen wir ferner die ersten Schritte der Faktorisierung durch Algorithmus 2.1, erkennen wir

$$\hat{d}_1^+ = -\tilde{d}_1^+ \tilde{d}_2^+ \quad \text{und} \quad \hat{l}_1^+ = -\tilde{l}_1^+ \tilde{l}_2^+$$

und insbesondere mit

$$|\tilde{d}_1^+ \tilde{d}_2^+|(1 + (\tilde{l}_1^+ \tilde{l}_2^+)^2) \;\; = \;\; |a_1^2 - \epsilon| + \text{sign}(a_1^2 - \epsilon) a_1 b_1$$

die Andeutung eines deutlich kleineren Elementwachstums im Vergleich zur Faktorisierung der Golub-Kahan Matrix (zumindest in den ersten Komponenten).

Bemerkung 2.10 (Beispiel 2.9 deutet einen Trend an.)
Für eine Reihe von Testproblemen ist die Beobachtung aus Beispiel 2.9 typisch: Die Faktorisierung von $T_{GK} - \mu I$ ist mit großem Elementwachstum verbunden, während die äquivalenten Faktorisierungen von $B^T B - \mu^2 I$ oder $BB^T - \mu^2 I$ in dieser Hinsicht moderat bleiben. Obwohl das Elementwachstum alleine noch keine Aussage darüber macht, ob eine Faktorisierung eine RRR ist oder nicht, sind wir bei der *a priori* Auswahl der Shift-Kandidaten teilweise erheblich eingeschränkt, wenn wir mit T_{GK} anstelle der Normalengleichungen arbeiten wollen. Wir werden diese in der Literatur noch nicht diskutierte Problematik in Abschnitt 3.1 weiter verfolgen.

Bemerkung 2.11 (Ist Satz 2.7 zu pessimistisch?)
Es ist wichtig zu erwähnen, daß in vielen Fällen die *a posteriori* Überprüfung der Konditionszahlen $\kappa_{\text{rel}}(\lambda_j^+) = \mathcal{O}(1)$ ergibt, obwohl eine scheinbar ungünstige Schranke aufgrund von sehr großem Elementwachstum oder nur sehr kleinem absoluten Abstand zum nächstgelegenen Eigenwert vorliegt. Das einzig sichere Kriterium ist also nach wie vor die *a posteriori* Bestimmung der relativen Konditionszahlen. Der Nutzen von Satz 2.7 liegt darin, in vielen Fällen gute Vorhersagen für geeignete Shift-Parameter τ zu liefern.

Bemerkung 2.12 (Reihenfolge der Shift-Kandidaten)
Als Reihenfolge für die Shift-Kandidaten ist zu empfehlen, zunächst eine Wahl „etwas kleiner“ als λ_f bzw. „etwas größer“ als λ_l zu versuchen. Sollte das *a posteriori* Kriterium der relativen Konditionszahlen nicht erfüllt sein, so probieren wir nacheinander $\lambda_f, \lambda_l, \lambda_{f+1}, \; \ldots, \lambda_{l-1}$ als Shift-Kandidaten

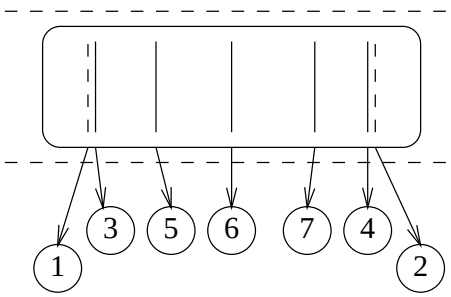


Abbildung 2.4: Zur Reihenfolge der Shift-Kandidaten.

durch, vgl. Abbildung 2.4. Durch gezielte Konstruktion von Testmatrizen lassen sich Situationen erzeugen, in denen *alle* Shift-Kandidaten scheitern. Eine Abhilfe für diese allerdings extrem seltenen Fälle beschreiben wir in Kapitel 4 (Stichworte Präkonditionierung und Ideale Systeme).

Bemerkung 2.13 (Andere Faktorisierungen als $L^+D^+(L^+)^T$)
Die Suche nach *a priori* Kriterien muß nicht notwendigerweise auf die LDL^T-Faktorisierung beschränkt bleiben, sondern kann beispielsweise auf allgemeine BABE-Faktorisierungen erweitert werden. Nach [27] sind jedoch zumindest beim Elementwachstum keine großen Variationen beim Übergang von $L^+D^+(L^+)^T$ auf $N_k^+G_k^+(N_k^+)^T$ zu erwarten. Im Rahmen dieser Arbeit durchgeführte numerische Experimente untermauern diese Aussage.

2.5 Zusammenfassung

In Algorithmus 2.5 kombinieren wir nun alle Konzepte. Das Kernstück einer Implementierung des RRR-Verfahrens ist der effiziente Aufbau der Pfade $\mathrm{path}(T, j)$ für $j = 1 : n$. Sind die Pfade bekannt, können wir die entsprechenden Faktorisierungen und Eigenvektorapproximationen leicht reproduzieren. Bei der Parallelisierung des RRR-Algorithmus läßt sich also die entscheidende Information sehr kompakt formulieren und kann in kleinen Datenpaketen versendet werden. Ist im Folgenden klar, welcher Pfad gemeint ist, lassen wir den Index j weg.
Algorithmus 2.5 kann so modifiziert werden, daß nur k ausgewählte Eigenpaare bestimmt werden. Der RRR-Algorithmus hat einen Aufwand von $\mathcal{O}(kn)$, wenn es gelingt, die Tiefe des Darstellungsbaumes $r_{\max} = \max_j\{\mathrm{rec}_j\}$ zu beschränken. Bei einem Großteil der in Abschnitt 4.3 getesteten Matrixklassen ist $r_{max} = 1$. Es kann jedoch vorkommen, daß sich Eigenwerte nicht isolieren lassen (z.B. bedingt durch zu hohe relative Konditionszahlen für alle

Algorithmus 2.5 Eine grobe Beschreibung des RRR-Algorithmus.

Gegeben: Eine Faktorisierung $L^{(0)}D^{(0)}(L^{(0)})^T$ von T sowie mit hoher relativer Genauigkeit bestimmte Näherungen $\bar{\lambda}_1^{(1)} \leq \ldots \leq \bar{\lambda}_n^{(1)}$.

Gesucht: Eine Matrix $\bar{Q}$ mit Näherungen an das Eigensystem von T. Pfade $\text{path}(T, j) := (\nu_j^{(1)}, \ldots, \nu_j^{(\text{rec}_j)})$, die für $j = 1 : n$ die Konstruktion von $\bar{Q}(:, j)$ beschreiben.

1: Beschreibe die Cluster-Struktur des Spektrums in einer Liste $\mathcal{L}$.
2: Bearbeite isolierte Eigenwerte (BABE-Faktorisierung).
3: **while** $\mathcal{L} \neq \emptyset$ **do**
4: Wähle einen Cluster aus $\mathcal{L}$.
5: {Sei r die Ebene im Darstellungsbaum, vgl. Abbildung 2.1. Die (bereits bekannten) Eigenwertnäherungen $\bar{\lambda}_f^{(r)} \leq \ldots \leq \bar{\lambda}_l^{(r)}$ seien bzgl. der Matrix $L^{(r-1)}D^{(r-1)}(L^{(r-1)})^T$ geclustert.}
Wähle einen geeigneten Shift $\nu^{(r)}$.
6: Faktorisiere $L^{(r-1)}D^{(r-1)}(L^{(r-1)})^T - \nu^{(r)}I = L^{(r)}D^{(r)}(L^{(r)})^T$.
7: Bestimme Näherungen an die Eigenwerte $\bar{\lambda}_f^{(r+1)} \leq \ldots \leq \bar{\lambda}_l^{(r+1)}$ von $L^{(r)}D^{(r)}(L^{(r)})^T$.
8: Bestimme Eigenvektorapproximationen $\bar{Q}(:, f : l)$ mit Hilfe von BABE-Faktorisierungen. Prüfe dabei *a posteriori*, ob die mit dem Shift $\nu^{(r)}$ erzeugte Matrix $L^{(r)}D^{(r)}(L^{(r)})^T$ eine partielle RRR bildet.
{Aktualisiere $\text{path}(T, j)$ für $j = f : l$ sowie die Liste $\mathcal{L}$.}
9: **if** $L^{(r)}D^{(r)}(L^{(r)})^T$ eine partielle RRR bildet **then**
10: Gruppiere $\bar{\lambda}_f^{(r+1)}, \ldots, \bar{\lambda}_l^{(r+1)}$ in Cluster und isolierte Elemente.
11: Für isolierte Elemente ergänze $\text{path}(T, j)$ um $\bar{\lambda}_f^{(r+1)}$. Bei Elementen, die zu einem der neuen Cluster gehören, ergänze $\text{path}(T, j)$ um $\nu^{(r)}$.
12: Füge die neuen Cluster in $\mathcal{L}$ ein.
13: **else**
14: Füge den Cluster $\bar{\lambda}_f^{(r)}, \ldots, \bar{\lambda}_l^{(r)}$ wieder in $\mathcal{L}$ ein. Markiere den Shift-Kandidaten $\nu^{(r)}$ als ungeeignet.
15: **end if**
16: **end while**

ermittelten Shift-Kandidaten).
Aber auch unabhängig von der numerischen Sichtweise lassen sich Matrizen konstruieren, bei denen eine große Anzahl von Ebenen im Darstellungsbaum abgearbeitet werden muß. So kann beispielsweise das Spektrum einer Matrix gegeben sein durch

$$\Lambda = \texttt{diag}([\epsilon, 1 \pm 10^{-3}, 1 \pm 10^{-5}, \ \ldots, 1 \pm 10^{-15}, 2])$$

Es liegt also ein Cluster mit 14 Eigenwerten vor. Wählen wir nun Shift-Parameter immer außerhalb der Cluster, so können pro Ebene nur zwei Elemente isoliert werden, vgl. Abbildung 2.5. Auf diese Weise lassen sich leicht weitere Beispiele finden, bei denen der Darstellungsbaum stark anwächst. In Einzelfällen kann also die quadratische Komplexität des RRR-Algorithmus gefährdet sein. Wir betonen jedoch ausdrücklich, daß solche problematischen Matrizen gezielt konstruiert werden müssen, während das Verfahren ansonsten seine hervorragenden Eigenschaften ausspielen kann. Außerdem werden wir in Kapitel 4 zusätzliche Maßnahmen (Stichworte Präkonditionierung und Ideale Systeme) zur Abfederung von Problemfällen vorschlagen.

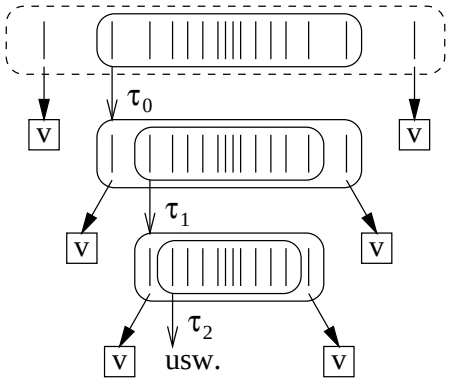


Abbildung 2.5: Cluster $[\epsilon, 1 \pm 10^{-3}, 1 \pm 10^{-5}, \ \ldots \ , 1 \pm 10^{-15}, 2]$ mit Substruktur. Pro Ebene können nur zwei Eigenwerte isoliert werden.

Zur Interpretation des RRR-Algorithmus als Variante der klassischen Inversen Iteration wiederholen wir noch einmal deren definierende Gleichung (1.3)

$$\text{Löse} \quad (T - \bar{\lambda}^{(i-1)} I) \cdot \bar{q}^{(i)} = \zeta^{(i)} \bar{q}^{(i-1)}, \quad i = 1, 2, \ \ldots$$

und stellen dieser eine vergleichbare Vorschrift für die Iterierten aus Algorithmus 2.5 gegenüber:

$$\text{Löse} \quad (N_{k_i}^{(i)} G_{k_i}^{(i)} (N_{k_i}^{(i)})^T) \cdot \bar{q}^{(k_i)} = \zeta^{(i)} \gamma_{k_i}^{(i)} e_{k_i}, \quad i = 1 : \text{rec}$$

Die Normierungen der Lösungsvektoren sind wieder durch die Parameter $\zeta^{(i)}$ angedeutet. Bei der Inversen Iteration wird in jedem Schritt über die rechte Seite $\bar{q}^{(i)}$ und ggf. über den Shift-Parameter $\bar{\lambda}^{(i-1)}$ variiert. Jedoch ist $\bar{\lambda}^{(i-1)} \approx \bar{\lambda}^{(0)}$ in jedem Iterationsschritt, so daß bei Clustern die beschriebenen Probleme auftreten. Diese können i.d.R. nur durch explizites Reorthogonalisieren behoben werden. Außerdem ist das Lösen des LGS entweder nur mit

einer LDL^T- oder einer URU^T-Faktorisierung möglich. Die spezielle Wahl der rechten Seite beim RRR-Algorithmus erlaubt das Verwenden der BABE-Faktorisierungen, so daß jedes LGS mit einem kleinen Residuum und geringerem Aufwand als bei der Inversen Iteration gelöst werden kann. Außerdem werden durch geeignete Shift-Strategien Systeme gelöst, deren Eigenwerte isoliert sind. Meistens bricht das Verfahren bereits mit sehr kleinem rec ab, während bei der Inversen Iteration i.d.R. häufiger iteriert werden muß, bis ein zufriedenstellendes Abbruchkriterium erreicht ist.
Die Übertragung des RRR-Algorithmus auf die **bSVD** ist Gegenstand dieser Dissertation und ausführlich im nächsten Kapitel beschrieben.

Kapitel 3

Kopplungen und die Adaption auf die bidiagonale SVD

In diesem Kapitel erörtern wir Strategien, wie der RRR-Algorithmus auf die **bSVD** anzuwenden ist. Wir stellen zunächst fest, daß der naheliegende Ansatz, die **tSEP** für B^TB, BB^T oder T_{GK} getrennt zu betrachten, in vielen Fällen nicht zum gewünschten Ergebnis führt. Diese Problematik motiviert unsere Strategie, den RRR-Algorithmus explizit nur auf B^TB anzuwenden, während die entsprechenden Faktorisierungen von BB^T implizit mitgeführt werden [49]. Dazu stellen wir eine Reihe von Kopplungsmechanismen bereit, deren störungstheoretischen Eigenschaften und praktische Implementierungen ausführlich diskutiert werden. Anschließend geben wir ein Gerüst für den sogenannten *Kernalgorithmus* an.

3.1 Black-Box Strategien

Die Anwendung des RRR-Algorithmus auf die **bSVD** soll den Genauigkeitsanforderungen genügen, die in Abschnitt 1.3.5 spezifiziert sind: Bei Durchführung in der Gleitpunktarithmetik verlangen wir sowohl numerische Orthogonalität als auch gute Kopplungen der Singulärvektorpaare bzgl. B. Wir konkretisieren zunächst die in Abschnitt 1.3.6 skizzierten Probleme.

3.1.1 Normalengleichungen

Die Zerlegung $B = U\Sigma V^T$ kann verknüpft werden mit den Eigenwertproblemen $B^TB = V\Sigma^2V^T$ und $BB^T = U\Sigma^2U^T$. Wenden wir den RRR-Algorithmus auf beide Probleme *getrennt* an, so erhalten wir Approximationen an U und V, die numerisch orthogonal sind. Sind große Singulärwerte von B stark geclustert, beobachten wir aber andererseits, daß diese Strategie teilweise nur sehr schlecht gekoppelte Singulärvektorpaare erzeugt ($\|B\bar{V}(:,j) - \bar{\sigma}_j\bar{U}(:,j)\|$ wird zu groß). Zur Ergründung der Ursachen gehen wir in vier Schritten vor:

1. Wir formulieren einen Satz über die Eigenwerte der LDL^T-Faktorisierungen von $B^TB - \mu^2I$ und $BB^T - \mu^2I$.
2. Wir diskutieren dessen Konsequenzen für den RRR-Algorithmus.
3. Wir geben ein ausführliches numerisches Beispiel.
4. Zusammenfassung.

1. Eigenwerte von $B^TB - \mu^2I$ und $BB^T - \mu^2I$

Wir tragen zunächst die Fakten bei Rechnung in exakter Arithmetik zusammen. Die Matrix B besitze einen Cluster $\sigma_f \leq \ldots \leq \sigma_l$ von Singulärwerten. Wir wählen einen Shift-Parameter „nahe bei“ einem Singulärwert aus diesem Cluster:

$$\mu^2 = (1 + \kappa_0\epsilon)\sigma_j^2 \tag{3.1}$$

Der Shift-Parameter μ ist typischerweise eine im Gleitpunktsystem darstellbare Zahl, die σ_j approximiert. Dann ist

$$\hat{\lambda}_j^+ = \sigma_j^2 - \mu^2 \text{ Eigenwert von } B^TB - \mu^2I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$$

und

$$\check{\lambda}_j^+ = \sigma_j^2 - \mu^2 \text{ Eigenwert von } BB^T - \mu^2I = \check{L}^+\check{D}^+(\check{L}^+)^T$$

Die Daten von $\hat{D}^+$, $\hat{L}^+$, $\check{D}^+$ und $\check{L}^+$ seien dabei mit Algorithmus 2.1 generiert.

Beim Übergang zur Gleitpunktarithmetik greifen wir nun auf die gemischte relative Stabilitätsanalyse aus Abschnitt 2.3, Satz 2.2 und Bemerkung 2.3 zurück. Analog zu der Zerlegung $B^TB - \mu^2I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$ arbeiten wir bei der Faktorisierung $BB^T - \mu^2I = \check{L}^+\check{D}^+(\check{L}^+)^T$ mit Störungen $\breve{B}$ für B sowie $\breve{D}^+$ und $\breve{L}^+$ für $\check{D}^+$ und $\check{L}^+$. Wir verknüpfen diese Ergebnisse nun mit der Fehleranalyse für Eigenwerte, deren Matrix eine partielle relativ robuste Repräsentation (RRR) besitzt.

Satz 3.1 (Über die Fehler der Eigenwerte $\hat{\lambda}_j^+$ und $\check{\lambda}_j^+$)
Wir betrachten die kommutierenden Diagramme

$$\begin{array}{ccc} B,\ (\sigma_j^2) & \xrightarrow[\text{Alg.2.1, links}]{\text{berechnet}} & \hat{D}^+, \hat{L}^+,\ (\hat{\lambda}_j^+) \\ \Big\downarrow \text{Störung} & & \Big\uparrow \text{Störung} \\ \widehat{B},\ (\widehat{\sigma}_j^2) & \xrightarrow[\text{Alg.2.1, links}]{\text{exakt}} & \widehat{D}^+, \widehat{L}^+,\ (\widehat{\lambda}_j^+) \end{array}$$

der Faktorisierung $B^TB - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und

$$\begin{array}{ccc} B,\ (\sigma_j^2) & \xrightarrow[\text{Alg.2.1, rechts}]{\text{berechnet}} & \check{D}^+, \check{L}^+,\ (\check{\lambda}_j^+) \\ \Big\downarrow \text{Störung} & & \Big\uparrow \text{Störung} \\ \breve{B},\ (\breve{\sigma}_j^2) & \xrightarrow[\text{Alg.2.1, rechts}]{\text{exakt}} & \breve{D}^+, \breve{L}^+,\ (\breve{\lambda}_j^+) \end{array}$$

der Faktorisierung $BB^T - \mu^2 I = \check{L}^+\check{D}^+(\check{L}^+)^T$. Die Singulärwerte der Störungen $\widehat{B}$ und $\breve{B}$ von B beschreiben wir durch

$$\widehat{\sigma}_j^2 = \sigma_j^2(1 + \widehat{\kappa}_1^{(j)}\epsilon) \quad \text{und} \quad \breve{\sigma}_j^2 = \sigma_j^2(1 + \breve{\kappa}_1^{(j)}\epsilon)$$

Wir setzen ferner voraus, daß $\hat{D}^+$ und $\hat{L}^+$ sowie $\check{D}^+$ und $\check{L}^+$ jeweils eine partielle RRR für die interessierenden Eigenwerte bilden, so daß also

$$\hat{\lambda}_j^+ = \widehat{\lambda}_j^+(1 + \widehat{\kappa}_2^{(j)}\epsilon) \quad \text{und} \quad \check{\lambda}_j^+ = \breve{\lambda}_j^+(1 + \breve{\kappa}_2^{(j)}\epsilon) \quad \text{für} \ \ j = f:l$$

Dann gilt für $j = f:l$

$$\left|\hat{\lambda}_j^+ - \check{\lambda}_j^+\right| = \mathcal{O}(\hat{\lambda}_j^+\epsilon) + \mathcal{O}(\check{\lambda}_j^+\epsilon) + \mathcal{O}(\sigma_j^2\epsilon) = \mathcal{O}(\sigma_j^2\epsilon) \tag{3.2}$$

Beweis: Die Darstellung der Eigenwerte können wir umformen zu

$$\begin{aligned} \hat{\lambda}_j^+ &= \widehat{\lambda}_j^+(1 + \widehat{\kappa}_2^{(j)}\epsilon) \\ &= (\widehat{\sigma}_j^2 - \mu^2)(1 + \widehat{\kappa}_2^{(j)}\epsilon) \\ &= (\sigma_j^2 - \mu^2)(1 + \widehat{\kappa}_2^{(j)}\epsilon) + \sigma_j^2\widehat{\kappa}_1^{(j)}\epsilon + \mathcal{O}(\sigma_j^2\epsilon^2) \end{aligned}$$

und entsprechend

$$\check{\lambda}_j^+ = (\sigma_j^2 - \mu^2)(1 + \breve{\kappa}_2^{(j)}\epsilon) + \sigma_j^2\breve{\kappa}_1^{(j)}\epsilon + \mathcal{O}(\sigma_j^2\epsilon^2)$$

Innerhalb des Clusters der Singulärwerte dominiert aber σ_j^2 gegenüber $\sigma_j^2 - \mu^2 \approx \hat{\lambda}_j^+ \approx \check{\lambda}_j^+$. □

2. Konsequenzen für den RRR-Algorithmus

Wir halten zunächst fest, daß $\sigma_j^2 - \mu^2$ meistens keine gute Wahl ist, um den Eigenwert $\hat{\lambda}_j^+$ der Faktorisierung $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ auf die benötigte hohe relative Genauigkeit zu approximieren. Wegen der Gleichungen (3.1) und (3.2) können sich insbesondere $\sigma_j^2 - \mu^2$ und $\hat{\lambda}_j^+$ in allen signifikanten Stellen unterscheiden.
Die kleinsten Eigenwerte von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $\check{L}^+\check{D}^+(\check{L}^+)^T$ können also stark voneinander abweichen. Es ist zu betonen, daß die Unterschiede bereits bei den exakten Eigenwerten auftreten und nicht erst bei den berechneten Näherungen. Verantwortlich sind also *Rundungsfehler bei den Zerlegungen* und nicht etwa Probleme bei den Eigenwert-Lösungsverfahren, die auf diese faktorisierten Darstellungen angewendet werden.
Im Gegensatz zur **bSVD** spielen die Abweichungen der Eigenwerte von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $\check{L}^+\check{D}^+(\check{L}^+)^T$ keine Rolle, wenn wir die *getrennt* gestellten **tSEP** $B^TB = V\Sigma^2V^T$ und $BB^T = U\Sigma^2U^T$ betrachten. Bei einem Eigenwertcluster $\sigma_f^2 \leq \ldots \leq \sigma_l^2$ genügt jeweils *eine beliebige RRR* von $B^TB - \hat{\mu}^2I$ und $B^TB - \check{\mu}^2I$, um den Cluster aufzulösen. Wir könnten sogar unterschiedliche Shift-Parameter wählen, um die beim **tSEP** geforderten numerischen Genauigkeitsanforderungen (vgl. Abschnitt 1.3.5) zu erfüllen.
Fatale Folgen haben unterschiedliche Eigenwerte von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $\check{L}^+\check{D}^+(\check{L}^+)^T$ jedoch, wenn wir uns für die Kopplungseigenschaften der Näherungen an die linken und rechten Singulärvektoren interessieren. Dazu präsentieren wir nun ein ausführliches Beispiel.

3. Auswirkungen bei numerischen Experimenten

Wir betrachten $B^TB = W_{21}^+ - \tau I$ mit $\tau < \lambda_1$ (vgl. Beispiel 1.18) und wenden unsere Implementierung des RRR-Algorithmus auf B^TB und BB^T getrennt an. Die Darstellungsbäume sind in beiden Fällen identisch: Für jeden Cluster

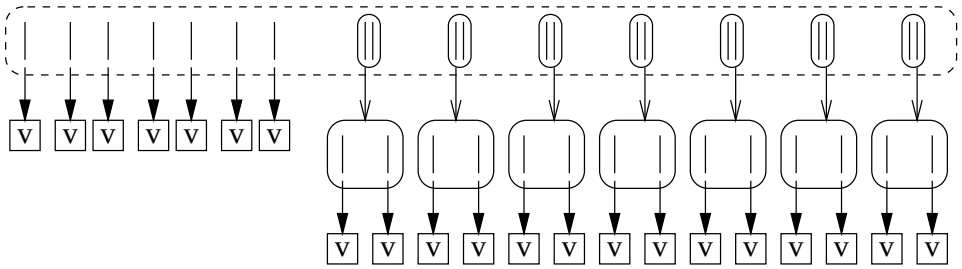


Abbildung 3.1: Darstellungsbaum für $B^TB = W_{21}^+ - \tau I$.

werden jeweils übereinstimmende Shift-Parameter gewählt. Diese sind in Tabelle 3.1 mit einem Stern gekennzeichnet. Siehe auch Abbildung 3.1.
Wir bilden also

$$B^TB - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T \quad \text{und} \quad BB^T - \mu^2 I = \check{L}^+\check{D}^+(\check{L}^+)^T$$

mit jeweils identischen Shift-Strategien. Für die Zerlegungen verwenden wir die QD-artigen Transformationen aus Algorithmus 2.1.
Durch die numerische Überprüfung der *a posteriori* Kriterien kann nachgewiesen werden, daß alle Faktorisierungen jeweils eine partielle RRR bilden. Daher sind die Eigenwerte $\hat{\lambda}_j^+$ und $\check{\lambda}_j^+$ auf hohe relative Genauigkeit bestimmt. Wir beobachten die durch Gleichung (3.2) zu erwartenden Abweichungen. Bei den jeweils kleineren Eigenwerten stimmen nur noch sehr wenige signifikante Stellen überein.

j	$\lambda_j = \sigma_j^2$	$\hat{\lambda}_j^+$	$\check{\lambda}_j^+$
8	*5.998048201384	8.6013525452083D-15	8.6448127042717D-15
9	6.006354023441	8.3058220572403D-03	8.3058220572403D-03
10	*7.001782477743	6.6765339180722D-15	6.9574116568442D-15
11	7.002244425002	4.6194725901782D-04	4.6194725901810D-04
12	8.002217522257	-1.6509327081740D-05	-1.6509327081365D-05
13	*8.002234031585	-1.2863778785396D-14	-1.2488543209404D-14
14	9.005951798617	-4.1091231558744D-07	-4.1091231507153D-07
15	*9.005952209529	-1.4882896339264D-14	-1.4366985384200D-14
16	10.040941115815	-7.0147628503429D-09	-7.0147629270521D-09
17	*10.040941122829	-1.2922467590762D-14	-1.2999176683819D-14
18	11.212678647305	-5.6426308574546D-11	-5.6426258580880D-11
19	*11.212678647362	-1.2794598835728D-14	-1.2744594323512D-14
20	12.748194182904	-9.0249074144178D-14	-9.0667129038281D-14
21	*12.748194182904	-1.8608310027439D-14	-1.8992680959841D-14

Tabelle 3.1: Vergleich der Eigenwertapproximationen. Für $j = 8 : 2 : 20$ zeigen die dritte und vierte Spalte jeweils die Näherungen an den j-ten und $(j+1)$-ten Eigenwert von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ bzw. $\check{L}^+\check{D}^+(\check{L}^+)^T$. Die jeweiligen Shift-Parameter sind mit einem Stern versehen.

Da sich die jeweils interessierenden Eigenwerte von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $\check{L}^+\check{D}^+(\check{L}^+)^T$ unterscheiden, werden die entsprechenden Approximationen an Eigenvektoren weitgehend unabhängig voneinander aufgestellt. Bei der Interpretation als linke und rechte Singulärvektoren stellen wir deswegen schlechte

Kopplungen innerhalb der invarianten Unterräume fest. Siehe Abbildungen 3.2 und 1.3.

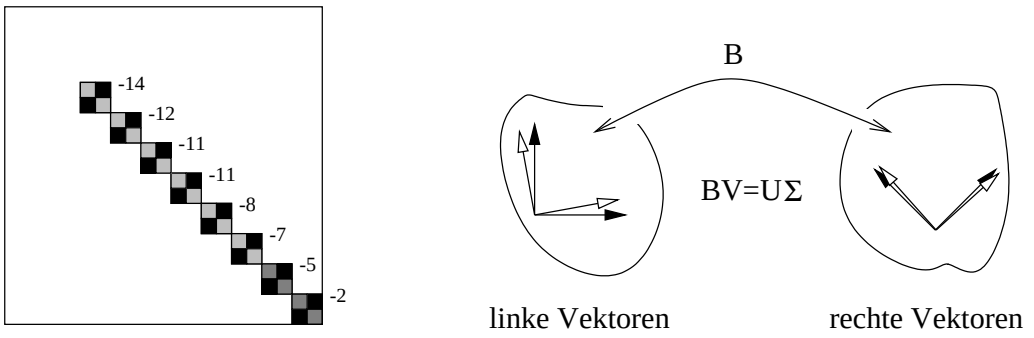


Abbildung 3.2: Links sind die Fehler von $U^T BV - \Sigma$ für die betrachtete Wilkinson-Matrix angedeutet. Die Zahlen geben deren Größenordnung an. So ist beispielsweise $U(:,16)^T BV(:,17) = \mathcal{O}(10^{-7})$. Rechts wird das Problem noch einmal symbolisch dargestellt: Nach der Störungstheorie sind bei engen Clustern die Unterräume sehr gut bestimmt. Deren exakte Basen (gefüllte Pfeile) werden durch numerisch orthogonale Vektoren (offene Pfeile) approximiert. Erfolgt die Konstruktion unabhängig voneinander, kann oft keine gute Kopplung bzgl. B mehr gewährleistet werden.

In diesem Beispiel sind alle Eigenwerte nach einmaligem Anwenden der Shift-Strategie isoliert. In Fällen, bei denen wir einen Darstellungsbaum mit mehreren Ebenen benutzen müssen, würde der RRR-Algorithmus auf beiden Problemen mit unterschiedlichen Shift-Parametern arbeiten und damit die Kopplungseigenschaften weiter negativ beeinflussen: $\text{path}(B^T B) \neq \text{path}(BB^T)$

4. Zusammenfassung der Probleme

Wir halten abschließend einige grundsätzliche Probleme beim getrennten Anwenden des RRR-Algorithmus auf die Normalengleichungen fest:

- Die Eigenwerte von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $\check{L}^+\check{D}^+(\check{L}^+)^T$ können sich stark unterscheiden:
$$\left|\hat{\lambda}_j^+ - \check{\lambda}_j^+\right| = \mathcal{O}(\sigma_j^2 \epsilon)$$
Deshalb kann eine gute Kopplung der Singulärvektoren nicht garantiert werden.

- Dieses Problem verstärkt sich bei Darstellungsbäumen mit mehreren Ebenen, d.h. wenn Translationen von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $\check{L}^+\check{D}^+(\check{L}^+)^T$

zu faktorisieren sind. In diesen Fällen erfolgt die Auswahl von Shift-Parametern möglicherweise nach unterschiedlichen Kriterien.

- Verwenden wir zusätzlich Techniken zur Verbesserung der Orthogonalität wie in Beispiel 1.31 (Bisektoren), kann die Kopplung der Singulärvektoren weiter beeinträchtigt werden.

Andererseits fallen solche Probleme für sehr kleine Singulärwerte nicht ins Gewicht. Für $\sigma_j \leq \sigma_n n\epsilon$ ist bereits

$$\|Bv - \sigma_j u\| \leq \|Bv\| + \|\sigma_j u\| = \mathcal{O}(\sigma_j) = \mathcal{O}(\sigma_n n\epsilon)$$

und damit eine schlechte Kopplung numerisch nicht mehr meßbar.

3.1.2 Golub-Kahan Matrix

Bei der Behandlung des **tSEP** für die Golub-Kahan Matrix $T_{GK} = Q\Lambda Q^T$ reicht es, nur die positiven Eigenwerte und die letzten n Spalten von Q anzunähern. Wir haben bereits festgestellt, daß bei der Anwendung eines numerischen Lösungsverfahrens Approximationen $\bar{Q}$ erzeugt werden, die numerisch orthogonal sind. Extrahieren wir $\bar{U} = \bar{Q}(2:2:2n, n+1:2n)$ und $\bar{V} = \bar{Q}(1:2:2n-1, n+1:2n)$, so sind auch die numerischen Kriterien für $\|B\bar{V} - \bar{U}\bar{\Sigma}\|$ erfüllt. Es gibt aber Fälle, in denen $\bar{U}$ und $\bar{V}$ selbst sehr weit von numerischer Orthogonalität entfernt sind, vgl. Beispiel 1.33.

Eine Fallstudie

Neben der oben skizzierten grundsätzlichen Problematik treten bei der Anwendung des RRR-Algorithmus auf die Golub-Kahan Matrix einige verfahrensspezifische Besonderheiten auf, die wir nun genauer untersuchen. Ein häufig beobachtetes Phänomen ist das Auftreten von großem Elementwachstum bei Faktorisierungen von $T_{GK} - \mu I$.
Als Erweiterung der bereits in Beispiel 2.9 dargestellten Problematik betrachten wir nun eine Bidiagonalmatrix B' mit Singulärwerten

$$\sigma_1 = \sqrt{\epsilon}, \quad \sigma_j = 1 + j\sqrt[4]{\epsilon} \quad \text{für } j = 2:9, \quad \sigma_{10} = \sqrt{2}$$

Aus B' bilden wir dann eine geleimte Matrix $B := B'(10, 1, 200\epsilon)$. (Auf diese Weise können wir beliebig viele Testprobleme erzeugen, bei denen die Situation ähnlich ist, vgl. Beispiel 1.19.) Wir wählen τ nun etwas kleiner als den kleinsten Singulärwert von B und vergleichen das Elementwachstum von $T_{GK} - \tau I = \tilde{L}^+\tilde{D}^+(\tilde{L}^+)^T$ und $B^TB - \tau^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$ in Tabelle 3.2 und Abbildung 3.3. Wir beobachten mit $elg(T_{GK}, \tau) = \mathcal{O}(10^8)$ ein viel größeres

Elementwachstum bei der Faktorisierung der Golub-Kahan Matrix als bei der Verwendung der Normalengleichung: $elg(B^TB, \tau^2) = \mathcal{O}(1)$. In diesem Beispiel sind die *geraden Pivotelemente* von $\tilde{D}^+$ für das große Elementwachstum verantwortlich. Andererseits ist auffällig, daß die Produkte $\tilde{d}^+_{2i-1}\tilde{d}^+_{2i}$ jeweils klein sind. Wir werden diese Beobachtung in Abschnitt 3.2 wieder aufgreifen.

i	$\tilde{d}^+_i$	$\lvert\tilde{d}^+_i\rvert(1+(\tilde{l}^+_i)^2)$	$\hat{d}^+_{i/2}$	$\lvert\hat{d}^+_{i/2}\rvert(1+(\hat{l}^+_{i/2})^2)$
1	-1.4901E-08	2.5586E+07		
2	2.5586E+07	2.5586E+07	3.8123E-01	1.0005E+00
3	-3.9102E-08	1.0129E+00		
4	1.0129E+00	2.0007E+00	3.9608E-08	1.0005E+00
⋮	⋮	⋮	⋮	⋮
39	-1.5099E-08	1.3065E+08		
40	1.3065E+08	1.3065E+08	1.9727E+00	1.9727E+00

Tabelle 3.2: Ausgewählte diagonale Pivotelemente und Terme, die das Elementwachstum beeinflussen. Vergleich der Matrizen $\tilde{L}^+\tilde{D}^+(\tilde{L}^+)^T$ (Golub-Kahan Matrix) und $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ (Normalengleichung).

Zusammenfassung der Probleme

Wir tragen einige Beobachtungen zusammen, die sich beim Anwenden des RRR-Algorithmus auf die Golub-Kahan Matrix nachteilig auswirken können. Das Kernproblem ist, daß Faktorisierungen von $T_{GK} - \mu I$ deutlich anfälliger für unerwünschte Effekte sind als die äquivalenten Faktorisierungen von $B^TB - \mu^2 I$ oder $BB^T - \mu^2 I$.

- Ein Cluster aus kleinen Singulärwerten von B führt zu eng beieinander liegenden Eigenwerten der Golub-Kahan Matrix, die symmetrisch um die Null herum angeordnet sind. Während der relative Abstand $\rho(-\sigma_1, \sigma_1) > 1$ unbedenklich ist, kann der absolute Abstand zwischen $-\sigma_1$ und σ_1 zu Verschlechterungen der Schranke des *a priori* Kriteriums führen, vgl. Satz 2.7 sowie Beispiele 2.8 und 2.9.

- Wählen wir den Shift τ ungünstig in der Nähe eines Ritz-Wertes der Golub-Kahan Matrix, so müssen wir mit hohem Elementwachstum oder sogar einem Breakdown der Faktorisierung rechnen. Da jeder Minor von T_{GK} singulär ist, tritt diese Situation im Vergleich zu den

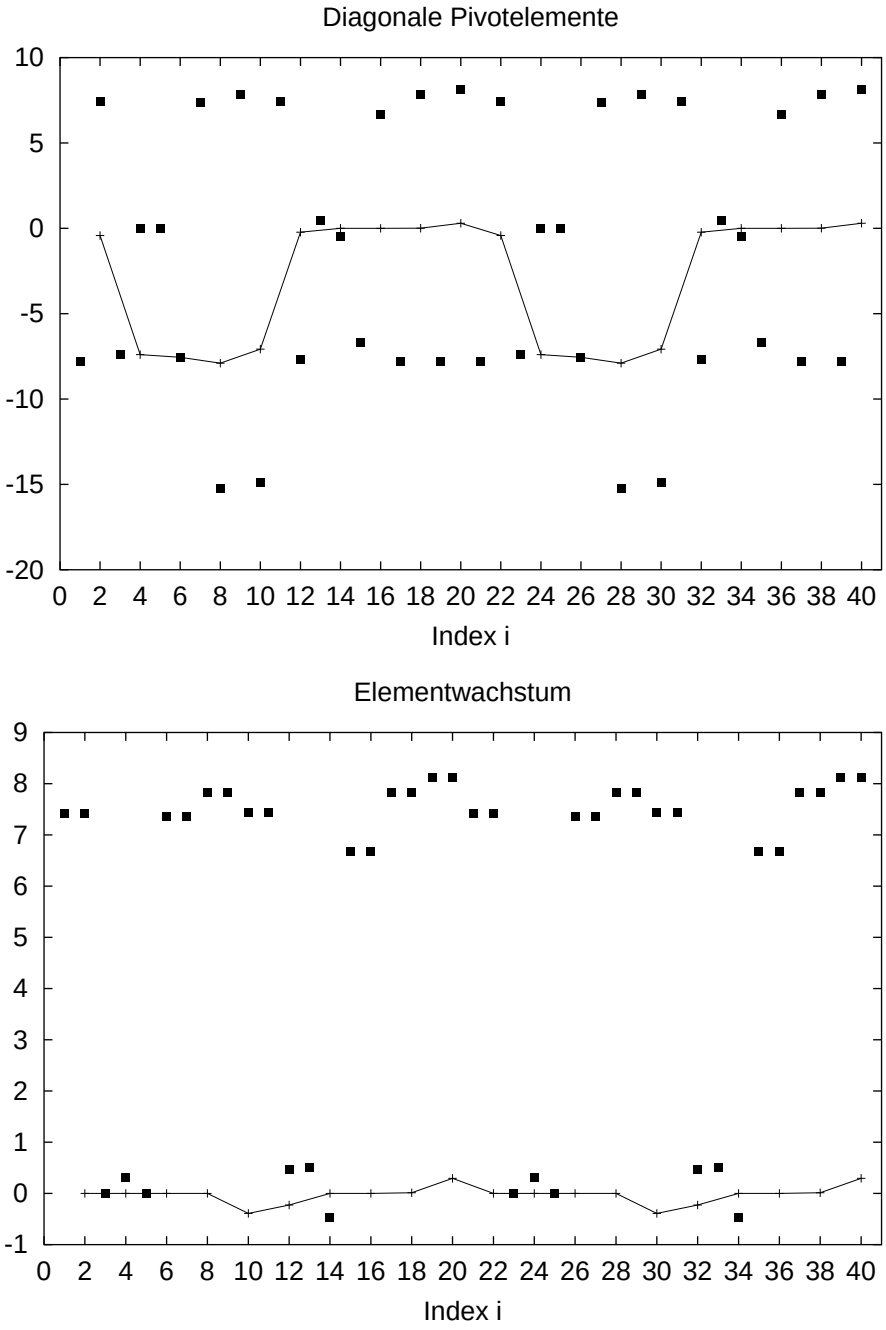


Abbildung 3.3: Vergleich der Größenordnung der diagonalen Pivotelemente (oben) und der Terme, die das Elementwachstum beeinflussen (unten). Dargestellt sind die Daten der Faktorisierungen $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ (Normalengleichung) und $\tilde{L}^+\tilde{D}^+(\tilde{L}^+)^T$ (Golub-Kahan Matrix). In den Diagrammen sind die Größen $\log_{10}(|\tilde{d}_i^+|)$ und $\log_{10}(|\tilde{d}_i^+|(1+(\tilde{l}_i^+)^2))$ als ausgefüllte Quadrate aufgetragen. Die Werte von $\log_{10}(|\hat{d}_{i/2}^+|)$ und $\log_{10}(|\hat{d}_{i/2}^+|(1+(\hat{l}_{i/2}^+)^2))$ sind als Kreuze markiert und zur Verdeutlichung durch Linien verbunden.

Normalengleichungen häufiger ein. Deswegen sind oft mehrere Shift-Kandidaten zu testen, und wir müssen verstärkt auf das *a posteriori* Kriterium (Kontrolle der relativen Konditionszahlen, Abschnitt 2.4) zurückgreifen, um gesicherte Ergebnisse zu erhalten. Dadurch kann sich der Aufwand gegenüber der Arbeit auf den Normalengleichungen merkbar erhöhen. Siehe auch Abbildung 3.4.

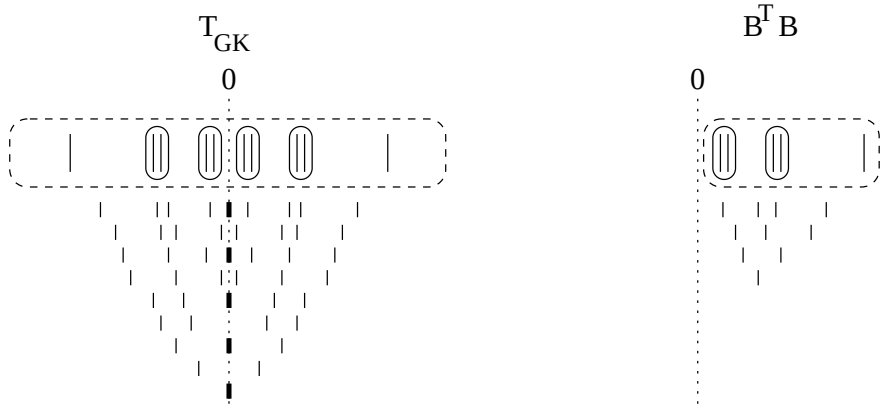


Abbildung 3.4: Eigen- und Ritzwerte von T_{GK} bzw. B^TB und BB^T.

- Kritisch sind große Schwankungen bei den diagonalen Pivotelementen auch für die Bestimmung einer Approximation an einen Eigenvektor q, aus der anschließend Näherungen an den linken und rechten Singulärvektor von B extrahiert werden sollen: Es kann zu einer „Verschiebung der Information“ von den kleinen auf die großen Komponenten von q kommen. Wenn betragsgroße und -kleine Komponenten von q gerade alternieren, können die gewonnenen Näherungen an u und v weit von numerischer Orthogonalität entfernt sein, obwohl die Näherung an q selbst hinreichend genau ist. Dieses Phänomen beobachten wir bei numerischen Experimenten, wenn wir Testläufe für geleimte Matrizen mit hoher Konditionszahl durchführen.

- Maßnahmen zur Verbesserung der numerischen Orthogonalität der mit dem RRR-Algorithmus ermittelten Näherungen an Q können ungewollten und schwer kontrollierbaren Einfluß auf die Qualität der extrahierten Näherungen an U und V haben. Siehe auch Beispiel 1.33 auf Seite 20.

- Als letztes Argument gegen Black-Box-Anwendung des RRR-Algorithmus auf die Golub-Kahan Matrix ist anzuführen, daß wir

innere Schleifen der Länge $2n$ anstatt n abarbeiten müssen. Im Vergleich zur zweimaligen Black-Box-Anwendung des RRR-Algorithmus auf B^TB bzw. BB^T verdoppelt sich damit der Rechenaufwand für einzelne Teilaufgaben, z.B. für die Bestimmung der interessierenden Eigenwerte auf Zwischenstufen im Darstellungsbaum. Dieser zusätzliche Aufwand schlägt sich in meßbar höheren Ausführungszeiten nieder.

Ein hybrider Algorithmus?

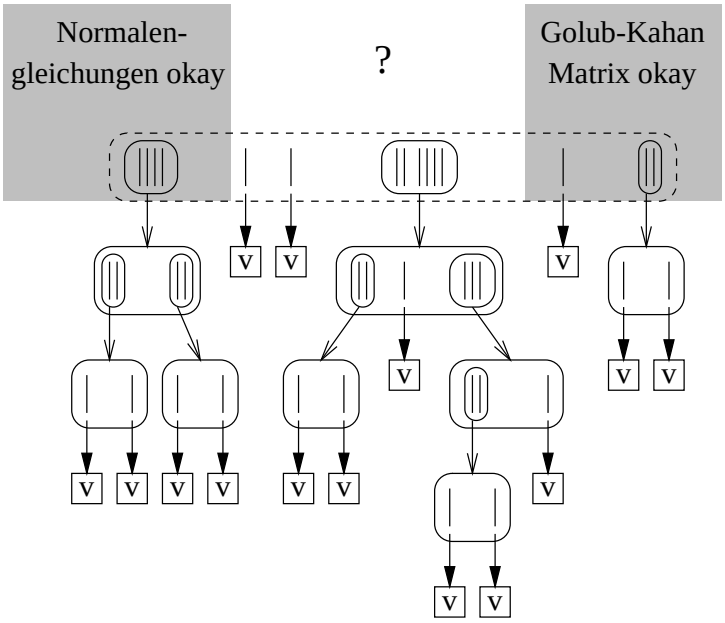


Abbildung 3.5: Ein hybrider Algorithmus.

Die oben genannten Schwierigkeiten bei Verwendung der Golub-Kahan Matrix treten verstärkt bei Clustern kleiner Singulärwerte auf. Für viele Testmatrizen erhalten wir jedoch durch Black-Box-Anwendung des RRR-Algorithmus auf T_{GK} Näherungen, die auch den Genauigkeitsanforderungen an die ursprüngliche **bSVD** sehr gut genügen. So ist beispielsweise das Wilkinson-Problem auf der Basis der Golub-Kahan Matrix hinreichend genau lösbar. Andererseits ist es in manchen Fällen günstiger, die Normalengleichungen zu verwenden. Zusammengefaßt bedeutet dies:

- Die Anwendung des RRR-Algorithmus auf die Golub-Kahan Matrix führt zu gut gekoppelten Näherungen an die Singulärvektoren. Problematisch sind Cluster kleiner Singulärwerte.

- Die Anwendung des RRR-Algorithmus auf die Normalengleichungen führt zu Näherungen an Singulärvektoren, die numerisch orthogonal sind. Schlechte Kopplungen treten bei Clustern großer Singulärwerte auf.

Diese Beobachtungen und die Adaptivität des RRR-Algorithmus legen die Verwendung eines hybriden Verfahrens nahe, vgl. Abbildung 3.5. Dieser Ansatz bleibt aber ingesamt unbefriedigend. Es fehlen allgemeingültige Aussagen, die für beliebige Spektren angeben, wann die Verwendung der Normalengleichungen angemessen ist und wo auf die Golub-Kahan Matrix zurückgegriffen werden sollte. Insbesondere ist nicht geklärt, ob immer eine der beiden Varianten zum Erfolg führt.

3.1.3 Der Lösungsansatz: Kopplungen

Nach den Beobachtungen der vorigen Abschnitte führt das getrennte Anwenden des RRR-Algorithmus auf B^TB, BB^T oder T_{GK} i.a. nicht zu Ergebnissen, die den Genauigkeitsanforderungen der ursprünglich gestellten **bSVD** genügen. Auch bei hybriden Strategien fehlen hinreichende Kriterien für ein generell zuverlässiges Verfahren.
Um Singulärvektorpaare zu bestimmen, die sowohl numerisch orthogonal als auch gut gekoppelt sind, schlagen wir eine Methode vor, die das **tSEP** $B^TB = V\Sigma^2V^T$ mit dem RRR-Algorithmus löst. Dabei führen wir aber die entsprechenden Faktorisierungen von BB^T implizit mit, vgl. Abbildung 3.6. Als Ergebnis dieser Strategie weisen wir weitaus günstigere Eigenschaften der Eigenwerte der so verknüpften Faktorisierungen nach. In numerischen Experimenten zeigt sich ferner, daß die auf diese Weise gebildeten Näherungen an die linken Singulärvektoren vorzüglich gekoppelt sind. Im Folgenden stellen wir ein System von Kopplungsmechanismen zur Verfügung, mit denen wir die Faktorisierungen von B^TB, BB^T und T_{GK} in Beziehung zueinander setzen können.
Wir nehmen an, daß wir den RRR-Algorithmus auf B^TB und BB^T mit identischen Shift-Parametern anwenden. Zu einem interessierenden Singulärvektorpaar bezeichnen wir die entsprechenden Pfade mit

$$\mathrm{path}(B^TB) = \mathrm{path}(BB^T) = (\nu^{(1)}, \ \ldots \ , \nu^{(\mathrm{rec})})$$

Wir bestimmen also beispielsweise BABE-Faktorisierungen

$$\begin{aligned} B^TB - \nu^{(1)}I &= \hat{L}^{(1)}\hat{D}^{(1)}(\hat{L}^{(1)})^T = \hat{U}^{(1)}\hat{R}^{(1)}(\hat{U}^{(1)})^T \\ &= \hat{N}_k^{(1)}\hat{G}_k^{(1)}(\hat{N}_k^{(1)})^T,\ k = 1\!:\!n \end{aligned}$$

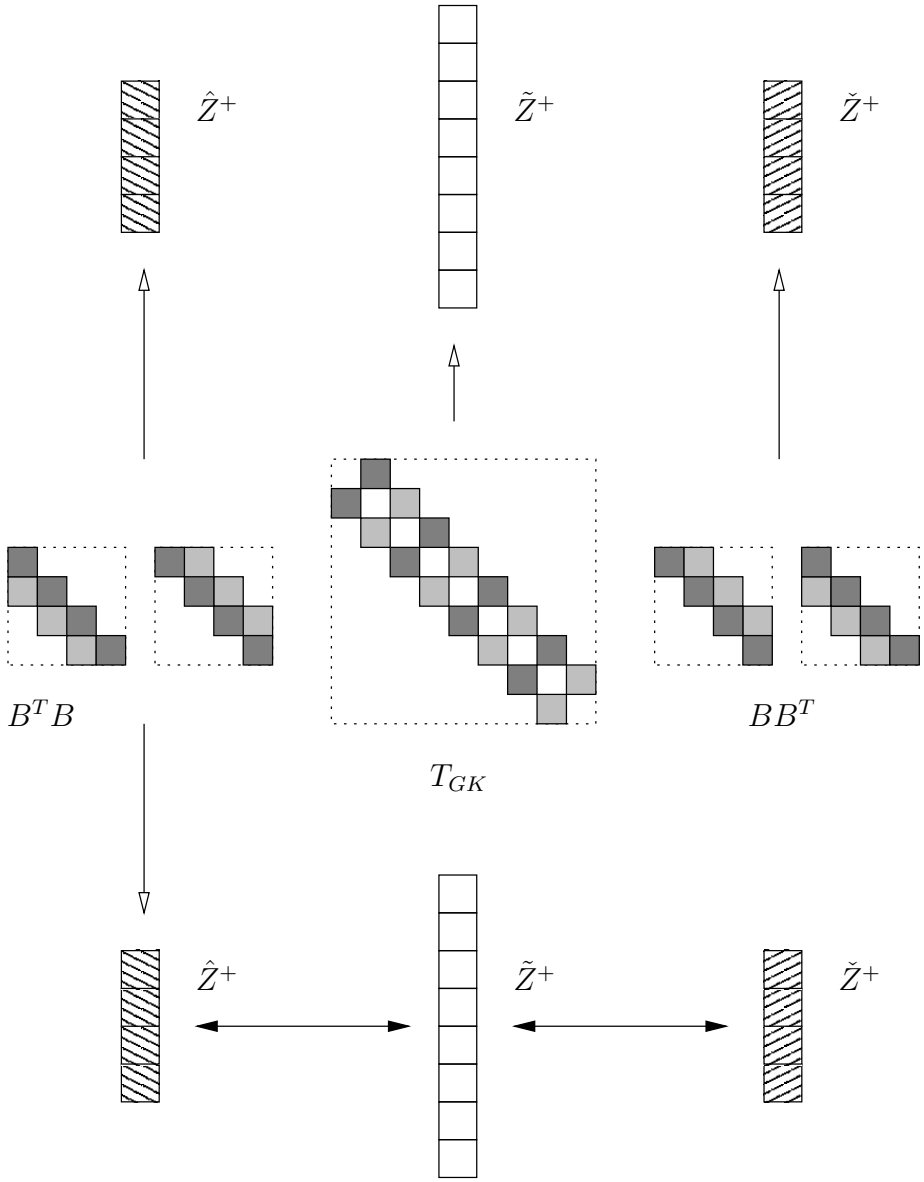


Abbildung 3.6: Grundidee der Kopplungen: Anstatt Translationen von B^TB, BB^T und T_{GK} explizit und unabhängig voneinander zu faktorisieren (offene Pfeile nach oben), wenden wir den RRR-Algorithmus auf B^TB an und benutzen Kopplungsmechanismen, mit denen die Beziehungen zwischen den Daten $\hat{Z}^+$, $\check{Z}^+$ und $\tilde{Z}^+$ beschrieben werden können (gefüllte Pfeile unten).

und

$$\begin{aligned}\hat{L}^{(r-1)}\hat{D}^{(r-1)}(\hat{L}^{(r-1)})^T - \nu^{(r)}I &= \hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T = \hat{U}^{(r)}\hat{R}^{(r)}(\hat{U}^{(r)})^T \\ &= \hat{N}_k^{(r)}\hat{G}_k^{(r)}(\hat{N}_k^{(r)})^T,\ k = 1\!:\!n\end{aligned}$$

mit Hilfe QD-artiger Transformationen, die wir in den Abschnitten 3.2 und 3.3 näher spezifizieren. Die entsprechenden Parametrisierungen speichern wir für $r = 1 : \text{rec}$ in Matrix-Tupeln $\hat{Z}^{(r)} = [\hat{D}^{(r)}, \hat{L}^{(r)}, \hat{R}^{(r)}, \hat{U}^{(r)}, \hat{\Gamma}^{(r)}]$. Faktorisierungen basierend auf BB^T bzw. T_{GK} fassen wir zu entsprechenden Matrix-Tupeln $\check{Z}^{(r)}$ bzw. $\tilde{Z}^{(r)}$ zusammen.

