Technische Universität Chemnitz
Fakultät für Mathematik

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Diplomarbeit

zum Thema

Locking und Purging für den Hamiltonischen Lanczos-Prozess

eingereicht von:

Martin Stoll

geb. am 16.02.1980 in Btzow

Hauptstraße 31
18246 Zepelin

10. Semester Mathematik mit
vertiefter Informatikausbildung
Matrikelnummer: 25317

Betreuer:

Prof. Dr. Peter Benner
(Technische Universitï¿½ Chemnitz)

30. April 2009

Aufgabenstellung

Es soll eine numerische Implementierung des Hamiltonischen Lanczos-Algorithmus' erstellt werden. Dabei ist zunï¿½hst der SR Algorithmus zur Lï¿½ung der projizierten Hamiltonischen Eigenwertprobleme zu implementieren. Der Hamiltonische Lanczos-Algorithmus soll dann mit einem impliziten Restart-Mechanismus versehen werden. Dabei soll zur Konvergenzbeschleunigung eine Methode zum "Locken" und "Purgen" konvergierter Ritz-Paare entwickelt werden. Hierbei soll eine Variante untersucht werden, die auf der Entwicklung einer geeigneten Krylov-Schur-artigen Zerlegung fr Hamiltonische Matrizen beruht.

Inhaltsverzeichnis

Verzeichnis der Algorithmen **3**

1 Motivation und grundlegende Begriffe **5**
1.1 Motivation 5
1.2 Grundlegende Begriffe 7

2 Der SR Algorithmus **11**
2.1 SR Zerlegung 11
2.2 Transformationsmatrizen 14
2.3 Implizite Shifts 16
2.3.1 Einfacher Shift 17
2.3.2 Doppelshift 18
2.3.3 Quadrupelshift 21
2.4 Bestimmung der Eigenwerte 27
2.5 Bestimmung der Eigenvektoren 29

3 Der symplektische Lanczos-Algorithmus mit impliziten Restarts **35**
3.1 Der Lanczos-Algorithmus für unsymmetrische Matrizen 35
3.2 Der symplektische Lanczos-Algorithmus 36
3.3 Der symplektische Lanczos-Algorithmus mit impliziten Restarts 39
3.3.1 Einzelshift 39
3.3.2 Doppelshift 40
3.3.3 Quadrupelshift 41
3.4 Abbruchkriterien 42

4 Deflationstechniken und Krylov-Schur-artige Zerlegung **47**
4.1 Die Krylov-Schur-artige Zerlegung 47
4.2 Tauschen von Eigenwerten und Eigenblöcken 51
4.3 *Locking* und *Purging* 54
4.4 Abbruchkriterien 54
4.5 Shift-und-Invert Strategien 57
4.5.1 Das quadratische Eigenwertproblem 57
4.5.2 Shift-und-Invert für das quadratische Eigenwertproblem 58
4.5.3 “Positive-Real-Balancing“ 59

5 Numerische Eigenschaften **61**
5.1 Zusammenbruch der symplektischen Lanczos-Verfahrens 61
5.2 Stabilität 64

6 Numerische Ergebnisse **67**
6.1 SR Algorithmus . . . 67
6.2 Krylov-Schur-artiger Algorithmus . . . 72
6.2.1 Das quadratische Eigenwertproblem und die Fichera-Ecke . . . 72
6.2.2 Wärmeleitungsgleichung . . . 77

Zusammenfassung **79**

Ausblick **79**

A MATLAB Routinen **80**
A.1 Grundlegende Routinen . . . 80
A.1.1 perm . . . 80
A.1.2 unperm . . . 80
A.1.3 housegen . . . 81
A.1.4 orthov . . . 81
A.1.5 orthow . . . 81
A.2 SR Algorithmus . . . 82
A.2.1 hsr . . . 82
A.2.2 implicit_sr_qs . . . 83
A.2.3 implicit_sr_qs3 . . . 83
A.2.4 complev . . . 84
A.2.5 evquasischur . . . 84
A.2.6 invlr . . . 85
A.2.7 hamlr . . . 85
A.3 Symplektischer Lanczos-Prozess mit impliziten Restarts . . . 86
A.3.1 irsl . . . 86
A.3.2 lanczosstart . . . 86
A.3.3 lanczosblow . . . 86
A.3.4 implicitsrds . . . 87
A.3.5 implicitsrqs . . . 88
A.3.6 implicitsrqs3 . . . 88
A.4 Krylov-Schur-artiger Algorithmus . . . 89
A.4.1 krschur . . . 89
A.4.2 lanczosstart_h . . . 89
A.4.3 lanczosblow_h . . . 90
A.4.4 orthQ . . . 90
A.4.5 upswap . . . 91
A.4.6 downswap . . . 91
A.4.7 invhamquadr . . . 92
A.4.8 jhessrow . . . 92
A.5 Externe Routinen . . . 92
A.5.1 gauss1 . . . 92
A.5.2 givens . . . 93
A.5.3 sinvert . . . 93

Literaturverzeichnis **94**

List of Algorithms

2.1 Bulge Chasing Algorithmus für den einfachen Shift 19
2.2 Bulge Chasing Algorithmus für den doppelten Shift 22
2.3 Bulge Chasing Algorithmus für den vierfachen Shift 28
2.4 Algorithmus Inverse Iteration 30
2.5 Algorithmus LU Zerlegung 32
3.1 Algorithmus für ein symplektisches Lanczos-Verfahren 38
3.2 IRSL Algorithmus mit Doppelshift 41
3.3 IRSL Algorithmus mit Quadrupelshift 43
4.1 Symplektischer Algorithmus für $y^T Q = \alpha e_{2k}^T$ 49
4.2 Zeilenweiser JHESS Algorithmus 50
4.3 Krylov-Schur-artiger Algorithmus 56
5.1 Klassisches symplektisches Gram-Schmidt-ähnliches Verfahren 66
5.2 Modifiziertes symplektisches Gram-Schmidt-ähnliches Verfahren 66
6.1 Berechnung des Residuums mittels Inverser Iteration 73

Kapitel 1

Motivation und grundlegende Begriffe

1.1 Motivation

In vielen Anwendungen ist es von großer Bedeutung, die Eigenwerte, der dem Problem zugeordneten Matrix $A \in \mathbb{R}^{n,n}$ zu finden. Ein Eigenwert ist eine reelle oder komplexe Zahl λ für die gilt, dass $Ax = \lambda x$, wobei $x \neq 0$ und $x \in \mathbb{R}^n$. Die Matrizen, die der Anwendung zu Grunde liegen, sind oft durch eine schwache Besetzungsstruktur gekennzeichnet. Das bedeutet, dass die Anzahl nz der Nichtnullelemente sehr klein ist im Vergleich zur Gesamtanzahl der Elemente n^2. Das Bestimmen aller Eigenwerte einer Matrix ist mit hohem Rechenaufwand verbunden und für die Lösung bestimmter Probleme überhaupt nicht notwendig. Ist man also an einer kleinen Zahl von Eigenwerten der Ausgangsmatrix interessiert, dann werden spezielle Verfahren verwendet, um diese auf möglichst effiziente Weise zu erhalten. Ein wichtiges Verfahren in diesem Zusammenhang ist das Arnoldi-Verfahren mit impliziten Restarts, welches 1992 von D. Sorensen in [31] vorgestellt wurde. Häufig besitzen die Matrizen, an deren Eigenwerten man interessiert ist, eine bestimmte Struktur, die nicht vom Arnoldi-Algorithmus mit impliziten Restarts berücksichtigt wird. Ein Algorithmus mit impliziten Restarts, der diese Struktur beachtet und gleichzeitig im Sinne des Sorensen'schen Algorithmus' funktioniert, birgt viele Vorteile. Das Erhalten der Struktur kann Stabilitätsvorteile bieten und ermöglicht eine Reduzierung des Rechenaufwands im Vergleich zu einem Verfahren, dass die spezielle Matrixstruktur nicht ausnutzt. Diese Arbeit beschäftigt sich mit einem solchen strukturerhaltenen Verfahren für die Klasse der Hamiltonischen Matrizen, die im Abschnitt (1.2) definiert werden. Das nachstehende Beispiel verdeutlicht die praktische Relevanz von Hamiltonischen Matrizen. Es ist der Kontrolltheorie entlehnt und wird durch einige für diese mathematische Disziplin typische Begriffe unterstützt, zu deren Erläuterung auf die Literatur zur Kontrolltheorie, beispielsweise [28], verwiesen wird.

Beispiel 1.1.1. Für die Bauern des Mittleren Westens der USA und Kanadas ist das Fahren von Traktoren eine ziemlich eintönige Angelegenheit. Nicht genug, dass viele Fahrer einschlafen und sich verletzen, es kommt dadurch auch zu erheblichen Schäden auf den Feldern und an den Bewässerungsanlagen. Ein solcher Traktor, dargestellt in Abbildung 1.1, wird mit einem sehr präzisen GPS [1] System ausgestattet. Um diesen Traktor optimal zu steuern, ist es notwendig, ein mathematisches Modell seiner Bewegungen zu verwenden. Dabei werden die nichtlinearen Bewegungsgleichungen benutzt.

[1] Das **G**lobal **P**ositioning **S**ystem wird vom U.S. Verteidigungsministerium betrieben

Abbildung 1.1: Ein Traktor der Marke John Deere ausgestattet mit speziellem GPS System

Um allerdings eine praktikable Möglichkeit zu besitzen, Einfluss auf das Verhalten des Traktors zu nehmen, ist es notwendig, Vereinfachungen durchzuführen. Dazu wird das mit den Bewegungsgleichungen aufgestellte Modell linearisiert und man erhält

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned}, \tag{1.1}$$

wobei $A \in \mathbb{R}^{n,m}$,$B \in \mathbb{R}^{n,m}$ und $C \in \mathbb{R}^{p,n}$. Das System (1.1) wird autonomes linear-quadratisches Optimalsteuerungsproblem oder linear-quadratisches Reglerproblem genannt. Dabei ist x der Zustand, u die Steuerung und y der Ausgang. Es muss nun eine optimale Steuerung u gefunden werden, die unter den Bedingungen von (1.1) das folgende Kostenfunktional minimiert:

$$\mathcal{J}(u) = \int_0^\infty (y^T R_1 y + u^T R_2 u) dt, \tag{1.2}$$

wobei $R_1 > 0$ und $R_2 > 0$. Die optimale Steuerung dieses Problems ist

$$u_* = -R_2^{-1} B^T X_* x.$$

Hier ist X_* die Lösung der folgenden Gleichung

$$A^T X + XA + C^T R_1 C - XBR_2^{-1} B^T X = 0. \tag{1.3}$$

Definiert man zwei neue Matrizen Q und G mit $C^T R_1 C = Q$ und $BR_2^{-1}B^T = G$, dann erhält man die folgende algebraische Riccati-Gleichung (CARE[2]):

$$Q + A^T X + XA - XGX = 0, \tag{1.4}$$

dabei sind $X, G, Q, A \in \mathbb{R}^{n,n}$ und $Q = Q^T$, sowie $G = G^T$. Weiterhin gilt $\dot{x} = Ax + Bu$ und es folgt mit der optimalen Steuerung $u_* = -R_2^{-1} B^T X_* x$:

$$\begin{aligned} \dot{x} &= Ax + Bu_* \\ &= (A - BR_2^{-1} B^T X_*) x \\ &= (A - GX_*) x \end{aligned}.$$

Unter der Bedingung, dass alle Eigenwerte von $(A - GX_*)$ in der linken Halbebene liegen, also $\sigma(A - GX_*) \subset \mathbb{C}^-$, ist X_* eine stabilisierende Lösung von (1.4). Mit der

[2]aus dem Englischen: **C**ontinous-time **A**lgebraic **R**iccati **E**quation

folgenden Aussage wird die Verbindung von optimaler Steuerung und Hamiltonischem Eigenwertproblem hergestellt. Eine Lösung X ist stabilisierend, wenn gilt

$$\left.\begin{array}{rcccl} H\begin{bmatrix} I_n \\ -X \end{bmatrix} & = & \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix}\begin{bmatrix} I_n \\ -X \end{bmatrix} & = & \begin{bmatrix} I_n \\ -X \end{bmatrix}(A-GX) \\ & & \sigma(A-GX) & = & \sigma(H)\cap\mathbb{C}^- \end{array}\right\} \tag{1.5}$$

und weiterhin ist $X_* = -ZY^{-1}$ stabilisierend, wenn die folgende Bedingung erfüllt ist:

$$\left.\begin{array}{rcccl} H\begin{bmatrix} Y \\ Z \end{bmatrix} & = & \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix}\begin{bmatrix} Y \\ Z \end{bmatrix} & = & \begin{bmatrix} Y \\ Z \end{bmatrix}\Lambda \\ & & \sigma(\Lambda) & = & \{\lambda : \lambda \in \sigma(H)\}\cap\mathbb{C}^- \end{array}\right\} \tag{1.6}$$

Es muss span $\left(\begin{bmatrix} Y \\ Z \end{bmatrix}\right)$ ein H-invarianter Unterraum sein, korrespondierend zu den Eigenwerten in der linken Halbebene. Das Problem der optimalen Steuerung eines Traktors hat also auf ein Eigenwertproblem bezüglich der Matrix H, die, wie im nächsten Abschnitt erläutert, von Hamiltonischer Struktur ist, geführt. Das Lösen eines solchen Hamiltonischen Eigenwertproblems soll das Thema dieser Arbeit sein und im Folgenden ausführlich erläutert werden.

1.2 Grundlegende Begriffe

Dieses Kapitel soll der Beschreibung der Begriffe dienen, die im weiteren Verlauf dieser Arbeit von ständigem Gebrauch sind.
Üblicherweise wird die Einheitsmatrix der Größe $n \times n$ mit I^n bezeichnet. Als J^n definieren wir die folgende Matrix:

$$J^n = \begin{bmatrix} 0 & I^n \\ -I^n & 0 \end{bmatrix} \in \mathbb{R}^{2n,2n}.$$

In den meisten Fällen wird aus dem Kontext klar sein welche Dimension die Matrix J^n hat und der Index vernachlässigt. Für die Inverse gilt die einfache Beziehung: $J^{-1} = J^T = -J$. Diese Matrix hilft uns bei den Definitionen für Hamiltonische und symplektische Matrizen.

Definition 1.2.1. *Eine Matrix $H \in \mathbb{R}^{2n,2n}$ heißt Hamiltonische Matrix, wenn gilt*

$$(JH)^T = JH.$$

Definition 1.2.2. *Eine Matrix $S \in \mathbb{R}^{2n,2n}$ heißt symplektische Matrix, wenn gilt*

$$S^T JS = J.$$

Eigenschaften:

1. Eine Matrix H ist genau dann Hamiltonisch, wenn

$$H = \begin{bmatrix} A & G \\ Q & -A^T \end{bmatrix} \in \mathbb{R}^{2n,2n},$$

wobei $G = G^T$ und $Q = Q^T$.

2. Die Inverse einer symplektischen Matrix ist $S^{-1} = J^T S^T J$.

3. $S^{-1}HS$ ist, wenn H Hamiltonisch und S symplektisch sind, eine Hamiltonische Matrix.

4. Wenn y ein Rechtseigenvektor zum Eigenwert λ ist, dann ist $(Jy)^T$ ein Linkseigenvektor zum Eigenwert $-\lambda$, wie man anhand der folgenden Rechnung sieht:

$$\begin{aligned}(Jy)^T H &= y^T J^T H \\ &= -y^T JH \\ &= -y^T (JH)^T \\ &= -y^T H^T J^T \\ &= -(Hy)^T J^T \\ &= -(\lambda y)^T J^T \\ &= -\lambda y^T J^T \\ &= -\lambda (Jy)^T\end{aligned}$$

.

Definition 1.2.3. *1. Eine Matrix*

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \in \mathbb{R}^{2n,2n}$$

heißt J-Hessenberg-Matrix, wenn H_{11}, H_{21}, H_{22} obere Dreiecksmatrizen sind und H_{12} eine obere Hessenbergmatrix.

2. Eine J-Hessenberg-Matrix

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \in \mathbb{R}^{2n,2n},$$

heißt unreduziert, wenn H_{21} nicht singulär und H_{12} eine unreduzierte Hessenbergmatrix ist,also alle Subdiagonaleinträge von Null verschieden sind.

3. Eine Matrix

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \in \mathbb{R}^{2n,2n}$$

heißt J-Dreiecksmatrix, wenn $H_{11}, H_{12}, H_{21}, H_{22}$ obere Dreiecksmatrizen sind und zusätzlich H_{21} nur Nullen auf der Subdiagonalen besitzt.

4. Eine Matrix

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \in \mathbb{R}^{2n,2n}$$

heißt J-Tridiagonalmatrix, wenn H_{11}, H_{21}, H_{22} Diagonalmatrizen sind und H_{22} eine Tridiagonalmatrix ist.

Bemerkung 1.2.4. *Aus der Definition für J-Hessenberg-Matrizen folgt, dass eine Hamiltonische J-Hessenberg-Matrix eine J-Tridiagonalmatrix ist. Dieses ist leicht einzusehen, wenn man die Eigenschaft* 1. *benutzt.*

Mit P sei die 'perfect shuffle' Permutation $P = [e_1, e_3, e_5, \cdots, e_{2n-1}, e_2, e_4, e_6, \cdots, e_{2n}]$ bezeichnet, dabei ist e_i der i-te Einheitsvektor. Für viele der folgenden Betrachtungen

ist es nützlich, zu permutierten Varianten der Matrizen überzugehen. Am Beispiel der Matrix $A \in \mathbb{R}^{2n\times 2n}$ wird die Wirkung der Permutation P erläutert. So erhält man

$$PAP^T = \begin{bmatrix} a_{1,1} & a_{1,n+1} & a_{1,2} & a_{1,n+2} & \cdots & a_{1,n} & a_{1,2n} \\ a_{n+1,1} & a_{n+1,n+1} & a_{n+1,2} & a_{n+1,n+2} & \cdots & a_{n+1,n} & a_{n+1,2n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n,1} & a_{n,n+1} & a_{n,2} & a_{n,n+2} & \cdots & a_{n,n} & a_{n,2n} \\ a_{2n,1} & a_{2n,n+1} & a_{1,2} & a_{2n,n+2} & \cdots & a_{2n,n} & a_{2n,2n} \end{bmatrix} \in \mathbb{R}^{2n,2n}.$$

Aus der Hamiltonischen J-Hessenberg-Matrix H,

$$H = \left[\begin{array}{ccccc|ccccc} \delta_1 & & & & & \beta_1 & \zeta_2 & & & \\ & \delta_2 & & & & \zeta_2 & \beta_3 & \zeta_3 & & \\ & & \delta_3 & & & & \zeta_3 & \ddots & \ddots & \\ & & & \ddots & & & & \ddots & \ddots & \zeta_n \\ & & & & \delta_n & & & & \zeta_n & \beta_n \\ \hline \nu_1 & & & & & -\delta_1 & & & & \\ & \nu_2 & & & & & -\delta_2 & & & \\ & & \nu_3 & & & & & -\delta_3 & & \\ & & & \ddots & & & & & \ddots & \\ & & & & \nu_n & & & & & -\delta_n \end{array}\right] \in \mathbb{R}^{2n,2n},$$

läßt sich durch eine Ähnlichkeitstransformation mit P eine obere Hessenbergmatrix

$$PHP^T = \left[\begin{array}{cc|cc|cc|cc} \delta_1 & \beta_1 & 0 & \zeta_2 & & & & \\ \nu_1 & -\delta_1 & 0 & 0 & & & & \\ \hline 0 & \zeta_2 & \delta_2 & \beta_2 & 0 & \zeta_3 & & \\ 0 & 0 & \nu_2 & -\delta_2 & 0 & 0 & & \\ \hline & & 0 & \zeta_3 & \ddots & & \ddots & \\ & & 0 & 0 & & \ddots & & \ddots \\ \hline & & & & \ddots & & \ddots & \\ & & & & & \ddots & & \ddots \\ \hline & & & & & & 0 & \zeta_n \\ & & & & & & 0 & 0 \\ \hline & & & & & & \delta_n & \beta_n \\ & & & & & & \nu_n & -\delta_n \end{array}\right] \in \mathbb{R}^{2n,2n}. \tag{1.7}$$

erhalten. Auch die Matrix J wird später in permutierter Form benötigt, welche wie folgt aussieht:

$$PJP^T = \left[\begin{array}{cc|cc|cc|cc} 0 & 1 & & & & & & \\ -1 & 0 & & & & & & \\ \hline & & 0 & 1 & & & & \\ & & -1 & 0 & & & & \\ \hline & & & & \ddots & & & \\ & & & & & \ddots & & \\ \hline & & & & & & 0 & 1 \\ & & & & & & -1 & 0 \end{array}\right] \in \mathbb{R}^{2n,2n}.$$

Um die Algorithmen vergleichen zu können, ist es wichtig, sich mit dem Aufwand des jeweiligen Verfahrens zu beschäftigen. Ein Maß für den Aufwand ist das Zählen der

floating point operations kurz flops, (vgl.[16]). Beispielsweise benötigt der Ausdruck $y = Ax + y$ mit $A \in \mathbb{R}^{m,n}$ $2nm$ flops.
Die Vektornorm $\|.\|_2$ wird im Verlaufe der Arbeit nur mit $\|.\|$ bezeichnet. Sollte die Verwendung von anderen Normen nötig sein, so werden diese mit einem Index versehen.

Kapitel 2

Der SR Algorithmus

2.1 SR Zerlegung

Für die impliziten Restarts nicht-Hamiltonischer Eigenwertprobleme spielt die Zerlegung $H - \mu I = QR$ eine immense Rolle. Leider ist der QR Algorithmus für Hamiltonische Eigenwertprobleme von keinem großen Nutzen, da er die spezielle Struktur dieser Matrizen nicht berücksichtigt. Aus diesem Grund ist es notwendig, eine QR ähnliche Zerlegung zu finden, die die Besonderheiten des Hamiltonischen Eigenwertproblems beachtet. Ein Algorithmus, der diesen Anforderungen genügt, wird in [8] eingeführt und soll an dieser Stelle zusammenfassend vorgestellt werden.