Definition 3.2 (Ein äquivalenter Pfad für T_{GK})
Wir definieren einen zu $\text{path}(B^TB) = \text{path}(BB^T) = (\nu^{(1)}, \ldots, \nu^{(\text{rec})})$ äquivalenten Pfad für T_{GK} durch

for $r = 1 : \text{rec}$ **do**
$\nu_{\text{sum}} = \sum_{j=1}^{r-1} \nu^{(j)}$
$\mu_{\text{sum}} = \sum_{j=1}^{r-1} \mu^{(j)}$
Wähle $\mu^{(r)}$ als betragskleinere Lösung von $(\mu_{\text{sum}} + \mu^{(r)})^2 = (\nu_{\text{sum}} + \nu^{(r)})$.
end for
$\text{path}(T_{GK}) := (\mu^{(1)}, \ldots, \mu^{(\text{rec})})$

Es gilt $\sum_{j=1}^{r} \nu^{(j)} = (\sum_{j=1}^{r} \mu^{(j)})^2$ für alle $r \leq \text{rec}$.

In den nächsten beiden Abschnitten untersuchen wir die Kopplungen zwischen den Daten von $\hat{Z}^{(r)}$, $\check{Z}^{(r)}$ und $\tilde{Z}^{(r)}$ und beschäftigen uns mit der Frage, wie die Umrechnungsmechanismen genutzt werden können, um gut gekoppelte Singulärvektorpaare zu erhalten, vgl. Abbildung 3.7. Es stellen sich signifikante Unterschiede zwischen Kopplungen auf der ersten Ebene ($r = 1$) und Kopplungen auf höheren Ebenen ($r > 1$) heraus.

	B^TB		T_{GK}		BB^T
Ebene 1	$\hat{\nu}^{(1)},\ \hat{Z}^{(1)}$	$\overset{?}{\longleftrightarrow}$	$\tilde{\mu}^{(1)},\ \tilde{Z}^{(1)}$	$\overset{?}{\longleftrightarrow}$	$\check{\nu}^{(1)},\ \check{Z}^{(1)}$
Ebene 2	$\hat{\nu}^{(2)},\ \hat{Z}^{(2)}$	$\overset{?}{\longleftrightarrow}$	$\tilde{\mu}^{(2)},\ \tilde{Z}^{(2)}$	$\overset{?}{\longleftrightarrow}$	$\check{\nu}^{(2)},\ \check{Z}^{(2)}$
...					
Ebene rec	$\hat{\nu}^{(\text{rec})},\ \hat{Z}^{(\text{rec})}$	$\overset{?}{\longleftrightarrow}$	$\tilde{\mu}^{(\text{rec})},\ \tilde{Z}^{(\text{rec})}$	$\overset{?}{\longleftrightarrow}$	$\check{\nu}^{(\text{rec})},\ \check{Z}^{(\text{rec})}$
	v		$q = P_{ps}\binom{u}{v}$		u

Abbildung 3.7: Pfade und Parametrisierungen zur Bestimmung von Singulärvektoren eines einzelnen Singulärwertes σ.

3.2 Kopplungen auf der ersten Ebene

Wir betrachten Zusammenhänge zwischen Faktorisierungen auf der ersten Ebene des Darstellungsbaumes. In diesem Fall ist $\nu^{(1)} = (\mu^{(1)})^2$. Wir kürzen $\mu^{(1)} = \mu$ ab und schreiben bei den Faktorisierungen anstelle der Indizes lediglich ein Pluszeichen, also beispielsweise

$$B^T B - \mu^2 I = \hat{L}^+ \hat{D}^+ (\hat{L}^+)^T$$

als übersichtlichere Abkürzung für

$$B^T B - \nu^{(1)} I = \hat{L}^{(1)} \hat{D}^{(1)} (\hat{L}^{(1)})^T$$

Bei Kopplungen auf der ersten Ebene untersuchen wir also Zerlegungen von

$$B^T B - \mu^2 I, \quad BB^T - \mu^2 I \quad \text{und} \quad T_{GK} - \mu I$$

Wir finden hier die günstige Situation vor, daß in allen drei Fällen die zu faktorisierende Matrix durch die *Eingangsdaten* der Matrix B gegeben ist. Wir können insbesondere die Eigenschaft $\texttt{diag}(T_{GK}) = 0$ (Nullvektor) intensiv ausnutzen.

3.2.1 Faktorisierungen

Wir verallgemeinern nun die Algorithmen 2.1 und 2.2 von LDL^T-Faktorisierungen auf allgemeine BABE-Faktorisierungen. Wir führen dazu weitere Hilfsvariablen

$$\hat{p}_i = \hat{r}_i^+ - b_{i-1}^2 = \frac{a_i \hat{u}_i^+}{b_i} \hat{p}_{i+1} - \mu^2 \quad \text{und} \quad \check{p}_{i+1} = \check{d}_{i+1}^+ - b_i^2 = \frac{a_{i+1} \check{l}_i^+}{b_i} \check{p}_i - \mu^2$$

ein und verwenden das folgende in [29] bewiesene Lemma, das nützliche Formeln für die Berechnung der Twist-Elemente bereitstellt. Siehe auch Gleichung (2.5) auf Seite 38.

Lemma 3.3 (Formeln für Γ^+)
Für eine symmetrisch tridiagonale Matrix T und einen Shift-Parameter τ mögen die Faktorisierungen

$$T - \tau I = L^+ D^+ (L^+)^T = U^+ R^+ (U^+)^T = N_k^+ G_k^+ (N_k^+)^T \,, \quad k = 1:n$$

existieren. Dann gilt

$$\det(T - \tau I) = d_1^+ \cdots d_{k-1}^+ \gamma_k^+ r_{k+1}^+ \cdots r_n^+ \tag{3.3}$$

$$\gamma_k^+ r_{k+1}^+ = d_k^+ \gamma_{k+1}^+ \tag{3.4}$$

$$\gamma_k^+ = s_k + p_k + \tau \tag{3.5}$$

Die Transformationen aus Algorithmus 3.1 erlauben dann eine gemischte Stabilitätsanalyse mit kleinen multiplikativen Störungen. Die Aussagen von Satz 2.2 sind also übertragbar auf BABE-Faktorisierungen. Siehe auch [29, 37] und Anhang B.

Algorithmus 3.1 Faktorisiere $B^TB - \mu^2 I$ (links) und $BB^T - \mu^2 I$ (rechts).

Gegeben: B, μ	**Gegeben:** B, μ
Gesucht: $\hat{Z}^+ = [\hat{D}^+, \hat{L}^+, \hat{S}, \hat{R}^+, \hat{U}^+, \hat{P}, \hat{\Gamma}^+]$	**Gesucht:** $\check{Z}^+ = [\check{D}^+, \check{L}^+, \check{P}, \check{R}^+, \check{U}^+, \check{S}, \check{\Gamma}^+]$
1: $\hat{s}_1 = -\mu^2$	1: $\check{s}_n = -\mu^2$
2: **for** $i = 1 : n-1$ **do**	2: **for** $i = n-1 : -1 : 1$ **do**
3: $\hat{d}_i^+ = \hat{s}_i + a_i^2$	3: $\check{r}_{i+1}^+ = \check{s}_{i+1} + a_{i+1}^2$
4: $\hat{l}_i^+ = \frac{a_i b_i}{\hat{d}_i^+}$	4: $\check{u}_i^+ = \frac{a_{i+1} b_i}{\check{r}_{i+1}^+}$
5: $\hat{s}_{i+1} = \frac{b_i^2}{\hat{d}_i^+}\hat{s}_i - \mu^2$	5: $\check{s}_i = \frac{b_i \check{u}_i^+}{a_{i+1}}\check{s}_{i+1} - \mu^2$
6: **end for**	6: **end for**
7: $\hat{d}_n^+ = \hat{s}_n + a_n^2$	7: $\check{r}_1^+ = \check{s}_1 + a_1^2$
8: $\hat{p}_n = a_n^2 - \mu^2$	8: $\check{p}_1 = a_1^2 - \mu^2$
9: **for** $i = n-1 : -1 : 1$ **do**	9: **for** $i = 1 : n-1$ **do**
10: $\hat{r}_{i+1}^+ = \hat{p}_{i+1} + b_i^2$	10: $\check{d}_i^+ = \check{p}_i + b_i^2$
11: $\hat{u}_i^+ = \frac{a_i b_i}{\hat{r}_{i+1}^+}$	11: $\check{l}_i^+ = \frac{a_{i+1} b_i}{\check{d}_i^+}$
12: $\hat{p}_i = \frac{a_i \hat{u}_i^+}{b_i}\hat{p}_{i+1} - \mu^2$	12: $\check{p}_{i+1} = \frac{a_{i+1}\check{l}_i^+}{b_i}\check{p}_i - \mu^2$
13: **end for**	13: **end for**
14: $\hat{r}_1^+ = \hat{p}_1$	14: $\check{d}_n^+ = \check{p}_n$
15: **for** $i = 1 : n$ **do**	15: **for** $i = 1 : n$ **do**
16: $\hat{\gamma}_i^+ = \hat{s}_i + \hat{p}_i + \mu^2$	16: $\check{\gamma}_i^+ = \check{s}_i + \check{p}_i + \mu^2$
17: **end for**	17: **end for**

Bemerkung 3.4 (Linke und rechte Seite von Algorithmus 3.1)
Die Matrix $B^{(\mathrm{rev})} = \mathtt{diag}([a_n, \ldots, a_1]) + \mathtt{diag}([b_{n-1}, \ldots, b_1]), 1)$ enthalte die Einträge von B in umgekehrter Reihenfolge. Die Faktorisierungen

$$B^TB - \mu^2 I = \hat{L}\hat{D}\hat{L}^T \quad \text{und} \quad B^{(\mathrm{rev})}(B^{(\mathrm{rev})})^T - \mu^2 I = \check{U}^{(\mathrm{rev})}\check{R}^{(\mathrm{rev})}(\check{U}^{(\mathrm{rev})})^T$$

seien mit den Zeilen 1 bis 7 der linken und rechten Seite von Algorithmus 3.1 berechnet. Dann gilt für $i = 1 : n$

$$\hat{d}_{n-i+1} = \check{r}_i^{(\mathrm{rev})}, \quad \hat{l}_{n-i} = \check{u}_i^{(\mathrm{rev})}, \quad \hat{s}_{n-i+1} = \check{s}_i^{(\mathrm{rev})}$$

Für die Faktorisierungen

$$B^TB - \mu^2 I = \hat{U}\hat{R}\hat{U}^T \quad \text{und} \quad B^{(\mathrm{rev})}(B^{(\mathrm{rev})})^T - \mu^2 I = \check{L}^{(\mathrm{rev})}\check{D}^{(\mathrm{rev})}(\check{L}^{(\mathrm{rev})})^T$$

gegeben durch die Zeilen 8 bis 14 gilt analog

$$\hat{r}_{n-i+1} = \check{d}_i^{(\mathrm{rev})}, \quad \hat{u}_{n-i} = \check{l}_i^{(\mathrm{rev})}, \quad \hat{p}_{n-i+1} = \check{p}_i^{(\mathrm{rev})}$$

und entsprechend den Zeilen 16

$$\hat{\gamma}_{n-i+1} = \check{\gamma}_i^{(\mathrm{rev})}$$

Wir können also die linke Seite von Algorithmus 3.1 auch definieren durch Anwenden der rechten Seite auf $B^{(\mathrm{rev})}$ und umgekehrt.

Algorithmus 3.2 Faktorisiere $T_{GK} - \mu I = \tilde{N}_k^+ \tilde{G}_k^+ (\tilde{N}_k^+)^T, k = 1 : 2n.$

Gegeben: $c := [a_1, b_1, a_2, \ldots, b_{n-1}, a_n]$, μ
Gesucht: $\tilde{Z}^+ = [\tilde{D}^+, \tilde{L}^+, \tilde{R}^+, \tilde{U}^+, \tilde{\Gamma}^+]$

1: $\tilde{d}_1^+ = -\mu$
2: **for** $i = 1 : 2n-1$ **do**
3: $\quad \tilde{l}_i^+ = \frac{c_i}{\tilde{d}_i^+}$
4: $\quad \tilde{d}_{i+1}^+ = -\frac{c_i^2}{\tilde{d}_i^+} - \mu \quad \{= -c_i \tilde{l}_i^+ - \mu\}$
5: **end for**
6: $\tilde{r}_{2n}^+ = -\mu$
7: **for** $i = 2n-1 : -1 : 1$ **do**
8: $\quad \tilde{u}_i^+ = \frac{c_i}{\tilde{r}_{i+1}^+}$
9: $\quad \tilde{r}_i^+ = -\frac{c_i^2}{\tilde{r}_{i+1}^+} - \mu \quad \{= -c_i \tilde{u}_i^+ - \mu\}$
10: **end for**
11: **for** $i = 1 : 2n$ **do**
12: $\quad \tilde{\gamma}_i^+ = \tilde{d}_i^+ + \tilde{r}_i^+ + \mu$
13: **end for**

Bemerkung 3.5 (Faktorisierungen von $T_{GK} - \mu I$ und $T_{GK} + \mu I$)
Beim Vergleich der Faktorisierungen

$$T_{GK} - \mu I = \tilde{N}_k^+ \tilde{G}_k^+ (\tilde{N}_k^+)^T \quad \text{und} \quad T_{GK} + \mu I = \tilde{\tilde{N}}_k^+ \tilde{\tilde{G}}_k^+ (\tilde{\tilde{N}}_k^+)^T$$

ergibt sich für $i = 1 : 2n$

$$\tilde{d}_i^+ = -\tilde{\tilde{d}}_i^+, \quad \tilde{l}_i^+ = -\tilde{\tilde{l}}_i^+, \quad \tilde{r}_i^+ = -\tilde{\tilde{r}}_i^+, \quad \tilde{u}_i^+ = -\tilde{\tilde{u}}_i^+, \quad \tilde{\gamma}_i^+ = -\tilde{\tilde{\gamma}}_i^+$$

3.2.2 Kopplungen

In [37], Theorem 3.3, sind Zusammenhänge zwischen LDL^T-Faktorisierungen von $B^TB-\mu^2 I$, $BB^T-\mu^2 I$ und $T_{GK}-\mu I$ in exakter Arithmetik dargestellt. Wir fassen die Ergebnisse in der ersten Zeile des nachfolgenden Lemmas zusammen und ergänzen die entsprechenden Zusammenhänge für allgemeinere BABE-Faktorisierungen.

Lemma 3.6 (Kopplung von $\hat{Z}^+ \leftrightarrow \tilde{Z}^+ \leftrightarrow \check{Z}^+$)
Wenn die entsprechenden Faktorisierungen existieren, dann gilt für $i = 1:n$

$$\begin{array}{rclcrcl} \mu\tilde{d}^+_{2i-1} &=& \hat{s}_i & & \mu\tilde{d}^+_{2i} &=& \check{p}_i \\ \mu\tilde{r}^+_{2i-1} &=& \hat{p}_i & & \mu\tilde{r}^+_{2i} &=& \check{s}_i \\ \mu\tilde{\gamma}^+_{2i-1} &=& \hat{\gamma}^+_i & & \mu\tilde{\gamma}^+_{2i} &=& \check{\gamma}^+_i \end{array} \qquad \text{([37], Theorem 3.3)}$$

Beweis: Stellvertretend für die ersten vier Formeln zeigen wir $\mu\tilde{r}^+_{2i-1} = \hat{p}_i$. Mit Hilfe von Algorithmus 3.2 drücken wir $\tilde{r}^+_{2i-1}$ durch $\tilde{r}^+_{2i+1}$ aus:

$$\tilde{r}^+_{2i-1} \overset{Z.9}{=} -\frac{c^2_{2i-1}}{\tilde{r}^+_{2i}} - \mu \overset{Z.9}{=} -\frac{a_i^2}{-\frac{c^2_{2i}}{\tilde{r}^+_{2i+1}} - \mu} - \mu = a_i^2 \frac{\tilde{r}^+_{2i+1}}{b_i^2 + \mu\tilde{r}^+_{2i+1}} - \mu$$

Algorithmus 3.1 liefert eine Beschreibung von $\hat{p}_i$ durch $\hat{p}_{i+1}$

$$\hat{p}_i \overset{Z.11,12}{=} \frac{a_i^2}{\hat{r}^+_{i+1}}\hat{p}_{i+1} - \mu^2 \overset{Z.10}{=} a_i^2 \frac{\hat{p}_{i+1}}{b_i^2 + \hat{p}_{i+1}} - \mu^2$$

und damit den Induktionsschluß.

Die Kopplungen von $\hat{\Gamma}^+$, $\check{\Gamma}^+$ und $\tilde{\Gamma}^+$ ergeben sich aus dem Vergleich der Zeile 12 von Algorithmus 3.2 mit den Zeilen 16 von Algorithmus 3.1. □

Mit Lemma 3.6 können wir also die Größen $\hat{Z}^+ \leftrightarrow \tilde{Z}^+ \leftrightarrow \check{Z}^+$ leicht ineinander umrechnen. Die nächste Folgerung erlaubt es, $\tilde{Z}^+$ zu eliminieren und damit $\hat{Z}^+ \leftrightarrow \check{Z}^+$ direkt miteinander zu verknüpfen.

Folgerung 3.7 (Direkte Kopplung $\hat{Z}^+ \leftrightarrow \check{Z}^+$)
Setzen wir $\hat{s}_{n+1} = \check{s}_0 = \hat{s}_1 = -\mu^2$ und $\hat{r}^+_{n+1} = \hat{p}_{n+1} = \check{d}^+_0 = \check{p}_0 = 1$, so ergibt sich für $i = 1:n$

$$\check{d}^+_i = \hat{s}_{i+1}\frac{\hat{d}^+_i}{\hat{s}_i}, \quad \check{p}_i = -\mu^2\frac{\hat{d}^+_i}{\hat{s}_i}, \quad \check{r}^+_i = \hat{p}_i\frac{\hat{r}^+_{i+1}}{\hat{p}_{i+1}}, \quad \check{s}_i = -\mu^2\frac{\hat{r}^+_{i+1}}{\hat{p}_{i+1}}$$

$$\hat{d}^+_i = \check{p}_i\frac{\check{d}^+_{i-1}}{\check{p}_{i-1}}, \quad \hat{s}_i = -\mu^2\frac{\check{d}^+_{i-1}}{\check{p}_{i-1}}, \quad \hat{r}^+_i = \check{s}_{i-1}\frac{\check{r}^+_i}{\check{s}_i}, \quad \hat{p}_i = -\mu^2\frac{\check{r}^+_i}{\check{s}_i}$$

$$\hat{\gamma}^+_i\check{s}_i = \check{\gamma}^+_i\hat{s}_i$$

Beweis: Stellvertretend für die ersten beiden Zeilen zeigen wir $\check{s}_i = -\mu^2 \frac{\hat{r}^+_{i+1}}{\hat{p}_{i+1}}$. Mit Zeile 9 von Algorithmus 3.2, Zeile 10 von Algorithmus 3.1 und dem vorigen Lemma ergibt sich

$$\check{s}_i = \mu\tilde{r}^+_{2i} = \mu\left(-\frac{c^2_{2i}}{\tilde{r}^+_{2i+1}} - \mu\right) = -\mu^2\left(\frac{b_i^2 + \mu\tilde{r}^+_{2i+1}}{\mu\tilde{r}^+_{2i+1}}\right) = -\mu^2\frac{\hat{r}^+_{i+1}}{\hat{p}_{i+1}}$$

Auf ähnliche Weise zeigen wir $\hat{p}_i = -\mu^2\frac{\check{r}^+_i}{\check{s}_i}$ und können $\check{r}^+_i = -\mu^{-2}\hat{p}_i\check{s}_i = \hat{p}_i\frac{\hat{r}^+_{i+1}}{\hat{p}_{i+1}}$ folgern.

Zum Beweis von $\hat{\gamma}^+_i\check{s}_i = \check{\gamma}^+_i\hat{s}_i$ verwenden wir die rekursive Darstellung $\tilde{\gamma}^+_{2i}\tilde{d}^+_{2i-1} = \tilde{r}^+_{2i}\tilde{\gamma}^+_{2i-1}$ aus Lemma 3.3 und beachten das vorige Lemma. □

Folgerung 3.8 (Kopplung der Pivotelemente)
Sei $\tilde{d}^+_{2n+1} = \tilde{r}^+_0 = -\mu$. Mit Lemma 3.6 und Folgerung 3.7 erhalten wir

$$\begin{array}{rclrcll}
\hat{d}^+_i &=& -\tilde{d}^+_{2i-1}\tilde{d}^+_{2i} & \check{d}^+_i &=& -\tilde{d}^+_{2i}\tilde{d}^+_{2i+1} & i = 1:n \\
\hat{l}^+_i &=& -\tilde{l}^+_{2i-1}\tilde{l}^+_{2i} & \check{l}^+_i &=& -\tilde{l}^+_{2i}\tilde{l}^+_{2i+1} & i = 1:n-1 \\
\hat{r}^+_i &=& -\tilde{r}^+_{2i-2}\tilde{r}^+_{2i-1} & \check{r}^+_i &=& -\tilde{r}^+_{2i-1}\tilde{r}^+_{2i} & i = 1:n \\
\hat{u}^+_i &=& -\tilde{u}^+_{2i-1}\tilde{u}^+_{2i} & \check{u}^+_i &=& -\tilde{u}^+_{2i}\tilde{u}^+_{2i+1} & i = 1:n-1 \\
\hat{\gamma}^+_i &=& \mu\tilde{\gamma}^+_{2i-1} & \check{\gamma}^+_i &=& \mu\tilde{\gamma}^+_{2i} & i = 1:n
\end{array}$$

Beweis: Wir betrachten wieder stellvertretend für alle Formeln

$$\check{r}^+_i = -\mu^{-2}\hat{p}_i\check{s}_i = -\tilde{r}^+_{2i-1}\tilde{r}^+_{2i}$$

□

Bemerkung 3.9 (Kopplungen in Matrix-Notation)
Wir nehmen an, daß wir durch Lösen von $\hat{N}^+_k\hat{G}^+_k(\hat{N}^+_k)^T \cdot v = \hat{\gamma}^+_k e_k$ und $\check{N}^+_k\check{G}^+_k(\check{N}^+_k)^T \cdot u = \check{\gamma}^+_k e_k$ ein singuläres Tripel (σ, u, v) von B *exakt* bestimmen können. Ein Vergleich der Komponenten von u und v ergibt für $i < k$

$$\frac{u_i}{v_i} = \prod_{j=i}^{k-1}\frac{\check{l}^+_j}{\hat{l}^+_j} = \prod_{j=i}^{k-1}\frac{\tilde{l}^+_{2j+1}}{\tilde{l}^+_{2j-1}} = \frac{\tilde{l}^+_{2k-1}}{\tilde{l}^+_{2i-1}}$$

und für $i \geq k$

$$\frac{u_{i+1}}{v_{i+1}} = \prod_{j=k}^{i}\frac{\check{u}^+_j}{\hat{u}^+_j} = \prod_{j=k}^{i}\frac{\tilde{u}^+_{2j+1}}{\tilde{u}^+_{2j-1}} = \frac{\tilde{u}^+_{2i+1}}{\tilde{u}^+_{2k-1}}\ .$$

Definieren wir also die Diagonalmatrix

$$\mathcal{X}_k := \mathtt{diag}\left(\left[\frac{\tilde{l}^+_{2k-1}}{\tilde{l}^+_1}, \ldots, \frac{\tilde{l}^+_{2k-1}}{\tilde{l}^+_{2k-3}}, 1, \frac{\tilde{u}^+_{2k+1}}{\tilde{u}^+_{2k-1}}, \ldots, \frac{\tilde{u}^+_{2n-1}}{\tilde{u}^+_{2k-1}}\right]\right)$$

so sind wir jetzt in der Lage, die klassische Umrechnung $u = \frac{1}{\sigma}B \cdot v$ zu ersetzen durch $u = \mathcal{X}_k \cdot v$.

In der Gleitpunktarithmetik ist die Verwendung der Bidiagonalmatrix kritisch, wenn beispielsweise die Einträge von B stark alternieren. Die berechnete Größe $\mathrm{fl}(\frac{1}{\sigma}B \cdot v)$ approximiert dann verursacht durch Auslöschung das tatsächliche Ergebnis u nur sehr schlecht. Diese negativen Effekte verstärken sich, wenn σ innerhalb eines Clusters liegt. Die klassische Umrechnung $u = \frac{1}{\sigma}B \cdot v$ ist also numerisch nicht praktikabel.
Nach Bemerkung 3.9 können wir zu jedem Singulärvektorpaar eine Diagonalmatrix $\mathcal{X}_k$ finden, die sich aus Daten zusammensetzt, bei denen der entsprechende Singulärwert als isoliert betrachtet werden kann. Im Gegensatz zur Umrechnung mit B werden die Komponenten der Vektoren ausschließlich unter Verwendung von Multiplikationen und Divisionen verknüpft, so daß eine Berechnung $\mathrm{fl}(\mathcal{X}_k \cdot v)$ rückwärts stabil ist.

3.2.3 Singulärvektorpaare für isolierte Singulärwerte

Wir geben nun ein Verfahren zur Bestimmung gut gekoppelter Singulärvektorpaare für isolierte Singulärwerte an. Dazu behandlen wir fünf Unterpunkte.

1. Umrechnung $\hat{Z}^+ \to \check{Z}^+$ auf der ersten Ebene.
2. Verhalten bei Breakdowns.
3. Bestimmung gekoppelter Singulärvektorpaare.
4. Minimierung des Rechenaufwands.
5. Nachiteration zur Verbesserung der Genauigkeit.

1. Umrechnung $\hat{Z}^+ \to \check{Z}^+$ auf der ersten Ebene

Das Verfahren basiert auf der Idee, die Faktorisierung von $B^TB - \mu^2 I$ explizit durchzuführen, während die Zerlegungen von $BB^T - \mu^2 I$ (und $T_{GK} - \mu I$) implizit aufgestellt werden. Zu diesem Zweck erlaubt Folgerung 3.7 die Formulierung der Umrechnung in Algorithmus 3.3 auf Seite 79. Der Datenfluß ist in Abbildung 3.8 dargestellt. Da wir ausschließlich Multiplikationen und Divisionen verwenden, ist die Transformation $\hat{Z}^+ \to \check{Z}^+$ rückwärts stabil mit hoher relativer Genauigkeit.

Bemerkung 3.10 ($\hat{Z}^+ \to \check{Z}^+$ oder $\check{Z}^+ \to \hat{Z}^+$?)
Sei wie in Bemerkung 3.4 die Matrix mit umgekehrt angeordneten Einträgen $B^{(\mathrm{rev})} = \mathtt{diag}([a_n, \ldots, a_1]) + \mathtt{diag}([b_{n-1}, \ldots, b_1]), 1)$ und eine Permutationsmatrix P mit $P \cdot x = x(n : -1 : 1)$ gegeben. Wegen $B^{(\mathrm{rev})} = PB^TP$ ist für

Algorithmus 3.3 Umrechnung $\hat{Z}^+ \to \check{Z}^+$ auf der ersten Ebene ($r = 1$).

Gegeben: $\hat{D}^+, \hat{S}, \hat{R}^+, \hat{P}, \hat{\Gamma}^+, B, \mu$
Gesucht: $\check{Z}^+ = [\check{D}^+, \check{L}^+, \check{P}, \check{R}^+, \check{U}^+, \check{S}, \check{\Gamma}^+]$

{Umrechnung $[\hat{D}^+, \hat{S}] \to [\check{D}^+, \check{L}^+, \check{P}]$}
for $i = 1 : n-1$ **do**
  $t = \frac{\hat{d}_i^+}{\hat{s}_i}$
  $\check{p}_i = -\mu^2 t$
  $\check{d}_i^+ = \hat{s}_{i+1} t$
  $\check{l}_i^+ = \frac{a_{i+1} b_i}{\check{d}_i^+}$
end for
$\check{d}_n^+ = \check{p}_n \quad \{= -\mu^2 \frac{\hat{d}_n^+}{\hat{s}_n}\}$
{Umrechnung $[\hat{R}^+, \hat{P}] \to [\check{R}^+, \check{U}^+, \check{S}]$}
$\check{s}_n = -\mu^2$
$\check{r}_n^+ = \hat{p}_n$
for $i = n-1 : -1 : 1$ **do**
  $t = \frac{\hat{r}_{i+1}^+}{\hat{p}_{i+1}}$
  $\check{s}_i = -\mu^2 t$
  $\check{r}_i^+ = \hat{p}_i t$
  $\check{u}_i^+ = \frac{a_{i+1} b_i}{\check{r}_{i+1}^+}$
end for
{Umrechnung $[\hat{\Gamma}^+, \hat{S}, \check{S}] \to \check{\Gamma}^+$}
for $i = 1 : n$ **do**
  $\check{\gamma}_i^+ = \hat{\gamma}_i^+ \frac{\check{s}_i}{\hat{s}_i}$
end for

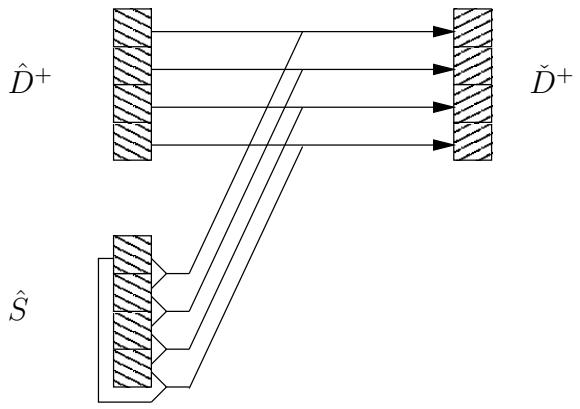


Abbildung 3.8: Datenfluß bei Umrechnung $[\hat{D}^+, \hat{S}] \to [\check{D}^+]$ (erste Ebene).

die Singulärwertzerlegungen $B = U\Sigma V^T$ und $B^{(\mathrm{rev})} = U^{(\mathrm{rev})}\Sigma(V^{(\mathrm{rev})})^T$ ein Zusammenhang gegeben durch $U = PV^{(\mathrm{rev})}$ und $V = PU^{(\mathrm{rev})}$. Bei dieser Umrechnung wirkt die Permutation auf die Zeilen der Matrizen. Wir können also ohne Beschränkung der Allgemeinheit grundsätzlich immer Kopplungen $\hat{Z}^+ \to \check{Z}^+$ betrachten. Wir geben in Abschnitt 4.1.1 an, wann es dennoch sinnvoll ist, mit $B^{(\mathrm{rev})}$ anstelle von B zu arbeiten.

2. Verhalten bei Breakdowns

Wir erörtern nun die Frage, wie sich Breakdowns bei den Faktorisierungen $B^TB - \mu^2 I = \hat{N}_k^+ \hat{G}_k^+ (\hat{N}_k^+)^T$ auf die Kopplungen auswirken. Stellvertretend für die gesamte BABE-Faktorisierung untersuchen wir hier wie in Bemerkung 2.6 (Seite 47) die Sequenz aus Gleichung (2.12)

$$\hat{d}_r^+ = 0, \quad \hat{l}_r^+ = \hat{s}_{r+1} = \hat{d}_{r+1}^+ = \infty, \quad \hat{l}_{r+1}^+ = 0, \quad \hat{s}_{r+2} = b_{r+1}^2 - \mu^2$$

In der Umrechnung $[\hat{D}^+, \hat{S}] \to [\check{D}^+, \check{L}^+, \check{P}]$ können wir die problematischen Elemente wie folgt umgehen:

$$\check{p}_r = -\mu^2 \tfrac{\hat{d}_r^+}{\hat{s}_r} = 0, \qquad \check{d}_r^+ = \check{p}_r + b_r^2 = b_r^2, \qquad \check{l}_r^+ = \tfrac{a_{r+1}}{b_r}$$
$$\check{p}_{r+1} = \tfrac{a_{r+1}\check{l}_r^+}{b_r}\check{p}_r - \mu^2 = -\mu^2, \qquad \check{d}_{r+1}^+ = \check{p}_{r+1} + b_{r+1}^2 = b_{r+1}^2 - \mu^2 = \hat{s}_{r+2}$$

Wir beachten weiter, daß wegen $\check{p}_r = \mu\tilde{d}_{2r}^+ = 0$

$$\hat{d}_r^+ = 0 \quad \Rightarrow \quad \tilde{d}_{2r}^+ = 0, \quad \tilde{d}_{2r+1}^+ = \infty$$

ein Breakdown der LDL^T-Faktorisierung von B^TB immer gekoppelt ist mit dem Auftreten einer Null in einem gradzahligem Pivotelement der impliziten LDL^T-Faktorisierung von T_{GK}. Dennoch ist das Produkt von $\tilde{d}_{2r}^+$ und $\tilde{d}_{2r+1}^+$ wegen $\check{d}_r^+ = b_r^2$ wohlbestimmt. Wird bei der impliziten Faktorisierung von T_{GK} ein ungerades Pivotelement Null, so erhalten wir wegen $0 = -\tilde{d}_{2r-2}^+\tilde{d}_{2r-1}^+ = \check{d}_{r-1}^+$ folgende Sequenz:

$$\check{d}_{r-1}^+ = 0, \quad \check{l}_{r-1}^+ = \check{p}_r = \check{d}_r^+ = \infty, \quad \check{l}_r^+ = 0, \quad \check{p}_{r+1} = a_{r+1}^2 - \mu^2$$

Wir sehen eine vorteilhafte Eigenschaft von Algorithmus 3.3 darin, daß er in diesem Falle keiner Modifikationen bedarf, wenn er für einen Rechner mit IEEE-Norm [2] implementiert wird. (Beachte $\hat{s}_r = 0 \Rightarrow \hat{s}_{r+1} = -\mu^2$. Auch hier ist das Produkt von $\tilde{d}_{2r-1}^+$ und $\tilde{d}_{2r}^+$ wegen $\hat{d}_r^+ = a_r^2$ wohlbestimmt.)

3. Bestimmung gekoppelter Singulärvektorpaare

Wir können die Ergebnisse des in Algorithmus 3.4 vorgeschlagenen Verfahrens wie folgt interpretieren. Sei

$$B^TB - \mu^2 I = \hat{N}_{\hat{k}}^+ \hat{G}_{\hat{k}}^+ (\hat{N}_{\hat{k}}^+)^T$$

Algorithmus 3.4 Bestimmung eines Singulärvektorpaares für isoliertes μ^2.

Gegeben: B, μ

Gesucht: Näherungen $(\bar{u}, \bar{v})$ an die zu μ gehörenden Singulärvektoren.

Faktorisiere $B^TB - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T = \hat{U}^+\hat{R}^+(\hat{U}^+)^T = \hat{N}^+_{\hat{k}}\hat{G}^+_{\hat{k}}(\hat{N}^+_{\hat{k}})^T$.

Löse $(\hat{N}^+_{\hat{k}}\hat{G}^+_{\hat{k}}(\hat{N}^+_{\hat{k}})^T)\cdot\bar{v} = \hat{\gamma}_{\hat{k}}e_{\hat{k}}$.

Verwende Algorithmus 3.3 für die Umrechnung $\hat{Z}^+ \to \check{Z}^+$.

Löse $(\check{N}^+_{\check{k}}\check{G}^+_{\check{k}}(\check{N}^+_{\check{k}})^T)\cdot\bar{u} = \check{\gamma}_{\check{k}}e_{\check{k}}$.

Kennen wir $\bar{v}$ als Lösung von

$$\hat{N}^+_{\hat{k}}\hat{G}^+_{\hat{k}}(\hat{N}^+_{\hat{k}})^T\cdot\bar{v} = \hat{\gamma}^+_{\hat{k}}e_{\hat{k}}$$

so können wir $\bar{u}$ als Lösung von

$$\check{N}^+_{\check{k}}\check{G}^+_{\check{k}}(\check{N}^+_{\check{k}})^T\cdot\bar{u} = \check{\gamma}^+_{\check{k}}e_{\check{k}}$$

bestimmen, *ohne* die Größen $\check{N}^+_{\check{k}}$ und $\check{G}^+_{\check{k}}$ durch eine explizite Faktorisierung

$$BB^T - \mu^2 I = \check{N}^+_{\check{k}}\check{G}^+_{\check{k}}(\check{N}^+_{\check{k}})^T$$

zu ermitteln.
Abbildung 3.9 zeigt alle Möglichkeiten, um Faktorisierungen implizit zu koppeln. Sind beispielsweise die Daten $\tilde{Z}^+$ aus einer expliziten Faktorisierung einer Translation der Golub-Kahan Matrix gegeben, so können wir die Daten der übrigen Faktorisierungen implizit bestimmen usw. Alle Transformationen sind rückwärts stabil.

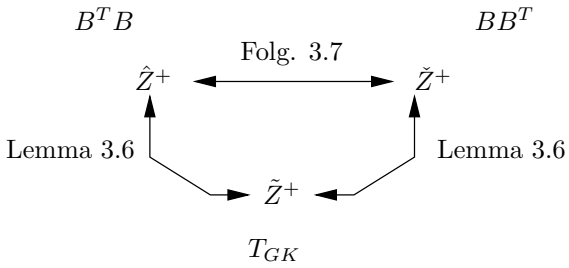


Abbildung 3.9: Rückwärts stabile Kopplungen auf der ersten Ebene.

Der Rest dieses Abschnitts enthält einige Hinweise für eine effiziente Implementierung von Algorithmus 3.4. Wir betrachten Möglichkeiten zur Verbesserung der Ausführungsgeschwindigkeit und der numerischen Genauigkeit.

4. Minimierung des Rechenaufwands

Für die Daten von U und V müssen wir $2n^2$ Gleitpunktzahlen abspeichern. Da der RRR-Algorithmus nur einen Bedarf von $\mathcal{O}(n)$ an zusätzlichem Arbeitsspeicher benötigt, lohnt es sich, in einer Vorverarbeitungsphase Größen wie $a_i b_i$, a_i^2 oder $1/a_i$ einmalig zu berechnen und separat abzuspeichern, um bei nachfolgenden Operationen Rechenaufwand einzusparen.
Wir betrachten noch einmal eine dahingehend optimierte Version der linken Seite von Algorithmus 3.1 zur Faktorisierung von $B^T B - \mu^2 I$:

1: $\tau = -\mu^2$
2: $\hat{s}_1 = \tau$
{2 Additionen, 3 Multiplikationen, 1 Division pro Durchlauf}
3: **for** $i = 1 : n-1$ **do**
4: $\quad \hat{d}_i^+ = \hat{s}_i + a_i^2$
5: $\quad \widehat{dd}_i^+ = \frac{1}{\hat{d}_i^+}$
6: $\quad \hat{l}_i^+ = a_i b_i \cdot \widehat{dd}_i^+$
7: $\quad \hat{s}_{i+1} = b_i^2 \cdot \hat{s}_i \cdot \widehat{dd}_i^+ + \tau$
8: **end for**
9: $\hat{d}_n^+ = \hat{s}_n + a_n^2$
10: $\hat{p}_n = a_n^2 + \tau$
{2 Additionen, 3 Multiplikationen, 1 Division pro Durchlauf}
11: **for** $i = n-1 : -1 : 1$ **do**
12: $\quad \hat{r}_{i+1}^+ = \hat{p}_{i+1} + b_i^2$
13: $\quad \widehat{rr}_{i+1}^+ = \frac{1}{\hat{r}_{i+1}^+}$
14: $\quad \hat{u}_i^+ = a_i b_i \cdot \widehat{rr}_{i+1}^+$
15: $\quad \widehat{pp}_i = a_i^2 \cdot \hat{p}_{i+1} \cdot \widehat{rr}_{i+1}^+$
16: $\quad \hat{p}_i = \widehat{pp}_i + \tau$
17: **end for**
18: $\hat{r}_1^+ = \hat{p}_1$
19: **for** $i = 1 : n$ **do**
20: $\quad \hat{\gamma}_i^+ = \hat{s}_i + \widehat{pp}_i$
21: **end for**

Für die in dieser Weise durchgeführten LDL^T- und URU^T-Faktorisierungen ist also jeweils ein Aufwand von ca. $2n$ Additionen, $3n$ Multiplikationen und n Divisionen zu verzeichnen. Die Bestimmung von $\hat{\Gamma}^+$ erfolgt dann mit n zusätzlichen Additionen und Vergleichsoperationen zur Bestimmung des betragsminimalen Twist-Elementes.

Kennen wir *von vorneherein* einen Index $\hat{k}$, so daß $|\hat{\gamma}_{\hat{k}}^+|$ hinreichend klein ist, so benötigen wir zum Aufstellen des rechten Singulärvektors lediglich die Daten von $\hat{N}_{\hat{k}}^+$ und können somit mehr als die Hälfte der Operationen einsparen. Einige Heuristiken zur Bestimmung von $\hat{k}$ ohne vollständige Berechnung von $\hat{\Gamma}^+$ sind in [26, 29] beschrieben und in der danach entwickelten LAPACK-Routine `DSTEGR` implementiert, vgl. Abschnitt 4.3.
Das Lösen von $(\hat{N}_{\hat{k}}^+ \hat{G}_{\hat{k}}^+ (\hat{N}_{\hat{k}}^+)^T) \cdot v = \hat{\gamma}_{\hat{k}}^+ e_{\hat{k}}$ ist mit einem Aufwand von n Multiplikationen verbunden. Die vollständige Umrechnung $\hat{Z}^+ \to \check{Z}^+$ mit Algorithmus 3.3 hat einen Aufwand von je $5n$ Multiplikationen und Divisionen. Für das Lösen von $(\check{N}_{\check{k}}^+ \check{G}_{\check{k}}^+ (\check{N}_{\check{k}}^+)^T) \cdot u = \check{\gamma}_{\check{k}}^+ e_{\check{k}}$ kommen auch hier n Multiplikationen hinzu. Wir zeigen nun einige Einsparpotentiale auf.

Bemerkung 3.11 (Optimierungsoption I)
Als erste Optimierungsmöglichkeit betrachten wir den Fall $|\check{\gamma}_{\hat{k}}^+| = \mathcal{O}(|\hat{\gamma}_{\hat{k}}^+|)$. Wir nehmen also an, daß durch den Twist-Index $\hat{k}$ ein hinreichend betragskleines Element von $\check{\Gamma}^+$ festgelegt ist. Diese leicht nachzuprüfende Bedingung ist für viele Matrizen erfüllt. Wir benötigen dann lediglich einen Teil der Größen aus Algorithmus 3.3 und können deswegen erhebliche Einsparungen an Rechenaufwand erzielen.

Sei k ein geeignet gewählter Index, z.B. $k = \hat{k}$
Berechne $\check{s}_k$ nach Folgerung 3.7: $\check{s}_k = \hat{s}_k \frac{\check{\gamma}_k^+}{\hat{\gamma}_k^+}$
if $|\check{s}_k| = \mathcal{O}(|\hat{s}_{\hat{k}}|)$ **then**
  for $i = 1 : k-1$ **do**
    $\check{l}_i^+ = a_{i+1} b_i \cdot \widehat{dd}_i^+ \cdot \frac{\hat{s}_i}{\hat{s}_{i+1}}$ {2 Multiplikationen, 1 Division}
  end for
  for $i = k : n-1$ **do**
    $\check{u}_i^+ = a_{i+1} b_i \cdot \widehat{r r}_{i+2}^+ \cdot \frac{\hat{p}_{i+1}}{\hat{p}_{i+2}}$ {2 Multiplikationen, 1 Division}
    {Es ist $\check{u}_{n-1}^+ = \frac{a_n b_{n-1}}{\hat{r}_n^+}$.}
  end for
end if

Die Bestimmung einer Näherung an den gekoppelten linken Singulärvektor hat also in dieser Form einen Aufwand von insgesamt $3n$ Multiplikationen und n Divisionen. Sollte $\check{\gamma}_k^+ = \mathcal{O}(\hat{\gamma}_{\hat{k}}^+)$ für $k = \hat{k}$ nicht erfüllt sein, so können wir inkrementell zunächst nach weiteren betragskleinen Elementen von $\check{\Gamma}^+$ suchen, indem wir die Informationen über $\hat{\Gamma}^+$ geschickt ausnutzen. Die Berechnung der Nebendiagonalelemente führen wir erst dann aus, wenn ein geeigneter Index k gefunden ist.

Bemerkung 3.12 (Optimierungsoption II)
Ausgehend von Bemerkung 3.9 können wir im Fall $|\check{\gamma}_k^+| = \mathcal{O}(|\hat{\gamma}_{\hat{k}}^+|)$ die Berechnung von $\check{L}^+$ und $\check{U}^+$ vollständig einsparen. Nach dem Lösen von

$$(B^T B - \mu^2 I) \cdot v = (\hat{N}_k^+ \hat{G}_k^+ (\hat{N}_k^+)^T) \cdot v = \hat{\gamma}_k^+ e_k$$

bestimmen wir die Komponenten für die Näherung an den linken Singulärvektor wie folgt:

$$u_k = v_k = 1$$

$$u_i = \frac{\tilde{u}_{2i-1}^+}{\tilde{u}_{2k-1}^+} v_i = \frac{a_k}{\widehat{pp}_k} \cdot \widehat{pp}_i \cdot \frac{1}{a_i} \cdot v_i \quad \text{für} \quad i = 1 : k-1$$

$$u_i = \frac{\tilde{l}_{2k-1}^+}{\tilde{l}_{2i-1}^+} v_i = \frac{a_k}{\hat{s}_k} \cdot \hat{s}_i \cdot \frac{1}{a_i} \cdot v_i \quad \text{für} \quad i = k+1 : n$$

Berechnen wir die beiden Quotienten $\frac{a_k}{\widehat{pp}_k}$ und $\frac{a_k}{\hat{s}_k}$ voraus, können wir im Vergleich zu dem in der vorigen Bemerkung dargestellten Verfahren auf weitere n Divisionen verzichten. Allerdings beschreiben die Größen $\widehat{pp}_i$ die *ungeshifteten* Hilfsvariablen der Quotienten-Differenzen-Transformation aus Algorithmus 3.3. Numerische Experimente zeigen, daß die Qualität der Näherung an u schlechter ist als bei Verwendung der Optimierungsoption I.

Bemerkung 3.13 (Weitere Optimierungsoptionen)
Für große n sind oft viele Komponenten von u und v vernachlässigbar klein und können auf Null gesetzt werden, ohne unsere numerischen Genauigkeitsanforderungen zu beeinträchtigen. Wir beschreiben in Abschnitt 4.2 detailliert weitere Möglichkeiten zur Aufwandsminimierung, die sich aus dieser Beobachtung ergeben (Stichwort Ideale Systeme).

5. Nachiteration zur Verbesserung der Genauigkeit

Ist eine Verbesserung der Genauigkeit unserer Näherungen erwünscht, empfiehlt sich die Durchführung einer Nachiteration. Im Anschluß an die Bearbeitung von

$$(N_k^+ G_k^+ (N_k^+)^T) \cdot x = \gamma_k^+ e_k$$

erfolgt eine weitere Lösung des LGS

$$(N_k^+ G_k^+ (N_k^+)^T) \cdot z = x$$

mit einem zusätzlichen Aufwand von $3n$ Multiplikationen und $2n$ Subtraktionen. Die Daten von $(G_k^+)^{-1}$ werden bereits bei den Faktorisierungen benötigt und können separat abgespeichert werden. Bei isolierten Eigen- bzw. Singulärwerten kann durch Nachiteration eine weitere Verbesserung der numerischen Qualität der Näherungen erzielt werden.

3.2.4 Vergleich der expliziten mit der gekoppelten Faktorisierung

Wir vergleichen nun numerische Eigenschaften der expliziten Faktorisierung mit der gekoppelten Variante in der Gleitpunktarithmetik. Dazu bezeichnen wir mit $[\check{D}^+, \check{L}^+, \check{P}]$ die Daten, die durch explizites Faktorisieren

$$BB^T - \mu^2 I = \check{L}^+ \check{D}^+ (\check{L}^+)^T$$

mit der rechten Seite von Algorithmus 3.1 generiert werden. Auf der anderen Seite haben wir Daten, die aus der Umrechnung

$$[\hat{D}^+, \hat{S}] \longrightarrow [\check{\hat{D}}^+, \check{\hat{L}}^+, \check{\hat{P}}]$$

durch Anwenden von Algorithmus 3.3 entstehen.

Eigenwerte der gekoppelten Faktorisierungen

Bei der Rechnung in exakter Arithmetik sind die Eigenwerte von $\hat{L}^+ \hat{D}^+ (\hat{L}^+)^T$, $\check{L}^+ \check{D}^+ (\check{L}^+)^T$ und $\check{\hat{L}}^+ \check{\hat{D}}^+ (\check{\hat{L}}^+)^T$ identisch. Bei der Durchführung in der Gleitpunktarithmetik haben wir in Satz 3.1 (Seite 61) bereits folgende Abweichung der interessierenden Eigenwerte nachgewiesen:

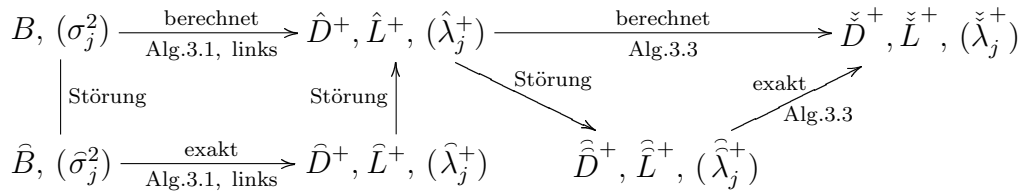
$$\left| \hat{\lambda}_j^+ - \check{\lambda}_j^+ \right| = \mathcal{O}(\sigma_j^2 \epsilon) \quad \text{für} \quad j = f : l$$

Kopplungen bringen hier eine deutliche Verbesserung. Im folgenden Satz zeigen wir:

$$\left| \frac{\hat{\lambda}_j^+ - \check{\hat{\lambda}}_j^+}{\hat{\lambda}_j^+} \right| = \mathcal{O}(\epsilon) \quad \text{für} \quad j = f : l$$

Die Eigenwerte von $\hat{L}^+ \hat{D}^+ (\hat{L}^+)^T$ und $\check{\hat{L}}^+ \check{\hat{D}}^+ (\check{\hat{L}}^+)^T$ haben also einen kleinen *relativen* Abstand. Die numerisch berechneten Daten dieser Matrizen sind also gut miteinander gekoppelt. Diese Eigenschaft überträgt sich auch auf die BABE-Faktorisierungen. Wir können also erwarten, daß die daraus gebildeten Approximationen an die *Singulärvektorpaare* gut gekoppelt sind (kleine Residuen). Auch die numerische Orthogonalität im Rahmen des RRR-Algorithmus ist gewährleistet: Ist $\hat{\lambda}_j^+$ isoliert, so gilt dies auch für $\check{\hat{\lambda}}_j^+$. Für die Eigenwerte der getrennt berechneten Faktorisierungen (vgl. Satz 3.1) kann dies i.a. nicht garantiert werden.

Satz 3.14 (Über den Abstand der Eigenwerte $\hat{\lambda}_j^+$ und $\breve{\hat{\lambda}}_j^+$)
Wir betrachten eine Erweiterung der kommutierenden Diagramme aus Satz 3.1:



Wir setzen weiter voraus, daß $\hat{D}^+$ und $\hat{L}^+$ eine partielle RRR für die interessierenden Eigenwerte bilden. Dann gilt

$$\breve{\hat{\lambda}}_j^+ = \hat{\lambda}_j^+(1 + \widehat{\hat{\kappa}}^{(j)}\epsilon) \quad \text{für} \quad j = f:l \tag{3.6}$$

Beweis: Es existiert eine Bidiagonalmatrix $\hat{B} = \texttt{diag}([\hat{a}_1, \ldots, \hat{a}_n]) + \texttt{diag}([\hat{b}_1, \ldots, \hat{b}_{n-1}]), 1)$, so daß

$$\hat{B}^T\hat{B} - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$$

exakt erfüllt ist. Wir betrachten die Rechnung in exakter Arithmetik

$$\begin{aligned} \breve{\hat{d}}_i^+ &= (\widehat{\hat{s}}_{i+1}\widehat{\hat{d}}_i^+)/\widehat{\hat{s}}_i & (A1)\\ \breve{\hat{l}}_i^+ &= (\widehat{\hat{a}}_{i+1}\widehat{\hat{b}}_i\widehat{\hat{s}}_i)/(\widehat{\hat{s}}_{i+1}\widehat{\hat{d}}_i^+) & (B1) \end{aligned}$$

und im Vergleich dazu die Analyse von Algorithmus 3.3 nach dem Modell der Gleitpunktarithmetik (Gleichung 1.3.2):

$$\begin{aligned} \breve{\hat{d}}_i^+ &= (\hat{s}_{i+1}\hat{d}_i^+)/\hat{s}_i \cdot (1+\epsilon_*^{(i+1)})(1+\epsilon_/) & (A2)\\ \breve{\hat{l}}_i^+ &= (\hat{a}_{i+1}\hat{b}_i)/\breve{\hat{d}}_i^+ \cdot (1+\epsilon_{\hat{a}\hat{b}})(1+\epsilon_{//})\\ &= (\hat{a}_{i+1}\hat{b}_i\hat{s}_i)/(\hat{s}_{i+1}\hat{d}_i^+) \cdot \tfrac{(1+\epsilon_{\hat{a}\hat{b}})(1+\epsilon_{//})}{(1+\epsilon_*^{(i+1)})(1+\epsilon_/)} & (B2) \end{aligned}$$

Mit Ausnahme von $\epsilon_*^{(i+1)}$ lassen wir für die multiplikativen Fehlerterme wieder die Indizes weg. Als Ansatz für die Störung wählen wir

$$\begin{aligned} \widehat{\hat{s}}_{i+1} &= \hat{s}_{i+1}(1+\epsilon_*^{(i+1)})\\ \widehat{\hat{d}}_i^+ &= \hat{d}_i^+(1+\epsilon_*^{(i)})(1+\epsilon_/)\\ \widehat{\hat{a}}_i &= \hat{a}_i(1+\epsilon_{\hat{a}\hat{b}})\\ \widehat{\hat{b}}_i &= \hat{b}_i(1+\epsilon_{//}) \end{aligned}$$

Ein Vergleich der rechten Seiten von $(A1)$ und $(A2)$ bzw. $(B1)$ und $(B2)$ vervollständigt die Rückwärtsanalyse.

Aus der Identität $\hat{\tilde{B}}\hat{\tilde{B}}^T - \mu^2 I = \check{\tilde{L}}^+\check{\tilde{D}}^+(\check{\tilde{L}}^+)^T$ erhalten wir $\check{\tilde{\lambda}}_j^+ = \hat{\tilde{\lambda}}_j^+$. Da $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ die interessierenden Eigenwerte auf hohe relative Genauigkeit bestimmt, folgt daraus die Behauptung

$$\check{\tilde{\lambda}}_j^+ = \hat{\lambda}_j^+(1 + \hat{\tilde{\kappa}}^{(j)}\epsilon)$$

mit einem moderaten Wert für $\hat{\tilde{\kappa}}^{(j)}$. □

Für den vorigen Satz haben wir vorausgesetzt, daß die (typischerweise) indefinite Matrix $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ eine partielle RRR bildet. Diese Eigenschaft läßt sich beispielsweise mit den *a priori* und *a posteriori* Kriterien aus Abschnitt 2.4 nachprüfen. Ferner verzichten wir nach Bemerkung 1.28 auf genauere Abschätzungen für den Fehler zwischen $\check{\tilde{\lambda}}_j^+$ und $\hat{\lambda}_j^+$. Es reicht zu wissen, daß $\hat{\tilde{\kappa}}^{(j)}$ kontrollierbar bleibt.

Auswirkungen bei numerischen Experimenten

Wir betrachten wie in Abschnitt 3.1.1 wieder die Bidiagonalmatrix B als Cholesky-Faktor einer Wilkinson-Matrix gemäß $B^TB = W_{21}^+ - \tau I$. Diesmal wenden wir unsere Implementierung des RRR-Algorithmus explizit nur auf B^TB an, während die Faktorisierungen von BB^T jetzt als Kopplungen mit Hilfe der Algorithmen 3.3 und 3.4 erzeugt werden. Der Darstellungsbaum für die Arbeit auf B^TB ist wieder durch Abbildung 3.2 skizziert. Beim Vergleich der Eigenwertapproximationen auf der ersten Ebene stellt sich nun heraus, daß die jeweils interessierenden Näherungen an $\hat{\lambda}_j^+$ und $\check{\tilde{\lambda}}_j^+$ sich nur um sehr wenige ULP voneinander unterscheiden. Für dieses Beispiel beobachten wir im numerischen Experiment mit $\max_{j=8:21}\{\hat{\tilde{\kappa}}^{(j)}\} \leq 2$ eine extrem kleine Abweichung. In vielen Fällen sind die jeweiligen Näherungen sogar identisch. Im Vergleich zu den in Tabelle 3.1 aufgelisteten Unterschieden zwischen $\hat{\lambda}_j^+$ und $\check{\lambda}_j^+$ wird der Vorteil von Kopplungsmechanismen deutlich.
Dieser schlägt sich auch bei der Güte der Kopplung der Singulärvektoren nieder. Im Falle der geclusterten Eigenwerte faktorisieren wir dazu

$$\hat{L}^+\hat{D}^+(\hat{L}^+)^T - \hat{\lambda}_j^+ I \quad \text{und} \quad \check{\tilde{L}}^+\check{\tilde{D}}^+(\check{\tilde{L}}^+)^T - \check{\tilde{\lambda}}_j^+ I$$

und lösen entsprechende Gleichungssysteme, um Approximationen an die rechten und linken Singulärvektoren zu erhalten. In diesem Beispiel ergibt sich

$$\max_j\{\|B\bar{V}(:,j) - \bar{\sigma}_j\bar{U}(:,j)\|\} < 0.256 \cdot n\epsilon\sigma_n$$

Eine weitere Beobachtung

Für numerisch gewonnene Daten $[\check{D}^+, \check{L}^+]$ (explizite Faktorisierung) und $[\check{\check{D}}^+, \check{\check{L}}^+]$ (Kopplung) läßt sich noch Folgendes feststellen: Vergleichen wir die Einträge von $\check{D}^+$ und $\check{\check{D}}^+$ komponentenweise, so ergeben sich deutliche Unterschiede. Für die ausmultiplizierten Matrizen ist jedoch

$$\left\| \check{L}^+ \check{D}^+ (\check{L}^+)^T - \check{\check{L}}^+ \check{\check{D}}^+ (\check{\check{L}}^+)^T \right\|_1$$

klein. Die Gleichung $BB^T - \mu^2 I = \check{\check{L}}^+ \check{\check{D}}^+ (\check{\check{L}}^+)^T$ wird also genauso gut erfüllt wie bei einer expliziten Faktorisierung. Darüberhinaus enthalten die Daten $[\check{\check{D}}^+, \check{\check{L}}^+]$ aber noch wichtige Bezüge zu $[\hat{D}^+, \hat{L}^+]$.

3.2.5 Der Anwendungsbereich von Kopplungen auf der ersten Ebene

Wir fassen die bisher erzielten neuen Ergebnisse zusammen. Zur Ermittlung von Singulärvektorpaaren auf der ersten Ebene ($r = 1$) verwenden wir Kopplungsmechanismen für die Umrechnungen der Zerlegungen

$$\begin{array}{rcccl} B^T B - \mu^2 I & = & \hat{L}^{(1)} \hat{D}^{(1)} (\hat{L}^{(1)})^T & = & \hat{U}^{(1)} \hat{R}^{(1)} (\hat{U}^{(1)})^T \\ BB^T - \mu^2 I & = & \check{L}^{(1)} \check{D}^{(1)} (\check{L}^{(1)})^T & = & \check{U}^{(1)} \check{R}^{(1)} (\check{U}^{(1)})^T \\ T_{GK} - \mu I & = & \tilde{L}^{(1)} \tilde{D}^{(1)} (\tilde{L}^{(1)})^T & = & \tilde{U}^{(1)} \tilde{R}^{(1)} (\tilde{U}^{(1)})^T \end{array}$$

und der zugehörigen BABE-Faktorisierungen.
Für isolierte Singulärwerte können wir mit Hilfe von Algorithmus 3.4 und den vorgeschlagenen Optimierungsoptionen gut gekoppelte Singulärvektorpaare erzeugen.
Bei geclusterten Singulärwerten bestimmen wir $B^T B - \mu^2 I = \hat{L}^{(1)} \hat{D}^{(1)} (\hat{L}^{(1)})^T$ explizit durch eine Faktorisierung und $\check{\check{L}}^{(1)} \check{\check{D}}^{(1)} (\check{\check{L}}^{(1)})^T$ durch implizites Koppeln. Satz 3.14 zeigt, daß bei den interessierenden Eigenwerten $\hat{\lambda}_j^{(1)}$ und $\check{\check{\lambda}}_j^{(1)}$ nur kleine *relative* Abweichungen auftreten. Sind diese Eigenwerte isoliert, können wir also Faktorisierungen

$$\hat{L}^{(1)} \hat{D}^{(1)} (\hat{L}^{(1)})^T - \hat{\lambda}_j^{(1)} I = \hat{N}_k^{(2)} \hat{G}_k^{(2)} (\hat{N}_k^{(2)})^T, \quad k = 1 : n$$

$$\check{\check{L}}^{(1)} \check{\check{D}}^{(1)} (\check{\check{L}}^{(1)})^T - \check{\check{\lambda}}_j^{(1)} I = \check{N}_k^{(2)} \check{G}_k^{(2)} (\check{N}_k^{(2)})^T, \quad k = 1 : n$$

bilden und durch Lösen von

$$(\hat{N}_{\hat{k}}^{(2)} \hat{G}_{\hat{k}}^{(2)} (\hat{N}_{\hat{k}}^{(2)})^T) \cdot \bar{v} = \hat{\gamma}_{\hat{k}} e_{\hat{k}}$$

$$(\check{N}_{\check{k}}^{(2)}\check{G}_{\check{k}}^{(2)}(\check{N}_{\check{k}}^{(2)})^T)\cdot\bar{u}=\check{\gamma}_{\check{k}}e_{\check{k}}$$

Näherungen an die Singulärvektorpaare erhalten, die sowohl numerisch orthogonal als auch gut gekoppelt sind. Mit der vorgeschlagenen Strategie ist die Übertragung des RRR-Algorithmus auf die **bSVD** für eine Vielzahl von Problemklassen abgeschlossen, vgl. Abbildung 3.10.

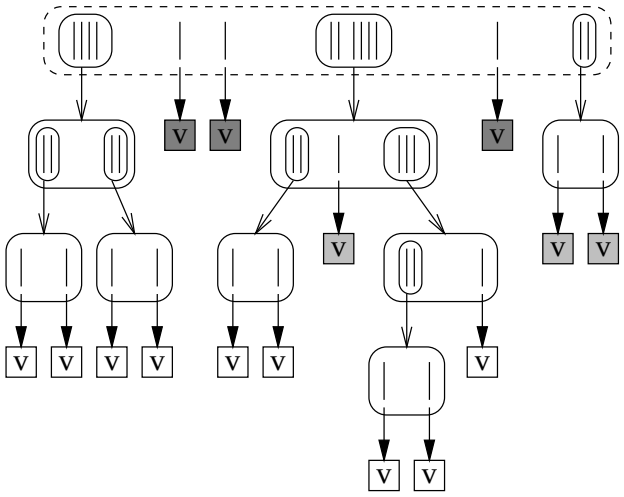


Abbildung 3.10: Anwendungsbereich von Kopplungen auf der ersten Ebene. Die dunkelgrau unterlegten Vektoren können komponentenweise gekoppelt werden. Für die hellgrau unterlegten Vektoren sind immerhin die Daten von $\hat{L}^{(1)}\hat{D}^{(1)}(\hat{L}^{(1)})^T$ und $\check{L}^{(1)}\check{D}^{(1)}(\check{L}^{(1)})^T$ gekoppelt. Damit wissen wir zumindest, daß die Eigenwerte nur um kleine relative Abstände voneinander abweichen.

Es bleibt zu klären, wie Kopplungen auf höheren Stufen ($r>1$) beschrieben und angewendet werden können.

3.3 Kopplungen auf höheren Ebenen

Wir untersuchen im folgenden Beziehungen auf höheren Ebenen im Darstellungsbaum ($r>1$). Faktorisierungen basierend auf B^TB, BB^T bzw. T_{GK} stellen wir als Matrix-Tupel $\hat{Z}^{(r)}$, $\check{Z}^{(r)}$ bzw. $\tilde{Z}^{(r)}$ dar, vgl. Abschnitt 3.1.3. Zur Vereinfachung verwenden wir innerhalb dieses Abschnitts jedoch wieder abkürzende Darstellungen. So schreiben wir $\mu=\mu^{(r)}$ und $\nu=\nu^{(r)}$ für die Shift-Parameter und sparen uns die Indizes bei Faktorisierungen wie

$$\begin{aligned}\hat{L}^{(r-1)}\hat{D}^{(r-1)}(\hat{L}^{(r-1)})^T-\nu^{(r)}I &= \hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T=\hat{U}^{(r)}\hat{R}^{(r)}(\hat{U}^{(r)})^T\\ &= \hat{N}_k^{(r)}\hat{G}_k^{(r)}(\hat{N}_k^{(r)})^T,\quad k=1\!:\!n\end{aligned}$$

indem wir sie wie folgt notieren:

$$\begin{aligned}\hat{L}\hat{D}\hat{L}^T - \nu I &= \hat{L}^+\hat{D}^+(\hat{L}^+)^T = \hat{U}^+\hat{R}^+(\hat{U}^+)^T \\ &= \hat{N}_k^+\hat{G}_k^+(\hat{N}_k^+)^T, \quad k = 1\!:\!n\end{aligned}$$

Der entscheidende Unterschied zur Situation auf der ersten Ebene ist, daß die Matrizen auf der linken Seite nun selbst bereits als LDL^T-Faktorisierung vorliegen und nicht durch die Daten der Eingangsmatrix B gegeben sind. Während die Matrix B^TB positiv definit ist, gilt dies für $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$ i.a. nicht. Entsprechend finden wir auf höheren Ebenen eine vergleichbare Eigenschaft zu $\texttt{diag}(T_{GK}) = 0$ nicht mehr vor. Insgesamt treffen wir hier also auf schwächere Voraussetzungen für Kopplungsmechanismen.

3.3.1 Faktorisierungen

Algorithmus 3.5 zeigt, wie die n BABE-Faktorisierungen auf höheren Ebenen bestimmt werden können. Wir verwenden dabei wieder Hilfsvariablen

$$s_{i+1} = d_{i+1}^+ - d_{i+1} = l_i^+ l_i s_i - \tau$$

zur Berechnung der Zerlegung $LDL^T - \tau I = L^+D^+(L^+)^T$ und

$$p_i = r_i^+ - d_{i-1}l_{i-1}^2 = \frac{d_i}{r_{i+1}^+}p_{i+1} - \tau$$

für $LDL^T - \tau I = U^+R^+(U^+)^T$. Zusammengenommen erhalten wir damit die sogenannte **dtwqds**-Transformation[1].

Bemerkung 3.15 (Verallgemeinerung der dtwqds-Transformation) In [29] wird gezeigt, daß für diese Transformation auch wieder eine gemischte Stabilitätsanalyse mit kleinen multiplikativen Störungen möglich ist. Wir können das Konzept von Faktorisierungen der Form

$$LDL^T - \tau I = N_k^+G_k^+(N_k^+)^T$$

erweitern auf Zerlegungen, bei denen die linke Seite nicht durch eine LDL^T- sondern auch durch eine BABE-Faktorisierung gegeben ist:

$$N_\kappa G_\kappa {N_\kappa}^T - \tau I = N_k^+G_k^+(N_k^+)^T$$

Die einschlägigen Sätze und Beweise sind im Anhang B dargestellt.

[1] **d**ifferential **tw**isted **q**uotient-**d**ifference with **s**hift

Algorithmus 3.5 Faktorisiere $LDL^T - \tau I = N_k^+ G_k^+ (N_k^+)^T, k = 1 : n.$

Gegeben: $[D, L]$, τ
Gesucht: $Z^+ = [D^+, L^+, S, R^+, U^+, P, \Gamma^+]$

{Faktorisiere $LDL^T - \tau I = L^+ D^+ (L^+)^T$.}
$s_1 = -\tau$
for $i = 1 : n-1$ **do**
  $d_i^+ = d_i + s_i$
  $l_i^+ = \frac{d_i l_i}{d_i^+}$
  $s_{i+1} = l_i^+ l_i s_i - \tau$
end for
$d_n^+ = d_n + s_n$
{Faktorisiere $LDL^T - \tau I = U^+ R^+ (U^+)^T$.}
$p_n = d_n - \tau$
for $i = n-1 : -1 : 1$ **do**
  $r_{i+1}^+ = d_i l_i^2 + p_{i+1}$
  $t = \frac{d_i}{r_{i+1}^+}$
  $u_i^+ = l_i t$
  $p_i = p_{i+1} t - \tau$
end for
$r_1^+ = p_1$
{Bestimme Γ^+}
for $i = 1 : n$ **do**
  $\gamma_i^+ = s_i + p_i + \tau$
end for

Bei der Abarbeitung eines Pfades im Darstellungsbaum sind u.U. mehrere LDL^T-Zerlegungen hintereinander auszuführen. In der folgenden Bemerkung stellen wir dar, daß diese sukzessiven Faktorisierungen störungstheoretisch zusammengefaßt werden können.

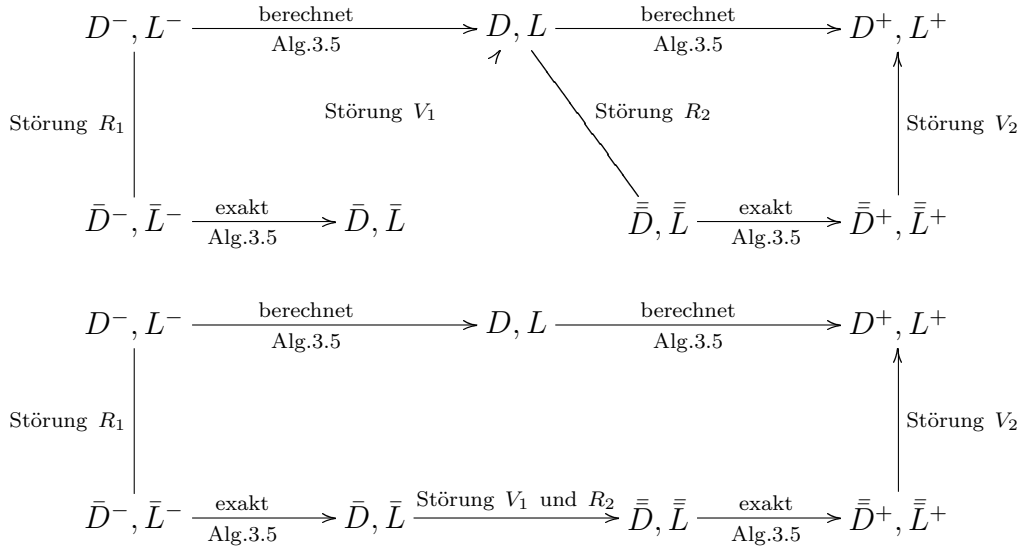
Bemerkung 3.16 (Sukzessive Faktorisierungen)
Nach [29] ist für die Faktorisierung

$$LDL^T - \tau I = L^+ D^+ (L^+)^T$$

mit dem ersten Teil von Algorithmus 3.5 eine gemischte Stabilitätsanalyse möglich. Auch die wiederholte Durchführung solcher Transformationen läßt sich mit relativer Störungstheorie beschreiben. Schalten wir eine weitere Faktorisierung

$$L^- D^- (L^-)^T - \tau^- I = LDL^T$$

vor, so sind die beiden folgenden Diagramme äquivalent:



3.3.2 Kopplungen

Wir stellen zunächst die Zusammenhänge zwischen den Faktorisierungen basierend auf B^TB, BB^T bzw. T_{GK} in exakter Arithmetik dar. Numerische Eigenschaften konkreter Umrechnungen behandeln wir in Abschnitt 3.3.3.

Beschreibung der Kopplungen auf höheren Ebenen

Wir sind zunächst an Beziehungen zwischen LDL^T-Faktorisierungen der Translationen von $\hat{L}\hat{D}\hat{L}^T$, $\check{L}\check{D}\check{L}^T$ und $\tilde{L}\tilde{D}\tilde{L}^T$ interessiert. Der nachfolgende Satz beschreibt eine Umrechnung $[\hat{D}^+, \hat{L}^+] \leftrightarrow [\tilde{D}^+, \tilde{L}^+] \leftrightarrow [\check{D}^+, \check{L}^+]$:

Satz 3.17 (Kopplung von $[\hat{D}^+, \hat{L}^+] \leftrightarrow [\tilde{D}^+, \tilde{L}^+] \leftrightarrow [\check{D}^+, \check{L}^+]$)
Wir nehmen an, daß der RRR-Algorithmus (vgl. Algorithmus 2.5) auf B^TB, BB^T und T_{GK} mit äquivalenten Shift-Strategien angewendet wird. Es gelte also $\text{path}(B^TB) = \text{path}(BB^T) = (\nu^{(1)}, \ldots, \nu^{(\text{rec})})$, während das Tupel $\text{path}(T_{GK}) = (\mu^{(1)}, \ldots, \mu^{(\text{rec})})$ gemäß der in Abschnitt 3.1.3 beschriebenen Vorschrift gewählt sei. Für ein fixiertes $r < \text{rec}$ setzen wir $\nu = \nu^{(r)}$ und $\mu = \mu^{(r)}$ und betrachten die drei Faktorisierungen

$$\begin{aligned} \hat{L}\hat{D}\hat{L}^T - \nu I &= \hat{L}^+\hat{D}^+(\hat{L}^+)^T \\ \check{L}\check{D}\check{L}^T - \nu I &= \check{L}^+\check{D}^+(\check{L}^+)^T \\ \tilde{L}\tilde{D}\tilde{L}^T - \mu I &= \tilde{L}^+\tilde{D}^+(\tilde{L}^+)^T \end{aligned}$$

Tritt bei diesen Zerlegungen kein Breakdown auf, so gilt

$$\begin{array}{rclcrcll} \hat{d}_i^+ & = & -\tilde{d}_{2i-1}^+\tilde{d}_{2i}^+ & & \check{d}_i^+ & = & -\tilde{d}_{2i}^+\tilde{d}_{2i+1}^+ & i=1:n \\ \hat{l}_i^+ & = & -\tilde{l}_{2i-1}^+\tilde{l}_{2i}^+ & & \check{l}_i^+ & = & -\tilde{l}_{2i}^+\tilde{l}_{2i+1}^+ & i=1:n-1 \end{array}$$

Dabei setzen wir $\tilde{d}_{2n+1}^+ = \tilde{d}_1^+$.

Beweis: Seien $\bar{\nu} = \sum_{j=1}^{r-1}\nu^{(j)}$ und $\bar{\mu} = \sum_{j=1}^{r-1}\mu^{(j)}$ sowie

$$\begin{aligned} H &:= \begin{pmatrix} \check{L}^+\check{D}^+(\check{L}^+)^T & 0 \\ 0 & \hat{L}^+\hat{D}^+(\hat{L}^+)^T \end{pmatrix} \qquad (3.7) \\ &= \begin{pmatrix} BB^T-(\bar{\mu}+\mu)^2 I & 0 \\ 0 & B^TB-(\bar{\mu}+\mu)^2 I \end{pmatrix} \\ &= \begin{pmatrix} (\bar{\mu}+\mu)I & B \\ B^T & (\bar{\mu}+\mu)I \end{pmatrix}\cdot\begin{pmatrix} -(\bar{\mu}+\mu)I & B \\ B^T & -(\bar{\mu}+\mu)I \end{pmatrix} \\ &= P_{ps}^T(T_{GK}+(\bar{\mu}+\mu)I)P_{ps}P_{ps}^T(T_{GK}-(\bar{\mu}+\mu)I)P_{ps} \\ &= P_{ps}^T\tilde{\tilde{L}}^+\tilde{\tilde{D}}^+(\tilde{\tilde{L}}^+)^T\check{L}^+\tilde{D}^+(\check{L}^+)^T P_{ps} \qquad (3.8) \end{aligned}$$

Die letzte Zeile ergibt sich aus Bemerkung 3.5 auf Seite 75: Die Faktorisierung $T_{GK}+(\bar{\mu}+\mu)I = \tilde{\tilde{L}}^+\tilde{\tilde{D}}^+(\tilde{\tilde{L}}^+)^T$ läßt sich aus der für $T_{GK}-(\bar{\mu}+\mu)I$ ableiten, indem wir die Vorzeichen der Einträge vertauschen.

Wir definieren nun $u, v \in \mathbb{R}^n$ und $q^+, q^- \in \mathbb{R}^{2n}$ wie folgt:

$$\begin{array}{rclcrcll} u_n & = & 1 & & v_n & = & 1 & \\ u_i & = & -\check{l}_i^+ u_{i+1} & & v_i & = & -\hat{l}_i^+ v_{i+1} & \text{für } i=n-1:-1:1 \end{array}$$

$$\begin{array}{rclcrcll} q_n^\pm & = & 1 & & q_{2n}^\pm & = & \pm 1 & \\ q_i^\pm & = & \tilde{l}_{2i}^+\tilde{l}_{2i+1}^+ q_{i+1}^\pm & & q_{i+n}^\pm & = & \tilde{l}_{2i-1}^+\tilde{l}_{2i}^+ q_{i+1+n}^\pm & \text{für } i=n-1:-1:1 \end{array}$$

Für *reguläres* H erhalten wir wegen Gleichung (3.7)

$$H\begin{pmatrix} v \\ u \end{pmatrix} = \hat{d}_n^+ e_n + \check{d}_n^+ e_{2n} \quad \text{und} \quad H\begin{pmatrix} v \\ -u \end{pmatrix} = \hat{d}_n^+ e_n - \check{d}_n^+ e_{2n}$$

und weiter

$$\operatorname{span}\left\{H\begin{pmatrix} u \\ v \end{pmatrix}, H\begin{pmatrix} u \\ -v \end{pmatrix}\right\} = \operatorname{span}\{e_n, e_{2n}\}$$

Durch Anwendung der Gleichung (3.8) ergibt sich

$$Hq^+ = -\tilde{d}_{2n-1}^+\tilde{d}_{2n}^+ e_n - \tilde{d}_1^+\tilde{d}_{2n}^+ e_{2n} \quad \text{und} \quad Hq^- = -\tilde{d}_{2n-1}^+\tilde{d}_{2n}^+ e_n + \tilde{d}_1^+\tilde{d}_{2n}^+ e_{2n}$$

und damit

$$\text{span}\{Hq^+, Hq^-\} = \text{span}\{e_n, e_{2n}\}$$

Wir haben vorausgesetzt, daß die Zerlegungen ohne Breakdown durchgeführt werden können. Zu untersuchen ist also noch der Fall $\hat{d}_n^+ = \check{d}_n^+ = \tilde{d}_{2n}^+ = 0$. Bei *singulärem* H ist $(\bar{\mu}+\mu)^2$ einfacher Eigenwert von B^TB bzw. BB^T und v bzw. u zugehöriger Eigenvektor, normiert durch $v_n = u_n = 1$. Es gilt also

$$\text{span}\left\{\begin{pmatrix} u \\ v \end{pmatrix}, \begin{pmatrix} u \\ -v \end{pmatrix}\right\} = kern\, H$$

Für singuläres H gilt ferner, daß $(\bar{\mu}+\mu)$ bzw. $-(\bar{\mu}+\mu)$ zwei einfache Eigenwerte von T_{GK} sind und q^+ bzw. q^- zugehörige Eigenvektoren, normiert durch $q_n^+ = q_n^- = 1$. Es gilt also auch

$$\text{span}\left\{q^+, q^-\right\} = kern\, H$$

und wir erhalten

$$\text{span}\left\{\begin{pmatrix} u \\ v \end{pmatrix}, \begin{pmatrix} u \\ -v \end{pmatrix}\right\} = \text{span}\{q^+, q^-\}$$

sowohl für reguläres als auch für singuläres H. Es existieren demnach Skalare $\phi^+, \phi^-, \psi^+, \psi^-$ mit

$$\left[q^+, q^-\right] = \left[\begin{pmatrix} u \\ v \end{pmatrix}, \begin{pmatrix} u \\ -v \end{pmatrix}\right] \cdot \begin{pmatrix} \phi^+ & \phi^- \\ \psi^+ & \psi^- \end{pmatrix}$$

Auswerten der Zeilen n und $2n$ dieses Systems ergeben $\phi^+ = \psi^- = 1$ und $\phi^- = \psi^+ = 0$. Daraus folgt $q^+ = \binom{u}{v}$ und $q^- = \binom{u}{-v}$. Ein Koeffizientenvergleich führt zu

$$\begin{array}{rcll} \hat{l}_i^+ & = & -\tilde{l}_{2i-1}^+ \tilde{l}_{2i}^+ & i = 1 : n-1 \\ \check{l}_i^+ & = & -\tilde{l}_{2i}^+ \tilde{l}_{2i+1}^+ & i = 1 : n-1 \end{array}$$

Die übrigen Formeln folgen aus den Identitäten

$$\begin{array}{rclrcll} a_i & = & \tilde{d}_{2i-1}^+ \tilde{l}_{2i-1}^+ & a_i b_i & = & \hat{d}_i^+ \hat{l}_i^+ & i = 1 : n-1 \\ b_i & = & \tilde{d}_{2i}^+ \tilde{l}_{2i}^+ & a_{i+1} b_i & = & \check{d}_i^+ \check{l}_i^+ & i = 1 : n-1 \end{array}$$

□

Wir erweitern dieses Resultat nun auf BABE-Faktorisierungen der Translationen von $\hat{L}\hat{D}\hat{L}^T$, $\check{L}\check{D}\check{L}^T$ und $\tilde{L}\tilde{D}\tilde{L}^T$.

Folgerung 3.18 ($[\hat{R}^+, \hat{U}^+, \hat{\Gamma}^+] \leftrightarrow [\tilde{R}^+, \tilde{U}^+, \tilde{\Gamma}^+] \leftrightarrow [\check{R}^+, \check{U}^+, \check{\Gamma}^+]$)
Die Kopplungen der BABE-Faktorisierungen

$$\begin{aligned} \hat{L}\hat{D}\hat{L}^T - \nu I &= \hat{N}^+_{\hat{k}} \hat{G}^+_{\hat{k}} (\hat{N}^+_{\hat{k}})^T \\ \check{L}\check{D}\check{L}^T - \nu I &= \check{N}^+_{\check{k}} \check{G}^+_{\check{k}} (\check{N}^+_{\check{k}})^T \\ \tilde{L}\tilde{D}\tilde{L}^T - \mu I &= \tilde{N}^+_{\tilde{k}} \tilde{G}^+_{\tilde{k}} (\tilde{N}^+_{\tilde{k}})^T \end{aligned}$$

sind vollständig beschrieben durch

$$\begin{array}{rclrcll} \hat{r}^+_i &=& -\tilde{r}^+_{2i-2}\tilde{r}^+_{2i-1} & \check{r}^+_i &=& -\tilde{r}^+_{2i-1}\tilde{r}^+_{2i} & i = 1:n \\ \hat{u}^+_i &=& -\tilde{u}^+_{2i-1}\tilde{u}^+_{2i} & \check{u}^+_i &=& -\tilde{u}^+_{2i}\tilde{u}^+_{2i+1} & i = 1:n-1 \\ \hat{\gamma}^+_i &=& (\bar{\mu}+\mu)\tilde{\gamma}^+_{2i-1} & \check{\gamma}^+_i &=& (\bar{\mu}+\mu)\tilde{\gamma}^+_{2i} & i = 1:n \end{array}$$

Dabei setzen wir $\tilde{r}^+_0 = \tilde{r}^+_{2n}$.

Beweis: Die Kopplungsformeln für $[\hat{R}^+, \hat{U}^+]$, $[\tilde{R}^+, \tilde{U}^+]$ und $[\check{R}^+, \check{U}^+]$ werden analog zum vorigen Beweis gezeigt. Die Beziehungen zwischen $[\hat{\Gamma}^+]$, $[\check{\Gamma}^+]$ und $[\tilde{\Gamma}^+]$ ergeben sich durch Ausnutzen der rekursiven Darstellung $\gamma^+_{i+1} = \frac{r^+_{i+1}}{d^+_i}\gamma^+_i$ $(i = 1 : n-1)$. □

Bemerkung 3.19 (Vergleich mit Kopplungen der ersten Ebene)
Durch Ausnutzen der speziellen Struktur der Golub-Kahan Matrix ist es bei Kopplungen auf der ersten Ebene möglich, zusätzliche Beziehungen wie etwa zwischen $\tilde{D}^+$ und $\hat{S}$ auf sehr kompakte Weise zu formulieren. Im gerade behandelten allgemeineren Fall sind diese günstigen Strukturen nicht mehr gegeben. Daher können wir hier im wesentlichen nicht mehr als die Aussagen aus Satz 3.17 und Folgerung 3.18 machen.

Mit den vorhandenen Ergebnissen können wir jedoch eine vergleichbare Interpretation wie in Bemerkung 3.9 angeben. Wir halten insbesondere fest, daß die Matrix-Notation der Kopplungen *unabhängig* von dem gewählten Verfahren zur Bildung der entsprechenden Faktorisierungen gültig ist.

Bemerkung 3.20 (Kopplungen in Matrix-Notation)
Sei $\sigma = \sum_{j=1}^{\text{rec}} \mu^{(j)}$. Wir nehmen an, daß wir (σ, u, v) exakt als Singulärtripel von B bestimmen können, indem wir $\hat{N}^+_k \hat{G}^+_k (\hat{N}^+_k)^T \cdot v = \hat{\gamma}^+_k e_k$ und $\check{N}^+_k \check{G}^+_k (\check{N}^+_k)^T \cdot u = \check{\gamma}^+_k e_k$ lösen. Definieren wir $\mathcal{X}_k$ wie in Bemerkung 3.9, so können wir auch hier die klassische Umrechnung $u = \frac{1}{\sigma} B \cdot v$ durch die Multiplikation mit einer Diagonalmatrix ersetzen: $u = \mathcal{X}_k \cdot v$.

3.3.3 Über die numerische Qualität der Kopplungen

Wie auch auf der ersten Ebene im Darstellungsbaum liefert ein Vergleich der Eigenwerte wieder ein gutes Maß für die Qualität der Kopplungen. Zu deren Beschreibung gehen wir schrittweise vor:

1. Wir betrachten die Umrechnung von $\tilde{Z}^+$ nach $[\hat{Z}^+, \check{Z}^+]$.
2. Wir formulieren einen Satz über die zugehörigen Eigenwerte.
3. Wir diskutieren dessen Konsequenzen für den RRR-Algorithmus.
4. Wir erörtern die alternative Umrechnung von $\hat{Z}^+$ über $\tilde{Z}^+$ nach $\check{Z}^+$.
5. Wir treffen einige Aussagen über die zugehörigen Eigenwerte.

1. Umrechnung von $\tilde{Z}^+$ nach $[\hat{Z}^+, \check{Z}^+]$

Mit dem Ziel, auch die Kopplungen auf höheren Ebenen in eine Implementierung des RRR-Algorithmus einzubetten, untersuchen wir nun den Einfluß der Gleitpunktarithmetik. Wir gehen zunächst davon aus, daß die Daten von $\tilde{Z}^+$ gegeben sind. Konkret betrachten wir die Umrechnung der LDL^T-Faktorisierungen, vgl. Algorithmus 3.6 und Abbildung 3.11.

Algorithmus 3.6 Umrechnung von $[\tilde{D}^+, \tilde{L}^+]$ nach $[\hat{D}^+, \hat{L}^+, \check{D}^+, \check{L}^+]$

Gegeben: $[\tilde{D}^+, \tilde{L}^+]$
Gesucht: $[\hat{D}^+, \hat{L}^+, \check{D}^+, \check{L}^+]$

1: **for** $i = 1 : n-1$ **do**
2: $\quad \hat{d}_i^+ = -\tilde{d}_{2i-1}^+ \tilde{d}_{2i}^+$
3: $\quad \check{d}_i^+ = -\tilde{d}_{2i}^+ \tilde{d}_{2i+1}^+$
4: $\quad \hat{l}_i^+ = -\tilde{l}_{2i-1}^+ \tilde{l}_{2i}^+$
5: $\quad \check{l}_i^+ = -\tilde{l}_{2i}^+ \tilde{l}_{2i+1}^+$
6: **end for**
7: $\hat{d}_n^+ = -\tilde{d}_{2n-1}^+ \tilde{d}_{2n}^+$
8: $\check{d}_n^+ = -\tilde{d}_1^+ \tilde{d}_{2n}^+$

Offensichtlich können wir diese Umrechnung als vorwärts und rückwärts stabil mit kleinen multiplikativen Störungen interpretieren.

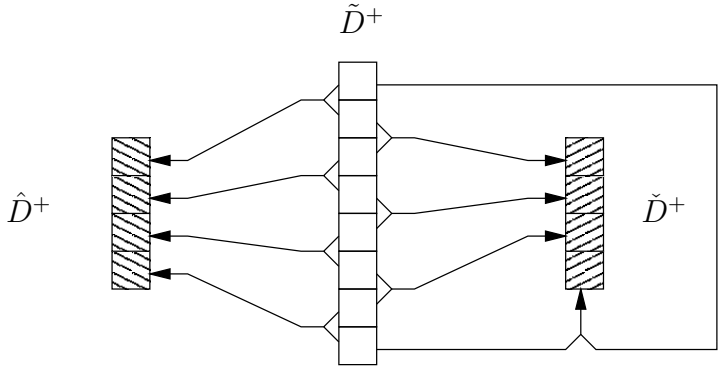


Abbildung 3.11: Datenfluß bei Umrechnung $\tilde{D}^+$ nach $[\hat{D}^+, \check{D}^+]$.

2. Über die Eigenwerte der Kopplung $\tilde{Z}^+ \rightarrow [\hat{Z}^+, \check{Z}^+]$

Für die Untersuchung der Eigenwerte betrachten wir zur Orientierung zunächst die Situation in exakter Arithmetik. Für den $(j+n)$-ten Eigenwert von $\tilde{L}^+\tilde{D}^+(\tilde{L}^+)^T$ gilt $\tilde{\lambda}^+ := \tilde{\lambda}^+_{j+n} = \sigma_j - \bar{\mu}$. Die zugehörigen Eigenwerte $\hat{\lambda}^+ := \hat{\lambda}^+_j$ bzw. $\check{\lambda}^+ := \check{\lambda}^+_j$ von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ bzw. $\check{L}^+\check{D}^+(\check{L}^+)^T$ können wir dann bestimmen durch

$$\check{\lambda}^+ = \hat{\lambda}^+ = \sigma_j^2 - \bar{\mu}^2 = \tilde{\lambda}^+(\sigma_j + \bar{\mu}) = \tilde{\lambda}^+(2\bar{\mu} + \tilde{\lambda}^+)$$

Die Auswirkungen der Gleitpunktarithmetik diskutieren wir im folgenden Satz. Dabei kombinieren wir wieder hintereinandergeschaltete Stabilitätsanalysen mit den Eigenschaften einer partiellen RRR.

Satz 3.21 (Über die Eigenwerte der Kopplungen)
Die Faktorisierung $\tilde{L}^+\tilde{D}^+(\tilde{L}^+)^T$ bilde eine partielle RRR für die interessierenden Eigenwerte. Dann bilden auch $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $\check{L}^+\check{D}^+(\check{L}^+)^T$ jeweils eine partielle RRR.
Ferner gilt für die Eigenwerte

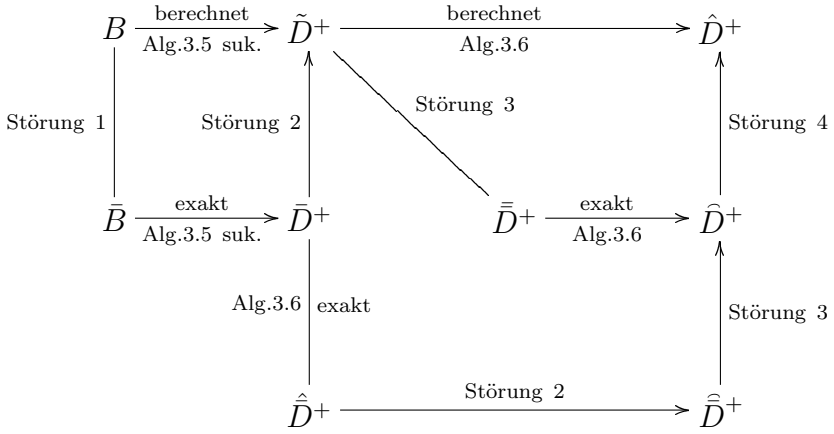
$$\hat{\lambda}^+ = \bar{\lambda}^+(2\bar{\mu} + \bar{\lambda}^+)(1 + \hat{\kappa}\epsilon) \qquad \text{und} \qquad \check{\lambda}^+ = \bar{\lambda}^+(2\bar{\mu} + \bar{\lambda}^+)(1 + \check{\kappa}\epsilon)$$

und damit

$$\left| \frac{\hat{\lambda}^+ - \check{\lambda}^+}{\hat{\lambda}^+} \right| = \epsilon \left| \frac{\hat{\kappa} - \check{\kappa}}{1 + \hat{\kappa}\epsilon} \right| = \mathcal{O}(\epsilon)$$

Beweis: Wir analysieren das nachfolgende Diagramm für die Umrechnung $B \rightarrow \tilde{D}^+ \rightarrow \hat{D}^+$. Die Variante $B \rightarrow \tilde{D}^+ \rightarrow \check{D}^+$ wir analog durchgeführt. Wir

notieren nur die jeweiligen diagonalen Pivotelemente repräsentativ für die Daten der entsprechenden LDL^T-Faktorisierung sowie des interessierenden Eigenwertes. Eine wichtige Beobachtung ist, daß einige Übergänge durch identische Störungen beschrieben sind.



Wir betrachten zunächst die gemischte Stabilitätsanalyse zur Faktorisierung

$$T_{GK} - \bar{\mu} I = \tilde{L}^+ \tilde{D}^+ (\tilde{L}^+)^T \qquad \text{also} \qquad B \xrightarrow[\text{Alg.3.5 suk.}]{\text{berechnet}} \tilde{D}^+$$

Dabei ist zu beachten, daß i.d.R. mehrere Zerlegungen sukzessiv mit den Shift-Parametern $\mu^{(1)}$ bis $\mu^{(r)}$ auszuführen sind. Wegen Bemerkung 3.16 können wir die Rundungsfehleranalyse aber zusammenfassend betrachten: Die Eingangsmatrix B geht durch kleine multiplikative Störungen über in eine Matrix $\bar{B}$. Dabei ändert sich der interessierende j-te Singulärwert nur geringfügig:

$$B \xrightarrow{\text{Störung 1}} \bar{B} \qquad \Rightarrow \qquad \bar{\sigma} = \sigma(1 + \kappa_1 \epsilon)$$

(Mit Lemma 1.27 lassen sich für κ_1 Fehlerschranken ermitteln. Nach Bemerkung 1.28 verzichten wir aber ansonsten wieder auf die genauere Diskussion solcher Schranken.) Der Eigenwert der durch den exakten Übergang

$$\bar{B} \xrightarrow[\text{Alg.3.5 suk.}]{\text{exakt}} \bar{D}^+$$

bestimmten Matrix $\bar{L}^+ \bar{D}^+ (\bar{L}^+)^T$ ist durch $\bar{\lambda}^+ = \bar{\sigma} - \bar{\mu}$ gegeben. Ausgehend von den Daten $\bar{D}^+$ (und $\bar{L}^+$) werden nun zwei Wege beschritten. Zum einen haben wir vorausgesetzt, daß $\tilde{L}^+ \tilde{D}^+ (\tilde{L}^+)^T$ eine partielle RRR bildet. Daher ändern sich die Eigenwerte beim folgenden Übergang nur geringfügig:

$$\bar{D}^+ \xrightarrow{\text{Störung 2}} \tilde{D}^+ \qquad \Rightarrow \qquad \tilde{\lambda}^+ = \bar{\lambda}^+ (1 + \kappa_2 \epsilon)$$

Zum anderen wenden wir Algorithmus 3.6 auf $\bar{D}^+$ in exakter Arithmetik an

$$\bar{D}^+ \xrightarrow[\text{Alg.3.6}]{\text{exakt}} \hat{\bar{D}}^+ \qquad \Rightarrow \qquad \hat{\bar{\lambda}}^+ = \bar{\lambda}^+(2\bar{\mu} + \bar{\lambda}^+)$$

Wir erhalten somit eine Darstellung für den Eigenwert $\hat{\bar{\lambda}}^+$ von $\hat{\bar{L}}^+\hat{\bar{D}}^+(\hat{\bar{L}}^+)^T$. Die Kernidee dieses Beweises ist, daß wir nun den Übergang

$$\bar{D}^+ \xrightarrow{\text{Störung 2}} \tilde{D}^+$$

simultan auch auf den Daten von $\hat{\bar{D}}^+$ durchführen können. Für $i = 1 : n$ gilt

$$\bar{d}^+_{2i-1} = \tilde{d}^+_{2i-1}(1 + \eta_{2i-1}\epsilon) \quad \text{und} \quad \bar{d}^+_{2i} = \tilde{d}^+_{2i}(1 + \eta_{2i}\epsilon)$$

und wir können wegen $\hat{\bar{d}}^+_i = -\bar{d}^+_{2i-1}\bar{d}^+_{2i}$ mit

$$\hat{\tilde{d}}^+_i := \hat{\bar{d}}^+_i(1 + \eta_{2i-1}\epsilon)(1 + \eta_{2i}\epsilon)$$

den Übergang

$$\hat{\bar{D}}^+ \xrightarrow{\text{Störung 2}} \hat{\tilde{D}}^+ \qquad \Rightarrow \qquad \hat{\tilde{\lambda}}^+ = \hat{\bar{\lambda}}^+(1 + \hat{\kappa}_3\epsilon)$$

definieren. Auch bei diesem ändern sich die interessierenden Eigenwerte nur marginal.

Wir untersuchen jetzt die Anwendung von Algorithmus 3.6 in Gleitpunktarithmetik, also den Übergang

$$\tilde{D}^+ \xrightarrow[\text{Alg.3.6}]{\text{berechnet}} \hat{D}^+$$

mit einer gemischten Stabilitätsanalyse. Im ersten Schritt stören wir die Daten von $\tilde{D}^+$ und $\hat{\tilde{D}}^+$ wieder simultan:

$$\tilde{D}^+ \xrightarrow{\text{Störung 3}} \bar{\tilde{D}}^+ \qquad \Rightarrow \qquad \hat{\tilde{D}}^+ \xrightarrow{\text{Störung 3}} \hat{D}^+$$

Auch diese Übergänge beeinträchtigen wieder nur die relative Genauigkeit der entsprechenden Eigenwerte:

$$\tilde{\lambda}^+ = \bar{\tilde{\lambda}}^+(1 + \kappa_4\epsilon) \qquad \text{und} \qquad \hat{\lambda}^+ = \hat{\tilde{\lambda}}^+(1 + \hat{\kappa}_5\epsilon)$$

Das exakte Anwenden von Algorithmus 3.6 auf $\bar{\tilde{D}}^+$ ergibt dann $\hat{D}^+$. Der abschließende Übergang

$$\hat{D}^+ \xrightarrow{\text{Störung 4}} \hat{D}^+ \qquad \Rightarrow \qquad \hat{\lambda}^+ = \hat{\lambda}^+(1 + \hat{\kappa}_6\epsilon)$$

vervollständigt das Diagramm.

Mit

$$\hat{\kappa} := \frac{(1+\hat{\kappa}_3\epsilon)(1+\hat{\kappa}_5\epsilon)(1+\hat{\kappa}_6\epsilon)-1}{\epsilon}$$

folgt die Behauptung für $\hat{\lambda}^+$. Damit bildet $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$eine partielle RRR.

Analog finden wir eine Darstellung für den Eigenwert $\check{\lambda}^+$ von $\check{L}^+\check{D}^+(\check{L}^+)^T$ und zeigen, daß auch diese Darstellung eine partielle RRR ist. □

Bemerkung 3.22
Die Faktorisierungen aus Algorithmus 3.5 werden generell mit einer gemischten Stabilitätsanalyse behandelt. Da die Umrechnungen mit Algorithmus 3.6 sowohl eine Vorwärts- als auch eine Rückwärtsanalyse erlauben, können wir im Beweis entweder auf Störung 3 oder auf Störung 4 verzichten. Insgesamt basiert die Aussage über die Eigenwerte jedoch auf einer gemischten Fehleranalyse.

3. Konsequenzen für den RRR-Algorithmus

Unter der Voraussetzung, daß die Faktorisierung von $T_{GK} - \bar{\mu}I$ eine RRR bildet, können wir wieder kleine *relative* Unterschiede zwischen den Eigenwerten der durch Kopplungen ermittelten Faktorisierungen $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $\check{L}^+\check{D}^+(\check{L}^+)^T$ erwarten. Führen wir die (sukzessiven) Zerlegungen von $B^TB - \bar{\mu}^2I$ und $BB^T - \bar{\mu}^2I$ explizit, also unabhängig voneinander aus, so müssen wir –wie in Satz 3.1 bereits diskutiert– mit schlechteren Ergebnissen rechnen.
Demgegenüber steht die Beobachtung, daß es für $\tilde{L}^+\tilde{D}^+(\tilde{L}^+)^T$ *a priori* oft viel schwieriger zu entscheiden ist, ob eine RRR vorliegt als für die entsprechenden Darstellungen $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ oder $\check{L}^+\check{D}^+(\check{L}^+)^T$, vgl. Beispiel 2.9 und die Fallstudie aus Abschnitt 3.1.2. Daher ist noch zu untersuchen, wie wir Umrechnungen der Art $[\hat{D}^+, \hat{L}^+] \rightarrow [\check{D}^+, \check{L}^+]$ auf höheren Ebenen im Darstellungsbaum durchführen können.

4. Umrechnung von $\hat{Z}^+$ über $\tilde{Z}^+$ nach $\check{Z}^+$

Wir betrachten nun den Fall, daß die Daten von $\hat{Z}^+$ gegeben sind und daraus die Daten von $\tilde{Z}^+$ und $\check{Z}^+$ ermittelt werden sollen. Bei Kopplungen auf der ersten Ebene im Darstellungsbaum können wir zu diesem Zweck Zusammenhänge zwischen den Hilfsvariablen $\hat{S}$, $\hat{P}$, $\check{S}$ sowie $\check{P}$ und den diagonalen Pivotelementen $\tilde{D}^+$ und $\tilde{R}^+$ ausnutzen. Vergleichbare Beziehungen wie in Lemma 3.6 oder Folgerung 3.7 sind bei Kopplungen auf höheren Ebenen

nicht mehr gegeben. In Algorithmus 3.7 wird deshalb ein Verfahren zur Umrechnung der LDL^T-Faktorisierungen vorgeschlagen, bei dem die geraden Pivotelemente von $\tilde{D}^+$ aus den Daten von $\hat{D}^+$ ermittelt werden (Zeile 3). Die nachfolgenden ungeraden Elemente werden dann mit Hilfe eines einzelnen Schrittes der **dstqds**-Transformation bestimmt (Zeilen 6 bis 8). Siehe auch Abbildung 3.12. Bei der URU^T-Faktorisierung ist die Situation ähnlich.

Algorithmus 3.7 Die Umrechnung $[\hat{D}^+] \to [\tilde{D}^+_{\text{gerade}}] \to [\tilde{D}^+_{\text{ungerade}}] \to [\check{D}^+]$.

Gegeben: $[\hat{D}^+, \tilde{D}, \tilde{L}, B, \mu]$
Gesucht: $[\check{D}^+, \check{L}^+, \tilde{D}^+, \tilde{L}^+]$

1: $\tilde{s}_1 = -\mu$; $\tilde{d}_1^+ = \tilde{d}_1 + \tilde{s}_1$; $\tilde{l}_1^+ = \frac{a_1}{\tilde{d}_1^+}$
2: **for** $i = 1 : n-1$ **do**
3: $\quad \tilde{d}_{2i}^+ = -\frac{\hat{d}_i^+}{\tilde{d}_{2i-1}^+}$
4: $\quad \tilde{s}_{2i} = \tilde{l}_{2i-1}^+ \tilde{l}_{2i-1} \tilde{s}_{2i-1} - \mu \quad \{= \tilde{d}_{2i}^+ - \tilde{d}_{2i}\}$
5: $\quad \tilde{l}_{2i}^+ = \frac{b_i}{\tilde{d}_{2i}^+}$
6: $\quad \tilde{s}_{2i+1} = \tilde{l}_{2i}^+ \tilde{l}_{2i} \tilde{s}_{2i} - \mu$
7: $\quad \tilde{d}_{2i+1}^+ = d_{2i+1} + \tilde{s}_{2i+1}$
8: $\quad \tilde{l}_{2i+1}^+ = \frac{a_i}{\tilde{d}_{2i+1}^+}$
9: $\quad \check{d}_i^+ = -\tilde{d}_{2i}^+ \tilde{d}_{2i+1}^+$
10: $\quad \check{l}_i^+ = \frac{a_{i+1} b_i}{\check{d}_i^+}$
11: **end for**
12: $\tilde{d}_{2n}^+ = -\frac{\hat{d}_n^+}{\tilde{d}_{2n-1}^+}$
13: $\check{d}_n^+ = -\tilde{d}_{2n} \tilde{d}_1$

5. Über die Eigenwerte der Kopplung $[\hat{Z}^+] \to [\tilde{Z}^+] \to [\check{Z}^+]$

Neben die explizite Faktorisierung für die Daten $\hat{D}^+$ und $\hat{L}^+$ tritt bei diesem Kopplungsansatz eine weitere explizite Teilfaktorisierung für die ungeraden Elemente der auf der Golub-Kahan Matrix basierenden $L^+D^+(L^+)^T$-Zerlegungen. Bei dieser in Algorithmus 3.7 vorgeschlagenen Mischform können wir die Auswirkungen auf die entsprechenden Eigenwerte nicht mehr wie in den Sätzen 3.14 und 3.21 kontrollieren.
Wir stützen uns daher auf numerische Beobachtungen, indem wir die Näherung an den interessierenden Eigenwert $\hat{\lambda}^+$ von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$ mit der Approximation für den Eigenwert $\check{\lambda}^+$ der durch Algorithmus 3.7 generierten

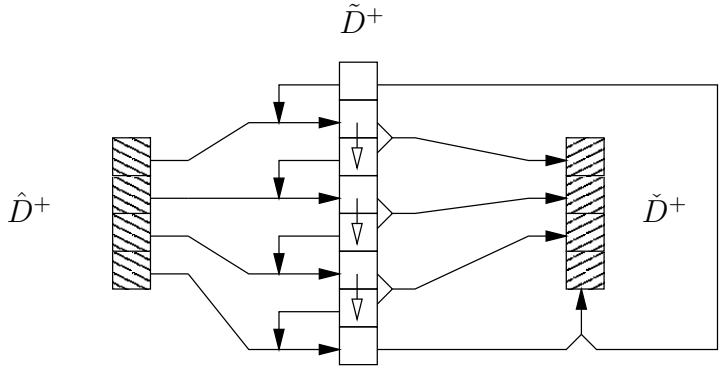


Abbildung 3.12: Datenfluß bei Umrechnung von $\hat{D}^+$ über $\tilde{D}^+$ nach $\check{D}^+$ (höhere Ebenen).

Matrix vergleichen. Dieses Kriterium läßt sich bei einer konkreten Implementierung *a posteriori* leicht überprüfen.
In Rechnerexperimenten finden wir viele Problemklassen, bei denen die Näherungen an $\hat{\lambda}^+$ und $\check{\lambda}^+$ nur sehr kleine *relative* Unterschiede aufweisen und die entsprechenden Singulärvektorpaare damit gut gekoppelt sind. Daneben gibt es aber auch Testmatrizen, für die $\hat{\lambda}^+$ und $\check{\lambda}^+$ stark voneinander abweichen. In diesen Fällen wirkt sich also die Teilfaktorisierung für die ungeraden Elemente zumindest auf die Eigenwerte negativ aus. Im folgenden Abschnitt geben wir Hinweise, wie dann fortzufahren ist (Stichwort Vergleichsvektoren). Wir verdeutlichen die Möglichkeiten zur Umrechnung der auf B^TB, BB^T und T_{GK} basierenden Faktorisierungen in Abbildung 3.13.