Definition 2.1.1. *Sei $A \in \mathbb{R}^{2n,2n}$. Die Zerlegung $A = SR$, mit einer symplektischen Matrix $S \in \mathbb{R}^{2n\times 2n}$ und einer J-Dreiecks-Matrix R, heißt SR Zerlegung der Matrix A.*

Viele Eigenschaften der SR Zerlegung kann man dem folgenden, aus [5] übernommenen, Satz entnehmen.

Satz 2.1.2. *Sei X eine $2n \times 2n$ nichtsinguläre Matrix. Dann gilt:*

1. *Eine Zerlegung $X = SR$, mit S und R wie in Definition 2.1.1,existiert genau dann, wenn alle Hauptminoren gerader Dimension von PX^TJXP^T von Null verschieden sind.*

2. *Sind $X = SR$ und $X = \tilde{S}\tilde{R}$ zwei SR Zerlegungen, dann existiert eine Matrix*

$$D = \begin{bmatrix} C & F \\ 0 & C^{-1} \end{bmatrix},$$

   wobei $C = diag(c_1, \cdots, c_n)$ und $F = diag(f_1, \cdots, f_n)$, so dass $\tilde{S} = SD^{-1}$ und $\tilde{R} = DR$

3. *Sei $X = \tilde{H}$ eine unreduzierte Hamiltonische J-Hessenberg-Matrix. Wenn $\tilde{H} - \mu I = SR$, mit $\mu \in \mathbb{R}$, und S und R, die die Voraussetzungen von (2.1.1) erfüllen, existieren, dann ist $\hat{H} = S^{-1}\tilde{H}S = RS + \mu I$ eine Hamiltonische J-Hessenberg-Matrix.*

4. *Wenn μ in 3. ein Eigenwert von $\tilde{H}$ ist, dann gilt $\hat{h}_{2k,2k} = \mu$, $\hat{h}_{k,k} = -\mu$ und $\hat{h}_{2k,k} = 0$.*

5. *Sei $X = \tilde{H}$ eine unreduzierte Hamiltonische J-Hessenberg-Matrix. Wenn die Zerlegung $(\tilde{H} - \mu I)(\tilde{H} + \mu I) = SR$ existiert, mit $\mu \in \mathbb{R}$ oder $\mu \in i\mathbb{R}$, und S sowie R nach Definition 2.1.1, dann ist $\hat{H} = S^{-1}\tilde{H}S$ eine Hamiltonische J-Hessenberg-Matrix.*

6. *Wenn μ aus* 5. *ein Eigenwert von $\tilde{H}$ ist, dann gilt: $\hat{h}_{k,2k-1} = h_{k-1,2k} = 0$ und die 2×2 Matrix*

$$\begin{bmatrix} \hat{h}_{k,k} & \hat{h}_{k,2k} \\ \hat{h}_{2k,k} & \hat{h}_{2k,2k} \end{bmatrix}$$

hat die Eigenwerte μ und $-\mu$.

7. *Sei $X = \tilde{H}$ eine unreduzierte Hamiltonische J-Hessenberg-Matrix. Wenn die Zerlegung*

$$(\tilde{H} - \mu_1 I)(\tilde{H} + \mu_1 I)(\tilde{H} - \mu_2 I)(\tilde{H} + \mu_2 I) = SR$$

existiert, mit $\mu_1, \mu_2 \in \mathbb{C}$ oder $\mu_1, \mu_2 \in \mathbb{R}$, und S sowie R nach Definition 2.1.1, dann ist $\hat{H} = S^{-1}\tilde{H}S$ eine Hamiltonische J-Hessenberg-Matrix.

8. *Wenn μ_1 und μ_2 aus* 7. *Eigenwerte von $\tilde{H}$ sind, dann gilt: $\hat{h}_{k-1,2k-2} = h_{k-2,2k-1} = 0$ und die 4×4 Matrix*

$$\begin{bmatrix} \hat{h}_{k-1,k-1} & & \hat{h}_{k-1,2k-1} & \hat{h}_{k-1,2k} \\ & \hat{h}_{k,k} & \hat{h}_{k,2k-1} & \hat{h}_{k,2k} \\ \hat{h}_{2k-1,k-1} & & \hat{h}_{2k-1,2k-1} & \\ & \hat{h}_{2k,k} & & \hat{h}_{2k,2k} \end{bmatrix}$$

hat die Eigenwerte μ_1,$-\mu_1$,μ_2 und $-\mu_2$.

Beweis.

- Der Beweis zu 1. ist in Theorem 11 aus [12] nachzulesen.
- Beweise zu den Aussagen 2., 3. und 5. sind in [8] zu finden.
- Der Beweis für 4. folgt dem Beweis des Theorems 2 aus [17]. Es wird angenommen, dass die SR Zerlegung für $\tilde{H} - \mu I$ existiert, das heißt $\tilde{H} - \mu I = SR$. Nützlich ist es an dieser Stelle, zu den permutierten Versionen der Matrizen überzugehen, dabei wird die permutierte Matrix den Index p tragen. Man erhält $\tilde{H}_p = P\tilde{H}P^T$, $S_p = PSP^T$ und $R_p = PRP^T$. Da $\tilde{H}$ eine unreduzierte J-Hessenberg-Matrix ist, ist $\tilde{H}_p$ eine unreduzierte obere Hessenbergmatrix. Aus der J-Dreiecksmatrix R wird mittels der Permutation P eine obere Dreiecksmatrix. Die permutierten Matrizen werden nun in der folgenden Weise zerlegt:

$$\begin{aligned} \tilde{H}_p &= \left[\begin{array}{c|cc} \tilde{H}_p^{2k,2k-2} & \tilde{h}_p^{2k-1} & \tilde{h}_p^{2k} \end{array}\right] \\ S_p &= \left[\begin{array}{c|cc} S_p^{2k,2k-2} & s_p^{2k-1} & s_p^{2k} \end{array}\right] \\ R_p &= \left[\begin{array}{c|cc} R_p^{2k-2,2k-2} & r_p^{2k-1} & r_p^{2k} \\ \hline 0 & (r_p)_{2k-1,2k-1} & (r_p)_{2k-1,2k} \\ 0 & (r_p)_{2k,2k-1} & (r_p)_{2k,2k} \end{array}\right]. \end{aligned}$$

  Die Spalten von $\tilde{H}_p^{2k,2k-2} - \mu I^{2k,2k-2}$, wobei $I^{2k,2k-2}$ die ersten $k-2$ Spalten der Einheitsmatrix repräsentiert, sind auf Grund der Unreduziertheit von $\tilde{H}_p$ linear unabhängig. Da S_p nicht singulär ist, folgt die Unabhängigkeit der Spalten von $S_p^{2k,2k-2}$. Mit

$$\tilde{H}_p^{2k,2k-2} - \mu I^{2k,2k-2} = S_p^{2k,2k-2} R_p^{2k-2,2k-2}$$

  ergibt sich, dass auch $R_p^{2k-2,2k-2}$ nicht singulär ist. Stellt man die gerade erwähnte Beziehung nach $S_p^{2k,2k-2}$ um, dann erhält man:

$$(\tilde{H}_p^{2k,2k-2} - \mu I^{2k,2k-2})(R_p^{2k-2,2k-2})^{-1} = S_p^{2k,2k-2}.$$

$S_p^{2k,2k-2}$ ist somit eine obere Hessenbergmatrix und $\tilde{H}_p^{2k,2k-2} - \mu I^{2k,2k-2}$ ist genau dann singulär, wenn $(r_p)_{2k,2k} = 0$ ist. Wenn also $\tilde{H}_p - \mu I$ singulär ist, dann ist $(r_p)_{2k,2k} = 0$. Aus diesem Grund wird das $(2k-1, 2k)$-te Element von $\hat{H}_p = P(RS - \mu I)P^T = PS^{-1}\tilde{H}SP^T$ zu Null. Das bedeutet, dass die Matrix $\hat{H}_p$ die folgende Gestalt hat:

$$\hat{H}_p = \left[\begin{array}{cc|cc|cc|cc} \hat{\delta}_1 & \hat{\beta}_1 & 0 & 0 & & & & \\ \hat{\nu}_1 & -\hat{\delta}_1 & 0 & \hat{\zeta}_2 & & & & \\ \hline 0 & \hat{\zeta}_2 & \hat{\delta}_2 & \hat{\beta}_2 & 0 & 0 & & \\ 0 & 0 & \nu_2 & -\hat{\delta}_2 & 0 & \hat{\zeta}_3 & & \\ \hline & & 0 & \hat{\zeta}_3 & \ddots & & \ddots & \\ & & 0 & 0 & & \ddots & & \ddots \\ \hline & & & & \ddots & & \ddots & & 0 & \hat{\zeta}_k \\ & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & 0 & \hat{\zeta}_k & -\mu & \hat{\beta}_k \\ & & & & & & 0 & 0 & 0 & \mu \end{array}\right],$$

also die Parameter $\hat{\delta}_k = -\mu$ und $\hat{\nu}_k = 0$ sind.

- Der Beweis für 6. kann auf analoge Weise geführt werden. Es existiere also die Zerlegung $(H - \mu I)(H + \mu I) = SR$. Das folgende Aufspalten der permutierten Matrizen ist wieder hilfreich:

$$\begin{aligned} \tilde{H}_p^2 &= \left[\begin{array}{c|cc} \left(\tilde{H}_p^2\right)^{2k,2k-2} & \left(\tilde{h}_p^2\right)^{2k-1} & \left(\tilde{h}_p^2\right)^{2k} \end{array}\right] \\ S_p &= \left[\begin{array}{c|cc} S_p^{2k,2k-2} & s_p^{2k-1} & s_p^{2k} \end{array}\right] \\ R_p &= \left[\begin{array}{c|cc} R_p^{2k-2,2k-2} & r_p^{2k-1} & r_p^{2k} \\ \hline 0 & (r_p)_{2k-1,2k-1} & (r_p)_{2k-1,2k} \\ 0 & (r_p)_{2k,2k-1} & (r_p)_{2k,2k} \end{array}\right]. \end{aligned}$$

Die Spalten von $\left(\tilde{H}_p^2\right)^{2k,2k-2} - \mu I^{2k,2k-2}$ sind auf Grund der Unreduziertheit von $\tilde{H}_p$ linear unabhängig. Ebenso sind die Spalten von $S_p^{2k,2k-2}$ unabhängig, da die Matrix S_p nicht singulär ist. Aus der Beziehung

$$\tilde{H}_p^{2k,2k-2} - \mu I^{2k,2k-2} = S_p^{2k,2k-2} R_p^{2k-2,2k-2}$$

ergibt sich, dass $R_p^{2k-2,2k-2}$ regulär ist. Dieses Resultat verwendend ist zu sehen, dass S_p eine obere Hessenbergmatrix mit einer zusätzlichen Nebendiagonale ist und das $(H-\mu I)(H+\mu I)$ genau dann singulär ist, wenn der 2×2 Block im Bereich rechts unten Null ist. Sei also nun μ ein Eigenwert, dann verschwinden die k-te und $2k$-te Zeile von R. Ein Koeffizientenvergleich von $\hat{H}_p^2 = RS + \mu^2 I$ erbringt das gewünschte Resultat, da $\nu_{k-1} \neq 0$ und die oben gemachten Voraussetzungen an S_p und $R_p^{2k-2,2k-2}$ erfüllt sind.

- Der Beweis für 8. kann auf analoge Weise geführt werden. Es existiere also die Zerlegung $(H - \mu_1 I)(H + \mu_1 I)(H - \mu_2 I)(H + \mu_2 I) = SR$. Die permutierten

Matrizen werden wie folgt aufgeteilt:

$$\tilde{H}_p^4 = \left[\left(\tilde{H}_p^4\right)^{2k,2k-4} \,\middle|\, \left(\tilde{h}_p^4\right)^{2k-3} \quad \left(\tilde{h}_p^4\right)^{2k-2} \quad \left(\tilde{h}_p^4\right)^{2k-1} \quad \left(\tilde{h}_p^4\right)^{2k} \right]$$

$$\tilde{H}_p^2 = \left[\left(\tilde{H}_p^2\right)^{2k,2k-4} \,\middle|\, \left(\tilde{h}_p^2\right)^{2k-3} \quad \left(\tilde{h}_p^2\right)^{2k-2} \quad \left(\tilde{h}_p^2\right)^{2k-1} \quad \left(\tilde{h}_p^2\right)^{2k} \right]$$

$$S_p = \left[\, S_p^{2k,2k-4} \,\middle|\, s_p^{2k-3} \quad s_p^{2k-2} \quad s_p^{2k-1} \quad s_p^{2k} \, \right]$$

$$R_p = \left[\begin{array}{c|cccc} R_p^{2k-4,2k-4} & r_p^{2k-3} & r_p^{2k-2} & r_p^{2k-1} & r_p^{2k} \\ \hline 0 & (r_p)_{2k-3,2k-3} & (r_p)_{2k-3,2k-2} & (r_p)_{2k-3,2k-1} & (r_p)_{2k-3,2k} \\ 0 & 0 & (r_p)_{2k-2,2k-2} & (r_p)_{2k-2,2k-1} & (r_p)_{2k-2,2k} \\ 0 & 0 & 0 & (r_p)_{2k-1,2k-1} & (r_p)_{2k-1,2k} \\ 0 & 0 & 0 & 0 & (r_p)_{2k,2k} \end{array}\right].$$

Die Spalten von

$$\left(\tilde{H}_p^4\right)^{2k,2k-4} - (\mu_1^2 + \mu_2^2)\left(H_p^2\right)^{2k,2k-4} + \mu_1^2\mu_2^2 I^{2k,2k-4}$$

sind auf Grund der Unreduziertheit von $\tilde{H}_p$ linear unabhängig. Ebenfalls sind die Spalten von $S_p^{2k,2k-4}$ unabhängig, da die Matrix S_p nicht singulär ist. Aus der Beziehung

$$\left(\tilde{H}_p^4\right)^{2k,2k-4} - (\mu_1^2 + \mu_2^2)\left(H_p^2\right)^{2k,2k-4} + \mu_1^2\mu_2^2 I^{2k,2k-4} = S_p^{2k,2k-4} R_p^{2k-4,2k-4}$$

folgt die Invertierbarkeit von $R_p^{2k-4,2k-4}$. Benutzt man die Eigenschaft, dass $R_p^{2k-4,2k-4}$ nicht singulär ist, dann zeigt sich, dass S_p eine obere Hessenbergmatrix mit drei zusätzlichen Nebendiagonalen ist und das

$$(H - \mu_1 I)(H + \mu_1 I)(H - \mu_2 I)(H + \mu_2 I)$$

genau dann singulär ist, wenn der verbleibende 4×4 Block verschwindet. Sind also μ_1 und μ_2 Eigenwerte von $\tilde{H}$, dann sind die Zeilen und Spalten $k-1$,k ,$2k-1$ und $2k$ von R gleich Null. Ein Koeffizientenvergleich, unter den gemachten Voraussetzungen an die Matrizen H_p, S_p und R_p, von

$$\hat{H}^4 - (\mu_1^2 + \mu_2^2)\hat{H}^2 = RS - \mu_1^2\mu_2^2 I$$

liefert die Richtigkeit der in 8. gemachten Aussage.

□

2.2 Transformationsmatrizen

Für die Umsetzung des SR Algorithmus' sind Transformationsmatrizen notwendig, mit denen die Ausgangsmatrix in eine symplektische Matrix S und eine J-Dreiecksmatrix R zerlegt werden kann. Dabei sollen die symplektischen Transformationen in der Matrix S akkumuliert werden. Eine genaue Beschreibung der notwendigen Transformationsmatrizen soll der nun folgende Abschnitt liefern.

Definition 2.2.1. *1. Für $k \in \{1, \cdots, n\}$ und $c, s \in \mathbb{R}$ mit $c^2 + s^2 = 1$ wird die Matrix*

$$J(k, c, s) = \begin{bmatrix} C & S \\ -S & C \end{bmatrix}$$

als symplektische Givensmatrix vom Typ I bezeichnet. Hierbei sind $C, S \in \mathbb{R}^{n,n}$ Diagonalmatrizen und es gilt $C = I_n + (c-1)e_k e_k^T$ und $S = s e_k e_k^T$.

2. Für $k \in \{1, \cdots, n\}$ und $c, s \in \mathbb{R}$ mit $c^2 + s^2 = 1$ wird die Matrix

$$R(k, c, s) = \begin{bmatrix} U(k, c, s) & 0 \\ 0 & U(k, c, s) \end{bmatrix} \in \mathbb{R}^{2n,2n}$$

als symplektische Givensmatrix vom Typ II bezeichnet, wobei

$$U(k, c, s) = diag(I_{k-1}, \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, I_{n-k-1}).$$

3. Für $k \in \{2, \cdots, n\}$ und $y \in \mathbb{R}$ heißt die Matrix

$$G(k, y) = \begin{bmatrix} D & Y \\ 0 & D^{-1} \end{bmatrix} \in \mathbb{R}^{2n,2n}$$

symplektische Gaußmatrix. Dabei ist die Matrix $Y \in \mathbb{R}^{n,n}$ von folgender Gestalt,

$$Y = \left(\frac{y}{(1+y^2)^{\frac{1}{4}}} \right) (e_{k-1} e_k^T + e_k e_{k-1}^T)$$

und D ebenfalls ein reelle $n \times n$ Matrix, mit

$$D = I_n + \left(\frac{1}{(1+y^2)^{\frac{1}{4}}} - 1 \right) (e_{k-1} e_k^T + e_k e_{k-1}^T).$$

4. Sei $k \in \{2, \cdots, n\}$ und $w \in \mathbb{R}^{n-k+1}$, dann heißt die Matrix

$$W(k, l, w) = \begin{bmatrix} V(k, l, w) & 0 \\ 0 & V(k, l, w) \end{bmatrix} \in \mathbb{R}^{2n,2n}$$

symplektische Householder Matrix, mit

$$V(k, l, w) = diag(I_{k-1}, I_l - \frac{2}{w^T w} w w^T, I_{n-l-k+1}).$$

Bemerkung 2.2.2. *1. $J(k, c, s)$ ist orthogonal und symplektisch.*

2. $J(k, c, s)$ ist eine Givensrotation in den Ebenen k und $n + k$.

3. $R(k, c, s)$ ist orthogonal und symplektisch.

4. $R(k, c, s)$ ist die direkte Summe zweier $n \times n$ Givensrotationen.

5. $H(k, l, w)$ ist orthogonal und symplektisch.

6. $H(k, l, w)$ ist die direkte Summe zweier $n \times n$ Householdermatrizen.

7. $G(k, y)$ ist symplektisch und nichtorthogonal.

8. $G(k,y)^{-1} = \begin{bmatrix} D^{-1} & -Y \\ 0 & D \end{bmatrix}$

9. *Es gilt* $cond_2(G(k,y)) = (1+y^2)^{\frac{1}{2}} + |y|$, *wobei* $cond_2(A) = \|A\|\|A^{-1}\|$

10. *Aus [8] geht hervor, dass die Matrix* $G(k,y)$ *unter allen Matrizen gleicher Funktionalität die kleinste Konditionszahl besitzt.*

Wenn man die Funktionsweise der Transformationsmatrizen betrachtet, kann man die Wirkung auf folgende Art darstellen:

- $J(k,c,s)(\alpha e_k + \beta e_{n+k}) = \gamma e_k$ mit $1 \leq k \leq n$,
- $U(k,c,s)(\alpha e_k + \beta e_{k+1}) = \gamma e_k$ mit $1 \leq k \leq n-1$,
- $G(k,y)(\alpha e_k + \beta e_{n+k-1}) = \gamma e_{n+k-1}$ mit $1 \leq k \leq n$.

Eine algorithmische Umsetzung der Transformationsroutinen und des kompletten SR Algorithmus' ist in [8] zu finden.

2.3 Implizite Shifts

An dieser Stelle soll zu den permutierten Varianten der Matrizen, also S_p, $\tilde{H}_p$ und R_p, übergegangen werden. Aus diesem Grund ist es notwendig, auch die Transformationsmatrizen im permutierten Zustand zu betrachten. Man erhält

$$G_p(k,y) = PG(k,y)P^T = diag(I_{2k-4}, \left[\begin{array}{cc|cc} r & & t & \\ & \frac{1}{r} & & \\ \hline & t & r & \\ & & & \frac{1}{r} \end{array}\right], I_{2n-2k})$$

mit $r = (1+y^2)^{-\frac{1}{4}}$, $t = yr$,

$$R_p(k,c,s) = PR(k,c,s)P^T = diag(I_{2k-2}, \left[\begin{array}{cc|cc} c & & s & \\ & c & & s \\ \hline -s & & c & \\ & -s & & c \end{array}\right], I_{2n-2k-2})$$

und

$$J_p(k,c,s) = PJ(k,c,s)P^T = diag(I_{2k-2}, \begin{bmatrix} c & s \\ -s & c \end{bmatrix}, I_{2n-2k}).$$

Wie auch im QR Algorithmus, ist es im SR Algorithmus wichtig, implizite Shifts zu verwenden. Wegen der Eindeutigkeit der Reduktion auf J-Hessenbergform, gilt das Pendant zum *impliziten Q Theorem*, siehe hierzu Theorem 3.3 auf Seite 116 in [33]. Mit solch einem Theorem ist es im Verlaufe des SR Algorithmus' möglich, implizite Shifts durchzuführen. Ein impliziter Shift berechnet die Matrizen S und R, die zum Beispiel aus der Zerlegung eines Doppelshifts $(H-\mu I)(H-\lambda I) = SR$ stammen, ohne dabei die explizite Zerlegung durchzuführen. Eine ausführliche Diskussion der impliziten Shifts ist den Kapiteln 3.3 und 3.4 aus [33] zu entnehmen. Es sollen nun der Einzel-, Doppel- und Quadrupelshift in ihrer impliziten Variante vorgestellt werden.