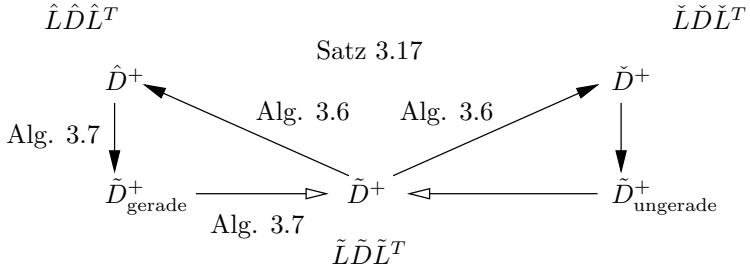


Abbildung 3.13: Kopplungen auf höheren Ebenen im Darstellungsbaum. Die ausgefüllten Pfeile bezeichnen rückwärts stabile Transformationen. Vergleiche mit Abbildung 3.9 auf Seite 81.

Mit den Kopplungsmechanismen für höhere Ebenen sind die theoretischen Vorarbeiten für die Anwendung des RRR-Algorithmus auf die **bSVD** ab-

geschlossen, vgl. Abbildung 3.14. Im nächsten Abschnitt skizzieren wir das Vorgehen für die Implementierung des Verfahrens.

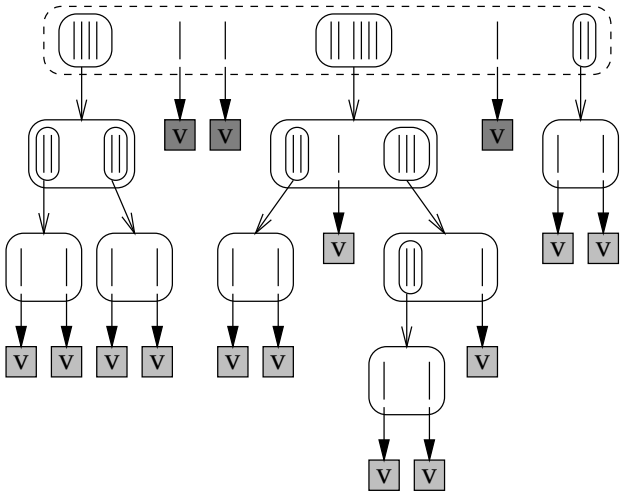


Abbildung 3.14: Anwendungsbereich von Kopplungen auf allen Ebenen.

3.4 Einbettung in den RRR-Algorithmus

In diesem Abschnitt formulieren wir den *Kernalgorithmus* zur Bestimmung der Singulärwertzerlegung einer bidiagonalen Matrix. Dieser basiert auf dem in [29] vorgeschlagenen Verfahren, in das wir die vorgestellten Ergebnisse über Kopplungen einbetten.
Unsere grundsätzliche Strategie besteht darin, den RRR-Algorithmus (vgl. Algorithmus 2.5) auf B^TB anzuwenden und die entsprechenden Faktorisierungen von BB^T und T_{GK} weitgehend implizit mit Hilfe der bereitgestellten Kopplungsmechanismen durchzuführen.
Nach einem Überblick in Abschnitt 3.4.1 erfolgt eine Beschreibung der wichtigsten Detailaufgaben. In Abschnitt 3.4.6 wird zusammenfassend diskutiert, inwieweit sich der Aufwand beim Übergang von dem **tSEP**- auf das **bSVD**-Lösungsverfahren erhöht.

3.4.1 Der Kernalgorithmus

Als ersten Baustein beschreiben wir in Algorithmus 3.8, wie in einer praktischen Implementierung Kopplungen auf höheren Ebenen im Darstellungsbaum realisiert werden können. Unter der Voraussetzung, daß die Dar-

Algorithmus 3.8 Kopplung der LDL^T-Faktorisierungen (höhere Stufen).

Gegeben: $[\hat{D}^{(\rho-1)}, \hat{L}^{(\rho-1)}, \tilde{D}^{(\rho-1)}, \tilde{L}^{(\rho-1)}]$ für $\rho > 1$ sowie die Shift-Parameter $\mu^{(\rho)}$ bzw. $\nu^{(\rho)}$.

Gesucht: $[\hat{D}^{(\rho)}, \hat{L}^{(\rho)}, \tilde{D}^{(\rho)}, \tilde{L}^{(\rho)}, \check{D}^{(\rho)}, \check{L}^{(\rho)}]$

1: Faktorisiere $\tilde{L}^{(\rho-1)}\tilde{D}^{(\rho-1)}(\tilde{L}^{(\rho-1)})^T - \mu^{(\rho)}I = \tilde{L}^{(\rho)}\tilde{D}^{(\rho)}(\tilde{L}^{(\rho)})^T$ mit Algorithmus 2.3.
2: Entscheide mit Hilfe von Satz 2.7, ob $\tilde{L}^{(\rho)}\tilde{D}^{(\rho)}(\tilde{L}^{(\rho)})^T$ *a priori* als RRR zu erkennen ist ($ok = 1$) oder nicht ($ok = 0$).
3: **if** $ok = 1$ **then**
4: Verwende Umrechnung aus Algorithmus 3.6.
5: **else**
6: Faktorisiere $\hat{L}^{(\rho-1)}\hat{D}^{(\rho-1)}(\hat{L}^{(\rho-1)})^T - \nu^{(\rho)}I = \hat{L}^{(\rho)}\hat{D}^{(\rho)}(\hat{L}^{(\rho)})^T$ mit Algorithmus 2.3.
7: Verwende Umrechnung aus Algorithmus 3.7.
8: **end if**

stellung $\tilde{L}^{(\rho)}\tilde{D}^{(\rho)}(\tilde{L}^{(\rho)})^T$ eine RRR bildet, können wir eine gute Qualität der Kopplungen voraussetzen, da die interessierenden Eigenwerte nur um kleine relative Abstände voneinander abweichen. In diesem Falle ist also das explizite Zerlegen einer Translation der Golub-Kahan Matrix vorzuziehen. Sonst faktorisieren wir auf der Basis von B^TB. Wir können die Qualität dieser Kopplungen nur *a posteriori* –also nach der Bestimmung der Eigenwertnäherungen– einschätzen. Gegenüber dem klassischen RRR-Algorithmus, bei dem nur eine einzelne Faktorisierung zu bestimmen ist, müssen wir in Algorithmus 3.8 mehrere Zerlegungen –teilweise auch von Matrizen der Dimension $2n$ statt n– durchführen, um gut gekoppelte Darstellungen zu erhalten. Die Auswirkungen dieser zusätzlich zu investierenden Arbeit auf das Gesamtverfahren untersuchen wir in Abschnitt 3.4.6 theoretisch und in Abschnitt 4.3 anhand numerischer Experimente.

Algorithmus 3.9 Reproduktion der LDL^T-Faktorisierungen.

Gegeben: B, path(B^TB), path(T_{GK})

Gesucht: $[\hat{D}^{(r-1)}, \hat{L}^{(r-1)}, \tilde{D}^{(r-1)}, \tilde{L}^{(r-1)}, \check{D}^{(r-1)}, \hat{L}^{(r-1)}]$

1: Faktorisiere $B^TB - \nu^{(1)}I = \hat{L}^{(1)}\hat{D}^{(1)}(\hat{L}^{(1)})^T$.
2: Zur Bestimmung von $\tilde{D}^{(1)}$, $\tilde{L}^{(1)}$, $\check{D}^{(1)}$ und $\check{L}^{(1)}$ verwende Umrechnungen aus Abschnitt 3.2.
3: **for** $\rho = 2 : r-1$ **do**
4: Bestimme die Zerlegungen der nächsten Ebene mit Algorithmus 3.8.
5: **end for**

Zu gegebenen äquivalenten Pfaden $\text{path}(B^TB) = (\nu^{(1)}, \ldots, \nu^{(r-1)})$ und $\text{path}(T_{GK}) = (\mu^{(1)}, \ldots, \mu^{(r-1)})$ können wir nun wie in Algorithmus 3.9 eine Folge von gekoppelten Faktorisierungen reproduzieren. Dabei gehen wir davon aus, daß die Pfade bereits so gewählt sind, daß jeweils Kopplungen guter Qualität entstehen.
Mit diesen Vorbereitungen können wir schließlich den Kernalgorithmus (siehe Seite 106) zur Bestimmung der **bSVD** beschreiben. Wir wählen dazu eine Darstellung wie bei dem klassischen RRR-Verfahren, vgl. Algorithmus 2.5 auf Seite 55.

3.4.2 Einige Implementierungsdetails

Wir gehen nun auf die wichtigsten Detailfragen einer praktischen Implementierung ein. Wir halten zunächst fest, daß Algorithmus 3.10 auch verwendet werden kann, um nur Teile des Spektrums zu bearbeiten (Adaptivität). Während der klassische RRR-Algorithmus (Algorithmus 2.5) dann k Näherungen an Eigenvektoren einer Matrix T erzeugt, erhalten wir mit k Approximationen an linke *und* rechte Singulärvektoren eine etwa doppelt so große Datenmenge an interessierenden Ergebnissen.

Näherungen an Singulärwerte

Die Singulärwertnäherungen $\bar{\sigma}_1 \leq \ldots \leq \bar{\sigma}_n$ erhalten wir entweder durch das QD-Verfahren (vgl. Abschnitt 1.5.3) oder mit Hilfe des Bisektionsverfahrens für Singulärwerte (vgl. Abschnitt 1.5.4). Ausschlaggebendes Kriterium ist die benötigte Rechenzeit. Hierbei ist zu beachten, daß existierende Implementierungen des QD-Verfahrens deutlich schneller laufen als die von Bisektionsverfahren. So ist es oft günstiger, das serielle QD-Verfahren zur Bestimmung von Approximationen aller Singulärwerte heranzuziehen, auch wenn $nprocs > 1$ Prozessoren eines Parallelrechners benutzt werden können oder nur Teile des Spektrums bestimmt werden sollen. Für welche Parameterkombinationen von k und $nprocs$ das Bisektionsverfahren das QD-Verfahren übertrifft, kann durch vorherige Kalibrierungsmessungen auf dem jeweils vorliegenden Rechner ermittelt werden.

Über die Liste $\mathcal{L}$

Algorithmus 3.10 ist mit Hinblick auf die später beschriebene Parallelisierung allgemein mit Hilfe einer Liste $\mathcal{L}$ formuliert. Jedes Element dieser Liste beschreibt einen bestimmten Eigenwertcluster im Darstellungsbaum, enthält also Informationen über

Algorithmus 3.10 Der Kernalgorithmus.

Gegeben: Eine bidiagonale Matrix B sowie deren mit hoher relativer Genauigkeit bestimmte Singulärwertnäherungen $\bar{\sigma}_1 \leq \ldots \leq \bar{\sigma}_n$.

Gesucht: Matrizen $\bar{U}$ und $\bar{V}$ mit Näherungen an die Basen aus linken und rechten Singulärvektoren.
Pfade $\text{path}(B^TB, j) := (\nu_j^{(1)}, \ldots, \nu_j^{(\text{rec}_j)})$, die für $j = 1 : n$ die Konstruktion von $\bar{U}(:, j)$ und $\bar{V}(:, j)$ beschreiben.

1: Beschreibe die Cluster-Struktur des Spektrums in einer Liste $\mathcal{L}$.
2: Bearbeite isolierte Singulärwerte unter Verwendung von Algorithmus 3.4.
3: **while** $\mathcal{L} \neq \emptyset$ **do**
4: Wähle einen Cluster aus $\mathcal{L}$.
5: {Sei r die Ebene im Darstellungsbaum.}
Reproduziere mit Hilfe von Algorithmus 3.9 die Matrizen $\hat{L}^{(r-1)}\hat{D}^{(r-1)}(\hat{L}^{(r-1)})^T$ und $\tilde{L}^{(r-1)}\tilde{D}^{(r-1)}(\tilde{L}^{(r-1)})^T$. Die (bereits bekannten) Eigenwertnäherungen $\bar{\bar{\lambda}}_f^{(r)} \leq \ldots \leq \bar{\bar{\lambda}}_l^{(r)}$ seien bzgl. der Matrix $\hat{L}^{(r-1)}\hat{D}^{(r-1)}(\hat{L}^{(r-1)})^T$ geclustert.
6: Wähle einen geeigneten Shift $\nu^{(r)}$.
7: Verwende Algorithmus 3.8 zur Bestimmung der Daten $[\hat{D}^{(r)}, \hat{L}^{(r)}, \tilde{D}^{(r)}, \tilde{L}^{(r)}, \check{D}^{(r)}, \hat{L}^{(r)}]$.
8: Bestimme Näherungen an die Eigenwerte $\bar{\bar{\hat{\lambda}}}_f^{(r+1)} \leq \ldots \leq \bar{\bar{\hat{\lambda}}}_l^{(r+1)}$ von $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$.
9: Bestimme Näherungen an die Eigenwerte $\bar{\bar{\check{\lambda}}}_f^{(r+1)} \leq \ldots \leq \bar{\bar{\check{\lambda}}}_l^{(r+1)}$ von $\check{L}^{(r)}\check{D}^{(r)}(\check{L}^{(r)})^T$.
10: Bestimme Singulärvektorapproximationen $\bar{U}(:, f : l)$ und $\bar{V}(:, f : l)$ durch *separates Zerlegen* in BABE-Faktorisierungen und Lösen der entsprechenden Gleichungssysteme. Prüfe dabei *a posteriori* die Fragen

1. Bildet $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$ eine partielle RRR?
2. Welche Qualität haben die Kopplungen?

{Aktualisiere $\text{path}(B^TB, j)$ für $j = f : l$ sowie die Liste $\mathcal{L}$.}
11: **if** beide Fragen positiv beantwortet werden können **then**
12: Behandele $\text{path}(B^TB, j)$ für $j = f : l$ sowie die Liste $\mathcal{L}$ analog zu den Zeilen 10 bis 12 aus Algorithmus 2.5.
13: **else**
14: Behandele $\text{path}(B^TB, j)$ für $j = f : l$ sowie die Liste $\mathcal{L}$ analog zur Zeile 14 aus Algorithmus 2.5.
15: **end if**
16: **end while**

- die aktuelle Ebene r im Darstellungsbaum
- die Lage und Größe durch Angabe der ersten (f) und letzten (l) Indizes
- und damit den Zugriff auf die bisher verwendeten Shift-Parameter $\mathrm{path}(B^TB, f:l)$
- den Zugriff auf die bereits früher berechneten interessierenden Eigenwertnäherungen für $\hat{L}^{(r-1)}\hat{D}^{(r-1)}(\hat{L}^{(r-1)})^T$

In den Zeilen 4 bzw. 11 bis 15 steuern wir durch Auswahl bzw. Aufdatierung der Listenelemente von $\mathcal{L}$ die Reihenfolge, in der der Darstellungsbaum abgearbeitet wird. Für serielle Implementierungen ist die *Depth-First-Strategie* (Tiefensuche) naheliegend. Die Liste $\mathcal{L}$ wird dann wie ein Stack behandelt. Der Vorteil dieser Strategie ist, daß die in Zeile 5 angeforderten Daten bereits im Arbeitsspeicher vorliegen und nicht reproduziert werden müssen.
Als Alternative für parallele Varianten bietet sich eine *Breadth-First-Strategie* (Breitensuche) an, bei der $\mathcal{L}$ als Queue zu verwenden ist. Diese Strategie erlaubt eine gute Lastverteilung zwischen den beteiligten Prozessoren, vgl. Abschnitt 5.3.2. Sie wird mit zusätzlichem Aufwand an Rechenoperationen erkauft, da in manchen Fällen die Faktorisierungen mehrmals aufgestellt werden müssen. Wir nehmen hier als Ergebnis vorweg, daß der zusätzliche Rechenaufwand i.d.R. klein ist gegenüber anderen Teilaufgaben wie etwa der nachfolgenden Bestimmung der Eigenwertnäherungen, vgl. Abschnitt 3.4.6.

3.4.3 Bestimmung von Eigenwertnäherungen

Die Bestimmung von Eigenwertapproximationen auf höheren Ebenen im Darstellungsbaum ist die weitaus zeitaufwendigste Teilaufgabe sowohl des klassischen RRR-Algorithmus als auch der Variante mit eingebetteten Kopplungen, vgl. Zeile 8. Wir skizzieren hier eine Modifikation des Verfahrens nach [29], das sich auch bei der Implementierung in der LAPACK-Routine `DSTEGR` bewährt hat, vgl. Abschnitt 4.3.

Eigenwerte von $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$

Bekannt sind die Daten aus der Faktorisierung

$$\hat{L}^{(r-1)}\hat{D}^{(r-1)}(\hat{L}^{(r-1)})^T - \nu^{(r)}I = \hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$$

sowie Näherungen $\bar{\hat{\lambda}}_f^{(r)} \leq \ldots \leq \bar{\hat{\lambda}}_l^{(r)}$ an die interessierenden Eigenwerte von $\hat{L}^{(r-1)}\hat{D}^{(r-1)}(\hat{L}^{(r-1)})^T$. Wir suchen Näherungen an die Eigenwerte der

nächsten Ebene $\hat{\lambda}_f^{(r+1)} \leq \ldots \leq \hat{\lambda}_l^{(r+1)}$ von $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$. Dazu gehen wir in mehreren Teilschritten vor.

1. Faktorisiere für $j = f : l$

$$\hat{L}^{(r-1)}\hat{D}^{(r-1)}(\hat{L}^{(r-1)})^T - \bar{\hat{\lambda}}_j^{(r)} I = \hat{N}_{k_j}^{(r,j)}\hat{G}_{k_j}^{(r,j)}(\hat{N}_{k_j}^{(r,j)})^T$$

2. Löse für $j = f : l$

$$\hat{N}_{k_j}^{(r,j)}\hat{G}_{k_j}^{(r,j)}(\hat{N}_{k_j}^{(r,j)})^T \cdot \bar{V}(:,j) = \hat{\gamma}_{k_j}^{(r,j)} e_{k_j}$$

3. Bestimme für $j = f : l$ Anfangsnäherungen χ_j durch

$$\chi_j = \frac{\bar{V}(:,j)^T \hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T \bar{V}(:,j)}{\bar{V}(:,j)^T \bar{V}(:,j)}$$

4. Bestimme für $j = f : l$ wie beim Bisektionsverfahren Intervallgrenzen α_j und β_j, so daß

$$\alpha_j \leq \chi_j < \beta_j \qquad \text{und} \qquad n_{\alpha_j} \leq j < n_{\beta_j}$$

5. Vereinige überlappende Intervalle.

6. Führe das Bisektionsverfahren solange aus, bis es $l - f + 1$ disjunkte Intervalle gibt oder die Grenzen von Intervallen, die mehr als einen Eigenwert beinhalten, hinreichend nahe beeinander liegen.

7. Verwende entweder weiterhin das Bisektionsverfahren, bis alle Intervalle hinreichend klein sind oder zur Beschleunigung die nachfolgend beschriebene abgesicherte Rayleigh-Quotienten-Iteration.

Der auf der folgenden Seite aufgeführte Algorithmus 3.11 ist eine Variante der *Rayleigh-Quotienten-Iteration (RQI)*, die sich die Eigenschaften von BABE-Faktorisierungen zunutze macht, siehe [29]. Insbesondere ist die *RQI-Korrektur* vereinfacht darstellbar:

$$\delta_{RQI} = \frac{z^T(T - \tau I)z}{z^T z} = \frac{z^T r}{z^T z} = \frac{\gamma_k^+}{z^T z} \tag{3.9}$$

Um zusätzlich sicher zu stellen, daß das Verfahren gegen den gesuchten j-ten Eigenwert konvergiert, bestimmen wir wie beim Bisektionsverfahren in jeder Iteration die Größe $n_\tau = \#\{g_i^+ | g_i^+ < 0\}$, also die Anzahl der Eigenwerte, die kleiner als τ sind. Für die nach dem oben beschriebenen Verfahren ermittelten Startwerte konvergiert das Verfahren i.d.R. bereits nach wenigen Schritten.

Algorithmus 3.11 Abgesicherte Rayleigh-Quotienten-Iteration (RQI).

Gegeben: Eine faktorisierte symmetrisch tridiagonale Matrix: LDL^T
Ein Intervall $[\alpha, \beta[$, das nur deren j-ten Eigenwert enthält.
Eine Näherung τ aus dem Intervall.
Eine obere Grenze maxit für die Iterationsschritte.
Eine Toleranzgrenze für die RQI-Korrektur, z.B. $tol = 4$.

Gesucht: Eine verbesserte Approximation τ an den j-ten Eigenwert.

1: iter $= 0$
2: $\delta_{RQI} = 0$
3: **repeat**
4: $\quad \tau = \tau + \delta_{RQI}$
5: $\quad$ Bestimme $LDL^T - \tau I = N_k^+ G_k^+ (N_k^+)^T$.
6: $\quad$ Bestimme dabei n_τ.
7: $\quad$ **if** $\neg(j \leq n_\tau \leq j+1)$ **then**
8: $\quad\quad$ Führe einige Schritte des Bisektionsverfahrens aus.
9: $\quad\quad$ Wähle τ aus dem aktualisierten Intervall $[\alpha, \beta[$.
10: $\quad\quad$ Gehe zu 2.
11: $\quad$ **else**
12: $\quad\quad$ Aktualisiere $[\alpha, \beta[$.
13: $\quad$ **end if**
14: $\quad$ Löse $N_k^+ G_k^+ (N_k^+)^T \cdot z = \gamma_k^+ e_k$.
15: $\quad \delta_{RQI} = \gamma_k^+ / z^T z$
16: $\quad$ iter $=$ iter $+ 1$
17: **until** $(|\delta_{RQI}|/|\tau| \leq tol \cdot \epsilon) \vee$ (iter $>$ maxit)

Eigenwerte von $\check{L}^{(r)} \check{D}^{(r)} (\check{L}^{(r)})^T$

Sind im Kernalgorithmus (Algorithmus 3.10) nach Erledigung von Zeile 8 Eigenwertnäherungen von $\hat{L}^{(r)} \hat{D}^{(r)} (\hat{L}^{(r)})^T$ bekannt, so erhalten wir die in Zeile 9 benötigten Eigenwertapproximationen an $\check{L}^{(r)} \check{D}^{(r)} (\check{L}^{(r)})^T$, indem wir für $j = f : l$ die abgesicherte Rayleigh-Quotienten-Iteration mit Startwert $\bar{\check{\lambda}}_j^{(r+1)}$ durchführen. Nach unseren Vorüberlegungen sollte der Unterschied zwischen $\bar{\check{\lambda}}_j^{(r+1)}$ und der zu ermittelnden Größe $\bar{\lambda}_j^{(r+1)}$ gering sein, so daß in vielen Fällen nur ein einziger Iterationsschritt nötig ist. Der Aufwand gegenüber dem klassischen RRR-Algorithmus erhöht sich hier also kaum.

Wie bereits eingangs angedeutet, stellt die Bestimmung von Eigenwertapproximationen in den Zeilen 8 und 9 die bei weitem aufwendigste Teilaufgabe des Kernalgorithmus (und des klassischen RRR-Algorithmus) dar. Sie hat also bei Testmatrizen mit vielen oder großen Singulärwertclustern maß-

geblichen Einfluß auf die Ausführungszeit. Um bei einer Parallelisierung die Arbeit möglichst gleichmäßig verteilen zu können, lohnt sich bei einer *Breadth-First-Strategie* der eventuell anfallende zusätzliche Rechenaufwand zur Reproduktion der aktuellen Faktorisierung (Zeile 5).
Es wird hier nochmals deutlich, daß wir bei der Auswahl eines Shift-Kandidaten $\nu^{(r)}$ in Zeile 6 möglichst gute *a priori* Kriterien benötigen, vgl. Abschnitt 2.4. Stellt sich der Shift-Kandidat *a posteriori* –also nach der aufwendigen Eigenwertbestimmung– als unzureichend heraus, so muß die in den Zeilen 8 und 9 geleistete Arbeit verworfen werden.

3.4.4 Bestimmung der Singulärvektoren

Wir beschreiben nun die Vorgehensweise bei den in Zeile 10 gestellten Teilaufgaben genauer. Wir können davon ausgehen, daß wir auf hohe relative Genauigkeit bestimmte Näherungen an die interessierenden Eigenwerte von $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$ und $\check{L}^{(r)}\check{D}^{(r)}(\check{L}^{(r)})^T$ kennen. (Ansonsten gehen wir im Kernalgorithmus zurück zu Zeile 6 und wählen einen anderen Shift-Kandidaten.) Zur Bestimmung von Approximationen an die Singulärvektoren faktorisieren wir für $j = f : l$ zunächst getrennt

$$\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T - \bar{\hat{\lambda}}_j^{(r+1)} I = \hat{N}_{\hat{k}}^{(r+1)}\hat{G}_{\hat{k}}^{(r+1)}(\hat{N}_{\hat{k}}^{(r+1)})^T \tag{3.10}$$

$$\check{L}^{(r)}\check{D}^{(r)}(\check{L}^{(r)})^T - \bar{\check{\lambda}}_j^{(r+1)} I = \check{N}_{\check{k}}^{(r+1)}\check{G}_{\check{k}}^{(r+1)}(\check{N}_{\check{k}}^{(r+1)})^T \tag{3.11}$$

und lösen anschließend die linearen Gleichungssysteme

$$\hat{N}_{\hat{k}}^{(r+1)}\hat{G}_{\hat{k}}^{(r+1)}(\hat{N}_{\hat{k}}^{(r+1)})^T \cdot \bar{V}(:,j) = \hat{\gamma}_{\hat{k}}^{(r+1)} e_{\hat{k}} \tag{3.12}$$

$$\check{N}_{\check{k}}^{(r+1)}\check{G}_{\check{k}}^{(r+1)}(\check{N}_{\check{k}}^{(r+1)})^T \cdot \bar{U}(:,j) = \check{\gamma}_{\check{k}}^{(r+1)} e_{\check{k}} \tag{3.13}$$

Bei numerischen Experimenten stellt sich heraus, daß die entsprechenden BABE-Faktorisierungen separat durch explizites Faktorisieren aufgestellt werden können, ohne das Residuum der Singulärvektoren zu beeinträchtigen. Auf höheren Ebenen im Darstellungsbaum verzichten wir also im letzten Schritt auf die Kopplungen.
Bei einer optimierten Implementierung ist die Bestimmung der Eigenwertnäherungen und der Singulärvektoren durch Anwendung der Rayleigh-Quotienten-Iteration enger verzahnt als hier dargestellt. So werden beispielsweise die in Zeile 4 von Algorithmus 3.11 gebildeten aktuellen Daten der BABE-Faktorisierung sowie die in Zeile 14 bestimmte Größe z als Näherung an den rechten Singulärvektor zwischengespeichert.

3.4.5 Die *a posteriori* Kriterien

Wir können nun klären, ob der in Zeile 6 ausgewählte Shift-Kandidat $\nu^{(r)}$ als geeignet eingestuft werden kann. Dazu sind folgende Fragen zu beurteilen:

1. Bildet $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$ eine partielle RRR?
2. Welche Qualität haben die Kopplungen?

1. Bildet $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$ eine partielle RRR?

Wir verwenden die in Abschnitt 2.4 eingeführten relativen Konditionszahlen und betrachten wie in Algorithmus 2.4 die Größe

$$\kappa_{max} = \max_{j=f:l} \left\{ \kappa_{\text{rel}} \left(\bar{\hat{\lambda}}_j^{(r+1)} \right) \right\}$$

Numerische Bestimmung von $\kappa_{\text{rel}}(\lambda^+)$

Ist λ^+ ein Eigenwert von LDL^T und q ein Vektor mit $LDL^Tq = \lambda^+ q$, so ist nach der Gleichung (2.17) die relative Konditionszahl gegeben durch

$$\kappa_{\text{rel}}\left(\lambda^+\right) = \left| \frac{q^T L|D|L^T q}{\lambda^+ q^T q} \right|$$

In der Gleitpunktarithmetik können wir nur eine Näherung $\bar{\lambda}^+$ an den Eigenwert bestimmen und $\bar{q}$ mit Hilfe einer BABE-Faktorisierung

$$N_k^+ G_k^+ (N_k^+)^T \cdot \bar{q} = \gamma_k^+ e_k \qquad \text{wobei} \qquad LDL^T - \bar{\lambda}^+ I = N_k^+ G_k^+ (N_k^+)^T$$

Wird die Größe $f = L^T\bar{q}$ naiv als Matrix-Vektor-Produkt bestimmt, so kann es bei großem Elementwachstum von D und damit auch von L bedingt durch Auslöschung zu Effekten kommen, die das Ergebnis für die Konditionszahl stark verfälschen können.

Das Problem

Wir nehmen an, daß LDL^T eine RRR für einen Eigenwert $\lambda^+ = \mathcal{O}(\epsilon^3)$ bildet, also eine relative Konditionszahl $\kappa_{\text{rel}}\left(\lambda^+\right) = \mathcal{O}(1)$ vorliegt. Damit ergibt sich für die in exakter Arithmetik berechnete Größe

$$\bar{q}^T L|D|L^T\bar{q} = f^T|D|f = \mathcal{O}(\epsilon^3)$$

Tritt jedoch in der Gleitpunktarithmetik bei der Matrix-Vektor-Multiplikation durch Auslöschung bedingt ein Fehler $\mathrm{fl}(f) = f + \mathcal{O}(\epsilon)$ auf, so ergibt sich

$$\mathrm{fl}(f^T|D|f) = \mathcal{O}(\epsilon^2)$$

Mit diesem Wert messen wir im wesentlichen nur noch den Fehler von $\mathrm{fl}(f)$. Das berechnete Ergebnis für die relative Konditionszahl hat eine Größenordnung von $\mathcal{O}(\epsilon^{-1})$. Basierend auf diesem berechneten Ergebnis würde dann das *a posteriori* Kriterium im RRR-Algorithmus negativ ausfallen, obwohl LDL^T eine RRR ist.

Eine Abhilfe

Wir schlagen aufgrund dieser Problematik ein numerisch stabileres Verfahren zur Bestimmung von $f = L^T\bar{q}$ vor. Es ist

$$\bar{q}_k = 1, \quad \bar{q}_i = -l_i^+\bar{q}_{i+1} \quad \text{für} \quad i < k, \quad \bar{q}_{i+1} = -u_i^+\bar{q}_i \quad \text{für} \quad i > k$$

Dann gilt für $i < k$

$$f_i = \bar{q}_i + l_i\bar{q}_{i+1} = \bar{q}_{i+1}(l_i - l_i^+) = \bar{q}_{i+1}l_i\frac{s_i}{d_i^+}$$

und für $k \leq i < n$

$$f_i = \bar{q}_i + l_i\bar{q}_{i+1} = \bar{q}_i(l_i - u_i^+) = \bar{q}_i\frac{p_{i+1}}{r_{i+1}^+}$$

Wir können also mit Hilfe der Daten der BABE-Faktorisierung die kritischen Additionen durch Multiplikationen und Divisionen ersetzen. Durch die hier neu vorgeschlagene Verwendung der Hilfsvariablen aus den QD-artigen Transformationen wird die korrekte Bestimmung der relativen Konditionszahlen auch für sehr kleine Eigenwerte ermöglicht.

Prüfung aller relativen Konditionszahlen

Mit den Ergebnissen der obigen Bemerkung können wir nun beurteilen, ob die Wahl des Shift-Kandidaten eine RRR für die interessierenden Eigenwerte von $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$ bildet oder nicht. Dazu werden wie beschrieben die entsprechenden relativen Konditionszahlen ermittelt. Wir müssen dazu $\bar{V}(:,j)$ für $j = f : l$ berechnen, auch wenn $\bar{\hat{\lambda}}_j^{(r+1)}$ wieder in einem Cluster liegt. Dennoch ist diese Arbeit nicht vergeblich, da sie dem ersten und zweiten Teilschritt bei der anstehenden Bestimmung neuer Eigenwertnäherungen auf der nächsthöheren Ebene im Darstellungsbaum entspricht, vgl. Abschnitt 3.4.3.

2. Welche Qualität haben die Kopplungen?

Während das erste Kriterium — Vorliegen einer RRR — für den klassischen RRR-Algorithmus hinreichend ist, müssen wir beim Kernalgorithmus für die **bSVD** zusätzlich noch das zweite Kriterium — die Qualität der Kopplungen — überprüfen. Die theoretischen Grundlagen sind in Abschnitt 3.3.3 dargestellt. Siehe Satz 3.21 und nachfolgende Bemerkungen.
Wir vergleichen also die Eigenwertnäherungen $\bar{\bar{\lambda}}_j^{(r+1)}$ mit $\bar{\bar{\lambda}}_j^{(r+1)}$ für $j = f : l$. Wenn diese nur einen kleinen relativen Abstand voneinander haben, so können wir die zugehörigen Singulärvektorpaare $\bar{U}(:, j)$ und $\bar{V}(:, j)$ als gut gekoppelt betrachten. (Die Größen $\bar{U}(:, j)$ und $\bar{V}(:, j)$ sind nach den Gleichungen (3.10), (3.11), (3.12) und (3.13) bekannt.)

Vergleichsvektoren

Weichen die Eigenwertnäherungen stärker voneinander ab, ist es trotzdem nicht ausgeschlossen, daß die Näherungen an die Singulärvektorpaare gut gekoppelt sind. Mit vertretbarem zusätzlichen Aufwand können wir die Approximation $\bar{\bar{\lambda}}_j^{(r+1)}$ anstelle von $\bar{\bar{\lambda}}_j^{(r+1)}$ zur Faktorisierung von $\check{L}^{(r)}\check{D}^{(r)}(\check{L}^{(r)})^T$ heranziehen. Im Unterschied zu Gleichung (3.11) bilden wir also

$$\check{L}^{(r)}\check{D}^{(r)}(\check{L}^{(r)})^T - \bar{\hat{\lambda}}_j^{(r+1)} I = \hat{\hat{N}}_{\hat{k}}^{(r+1)} \hat{\hat{G}}_{\hat{k}}^{(r+1)} (\hat{\hat{N}}_{\hat{k}}^{(r+1)})^T$$

Durch Lösen von

$$\hat{\hat{N}}_{\hat{k}}^{(r+1)} \hat{\hat{G}}_{\hat{k}}^{(r+1)} (\hat{\hat{N}}_{\hat{k}}^{(r+1)})^T \cdot \hat{\bar{U}}(:, j) = \hat{\hat{\gamma}}_{\hat{k}}^{(r+1)} e_{\hat{k}}$$

erhalten wir einen *Vergleichsvektor* $\hat{\bar{U}}(:, j)$ zu $\bar{U}(:, j)$. Während $\hat{\bar{U}}(:, j)$ ein Vektor ist, der gut an $\bar{V}(:, j)$ gekoppelt ist, garantiert $\bar{U}(:, j)$ numerische Orthogonalität.
Wir messen nun den Winkel zwischen beiden Vektoren, also

$$\cos(\theta) = \frac{\hat{\bar{U}}(:, j)^T \bar{U}(:, j)}{\left\|\hat{\bar{U}}(:, j)\right\| \left\|\bar{U}(:, j)\right\|}$$

Stellt sich heraus, daß dieser Winkel klein ist, so macht es numerisch keinen Unterschied, ob wir eine Näherung an den linken Singulärvektor mit dem Shift $\bar{\bar{\lambda}}_j^{(r+1)}$ oder $\bar{\bar{\lambda}}_j^{(r+1)}$ bestimmen. Insofern können wir $\bar{U}(:, j)$ und $\bar{V}(:, j)$ als gut gekoppelt betrachten, auch wenn $\bar{\bar{\lambda}}_j^{(r+1)}$ und $\bar{\bar{\lambda}}_j^{(r+1)}$ stärker voneinander abweichen. Es ist empfehlenswert, den Vergleichsvektor $\hat{\bar{U}}(:, j)$ zu überprüfen, bevor der in Zeile 6 gewählte Shift-Kandidat $\nu^{(r)}$ als ungeeignet eingestuft wird.

Kurze Aufwandsbetrachtung

Wir schließen die Diskussion der beiden Leitfragen

1. Bildet $\hat{L}^{(r)}\hat{D}^{(r)}(\hat{L}^{(r)})^T$ eine partielle RRR?
2. Welche Qualität haben die Kopplungen?

für die *a posteriori* Kriterien ab, indem wir den Aufwand analysieren, der bei der Einbettung der Kopplungen im Kernalgorithmus gegenüber dem klassischen RRR-Algorithmus entsteht. Das zweite Kriterium läßt sich weitgehend mit den Informationen überprüfen, die für das erste ohnehin benötigt werden. Sollte wie oben skizziert die Berechnung eines Vergleichsvektors $\hat{\tilde{U}}(:,j)$ erforderlich sein, so können wir dies mit vertretbarem zusätzlichen Arbeitsvolumen realisieren.

3.4.6 Abschließende Beurteilung

Als wichtiges Ergebnis ist zunächst zu betonen, daß die Einbettung der Kopplungen möglich ist, *ohne die grundlegende Struktur des Verfahrens zu ändern.* Beim Übergang vom klassischen RRR-Algorithmus (Algorithmus 2.5) zum Kernalgorithmus für die **bSVD** (Algorithmus 3.10) sind jeweils nur einzelne Teilaufgaben zu modifizieren.
Zur Aufwandsbetrachtung halten wir nochmals fest, daß der klassische RRR-Algorithmus bei Lösung des **tSEP**

$$T = Q\Lambda Q^T$$

mit Näherungen an Q ein Volumen von ca. n^2 interessierenden Daten erzeugt, während wir beim Kernalgorithmus zur Lösung von

$$B = U\Sigma V^T$$

mit Approximationen an die linken und rechten Singulärvektoren etwa die zweifache Menge an Ergebnissen erhalten. Im Vergleich zum **tSEP** müssen zusätzlich noch gute Kopplungen gewährleistet sein. Eine Verdopplung des Aufwands gegenüber dem klassischen Algorithmus ist also vertretbar.
Inwieweit sich der Arbeitsumfang für die einzelnen Teilaufgaben auf die Lösung des gesamten Problems auswirkt, hängt auch vom jeweiligen Darstellungsbaum ab. Die Anzahl und Größe der Cluster sowie deren Ebene r beeinflussen also den Vergleich zwischen dem klassischen und dem Kernalgorithmus.
Die Situation für isolierte Singulärwerte ist bereits in Abschnitt 3.2 beschrieben. Wir betrachten daher nur noch den Fall geclusterter Singulärwerte. Wir

bezeichnen dabei mit c die Anzahl aller zu bearbeitenden Cluster und mit e die Anzahl aller zu approximierenden Eigenwerte inklusive der Zwischenstufen.

Teilaufgabe	Abhängigkeit	Faktor
Zeile 5	c	ca. 2.5 bis 4.5
Zeile 6	c und e	0
Zeile 7	c	ca. 2.5 bis 4.5
Zeile 8	e	1
Zeile 9	e	< 1, u.U. $\ll 1$
Zeile 10		
Vektoren	e	2
Kriterien	e	1 oder > 1

Tabelle 3.3: Zusätzlicher Aufwand des Kernalgorithmus gegenüber dem klassischen RRR-Algorithmus. Aufgelistet sind die Teilaufgaben und deren Abhängigkeit von den Parametern c und e. Die dritte Spalte gibt den Faktor an, um den sich die Kosten erhöhen.

Tabelle 3.3 faßt die Situation zusammen. Für die Ermittlung gut gekoppelter faktorisierter Darstellungen in den Zeilen 5 und 7 ist teilweise erheblicher Mehraufwand erforderlich als bei der Bestimmung einer einzelnen Faktorisierung im RRR-Algorithmus (vergleiche Algorithmus 3.8, Seite 104). Dagegen können wir bei den Lösungsverfahren für die Eigenwerte (Abschnitt 3.4.3) in Zeile 9 häufig auf Informationen zurückgreifen, die bereits bei Bearbeitung der Teilaufgabe aus Zeile 8 benötigt werden. Daher ist in manchen Fällen kaum nennenswerter Zusatzaufwand zu verbuchen. Ähnlich stellt sich die Lage bei der Überprüfung der *a posteriori* Kriterien (Zeile 10) dar. Hier entstehen nur (geringe) Mehrkosten, wenn Vergleichsvektoren (Abschnitt 3.4.5) aufgestellt werden müssen.
Die Erzeugung der gekoppelten Matrizen (Zeilen 5 und 7) ist i.d.R. mit geringerem Aufwand verbunden als die anschließende (iterative) Bestimmung von Eigenwertnäherungen (Zeilen 8 und 9). Pro Cluster benötigen wir jeweils eine faktorisierte Matrix, von der dann mehrere (mindestens zwei) interessierende Eigenwerte zu approximieren sind. Also fällt der Anteil der Faktorisierungsaufgaben umso geringer aus, je größer ein Cluster ist.
Praktische numerische Experimente in Abschnitt 4.3 zeigen, daß der Aufwand dominiert wird von den Zeilen 8 und 10, während die Rechenzeiten für die Zeilen 5, 6, 7 und 9 nur von marginaler Bedeutung sind. Insofern ist zu erwarten, daß sich der Arbeitsumfang bei Übertragung auf die **bSVD** nicht mehr als verdoppelt.

Die Kopplungsmechanismen für höhere Ebenen aus Abschnitt 3.3 zielen darauf ab, das zweite Kriterium aus Zeile 10 zu erfüllen, wenn auch das erste Kriterium gültig ist. Ist dies nicht der Fall, müssen wir im Kernalgorithmus in Zeile 6 einen neuen Shift-Kandidaten bestimmen und die nachfolgenden Schritte wiederholen, während der klassische RRR-Algorithmus zum nächsten Cluster übergehen kann. Dadurch kann sich die Aufwandsanalyse zu Ungunsten des Kernalgorithmus verschieben. Obwohl sich Testmatrizen konstruieren lassen, bei denen diese Problematik hervortritt, greifen die in diesem Kapitel vorgeschlagenen Strategien im Normalfall sehr gut, d.h. daß mit dem ersten auch das zweite Kriterium erfüllt ist.
In nächsten Kapitel beschreiben wir ein **bSVD**-Lösungsverfahren, dessen Herzstück mit Algorithmus 3.10 gegeben ist. Wir binden jedoch zusätzlich einige Optionen zur Verbesserung der Ausführungsgeschwindigkeit und der Genauigkeit ein, die auch die o.g. Problematik weiter abschwächen.

Kapitel 4

Ein bSVD-Lösungsverfahren

Durch Anwendung der im vorigen Kapitel eingeführten Kopplungen steht mit dem Kernalgorithmus (Algorithmus 3.10) eine Methode zur Bestimmung der **bSVD** bereit. Dabei bleiben die vorteilhaften Eigenschaften (Komplexität, Genauigkeit, Adaptivität, Parallelisierbarkeit, vgl. Tabelle 1.1) des klassischen RRR-Algorithmus zur Lösung des **tSEP** erhalten.
In diesem Kapitel zeigen wir weitere Möglichkeiten zur Verbesserung auf, bevor wir zur Diskussion der seriellen numerischen Ergebnisse übergehen. Das Vorgehen zur Parallelisierung beschreiben wir im nachfolgenden Kapitel.

4.1 Vorverarbeitung

Der Kernalgorithmus ist ein sehr gut geeignetes Werkzeug zur effizienten **SVD** einer bidiagonalen Matrix. Dennoch weist er einige kritische Punkte auf, die bei gezielter Konstruktion von Testmatrizen offenkundig werden:

- Breakdowns bei Faktorisierungen.
- Zu hohe relative Konditionszahlen für alle Shift-Kandidaten.
- Keine hinreichende Qualität der Kopplungen.
- Sehr enge Cluster, die sich durch Shift-Strategien nicht auflösen lassen. Der Darstellungsbaum würde dann unbegrenzt wachsen. Wir identifizieren das Problem dadurch, daß eine gewisse Schranke recmax für die Anzahl der Ebenen überschritten wird.

Einige der Probleme treten schon beim klassischen RRR-Algorithmus auf, während schlechte Kopplungen erst beim Kernalgorithmus ins Gewicht fallen können. Zur Absicherung des Lösungsverfahrens ist es daher empfehlenswert, die Eingangsmatrix B vor der Anwendung von Algorithmus 3.10 geeignet zu präparieren.

4.1.1 Flipping und Splitting

Für Bidiagonalmatrizen, bei denen starke Unterschiede in den Größenordnungen der Einträge auftreten, ist es nach [21, 44] günstiger, Zerlegungen bei den betragsgrößten Elementen zu beginnen. Da der Kernalgorithmus oft LDL^T-Faktorisierungen verwendet, sollte für $|a_1| \ll |a_n|$ anstelle von B mit $B^{(\mathrm{rev})}$ gearbeitet werden, vgl. Bemerkungen 3.4 und 3.10. Wenden wir den Kernalgorithmus auf $B^{(\mathrm{rev})}$ an, so erhalten wir U durch den klassischen RRR-Algorithmus, während die Daten von V durch Kopplungsmechanismen bestimmt werden. Das sogenannte *Flipping* von B kann sich nachteilig auf die Ausführungsgeschwindigkeit auswirken, da die Zeilen von U und V permutiert (umkopiert) werden müssen.
Obwohl die Eingangsmatrix B unreduziert ist, kann es vorkommen, daß einige ihrer Einträge numerisch als vernachlässigbar klein angesehen werden können. In [21] und der danach implementierten LAPACK-Routine `DBDSQR` sind Kriterien angegeben, nach denen diese Einträge auf Null gesetzt werden können, ohne die relative Genauigkeit der Singulärwerte zu beeinträchtigen. Ist die so modifizierte n-dimensionale Matrix B aber nicht mehr unreduziert, so kann sie in kleinere Probleme zerlegt werden (*Splitting*):

$$B = \mathtt{diag}(B^{(1)}, \ \ldots \ , B^{(K)}), \quad \text{löse} \quad B^{(k)} = U^{(k)}\Sigma^{(k)}(V^{(k)})^T \quad \text{für } k = 1 : K$$

Verzichten wir auf eine hohe relative Genauigkeit der Näherungen an die *Singulärwerte* und sind lediglich an der Erfüllung der in Abschnitt 1.3.5 angegebenen Bedingungen für die *Singulärvektoren* interessiert, so können wir weitergehende Splitting-Kriterien anwenden, vgl. Abschnitt 4.3.3.
Unter gewissen Bedingungen kann die Ausgangsmatrix also in mehrere Teilprobleme zerfallen. Für kleinere Matrixdimensionen (z.B. ≤ 20) empfiehlt sich trotz der quadratischen Komplexität des Kernalgorithmus die Verwendung des QR-Verfahrens mit kubischem Aufwand.

4.1.2 Präkonditionierung

Die Gefahr von Breakdowns bei den Faktorisierungen im Kernalgorithmus läßt sich durch geeignete *Präkonditionierung* von B reduzieren.

Beschreibung eines Präkonditionierungsschrittes

Wir führen eine gewisse Anzahl von einzelnen Schritten des QR-Verfahrens durch, vgl. Abschnitt 1.5.2, Gleichung (1.1). Im ersten Schritt geht $B = B_0$ also über in eine Bidiagonalmatrix

$$B_1 = H_0^T B_0 G_0$$

Dabei repräsentieren H_0 und G_0 die Akkumulation von je $n-1$ Givensrotationen. Kennen wir die **bSVD** $B_1 = U'\Sigma V'^T$, so erhalten wir die gesuchten Größen $U = G_0 U'$ und $V = H_0 V'$ mit einem Aufwand von $\mathcal{O}(n^2)$. Beim Aufdatieren der Spalten

$$U(:,j) = G_0 U'(:,j) \qquad V(:,j) = H_0 V'(:,j) \tag{4.1}$$

ist wieder Adaptivität gegeben: Wir können k Spalten mit $\mathcal{O}(kn)$ Operationen aktualisieren.

Ziel der Präkonditionierung

Durch das Vorschalten einiger Schritte des QR-Verfahrens lösen wir also ein äquivalentes System, dessen bestimmende Matrix vom Kernalgorithmus oft besser verarbeitet werden kann als die Originalmatrix B. Da diese Art der Präkonditionierung Teil eines Diagonalisierungsprozesses ist, verschieben sich typischerweise betragsgroße Nebendiagonalelemente auf die Diagonale, während betragskleine Nebendiagonalelemente weiter verringert werden. Dadurch erhöht sich die Wahrscheinlichkeit, die Matrix in kleinere Probleme zerlegen zu können (Splitting). Durch die zunehmende Diagonaldominanz sind auch die LDL^T-Faktorisierungen im Kernalgorithmus weniger anfällig für Breakdowns.
Der Einfluß der Vorverarbeitungsphase ist also generell positiv zu bewerten. Sie ist jedoch im Vergleich zum Kernalgorithmus mit nicht unerheblichem zusätzlichen Aufwand verbunden: Sowohl der Kernalgorithmus als auch das Aufdatieren der Singulärvektoren haben quadratische Komplexität. Bei der Implementierung wird daher die nachfolgend beschriebene Strategie gewählt, die in Abhängigkeit vom zu bearbeitenden Testproblem durch Einstellung der entsprechenden Parameter gesteuert werden kann.

Statische und dynamische Präkonditionierung

Bei der *statischen* Präkonditionierung führen wir p_s Schritte des QR-Verfahrens aus und datieren anschließend alle Singulärvektoren auf.

Die *dynamische* Präkonditionierung fußt auf der Beobachtung, daß unter den n zu bestimmenden Singulärvektorpaaren oft nur sehr wenige wirklich problematisch sind. Nur für diese werden dann bis zu p_d weitere Vorverarbeitungsschritte durchgeführt. Nach jeder Präkonditionierung wird der Kernalgorithmus aufgerufen, um die noch verbliebenen Vektoren zu ermitteln. Gelingt dies, nutzen wir die Möglichkeit zur adaptiven Aufdatierung, siehe Gleichung (4.1). Sonst bearbeiten wir die Bidiagonalmatrix mit einer neuen QR-Transformation.

4.1.3 Der Gesamtalgorithmus

Wir konkretisieren die bisher beschriebenen Möglichkeiten zur Vorverarbeitung in Algorithmus 4.1. Dieser stellt das Gerüst für die in den numerischen Experimenten verwendete Implementierung dar.
Wir führen einen Vektor err ein, der für $j = 1 : n$ angibt, ob der Kernalgorithmus das entsprechende Singulärvektorpaar erfolgreich bestimmen kann ($\text{err}_j = 0$) oder nicht ($\text{err}_j = 1$). Für die maximale Anzahl der Präkonditionierungsschritte erweist sich aufgrund numerischer Experimente eine Voreinstellung $p_s = 1$ und $p_d = 10$ als günstig. Der Speicherbedarf einer einzelnen akkumulierten Givenstransformation beträgt $2n$ Gleitkommazahlen. Im Gesamtalgorithmus stellt dies keine größere Ressourcenbelastung dar. Aus Gründen der Übersichtlichkeit haben wir in Algorithmus 4.1 auf die Option verzichtet, für kleine Dimensionen $\mathcal{N} \leq 20$ anstelle des Kernalgorithmus das Standard-QR-Verfahren zu verwenden.
Durch Modifikation der Startmenge J_{p_s} kann der Gesamtalgorithmus auch zur adaptiven Bestimmung von Teilen des Spektrums verwendet werden. Typischerweise können beim ersten Aufruf des Kernalgorithmus in Zeile 8 die meisten der in J_{p_s} spezifizierten Singulärvektorpaare bestimmt werden. Sollte für ein Singulärvektorpaar ein Fehler aufgetreten sein, so ist der gesamte zugehörige Cluster neu zu bearbeiten (vgl. Zeilen 9 und 10). Nach jedem weiteren Präkonditionierungsschritt werden die zufriedenstellend approximierten Vektoren (Indexmenge $J_p \backslash J_{p+1}$) aufdatiert. Sollte die maximale Anzahl $p_s + p_d$ an vorgesehenen QR-Transformationen überschritten werden, so müssen wir den Gesamtalgorithmus als gescheitert betrachten (Zeilen 16 bis 18). Eine Abhilfe besteht dann darin, die Anzahl p_s der statischen Präkonditionierungsschritte deutlich zu erhöhen und den Gesamtalgorithmus für die Problemfälle erneut zu starten. Ansonsten sollten andere Lösungsverfahren herangezogen werden. Welches dabei gewählt wird, kann u.a. davon abhängen, wie die Informationen aus dem Gesamtalgorithmus noch gewinnbringend eingesetzt werden können.
In Abschnitt 4.3 geben wir Matrizen an, für die die aktuelle Implementierung

Algorithmus 4.1 Gerüst des Gesamtalgorithmus.

Gegeben: Eine bidiagonale Matrix $B \in \mathbb{R}^{n \times n}$.
Gesucht: Näherungen an die Größen Σ, U, V der **bSVD**.
Ein Fehlervektor err.

1: **Flipping:**
Bestimme, ob mit $B_0 = B$ oder $B_0 = B^{(\mathrm{rev})}$ gearbeitet werden soll.
2: **Statische Präkonditionierung:**
Transformiere $B_{p+1} = H_p^T B_p G_p$ für $p = 0 : p_s - 1$.
3: **Splitting:**
Teile B_{p_s} in kleinere Probleme $B_{p_s}^{(1)}, \ \ldots\ , B_{p_s}^{(K)}$ auf.
4: **for** $k = 1 : K$ **do**
5: {Sei $\mathcal{B}_{p_s} := B_{p_s}^{(k)}$ und $\mathcal{N}$ die Dimension von $\mathcal{B}_{p_s}$. Ferner seien $\mathcal{U}$ und $\mathcal{V}$ entsprechend gewählte Submatrizen von U und V und ε der Teilvektor von err.}
Setze $p = p_s$ und $J_{p_s} = \{1, \ \ldots\ , \mathcal{N}\}$.
6: Bestimme Approximationen an die Singulärwerte von $\mathcal{B}_{p_s}$.
7: **while** $(J_p \neq \emptyset) \wedge (p \leq p_s + p_d)$ **do**
8: Für alle $j \in J_p$ bestimme $(\mathcal{U}(:,j), \mathcal{V}(:,j), \varepsilon_j)$ durch Anwendung des Kernalgorithmus (Algorithmus 3.10).
9: $J' = \{j' \in J_p \,|\, \varepsilon_{j'} = 1\}$
10: $J_{p+1} = \{j \in J_p \,|\, \exists j' \in J',$ s.d. σ_j im gleichen Cluster wie $\sigma_{j'}$ liegt.$\}$
11: Für alle $j \in J_p \backslash J_{p+1}$ datiere $\mathcal{U}(:,j)$ und $\mathcal{V}(:,j)$ mit $\mathcal{H}_{p_s}, \ \ldots\ , \mathcal{H}_{p-1}$ bzw. $\mathcal{G}_{p_s}, \ \ldots\ , \mathcal{G}_{p-1}$ auf.
12: $\mathcal{B}_{p+1} = \mathcal{H}_p^T \mathcal{B}_p \mathcal{G}_p$ {Neuer Präkonditionierungsschritt.}
13: $p = p + 1$
14: Aktualisiere ε.
15: **end while**
16: **if** $p > p_s + p_d$ **then**
17: {Breakdown des Gesamtalgorithmus.} **exit**
18: **end if**
19: **end for**
20: **Statische Präkonditionierung:**
Datiere U und V mit $H_0, \ \ldots\ , H_{p_s-1}$ bzw. $G_0, \ \ldots\ , G_{p_s-1}$ auf.
21: **Flipping:**
Permutiere ggf. die Zeilen von U und V.

des Gesamtalgorithmus mit der o.a. Voreinstellung der Parameter p_s und p_d tatsächlich scheitert, vgl. Testprobleme P4 (a) und P5 (a). Für den Großteil der betrachteten Testmatrizen liefert Algorithmus 4.1 jedoch zufriedenstellende Ergebnisse und besticht vor allem durch niedrige Ausführungszeiten. Zur Effizienzsteigerung des Kernalgorithmus können wir also die Eingangsmatrix B einer Vorverarbeitungsphase unterziehen. Eine andere Möglichkeit besteht darin, den Kernalgorithmus selbst um Optionen zu erweitern, die ebenfalls zu Verbesserungen der Ausführungsgeschwindigkeit und Genauigkeit führen können. Diese beschreiben wir in folgenden Abschnitt.