2.3.1 Einfacher Shift

Der einfache Shift, vorgestellt in [5], ist ein Shift der Art $H_p - \mu I$. Denkt man wieder an das *implizite Q Theorem*, dann ist mit dem Berechnen der ersten Spalte von $H_p - \mu I$ zu beginnen. Berücksichtigt man die Hamiltonische Struktur der Ausgangsmatrix, dann ist vollkommen klar, dass die erste Spalte folgende Gestalt hat: $x = (H_p - \mu I)e_1 = [\delta_1 - \mu, \nu_1, 0, \cdots, 0]$. Der Vektor x muss nun mittels einer symplektischen Transformation auf ein Vielfaches von e_1 gebracht werden. Dieses kann mit einer Matrix vom Typ $J_p(1, c, s)$ durchgeführt werden. Die Matrix $H_p^{(1)} = J_p(1, c, s) H_p J_p^T(1, c, s)$ besitzt nun keine Hamiltonische J-Hessenberg Struktur mehr. Man erhält

$$H_p^{(1)} = \left[\begin{array}{cc|cc|cc|cc|cc} x & x & 0 & x & & & & & & \\ x & x & 0 & \otimes & & & & & & \\ \hline \otimes & x & x & x & 0 & x & & & & \\ 0 & 0 & x & x & 0 & 0 & & & & \\ \hline & & 0 & x & \ddots & & \ddots & & & \\ & & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & \ddots & & \ddots & & 0 & x \\ & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & 0 & x & x & x \\ & & & & & & 0 & 0 & x & x \end{array}\right].$$

Dabei sei x ein beliebiges Matrixelement und $\otimes$ bezeichnet die durch die Ähnlichkeitstransformationen hinzugekommenen Elemente. Da nun in der Matrix ein Buckel (bulge) entstanden ist, müssen wir einen 'bulge chasing' Algorithmus anwenden, der $H_p^{(1)}$ wieder auf die gewünschte Hamiltonische J-Hessenberggestalt bringt. Dieses kann mit dem JHESS Algorithmus aus [8] geschehen, der beliebige Matrizen auf J-Hessenberg-Gestalt bringt. Wenn man vorher die genaue Struktur von $H_p^{(1)}$ berücksichtigt, kann eine immense Vereinfachung des JHESS Verfahrens herbeigeführt werden, da diese Matrix sich nicht an vielen Stellen von einer permutierten J-Hessenberg-Matrix unterscheidet. Wenn wir den Eintrag an Position $(3, 1)$ eliminieren wollen, ohne dabei die Nullen der ersten Spalte zu verlieren, müssen wir eine Matrix vom Typ $G_p(2, y)$ verwenden. Zur Anwendbarkeit dieser Transformation ist es notwendig, dass der Eintrag $(2, 1)$ von Null verschieden ist. Die Notwendigkeit dieser Bedingung wird im Kapitel 2.5 erläutert. $H_p^{(2)} = G_p(2, y) H_p^{(1)} G_p^{-1}(2, y)$ hat dann die folgende Gestalt:

$$H_p^{(2)} = \left[\begin{array}{cc|cc|cc|cc|cc} x & x & \otimes & x & & & & & & \\ x & x & 0 & 0 & & & & & & \\ \hline 0 & x & x & x & 0 & x & & & & \\ 0 & \otimes & x & x & 0 & 0 & & & & \\ \hline & & 0 & x & \ddots & & \ddots & & & \\ & & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & \ddots & & \ddots & & 0 & x \\ & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & 0 & x & x & x \\ & & & & & & 0 & 0 & x & x \end{array}\right].$$

Der Eintrag $(2,4)$ dieser Matrix wurde ebenfalls zu Null, dies ist der Hamiltonischen Struktur von $H_p^{(2)}$ geschuldet. Als nächstes muss der Eintrag $(4,2)$ verschwinden. Hier wird eine Transformation des Typs $J_p(2,c,s)$ verwendet und es ergibt sich

$$H_p^{(3)} = J_p(2,c,s)H_p^{(2)}J_p^T(2,c,s) = \left[\begin{array}{cc|cc|cc|cc|cc} x & x & 0 & x & & & & & & \\ x & x & 0 & 0 & & & & & & \\ \hline 0 & x & x & x & 0 & x & & & & \\ 0 & 0 & x & x & 0 & \otimes & & & & \\ \hline & & \otimes & x & \ddots & & \ddots & & & \\ & & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & \ddots & & \ddots & & 0 & x \\ & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & 0 & x & x & x \\ & & & & & & 0 & 0 & x & x \end{array}\right].$$

Wie man leicht sieht, ist der Buckel von $H_p^{(1)}$ zu $H_p^{(3)}$ genau um zwei Spalten und zwei Zeilen gewandert. Durch Fortsetzen der Transformation von $H_p^{(3)}$ läï¿½ sich der Buckel nach rechts unten aus der Matrix heraus bewegen. Zur Verdeutlichung sei die letzte, noch einen Buckel enthaltene, Matrix H_p^{2n-1} dargestellt:

$$H_p^{(2n-1)} = \left[\begin{array}{cc|cc|cc|cc|cc} x & x & 0 & x & & & & & & \\ x & x & 0 & 0 & & & & & & \\ \hline 0 & x & x & x & 0 & x & & & & \\ 0 & 0 & x & x & 0 & 0 & & & & \\ \hline & & 0 & x & \ddots & & \ddots & & & \\ & & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & \ddots & & \ddots & & \otimes & x \\ & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & 0 & x & x & x \\ & & & & & & 0 & \otimes & x & x \end{array}\right],$$

Der Algorithmus 2.1 liefert eine formale Beschreibung, des oben dargestellten Beispiels.

2.3.2 Doppelshift

Der Doppelshift, ebenso bereits vorgestellt in [5], ist ein Shift der Form $(H_p-\mu_1 I)(H_p-\mu_2 I)$, wobei μ_1 und μ_2 die Eigenwerte der Matrix

$$H_j = \left[\begin{array}{cc} \delta_j & \beta_j \\ \nu_j & -\delta_j \end{array}\right]$$

sind, diese sind entweder beide reell oder rein imaginär. Im komplexen Fall wird durch den Doppelshift garantiert, dass die Rechnung im Reellen bleibt, es entsteht $(H_p - \mu I)(H_p - \bar{\mu} I) = H_p^2 - 2Re(\mu)H_p + |\mu|^2 I$. Eine Diskussion der effizienten Berechnung des Ausdrucks $H_p^2 - 2Re(\mu)H_p + |\mu|^2 I$ ist [33] auf den Seiten 118 ff. zu entnehmen. Auch im Fall des komplexen Doppelshifts gilt es wieder, die erste Spalte der Matrix

Algorithmus 2.1 Bulge Chasing Algorithmus für den einfachen Shift

INPUT : $H_p \in \mathbb{R}^{2k\times 2k}$ und $S_p \in \mathbb{R}^{2n\times 2k}$
OUTPUT : $H_p \in \mathbb{R}^{2k\times 2k}$ und $S_p \in \mathbb{R}^{2n\times 2}$ nach dem Shift

Wähle Shift μ.
Berechne $J_p(1,c,s)$.
$H_p = J_p(1,c,s)H_pJ_p^T(1,c,s)$
$S_p = S_PJ_p^T(1,c,s)$
for $i = 3,5,\cdots,2k-1$ **do**
  Berechne $G_p(\frac{i+1}{2},y)$.
  $H_p = G_p(\frac{i+1}{2},y)H_pG_p^{-1}(\frac{i+1}{2},y)$
  $S_p = S_PG_p^{-1}(\frac{i+1}{2},y)$
  Berechne $J_p(\frac{i+1}{2},y)$.
  $H_p = J_p(\frac{i+1}{2},y)H_pJ_p^T(\frac{i+1}{2},y)$
  $S_p = S_PJ_p^T(\frac{i+1}{2},y)$
end for

$(H_p-\mu_1 I)(H_p-\mu_2 I)$ zu berechnen. Man erhält $x = (H_p-\mu_1 I)(H_p-\mu_2 I)e_1 = (\delta_1^2 + \beta_1\nu_1 - 2Re(\mu)\delta_1 + |\mu|^2, 0, \nu_1\zeta_2, 0, \cdots, 0]^T$. Für das Durchführen eines impliziten Shifts ist es wiederum notwendig, x in ein Vielfaches von e_1 zu überführen. In diesem Fall ist die Transformation $R_p(1,c,s)$ zu wählen. Das Anwenden von $R_p(1,c,s)$ zerstört die Hamiltonische J-Hessenberg-Struktur und $H_p^{(1)} = R_p(1,c,s)H_pR_p^T(1,c,s)$ besitzt einen Buckel, der in einem bulge chasing Algorithmus beseitigt werden muss. Wie man an

$$H_p^{(1)} = R_p(1,c,s)H_pR_p^T(1,c,s) = \left[\begin{array}{cc|cc|cc|cc|cc|cc} x & x & \otimes & x & 0 & \otimes & & & & & & \\ x & x & \otimes & \otimes & 0 & 0 & & & & & & \\ \hline \otimes & x & x & x & 0 & x & 0 & 0 & & & & \\ \otimes & \otimes & x & x & 0 & 0 & 0 & 0 & & & & \\ \hline 0 & \otimes & 0 & x & x & x & 0 & x & & & & \\ 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\ \hline & & 0 & 0 & 0 & x & \ddots & & \ddots & & & \\ & & 0 & 0 & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & & & \ddots & & \ddots & & 0 & x \\ & & & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & & & 0 & x & x & x \\ & & & & & & & & 0 & 0 & x & x \end{array}\right]$$

sieht, gilt es acht zusätzliche Einträge aus der Matrix zu schieben, ohne dabei die vorhandene Struktur zu stören, da das Erhalten dieser in unserem numerischen Algorithmus Priorität hat. Um die Nullen der ersten Spalte beizubehalten und gleichzeitig den Eintrag an der Stelle $(4,1)$ zu beseitigen, ist die Givenstransformation $J_p(3,c,s)$

zu verwenden. Das Resultat ist

$$H_p^{(2)} = J_p(3,c,s)H_p^{(1)}J_p^T(3,c,s) = \left[\begin{array}{cc|cc|cc|cc|cc|cc} x & x & \otimes & x & 0 & \otimes & & & & & & \\ x & x & 0 & \otimes & 0 & 0 & & & & & & \\ \hline \otimes & x & x & x & 0 & x & 0 & 0 & & & & \\ 0 & \otimes & x & x & 0 & \otimes & 0 & 0 & & & & \\ \hline 0 & \otimes & \otimes & x & x & x & 0 & x & & & & \\ 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\ \hline & & 0 & 0 & 0 & x & \ddots & & \ddots & & & \\ & & 0 & 0 & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & & & \ddots & & \ddots & & 0 & x \\ & & & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & & & 0 & x & x & x \\ & & & & & & & & 0 & 0 & x & x \end{array}\right].$$