4.2 Ideale Systeme bei sehr engen Clustern

Für Matrizen, deren Eigenwerte eng beieinander liegen, besitzt der Darstellungsbaum des RRR-Algorithmus sehr viele Ebenen. Es kann sogar vorkommen, daß es mit den zur Verfügung stehenden Shift-Parametern nicht gelingt, einen Cluster bis zum Erreichen einer gewissen Ebene recmax aufzulösen. Das Auftreten *untrennbarer Eigenwerte* führt also zu einem Scheitern des Kernalgorithmus (Breakdown). In diesem Abschnitt stellen wir das Konzept *idealer Systeme* aus Eigen- bzw. Singulärvektoren vor, mit dessen Hilfe die Darstellungsbäume erheblich reduziert werden können. Die hier beschriebene Strategie basiert auf Heuristiken, die teilweise durch theoretische Resultate und teilweise durch Beobachtungen bei numerischen Experimenten motiviert sind. Mit geringem Aufwand können wir oft erhebliche Verbesserungen erzielen:

- Bei der erfolgreichen Konstruktion idealer Systeme kann neben signifikanten Einsparungen an Rechenoperationen und Speicherplatz oft auch das Problem untrennbarer Eigenwerte gelöst werden.
- Ist die Erzeugung eines idealen Systems nicht möglich, soll dies mit der Strategie möglichst früh erkannt werden. Dazu legen wir Wert auf einfach zu überprüfende Kriterien, die allerdings in manchen Situationen zu pessimistisch sein können.

Theoretische Grundlagen über die Struktur invarianter Unterräume eng geclusterter Eigenwerte sind in [64] beschrieben. In [28] ist dargestellt, wie diese mit Hilfe von BABE-Faktorisierungen verwendet werden können, um die *Träger* einzelner Vektoren zu minimieren. Die für die LAPACK-Routine `DSTEGR` verwendete Implementierung berücksichtigt bereits diese Ergebnisse. Modifikationen dieser Ansätze münden in die hier skizzierte Vorgehensweise zur Konstruktion eines numerisch orthogonalen Systems aus *allen* zum Cluster gehörenden Eigenvektornäherungen.

Ist innerhalb des RRR-Algorithmus ein Cluster der Größe $s = l - f + 1$ zu bearbeiten, so sind für $s \ll n$ durchaus alternative Verfahren in Betracht zu ziehen. Für $s = 2$ können beispielsweise Bisektoren eingesetzt werden, um zwei Eigenvektorapproximationen gegeneinander zu orthogonalisieren, vgl. Beispiel 1.31. Diese Variante ist mit deutlich weniger Aufwand verbunden als die *a priori* Auswahl eines Shift-Kandidaten, die Bestimmung der beiden Eigenwertnäherungen und die *a posteriori* Überprüfung der relativen Konditionszahlen. Für moderate Clustergrößen (z.B. $s \leq 20$) kann auch allgemeiner eine explizite Reorthogonalisierung wie im modifizierten Gram-Schmidt-Verfahren (Algorithmus 1.2) erfolgen, um auf diese Weise den Darstellungsbaum zu reduzieren.
Die in diesem Abschnitt beschriebene Strategie eignet sich speziell für Cluster mit wenigen Elementen, deren Eigenwerte sehr eng beieinander liegen.

4.2.1 Sehr enge Cluster trotz Unreduziertheit

Spätestens nach der Anwendung der Splitting-Kriterien aus dem vorigen Abschnitt können wir davon ausgehen, daß die zu behandelnde Tridiagonalmatrix T unreduziert ist. In exakter Arithmetik sind damit ihre Eigenwerte paarweise disjunkt und alle Komponenten der Eigenvektoren von Null verschieden, vgl. Bemerkung 1.13. Dennoch können wir unreduzierte Matrizen konstruieren, deren Eigenwerte als Gleitpunktzahlen identisch sind oder nur um wenige ULP voneinander abweichen. Solche Eigenwerte betrachten wir als *sehr eng geclustert*. In der Definition 1.15 entspricht dies also $t = \mathcal{O}(\epsilon)$.
Testprobleme mit sehr engen Clustern sind Wilkinson-Matrizen (Beispiel 1.18) oder geleimte Matrizen (Beispiel 1.19). Durch geeignete Wahl der Parameter lassen sich diese so konstruieren, daß bei Abarbeitung des RRR-Algorithmus untrennbare Eigenwerte oder Schwierigkeiten bei den Faktorisierungen auftreten: Numerisch identische Eigenwerte fallen oft mit den Eigenwerten der Minoren von T zusammen. Geleimte Matrizen werden verwendet, um **SEP**- und **SVD**-Lösungsverfahren in extremen Situationen zu testen.
Da der klassiche RRR-Algorithmus und der daraus abgeleitete Kernalgorithmus in diesen Fällen auf die beschriebenen Probleme stoßen, empfiehlt es sich, die Eigenschaften von Matrizen mit sehr engen Eigenwertclustern genauer zu untersuchen.

4.2.2 Minimierbare Träger trotz Unreduziertheit

Nach Bemerkung 1.13 sind alle Komponenten der Eigenvektoren einer unreduzierten symmetrisch tridiagonalen Matrix von Null verschieden. In diesem

Abschnitt diskutieren wir, unter welchen Umständen die Einträge von Approximationen als vernachlässigbar klein angesehen werden können.

Über Hügel und Täler

Für die nachfolgenden Beobachtungen ist der Begriff der *Envelope* nützlich [64].

Definition 4.1 (Envelope)
Sei $\lambda_f \leq \ \ldots \leq \lambda_l$ ein Eigenwertcluster von T mit zugehörigen $s = l - f + 1$ Eigenvektoren $Q(:, f:l) \in \mathbb{R}^n$. Sei

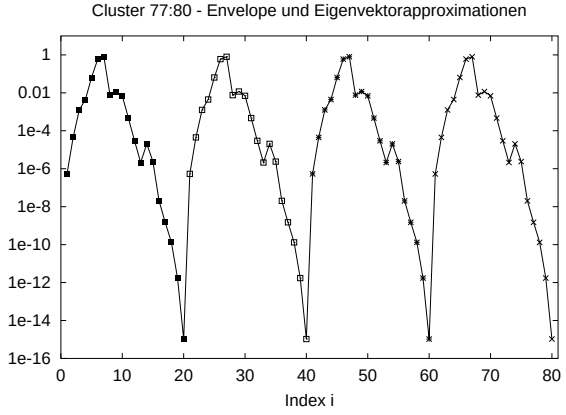
$$\mathcal{S}^{f:l} := \text{span}\{Q(:,f),\ \ldots\ , Q(:,l)\}$$

der zugehörige invariante Unterraum. Für $\mathcal{S}^{f:l}$ definieren wir die *Envelope* $\mathcal{E}^{f:l} \in \mathbb{R}^n$ komponentenweise durch

$$\mathcal{E}_i^{f:l} := \max\{|z_i| : z \in \mathcal{S}^{f:l} \wedge \|z\| = 1\}$$

Beispiel 4.2 (Die Matrix W_{51}^+)
Wir betrachten einige invariante Unterräume der Matrix W_{51}^+. In Abbildung 4.1 auf der folgenden Seite sind sowohl die Envelopen als auch die Komponenten der zugehörigen Eigenvektorapproximationen logarithmisch skaliert dargestellt. Dabei sind nur Elemente erfaßt, deren Betrag größer als die Maschinengenauigkeit ϵ ist. Es fällt auf, daß einige Einträge sehr klein werden. Die Bereiche mit den betragsgrößten Komponenten können sehr schmal ausfallen.

Beispiel 4.3 (Eine geleimte Matrix)
Die Einträge einer Basismatrix $\mathcal{T} \in \mathbb{R}^{20\times 20}$ seien mit Zufallszahlen zwischen 0 und 1 belegt. Wir bilden daraus die dreifach geleimte Matrix $\mathcal{T}(20, 3, 200\epsilon) \in \mathbb{R}^{80\times 80}$. In der Abbildung ist die Struktur des zu den vier betragsgrößten Eigenwerten gehörenden invarianten Unterraumes skizziert.



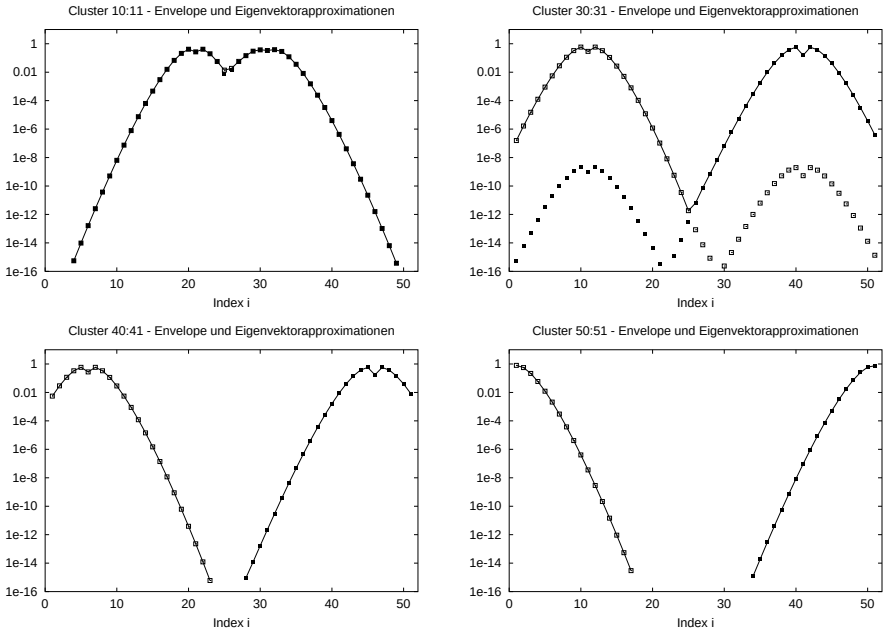


Abbildung 4.1: Komponentenweise Darstellung der Envelopen und Eigenvektorapproximationen. Die Envelopen werden jeweils mit einer durchgezogenen Linie, die Einträge der Vektoren durch offene und geschlossene Quadrate angedeutet.

In der Kontur der Envelope können wir vier „Hügel“ –also Bereiche mit betragsgroßen Einträgen– ausmachen, die durch drei „Täler“ getrennt sind. Ähnlich lassen sich die Envelopen des vorigen Beispiels interpretieren.

Die in den beiden Beispielen gegebene Struktur der Envelope finden wir bei vielen Testproblemen wieder. Wir notieren daher folgende

Beobachtung:
Die Envelope $\mathcal{E}^{f:l}$ eines engen Clusters von s Eigenwerten hat s „Hügel“, die durch $s-1$ „Täler“ getrennt sind.

Wirkung des QR-Verfahrens auf geleimte Matrizen

Für geleimte Matrizen läßt sich die Beobachtung weiter bestätigen, wenn das QR-Verfahren zur Bestimmung von Eigenvektornäherungen herangezogen wird. Nach wenigen Schritten zerfällt die Transformierte der in Beispiel 4.3 erwähnten Matrix $T_0 = \mathcal{T}(20, 3, 200\epsilon)$ durch Anwenden geeigneter

Splitting-Kriterien in vier Teilmatrizen:

$$T_i = (Q_{i-1} \cdots Q_0) T_0 (Q_{i-1} \cdots Q_0)^T$$

mit

$$T_i = \begin{pmatrix} \mathcal{T}^{(1)} & & & \\ & \mathcal{T}^{(2)} & & \\ & & \mathcal{T}^{(3)} & \\ & & & \mathcal{T}^{(4)} \end{pmatrix}$$

Bis auf kleine additive Störungen sind die Matrizen $\mathcal{T}^{(k)}$ für $k = 1:4$ orthogonal ähnlich zu $\mathcal{T}$. (Dabei ist $\mathcal{T}$ die Basismatrix aus Beispiel 4.3). Nach Satz 1.20 kann der Unterschied der Eigensysteme der $\mathcal{T}^{(k)}$ durch kleine additive Störungen beschrieben werden. Bei der Spektralzerlegung von T_i erhalten wir also vier Gruppen von je 20 Eigenvektorapproximationen, bei denen jeweils nur 20 aufeinanderfolgende Komponenten von Null verschieden sind. Da bei engen Eigenwertclustern typischerweise jede der vier Gruppen mit einem Eigenpaar beteiligt ist, bildet sich die in der Beobachtung beschriebene Struktur der Envelope heraus.

Definition 4.4 (Träger, Support)
Mit

$$\mathrm{supp}(z) := \{i \in \{1, \ \ldots \ , n\} \mid z_i \neq 0\}$$

bezeichnen wir den *Träger* (engl. *support*) einer Eigenvektorapproximation $z \in \mathbb{R}^n$. Wir beachten, daß im Gegensatz zu den exakten Eigenvektoren viele Komponenten von z Null sein können, obwohl die Eingangsmatrix unreduziert ist.

Ideale Systeme

Wie bereits in den Abschnitten 1.3.5 und 1.3.6 diskutiert, kann der invariante Unterraum $\mathcal{S}^{f:l}$ eines sehr engen Eigenwertclusters mit weitgehend beliebig gewählten Basen aus Eigenvektornäherungen beschrieben werden, ohne die Genauigkeitsanforderungen für das **tSEP** zu beeinträchtigen. Unter diesen Basen bezeichnen wir diejenigen als *ideale Systeme*, deren Vektoren möglichst gut jeweils einen „Hügel“ der Envelope nachbilden. Anders formuliert sollten sich die Träger der Vektoren möglichst wenig überlappen. Die Vorteile idealer Systeme sind leicht einzusehen:

1. Wenn die Träger der Eigenvektornäherungen klein sind, werden weniger Rechenoperationen und Speicherplatz erforderlich.

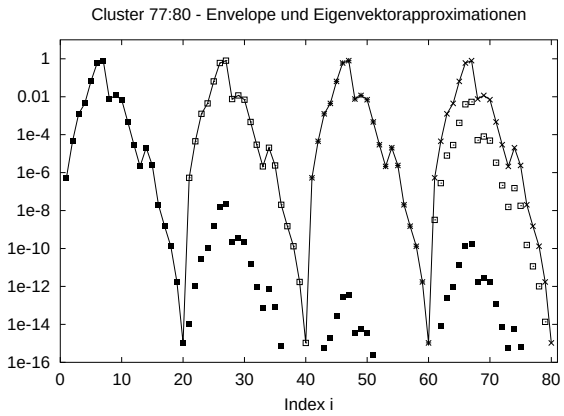
2. Geringe oder gar keine Überlappungen zweier Träger haben positive Auswirkungen auf die numerische Orthogonalität.

3. In vielen Anwendungen ist durch die Aufgabenstellung klar, daß die Bereiche betragsgroßer Komponenten der exakten Eigenvektoren disjunkt sind. Ideale Systeme geben diesen Sachverhalt am besten wieder.

Wegen der Splitting-Kriterien zeichnet sich das QR-Verfahren durch die Eigenschaft aus, ein ideales System numerisch orthogonaler Eigenvektorapproximationen konstruieren zu können. Auch beim Divide-and-Conquer Ansatz erfolgt die Erstellung idealer Systeme oft in natürlicher Weise, da hier die Eingangsmatrix ebenfalls zerlegt wird:

$$T = \mathtt{diag}(T_1, T_2) + xx^T$$

Siehe Abschnitt 1.5.1. Ferner tritt auch bei nicht geleimten Matrizen mit engen Eigenwertclustern das Phänomen der *Deflation* auf, wodurch die drei o.g. gewünschten Eigenschaften erzielt werden.
Verfahren wie die Inverse Iteration und auch der RRR-Algorithmus verändern die Eingangsmatrix T dagegen nicht. Überspitzt formuliert wird bei einem engen Cluster ein beliebiger Vektor aus $\mathcal{S}^{f:l}$ approximiert und die übrigen orthogonal dazu konstruiert. Ein typisches Ergebnis könnte so aussehen:



Die beiden folgenden Abschnitte dienen dem Ziel, den RRR-Algorithmus so zu modifizieren, daß die Konstruktion idealer Systeme möglich ist. Zunächst beschreiben wir, wie der Träger einer einzelnen Eigenvektorapproximation möglichst klein gewählt werden kann. Danach skizzieren wir eine Strategie, mit der alle Vektoren des Clusters zu einem idealen System zusammengefügt werden.

4.2.3 Trägerminimierung für eine Eigenvektorapproximation

In diesem Abschnitt entwickeln wir ein Verfahren zur Trägerminimierung einer Eigenvektorapproximation. Dazu leiten wir aus der in [64] skizzierten Idee zur Faktorisierung geeigneter Submatrizen konkret implementierbare Kriterien her. Der Vorschlag aus [28] behandelt isolierte Eigenwerte, während der hier beschriebene Minimierungsansatz bereits darauf angelegt ist, ein ideales System für Cluster zu erzielen.

Ein LGS mit modifizierter rechter Seite

In Abschnitt 2.2 haben wir das Verfahren zur Bestimmung einer Eigenvektorapproximation als Lösen eines LGS

$$N_k^+ G_k^+ (N_k^+)^T \cdot \bar{q}^{(k)} = r^{(k)} = \gamma_k^+ e_k$$

durch die Vorschrift (2.3) hergeleitet. Mit Hilfe zweier Parameter k_l und k_u modifizieren wir diese nun zu

$$\begin{array}{lcll} \bar{\varrho}_k^{(k)} & = & 1 & \\ \bar{\varrho}_i^{(k)} & = & -l_i^+ \bar{\varrho}_{i+1}^{(k)} & \text{für } i = k-1 : -1 : k_l \\ \bar{\varrho}_i^{(k)} & = & 0 & \text{für } i = k_l - 1 : -1 : 1 \\ \bar{\varrho}_{i+1}^{(k)} & = & -u_i^+ \bar{\varrho}_i^{(k)} & \text{für } i = k : k_u - 1 \\ \bar{\varrho}_{i+1}^{(k)} & = & 0 & \text{für } i = k_u : n-1 \end{array} \tag{4.2}$$

Formal entspricht dies der Lösung von

$$N_k^+ G_k^+ (N_k^+)^T \cdot \bar{\varrho}^{(k)} = \bar{r}^{(k)} \tag{4.3}$$

wobei die rechte Seite $\bar{r}^{(k)}$ gegeben ist durch

$$\bar{r}^{(k)} = \gamma_k^+ e_k + d_{k_l-1}^+ l_{k_l-1}^+ (\bar{q}_{k_l} e_{k_l-1} - \bar{q}_{k_l-1} e_{k_l}) + r_{k_u+1}^+ u_{k_u}^+ (\bar{q}_{k_u} e_{k_u+1} - \bar{q}_{k_u+1} e_{k_u})$$

Gegenüber $r^{(k)}$ verändert sich das neue Residuum also in maximal vier Komponenten, vgl. Abbildungen 2.2 (Seite 39) und 4.2 (folgende Seite).
Wir können dieses Vorgehen auch interpretieren als eine BABE-Faktorisierung der mit einer leichten Störung versehenen Submatrix $T(k_l : k_u, k_l : k_u)$. Es wird dann ein $(k_u - k_l + 1)$-dimensionales LGS gelöst [64]. Wir sehen einen Vorteil der hier gewählten Variante darin, einfach zu ermittelnde Kriterien zur Auswahl der Parameter k_l und k_u angeben zu können.

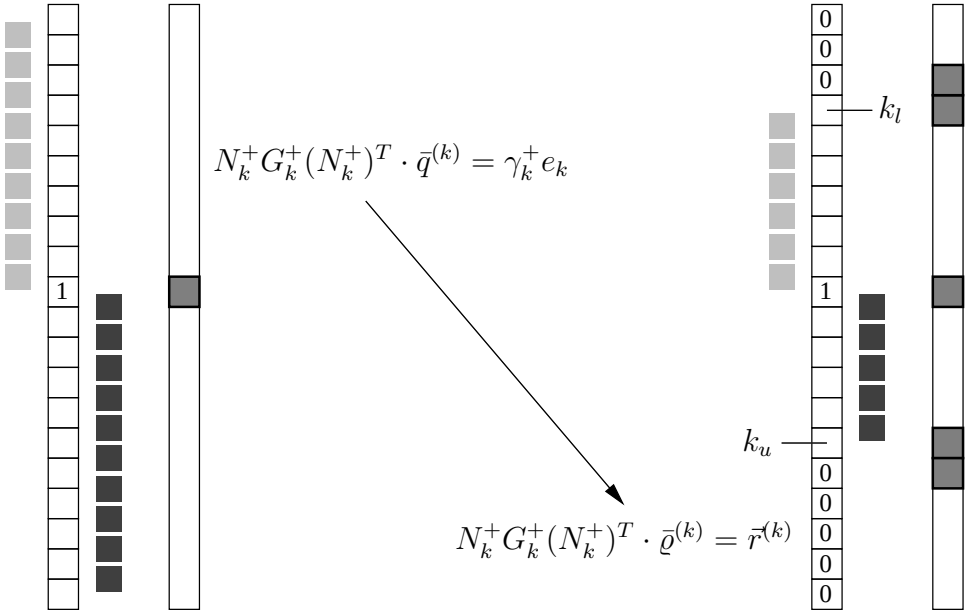


Abbildung 4.2: Lösen der Gleichung (4.3) mit modifiziertem Residuum.

Ein Abbruchkriterium

Der durch $k_u - k_l + 1$ gegebene Umfang des Trägers von $\bar{r}^{(k)}$ soll minimiert werden, ohne daß sich $\|\bar{r}^{(k)}\|$ gegenüber $\|r^{(k)}\| = |\gamma_k^+|$ erheblich vergrößert. Wegen

$$\|\bar{r}^{(k)} - r^{(k)}\| \leq |d_{k_l-1}^+ l_{k_l-1}^+|(|\bar{q}_{k_l}| + |\bar{q}_{k_l-1}|) + |r_{k_u+1}^+ u_{k_u}^+|(|\bar{q}_{k_u+1}| + |\bar{q}_{k_u}|)$$

akzeptieren wir $k_l < k$ als den maximalen Index von $\bar{q}^{(k)}$ mit

$$|d_{k_l-1}^+ l_{k_l-1}^+|(|\bar{q}_{k_l}| + |\bar{q}_{k_l-1}|) \leq \epsilon|\gamma_k^+| \qquad \text{und} \qquad |\bar{q}_{k_l-1}| \leq \epsilon, \quad |\bar{q}_{k_l}| \leq \epsilon \quad (4.4)$$

und $k_u > k$ als den minimalen Index, für den

$$|r_{k_u+1}^+ u_{k_u}^+|(|\bar{q}_{k_u+1}| + |\bar{q}_{k_u}|) \leq \epsilon|\gamma_k^+| \qquad \text{und} \qquad |\bar{q}_{k_u}| \leq \epsilon, \quad |\bar{q}_{k_u+1}| \leq \epsilon \quad (4.5)$$

gilt. Wir können also $\bar{\varrho}^{(k)}$ ermitteln, indem wir die Berechnung von $\bar{q}^{(k)}$ bei geeigneten Werten von k_l und k_u abbrechen und die restlichen Einträge auf Null setzen. Dabei wird die Norm des Residuums schlimmstenfalls verdreifacht.

Bemerkung 4.5 (Alternative mit RQI-Korrektur)
Wir können $\bar{\varrho}^{(k)}$ ebenfalls als hinreichend genaue Eigenvektorapproximation akzeptieren, wenn die RQI-Korrektur (Gleichung 3.9) klein genug wird.

Wegen

$$(\bar{\varrho}^{(k)})^T \bar{r}^{(k)} = \gamma_k^+ - d_{k_l-1}^+ l_{k_l-1}^+ \bar{q}_{k_l-1} \bar{q}_{k_l} - r_{k_u+1}^+ u_{k_u}^+ \bar{q}_{k_u} \bar{q}_{k_u+1}$$

können also auch die Terme $d_{k_l-1}^+ l_{k_l-1}^+ \bar{q}_{k_l-1} \bar{q}_{k_l}$ und $r_{k_u+1}^+ u_{k_u}^+ \bar{q}_{k_u} \bar{q}_{k_u+1}$ für Abbruchkriterien herangezogen werden.

Bei der Trägerminimierung für isolierte Eigenwerte wird in [28] geprüft, ob zwei aufeinanderfolgende Elemente von $\bar{q}^{(k)}$ betragsmäßig kleiner als die Maschinengenauigkeit ϵ sind. Ist dies der Fall, so werden alle nachfolgenden Einträge auf Null gesetzt. Das folgende Beispiel zeigt, daß dieses Kriterium bei Clustern i.A. ungenügend ist und daß stattdessen die in den Gleichungen (4.4) und (4.5) beschriebenen Auswirkungen auf das Residuum bzw. die RQI-Korrektur berücksichtigt werden sollten.

Beispiel 4.6 (Minimierung des Trägers)
Die Eigenwerte einer gegebenen Basismatrix $T \in \mathbb{R}^{30\times 30}$ seien zufällig zwischen ϵ^2 und 1 gestreut. Wir bilden daraus die zweifach geleimte Matrix $T(30, 2, 10^{-4}) \in \mathbb{R}^{90\times 90}$ und betrachten die Näherung an den zu $\lambda_5 \approx 100\epsilon^2$ gehörenden Eigenvektor, vgl. Abbildung 4.3. Das minimale Twist-Element finden wir an Position 58. Mit dem Abbruchkriterium aus den Gleichungen (4.4) und (4.5) werden alle gewichtigen Einträge erfaßt. Die Parameter $k_l = 26$ und $k_u = 90$ bilden eine Unter- und Obergrenze für $\mathrm{supp}(\bar{\varrho}^{(58)})$.
Im *Anschluß an die Normierung* können wir dann noch alle betragsmäßig unterhalb der Maschinengenauigkeit ϵ liegenden Elemente auf Null setzen, ohne die Genauigkeit des skalierten Vektors wesentlich zu verschlechtern. Dies bewirkt oft eine weitere Verringerung des Umfangs des Trägers.
Die drei kleinsten Eigenwerte von $T(30, 2, 10^{-4})$ sind isoliert, während das restliche Spektrum aufgeteilt ist in 29 Cluster mit je drei Elementen. Je größer die Eigenwerte sind, desto enger liegen sie beieinander. In Abbildung 4.4 wird deutlich, daß die Träger dann auf immer kleinere Bereiche minimiert werden können.

Zusammenfassung der bisherigen Ergebnisse

Die Trägerminimierung für eine einzelne Eigenvektorapproximation führt auch bei vielen anderen Testproblemen zu positiven Effekten bzgl. Rechenaufwand, Speicherbedarf und Genauigkeit. Das Abbruchkriterium kann simultan zur Berechnung der Näherung überprüft werden, ist aber mit einem Mehraufwand an Operationen verbunden. (Im schlechtesten Fall sind pro Schritt je eine zusätzliche Multiplikation und Addition sowie zwei Vergleichsoperationen durchzuführen.) Bei einer Implementierung sollte das Abbruchkriterium optional zuzuschalten sein.

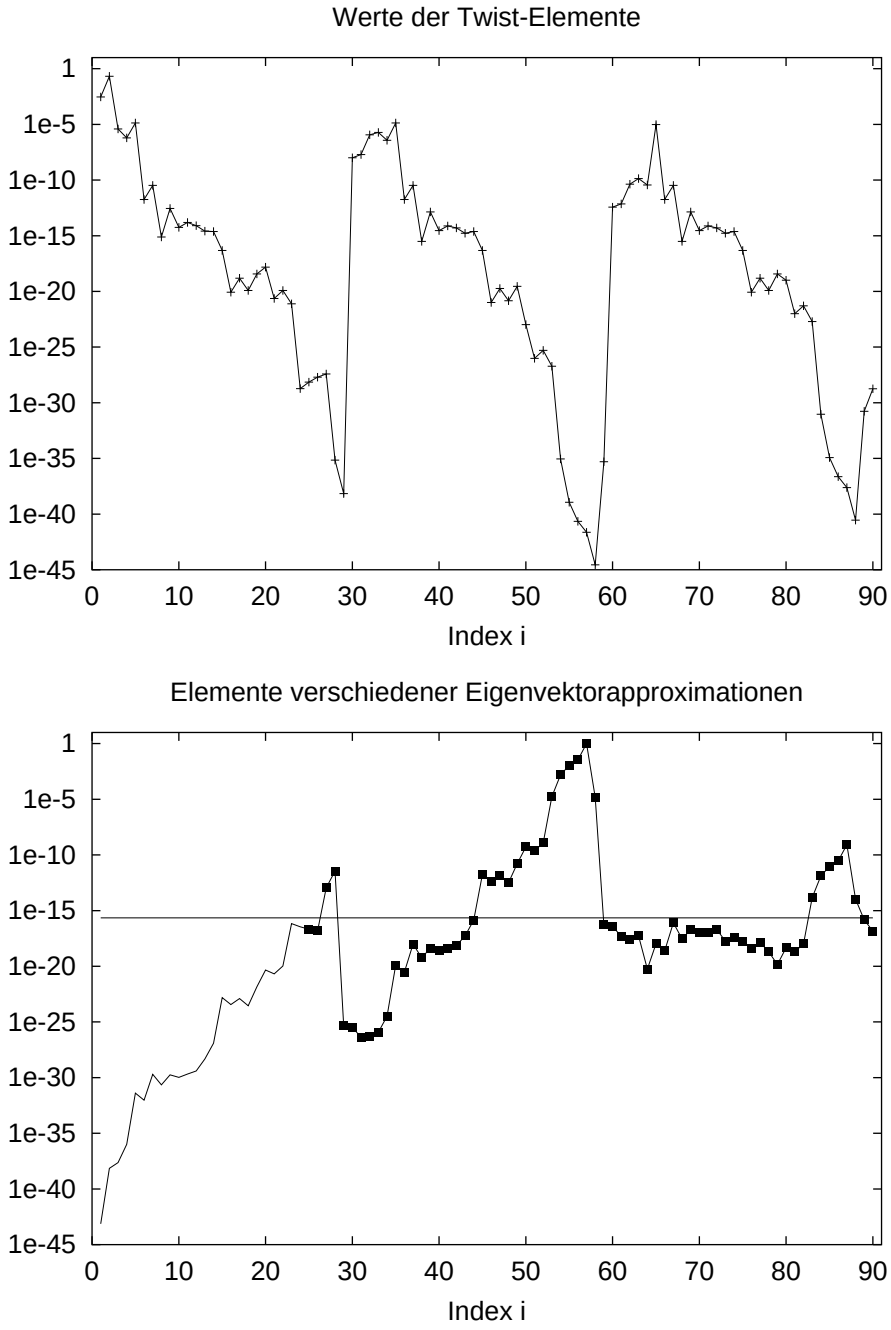


Abbildung 4.3: Zu Beispiel 4.6. Die obere Abbildung zeigt die Werte der Twist-Elemente logarithmisch skaliert. Darunter sind die Einträge der Lösung $\bar{q}^{(58)}$ mit der durchgezogenen Linie markiert. Die schwarzen Quadrate bezeichnen die von Null verschiedenen Elemente von $\bar{\varrho}^{(58)}$. Ein Kriterium, das nur die Einträge, nicht aber die Auswirkungen auf das Residuum einbezieht, würde die Komponenten $1:44$ und $60:90$ zu Null setzen.

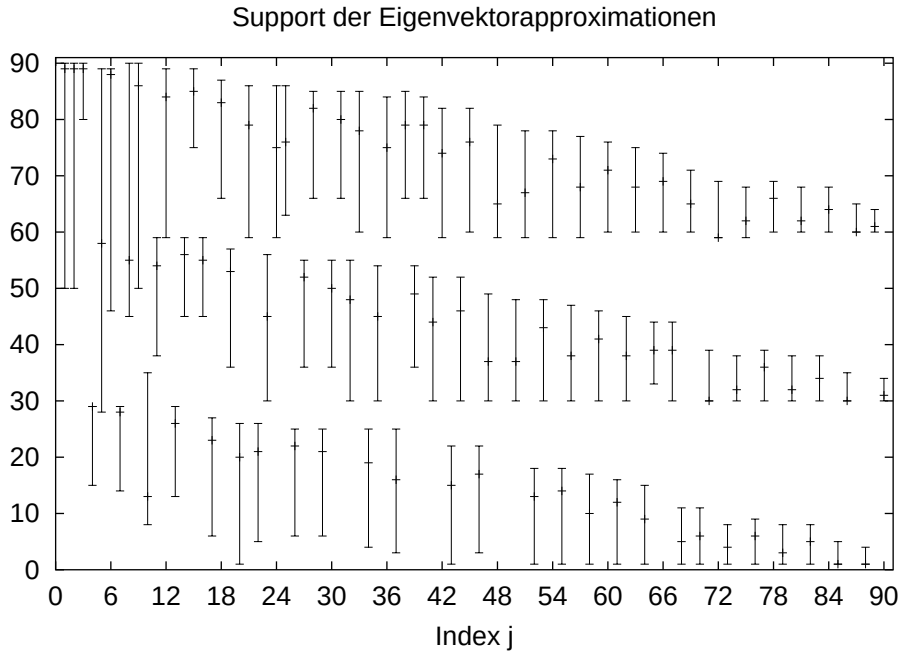


Abbildung 4.4: Minimierte Träger aller Eigenvektorapproximationen aus Beispiel 4.6. Für jede Näherung ist der durch k_l und k_u begrenzte Träger sowie die darin enthaltene Position k des minimalen Twist-Elementes dargestellt.

Weitere signifikante Einsparungen können vor allem dann erzielt werden, wenn basierend auf der Trägerminimierung die Konstruktion eines idealen Systems gelingt.

4.2.4 Ideale Systeme mit dem RRR-Algorithmus

Zur Erzeugung eines idealen Systems für alle Elemente eines Eigenwertclusters $\lambda_f \leq \ \ldots \leq \lambda_l$ sind die Träger der $s = l - f + 1$ Eigenvektorapproximationen so zu minimieren, daß möglichst wenige Überlappungen auftreten. Es sind also s Gleichungssysteme mit geeigneten Residuen zu lösen, vgl. Abbildung 4.5.
Das nachfolgend beschriebene Vorgehen besteht darin, die Elemente des Clusters in einer gewissen Reihenfolge zu bearbeiten. Zunächst wird eine geeignete Twist-Position ausgewählt und das entsprechende Gleichungssystem gelöst. Simultan erfolgt die Trägerminimierung. Dabei können auch geringfügige Überlappungen der Träger akzeptiert werden.
Die Strategie basiert weitgehend auf Heuristiken und Erfahrungen aus nu-

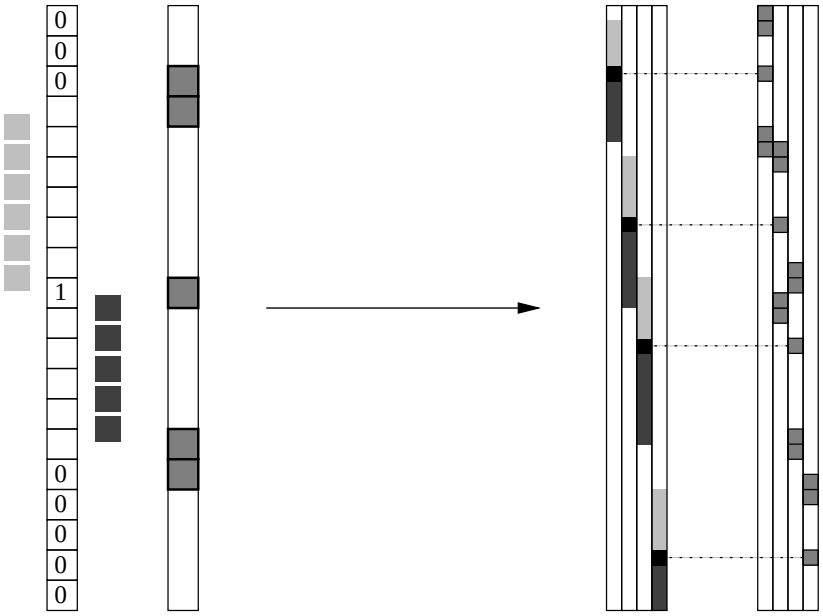


Abbildung 4.5: Von der einfachen Trägerminimierung zu idealen Systemen.

merischen Experimenten. Für die Teilprobleme sind vor allem schnell überprüfbare Kriterien interessant, um im Falle eines Scheiterns nur moderaten zusätzlich investierten Rechenaufwand vertreten zu müssen.

Reihenfolge der Abarbeitung

Innerhalb eines Clusters kann eine Substruktur der Eigenwerte vorliegen. Beispielsweise sind die ersten und letzten beiden Elemente von

$$[1, 1+\epsilon, 1+\sqrt{\epsilon}, 1+2\sqrt{\epsilon}, 1+2\sqrt{\epsilon}+\epsilon] \tag{4.6}$$

jeweils sehr eng geclustert, während das mittlere innerhalb der Substruktur als „isoliert“ betrachtet werden kann. Es empfiehlt sich, zuerst die engen Cluster und danach die „isolierten“ Eigenwerte zu bearbeiten. Bei mehreren engen Clustern beginnen wir mit dem, der die meisten Eigenwerte enthält.

Auswahl von s geeigneten Twist-Positionen

Innerhalb eines Clusters variieren die Größenordnungen der Twist-Elemente bei den einzelnen BABE-Faktorisierungen nur wenig. Insbesondere lassen sich in Analogie zu den Envelopen oft s „Täler“ ausmachen, vgl. Abbildung 4.3. Pro Eigenwert stellen wir eine Liste hinreichend kleiner Twist-Elemente aus den „Tälern“ auf.

Wir bestimmen dann nacheinander minimierte Träger. Dabei führen wir Buch über die Menge der noch nicht belegten Indizes. Die Auswahlmöglichkeiten für diese freien Indizes verringern sich also mit jeder erfolgreich bearbeiteten Eigenvektorapproximation. Kann kein hinreichend kleines Twist-Element mit einem freien Index mehr gefunden werden, so wird die Konstruktion des idealen Systems abgebrochen.
In Beispiel 4.6 liegt diese ungünstige Situation vor: Wir nehmen an, daß die zu λ_5 gehörende Eigenvektorapproximation $\bar{\varrho}^{(58)}$ zuerst bestimmt wird. Ferner können wir davon ausgehen, daß sich die Lage der „Täler“ bei der Behandlung der beiden übrigen Eigenwerte des Clusters nicht ändert. Nach Bestimmung des Trägers von $\bar{\varrho}^{(58)}$ sind aber nur noch die Indizes $1 : 23$ frei. Unter diesen finden wir jedoch keine Kandidaten für hinreichend kleine Twist-Elemente mehr. Bei engeren Clustern ist die Situation jedoch häufig günstiger, vgl. Abbildung 4.4.

Kleine Überlappungen der Träger

Im optimalen Fall sind die Träger aller Eigenvektornäherungen eines Clusters paarweise disjunkt. Geringe Überlappungen können jedoch unter gewissen Bedingungen ebenfalls akzeptiert werden. Ist die Schnittmenge zweier Träger klein, läßt sich mit moderatem Aufwand beispielsweise das Skalarprodukt auswerten, um zu überprüfen, ob die Genauigkeitsanforderungen erfüllt sind. Bei numerisch identischen Eigenwerten können oft lineare Abhängigkeiten zwischen den überlappenden Teilen durch Auswahl geeigneter Linearkombinationen ausgenutzt werden.
Unberührt von diesen auf die spezielle Struktur zugeschnittenen Ansätzen bleibt die Möglichkeit, partielle Reorthogonalisierungsmaßnahmen mit Varianten des Gram-Schmidt-Verfahrens (Algorithmus 1.2) durchzuführen. Einzelne Schritte müssen dabei nur auf der Schnitt- bzw. Vereinigungsmenge der jeweiligen Träger durchgeführt werden.
Wir verzichten hier auf die Darstellung aller Details einer konkreten Implementierung.

Unterschiede zwischen erster und höheren Ebenen

Wie bereits erwähnt erhöhen sich die Chancen zur erfolgreichen Konstruktion eines idealen Systems, je enger die Eigenwerte geclustert sind. Neben der Größe s des Clusters kann also auch seine Substruktur *a priori* für oder gegen die Anwendung sprechen. Ideale Systeme und der RRR-Algorithmus ergänzen sich dabei sehr gut: Enge Cluster begünstigen schmale Träger der Eigenvektornäherungen, während sie im Darstellungsbaum das Abarbeiten

mehrerer Ebenen erzwingen. Die Eigenwerte weniger enger Cluster können dagegen bereits nach einmaligem Anwenden der Shift-Strategie isoliert werden, während die Bildung eines idealen Systems oft scheitert.
Die meisten idealen Systeme können bereits auf der ersten Ebene — also vor Einsatz der Shift-Strategie mit Berechnung neuer Eigenwertnäherungen — erfolgreich konstruiert werden. Da auf höheren Ebenen enge Cluster immer seltener vorkommen, sinkt dort die Wirksamkeit der Strategie. Dennoch empfiehlt sich ihr Einsatz auch hier, um Cluster mit Substrukturen wie in Gleichung (4.6) zu bearbeiten.

4.2.5 Übertragung auf die bSVD

Beim Übergang vom **tSEP** auf die **bSVD** müssen zwei ideale Systeme aus Singulärvektornäherungen konstruiert werden, die gut gekoppelt sind. Es ergeben sich daraus in natürlicher Weise bei allen Teilproblemen zusätzliche Bedingungen, die wieder ohne größeren algorithmischen Mehraufwand eingebettet werden können.
Auf der ersten Ebene empfiehlt sich zur Erhaltung guter Kopplungen die Faktorisierung der Golub-Kahan Matrix. Wir minimieren die Träger der entsprechenden Eigenvektornäherungen und extrahieren anschließend daraus Approximationen an linke und rechte Singulärvektoren. Mit Ausnahme der Gefahr von Breakdowns fallen die Gründe, die bei der Übertragung des RRR-Algorithmus auf die **bSVD** gegen die Verwendung der Golub-Kahan Matrix sprechen, für die Konstruktion idealer Systeme *nicht* ins Gewicht.
Beim Auftreten von Breakdowns bleibt als Alternative die Verwendung gekoppelter Normalengleichungen, vgl. Kapitel 3. Diese sind auch auf höheren Ebenen im Darstellungsbaum zu benutzen. Für die explizit faktorisierte Normalengleichung wenden wir die in den Abschnitten 4.2.3 und 4.2.4 skizzierte Strategie an. Nachdem so eine Approximation an den rechten Singulärvektor ermittelt ist, verwenden wir die durch Kopplung gewonnenen Daten, um eine Näherung an den linken Singulärvektor zu bestimmen. Auch dieser Vorgang geht einher mit der Überprüfung der Abbruchkriterien (4.4) und (4.5).
Die *Reihenfolge der Abarbeitung* wird durch die Eigenwerte der explizit faktorisierten Matrix festgelegt. Bei der *Auswahl von s geeigneten Twist-Positionen* können wir ausnutzen, daß die „Täler“ von $\hat{\Gamma}^+$ und $\check{\Gamma}^+$ ohnehin weitgehend übereinstimmen. Meist können identische Twist-Positionen gewählt werden, vgl. Bemerkung 3.11. Bei *kleinen Überlappungen der Träger* sind Orthogonalisierungsmaßnahmen aufeinander abzustimmen.
Die entstehenden Träger der linken und rechten Singulärvektorapproximation sollten sich weitgehend überlappen. Sind gleichzeitig die Träger aller Sin-

gulärvektorpaare disjunkt, so ergibt sich aus den Eigenschaften des idealen Systems eine gute Qualität der Kopplungen.

Der Aufbau idealer Systeme für die **bSVD** läßt sich also durch leichte Modifikationen der Strategie für das **tSEP** realisieren. Wegen der eingebetteten Zusatzbedingungen ist die Gefahr eines Scheiterns prinzipiell höher. In der Praxis zeigen sich jedoch kaum negative Auswirkungen.

Mit der Vorverarbeitung und dem Konzept idealer Systeme ist die Präsentation nützlicher Ergänzungen zum Kernalgorithmus (Algorithmus 3.10) abgeschlossen, vgl. Abbildung 4.6. Zusammengenommen bilden sie ein **bSVD**-Lösungsverfahren, das die vorzüglichen Eigenschaften des klassischen RRR-Algorithmus erhält, aber gleichzeitig effiziente Alternativen in den kritischen Fällen bietet. Die praktische Realisierung dieses Verfahrens liegt mit der im folgenden Abschnitt eingesetzten Implementierung der Routine `DBSVDSolv` vor.

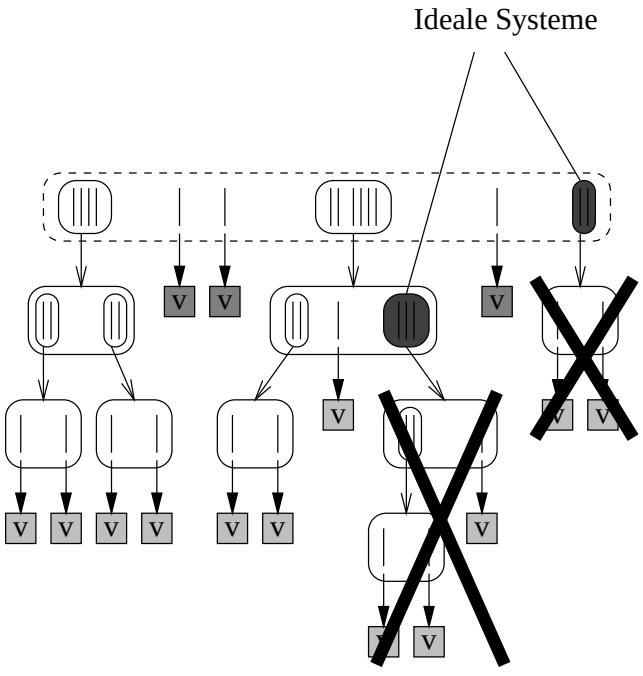


Abbildung 4.6: Ideale Systeme verkleinern den Darstellungsbaum.

4.3 Numerische Experimente

In diesem Abschnitt diskutieren wir die Ergebnisse serieller numerischer Experimente. Nach einer kurzen Beschreibung der Rahmenbedingungen vergleichen wir die Implementierung der Routine `DBSVDSolv` mit den LAPACK-Routinen `DSTEGR`, `DBDSDC` und `DBDSQR`. Anschließend untersuchen wir die Auswirkungen von Kopplungen, Präkonditionierung und der Konstruktion idealer Systeme genauer.

4.3.1 Rahmenbedingungen

Hardware

Die in diesem Abschnitt präsentierten Ergebnisse stammen von einer SUN Enterprise 450. Diese verfügt über vier UltraSPARC II Prozessoren (300MHz) mit einer Peak-Performance von 600 MFlops. Es stehen 1024 MB gemeinsamer Arbeitsspeicher zur Verfügung. Wir ziehen Messungen in doppelter Genauigkeit auf einem Prozessor zur Beurteilung der Verfahren heran.

Implementierungen

Wir vergleichen das neu entwickelte Lösungsverfahren `DBSVDSolv` einerseits mit den LAPACK-Routinen `DBDSDC` und `DBDSQR`, die das Divide-and-Conquer bzw. QR-Verfahren zur **bSVD** implementieren. In der LAPACK-Version 3.0 sind einige Vorschläge aus [29] zur Lösung des **tSEP** mit dem RRR-Algorithmus als Routine `DSTEGR` umgesetzt. Es ist zu beachten, daß in dieser Implementierung noch nicht alle in Kapitel 2 dargestellten Eigenschaften realisiert sind. Für zukünftige Versionen sind bedeutende Erweiterungen geplant. In der weitgehend unabhängig entwickelten Routine `DBSVDSolv` sind — insbesondere bei der Realisierung von Algorithmus 2.5 — viele Teilprobleme mit unterschiedlichen Ansätzen gelöst. Ein direkter Vergleich der LAPACK-Routine mit `DBSVDSolv` ohne Kopplungen führt daher zu uneinheitlichen Ergebnissen.
Beispielsweise wird in `DSTEGR` versucht, anhand von Gershgorin-Abschätzungen *a priori* eine geeignete Twist-Position zu finden. Gelingt dies, so müssen die LDL^T- und URU^T-Faktorisierungen nur bis zu dieser Twist-Position durchgeführt werden. Dadurch wird die Anzahl der Rechenoperationen halbiert.
In der aktuellen LAPACK-Implementierung wird die in Abschnitt 2.4 beschriebene Problematik der Bestimmung einer RRR teilweise umgangen. Die Auswahl der Shift-Kandidaten erfolgt so, daß die Translation entweder positiv oder negativ definit ist. Mit dieser Strategie können also nur Cluster mit

$f = 1$ oder $l = n$ behandelt werden. In allen anderen Fällen wird auf Inverse Iteration zurückgegriffen.
Die getestete Version von `DBSVDSolv` umfaßt den um die Präkonditionierung ($p_s = 1$ und $p_d = 10$) und die Konstruktion idealer Systeme (für Clustergrößen $s \leq 20$) ergänzten Kernalgorithmus. Bei der Bestimmung von Singulärvektornäherungen wird grundsätzlich das Verfahren zur Trägerminimierung angewendet. Für die Kopplungen verwenden wir die Optimierungsoption I aus Bemerkung 3.11. Bei isolierten Eigenwerten führen wir die in Abschnitt 3.2.3 beschriebene Nachiteration zur Verbesserung der Genauigkeit durch. Teilprobleme mit einer Dimension ≤ 20 werden mit dem QR-Verfahren gelöst.
Für verschiedene Testmatrizen zeigen die beiden RRR-basierten Implementierungen aufgrund unterschiedlicher Designentscheidungen ein voneinander abweichendes Verhalten. Einige gravierende Unterschiede sind nachfolgend dokumentiert.

4.3.2 Testprobleme

Alle Routinen der LAPACK-Bibliothek sowie `DBSVDSolv` sind als universelle Lösungsverfahren angelegt. Es sollen also möglichst gute Ergebnisse für beliebige (tri- oder bidiagonale) Matrizen erzielt werden, ohne daß im Einzelfall zusätzlich bekannte Eigenschaften ausgenutzt werden. Die Verfahren werden anhand einer Serie ausgewählter Testprobleme verglichen. Die *Standard-Matrizen* beinhalten die in [23] vorgeschlagenen Matrixklassen zum Testen von LAPACK-Routinen. Darüberhinaus untersuchen wir das Verhalten der Lösungsverfahren für *geleimte Matrizen*. In der nachfolgenden Auflistung beziehen wir uns auf positiv definite symmetrische Tridiagonalmatrizen T und erhalten B als deren Cholesky-Faktor.

Standard-Matrizen

Einige Testprobleme sind durch Vorgabe der Eigenwerte festgelegt [23]. Der Parameter η kann variiert werden (z.B. $\eta = \epsilon =$ Maschinengenauigkeit). Für Skizzen der Spektren siehe Abbildung 4.7.

P1: Geometrische Verteilung: $\lambda_j = \eta^{(n-j)/(n-1)}$, $j = 1 : n$

P2: Arithmetische Verteilung: $\lambda_j = \eta + (1 - \eta)(j - 1)/(n - 1)$, $j = 1 : n$

P3: Cluster bei η: $\lambda_j \approx \eta$, $j = 1 : n - 1$, $\lambda_n = 1$

P4: Cluster bei 1: $\lambda_1 = \eta$, $\lambda_j \approx 1$, $j = 2 : n$

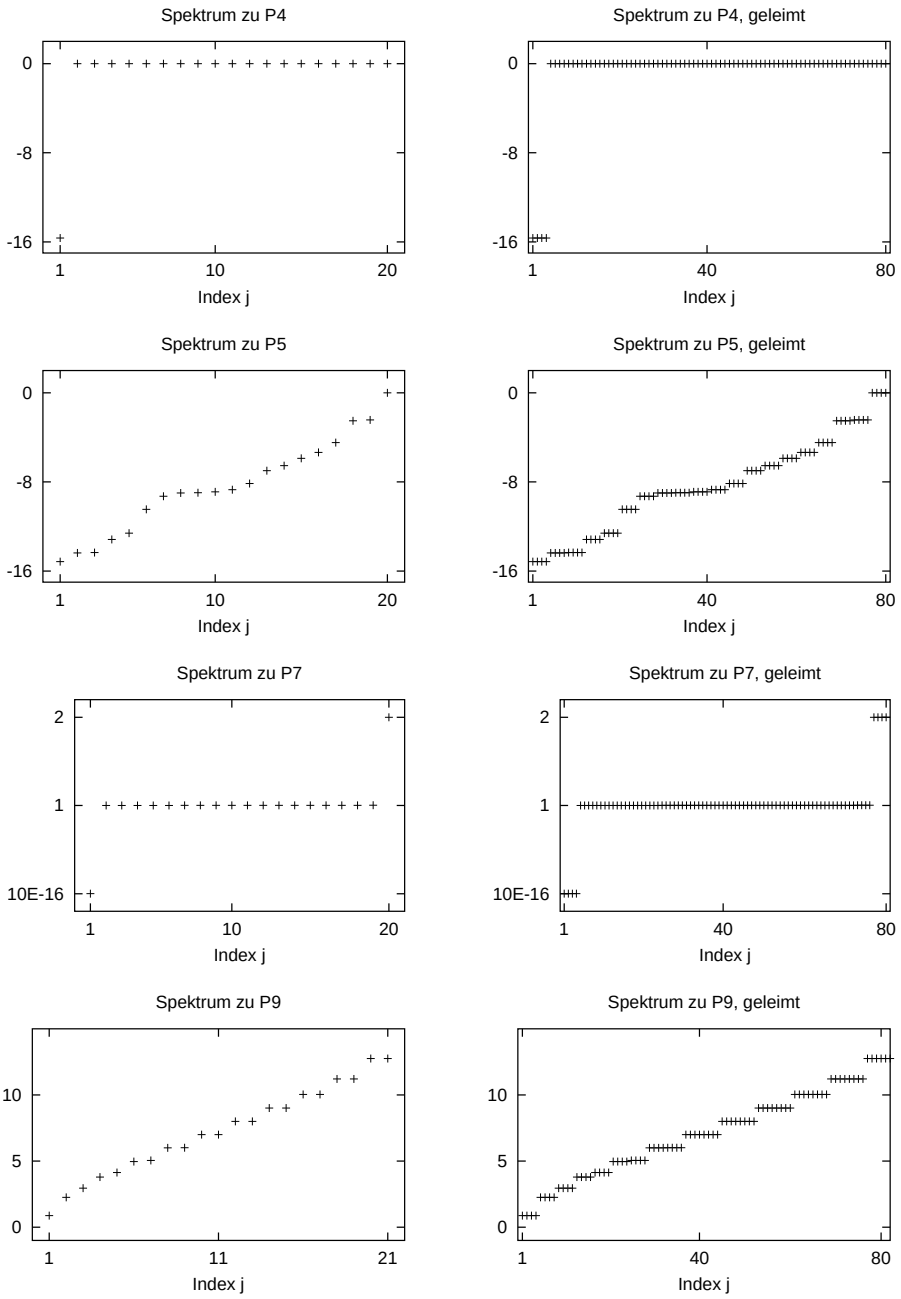


Abbildung 4.7: Spektren einiger Testprobleme für $\eta = \epsilon$. Aufgetragen sind die zum jeweiligen Index gehörenden Eigenwerte. In den oberen vier Abbildungen ist die vertikale Achse logarithmisch skaliert.

P5: Zufällige geometrische Verteilung:
$\lambda_j = r_j \eta^{(n-j)/(n-1)}$, $j = 1 : n$, wobei $r_j \in]0, 1]$ Zufallszahlen sind.

P6: Zufällige arithmetische Verteilung: Analog zu P5.

P7: Innerer Cluster: $\lambda_1 = \eta$, $\lambda_j = 1 + \sqrt{\eta}(j-1)$, $j = 2 : n-1$, $\lambda_n = 2$

Weitere Testmatrizen sind durch ihre Einträge vorgegeben:

P8: 1-2-1 Matrix: Die Diagonaleinträge werden auf 2, die Nebendiagonaleinträge auf 1 gesetzt.

P9: Wilkinson-Struktur: $T = B^T B = W_n^+ + 2I$ (Beispiel 1.18)

Geleimte Matrizen

Ausgehend von einer Basismatrix $T_{P_i} \in \mathbb{R}^{n \times n}$ bilden wir wie in Beispiel 1.19 mit $T_{P_i}(n, k, g) \in \mathbb{R}^{(k+1)n \times (k+1)n}$ eine k-fach geleimte Matrix. Typischerweise besitzt $T_{P_i}(n, k, g)$ Cluster der Größe $k+1$, die in der Nähe der Eigenwerte der Basismatrix liegen. Mit dem Paramter g (z.B. 200ϵ, $\sqrt{\epsilon}$, 10^{-4}) kann die Enge der Cluster beeinflußt werden. Wie bereits erwähnt lassen sich damit gezielt Testprobleme erzeugen, mit denen das Verhalten der Lösungsverfahren in Extremfällen beobachtet werden kann.

4.3.3 Vergleich der Lösungsverfahren

Zum Vergleich der Verfahren werden die Ausführungszeiten, die numerische Orthogonalität und die Residuen herangezogen, vgl. Abschnitt 1.3.5.

Ausführungszeiten

Die in den Tabellen 4.1 und 4.2 aufgelisteten Ausführungszeiten zeigen, daß die quadratische Komplexität der RRR-basierten Ansätze mit wachsender Martrixdimension zu überlegener Leistung führen. Mit Ausnahme des QR-Verfahrens zeigen die Implementierungen jedoch in Abhängigkeit von der Problemklasse stark variierendes Verhalten.

Splitting

Für die Testprobleme P1, P3 und P5 profitieren die Lösungsverfahren oft von der Möglichkeit, die n-dimensionale Ausgangsmatrix auf wesentlich kleinere Teilmatrizen reduzieren zu können, vgl. Abschnitt 4.1.1. Werden die in Abschnitt 1.3.5 genannten Größen als Maß für die Genauigkeit der Eigen-

Problem	n	DSTEGR	DBSVDSolv	DBDSDC	DBDSQR
P1	500	0.09	0.31	0.38	11.80
P1	1000	0.36	1.28	1.82	130.27
P1	2000	1.54	5.22	7.95	
P1	4000	6.37	21.51	45.49	
P2	500	0.28	0.44	1.74	23.91
P2	1000	1.25	1.76	11.31	272.46
P2	2000	5.20	7.20	75.10	
P2	4000	21.41	29.94		
P3	500	0.04	0.03	0.08	32.28
P3	1000	0.17	0.09	0.53	363.84
P3	2000	0.74	0.31	2.68	
P3	4000	2.99	1.18	12.91	
P4	500	0.43	1.35	0.44	36.27
P4	1000	1.84	5.41	1.87	369.46
P4	2000	7.38	21.13	8.32	
P4	4000	30.38	94.40	37.91	
P5	500	0.07	0.31	0.27	3.65
P5	1000	0.26	1.39	1.11	36.29
P5	2000	0.94	5.82	5.19	
P5	4000	3.63	18.34	12.97	
P6	500	0.39	0.80	0.54	32.61
P6	1000	1.56	2.33	2.08	331.98
P6	2000	5.92	8.09	7.88	
P6	4000	22.82	31.63	37.39	
P7	500	3.32	2.22	1.97	42.13
P7	1000	33.70	9.22	13.55	484.05
P7	2000	?	37.91	92.89	
P7	4000	?	161.63		
P8	500	0.29	0.66	2.30	29.45
P8	1000	1.25	2.76	16.61	330.38
P8	2000	5.23	11.54	119.17	
P8	4000	22.68	48.30		
P9	501	0.89	0.57	0.32	25.62
P9	1001	4.30	2.43	1.57	282.77
P9	2001	21.38	9.22	6.98	
P9	4001	91.58	36.80	38.81	

Tabelle 4.1: Ausführungszeiten in Sekunden für $\eta = \epsilon$. Mittelung über drei Läufe. Ergebnisse einzelner Messungen weichen um maximal ein Prozent vom Mittelwert ab.

und Singulärvektoren verwendet (und insbesondere keine hohen relativen Genauigkeiten für die Eigen- und Singulärwerte gefordert), so läßt sich folgendes Splitting-Kriterium anwenden: In den LAPACK-Routinen `DBDSDC` und `DSTEGR` wird zunächst das betragsgrößte Element $b_{\max}$ von B bzw. $t_{\max}$ von $T = B^T B$ bestimmt. Danach werden alle Nebendiagonalelemente mit $b_i < \epsilon b_{\max}$ bzw. $a_i b_i < \epsilon t_{\max}$ auf Null gesetzt. Für die o.a. Testprobleme gilt $b_{\max} \approx t_{\max}$, während gleichzeitig viele $a_i \ll 1$ sind. Bei Abarbeitung der Routine `DSTEGR` greift das auf $T = B^T B$ angewendete Splitting-Kriterium demnach häufiger als bei der Matrix B selbst. Die übrigen Implementierungen bearbeiten also Teilprobleme mit größerer Dimension. An dieser Stelle wird nochmals deutlich, daß für die bei der **bSVD** geforderten guten Kopplungen oft erheblicher Mehraufwand nötig ist als beim **tSEP**.
Als leichte Verbesserung gegenüber dem Ansatz aus der Routine `DBDSDC` werden in `DBSVDSolv` wegen des Bezugs zur Golub-Kahan Matrix zusätzlich alle Diagonalelemente mit $a_i < \epsilon b_{\max}$ auf Null gesetzt.

Problem	n	`DSTEGR`	`DBSVDSolv`	`DBDSDC`
P1	500	0.26	0.39	0.68
P1	1000	1.11	1.54	3.27
P1	2000	4.65	6.26	16.52
P1	4000	18.93	26.23	94.82
P3	500	0.45	1.41	0.41
P3	1000	1.80	6.08	1.77
P3	2000	6.06	23.41	7.17
P3	4000	25.09	101.26	37.29
P5	500	0.22	0.46	0.39
P5	1000	1.01	1.78	1.73
P5	2000	3.15	5.48	7.21
P5	4000	14.38	32.15	37.33

Tabelle 4.2: Ausführungszeiten in Sekunden für $\eta = 10^{-4}$.

Um die Effekte des Splittings auszuschließen und damit einen Vergleich der eigentlichen Kernalgorithmen zu erhalten, sind in Tabelle 4.2 die Ausführungszeiten für die Parametereinstellung $\eta = 10^{-4}$ aufgelistet.

Vergleich der RRR-Verfahren

Aufgrund verschiedener Lösungsansätze in Detailfragen fällt der Vergleich zwischen dem neu vorgeschlagenen Verfahren `DBSVDSolv` gegenüber dem RRR-basierten Ansatz für das **tSEP** bei den betrachteten Problemklassen

unterschiedlich aus. Da für die **bSVD** etwa die doppelte Datenmenge erzeugt wird, sind Ausführungszeiten $t_{\texttt{DBSVDSolv}} \leq 2t_{\texttt{DSTEGR}}$ positiv zu werten. Wir beachten, daß in der LAPACK-Routine auf Absicherungsmaßnahmen wie die Präkonditionierung (Abschnitt 4.1.2) oder die Nachiteration (Abschnitt 3.2.3) verzichtet wird. Wie in den folgenden Abschnitten demonstriert, werden die so gewonnenen niedrigen Ausführungszeiten aber erkauft durch Einbußen bei der Genauigkeit. Für gewisse Testprobleme kommt es sogar zu Breakdowns bei der Durchführung von `DSTEGR`.
Bei Testmatrizen mit vielen oder großen Singulärwertclustern (etwa P3 ($\eta = 10^{-4}$), P4 und P7 ($\eta = \epsilon$) sowie P8 und P9) liegen die Ausführungszeiten der Routine `DBSVDSolv` deutlich höher als bei weitgehend isolierten Spektren. Die Abarbeitung der höheren Ebenen im Darstellungsbaum erfordert das Bestimmen einer großen Anzahl neuer Eigenwertapproximationen. Bei den in Abschnitt 4.3.4 aufgeführten detaillierten Messungen wird deutlich, daß die Suche nach diesen Näherungen zeitaufwendig ist. Zur weiteren Verbesserung von `DBSVDSolv` sind Optimierungen dieser Teilaufgabe beispielsweise durch effiziente Parallelisierung der wichtigste Ansatzpunkt.
Bei Testproblemen mit inneren Clustern wie etwa P7 greift die Routine `DSTEGR` intern auf Inverse Iteration mit expliziter Reorthogonalisierung zurück. Die dann kubische Komplexität wird beim Übergang von $n = 500$ auf $n = 1000$ deutlich. Bei höheren Matrixdimensionen terminiert die aktuelle Version nicht (Breakdown).
Für Matrizen mit Wilkinson-Struktur (P9) erzeugt die aktuelle Version von `DSTEGR` fehlerhafte Ergebnisse. So wird beispielsweise für W_{69}^+ eine Näherung $\bar{Q}$ mit $\bar{Q}(:,43)^T\bar{Q}(:42) = \mathcal{O}(1)$ erzeugt. Überdies enthalten einige Lösungen für die in Tabelle 4.1 getesteten Matrizen ungültige Einträge (`NaN`), ohne daß dies durch eine Fehlermeldung angezeigt wird.

Genauigkeit, Orthogonalität

Zur Einschätzung der numerischen Genauigkeit der vier Verfahren sind für $n = 1000$ und alle neun Testprobleme ($\eta = \epsilon$) die Abweichung von der Orthogonalität sowie die Residuen dargestellt. In Abbildung 4.8 sind für `DSTEGR` die Größen

$$\log_{10}\left(\frac{\max_{i,j}\{\bar{Q}(:,i)^T\bar{Q}(:,j)\}}{n\epsilon}\right)$$

und für die **bSVD**-Lösungsverfahren

$$\log_{10}\left(\frac{\max_{i,j}\{\bar{U}(:,i)^T\bar{U}(:,j), \bar{V}(:,i)^T\bar{V}(:,j)\}}{n\epsilon}\right)$$

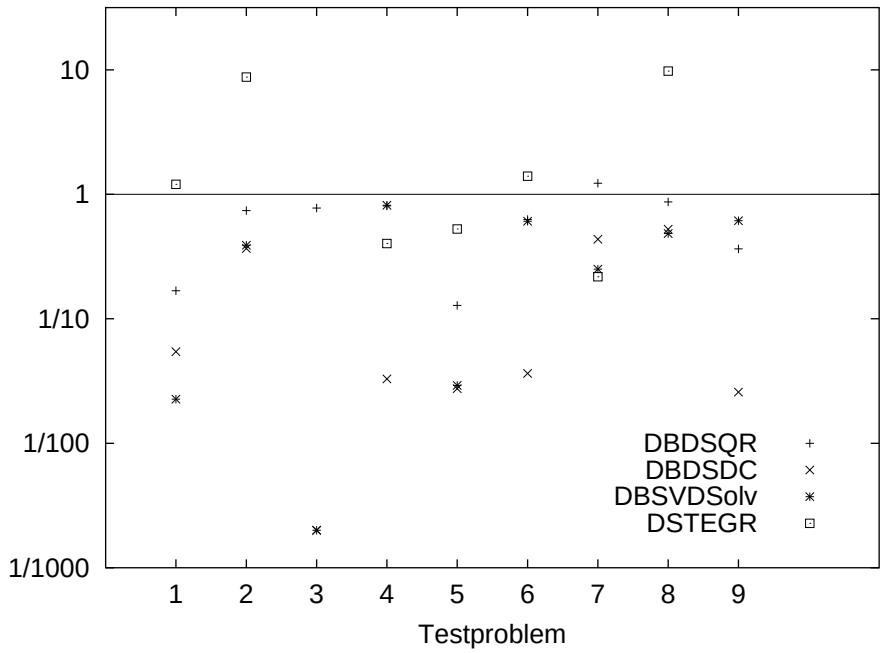


Abbildung 4.8: Skalierte Abweichung von Orthogonalität.

aufgetragen. Gute numerische Orthogonalität ist gegeben, wenn die Ergebnisse kleiner als eins sind. In der Abbildung ist diese Schranke mit der durchgezogenen Linie angedeutet. Nicht erfaßt sind Approximationen, die mehr als einen Faktor 10^{-3} unterhalb der geforderten Genauigkeit liegen. Dies ist insbesondere für Testprobleme wie P3 der Fall, bei denen bereits Einheitsmatrizen gute Approximationen an Eigen- und Singulärvektorbasen sind. Diese Eigenschaft wird von der Routine `DSTEGR` auffällig gut wiedergegeben.

In den meisten anderen Fällen werden die Anforderungen an die numerische Orthogonalität von den übrigen drei Verfahren deutlich besser erfüllt. Bei mehreren Testproblemen überschreitet das RRR-basierte Verfahren für das **tSEP** außerdem die Akzeptanzgrenze.

Genauigkeit, Residuen

Abbildung 4.9 gibt die Größen

$$\log_{10}\left(\frac{\max_j\{||T\bar{Q}(:,j)-\bar{\lambda}_j\bar{Q}(:,j)||\}}{n\epsilon|\lambda_n|}\right)$$

und

$$\log_{10}\left(\frac{\max_j\{||B\bar{V}(:,j)-\bar{\sigma}_j\bar{U}(:,j)||\}}{n\epsilon\sigma_n}\right)$$

wieder. Es zeigt sich, daß die in Kapitel 3 beschriebenen Kopplungsmechanismen zu zufriedenstellenden Ergebnissen führen.

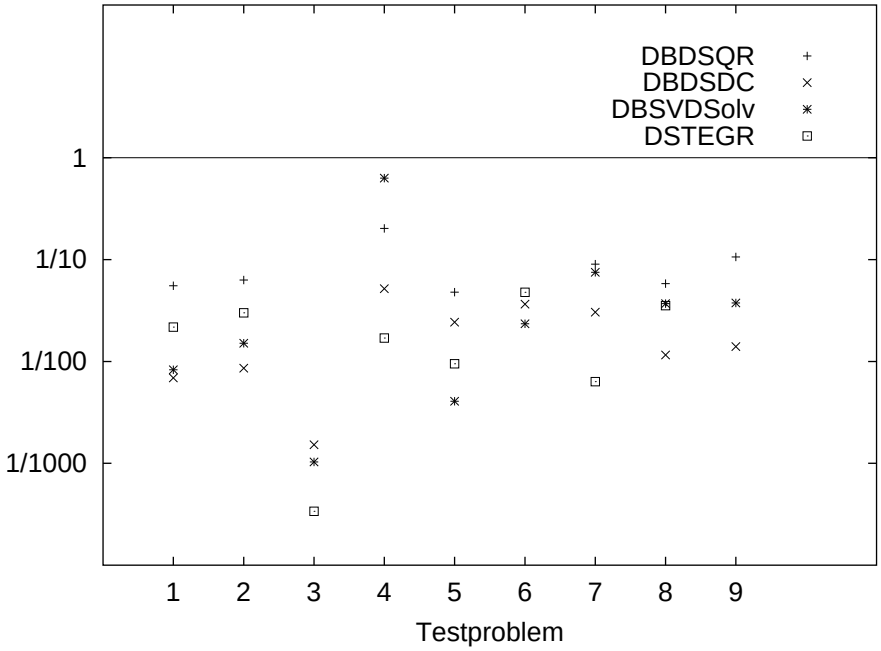


Abbildung 4.9: Skalierte Residuen.

4.3.4 Details zur Routine `DBSVDSolv`

Für die Beurteilung der eingeführten Konzepte zu Kopplungen, Präkonditionierung und idealen Systemen sind in den nachfolgenden Tabellen die Anteile der jeweiligen Maßnahmen an der Gesamtausführungszeit aufgeschlüsselt.

Gliederung in Teilaufgaben

Die Initialisierung sowie das Anwenden der Splitting-Kriterien und des QR-Verfahrens bei kleinen Teilmatrizen sind als erste Aufgabe der *Vorverarbeitungsphase* zusammengefaßt. Hinzu kommen die Anteile für das Flipping sowie der statischen und dynamischen Präkonditionierung. Die Bestimmung

der *Singulärwerte* mit dem QD-Verfahren schließt sich an, bevor der eigentliche Kernalgorithmus gestartet wird. Auf der *ersten Stufe* erfolgt die Erzeugung der Vektoren zu isolierten Singulärwerten (explizite Faktorisierung, Aufstellen der gekoppelten Darstellung, Lösen der Gleichungssysteme, Nachiteration) sowie die Konstruktion idealer Systeme. Auch die Arbeit auf den *höheren Stufen* ist in verschiedene Teile gegliedert, um den zusätzlichen Aufwand bei der Einbettung von Kopplungen einschätzen zu können.
Während die jeweils linken Spalten Prozentzahlen wiedergeben, ist rechts die Menge der verarbeiteten Daten eingetragen. Bei *isolierten Vektoren* entspricht dies der Anzahl der isolierten Singulärwerte von B (ohne Splitting). Zu den auf der *ersten Stufe* erfolgreich behandelten Singulärvektorpaaren zählen außerdem die durch die erfolgreiche Konstruktion *idealer Systeme* generierten Vektoren. Daneben sind auch die mit *idealen Systemen* auf höheren Stufen gewonnenen Vektoren aufgeführt, während die Zahl in der letzten Zeile angibt, wieviele Eigenwertnäherungen zu ermitteln sind. Da die Matrix B von vorneherein in kleinere Probleme zerlegt werden und der zugehörige Darstellungsbaum mehrere Stufen haben kann, müssen sich die Werte nicht unbedingt zu n addieren.
Pro Doppelspalte ist das Verhalten für eine bestimmte Testmatrix aufgezeichnet. Bei den Problemen P1 bis P9 betrachten wir die jeweilige Vertreterin mit $n = 2000$ und $\eta = \epsilon$.

Vorverarbeitung

Der *absolute* Anteil der statischen Präkonditionierung ($p_s = 1$) ist bei gleichbleibender Matrixdimension etwa konstant. Wie in Abschnitt 4.1 beschrieben verbessert die Anwendung von Glättungsschritten die Genauigkeit des Kernalgorithmus bei moderatem zusätzlichen Aufwand.
Beim Testproblem P7 (a) findet die dynamische Präkonditionierung Anwendung. Wir untersuchen dazu die aus der geleimten Matrix $T(100, 9, \sqrt{\epsilon}) \in \mathbb{R}^{1000\times 1000}$ entstehende Bidiagonale. Als Basismatrix dient die Repräsentantin von P7 (interner Cluster) mit $\eta = \sqrt{\epsilon}$. Mit der gegebenen Clusterstruktur wächst bei B_{p_s} für hundert Singulärwerte der Darstellungsbaum über die Stufe recmax = 8 hinaus. Es erfolgt dann ein weiterer Schritt des QR-Verfahrens. Mit der neuen Matrix B_{p_s+1} gelingt die Bestimmung der restlichen Singulärvektorpaare, die adaptiv aufdatiert werden. Die dynamische Präkonditionierung selbst hat also etwa ein Zehntel des Aufwands der statischen Vorverarbeitung. Hinzu kommt natürlich noch die vergeblich investierte Arbeit für die hundert Vektoren bei der Vorgängermatrix B_{p_s}.
Der Ansatz, eine Eingangsmatrix zu *flippen*, begünstigt ebenfalls die Arbeit des Kernalgorithmus. Durch die anschließend erforderliche Permutation der

	P1		P3		P3 (a)		P4		P4 (a)	
Gesamt	**5.30**		**0.30**		**22.45**		**21.39**		**7.02**	
Init, Splitting, QR	5.6%		99.1%		1.3%		1.4%		1.1%	
Flipping	0.0%		0.0%		0.0%		0.0%		0.0%	
statische Präk.	12.3%		0.0%		3.2%		3.1%		2.3%	
dynamische Präk.	0.0%		0.0%		0.0%		0.0%		0.0%	
Vorverarbeitung	**17.9**%		**99.1**%		**4.5**%		**4.5**%		**3.4**%	
Singulärwerte	**12.3**%		**0.0**%		**11.3**%		**11.0**%		**6.7**%	
isolierte Vektoren		1946		0		1		1		0
Faktorisierung	30.7%		0.0%		0.0%		0.0%		0.0%	
Lösung v	4.6%		0.0%		0.0%		0.0%		0.0%	
Nachiteration v	3.4%		0.0%		0.0%		0.0%		0.0%	
Kopplung	21.9%		0.0%		0.0%		0.0%		0.0%	
Lösung u	4.5%		0.0%		0.0%		0.0%		0.0%	
Nachiteration u	3.4%		0.0%		0.0%		0.0%		0.0%	
ideale Systeme	0.0%	0	0.0%	0	0.0%	0	0.0%	0	0.1%	10
erste Stufe	**69.7**%	1946	**0.0**%	0	**0.0**%	1	**0.0**%	1	**0.1**%	10
Reproduktion	0.0%		0.0%		0.0%		0.0%		0.2%	
ideale Systeme	0.0%	0	0.0%	0	15.1%	1183	15.1%	1139	23.4%	968
Bestimmung RRR	0.0%		0.0%		0.5%		0.7%		0.4%	
Kopplung	0.0%		0.0%		0.4%		0.4%		0.2%	
neue Eigenwerte	0.0%		0.0%		44.9%		43.6%		42.1%	
Vektoren v	0.0%		0.0%		11.4%		11.9%		11.8%	
Vektoren u	0.0%		0.0%		11.6%		12.3%		11.6%	
höhere Stufen	**0.0**%	0	**0.0**%	0	**83.8**%	2170	**84.0**%	2166	**89.6**%	1269

	P5		P5 (a)		P6		P6 (a)		P6 (b)	
Gesamt	**5.75**		**1.52**		**8.18**		**1.93**		**47.51**	
Init, Splitting, QR	5.8%		4.9%		3.6%		3.9%		0.2%	
Flipping	37.4%		46.3%		0.0%		0.0%		0.0%	
statische Präk.	8.3%		11.1%		8.2%		8.4%		0.3%	
dynamische Präk.	0.0%		0.0%		0.0%		0.0%		0.0%	
Vorverarbeitung	**51.5**%		**62.3**%		**11.8**%		**12.2**%		**0.5**%	
Singulärwerte	**2.6**%		**5.9**%		**21.2**%		**17.4**%		**0.7**%	
isolierte Vektoren		1536		4		1093		0		0
Faktorisierung	18.1%		0.1%		11.4%		0.0%		0.0%	
Lösung v	2.8%		0.0%		1.7%		0.0%		0.0%	
Nachiteration v	1.6%		0.0%		1.1%		0.0%		0.0%	
Kopplung	15.1%		0.1%		8.9%		0.0%		0.0%	
Lösung u	2.7%		0.0%		1.7%		0.0%		0.0%	
Nachiteration u	1.6%		0.0%		1.1%		0.0%		0.0%	
ideale Systeme	2.0%	60	31.3%	996	23.4%	788	24.3%	920	0.0%	0
erste Stufe	**45.2**%	1596	**31.6**%	1000	**49.8**%	1881	**24.3**%	920	**0.0**%	0
Reproduktion	0.0%		0.0%		0.0%		0.0%		0.0%	
ideale Systeme	0.1%	2	0.0%	0	1.0%	0	6.2%	70	0.0%	0
Bestimmung RRR	0.0%		0.0%		1.2%		0.5%		1.1%	
Kopplung	0.0%		0.0%		0.4%		0.2%		0.5%	
neue Eigenwerte	0.2%		0.0%		7.1%		24.3%		71.8%	
Vektoren v	0.1%		0.0%		3.4%		7.5%		12.0%	
Vektoren u	0.1%		0.0%		3.8%		7.2%		13.0%	
höhere Stufen	**0.5**%	2	**0.0**%	0	**16.9**%	127	**45.8**%	98	**98.4**%	3693

	P7		P7 (a)		P8		P9		P9 (a)	
Gesamt	**37.05**		**10.42**		**11.67**		**9.33**		**50.65**	
Init, Splitting, QR	0.8%		0.7%		2.5%		3.2%		0.6%	
Flipping	0.0%		0.0%		0.0%		0.0%		0.0%	
statische Präk.	1.9%		1.5%		5.8%		7.4%		1.4%	
dynamische Präk.	0.0%		0.2%		0.0%		0.0%		0.0%	
Vorverarbeitung	**2.7**%		**2.4**%		**8.3**%		**10.6**%		**1.9**%	
Singulärwerte	**4.5**%		**1.9**%		**13.2**%		**42.9**%		**7.9**%	
isolierte Vektoren		2		0		1461		9		9
Faktorisierung	0.0%		0.0%		10.6%		0.1%		0.0%	
Lösung v	0.0%		0.0%		3.3%		0.0%		0.0%	
Nachiteration v	0.0%		0.0%		5.6%		0.0%		0.0%	
Kopplung	0.0%		0.0%		7.7%		0.1%		0.0%	
Lösung u	0.0%		0.0%		3.3%		0.0%		0.0%	
Nachiteration u	0.0%		0.0%		5.6%		0.0%		0.0%	
ideale Systeme	0.0%	0	0.0%	0	0.0%	0	44.2%	1980	0.0%	0
erste Stufe	**0.0**%	2	**0.0**%	0	**36.4**%	1461	**44.4**%	1989	**0.0**%	9
Reproduktion	0.0%		0.0%		0.0%		0.0%		0.0%	
ideale Systeme	0.0%	0	0.6%	27	0.0%	0	0.3%	0	0.0%	0
Bestimmung RRR	0.0%		0.2%		0.0%		0.2%		10.3%	
Kopplung	0.0%		0.1%		0.0%		0.1%		2.0%	
neue Eigenwerte	45.3%		57.4%		26.5%		0.7%		57.1%	
Vektoren v	23.6%		18.5%		7.8%		0.3%		9.5%	
Vektoren u	23.6%		18.6%		7.6%		0.3%		10.4%	
höhere Stufen	**92.5**%	2582	**95.3**%	1214	**42.0**%	539	**1.9**%	12	**89.3**%	2224

Matrizen U und V entsteht aber ein erheblicher Mehraufwand an Speicheroperationen, vgl. Probleme P5 und P5 (a).
Zu den Auswirkungen der Splitting-Kriterien vergleichen wir P3 ($n = 2000$, $\eta = \epsilon$) mit P3 (a) ($n = 2000, \eta = 10^{-4}$). Während die erste Matrix in sehr kleine Teile zerlegt werden kann, kommt bei der Zweiten der Kernalgorithmus zum Einsatz. Der Darstellungsbaum hat dann eine ähnliche Gestalt wie bei den zu P7 verwandten Testproblemen.

Einbettung der Kopplungen

In Abschnitt 3.2 wird der durch die Einbettung der Kopplungen theoretisch zu erwartende zusätzliche Arbeitsbedarf bei isolierten Singulärwerten diskutiert. Zu den praktischen Auswirkungen ist für die erste Stufe das Folgende festzuhalten (siehe Algorithmus 3.4): Der Aufwand für das explizite *Faktorisieren* von $B^TB - \mu^2 I$ ist höher als der für die Umrechnung zu der *gekoppelten Darstellung* von $BB^T - \mu^2 I$. Das *Lösen* der durch die entsprechenden BABE-Faktorisierungen gegebenen Gleichungssysteme benötigt deutlich weniger Zeit, insbesondere wenn hier das Verfahren zur Trägerminimierung angewendet wird. Obwohl die *Nachiteration* den Kernalgorithmus verlangsamt, sollte sie zur Absicherung der Genauigkeit verwendet werden.
Wir betrachten die in der Routine `DSTEGR` verwendeten Gershgorin-Abschätzungen zur *a priori* Bestimmung einer geeigneten Twist-Position als vielversprechenden Ansatzpunkt, um den Aufwand für die Faktorisierung und die Kopplung weiter zu reduzieren.
Bei der Arbeitsverteilung auf den *höheren Stufen* fällt ein deutliches Übergewicht für die Aufgaben der Eigenwert- und Singulärvektorbestimmung auf, vgl. Abschnitt 3.4.6. Für die *Depth-First-Strategie* (Tiefensuche) hat die Problematik, u.U. Faktorisierungen *reproduzieren* zu müssen, kaum Einfluß auf den Aufwand. Ebenso verhält es sich mit der *Ermittlung einer RRR* und der zugehörigen *Kopplung*, wenngleich die dort gewählten Shift-Parameter entscheidende Auswirkungen auf die Entwicklung des Darstellungsbaumes haben. Hier ist noch Spielraum für Verbesserungen der *a priori* Kriterien gegeben, vgl. Abschnitt 2.4.
Die *Bestimmung von Eigenwertnäherungen* stellt i.d.R. das arbeitsintensivste Teilproblem dar. Dabei ist durch die Verwendung der abgesicherten Rayleigh-Quotienten-Iteration (Algorithmus 3.11) die Bestimmung der zur expliziten Faktorisierung gehörenden Eigenwerte $\bar{\hat{\lambda}}_j^+$ eng mit dem Aufstellen der *Singulärvektornäherungen* $\bar{V}(:, j)$ verzahnt. Die drittletzte Zeile umfaßt ferner die Überprüfung der *a posteriori* Kriterien für die RRR.
Bei bekanntem $\bar{\hat{\lambda}}_j^+$ läßt sich der Eigenwert der gekoppelten Matrix im Nor-

malfall durch ein bis zwei Schritte des RQI-Verfahrens approximieren. Die Arbeit für die Bestimmung der rechten *Singulärvektornäherung* $\bar{U}(:, j)$ beinhaltet also gleichzeitig schon die Bestimmung des gekoppelten Eigenwertes. Die Einbettung der Kopplungen auf höheren Stufen erfolgt damit bei vielen Testproblemen nur auf Kosten des in der vorletzten Zeile angegebenen Mehraufwands.
Zur Verbesserung der Ausführungszeiten bleibt eine weitere Optimierung des Eigenwertlösungsverfahrens das wichtigste Ziel. Durch die Konstruktion idealer Systeme läßt sich jedoch die Anzahl der zu bestimmenden Eigenwerte signifikant reduzieren.

Ideale Systeme

Für gezielt konstruierte geleimte Matrizen können wir das Verhalten der Routine `DBSVDSolv` in Extremfällen beobachten.

- P4 (a): $T(100, 9, 200\epsilon)$, Basismatrix aus P4 mit $\eta = \epsilon$.
- P5 (a): $T(100, 9, \sqrt{\epsilon})$, Basismatrix aus P5 mit $\eta = \sqrt{\epsilon}$.
- P6 (a),(b): $T(100, 9, \sqrt{\epsilon})$, Basismatrix aus P6 mit $\eta = \sqrt{\epsilon}$.

Diese Testprobleme haben wie die Wilkinson-Matrizen (P9) die gemeinsame Eigenschaft, daß sie über eine große Anzahl von Clustern verfügen, die erst auf höheren Stufen des Darstellungsbaumes abgearbeitet werden können. Diese Schwierigkeit kann aber durch ideale Systeme umgangen werden. In den Tabellen unterscheiden sich die Spalten P6 (a) und P6 (b) bzw. P9 und P9 (a) darin, daß in der einen Variante das Konstruktionsverfahren für alle Cluster mit $s = l - f + 1 \leq 20$ versucht und in der anderen grundsätzlich unterbunden wird. Es stellt sich heraus, daß bereits auf der ersten Ebene weit über 90% der Vektoren erledigt werden können. Ohne ideale Systeme müssen weitaus aufwendigere Arbeitsschritte durchgeführt werden. Dies führt zu frappanten Erhöhungen der Ausführungszeiten.
Die Testprobleme P4 (a) und P5 (a) sind noch weitaus ungünstiger gewählt. Für P4 (a) liegen extrem enge Cluster vor: Auch auf höheren Stufen stimmen die Eigenwertnäherungen in allen signifikaten Stellen überein. Bei P5 (a) führen enge Cluster und eine hohe Konditionszahl zu häufigen Breakdowns bei Faktorisierungen und zu Problemen, überhaupt geeignete relativ robuste Repräsentationen (RRR) zu finden. Ohne den Einsatz idealer Systeme würde die Routine `DBSVDSolv` auch nach $p_d = 10$ dynamischen Präkonditionierungsschritten keine vollständige Singulärvektorbasis erzeugen können, vgl. Abschnitt 4.1.3. Beim Ablauf von P4 (a) zeigt sich außerdem,

daß sich die Investition in die Konstruktion idealer Systeme auch auf höheren Ebenen lohnt, da die Cluster über Substrukturen verfügen, die erst nach mehrmaliger Anwendung der Shift-Strategie isoliert werden können.

Verhalten der übrigen Lösungsverfahren bei geleimten Matrizen

Während das QR-Verfahren und der Divide-and-Conquer Ansatz auch für die geleimten Testprobleme korrekte Ergebnisse liefern, treten bei der aktuellen Implementierung von `DSTEGR` Schwierigkeiten auf. Selbst für „einfache" Matrizen wie $T(32, 1, \epsilon)$ zur P8-Basismatrix terminiert die Routine nicht. Es sei jedoch darauf verwiesen, daß die aktuelle LAPACK-Routine nur eine vorläufige Implementierung des RRR-Algorithmus darstellt und noch einen ständigen Verbesserungsprozeß durchläuft. Trotzdem macht aufgrund des kritischen Verhaltens von `DSTEGR` für Matrixdimensionen ≥ 1000 nur noch ein Vergleich mit der Routine `DBDSDC` Sinn.

	n	k	`DBSVDSolv`	`DBDSDC`	n	k	`DBSVDSolv`	`DBDSDC`
P4	1000	0	5.41	1.87	100	9	7.02	1.71
	2000	0	21.13	8.32	100	19	22.76	7.75
					200	9	22.95	7.85
P5	1000	0	1.39	1.11	100	9	1.52	0.61
	2000	0	5.82	5.19	100	19	6.54	2.82
					200	9	9.69	7.59
P6	1000	0	2.33	2.08	100	9	1.93	1.95
	2000	0	8.09	7.88	100	19	4.06	8.53
					200	9	8.76	8.69
P7	1000	0	9.22	13.55	10	99	10.42	10.61
	2000	0	37.91	92.89	10	199	43.70	38.61
					20	99	55.89	75.22

Tabelle 4.3: Ausführungszeiten für einige geleimte Matrizen. Im rechten Abschnitt bezeichnet n die Dimension der Basismatrix.

In Tabelle 4.3 werden die Ausführungszeiten für Standardmatrizen denen für geleimte Matrizen gleicher Dimension gegenübergestellt. Es wird wiederum deutlich, daß das Verhalten stark von der Struktur des Spektrums abhängig ist. Es bestätigt sich, daß der Divide-and-Conquer Ansatz seine Stärken bei engen Clustern besonders gut ausspielen kann, während der RRR-basierte Ansatz vertretbare Ausführungszeiten vor allem durch erfolgreiche Konstruktion idealer Systeme erzielt.

4.3.5 Zusammenfassung

Das QR-Verfahren erweist sich bei allen Testproblemen als stabile Methode, die sowohl bei den Ausführungszeiten als auch bei der Qualität der Ergebnisse kaum Abhängigkeiten vom Spektrum aufweist.
Die übrigen drei Verfahren sind durchweg schneller, zeigen aber bei unterschiedlichen Testproblemen variierendes Verhalten. So kann der Divide-and-Conquer Ansatz aus Matrizen mit engen Clustern Vorteile ziehen, während die RRR-basierten Methoden dort vergleichsweise hohe Ausführungszeiten haben. Dennoch erweist sich mit wachsender Dimension die quadratische Komplexität als überlegen.
Aufgrund verschiedener Gewichtungen der Lösungsstrategien weisen die Routinen `DSTEGR` und `DBSVDSolv` starke Unterschiede bzgl. der erzielbaren Genauigkeit und der numerischen Zuverlässigkeit auf. Für eine endgültige Implementierung mit Bibliotheksstandard sollten die jeweils besseren Eigenschaften kombiniert werden.
Die Einbettung von Kopplungen führt weder zu großem algorithmischen Aufwand noch zu einer übermäßigen Steigerung der Rechenzeit, da viele für den klassischen RRR-Algorithmus ohnehin benötigte Informationen geschickt ausgenutzt werden können.
Als weiteres Ergebnis halten wir fest, daß der so entstandene Kernalgorithmus alleine bei gezielt konstruierten Testproblemen auf Schwierigkeiten stößt. Die zu deren Abfederung eingeführten Vorschläge zur Präkonditionierung und Konstruktion idealer Systeme erweisen sich bei moderatem zusätzlichen Aufwand als zugkräftig.
Wir sehen die *Kombination* des Kernalgorithmus mit der Konstruktion idealer Systeme als ein Verfahren an, das sowohl das **tSEP** als auch die **bSVD** mit quadratischem Aufwand bestimmen kann.

Kapitel 5

Parallelisierung

Bei der Übertragung auf parallele Rechnerarchitekturen sind zwei Aspekte zu diskutieren: Zum einen schlagen wir Strategien vor, die sich auf den Kernalgorithmus selbst beziehen. Auf der anderen Seite ist zu untersuchen, welche der übrigen Bestandteile des gesamten Lösungsverfahrens (Präkonditionierung, ideale Systeme) mit vertretbarem Aufwand in eine parallele Version der Routine `DBSVDSolv` eingehen können. Nach einer kurzen Beschreibung der Entwicklungsumgebung erörtern wir zunächst das Vorgehen zur Parallelisierung der den Kernalgorithmus umgebenden Arbeitsschritte, bevor dieser selbst untersucht wird. Numerische Experimente zur Beurteilung der Skalierungseigenschaften runden das Kapitel ab.

5.1 Einführung

Das Programmiermodell

Die ScaLAPACK-Bibliothek (Scalable Linear Algebra PACKage [10, 67]) ist eine Zusammenfassung von Algorithmen der numerischen linearen Algebra für Parallelrechner mit verteiltem Speicher. Neben den LAPACK- und BLAS-Routinen greift sie auf die Bausteine PBLAS und BLACS (Basic Linear Algebra Communication Subprograms [32]) zurück. Die BLACS stellen eine Schnittstelle zu den unterliegenden Message-Passing-Systemen wie MPI oder PVM dar. Die PBLAS stehen als parallele Variante der BLAS zur einfachen Abwicklung von Matrix-Matrix-Operationen bereit.

Numerische Lösungsverfahren können damit plattformunabhängig entwickelt werden, während auf der Herstellerseite die Möglichkeit besteht, die BLAS- und BLACS-Bibliotheken auf die jeweilige Hardware hin zu optimieren.

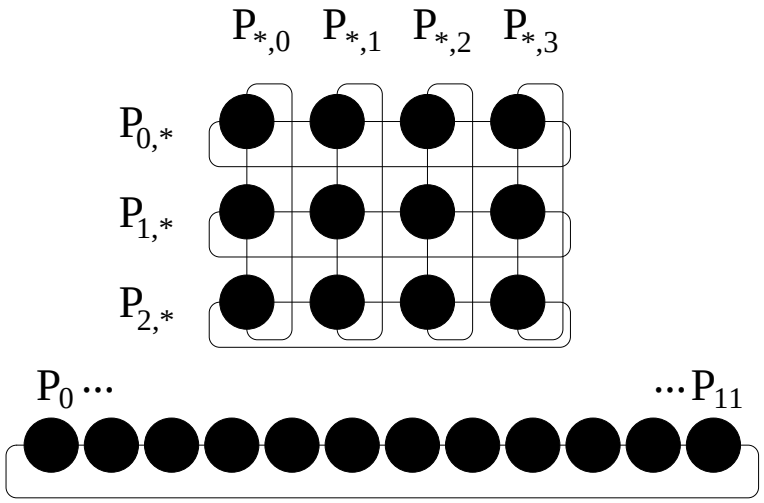


Abbildung 5.1: Darstellung eines Prozessorgitters ($nprow = 3$, $npcol = 4$) und eines Ringes ($nprocs = 12$). Die Kreise symbolisieren die Prozessoren, während die durchgezogenen Linien die (logischen) Verbindungen im orthogonalen Gitter veranschaulichen [47].

Die BLACS unterstützen die Aufteilung der Prozessoren in orthogonale Gitter mit *nprow* Zeilen und *npcol* Spalten, vgl. Abbildung 5.1. Eine Matrix A wird in rechteckigen Blöcken der Größe $m_b \times n_b$ über die Knoten verteilt. Am rechten und am unteren Rand können kleinere Formate entstehen. Der Prozessor $P_{k,l}$ verwaltet die Blöcke

$$A_{ij} = A(i \cdot m_b + 1 : i \cdot m_b + m_b, j \cdot n_b + 1 : j \cdot n_b + n_b)$$

mit $i \equiv k \bmod nprow$ und $j \equiv l \bmod npcol$ (*blockzyklisch-zweidimensionales Layout*, vgl. Abbildung 5.2).
Durch Auswahl der Parameter *nprow*, *npcol*, m_b und n_b können wir das Kommunikationsverhalten und damit die Skalierungseigenschaften einer Anwendung beeinflussen. Je nach Implementierung der BLAS- und BLACS-Routinen ergeben sich unterschiedliche Werte für eine optimale Leistung des Parallelrechners.

Hardware

Die hier vorgestellten Ergebnisse resultieren aus Experimenten mit einem Workstation-Cluster. Dabei sind 128 Compaq DS10 Einheiten mit einem

	$P_{*,0}$	$P_{*,1}$	$P_{*,0}$	$P_{*,1}$
$P_{0,*}$	A_{00}	A_{01}	A_{02}	A_{03}
$P_{1,*}$	A_{10}	A_{11}	A_{12}	A_{13}
$P_{0,*}$	A_{20}	A_{21}	A_{22}	A_{23}
$P_{1,*}$	A_{30}	A_{31}	A_{32}	A_{33}
$P_{0,*}$	A_{40}	A_{41}	A_{42}	A_{43}

	$P_{*,0}$		$P_{*,1}$	
$P_{0,*}$	A_{00}	A_{02}	A_{01}	A_{03}
$P_{1,*}$	A_{10}	A_{12}	A_{11}	A_{13}
$P_{0,*}$	A_{20}	A_{22}	A_{21}	A_{23}
$P_{1,*}$	A_{30}	A_{32}	A_{31}	A_{33}
$P_{0,*}$	A_{40}	A_{42}	A_{41}	A_{43}

	$P_{*,0}$	$P_{*,1}$	$P_{*,0}$	$P_{*,1}$
$P_{0,*}$	A_{00}	A_{01}	A_{02}	A_{03}
$P_{0,*}$	A_{20}	A_{21}	A_{22}	A_{23}
$P_{0,*}$	A_{40}	A_{41}	A_{42}	A_{43}
$P_{1,*}$	A_{10}	A_{11}	A_{12}	A_{13}
$P_{1,*}$	A_{30}	A_{31}	A_{32}	A_{33}

	$P_{*,0}$		$P_{*,1}$	
$P_{0,*}$	A_{00}	A_{02}	A_{01}	A_{03}
$P_{0,*}$	A_{20}	A_{22}	A_{21}	A_{23}
$P_{0,*}$	A_{40}	A_{42}	A_{41}	A_{43}
$P_{1,*}$	A_{10}	A_{12}	A_{11}	A_{13}
$P_{1,*}$	A_{30}	A_{32}	A_{31}	A_{33}

Abbildung 5.2: Verteilung einer Matrix in fünf Zeilen- und vier Spaltenblöcke über ein 2×2-Prozessorgitter. Links oben die globale Ansicht, rechts oben die Verteilung innerhalb der Prozessorpalten, links unten die Verteilung innerhalb der Prozessorzeilen, rechts unten lokale Ansicht der Daten [47].

Myrinet-Netzwerk zusammengeschaltet. Die Alpha 21264 Prozessoren haben eine Frequenz von 616 MHz (Peak-Performance 1233 MFlops). Jeder Knoten verfügt über 256 MB Hauptspeicher. Rechenintensive mathematische Teilaufgaben — z.B. alle Aufrufe der BLAS- und LAPACK-Routinen — werden an die Compaq Extended Math Library (CXML) weitergeleitet, während die Kommunikation über ParaStation-Software [62] abgewickelt wird.
Tabelle 5.1 vergleicht die Ausführungszeiten der Routine `DBSVDSolv` auf der SUN Enterprise 450 und einem Knoten des Workstation-Clusters. Es stellt sich heraus, daß die Alpha Prozessoren für fast alle Testprobleme weitaus niedrigere Ausführungszeiten benötigen. Insbesondere zeigt sich indirekt, daß sie näher an der theoretisch erreichbaren Peak-Performance liegen als die UltraSPARC II Rechner: Bei etwa doppelter Taktrate benötigt ein Alpha-Prozessor meist deutlich weniger als die Hälfte der Ausführungszeit, die auf der SUN erzielt werden kann.

Implementierungen

In der ScaLAPACK-Bibliothek liegt eine parallele Version des QR-Verfahrens eingebettet in der Routine `PDGESVD` vor. Dort wird eine vollbesetzte Ausgangsmatrix A zunächst auf Bidiagonalform reduziert und diese dann wie beschrieben diagonalisiert. Da das QR-Verfahren inhärent seriell ist, sind in

	SUN (300MHz)	Alpha (616 MHz)
P1	5.2	1.7
P2	7.2	2.4
P3	0.3	0.8
P4	21.1	9.9
P5	5.8	2.7
P6	8.1	2.7
P7	37.9	15.1
P8	11.5	4.1
P9	9.2	2.9

Tabelle 5.1: Vergleich der seriellen Ausführungszeiten in Sekunden für $n = 2000$ und $\eta = \epsilon$. Mittelung über drei Läufe.

der Routine nur die Aufdatierungen mit den geblockten Givensrotationen für verteilte Rechnerarchitekturen angelegt.
Die Routine `PDBSVDSolv` beinhaltet eine parallele Implementierung des in den vorigen Kapiteln eingeführten **bSVD**-Lösungsverfahrens. Die Parallelisierungsstrategien werden in den nachfolgenden Abschnitten genauer beschrieben. Ein Äquivalent zu dem RRR-basierten Ansatz der Routine `DSTEGR` liegt in der ScaLAPACK-Bibliothek nicht vor.

5.2 Parallelisierung des Gesamtverfahrens

Für die wichtigsten Teilaufgaben des **bSVD**-Lösungsverfahrens erörtern wir ihr Potential für die Arbeit auf verteilten Rechnerarchitekturen. Dem eigentlichen Kernalgorithmus ist dabei erst der folgende Abschnitt gewidmet, während die Parallelisierung der umrahmenden Schritte direkt im Anschluß erfolgt.
Es stellt sich hier heraus, daß sehr gute Lastverteilung und damit hohe Skalierbarkeit oft nur mit Algorithmen erzielt werden können, die auch bei größeren Prozessorzahlen deutlich höhere absolute Ausführungszeiten besitzen als die mit inhärent seriellen Anteilen belasteten Alternativen. Beispielsweise ist das Bisektionsverfahren auch auf größeren Prozessorzahlen immer noch langsamer als das serielle QD-Verfahren, vgl. Tabelle 5.2 auf Seite 161. Die Konstruktion idealer Systeme gehört ebenfalls nicht zu den effizient verteilbaren Aufgaben, erspart aber das weitaus aufwendigere Abarbeiten von höheren Ebenen im Darstellungsbaum.
Die Programmierung der Routine `PDBSVDSolv` ist so gestaltet, daß durch

Einstellung von Parametern die Gewichtung zwischen hoher Skalierbarkeit und dem Einsatz serialisierender Lösungsverfahren in Abhängigkeit von der Rechnerkonfiguration vorgenommen werden kann.

5.2.1 Verteilung der Daten, Kommunikation

Die gesuchten Singulärvektorpaare werden durch das Lösen eines Gleichungssystems bestimmt, dessen Matrix durch eine BABE-Faktorisierung gegeben ist, vgl. Abschnitt 2.2. Bezogen auf das blockzyklisch-zweidimensionale Layout der ScaLAPACK-Bibliothek ist daher die Verwendung eines Prozessorgitters G_{Diag} mit einer Zeile und *nprocs* Spalten naheliegend. Wir bezeichnen dessen Knoten mit $P_0, \ldots, P_{nprocs-1}$. Jeder Prozessor speichert also etwa $n/nprocs$ vollständige Singulärvektorpaare. Gemäß der Sortierung der Singulärwerte sind diese block-spaltenzyklisch verteilt.
Ist eine Bidiagonalisierung (z.B. mit der Routine `PDGEBRD`, vgl. Abschnitt 1.4.4) vorgeschaltet, so ist n_b i.d.R. abhängig von der Optimierung der BLAS- und BLACS-Bibliotheken vorgegeben. Auch die Zeilen sind typischerweise geblockt verteilt. Die Rück-Akkumulation wird ebenfalls auf dem Gitter G_{Bid} abgewickelt. Beim Wechseln der Gitter ist dann innerhalb der Prozessorzeilen die Kommunikation großer Datenmengen erforderlich, vgl. Abbildung 5.3. Das Verfahren zur Trägerminimierung kann hier erhebliche Einsparungen ermöglichen, vgl. Abschnit 4.2.

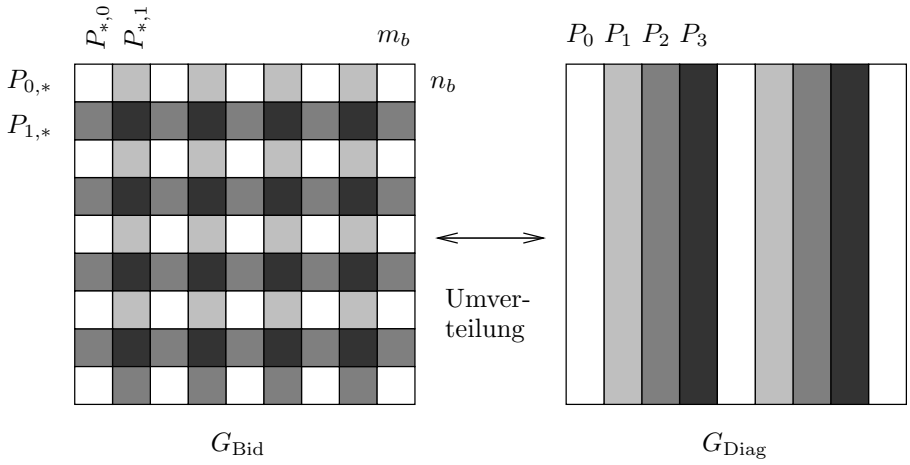


Abbildung 5.3: Layout für vier Prozessoren. Bei der Bidiagonalisierung und Rück-Akkumulation sind die vollbesetzten Matrizen zeilen- und spaltenweise blockzyklisch auf dem Gitter G_{Bid} verteilt. Bei der Diagonalisierung auf G_{Diag} sind nur die Spalten von U und V blockzyklisch angelegt.

Trotz des Aufwandes bei der Umverteilung ist das Gitter G_{Bid} keine Alternative für die Diagonalisierung: Jedes Lösen eines durch eine BABE-Faktorisierung gegebenen LGS wäre mit extrem hohen Kommunikationskosten verbunden.
Zur Lösung der **bSVD** speichert jeder Prozessor die Daten der Bidiagonalmatrix B sowie die daraus abgeleiteten Hilfsgrößen. Geblockte Kommunikation findet an genau spezifizierten Stellen im Programmablauf statt, etwa nach einer verteilten Bestimmung von Eigenwertapproximationen auf höheren Ebenen im Darstellungsbaum. Diese erfolgt entweder durch globale Broadcast- oder Point-to-Point-Operationen. Bei letzteren werden nur sehr kleine Datenmengen kommuniziert. Wir versenden lediglich die Pfade, vgl. Abschnitt 2.5 und Algorithmus 2.5. Die empfangenden Prozessoren reproduzieren dann die entsprechenden Faktorisierungen. Bei dieser Variante fällt das Versenden von Paketen der Größe $\mathcal{O}(1)$ und die Reproduktion durch $\mathcal{O}(n)$ Rechenoperationen an. Dies ist auf Parallelrechnern typischerweise günstiger als die Kommunikation vollständiger Singulärvektorpaare mit einem Umfang von $\mathcal{O}(2n)$.

Definition 5.1 (Heimprozessor, beherbergte Vektoren)
Seien $\sigma_1 \leq \ldots \leq \sigma_n$ die Singulärwerte von B und n_b die Breite der Blockspalten. Für $j = 1 : n$ ist P_h mit $h \equiv (j-1)/n_b \mod nprocs$ (ganzzahlige Division) der *Heimprozessor* oder *Heimknoten* des j-ten Singulärvektorpaares. Die Menge der in diesem Sinne zu P_h gehörenden Vektoren bezeichnen wir auch als *beherbergte Vektoren*.

Um den Aufwand an Kommunikation und Reproduktion zu minimieren, sollten Singulärvektorpaare möglichst auf ihren Heimprozessoren generiert werden. Es stellt sich damit die Frage, inwieweit dieses Ziel mit einer guten Lastverteilung zu vereinbaren ist.

5.2.2 Vorverarbeitung

Bearbeitung der Eingangsmatrix

Die Splitting- und Flipping-Kriterien für die Ausgangsmatrix B führt jeder Prozessor für sich durch. Ebenso erledigen bei der statischen und dynamischen Präkonditionierung alle Knoten simultan dieselbe Arbeit zur Erzeugung der Transformierten. Zerfällt die Bidiagonalmatrix in Teilprobleme, so kann jeder Prozessor bestimmen, ob er an der Lösung beteiligt ist oder nicht. Bei Matrixdimensionen ≤ 20 führen alle beteiligten Knoten das QR-Verfahren durch, behalten aber nur „ihre“ Singulärvektorpaare. Da alle bislang genannten Vorverarbeitungsmaßnahmen nur einen Bruchteil des

Gesamtaufwands ausmachen, lohnt sich eine feinere Parallelisierung kaum.