Auch hier hilft die Hamiltonische Struktur der permutierten Matrix und läßt zusätzlich zu $(4,1)$ auch $(2,3)$ zu Null werden. Der Eintrag $(3,2)$ wird durch $G_p(2,y)$ annulliert, wenn das Element $(2,1)$ von Null verschieden ist. Das Transformieren mit $G_p(2,y)$ liefert

$$H_p^{(3)} = G_p(2,y)H_p^{(2)}G_p^{-1}(2,y) = \left[\begin{array}{cc|cc|cc|cc|cc|cc} x & x & \otimes & x & 0 & \otimes & & & & & & \\ x & x & 0 & 0 & 0 & 0 & & & & & & \\ \hline 0 & x & x & x & 0 & x & 0 & 0 & & & & \\ 0 & \otimes & x & x & 0 & \otimes & 0 & 0 & & & & \\ \hline 0 & \otimes & \otimes & x & x & x & 0 & x & & & & \\ 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\ \hline & & 0 & 0 & 0 & x & \ddots & & \ddots & & & \\ & & 0 & 0 & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & & & \ddots & & \ddots & & 0 & x \\ & & & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & & & 0 & x & x & x \\ & & & & & & & & 0 & 0 & x & x \end{array}\right].$$

Eine symplektische Givensrotation $J_p(2,c,s)$ vom Typ I ist beim Entfernen des Eintrags $(4,2)$ hilfreich und liefert bei einer Ähnlichkeitstransformation folgendes Resultat:

$$H_p^{(4)} = J_p(2,c,s)H_p^{(3)}J_p^T(2,c,s) = \left[\begin{array}{cc|cc|cc|cc|cc|cc} x & x & 0 & x & 0 & \otimes & & & & & & \\ x & x & 0 & 0 & 0 & 0 & & & & & & \\ \hline 0 & x & x & x & 0 & x & 0 & 0 & & & & \\ 0 & 0 & x & x & 0 & \otimes & 0 & 0 & & & & \\ \hline 0 & \otimes & \otimes & x & x & x & 0 & x & & & & \\ 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\ \hline & & 0 & 0 & 0 & x & \ddots & & \ddots & & & \\ & & 0 & 0 & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & & & \ddots & & \ddots & & 0 & x \\ & & & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & 0 & x & x & x & & \\ & & & & & & 0 & 0 & x & x & & \end{array}\right].$$

Das einzig übriggebliebene Element, welches die beiden ersten Spalten noch von der Hamiltonischen J-Hessenberg-Struktur trennt, ist das an Position $(5,2)$. Hier hilft eine Matrix des Typs $R_p(2,c,s)$, bei der es sich wieder um eine Givensrotation, allerdings vom Typ II, handelt. Diese auf $H_p^{(4)}$ angewendet, liefert

$$H_p^{(5)} = R_p(2,c,s)H_p^{(4)}R_p^T(2,c,s) = \left[\begin{array}{cc|cc|cc|cc|cc|cc} x & x & 0 & x & & & & & & & & \\ x & x & 0 & 0 & & & & & & & & \\ \hline 0 & x & x & x & \otimes & x & 0 & \otimes & & & & \\ 0 & 0 & x & x & \otimes & \otimes & 0 & 0 & & & & \\ \hline & & \otimes & x & x & x & 0 & x & & & & \\ & & \otimes & \otimes & x & x & 0 & 0 & & & & \\ \hline & & 0 & \otimes & 0 & x & \ddots & & \ddots & & & \\ & & 0 & 0 & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & & & \ddots & & \ddots & & 0 & x \\ & & & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & 0 & x & x & x & & \\ & & & & & & 0 & 0 & x & x & & \end{array}\right].$$

Ebenso wie im Einzelshiftfall ist es auch hier gelungen, mit Hilfe der Transformationen den Buckel zwei Spalten und zwei Zeilen nach rechts unten in der Matrix zu bewegen. Man kann durch mehrmaliges Anwenden der Transformationen die Störung der Hamiltonischen Struktur aus der Matrix herausschieben. Eine formale Beschreibung der allgemeinen Verfahrensweise sei deshalb in Algorithmus 2.2 dargestellt.

2.3.3 Quadrupelshift

Als letzter Shift soll hier der Quadrupelshift,$(H_p-\mu_1 I)(H_p+\mu_1 I)(H_p-\mu_2 I)(H_p+\mu_2 I)$ genauer betrachtet und eine implizite Verfahrensweise entwickelt werden. Die folgenden Ergebnisse wurden unabhängig zu den teilweise analogen Resultaten in [35] erzielt. Hier

Algorithmus 2.2 Bulge Chasing Algorithmus für den doppelten Shift

INPUT : $H_p \in \mathbb{R}^{2k\times 2k}$ und $S_p \in \mathbb{R}^{2n\times 2k}$
OUTPUT : $H_p \in \mathbb{R}^{2k\times 2k}$ und $S_p \in \mathbb{R}^{2n\times 2}$ nach dem Shift

Wähle Shift μ.
Berechne $R_p(1,c,s)$.
$H_p = R_p(1,c,s)H_pR_p^T(1,c,s)$
$S_p = S_PR_p^T(1,c,s)$
for $i = 3,5,\cdots,2k-1$ **do**
  Berechne $J_p(\frac{i+1}{2}+1,c,s)$.
  $H_p = J_p(\frac{i+1}{2}+1,c,s)H_pJ_p^T(\frac{i+1}{2}+1,c,s)$
  $S_p = S_pJ_p^T(\frac{i+1}{2}+1,c,s)$
  Berechne $G_p(\frac{i+1}{2},c,s)$.
  $H_p = G_p(\frac{i+1}{2},c,s)H_pG_p^{-1}(\frac{i+1}{2},c,s)$
  $S_p = S_pG_p^{-1}(\frac{i+1}{2},c,s)$
  Berechne $J_p(\frac{i+1}{2},c,s)$.
  $H_p = J_p(\frac{i+1}{2},c,s)H_pJ_p^T(\frac{i+1}{2},c,s)$
  $S_p = S_pJ_p^T(\frac{i+1}{2},c,s)$
  Berechne $R_p(\frac{i+1}{2},c,s)$.
  $H_p = R_p(\frac{i+1}{2},c,s)H_pR_p^T(\frac{i+1}{2},c,s)$
  $S_p = S_pR_p^T(\frac{i+1}{2},c,s)$
end for

sind die Shifts die Eigenwerte der Hamiltonischen Matrix

$$H_j = \begin{bmatrix} \delta_j & 0 & \beta_j & \zeta_j \\ 0 & \delta_{j+1} & \zeta_j & \beta_{j+1} \\ \nu_j & 0 & -\delta_j & 0 \\ 0 & \nu_{j+1} & 0 & -\delta_{j+1} \end{bmatrix}.$$

Eine genauere Untersuchung von $(H_p-\mu_1 I)(H_p+\mu_1 I)(H_p-\mu_2 I)(H_p+\mu_2 I)$ ist für die effiziente Berechnung der Shifts notwendig. Dabei ist

$$(H_p-\mu_1 I)(H_p+\mu_1 I)(H_p-\mu_2 I)(H_p+\mu_2 I) = H_p^4 - (\mu_1^2+\mu_2^2)H_p^2 + \mu_1^2\mu_2^2 I,$$

wobei

$$(\mu_1^2+\mu_2^2) = \delta_j^2 + \nu_j\beta_j + \delta_{j+1}^2 + \nu_{j+1}\beta_{j+1},$$
$$\mu_1^2\mu_2^2 = (\delta_j^2+\nu_j\beta_j)(\delta_{j+1}^2+\nu_{j+1}\beta_{j+1}) - \nu_j\nu_{j+1}\zeta_j^2$$

effiziente Berechnungen der Shiftgrößen darstellen. Auch ist es wieder unabdingbar, die erste Spalte

$$x = (H_p-\mu_1 I)(H_p+\mu_1 I)(H_p-\mu_2 I)(H_p+\mu_2 I)e_1 = [x_1,x_2,x_3,0,\cdots,0]^T$$

mit

$$\begin{aligned} x_1 &= (\delta_1^2+\nu_1\beta_1)^2 + \zeta_2^2\nu_1\nu_2 + t(\delta_1^2+\nu_1\beta_1) + d \\ x_2 &= \nu_1\zeta_2\left[(\delta_1^2+\nu_1\beta_1) + (\delta_2^2+\nu_2\beta_2) + t\right] \\ x_3 &= \nu_1\nu_2\zeta_2\zeta_3 \end{aligned}$$

und $t = \delta_j^2+\nu_j\beta_j+\delta_{j+1}^2+\nu_{j+1}\beta_{j+1}$, sowie $d = (\delta_j^2+\nu_j\beta_j)(\delta_{j+1}^2+\nu_{j+1}\beta_{j+1}) - \nu_j\nu_{j+1}\zeta_j^2$, zu berechnen. Es gilt x mittels einer symplektischen Transformation auf ein Vielfaches von e_1 zu reduzieren. Dafür wird eine symplektische Householder Matrix $W_p(1,3,w) =$

$PW(1,3,w)P^T$ verwandt. Die Ähnlichkeitstransformation von $H_p = PHP^T$ mit $W_p(1,3,w)$ ergibt

$$H_p^{(1)} = W_p(1,3,w)H_pW_p^T(1,3,w)$$

$$= \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc}
x & x & \otimes & x & \otimes & \otimes & 0 & \otimes & & & & & & \\
x & x & \otimes & \otimes & \otimes & \otimes & 0 & 0 & & & & & & \\
\hline
\otimes & x & x & x & \otimes & x & 0 & \otimes & & & & & & \\
\otimes & \otimes & x & x & \otimes & \otimes & 0 & 0 & & & & & & \\
\hline
\otimes & \otimes & \otimes & x & x & x & 0 & x & & & & & & \\
\otimes & \otimes & \otimes & \otimes & x & x & 0 & 0 & & & & & & \\
\hline
0 & \otimes & 0 & \otimes & 0 & x & x & x & 0 & x & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\
\hline
 & & & & & & 0 & x & \ddots & & \ddots & & & \\
 & & & & & & 0 & 0 & & \ddots & & \ddots & & \\
\hline
 & & & & & & & & \ddots & & \ddots & & 0 & x \\
 & & & & & & & & & \ddots & & \ddots & 0 & 0 \\
\hline
 & & & & & & & & & & 0 & x & x & x \\
 & & & & & & & & & & 0 & 0 & x & x
\end{array}\right].$$