Aufdatierungen

Nach Beendigung des Kernalgorithmus werden die Vektoren nur noch von ihren Heimprozessoren bearbeitet.
Wird die Eingangsmatrix geflippt, so sind die Zeilen von U und V zu permutieren. Aufgrund des gewählten Datenlayouts ist dazu keine Kommunikation notwendig. In dieser Phase haben alle Prozessoren ein etwa gleich großes Arbeitsvolumen zu bewältigen.
Wie in Abschnitt 4.1.2 beschrieben kann das Aufdatieren der Matrizen U und V mit den Transformationen aus der Präkonditionierung adaptiv erfolgen, also insbesondere ohne Kommunikationsaufwand. Im statischen Fall liegt auch hier wieder gute Lastverteilung vor. Für die Aufdatierung nach dynamischen Präkonditionierungsschritten kann es in ungünstigen Situationen zu schlecht verteiltem Arbeitsvolumen kommen. Da diese aber eher selten auftreten und dann im Vergleich zum statischen Fall mit deutlich geringerem Aufwand verbunden sind, enthält die Routine `PDBSVDSolv` keine Ansätze für eine bessere Lastverteilung.

5.2.3 Singulärwerte

Nach Tabelle 1.1 stehen das QD- und das Bisektionsverfahren zur Bestimmung der Singulärwerte zur Auswahl. Tabelle 5.2 vergleicht die Ausführungszeiten der seriellen Routine `DLASQ1` mit der parallelen Routine `PDSTEBZ` für verschiedene Prozessorzahlen des Workstation-Clusters. Trotz der inhärenten Serialität kann sich der Einsatz des QD-Verfahrens auch bei größeren Prozessorzahlen lohnen. Interessieren nur $k < n$ Sin-

		QD	Bis, $p = 1$	$p = 4$	$p = 16$	$p = 32$	$p = 64$
P1	$n = 2000$	0.19	7.23	3.63	1.00	0.73	
P1	$n = 16000$	14.40	448.00	153.00	46.80	24.80	13.30
P4	$n = 2000$	0.80	15.90	4.25	1.08	0.78	
P4	$n = 16000$	50.00	905.00	237.00	62.20	33.10	17.70

Tabelle 5.2: QD-Verfahren für B vs. Bisektionsverfahren für T_{GK}. Notiert sind die Ausführungszeiten zur Bestimmung aller Singulärwerte. Für $n = 2000$ ist beim Übergang von 16 auf 32 Knoten die lokale Problemgröße bereits so klein, daß die gemessenen Zeiten stark durch die parallelen Grundkosten beeinflußt werden.

gulärvektorpaare, verschiebt sich das Gewicht zugunsten des Bisektionsverfahrens. Zur Bestimmung der Umschlagpunkte sollte für jeden Parallelrechner eine Kalibrierungsmeßreihe vorausgehen.
In der Implementierung von `PDBSVDSolv` kann wahlweise die parallele Routine `PDSTEBZ` oder die serielle Routine `DLASQ1` zugeschaltet werden. Es ist jedoch zu beachten, daß das Bisektionsverfahren deutlich beschleunigt werden kann, z.B. durch die abgesicherte Rayleigh-Quotienten-Iteration (Algorithmus 3.11) oder durch Anwendung des Newton-Verfahrens. Außerdem bringen Vektorisierungsstrategien Geschwindigkeitsvorteile [55]. Neben den Routinen `PDSTEBZ` und `DLASQ1` bieten sich also noch Alternativen an.

5.3 Parallelisierung des Kernalgorithmus

Aufgrund der Adaptivität des Kernalgorithmus kann eine Parallelisierung dieser Teilaufgabe effizient gestaltet werden. Wichtigster Bestandteil ist die Optimierung der Eigenwertverfahren auf höheren Stufen.

5.3.1 Der Kernalgorithmus: Die erste Stufe

Isolierte Singulärwerte

Konform mit der grundsätzlichen Strategie, Vektorpaare möglichst auf ihren Heimprozessoren zu bestimmen, übernimmt für alle isolierten Singulärwerte σ_j der jeweils zugeordnete Knoten P_h die Durchführung von Algorithmus 3.4. Es ist wichtig zu betonen, daß im parallelen Verfahren diese Arbeit erst ausgeführt wird, *nachdem* alle Pfade des Darstellungsbaumes bestimmt sind, vgl. Algorithmus 5.1.
Sind alle Singulärwerte isoliert, so gewährleistet die spaltenweise blockzyklische Abspeicherung eine gute Lastverteilung. Unter diesen günstigen Voraussetzungen ergeben sich hervorragende Skalierungseigenschaften des Kernalgorithmus, wie Experimente mit der in Abschnitt 4.3.2 beschriebenen Problemklasse P1 ($\eta = 10^{-4}$) zeigen.

Numerische Experimente

Wir betrachten zunächst das Skalierungsverhalten des Gesamtverfahrens *ohne* die Berechnung der Singulärwerte. Auf einem Knoten des Workstation-Clusters kann die aktuelle Version der Routine `PDBSVDSolv` ein System aus Singulärvektoren der Größe $n = 3200$ komplett im Hauptspeicher verwalten. Andererseits sind die Rechenzeiten für solche kleinen Matrixdimensio-

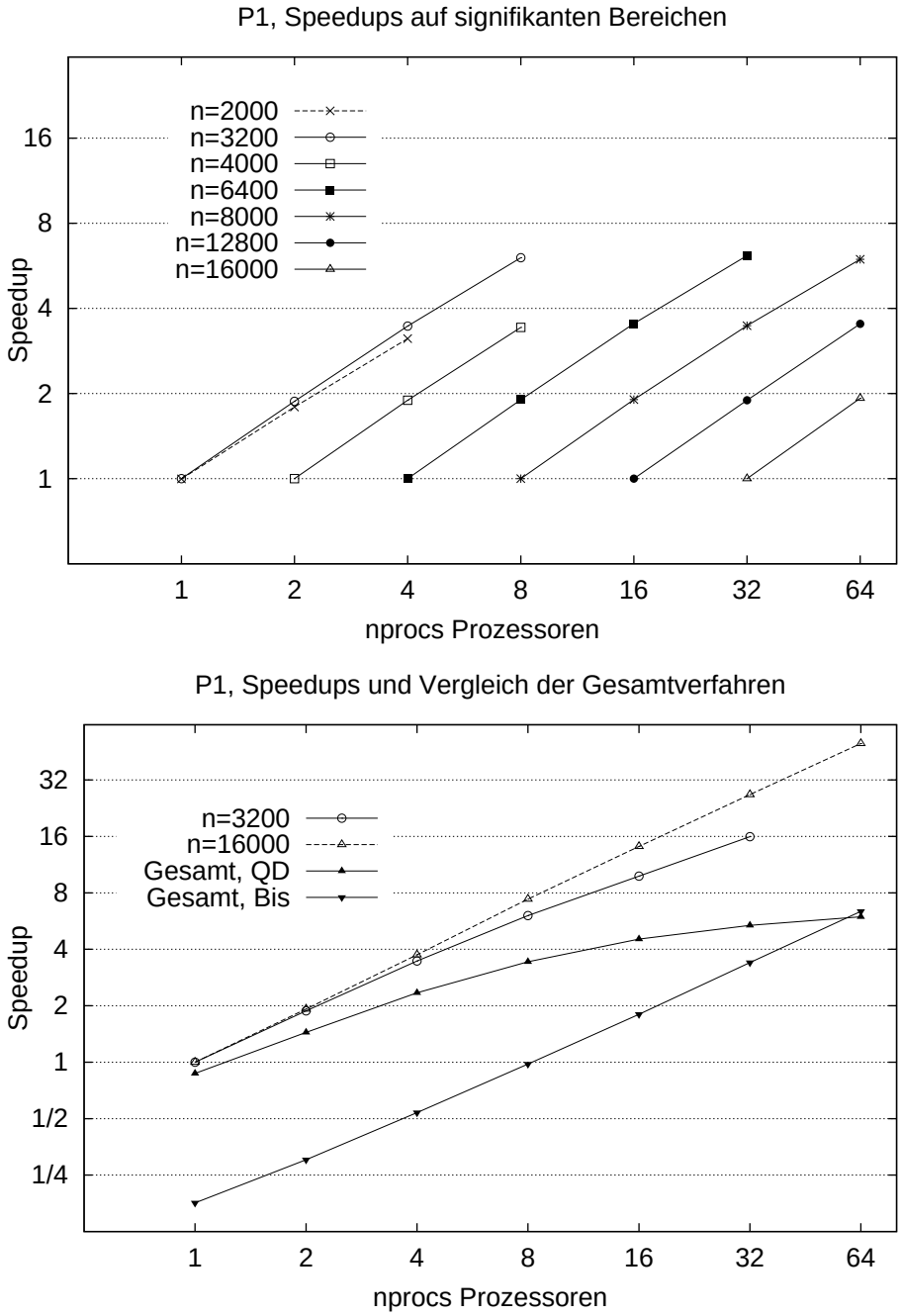


Abbildung 5.4: Skalierungseigenschaften für Problem P1. In der unteren Abbildung bezeichnen die mit $n = 3200$ und $n = 16000$ markierten Graphen die Speedups für das **bSVD**-Lösungsverfahren *ohne* die Einbeziehung des Aufwandes zur Bestimmung der Singulärwerte. Die Graphen von „Gesamt, QD“ bzw. „Gesamt, Bis“ geben für $n = 16000$ das Skalierungsverhalten bei Kombination mit dem QD- bzw. Bisektionsverfahren wieder.

nen gering, so daß beim Übergang zu höheren Prozessorzahlen der Einfluß der parallelen Grundkosten die Ergebnisse verfälscht.

Für die Einschätzung der Speedups bei festgehaltener globaler Matrixdimension sind daher in Abbildung 5.4 (oben) die signifikanten Bereiche für verschiedene Problemgrößen dargestellt. Referenzwerte sind also die Zeitmessungen t_{ser} der seriellen Implementierung für $n = 2000$ und $n = 3200$ und ansonsten t_{minprocs} für die entsprechende minimal wählbare Prozessorzahl bei höheren Werten für n. Der Parameter minprocs läßt sich also jeweils an dem zu Speedup = 1 gehörenden Abszissenwert ablesen. Es ist zu beachten, daß die *absoluten* Ausführungszeiten für die jeweils maximal verwendete Knotenmenge gering sind, da der hier betrachtete Fall isolierter Spektren nur wenig Rechenaufwand hat. In dieser Hinsicht aussagekräftiger sind die Experimente für das Testproblem P4, vgl. Abbildung 5.9.

Für die Problemgröße $n = 16000$ können wir auch für geringe Prozessorzahlen zuverlässige Zeitmessungen erzielen, wenn wir die berechneten Singulärvektoren nicht explizit in einem Feld der Größe $\mathcal{O}(n^2)$ ablegen, sondern fortlaufend ein kleineres — in den Hauptspeicher passendes — Feld überschreiben. Die eigentlich gesuchte Basis aus Singulärvektoren wird dabei natürlich zerstört. Jedoch läßt sich mit dieser Modifikation ein sehr gutes Skalierungsverhalten der Routine `PDBSVDSolv` nachweisen, vgl. den mit $n = 16000$ markierten Graphen in Abbildung 5.4 unten. Zum Vergleich ist der Graph für $n = 3200$ aufgetragen. Dieser ist dabei um die — allerdings wegen der parallelen Grundkosten verfälschten — Zeitmessungen für 16 und 32 Knoten verlängert. Bei höheren absoluten Ausführungszeiten (wie für Problem P4) liegen die Werte für die Speedups deutlich näher beieinander.

Insgesamt halten wir fest, daß die inhärente Parallelität für isolierte Spektren auch bei der konkreten Implementierung mit äußerst positiven Resultaten umgesetzt werden kann.

Die Bestimmung der Anfangsnäherungen für die Singulärwerte kann entweder mit dem QD- oder dem Bisektionsverfahren erfolgen. Für einen Vergleich ermitteln wir $\frac{t_{\text{ser}}}{t_{\text{ser,QD}}}$ bzw. $\frac{t_{\text{ser}}}{t_{\text{ser,Bis}}}$ mit den jeweiligen seriellen Ausführungszeiten der Routine `DBSVDSolv` und daraus die Speedups beider Varianten des Gesamtverfahrens. Beim Übergang zu größeren Prozessorzahlen schlägt naturgemäß der serialisierende Anteil des QD-Verfahrens zu Buche. Dennoch sind die absoluten Ausführungszeiten auch bei 32 Prozessoren immer noch besser als in Kombination mit dem Bisektionsverfahren. In Verbindung mit Tabelle 5.2 wird ferner deutlich, daß die Ermittlung der Singulärwerte von B zum dominierenden Teil bei der Bestimmung der **bSVD** wird.

Ideale Systeme

Die Konstruktion idealer Systeme ist ähnlich wie das Gram-Schmidt-Verfahren nur mit vergleichsweise hohem Kommunikationsaufwand parallelisierbar, da zur Bestimmung eines Vektors Informationen über alle bis dahin generierten Vorgänger benötigt werden. Wie bei der Behandlung sehr kleiner Bidiagonalmatrizen wird deshalb grundsätzlich folgende Strategie angewendet: Für Cluster mit einer Größe $s \leq 20$ führen alle beteiligten Knoten die Konstruktion des idealen Systems durch. Im Erfolgsfall werden nur die beherbergten Singulärvektorpaare abgespeichert und die übrigen verworfen. Dieses Vorgehen kann zu ungünstiger Lastverteilung führen. In Abbildung 5.5 sind solche Situationen bei einer Blockgröße $n_b = 4$ skizziert.

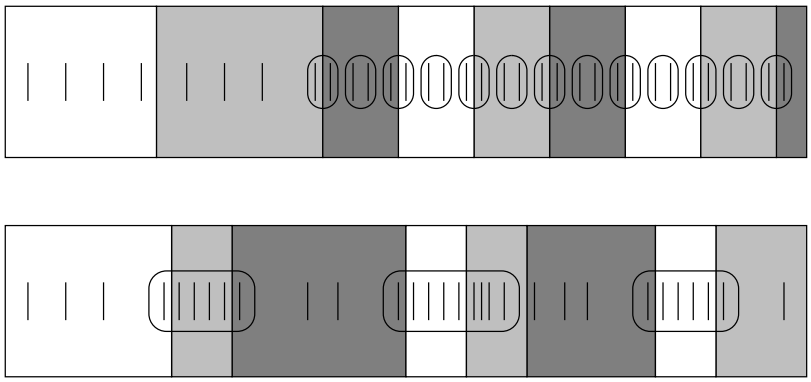


Abbildung 5.5: Ungünstige Verteilung bei idealen Systemen. Oben: Liegen wie bei Matrizen mit Wilkinson-Struktur zahlreiche kleine Cluster genau auf den Grenzen des spaltenzyklischen Layouts, so wird auf den anliegenden Prozessoren vielfach identische Arbeit ausgeführt. Unten: Möglicherweise beherbergt ein Knoten nur ein einziges Singulärvektorpaar. Trotzdem müssen alle Elemente des Clusters bearbeitet werden.

Ein Weglassen dieser Option kann jedoch das Abarbeiten weitaus umfangreicherer Darstellungsbäume notwendig machen. Der dann teilweise erheblich ansteigende Rechenaufwand ist nur schwer mit einer „ideal skalierenden“ Implementierung auszugleichen. Für eine effiziente Parallelisierung sind also Kompromisse einzugehen.
Beispielsweise können wir die Konstruktion idealer Systeme auf die einfache Trägerminimierung reduzieren. Dann kann ein Cluster komplett von einem einzigen Knoten abgearbeitet werden. Im Erfolgsfall –d.h. wenn alle Träger paarweise disjunkt sind– bekommen die übrigen Heimprozessoren pro Singulärvektorpaar neben der Twist-Position k auch die Ober- und Untergren-

zen k_u und k_l zugesendet. Basierend auf diesen Informationen kann dann eine BABE-Faktorisierung effizient aufgestellt und das entsprechende Gleichungssystem gelöst werden. Bei dieser Variante fallen allerdings die Möglichkeiten weg, mit Bisektoren oder Maßnahmen bei kleinen Trägerüberlappungen die Konstruktion idealer Systeme möglichst oft erfolgreich abzuschließen.
Wie im seriellen Fall kann auch bei der Routine `PDBSVDSolv` die Konstruktion idealer Systeme durch Einstellung entsprechender Parameter gesteuert werden.

Numerische Experimente

Wir untersuchen stellvertretend für Matrizen mit vielen Clustern geringen Umfangs das Testproblem P9 (Wilkinson-Struktur). Siehe dazu auch die Tabelle auf Seite 149. Wir vergleichen die Ausführungszeiten des Gesamtalgorithmus (ohne Singulärwerte, fortlaufende Überschreibung) einmal mit und einmal ohne Konstruktion idealer Systeme in Abbildung 5.6.

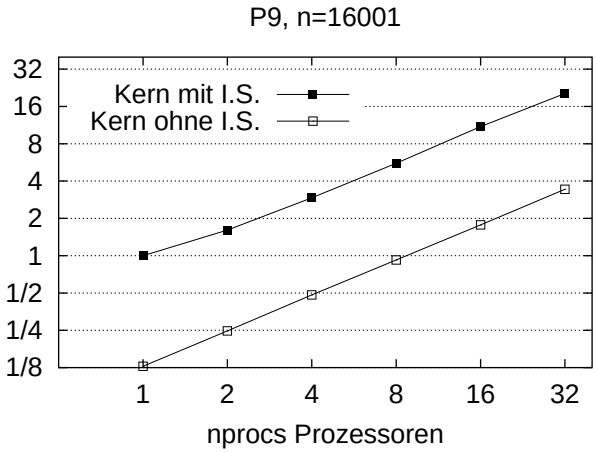


Abbildung 5.6: Skalierungseigenschaften beim Problem P9. Referenzgröße ist hier t_{ser} für den Zeitbedarf mit Konstruktion idealer Systeme (I.S.).

Es wird deutlich, daß sich die theoretische schlechte Parallelisierbarkeit dieses Ansatzes im Vergleich zu den ansonsten deutlich erhöhten Ausführungszeiten kaum negativ auswirkt. Zumindest bei den betrachteten Testproblemen sprechen die absoluten Zeiten also deutlich für den Einsatz idealer Systeme.
Bei vielen vergleichbaren Problemen wie etwa P6 (a/b) ist meistens nur der Übergang von einem auf zwei Knoten mit Einbußen verbunden, vgl. Tabelle 5.3. Wie zu erwarten stellt sich bei Erhöhung der Blockgröße n_b eine Reduktion der redundanten Arbeit und Kommunikation ein.

	P9, $n = 2001$		P6, $T(100, 9, \sqrt{\epsilon})$	
	mit I.S.	ohne I.S.	mit I.S.	ohne I.S.
$n_b = 4$	1.61	1.93	1.56	1.91
$n_b = 20$	1.79	1.95	1.63	1.92

Tabelle 5.3: Zeitfaktor beim Übergang von einem auf zwei Prozessoren.

Für Testprobleme, bei denen die Konstruktion idealer Systeme nicht ins Gewicht fällt, führt die Variation von n_b zu keinen nennenswerten Änderungen der Ausführungszeiten. Wir verwenden daher in den aufgeführten Experimenten grundsätzlich $n_b = 4$.
Die weitgehende Unabhängigkeit des **bSVD**-Lösungsverfahrens von der Blockgröße erweist sich in Kombination mit der vorgeschalteten Bidiagonalisierungsphase und der anschließenden Rück-Akkumulation als vorteilhaft. Im Gesamtkontext können wir nämlich die Parametereinstellung für das Gitter G_{Bid} optimieren und dann für G_{Diag} übernehmen. Wir ersparen uns dadurch die kommunikationsintensive Umverteilung der Spaltenblöcke, vgl. Abschnitt 5.2.1.

5.3.2 Der Kernalgorithmus: Höhere Stufen

Auf höheren Stufen im Darstellungsbaum erfolgt die Parallelisierung zweigleisig. Neben dem Aufteilen der Liste $\mathcal{L}$ wird die Bestimmung von Eigenwertapproximationen gesondert behandelt. Es sind dabei folgende Anforderungen zu erfüllen:

- Singulärvektorpaare sollen möglichst auf ihren Heimprozessoren bearbeitet werden.
- Gute Lastverteilung zwischen zwei blockierenden globalen Broadcast-Operationen.
- Bei Point-to-Point-Kommunikation sollen nur wenige Daten (Pfade) versendet werden.

Wir geben mit Algorithmus 5.1 eine grobe Beschreibung des in der Routine `PDBSVDSolv` implementierten Vorgehens. Die einzelnen Arbeitsschritte werden im Folgenden kommentiert.

Algorithmus 5.1 Der parallele Kernalgorithmus.

Gegeben: Eine bidiagonale Matrix B sowie Näherungen $\bar{\sigma}_1 \leq \ldots \leq \bar{\sigma}_n$ an die Singulärwerte. Diese sind auf allen *nprocs* Prozessoren bekannt.

Gesucht: Matrizen $\bar{U}$ und $\bar{V}$ mit Näherungen an die Basen aus linken und rechten Singulärvektoren. Diese sind spaltenweise blockzyklisch über die Prozessoren verteilt.
Pfade $\text{path}(B^TB, j) := (\nu_j^{(1)}, \ldots, \nu_j^{(\text{rec}_j)})$, die für $j = 1 : n$ die Konstruktion von $\bar{U}(:, j)$ und $\bar{V}(:, j)$ beschreiben.

1: {*Alle Prozessoren*} Beschreibe die Cluster-Struktur des Spektrums in einer Liste $\mathcal{L}$.
2: **while** $\mathcal{L} \neq \emptyset$ **do**
3: {*Alle Prozessoren*} Verteile Elemente aus $\mathcal{L}$ auf Knoten. Erzeuge also Listen $\mathcal{L}_0, \ldots, \mathcal{L}_{nprocs-1}$.
4: {*Prozessor* P_p} **Für alle Elemente aus** $\mathcal{L}_p$

- Führe die Teilaufgaben Reproduktion, *a priori* Auswahl eines Shift-Parameters $\nu^{(r)}$, Erzeugung der aktuellen Faktorisierung und Kopplung wie im seriellen Kernalgorithmus durch.
- Bestimme wie im Abschnitt 3.4.3 Anfangsnäherungen χ_j der interessierenden Eigenwerte sowie einschließende Intervallgrenzen α_j und β_j.

5: {*Globalisiere*} Shift-Parameter, Anfangsnäherungen, Intervalle
6: {*Alle Prozessoren*} Vereinige Intervalle und verteile sie auf Knoten. Erzeuge also Listen $\mathcal{I}_0, \ldots, \mathcal{I}_{nprocs-1}$.
7: {*Prozessor* P_p} Bearbeite Liste $\mathcal{I}_p$ zur Bestimmung der Eigenwertnäherungen $\bar{\tilde{\lambda}}_j$. Bestimme Eigenwertnäherungen $\bar{\bar{\lambda}}_j$ der gekoppelten Matrix. Überprüfe dabei die *a posteriori* Kriterien. Aktualisiere $\text{path}(B^TB, j)$ analog zum seriellen Kernalgorithmus.
8: {*Globalisiere*} $\text{path}(B^TB)$
9: {*Alle Prozessoren*} Aktualisiere $\mathcal{L}$ analog zum seriellen Kernalgorithmus.
10: **end while**
11: {*Prozessor* P_p} Bestimme alle beherbergten Singulärvektorpaare.
Bearbeite isolierte Singulärwerte unter Verwendung von Algorithmus 3.4, geclusterte Singulärwerte durch Reproduktion mit Pfaden.

Ein- und Ausgangsdaten, Hilfsgrößen

Die interessierenden Singulärwerte der Matrix B sind vor Aufruf von Algorithmus 5.1 allen Prozessoren bekannt zu machen. Es werden spaltenweise blockzyklisch verteilte Matrizen $\bar{U}$ und $\bar{V}$ mit Näherungen an linke und rechte Singulärvektoren gesucht. Ferner sollen die zu deren Konstruktion erforderlichen Pfade path(B^TB) bestimmt werden.
Vor den jeweiligen Lastausgleichsschritten sind Steuerungsinformationen durch Broadcast-Operationen zu *globalisieren*. Neben numerischen Daten wie Eigenwertapproximationen oder Shift-Parametern sind auch lokal erzeugte algorithmische Ergebnisse (u.a. Fehlervektor err, Indizes der Shift-Parameter) mitzuteilen. Aus Gründen der Übersichtlichkeit gehen wir auf derartige Details nicht weiter ein.

Über die Liste $\mathcal{L}$

Zunächst erfaßt jeder Knoten für sich die Cluster-Struktur der Singulärwerte in einer globalen Liste $\mathcal{L}$. Der Kommentar „*Alle Prozessoren*“ bedeutet also, daß auf gleichlautenden Daten identische Arbeitsschritte durchgeführt werden. In Zeile 3 ermittelt jeder Knoten eine lokale Liste $\mathcal{L}_p$. Dabei sollen die Elemente aus $\mathcal{L}$ möglichst gleichmäßig gestreut werden. Die Routine `PDBSVDSolv` verwendet dabei eine durch numerische Experimente motivierte Heuristik, die wir als *statische Lastverteilung* bezeichnen. Die Elemente werden in einer vorgegebenen Reihenfolge delegiert. Zur groben Einschätzung der jeweilgen Auslastung wird inkrementell festgehalten, wieviele Cluster und Eigenwerte jeder Knoten bereits zu behandeln hat.

1. Cluster, die komplett von einem Knoten beherbergt werden, sind diesem zuzuordnen.

2. Bei Clustergrößen $\geq n_b \cdot nprocs$ bearbeitet jeder Prozessor die ihm durch die Blockverteilung zugewiesenen Eigenwerte.

3. Unter den restlichen Clustern werden zunächst diejenigen verteilt, die von nur zwei Knoten beherbergt werden. Die Wahl fällt auf einen der beiden Heimprozessoren.

4. Übrig bleiben die Cluster, an denen mehr als zwei Knoten beteiligt sind, die jeweils eine Anzahl $\leq n_b$ Eigenwerte beherbergen. Hier wird versucht, die Cluster vollständig einem Prozessor zuzuordnen. Kommt es dabei zu sehr ungünstiger Lastverteilung, werden die Eigenwerte stattdessen einzeln vergeben.

Ab dem dritten Schritt erhält derjenige Knoten den Zuschlag, der bis dahin die wenigsten Eigenwerte zu bearbeiten hat. Dieses Verfahren beschreitet einen Mittelweg zwischen ausgeglichener Lastverteilung und der Anforderung, möglichst viele Singulärvektorpaare auf ihren Heimprozessoren zu bearbeiten. Es trägt auch der Beobachtung Rechnung, daß die meisten Cluster durch einmaliges Anwenden der Shift-Strategie aufgelöst werden können. Abbildung 5.7 verdeutlicht das Vorgehen.

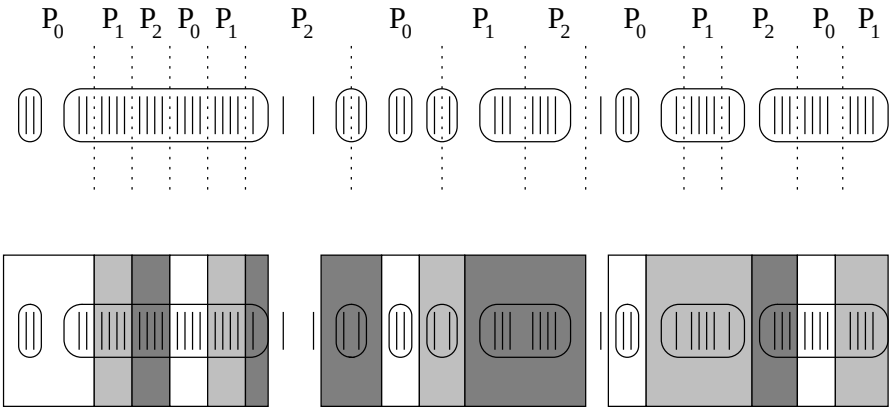


Abbildung 5.7: Parallele Bestimmung von Eigenwertapproximationen. Oben: Blockzyklisches Layout. Unten: Nach statischer Lastverteilung.

Abarbeitung der lokalen Listen $\mathcal{L}_p$

In Zeile 4 sind die Aufgaben beschrieben, die jeder *Prozessor* P_p lokal ausführt. Ein Element der Liste $\mathcal{L}_p$ erfordert die Ermittlung der entsprechenden Faktorisierung sowie die Bestimmung von Anfangsnäherungen χ_j an die dem Knoten P_p zugeordneten Eigenwerte. Der erste Teil faßt also die Zeilen 5 bis 7 aus dem seriellen Kernalgorithmus (Algorithmus 3.10) zusammen.
Die Bestimmung der Eigenwertnäherungen $\bar{\hat{\lambda}}_j$ wird bei der Parallelisierung in zwei Abschnitte zerlegt. Von den sieben Schritten des in Abschnitt 3.4.3 diskutierten Verfahrens können die ersten vier lokal im Anschluß an die Faktorisierung durchgeführt werden. Nur beim vierten Schritt — der Bestimmung der Intervallgrenzen α_j und β_j — muß mit variierendem Aufwand pro Eigenwert gerechnet werden. In vielen Fällen sind die Anfangsnäherungen χ_j aber bereits so gut, daß die absichernden Intervalle in wenigen Schritten gebildet werden können.

Von den Intervallen zu Singulärvektorpaaren

Nachdem jeder Prozessor seine lokale Liste $\mathcal{L}_p$ in eine gewisse Anzahl von Intervallen umgewandelt hat, müssen diese in einem Globalisierungsschritt bekannt gemacht werden: Da manche Eigenwerte eines Clusters auf verschiedenen Knoten in Intervalle eingeschlossen werden, kann es zu Überlappungen kommen. Es droht dann die Gefahr, daß einige Eigenwerte mehrfach und andere überhaupt nicht angenähert werden. Außerdem kann der Aufwand für die Bearbeitung der Intervalle sehr unterschiedlich ausfallen: Es ist schwierig vorherzusagen, wieviele Schritte des Bisektionsverfahrens und der abgesicherten Rayleigh-Quotienten-Iteration notwendig sind, um gute Eigenwertnäherungen zu erhalten.
Im Sinne der numerischen Absicherung und einer guten Lastverteilung werden in Zeile 5 also die lokal ermittelten Daten globalisiert. (Anstelle einer Broadcast-Operation könnten die Knoten auch gezielter untereinander Informationen austauschen. Dies lohnt sich jedoch nur, wenn sehr wenige Prozessoren an einem Cluster beteiligt sind.) Anschließend werden eventuell überlappende Intervalle zusammengefaßt und bereits konvergierte Eigenwertapproximationen als erledigt markiert. Die restlichen Intervalle werden nun in Listen $\mathcal{I}_p$ den Prozessoren zugeordnet.

- Da durch den Einsatz der Rayleigh-Quotienten-Iteration mit den Näherungen $\bar{\tilde{\lambda}}_j$ auch bereits die rechten Singulärvektoren $\bar{V}(:, j)$ approximiert werden, sollten Intervalle wieder möglichst auf ihren Heimprozessoren bearbeitet werden.

- Verbesserter Lastausgleich läßt sich hier erzielen, indem wir durch einen Parameter *loops* vorgeben, wieviele innere Schleifen (engl. *inner loops*) jeder Prozessor bei der Bearbeitung seiner Liste $\mathcal{I}_p$ durchführen soll. Jede pro Intervall abgearbeitete Teilaufgabe aus Zeile 7 kann dann durch Bewertungszahlen gewichtet werden. Überschreitet deren Summe den Parameter *loop*, können wir nicht konvergierte Intervalle durch Wiederholen der Zeilen 5 und 6 erneut verteilen. Zusätzlich zur *statischen* können wir also eine *dynamische Lastverteilung* durchführen.

Abschluß der Breitensuche

Der parallelisierte Kernalgorithmus arbeitet den Darstellungsbaum mit einer *Breadth-First-Strategie* ab: Alle Elemente der globalen Liste $\mathcal{L}$ werden bei einem Durchlauf der **while**-Schleife (Zeilen 2 bis 10) behandelt, vgl. Abbildung 5.8. Nach Überprüfung der *a posteriori* Kriterien in Zeile 7 ist geklärt, ob die Shift-Parameter $\nu^{(r)}$ geeignet sind oder nicht. Dementsprechend wird

path($B^T B$) aufdatiert. Im Anschluß an einen Globalisierungsschritt in Zeile 8 können wir $\mathcal{L}$ aktualisieren und ggf. einen weiteren Schleifendurchlauf beginnen.

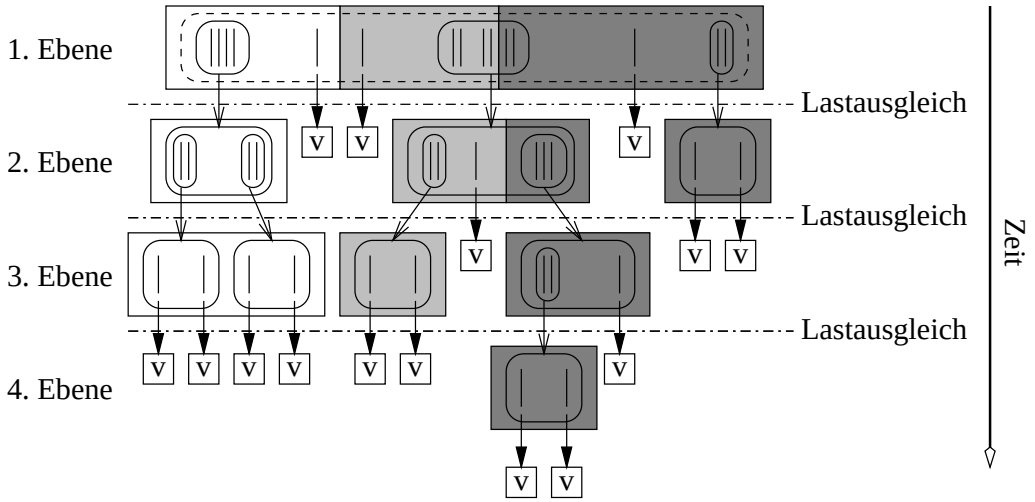


Abbildung 5.8: Breitensuche im Darstellungsbaum: Ein Durchlauf der **while**-Schleife entspricht der Abarbeitung einer vollständigen Ebene. Die Berechnung der Vektoren erfolgt erst, nachdem alle Pfade ermittelt sind.

Bestimmung aller beherbergten Singulärvektorpaare

Nach Erzeugung des gesamten Darstellungsbaumes bestimmen wir in Zeile 11 die Singulärvektoren, die noch nicht explizit auf ihren Heimprozessoren vorliegen. Darunter fallen insbesondere die zu isolierten Singulärwerten gehörenden Paare. Für die übrigen können wir mit den in den Pfaden vorliegenden Informationen die Faktorisierungs- und Kopplungsschritte reproduzieren.

Numerische Experimente

Wir untersuchen stellvertretend P4 für alle Testprobleme, bei denen die Hauptarbeit auf den höheren Stufen des Darstellungsbaumes anfällt. Die Ergebnisse der numerischen Experimente demonstrieren wir analog zu der im vorigen Abschnitt behandelten Matrixklasse P1.
Aufgrund der insgesamt höheren absoluten Ausführungszeiten können wir in Abbildung 5.9 (oben) die Speedups auf größeren signifikanten Bereichen

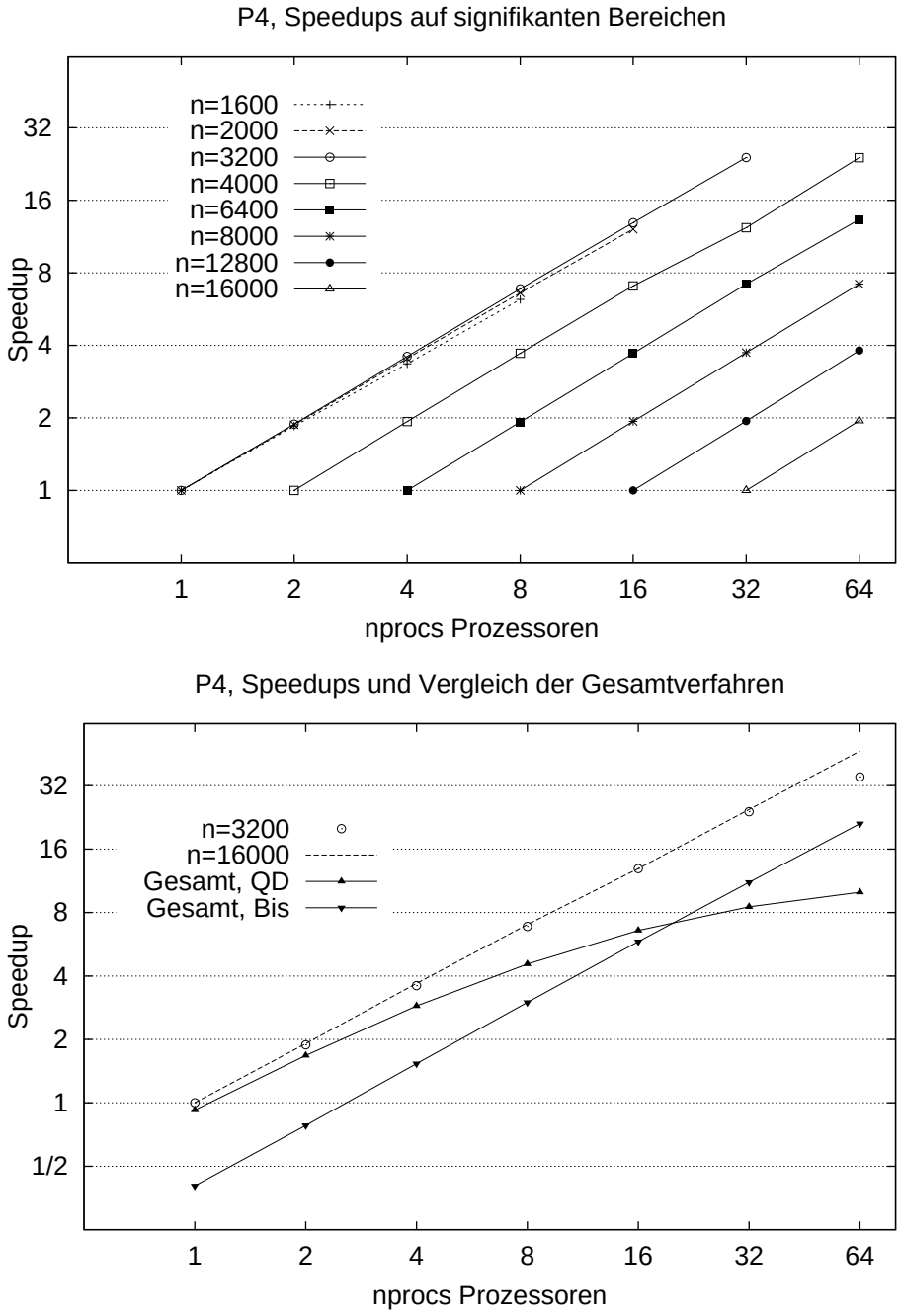


Abbildung 5.9: Skalierungseigenschaften beim Problem P4. Wie in Abbildung 5.4 bezeichnen im unteren Teil die mit $n = 3200$ und $n = 16000$ markierten Graphen die Speedups für das **bSVD**-Lösungsverfahren *ohne* die Einbeziehung des Aufwandes zur Bestimmung der Singulärwerte. Die Graphen von „Gesamt, QD“ bzw. „Gesamt, Bis“ geben für $n = 16000$ das Skalierungsverhalten bei Kombination mit dem QD- bzw. Bisektionsverfahren wieder.

heranziehen, um die Skalierungseigenschaften zufriedenstellend zu beurteilen. Außerdem unterscheidet sich das Beschleunigungsverhalten zwischen $n = 3200$ und $n = 16000$ in der unteren Darstellung kaum. Die Modifikation, bei der größeren Matrix die berechneten Singulärvektoren fortlaufend zu überschreiben, hat also bzgl. der Speedups keine Auswirkungen.
Durch im Vergleich zu P1 generell höhere Ausführungszeiten zur Bestimmung der Singulärvektoren verschiebt sich auch die Gewichtung gegenüber dem vorausgehenden QD- bzw. Bisektionsverfahren für die Singulärwerte von B. Darüberhinaus beobachten wir hier einen Umschlagpunkt zugunsten der Routine `PDSTEBZ` für Prozessorzahlen zwischen 16 und 32.
Die oben beschriebene *statische Lastverteilung* gewährleistet bereits eine sehr zufriedenstellende Parallelisierung für die Bestimmung der Eigenwertapproximationen auf höheren Stufen. Bei der *dynamischen Lastverteilung* ist eine weiter verbesserte Delegation des Arbeitsvolumens verbunden mit zusätzlichem Kommunikationsaufwand, der Einfluß auf die absoluten Ausführungszeiten nimmt. Auch zugunsten einer besseren Wartbarkeit verzichten wir daher auf eine konkrete Implementierung für die Routine `PDBSVDSolv`.

Abschließende Beurteilung

Bei der Parallelisierung des in dieser Arbeit eingeführten **bSVD**-Lösungsverfahrens steht insgesamt das Abwägen zwischen dem Design skalierbarer Lösungsstrategien und der Verwendung weniger gut verteilbarer, aber schnellerer und besser wartbarer Algorithmen im Mittelpunkt.
Numerische Experimente weisen nach, daß der Kernalgorithmus selbst wegen der inhärenten Adaptivität grundsätzlich hervorragend zur Parallelisierung geeignet ist. Bei den Skalierungseigenschaften deuten sich Ähnlichkeiten zum Bisektionsverfahren an. Dabei sind die Ausführungszeiten für die Bestimmung der Vektoren wesentlich geringer als die für die Singulärwerte.
Es zeigt sich, daß bereits durch Verwenden „einfacher“ Parallelisierungsansätze zufriedenstellende Ergebnisse erzielt werden können. Hier ist sicherlich noch Raum für weitere Optimierungen gegeben. Diese sollten dann erfolgen, wenn eine ausgereifte serielle Version vorliegt. Die aktuelle Implementierung der Routine `PDBSVDSolv` bietet durch Belegung einschlägiger Parameter Möglichkeiten, die Gewichtung zwischen den einzelnen algorithmischen Komponenten zu steuern.

Kapitel 6

Zusammenfassung und Ausblick

Der RRR-Algorithmus ist ein Lösungsverfahren für das tridiagonale symmetrische Eigenwertproblem mit bestechenden Eigenschaften bzgl. Komplexität, Genauigkeit, Adaptivität und Parallelisierbarkeit [29]. Der wesentliche Beitrag dieser Arbeit besteht in der Übertragung dieses Verfahrens auf die Bestimmung der bidiagonalen Singulärwertzerlegung $B = U\Sigma V^T$.
Neben der numerischen Orthogonalität müssen zusätzlich gute Kopplungen der Singulärvektorpaare gewährleistet werden. Wie bei vielen primär für Eigenwertaufgaben entwickelten Lösungsverfahren stoßen wir auch beim RRR-Algorithmus auf Probleme. Für die interessierenden Eigenwerte separat durchgeführter Faktorisierungen $B^TB - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $BB^T - \mu^2 I = \check{L}^+\check{D}^+(\check{L}^+)^T$ gilt:

$$\left|\hat{\lambda} - \check{\lambda}\right| = \mathcal{O}(\epsilon\mu^2)$$

Dieses Resultat ist insbesondere für betragskleine Eigenwerte unbefriedigend und führt zu schlechten Kopplungen der Vektoren. Die alternative Verwendung der Golub-Kahan Matrix T_{GK} ist numerisch weniger stabil als die Arbeit auf den Normalengleichungen. Außerdem ist bei diesem Ansatz die numerische Orthogonalität gefährdet.
Zur Lösung dieser Problematik setzen wir die Daten der Zerlegungen von $B^TB - \mu^2 I$, $BB^T - \mu^2 I$ und $T_{GK} - \mu I$ untereinander in Beziehung, beispielsweise durch

$$\hat{d}_j^+ = -\tilde{d}_{2j-1}^+\tilde{d}_{2j}^+ \quad \text{und} \quad \check{d}_j^+ = -\tilde{d}_{2j}^+\tilde{d}_{2j+1}^+$$

Benutzen wir diese Umrechnungen zwischen den Faktorisierungen, so erhalten wir für die Eigenwerte eine deutlich bessere Aussage:

$$\left| \frac{\hat{\lambda} - \check{\lambda}}{\hat{\lambda}} \right| = \mathcal{O}(\epsilon)$$

Auch für die Singulärvektoren können wir die klassische Notation $u = \frac{1}{\sigma} Bv$ durch eine Formulierung $u = \mathcal{X}_k v$ mit einer diagonalen Matrix ersetzen.
Der RRR-Algorithmus mit eingebetteten Kopplungen bildet das Kernstück eines Gesamtlösungsverfahrens für die bidiagonale Singulärwertzerlegung. Zur Verbesserung der Genauigkeit und Ausführungsgeschwindigkeit erweisen sich zusätzliche Maßnahmen wie eine Präkonditionierung oder die Konstruktion idealer Systeme als sinnvolle Ergänzungen.
Die theoretischen Vorarbeiten münden in die Entwicklung einer Routine `DBSVDSolv`. In numerischen Experimenten bestätigt sich die gute Erfüllung der Genauigkeitsanforderungen. Gleichzeitig schlägt sich die quadratische Komplexität des neu eingeführten Verfahrens in teilweise überlegenen Ausführungszeiten gegenüber den bisherigen Standard-Implementierungen des QR-Verfahrens und des Divide-and-Conquer Algorithmus nieder. Wir sehen hier noch weiteres Optimierungspotential, insbesondere wenn eine Verknüpfung mit den in der aktuellen LAPACK-Routine `DSTEGR` verwendeten Ansätzen realisiert wird.
Die inhärente Parallelität der RRR-basierten Verfahren läßt sich bereits mit einfachen Mitteln in eine gut skalierende Implementierung umsetzen. Bei der gewählten Strategie wird zunächst der Darstellungsbaum in einer Breitensuche durchlaufen. Durch Auswerten der Pfadinformationen reproduzieren die Heimprozessoren anschließend die noch fehlenden Singulärvektoren. Die Beschleunigung des Bisektionsverfahrens zur Bestimmung der Singulärwerte rückt wieder stärker in den Blickpunkt zukünftiger Aufgabenstellungen.

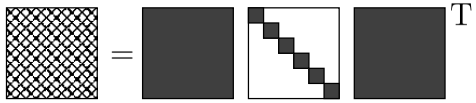
Anhang A

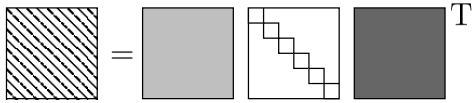
Die wichtigsten Fakten in Stichworten

Die folgenden Seiten sind als Orientierungshilfe für die in dieser Arbeit entwickelten Ergebnisse gedacht. Die Priorität liegt hierbei auf der Übersichtlichkeit. Daher führen wir die wichtigsten Punkte stichwortartig auf und verzichten an einigen Stellen auf die formal korrekte Darstellung. Folgende Themen sind noch einmal zusammengefaßt:

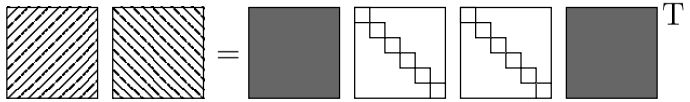
- **SEP** und **SVD** in exakter und in Gleitpunktarithmetik.
- Überblick gängiger Lösungsverfahren.
- Anwendung des RRR-Algorithmus auf die **bSVD**.
- Zusammenfassung der Resultate über Eigenwerte.
- Beziehungen unter den Algorithmen.
- Die Komponenten des Gesamtalgorithmus.
- Numerische Experimente, Parallelisierung.

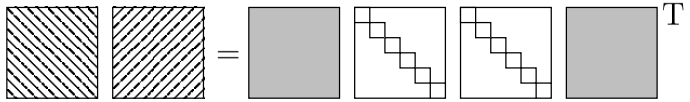
SEP und SVD, Spezialfall tSEP und bSVD

$H = Q\Lambda Q^T$ 

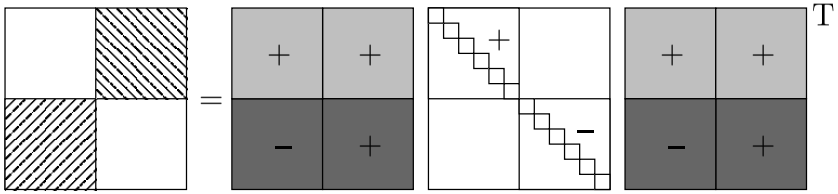
$A = U\Sigma V^T$ 

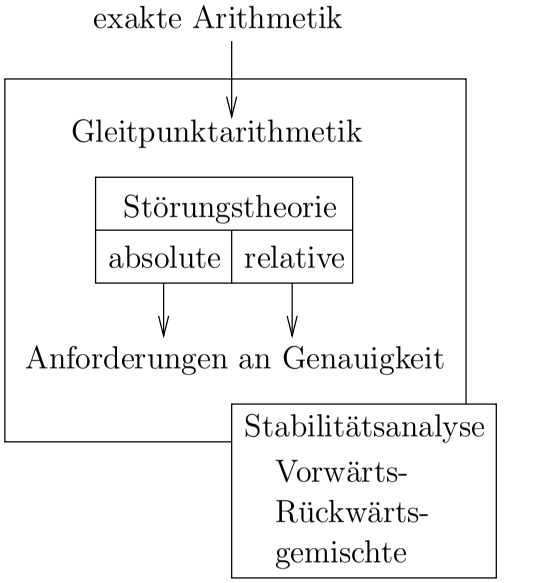
Normalengleichungen:

$A^T A = V\Sigma^2 V^T$ 

$AA^T = U\Sigma^2 U^T$ 

Jordan-Wielandt-Darstellung:





Überblick gängiger Lösungsverfahren

Abschnitt 1.4:
Lösungsverfahren für **SEP** und **SVD**

Abschnitt 1.5:
Lösungsverfahren für **tSEP** und **bSVD**

Kapitel 2:
Der RRR-Algorithmus für das **tSEP**

Relativ robuste Repräsentationen
QD-artige Transformationen
BABE-Faktorisierungen für Eigenvektoren
Shift-Strategie für Cluster
Darstellungsbaum

Die Darstellung fächert sich mit zunehmender Spezialisierung auf. Unter den Lösungsverfahren für allgemeine Matrizen konzentrieren wir uns auf diejenigen, die erst nach einer Vorverarbeitungsphase mit dem iterativen Prozeß auf der tri- oder bidiagonalen Matrix beginnen. Unter diesen heben wir wiederum den RRR-Algorithmus für das **tSEP** besonders hervor.

Anwendung des RRR-Algorithmus auf die bSVD mit Black-Box Strategie

Probleme bei Normalengleichungen: Schlechtes Residuum bei Clustern großer Eigenwerte
Probleme bei Golub-Kahan Matrix: Faktorisierungen bei Clustern kleiner Eigenwerte kritisch
Ein hybrider Ansatz?

Anwendung des RRR-Algorithmus auf die bSVD mit Kopplungen

	Erste Ebene	Höhere Ebenen	
Faktorisierungen	Abschnitt 3.2.1	Abschnitt 3.3.1	
Kopplungen (exakt)	Abschnitt 3.2.2	Abschnitt 3.3.2	
$\hat{Z}^+ \leftrightarrow \tilde{Z}^+ \leftrightarrow \check{Z}^+$	Lemma 3.6		
$\hat{Z}^+ \leftrightarrow \check{Z}^+$	Folgerung 3.7		
Pivot-Darstellung	Folgerung 3.8	Satz 3.17, Folgerung 3.18	
Matrix-Notation	Bemerkung 3.9	Bemerkung 3.20	
Kopplungen (numerisch)	$\hat{Z}^+ \leftrightarrow \check{Z}^+$	$\tilde{Z}^+ \rightarrow \hat{Z}^+, \check{Z}^+$	$\hat{Z}^+ \rightarrow \tilde{Z}^+ \rightarrow \check{Z}^+$
Umrechnung	Algorithmus 3.3	Algorithmus 3.6	Algorithmus 3.7
Datenfluß	Abbildung 3.8	Abbildung 3.11	Abbildung 3.12
Rückwärtstabilität	ja, Abbildung 3.9	ja, Abbildung 3.13	nein, Abbildung 3.13
Eigenwerte	Satz 3.14	Satz 3.21	nur *a posteriori*
Anwendungsbereich	Darstellungsbaum in Abbildung 3.10	Darstellungsbaum in Abbildung 3.14	

Zusammenfassung der Resultate über Eigenwerte

Satz 2.2 und Bemerkung 2.3:
Durchführung einer gemischten Stabilitätsanalyse für die Zerlegung $B^TB - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$ analog zu den Theoremen aus [29, 37, 38].