Wieder werden mit $\otimes$ die zusätzlichen Elemente bezeichnet. Diese müssen im Verlauf des Algorithmus' aus der Matrix heraus geschoben werden. Dazu eliminieren wir zuerst den Eintrag $(6,1)$ mit Hilfe der Transformation $J_p(3,c,s)$ unter der Prämisse, dass die vorhandenen Nullen der ersten beiden Zeilen und Spalten erhalten bleiben. Es entsteht

$$H_p^{(2)} = J_p(3,c,s)H_p^{(1)}J_p^T(3,c,s)$$

$$= \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc}
x & x & \otimes & x & \otimes & \otimes & 0 & \otimes & & & & & & \\
x & x & \otimes & \otimes & 0 & \otimes & 0 & 0 & & & & & & \\
\hline
\otimes & x & x & x & \otimes & x & 0 & \otimes & & & & & & \\
\otimes & \otimes & x & x & \otimes & \otimes & 0 & 0 & & & & & & \\
\hline
\otimes & \otimes & \otimes & x & x & x & 0 & x & & & & & & \\
0 & \otimes & \otimes & \otimes & x & x & 0 & \otimes & & & & & & \\
\hline
0 & \otimes & 0 & \otimes & \otimes & x & x & x & 0 & x & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\
\hline
 & & & & & & 0 & x & \ddots & & \ddots & & & \\
 & & & & & & 0 & 0 & & \ddots & & \ddots & & \\
\hline
 & & & & & & & & \ddots & & \ddots & & 0 & x \\
 & & & & & & & & & \ddots & & \ddots & 0 & 0 \\
\hline
 & & & & & & & & & & 0 & x & x & x \\
 & & & & & & & & & & 0 & 0 & x & x
\end{array}\right].$$

Dank der Hamiltonischen Struktur der Matrix $H_p^{(1)}$ wird mit Ausführen der Transformation auch das Element $(2,5)$ eliminiert. Im Folgenden wird mit der Matrix $J_p(2,c,s)$

multipliziert, so dass das Element $(4,1)$ verschwindet. Das Ergebnis ist

$$H_p^{(3)} = J_p(2,c,s)H_p^{(2)}J_p^T(2,c,s)$$

$$= \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc} x & x & \otimes & x & \otimes & \otimes & 0 & \otimes & & & & & & \\ x & x & 0 & \otimes & \otimes & 0 & 0 & 0 & & & & & & \\ \hline \otimes & x & x & x & \otimes & x & 0 & \otimes & & & & & & \\ 0 & \otimes & x & x & \otimes & \otimes & 0 & \otimes & & & & & & \\ \hline \otimes & \otimes & \otimes & x & x & x & 0 & x & & & & & & \\ 0 & \otimes & \otimes & \otimes & x & x & 0 & \otimes & & & & & & \\ \hline 0 & \otimes & \otimes & \otimes & \otimes & x & x & x & 0 & x & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\ \hline & & & & & & 0 & x & \ddots & & \ddots & & & \\ & & & & & & & & & \ddots & & \ddots & & \\ & & & & & & 0 & 0 & & & & & & \\ \hline & & & & & & & & \ddots & & \ddots & & 0 & x \\ & & & & & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & & & & & 0 & x & x & x \\ & & & & & & & & & & 0 & 0 & x & x \end{array}\right].$$

Eine Givenstransformation $R_p(2,c,s)$ vom Typ II erhält die bereits vorhandenen Nullen der ersten Spalte und transformiert die Matrix so, dass auch das Element $(5,1)$ gleich Null ist und die Matrix folgende Gestalt hat:

$$H_p^{(4)} = R_p(2,c,s)H_p^{(3)}R_p^T(2,c,s)$$

$$= \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc} x & x & \otimes & x & \otimes & \otimes & 0 & \otimes & & & & & & \\ x & x & 0 & \otimes & 0 & 0 & 0 & 0 & & & & & & \\ \hline \otimes & x & x & x & \otimes & x & 0 & \otimes & & & & & & \\ 0 & \otimes & x & x & \otimes & \otimes & 0 & \otimes & & & & & & \\ \hline 0 & \otimes & \otimes & x & x & x & 0 & x & & & & & & \\ 0 & \otimes & \otimes & \otimes & x & x & 0 & \otimes & & & & & & \\ \hline 0 & \otimes & \otimes & \otimes & \otimes & x & x & x & 0 & x & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\ \hline & & & & & & 0 & x & \ddots & & \ddots & & & \\ & & & & & & & & & \ddots & & \ddots & & \\ & & & & & & 0 & 0 & & & & & & \\ \hline & & & & & & & & \ddots & & \ddots & & 0 & x \\ & & & & & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & & & & & 0 & x & x & x \\ & & & & & & & & & & 0 & 0 & x & x \end{array}\right].$$

Nun gilt es, das Matrixelement $(3,1)$ zu eliminieren. Das funktioniert nur unter der Voraussetzung, dass das Element $(2,1)$ von Null verschieden ist, da eine symplektische

Gaußtransformation verwendet werden muss. Die so erhaltene Matrix

$$H_p^{(5)} = G_p(2,y)H_p^{(4)}G_p^T(2,y)$$

$$= \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc}
x & x & \otimes & x & \otimes & \otimes & 0 & \otimes & & & & & & \\
x & x & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & \\
\hline
0 & x & x & x & \otimes & x & 0 & \otimes & & & & & & \\
0 & \otimes & x & x & \otimes & \otimes & 0 & \otimes & & & & & & \\
\hline
0 & \otimes & \otimes & x & x & x & 0 & x & & & & & & \\
0 & \otimes & \otimes & \otimes & x & x & 0 & \otimes & & & & & & \\
\hline
0 & \otimes & \otimes & \otimes & \otimes & x & x & x & 0 & x & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\
\hline
& & & & & & 0 & x & \ddots & & \ddots & & & \\
& & & & & & & & & \ddots & & \ddots & & \\
& & & & & & 0 & 0 & & & & & & \\
\hline
& & & & & & & & \ddots & & \ddots & & 0 & x \\
& & & & & & & & & \ddots & & \ddots & 0 & 0 \\
\hline
& & & & & & & & & & 0 & x & x & x \\
& & & & & & & & & & 0 & 0 & x & x
\end{array}\right]$$

weist eine erste Spalte auf, in der alle zusätzlichen Elemente verschwunden sind. Nun müssen die Störungen der zweiten Spalte eliminiert werden. Dazu benutzt man die permutierte symplektische Givenstransformation $J_p(3,c,s)$. Die Ähnlichkeitstransformation mit dieser liefert

$$H_p^{(6)} = J_p(3,c,s)H_p^{(5)}J_p^T(3,c,s)$$

$$= \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc}
x & x & \otimes & x & 0 & \otimes & 0 & \otimes & & & & & & \\
x & x & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & \\
\hline
0 & x & x & x & \otimes & x & 0 & \otimes & & & & & & \\
0 & \otimes & x & x & \otimes & \otimes & 0 & \otimes & & & & & & \\
\hline
0 & \otimes & \otimes & x & x & x & 0 & x & & & & & & \\
0 & 0 & \otimes & \otimes & x & x & 0 & \otimes & & & & & & \\
\hline
0 & \otimes & \otimes & \otimes & \otimes & x & x & x & 0 & x & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\
\hline
& & & & & & 0 & x & \ddots & & \ddots & & & \\
& & & & & & & & & \ddots & & \ddots & & \\
& & & & & & 0 & 0 & & & & & & \\
\hline
& & & & & & & & \ddots & & \ddots & & 0 & x \\
& & & & & & & & & \ddots & & \ddots & 0 & 0 \\
\hline
& & & & & & & & & & 0 & x & x & x \\
& & & & & & & & & & 0 & 0 & x & x
\end{array}\right].$$

Nachdem $(6,2)$ verschwunden ist, wird auch das Element $(4,2)$ eliminiert. Die Transformation mit $J_p(2,c,s)$ erbringt das gewünschte Resultat:

$$
\begin{aligned}
H_p^{(7)} &= J_p(2,c,s) H_p^{(6)} J_p^T(2,c,s) \\
&= \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc}
x & x & 0 & x & 0 & \otimes & 0 & \otimes & & & & & & \\
x & x & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & \\
\hline
0 & x & x & x & \otimes & x & 0 & \otimes & & & & & & \\
0 & 0 & x & x & \otimes & \otimes & 0 & \otimes & & & & & & \\
\hline
0 & \otimes & \otimes & x & x & x & 0 & x & & & & & & \\
0 & 0 & \otimes & \otimes & x & x & 0 & \otimes & & & & & & \\
\hline
0 & \otimes & \otimes & \otimes & \otimes & x & x & x & 0 & x & & & & \\
0 & 0 & 0 & 0 & 0 & 0 & x & x & 0 & 0 & & & & \\
\hline
 & & & & & & 0 & x & \ddots & & \ddots & & & \\
 & & & & & & 0 & 0 & & \ddots & & \ddots & & \\
\hline
 & & & & & & & & \ddots & & \ddots & & 0 & x \\
 & & & & & & & & & \ddots & & \ddots & 0 & 0 \\
\hline
 & & & & & & & & & & 0 & x & x & x \\
 & & & & & & & & & & 0 & 0 & x & x
\end{array}\right].
\end{aligned}
$$

Der Eintrag $(7,2)$ der Matrix $H_p^{(7)}$ wird durch die Transformation mit $R_p(3,c,s)$ ausgelöscht, so dass das folgende Ergebnis erzeugt wird:

$$
\begin{aligned}
H_p^{(8)} &= R_p(3,c,s) H_p^{(7)} R_p^T(3,c,s) \\
&= \left[\begin{array}{cc|cc|cc|cc|cc|cc|cc}
x & x & 0 & x & 0 & \otimes & 0 & 0 & & & & & & \\
x & x & 0 & 0 & 0 & 0 & 0 & 0 & & & & & & \\
\hline
0 & x & x & x & \otimes & x & \otimes & \otimes & & & & & & \\
0 & 0 & x & x & \otimes & \otimes & \otimes & \otimes & & & & & & \\
\hline
0 & \otimes & \otimes & x & x & x & \otimes & x & 0 & \otimes & & & & \\
0 & 0 & \otimes & \otimes & x & x & \otimes & \otimes & & & & & & \\
\hline
0 & 0 & \otimes & \otimes & \otimes & x & x & x & 0 & x & & & & \\
0 & 0 & \otimes & \otimes & \otimes & \otimes & x & x & 0 & 0 & & & & \\
\hline
 & & & & 0 & \otimes & 0 & x & \ddots & & \ddots & & & \\
 & & & & 0 & 0 & 0 & 0 & & \ddots & & \ddots & & \\
\hline
 & & & & & & & & \ddots & & \ddots & & 0 & x \\
 & & & & & & & & & \ddots & & \ddots & 0 & 0 \\
\hline
 & & & & & & & & & & 0 & x & x & x \\
 & & & & & & & & & & 0 & 0 & x & x
\end{array}\right].
\end{aligned}
$$

Um den Buckel genau zwei Zeilen und zwei Spalten nach rechts unten bewegt zu haben, muss noch das Element $(5,2)$ entfernt werden. Dabei hilft die Transformation mit

$R_p(2,c,s)$ und es resultiert

$$H_p^{(9)} = R_p(2,c,s) H_p^{(8)} R_p^T(2,c,s)$$

$$= \left[\begin{array}{cc|cc|cc|cc|cc|cc} x & x & 0 & x & & & & & & & & \\ x & x & 0 & 0 & & & & & & & & \\ \hline 0 & x & x & x & \otimes & x & \otimes & \otimes & 0 & \otimes & & \\ 0 & 0 & x & x & \otimes & \otimes & \otimes & \otimes & 0 & 0 & & \\ \hline & & \otimes & x & x & x & \otimes & x & 0 & \otimes & & \\ & & \otimes & \otimes & x & x & \otimes & \otimes & 0 & 0 & & \\ \hline & & \otimes & \otimes & \otimes & x & x & x & 0 & x & & \\ & & \otimes & \otimes & \otimes & \otimes & x & x & 0 & 0 & & \\ \hline & & 0 & \otimes & 0 & \otimes & 0 & x & \ddots & & \ddots & & & \\ & & 0 & 0 & 0 & 0 & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & & & & & \ddots & & \ddots & & 0 & x \\ & & & & & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & & & & & 0 & x & x & x \\ & & & & & & & & & & 0 & 0 & x & x \end{array}\right].$$

Eine allgemeine Vorschrift, die das Hinausschieben des Buckels in einem impliziten Quadrupelshift realisiert, wird im Algorithmus 2.3 beschrieben.