Satz 3.1:
Berechnen wir $B^TB - \mu^2 I = \hat{L}^+\hat{D}^+(\hat{L}^+)^T$ und $BB^T - \mu^2 I = \check{L}^+\check{D}^+(\check{L}^+)^T$ mit separaten Zerlegungen, so gilt für die exakten Eigenwerte der gestörten LDL^T-Faktorisierungen

$$\left|\hat{\lambda} - \check{\lambda}\right| = \mathcal{O}(\epsilon\mu^2)$$

Abschnitt 3.1.3, Abbildung 3.6:
Idee: Statt separate Zerlegungen zu berechnen, nutzen wir Beziehungen zwischen den Daten von $\hat{L}^+\hat{D}^+(\hat{L}^+)^T$und $\check{L}^+\check{D}^+(\check{L}^+)^T$sowie einer entsprechenden Zerlegung $\tilde{L}^+\tilde{D}^+(\tilde{L}^+)^T$einer Translation der Golub-Kahan Matrix aus. Wir unterscheiden dabei Kopplungen auf der ersten Ebene

$$B, T_{GK} \Rightarrow (\hat{D}^+ \leftrightarrow \tilde{D}^+ \leftrightarrow \check{D}^+)$$

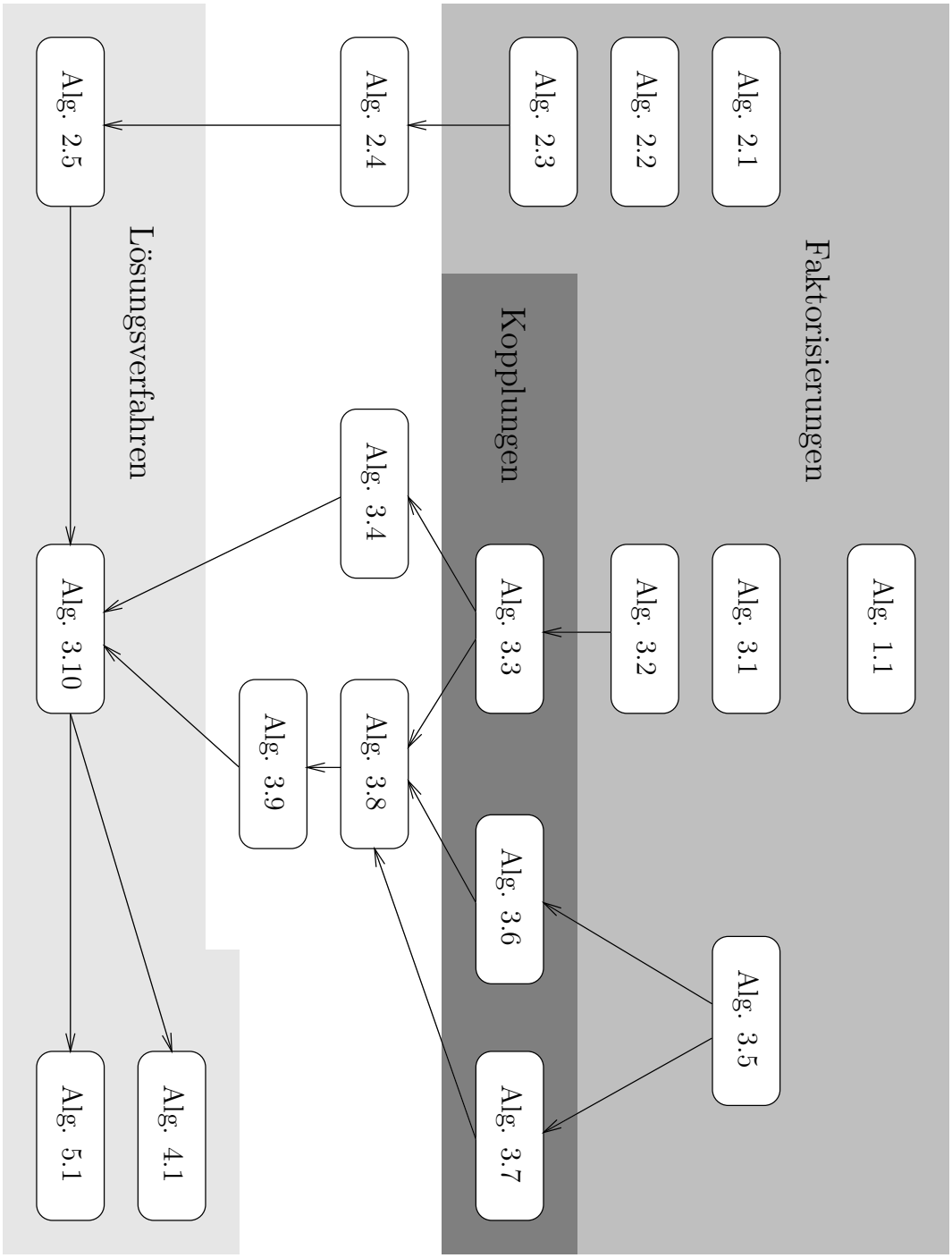
und Kopplungen auf höheren Ebenen:

$$\hat{D}, \tilde{D}, \check{D} \Rightarrow (\hat{D}^+ \leftrightarrow \tilde{D}^+ \leftrightarrow \check{D}^+)$$

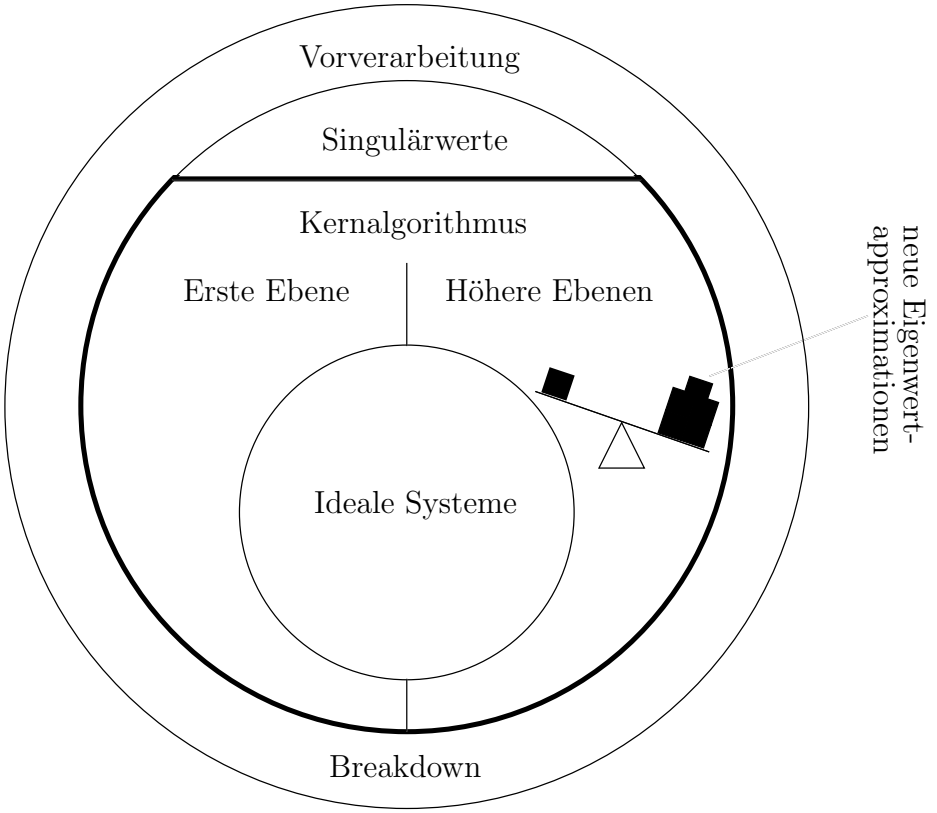
Abschnitt 3.2:
Mit Kopplungen auf der ersten Ebene erhalten wir nach Satz 3.14

$$\left|\frac{\hat{\lambda} - \check{\lambda}}{\hat{\lambda}}\right| = \mathcal{O}(\epsilon)$$

Abschnitt 3.3:
Nach Bemerkung 3.16 überträgt sich die gemischte Stabilitätsanalyse für eine einzelne Zerlegung auf sukzessive Faktorisierungen. Benutzen wir die Umrechnung $\tilde{D}^+ \rightarrow [\hat{D}^+, \check{D}^+]$, so erhalten wir mit Satz 3.21 ein vergleichbares Ergebnis wie in Satz 3.14. Bei Verwendung der Umrechnung $\hat{D}^+ \rightarrow \tilde{D}^+ \rightarrow \check{D}^+$ ziehen wir uns auf die *a posteriori* Überprüfung der Eigenwerte zurück.

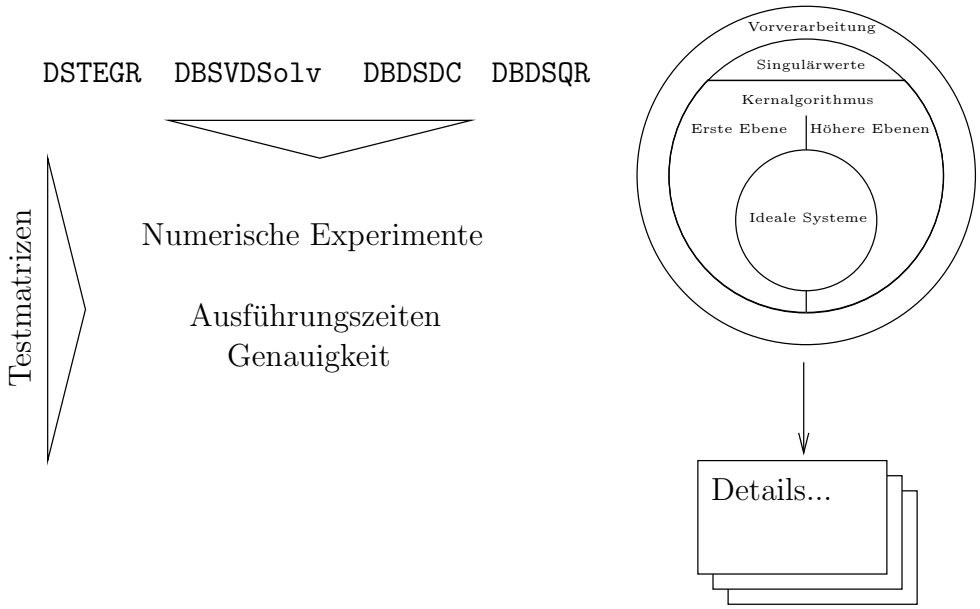

Faktorisierungen
Kopplungen
Lösungsverfahren
Alg. 1.1
Alg. 2.1
Alg. 2.2
Alg. 2.3
Alg. 2.4
Alg. 2.5
Alg. 3.1
Alg. 3.2
Alg. 3.3
Alg. 3.4
Alg. 3.5
Alg. 3.6
Alg. 3.7
Alg. 3.8
Alg. 3.9
Alg. 3.10
Alg. 4.1
Alg. 5.1

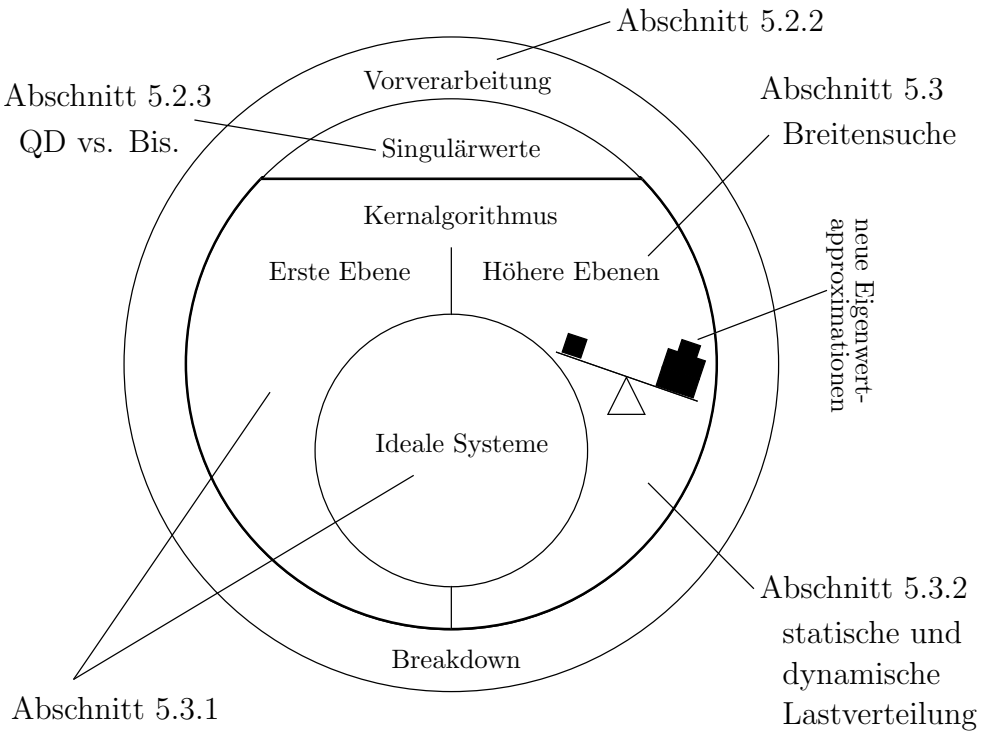
Der Gesamtalgorithmus



Das in dieser Arbeit vorgeschlagene **bSVD**-Lösungsverfahren setzt sich aus mehreren Komponenten zusammen. Der Kernalgorithmus basiert auf dem RRR-Verfahren für das **tSEP**, in das die Kopplungsmechanismen zur Bestimmung der Singulärvektoren eingebettet sind. Wir beachten die Unterschiede für isolierte Singulärwerte (erste Ebene) und Cluster (höhere Ebenen). Bei deren Behandlung macht die Bestimmung neuer Eigenwertapproximationen den überwiegenden Anteil aus. Hochgenaue Anfangsnäherungen an die Singulärwerte können entweder mit dem QD- oder dem Bisektionsverfahren ermittelt werden. Zur Verbesserung der numerischen Eigenschaften nehmen wir Maßnahmen „von außen“ (Vorverarbeitung: Splitting, Flipping, Präkonditionierung; Breakdown) und „von innen“ (Ideale Systeme) vor.

Numerische Experimente, Parallelisierung





Anhang B

Rundungsfehleranalysen für BABE-Faktorisierungen

In diesem Anhang entwickeln wir Algorithmen zur Bestimmung der Faktorisierung

$$N_\kappa G_\kappa N_\kappa^T - \tau I = N_k^+ G_k^+ (N_k^+)^T, k = 1 : n$$

Die linke Seite ist also nicht wie im Rest der Arbeit als LDL^T-Faktorisierung, sondern als allgemeinere BABE-Faktorisierung gegeben. Wir diskutieren diese Generalisierung durch eine gemischte relative Stabilitätsanalyse. Siehe auch Bemerkung 2.3 auf Seite 45, Bemerkung 2.13 auf Seite 54 sowie Bemerkung 3.15 auf Seite 90.

B.1 BABE-Faktorisierungen für $\hat{L}\hat{D}\hat{L}^T - \tau I$ und $\check{U}\check{R}\check{U}^T - \tau I$

In Algorithmus 3.5 (Seite 91) ist beschrieben, wie aus einer Translation einer als LDL^T-Faktorisierung gegebenen Matrix die entsprechende BABE-Faktorisierung gebildet werden kann:

$$\hat{L}\hat{D}\hat{L}^T - \tau I = \hat{N}_{\hat{k}}^+ \hat{G}_{\hat{k}}^+ (\hat{N}_{\hat{k}}^+)^T, \hat{k} = 1 : n \tag{B.1}$$

Wir entwickeln daraus nun ein Verfahren, um die BABE-Faktorisierung der Translation einer als URU^T-Faktorisierung gegebenen Matrix zu bilden:

$$\check{U}\check{R}\check{U}^T - \tau I = \check{N}_{\check{k}}^{+}\check{G}_{\check{k}}^{+}(\check{N}_{\check{k}}^{+})^T, \check{k} = 1:n \tag{B.2}$$

Dazu verwenden wir die in Bemerkung 3.10 (Seite 78) eingeführte Permutationsmatrix P mit $P \cdot x = x(n:-1:1)$ und schreiben Gleichung (B.2) um zu

$$P\check{U}PP\check{R}PP\check{U}^T P - \tau I = P\check{N}_{\check{k}}^{+}PP\check{G}_{\check{k}}^{+}PP(\check{N}_{\check{k}}^{+})^T P, \check{k} = 1:n \tag{B.3}$$

Dabei sind $P\check{R}P$ und $P\check{G}_{\check{k}}^{+}P$ Diagonalmatrizen, $P\check{U}P$ eine untere Bidiagonalmatrix mit Einsen auf der Diagonalen und $P\check{N}_{\check{k}}^{+}P$ eine verdrehte Matrix mit Twist-Position $n+1-\check{k}$. Aus

$$\begin{aligned}\check{N}_{\check{k}}^{+} &= I &+ \quad & \texttt{diag}([\check{l}_1^{+},\ldots,\check{l}_{\check{k}-1}^{+},0,\ldots,0],-1)\\ &&+ \quad & \texttt{diag}([0,\ldots,0,\check{u}_{\check{k}}^{+},\ldots,\check{u}_{n-1}^{+}],1)\end{aligned}$$

wird also

$$\begin{aligned}P\check{N}_{\check{k}}^{+}P &= I &+ \quad & \texttt{diag}([\check{u}_{n-1}^{+},\ldots,\check{u}_{\check{k}}^{+},0,\ldots,0],-1)\\ &&+ \quad & \texttt{diag}([0,\ldots,0,\check{l}_{\check{k}-1}^{+},\ldots,\check{l}_1^{+}],1)\end{aligned}$$

während bei den Diagonalmatrizen die Einträge in umgekehrter Reihenfolge stehen:

$$P\check{G}_{\check{k}}^{+}P = \texttt{diag}([\check{r}_n^{+},\ \ldots,\check{r}_{\check{k}+1}^{+},\check{\gamma}_{\check{k}}^{+},\check{d}_{\check{k}-1}^{+},\ \ldots,\check{d}_1^{+}])$$

Die Besetztheitsstruktur aus Gleichung (B.3) entspricht damit bis auf die Twist-Position der aus Gleichung (B.1). Durch Umnummerieren können wir also aus Algorithmus 3.5 ein Verfahren zur Berechnung der Zerlegung (B.2) erzeugen:

$$\begin{array}{ccc}\hat{d}_i = \check{r}_{n+1-i}, & & \hat{l}_i = \check{u}_{n-i}\\ \hat{s}_i = \check{s}_{n+1-i}, & \hat{d}_i^{+} = \check{r}_{n+1-i}^{+}, & \hat{l}_i^{+} = \check{u}_{n-i}^{+}\\ \hat{p}_i = \check{p}_{n+1-i}, & \hat{r}_i^{+} = \check{d}_{n+1-i}^{+}, & \hat{u}_i^{+} = \check{l}_{n-i}^{+}\\ & \hat{\gamma}_i^{+} = \check{\gamma}_{n+1-i}^{+} & \end{array} \tag{B.4}$$

Dementsprechend wird auf der linken Seite von Algorithmus B.1 die Zerlegung (B.1) und auf der rechten Seite die Zerlegung (B.2) realisiert.

Algorithmus B.1 Faktorisiere $\hat{L}\hat{D}\hat{L}^T - \tau I$ (links) und $\check{U}\check{R}\check{U}^T - \tau I$ (rechts).

Gegeben: $[\hat{D}, \hat{L}]$, τ
Gesucht: $\hat{Z}^+ = [\hat{D}^+, \hat{L}^+, \hat{S}, \hat{R}^+, \hat{U}^+, \hat{P}, \hat{\Gamma}^+]$

$\hat{s}_1 = -\tau$
for $i = 1 : n-1$ **do**
  $\hat{d}_i^+ = \hat{d}_i + \hat{s}_i$
  $\hat{l}_i^+ = \frac{\hat{d}_i \hat{l}_i}{\hat{d}_i^+}$
  $\hat{s}_{i+1} = \hat{l}_i^+ \hat{l}_i \hat{s}_i - \tau$
end for
$\hat{d}_n^+ = \hat{d}_n + \hat{s}_n$
$\hat{p}_n = \hat{d}_n - \tau$
for $i = n-1 : -1 : 1$ **do**
  $\hat{r}_{i+1}^+ = \hat{d}_i \hat{l}_i^2 + \hat{p}_{i+1}$
  $t = \frac{\hat{d}_i}{\hat{r}_{i+1}^+}$
  $\hat{u}_i^+ = \hat{l}_i t$
  $\hat{p}_i = \hat{p}_{i+1} t - \tau$
end for
$\hat{r}_1^+ = \hat{p}_1$
for $i = 1 : n$ **do**
  $\hat{\gamma}_i^+ = \hat{s}_i + \hat{p}_i + \tau$
end for

Gegeben: $[\check{R}, \check{U}]$, τ
Gesucht: $\check{Z}^+ = [\check{D}^+, \check{L}^+, \check{P}, \check{R}^+, \check{U}^+, \check{S}, \check{\Gamma}^+]$

$\check{s}_n = -\tau$
for $i = n-1 : -1 : 1$ **do**
  $\check{r}_{i+1}^+ = \check{s}_{i+1} + \check{r}_{i+1}$
  $\check{u}_i^+ = \frac{\check{r}_{i+1} \check{u}_i}{\check{r}_{i+1}^+}$
  $\check{s}_i = \check{u}_i^+ \check{u}_i \check{s}_{i+1} - \tau$
end for
$\check{r}_1^+ = \check{s}_1 + \check{r}_1$
$\check{p}_1 = \check{r}_1 - \tau$
for $i = 1 : n-1$ **do**
  $\check{d}_i^+ = \check{r}_{i+1} \check{u}_i^2 + \check{p}_i$
  $t = \frac{\check{r}_{i+1}}{\check{d}_i^+}$
  $\check{l}_i^+ = \check{u}_i t$
  $\check{p}_{i+1} = \check{p}_i t - \tau$
end for
$\check{d}_n^+ = \check{p}_n$
for $i = 1 : n$ **do**
  $\check{\gamma}_i^+ = \check{s}_i + \check{p}_i + \tau$
end for

B.2 Stabilitätsanalyse

Wir geben im folgenden die Ergebnisse einer gemischten Rundungsfehleranalyse mit kleinen multiplikativen Störungen für BABE-Faktorisierungen von $\hat{L}\hat{D}\hat{L}^T - \tau I$ und $\check{U}\check{R}\check{U}^T - \tau I$ an. Zunächst werden wieder die Eingangsdaten komponentenweise um kleine Beträge multiplikativ gestört. Auf diesen leicht modifizierten Eingangsdaten wird die entsprechende Faktorisierung in exakter Arithmetik angewendet. Das Ergebnis dieser Operation kann dann wiederum als leichte Störung des in Gleitpunktarithmetik bestimmten Resultats interpretiert werden.

Satz B.1 (Stabilitätsanalyse für die Zerlegung (B.1))
Wir nehmen an, daß die linke Seite von Algorithmus B.1 ohne Unter- und Überlauf durchgeführt werden kann. Dann kommutiert das nachfolgende

Diagramm.

$$\begin{array}{ccc} \hat{D}, \hat{L} & \xrightarrow[\text{Alg.B.1, links}]{\text{berechnet}} & \hat{D}^+, \hat{L}^+, \hat{R}^+, \hat{U}^+, \hat{\Gamma}^+ \\ \Big| & & \Big\uparrow \\ \text{Störung (B.5)} & & \text{Störung (B.6)} \\ \Big| & & \Big\uparrow \\ \widehat{D}, \widehat{L} & \xrightarrow[\text{Alg.B.1, links}]{\text{exakt}} & \widehat{D}^+, \widehat{L}^+, \widehat{R}^+, \widehat{U}^+, \widehat{\Gamma}^+ \end{array}$$

Die multiplikativen Störungen sind dabei wie folgt festgelegt:

$$\begin{array}{rclcll} \widehat{d}_i &=& \hat{d}_i(1+\kappa_{\hat{d}}^{(i)}\epsilon), & |\kappa_{\hat{d}}^{(i)}| &\leq 1+\mathcal{O}(\epsilon), & 1 \leq i < \hat{k} \\ \widehat{l}_i &=& \hat{l}_i(1+\kappa_{\hat{l}}^{(i)}\epsilon), & |\kappa_{\hat{l}}^{(i)}| &\leq 3+\mathcal{O}(\epsilon), & 1 \leq i < \hat{k} \\ \widehat{d}_{\hat{k}} &=& \hat{d}_{\hat{k}}(1+\kappa_{\hat{d}}^{(\hat{k})}\epsilon), & |\kappa_{\hat{d}}^{(\hat{k})}| &\leq 4+\mathcal{O}(\epsilon) & \\ \widehat{l}_{\hat{k}} &=& \hat{l}_{\hat{k}}(1+\kappa_{\hat{l}}^{(\hat{k})}\epsilon), & |\kappa_{\hat{l}}^{(\hat{k})}| &\leq 7/2+\mathcal{O}(\epsilon) & \\ \widehat{d}_i &=& \hat{d}_i(1+\kappa_{\hat{d}}^{(i)}\epsilon), & |\kappa_{\hat{d}}^{(i)}| &\leq 3+\mathcal{O}(\epsilon), & \hat{k} < i \leq n \\ \widehat{l}_i &=& \hat{l}_i(1+\kappa_{\hat{l}}^{(i)}\epsilon), & |\kappa_{\hat{l}}^{(i)}| &\leq 3+\mathcal{O}(\epsilon), & \hat{k} < i < n \end{array} \tag{B.5}$$

$$\begin{array}{rclcll} \widehat{d}_i^+ &=& \hat{d}_i^+(1+\kappa_{\hat{d}^+}^{(i)}\epsilon), & |\kappa_{\hat{d}^+}^{(i)}| &\leq 2+\mathcal{O}(\epsilon), & 1 \leq i < \hat{k} \\ \widehat{l}_i^+ &=& \hat{l}_i^+(1+\kappa_{\hat{l}^+}^{(i)}\epsilon), & |\kappa_{\hat{l}^+}^{(i)}| &\leq 3+\mathcal{O}(\epsilon), & 1 \leq i < \hat{k} \\ \widehat{\gamma}_{\hat{k}}^+ &=& \hat{\gamma}_{\hat{k}}^+(1+\kappa_{\hat{\gamma}^+}^{(\hat{k})}\epsilon) & |\kappa_{\hat{\gamma}^+}^{(\hat{k})}| &\leq 2+\mathcal{O}(\epsilon) & \\ \widehat{u}_{\hat{k}}^+ &=& \hat{u}_{\hat{k}}^+(1+\kappa_{\hat{u}^+}^{(\hat{k})}\epsilon) & |\kappa_{\hat{u}^+}^{(\hat{k})}| &\leq 9/2+\mathcal{O}(\epsilon) & \\ \widehat{r}_i^+ &=& \hat{r}_i^+(1+\kappa_{\hat{r}^+}^{(i)}\epsilon), & |\kappa_{\hat{r}^+}^{(i)}| &\leq 2+\mathcal{O}(\epsilon), & \hat{k} < i \leq n \\ \widehat{u}_i^+ &=& \hat{u}_i^+(1+\kappa_{\hat{u}^+}^{(i)}\epsilon), & |\kappa_{\hat{u}^+}^{(i)}| &\leq 4+\mathcal{O}(\epsilon), & \hat{k} < i < n \end{array} \tag{B.6}$$

Beweis: Der Satz wird in [29] als Theorem 4.4.4 bewiesen. □

Satz B.2 (Stabilitätsanalyse für die Zerlegung (B.2))
Wir nehmen an, daß die rechte Seite von Algorithmus B.1 ohne Unter- und Überlauf durchgeführt werden kann. Dann kommutiert das nachfolgende Diagramm.

$$\begin{array}{ccc} \check{R}, \check{U} & \xrightarrow[\text{Alg.B.1, rechts}]{\text{berechnet}} & \check{D}^+, \check{L}^+, \check{R}^+, \check{U}^+, \check{\Gamma}^+ \\ \Big| & & \Big\uparrow \\ \text{Störung (B.7)} & & \text{Störung (B.8)} \\ \Big| & & \Big\uparrow \\ \breve{R}, \breve{U} & \xrightarrow[\text{Alg.B.1, rechts}]{\text{exakt}} & \breve{D}^+, \breve{L}^+, \breve{R}^+, \breve{U}^+, \breve{\Gamma}^+ \end{array}$$

Die multiplikativen Störungen sind dabei wie folgt festgelegt:

$$\begin{array}{rcll}
\check{r}_i &=& \check{r}_i(1+\kappa_{\check{r}}^{(i)}\epsilon), & |\kappa_{\check{r}}^{(i)}| \leq 3+\mathcal{O}(\epsilon), \quad 1 \leq i < \check{k} \\
\check{u}_i &=& \check{u}_i(1+\kappa_{\check{u}}^{(i)}\epsilon), & |\kappa_{\check{u}}^{(i)}| \leq 3+\mathcal{O}(\epsilon), \quad 1 \leq i < \check{k}-1 \\
\check{r}_{\check{k}} &=& \check{r}_{\check{k}}(1+\kappa_{\check{r}}^{(\check{k})}\epsilon), & |\kappa_{\check{r}}^{(\check{k})}| \leq 4+\mathcal{O}(\epsilon) \\
\check{u}_{\check{k}-1} &=& \check{u}_{\check{k}-1}(1+\kappa_{\check{u}}^{(\check{k}-1)}\epsilon), & |\kappa_{\check{u}}^{(\check{k}-1)}| \leq 7/2+\mathcal{O}(\epsilon) \\
\check{r}_i &=& \check{r}_i(1+\kappa_{\check{r}}^{(i)}\epsilon), & |\kappa_{\check{r}}^{(i)}| \leq 1+\mathcal{O}(\epsilon), \quad \check{k} < i \leq n \\
\check{u}_i &=& \check{u}_i(1+\kappa_{\check{u}}^{(i)}\epsilon), & |\kappa_{\check{u}}^{(i)}| \leq 3+\mathcal{O}(\epsilon), \quad \check{k} \leq i < n
\end{array} \tag{B.7}$$

$$\begin{array}{rcll}
\check{d}_i^+ &=& \check{d}_i^+(1+\kappa_{\check{d}^+}^{(i)}\epsilon), & |\kappa_{\check{d}^+}^{(i)}| \leq 2+\mathcal{O}(\epsilon), \quad 1 \leq i < \check{k} \\
\check{l}_i^+ &=& \check{l}_i^+(1+\kappa_{\check{l}^+}^{(i)}\epsilon), & |\kappa_{\check{l}^+}^{(i)}| \leq 4+\mathcal{O}(\epsilon), \quad 1 \leq i < \check{k}-1 \\
\check{\gamma}_{\check{k}}^+ &=& \check{\gamma}_{\check{k}}^+(1+\kappa_{\check{\gamma}^+}^{(\check{k})}\epsilon), & |\kappa_{\check{\gamma}^+}^{(\check{k})}| \leq 2+\mathcal{O}(\epsilon) \\
\check{l}_{\check{k}-1}^+ &=& \check{l}_{\check{k}-1}^+(1+\kappa_{\check{l}^+}^{(\check{k}-1)}\epsilon), & |\kappa_{\check{l}^+}^{(\check{k}-1)}| \leq 9/2+\mathcal{O}(\epsilon) \\
\check{r}_i^+ &=& \check{r}_i^+(1+\kappa_{\check{r}^+}^{(i)}\epsilon), & |\kappa_{\check{r}^+}^{(i)}| \leq 2+\mathcal{O}(\epsilon), \quad \check{k} < i \leq n \\
\check{u}_i^+ &=& \check{u}_i^+(1+\kappa_{\check{u}^+}^{(i)}\epsilon), & |\kappa_{\check{u}^+}^{(i)}| \leq 3+\mathcal{O}(\epsilon), \quad \check{k} \leq i < n
\end{array} \tag{B.8}$$

Beweis: Die Zerlegung (B.2) läßt sich durch Umnummerierung gemäß den Gleichungen (B.4) aus der Zerlegung (B.1) erzeugen. Mit Anwendung von Satz B.1 folgt daher die Behauptung. □

B.3 BABE-Faktorisierungen für $N_\kappa G_\kappa N_\kappa^T - \tau I$

Für die BABE-Faktorisierungen von $\hat{L}\hat{D}\hat{L}^T - \tau I$ und $\check{U}\check{R}\check{U}^T - \tau I$ sind also Algorithmen bekannt, die einer gemischten Stabilitätsanalyse genügen. Wir untersuchen nun noch den Fall, daß die linke Seite selbst durch eine verdrehte Faktorisierung gegeben ist:

$$N_\kappa G_\kappa N_\kappa^T - \tau I = L^+ D^+ (L^+)^T \tag{B.9}$$

$$N_\kappa G_\kappa N_\kappa^T - \tau I = U^+ R^+ (U^+)^T \tag{B.10}$$

$$N_\kappa G_\kappa N_\kappa^T - \tau I = N_k^+ G_k^+ (N_k^+)^T, k = 1:n \tag{B.11}$$

Dazu betrachten wir Partitionierungen $G_\kappa = \texttt{diag}([D_\mathrm{o}, \gamma_\kappa, R_\mathrm{u}])$ mit $D_\mathrm{o} = \texttt{diag}([d_1. \ \ldots, d_{\kappa-1}]) \in \mathbb{R}^{\kappa-1}$ und $R_\mathrm{u} = \texttt{diag}([r_{\kappa+1}, \ \ldots, r_n]) \in \mathbb{R}^{n-\kappa}$ sowie

$$N_\kappa = \begin{pmatrix} L_\mathrm{o} & 0 & 0 \\ l_{\kappa-1} \cdot e_{\kappa-1}^T & 1 & u_\kappa \cdot e_1^T \\ 0 & 0 & U_\mathrm{u} \end{pmatrix}$$

Dabei sind $L_{\mathrm{o}} = I + \mathtt{diag}([l_1, \ldots, l_{\kappa-2}], -1) \in \mathbb{R}^{(\kappa-1)\times(\kappa-1)}$ und $U_{\mathrm{u}} = I + \mathtt{diag}([u_{\kappa+1}, \ldots, u_{n-1}], 1) \in \mathbb{R}^{(n-\kappa)\times(n-\kappa)}$ Bidiagonalmatrizen sowie $e_{\kappa-1} \in \mathbb{R}^{\kappa-1}$ bzw. $e_1 \in \mathbb{R}^{n-\kappa}$ der letzte bzw. erste Einheitsvektor entsprechender Dimension. Blockweises Auswerten von $N_\kappa G_\kappa N_\kappa^T - \tau I$ ergibt dann

$$\begin{pmatrix} L_{\mathrm{o}} D_{\mathrm{o}} L_{\mathrm{o}}^T - \tau I & d_{\kappa-1} l_{\kappa-1} \cdot e_{\kappa-1} & 0 \\ d_{\kappa-1} l_{\kappa-1} \cdot e_{\kappa-1}^T & d_{\kappa-1} l_{\kappa-1}^2 + \gamma_\kappa + r_{\kappa+1} u_\kappa^2 - \tau & e_1^T \cdot r_{\kappa+1} u_\kappa \\ 0 & r_{\kappa+1} u_\kappa \cdot e_1 & U_{\mathrm{u}} R_{\mathrm{u}} U_{\mathrm{u}}^T - \tau I \end{pmatrix}$$

Eine entsprechende Partitionierung der Faktorisierung $L^+ D^+ (L^+)^T$ von der rechten Seite der Gleichung (B.9) ergibt

$$\begin{pmatrix} L_{\mathrm{o}}^+ D_{\mathrm{o}}^+ (L_{\mathrm{o}}^+)^T & d_{\kappa-1}^+ l_{\kappa-1}^+ \cdot e_{\kappa-1} & 0 \\ d_{\kappa-1}^+ l_{\kappa-1}^+ \cdot e_{\kappa-1}^T & d_\kappa^+ + d_{\kappa-1}^+ (l_{\kappa-1}^+)^2 & e_1^T \cdot d_\kappa^+ l_\kappa^+ \\ 0 & d_\kappa^+ l_\kappa^+ \cdot e_1 & L_{\mathrm{u}}^+ D_{\mathrm{u}}^+ (L_{\mathrm{u}}^+)^T + d_\kappa^+ (l_\kappa^+)^2 \cdot e_1 e_1^T \end{pmatrix}$$

Der Vergleich der Blöcke liefert dann

$$\begin{aligned} L_{\mathrm{o}} D_{\mathrm{o}} L_{\mathrm{o}}^T - \tau I &= L_{\mathrm{o}}^+ D_{\mathrm{o}}^+ (L_{\mathrm{o}}^+)^T && \text{(B.12)} \\ d_{\kappa-1} l_{\kappa-1} &= d_{\kappa-1}^+ l_{\kappa-1}^+ && \text{(B.13)} \\ d_{\kappa-1} l_{\kappa-1}^2 + \gamma_\kappa + r_{\kappa+1} u_\kappa^2 - \tau &= d_\kappa^+ + d_{\kappa-1}^+ (l_{\kappa-1}^+)^2 && \text{(B.14)} \\ r_{\kappa+1} u_\kappa &= d_\kappa^+ l_\kappa^+ && \text{(B.15)} \\ U_{\mathrm{u}} R_{\mathrm{u}} U_{\mathrm{u}}^T - \tau I &= L_{\mathrm{u}}^+ D_{\mathrm{u}}^+ (L_{\mathrm{u}}^+)^T + d_\kappa^+ (l_\kappa^+)^2 \cdot e_1 e_1^T && \text{(B.16)} \end{aligned}$$

Die Zerlegung (B.12) sowie die Auflösung der Gleichung (B.13) wird mit Hilfe der Zeilen 1 bis 7 der linken Seite von Algorithmus B.1 berechnet. Wir brechen dabei die Schleife bereits bei $\kappa-1$ statt bei $n-1$ ab. Wir verwenden

$$\hat{s}_\kappa = \hat{d}_\kappa^+ - \hat{d}_\kappa = d_{\kappa-1} l_{\kappa-1}^2 - d_{\kappa-1}^+ (l_{\kappa-1}^+)^2 - \tau$$

und schreiben Gleichung (B.14) um zu

$$d_\kappa^+ - r_{\kappa+1} u_\kappa^2 = \hat{s}_\kappa + \gamma_\kappa \qquad \text{(B.17)}$$

Die linke Seite von (B.17) entspricht aber gerade dem noch fehlenden Wert von $\check{p}_\kappa$. Nun können wir die Faktorisierung (B.16) mit Hilfe der Zeilen 8 bis 15 der rechten Seite von Algorithmus B.1 bestimmen. Die Schleife startet dabei beim Index κ. Die linke Seite des neuen Algorithmus B.2 bestimmt also die LDL^T-Faktorisierung von $N_\kappa G_\kappa N_\kappa^T - \tau I$. Die auf der rechten Seite von Algorithmus B.2 dargestellte URU^T-Faktorisierung erhalten wir analog.
Es sind also Algorithmen für die Faktorisierungen (B.9) und (B.10) bekannt. Für die Zerlegung (B.11) müssen wir lediglich noch die Werte von Γ^+ bestimmen. Diese ergeben sich aber wieder zu

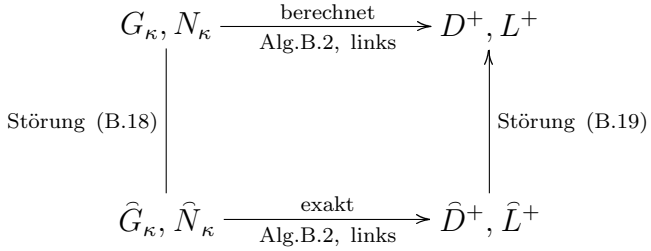
$$\begin{aligned} \gamma_i^+ &= \hat{s}_i + \hat{p}_i + \tau, \quad 1 \le i \le \kappa \\ \gamma_i^+ &= \check{s}_i + \check{p}_i + \tau, \quad \kappa < i \le n \end{aligned}$$

Algorithmus B.2 Faktorisiere $N_\kappa G_\kappa N_\kappa^T - \tau I$.

Gegeben: $[G_\kappa, N_\kappa]$, τ	**Gegeben:** $[G_\kappa, N_\kappa]$, τ
Gesucht: LDL^T-Faktorisierung	**Gesucht:** URU^T-Faktorisierung
$[D^+, L^+, \hat{S}, \check{P}]$	$[R^+, U^+, \hat{P}, \check{S}]$
$\hat{s}_1 = -\tau$	$\check{s}_n = -\tau$
for $i = 1 : \kappa - 1$ **do**	**for** $i = n - 1 : -1 : \kappa$ **do**
$d_i^+ = d_i + \hat{s}_i$	$r_{i+1}^+ = \check{s}_{i+1} + r_{i+1}$
$l_i^+ = \frac{d_i l_i}{d_i^+}$	$u_i^+ = \frac{r_{i+1} u_i}{r_{i+1}^+}$
$\hat{s}_{i+1} = l_i^+ l_i \hat{s}_i - \tau$	$\check{s}_i = u_i^+ u_i \check{s}_{i+1} - \tau$
end for	**end for**
$\check{p}_\kappa = \hat{s}_\kappa + \gamma_\kappa$	$\hat{p}_\kappa = \check{s}_\kappa + \gamma_\kappa$
for $i = \kappa : n - 1$ **do**	**for** $i = \kappa - 1 : -1 : 1$ **do**
$d_i^+ = r_{i+1} u_i^2 + \check{p}_i$	$r_{i+1}^+ = d_i l_i^2 + \hat{p}_{i+1}$
$t = \frac{r_{i+1}}{d_i^+}$	$t = \frac{d_i}{r_{i+1}^+}$
$l_i^+ = u_i t$	$u_i^+ = l_i t$
$\check{p}_{i+1} = \check{p}_i t - \tau$	$\hat{p}_i = \hat{p}_{i+1} t - \tau$
end for	**end for**
$d_n^+ = \check{p}_n$	$r_1^+ = \hat{p}_1$

B.4 Stabilitätsanalyse

Satz B.3 (Stabilitätsanalyse für die Zerlegung (B.9))
Wir nehmen an, daß die linke Seite von Algorithmus B.2 ohne Unter- und Überlauf durchgeführt werden kann. Dann kommutiert das nachfolgende Diagramm.



Die multiplikativen Störungen sind dabei wie folgt festgelegt:

$$\begin{array}{rclcll} \widehat{d}_i &=& d_i(1+\kappa_d^{(i)}\epsilon), & |\kappa_d^{(i)}| &\leq& 1+\mathcal{O}(\epsilon), \quad 1 \leq i < \kappa \\ \widehat{l}_i &=& l_i(1+\kappa_l^{(i)}\epsilon), & |\kappa_l^{(i)}| &\leq& 3+\mathcal{O}(\epsilon), \quad 1 \leq i < \kappa \\ \widehat{\gamma}_\kappa &=& \gamma_\kappa(1+\kappa_\gamma^{(\kappa)}\epsilon), & |\kappa_\gamma^{(\kappa)}| &\leq& 1+\mathcal{O}(\epsilon) \\ \widehat{r}_i &=& r_i(1+\kappa_r^{(i)}\epsilon), & |\kappa_r^{(i)}| &\leq& 1+\mathcal{O}(\epsilon), \quad \kappa < i \leq n \\ \widehat{u}_i &=& u_i(1+\kappa_u^{(i)}\epsilon), & |\kappa_u^{(i)}| &\leq& 3+\mathcal{O}(\epsilon), \quad \kappa \leq i < n \end{array} \tag{B.18}$$

$$\begin{array}{rclcll} \widehat{d}_i^+ & = & d_i^+(1+\kappa_{d^+}^{(i)}\epsilon), & |\kappa_{d^+}^{(i)}| & \leq & 2+\mathcal{O}(\epsilon), \quad 1 \leq i < \kappa \\ \widehat{l}_i^+ & = & l_i^+(1+\kappa_{l^+}^{(i)}\epsilon), & |\kappa_{l^+}^{(i)}| & \leq & 3+\mathcal{O}(\epsilon), \quad 1 \leq i < \kappa \\ \widehat{d}_i^+ & = & d_i^+(1+\kappa_{d^+}^{(i)}\epsilon), & |\kappa_{d^+}^{(i)}| & \leq & 2+\mathcal{O}(\epsilon), \quad \kappa \leq i \leq n \\ \widehat{l}_i^+ & = & l_i^+(1+\kappa_{l^+}^{(i)}\epsilon), & |\kappa_{l^+}^{(i)}| & \leq & 4+\mathcal{O}(\epsilon), \quad \kappa \leq i < n \end{array} \tag{B.19}$$

Beweis: Die Störungen für die Zerlegungen $LDL^T - \tau I = L^+D^+(L^+)^T$ bzw. $URU^T - \tau I = L^+D^+(L^+)^T$ stammen aus Satz B.1 bzw. aus Satz B.2. Es ist nur noch eine Analyse für die in Zeile 7 angegebene Operation durchzuführen. Wir betrachten zunächst die Rechnung in exakter Arithmetik

$$\breve{p}_\kappa = \widehat{s}_\kappa + \widehat{\gamma}_\kappa$$

Aus den Beweisen zu Theorem 4.4.4 in [29] sowie aus dem Beweis zu Satz 2.2 auf Seite 43 wissen wir, daß die Störung

$$\widehat{s}_\kappa = \hat{s}_\kappa(1+\epsilon_s) \quad \text{mit} \quad |\epsilon_s| \leq 1+\mathcal{O}(\epsilon)$$

bereits festgelegt ist. Nach dem Modell der Gleitpunktarithmetik (1.3.2) erhalten wir

$$\breve{p}_\kappa = (\hat{s}_\kappa + \gamma_\kappa)(1+\epsilon_+)$$

Als Wahl für die Störungsansätze genügt nun

$$\breve{p}_\kappa = \breve{p}_\kappa(1+\epsilon_s)/(1+\epsilon_+) \quad \text{und} \quad \widehat{\gamma}_\kappa = \gamma_\kappa(1+\epsilon_s)$$

□

Analog zum obigen Satz können wir nun eine Stabilitätsanalyse für die Zerlegung (B.10) und letztlich für die Faktorisierung (B.11) durchführen. Wir verzichten hier aber auf die Angabe der entsprechenden Sätze und schließen den Anhang damit ab.

Literaturverzeichnis

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, PA, 3rd edition, 1999.

[2] ANSI/IEEE. *IEEE Standard for Binary Floating Point Arithmetic*. Std 754-1985 edition, 1985.

[3] ARPACK: A collection of Fortran77 subroutines designed to solve large scale eigenvalue problems.
`http://www.caam.rice.edu/software/ARPACK`.

[4] A. Barrlund. The p-relative distance is a metric. *SIAM J. Matrix Anal. Appl.*, 21(2), 2000.

[5] M. Bečka and M. Vajteršic. Block-Jacobi SVD algorithms for distributed memory systems I : Hypercubes and rings. *Parallel Algorithms and Applications*, 13:265–287, 1999.

[6] C. Bischof. Computing the singular value decomposition on a distributed system of vector processors. *Parallel Comput.*, 11:171–186, 1989.

[7] C. Bischof, S. Huss-Ledermann, X. Sun, A. Tsao, and T. Turnbull. Parallel performance of a symmetric eigensolver based on the invariant subspace decomposition approach. In *Proceedings of Scalable High Performance Computing Conference*, March 1994.

[8] C. Bischof, B. Lang, and X. Sun. The SBR toolbox - Software for successive band reduction. Erscheint in *ACM Trans. Math. Soft.*, 1996.

[9] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen. Rank-one modification of the symmetric eigenproblem. *Numer. Math.*, 31:31–48, 1978.

[10] J. Choi, J. Demmel, I. Dhillon, J. Dongarra, S. Ostrouchov, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. ScaLAPACK: A portable linear algebra library for distributed memory computers—design issues and performance. *Computer Phys. Comm.*, 97:1–15, 1996.

[11] J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker, and R. C. Whaley. A proposal for a set of parallel basic linear algebra subprograms. In J. Dongarra, K. Masden, and J. Waśniewski, editors, *Applied Parallel Computing*, pages 107–114. Springer Verlag, 1995.

[12] J. Choi, J. J. Dongarra, and D. W. Walker. The design of a parallel dense linear algebra software library: Reduction to Hessenberg, tridiagonal, and bidiagonal form. *Numer. Alg.*, 10:379–399, 1995.

[13] A. K. Cline. Everything you always wanted to know about singular values—but were afraid to ask. Unveröffentlichtes Manuskript.

[14] J. M. Cuppen. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.*, 36:177–195, 1981.

[15] C. Davis and W. Kahan. Some new bounds on perturbation subspaces. *Bull. Am. Math. Soc.*, 77(4):863–868, 1969.

[16] C. Davis and W. Kahan. The rotation of eigenvectors by a perturbation III. *SIAM J. Numer. Anal.*, 7:248–263, 1970.

[17] J. Demmel. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[18] J. Demmel, I. Dhillon, and H. Ren. On the correctness of some bisection-like parallel eigenvalue algorithms in floating point arithmetic. *ETNA, Electron. Trans. Numer. Anal.*, 3:116–149, 1997.

[19] J. Demmel, M. Gu, S. Eisenstat, I. Slapnicar, K. Veselić, and Z. Drmač. Computing the singular value decomposition with high relative accuracy. *Lin. Alg. Appl.*, 299(1-3):21–80, 1992.

[20] J. Demmel, M. Heath, and H. van der Vorst. Parallel numerical linear algebra. *Acta Numerica*, 2, 1993.

[21] J. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Stat. Comput.*, 11(5):873–912, 1990.

[22] J. Demmel and P. Koev. Accurate inverses through accurate minors for structured matrices. Erscheint in *Lin. Alg. Appl.*, Januar 2001.

[23] J. Demmel and A. McKenney. A test matrix generation suite. Technical Report MCS-P69-0389, Argonne National Laboratory, March 1989. LAPACK Working Note #9.

[24] J. Demmel and K. Veselić. Jacobi's method is more accurate than QR. *SIAM J. Matrix Anal. Appl.*, 13(4):1204–1246, 1992.

[25] I. Dhillon. Current inverse iteration software can fail. *BIT Num. Math.*, 38(4):685–704, 1998.

[26] I. Dhillon and B. Parlett. Fernando's solution to Wilkinson's problem: An application of double factorization. *Lin. Alg. Appl.*, 267:247–279, 1997.

[27] I. Dhillon and B. Parlett. Relatively robust representations for symmetric tridiagonals. Erscheint in *Lin. Alg. Appl.*, Januar 1999.

[28] I. Dhillon and B. Parlett. Orthogonal eigenvectors and relative gaps. Zur Publikation eingereicht, Februar 2000.

[29] I. S. Dhillon. *A new $\mathcal{O}(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem.* PhD thesis, University of California, Berkeley, CA, 1997.

[30] J. Dongarra and D. Sorensen. A fully parallel algorithm for the symmetric eigenvalue problem. *SIAM J. Sci. Stat. Comput.*, 8(2):139–154, 1987.

[31] J. Dongarra and F. Tisseur. Parallelizing the divide and conquer algorithm for the symmetric tridiagonal eigenvalue problem on distributed memory architectures. *SIAM J. Sci. Stat. Comput.*, 6(20):2223–2236, 1999.

[32] J. Dongarra and R. Whaley. A User's Guide to the BLACS v1.0. Technical Report CS-95-281, University of Tennessee at Knoxville, June 1995. LAPACK Working Note #94.

[33] J. J. Dongarra, J. Du Croz, S. Hammarling, and I. Duff. A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Soft.*, 16(1):1–17, March 1990.

[34] D. Faddeev, V. Kublanovskaya, and V. Faddeeva. Sur les systèmes linèaires algèbraiques de matrices rectangulaires et mal-conditionnèes. *Colloq. Int. du C.N.R.S*, 165:161–170, 1968.

[35] K. Fernando. Computing an eigenvector of a tridiagonal when the eigenvalue is known. *Z. angew. Math. Mech.*, 76:299–302, 1996.

[36] K. Fernando. On computing an eigenvector of a tridiagonal matrix. I: Basic results. *SIAM J. Matrix Anal. Appl.*, 18(4):1013–1034, 1997.

[37] K. Fernando. Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices. *SIAM J. Matrix Anal. Appl.*, 20(2):373–399, 1998.

[38] K. Fernando and B. Parlett. Accurate singular values and differential qd algorithms. *Numer. Math.*, 67(2):191–229, 1994.

[39] S. Godunov, A. Antonov, O. Kiriljuk, and V. Kostin. *Guaranteed Accuracy in Numerical Linear Algebra*. Kluwer Academic, 1993.

[40] S. Godunov, V. Kostin, and A. Mitchenko. Computation of an eigenvector of a symmetric tridiagonal matrix. *Sib. Math. J.*, 26:684–696, 1985.

[41] D. Goldberg. What every computer scientist should know about floating point arithmetic. *ACM Surveys*, 23:5–48, 1991.

[42] G. Golub. Some modified matrix eigenvalue problems. *SIAM Rev.*, 15:318–334, 1973.

[43] G. Golub and W. Kahan. Calculating the singular values and pseudo-inverse of a matrix. *SIAM J. Numer. Anal.*, 2(2):205–224, 1965.

[44] G. Golub and C. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 3rd edition, 1996.

[45] A. Greenbaum and J. Dongarra. Experiments with QR/QL methods for the symmetric tridiagonal eigenproblem. Technical Report CS-89-92, University of Tennessee at Knoxville, November 1989. LAPACK Working Note #17.

[46] R. Gregory and D. Karney. *A collection of matrices for testing computional algorithms*. John Wiley, New York, 1969.

[47] B. Großer. Parallele Reduktionsalgorithmen zur Berechnung der Singulärwertzerlegung. Diplomarbeit, Fachbereich Mathematik, Bergische Universität GH Wuppertal, April 1997.

[48] B. Großer and B. Lang. Efficient parallel reduction to bidiagonal form. *Parallel Comput.*, 25(8):969–986, 1999.

[49] B. Großer and B. Lang. An $\mathcal{O}(n^2)$ algorithm for the bidiagonal SVD. Erscheint in *Lin. Alg. Appl.*, Januar 2001.

[50] M. Gu and S. Eisenstat. A stable and efficient algorithm for the rank-1 modification of the symmetric eigenproblem. *SIAM J. Matrix Anal. Appl.*, 15:1266–1276, 1994.

[51] M. Gu and S. Eisenstat. A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM J. Matrix Anal. Appl.*, 16(1):79–92, 1995.

[52] M. Gu and S. Eisenstat. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix Anal. Appl.*, 16(1):172–191, 1995.

[53] N. Higham. A survey of componentwise perturbation theory in numerical linear algebra, 1994. In W. Gautschi, editor, *Mathematics of Computation 1943–1993: A Half Century of Computational Mathematics*, volume 48 of *Proceedings of Symposia in Applied Mathematics*, pages 49–77. American Mathematical Society, Providence, RI, USA, 1994.

[54] W. Kahan. Accurate eigenvalues of a symmetric tridiagonal matrix. Technical Report CS41, Stanford University, July 1966.

[55] B. Lang. Parallele Berechnung von Eigensystemen symmetrischer Bandmatrizen. *Z. angew. Math. Mech.*, 74:T 541–T 544, 1994.

[56] B. Lang. Parallel reduction of banded matrices to bidiagonal form. *Parallel Comput.*, 22(1):1–18, 1996.

[57] LAPACK: Linear Algebra PACKage - a numerical software library. `http://www.netlib.org/lapack`.

[58] R. Li. Relative perturbation theory. I: Eigenvalue and singular value variations. *SIAM J. Matrix Anal. Appl.*, 19(4):956–982, 1998.

[59] R. Li. Relative perturbation theory. II: Eigenspace and singular subspace variations. *SIAM J. Matrix Anal. Appl.*, 20(2):471–492, 1998.

[60] O. Marques and B. Parlett. An implementation of the dqds algorithm (positive case). *Lin. Alg. Appl.*, 309:217–259, 2000.

[61] MATLAB: The language of technical computing. `http://www.mathworks.com`.

[62] ParaStation: A communication sub-system. `http://ParaStation.ira.uka.de/`.

[63] B. Parlett. *The Symmetric Eigenvalue Problem*. Prentice Hall, Englewood Cliffs, 1980.

[64] B. Parlett. Invariant subspaces for tightly clustered eigenvalues of tridiagonals. *BIT Num. Math.*, 36(3):542–652, 1996.

[65] H. Rutishauser. Der Quotienten-Differenzen-Algorithmus. *Z. angew. Math. Phys.*, 5:233–251, 1954.

[66] J. Rutter. A serial implementation of Cuppen's divide and conquer algorithm for the symmetric eigenvalue problem. Technical Report UCB//CSD-94-799, University of California at Berkeley, February 1994. LAPACK Working Note #69.

[67] ScaLAPACK: Scalable Linear Algebra PACKage - a parallel numerical software library. `http://www.netlib.org/scalapack/index.html`.