2.4 Bestimmung der Eigenwerte

Sei H eine Hamiltonische J-Hessenberg-Matrix. Das Durchführen von SR Schritten, das heißt Benutzen der verschiedenen Shifts, transformiert die ursprüngliche Matrix H auf eine neue Hamiltonische J-Hessenberg-Matrix

$$\tilde{H} = \left[\begin{array}{cccccccc} \tilde{H}_{1,1} & & & & \tilde{H}_{1,n+1} & & & \\ & \tilde{H}_{2,2} & & & & \tilde{H}_{2,n+2} & & \\ & & \ddots & & & & \ddots & \\ & & & \tilde{H}_{n,n} & & & & \tilde{H}_{n,2n} \\ 0 & & & & \tilde{H}_{n+1,n+1} & & & \\ & \ddots & & & & \tilde{H}_{n+2,n+2} & & \\ & & \ddots & & & & \ddots & \\ & & & 0 & & & & \tilde{H}_{2n,2n} \end{array}\right], \qquad (2.1)$$

die eine Blockstruktur besitzt. Dabei sind die Blöcke $\tilde{H}_{jj}$ und ihre Entsprechungen rechts oben und rechts unten von der Größe (1×1) oder (2×2). Die permutierte Form $\tilde{H}_p$ dieser Matrix zeigt, dass im Vergleich zur Ausgangsmatrix H Deflation stattgefunden hat. Das Durchführen von SR Schritten hat bewirkt, dass viele der ursprünglichen Nebendiagonalelemente der oberen Hessenbergmatrix H_p vernachlässigbar klein wurden. Aus diesem Grund kann nun das ursprüngliche Eigenwertproblem der Matrix H in kleinere Eigenwertprobleme der Größe 2×2,

$$H_j^2 = \left[\begin{array}{cc} \delta_j & \beta_j \\ \nu_j & -\delta_j \end{array}\right]$$

Algorithmus 2.3 Bulge Chasing Algorithmus für den vierfachen Shift

INPUT : $H_p \in \mathbb{R}^{2k\times 2k}$ und $S_p \in \mathbb{R}^{2n\times 2k}$
OUTPUT : $H_p \in \mathbb{R}^{2k\times 2k}$ und $S_p \in \mathbb{R}^{2n\times 2}$ nach dem Shift

Wähle Shifts μ_1 und μ_2.
Berechne $H_p(1,3,w)$.
$H_p = H_p(1,3,w)H_pH_p^T(1,3,w)$
$S_p = S_PH_p^T(1,3,w)$
for $i = 3,5,\cdots,2k-1$ **do**
  Berechne $J_p(\frac{i+1}{2}+1,c,s)$.
  $H_p = J_p(\frac{i+1}{2}+1,c,s)H_pJ_p^T(\frac{i+1}{2}+1,c,s)$
  $S_p = S_pJ_p^T(\frac{i+1}{2}+1,c,s)$
  Berechne $J_p(\frac{i+1}{2},c,s)$
  $H_p = J_p(\frac{i+1}{2},c,s)H_pJ_p^T(\frac{i+1}{2},c,s)$
  $S_p = S_pJ_p^T(\frac{i+1}{2},c,s)$
  Berechne $R_p(\frac{i+1}{2},c,s)$.
  $H_p = R_p(\frac{i+1}{2},c,s)H_pR_p^T(\frac{i+1}{2},c,s)$
  $S_p = S_pR_p^T(\frac{i+1}{2},c,s)$
  Berechne $G_p(\frac{i+1}{2},y)$.
  $H_p = G_p(\frac{i+1}{2},y)H_pG_p^T(\frac{i+1}{2},y)$
  $S_p = S_pG_p^{-1}(\frac{i+1}{2},y)$
  Berechne $J_p(\frac{i+1}{2}+1,c,s)$.
  $H_p = J_p(\frac{i+1}{2}+1,c,s)H_pJ_p^T(\frac{i+1}{2}+1,c,s)$
  $S_p = S_pJ_p^T(\frac{i+1}{2}+1,c,s)$
  Berechne $J_p(\frac{i+1}{2},c,s)$.
  $H_p = J_p(\frac{i+1}{2},c,s)H_pJ_p^T(\frac{i+1}{2},c,s)$
  $S_p = S_pJ_p^T(\frac{i+1}{2},c,s)$
  Berechne $R_p(\frac{i+1}{2}+1,c,s)$
  $H_p = R_p(\frac{i+1}{2}+1,c,s)H_pR_p^T(\frac{i+1}{2}+1,c,s)$
  $S_p = S_pR_p^T(\frac{i+1}{2}+1,c,s)$
  Berechne $R_p(\frac{i+1}{2},c,s)$
  $H_p = R_p(\frac{i+1}{2},c,s)H_pR_p^T(\frac{i+1}{2},c,s)$
  $S_p = S_pR_p^T(\frac{i+1}{2},c,s)$
end for

oder 4×4,

$$H_j^4 = \begin{bmatrix} \delta_j & 0 & \beta_j & \zeta_j \\ 0 & \delta_{j+1} & \zeta_j & \beta_{j+1} \\ \nu_j & 0 & -\delta_j & 0 \\ 0 & \nu_{j+1} & 0 & -\delta_{j+1} \end{bmatrix}.$$

zerlegt werden. Das Lösen dieser Eigenwertprobleme fällt weit weniger schwer und so gelingt es die Eigenwerte von H_j^2 in der folgenden Weise aufzuschreiben:

$$\lambda_{1/2} = \pm\sqrt{\delta_j^2 + \nu_j\beta_j}.$$

Für H_j^4 stellen sich nach einigem Rechnen die folgenden Eigenwerte heraus:

$$\lambda_{1/2} = \pm\sqrt{\frac{\delta_j^2 + \nu_j\beta_j + \delta_{j+1}^2 + \nu_{j+1}\beta_{j+1}}{2} + \sqrt{\frac{(\delta_j^2 + \nu_j\beta_j - \delta_{j+1}^2 - \nu_{j+1}\beta_{j+1})^2}{4} + \nu_j\nu_{j+1}\zeta_j^2}}$$

und

$$\lambda_{3/4} = \pm\sqrt{\frac{\delta_j^2 + \nu_j\beta_j + \delta_{j+1}^2 + \nu_{j+1}\beta_{j+1}}{2} - \sqrt{\frac{(\delta_j^2 + \nu_j\beta_j - \delta_{j+1}^2 - \nu_{j+1}\beta_{j+1})^2}{4} + \nu_j\nu_{j+1}\zeta_j^2}}$$

Um Auslöschungseffekte zu vermeiden sollte bei der expliziten numerischen Berechnung der Eigenwerte der Satz von Vieta benutzt werden.

2.5 Bestimmung der Eigenvektoren

Studiert man in [8] den Teil über das Erhalten der Eigenvektoren, dann erkennt man, dass die Matrix $\tilde{H}$ symplektischen Ähnlichkeitstransformationen unterworfen wird, bis bei reellen Eigenwerten auf der Diagonalen von $\tilde{H}_{11}$ der Eigenwert $-\lambda$ steht und im komplexen Fall eine 2×2 Matrix Δ, die die komplexen Eigenwerte $-\lambda$ und $-\mu$ besitzt. Hier symbolisiert das Minuszeichen immer einen Eigenwert mit negativem Realteil. Im reellen ist es mit dem Akkumulieren der Transformationen gelungen, den Eigenvektor zu $-\lambda$ zu erhalten. Im komplexen Fall sind weitere Transformationen notwendig bis die Eigenvektoren zu $-\lambda$ und $-\mu$ explizit vorliegen. Die Matrix

$$\Delta = \begin{bmatrix} \delta_{11} & \delta_{12} \\ \delta_{21} & \delta_{22} \end{bmatrix}$$

kann mit einer Givensrotation auf folgende Gestalt transformiert werden,

$$\begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{bmatrix} \delta_{11} & \delta_{12} \\ \delta_{21} & \delta_{22} \end{bmatrix} \begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} -\alpha & c \\ b & -\alpha \end{bmatrix},$$

wobei gilt, dass $-\lambda = -\alpha + i\beta$ und $-\mu = -\alpha - i\beta$, sowie $i\beta = \sqrt{bc}$. In Abhängigkeit von den unterschiedlichen Vorzeichen, die b und c besitzen, kann die Transformation mit einer Diagonalmatrix $D = \mathrm{diag}(d_{11}, d_{22})$ eine endgültige Gestalt:

$$\begin{bmatrix} 1/d_{11} & 0 \\ 0 & 1/d_{22} \end{bmatrix} \begin{bmatrix} -\alpha & b \\ c & -\alpha \end{bmatrix} \begin{bmatrix} d_{11} & 0 \\ 0 & d_{22} \end{bmatrix} = \begin{bmatrix} -\alpha & \beta \\ -\beta & -\alpha \end{bmatrix} \tag{2.2}$$

oder

$$\begin{bmatrix} 1/d_{11} & 0 \\ 0 & 1/d_{22} \end{bmatrix} \begin{bmatrix} -\alpha & b \\ c & -\alpha \end{bmatrix} \begin{bmatrix} d_{11} & 0 \\ 0 & d_{22} \end{bmatrix} = \begin{bmatrix} -\alpha & -\beta \\ \beta & -\alpha \end{bmatrix}, \tag{2.3}$$

liefern. Nehmen wir an, dass im ersten Fall $b < 0$ und $c > 0$, dann ergibt sich aus den Beziehungen (2.2) und (2.3), $d_{11} = \sqrt{\frac{-\beta}{b}}$ oder $d_{11} = \sqrt{\frac{\beta}{b}}$. Da auch hier die Rechnung reell bleiben soll, ist $d_{11} = \sqrt{\frac{-\beta}{b}}$ zu wählen. Auch im zweiten Fall, mit $b > 0$ und $c < 0$, erhält man die bereits bekannten Formeln für d_{11}. Um den Übergang ins Komplexe zu vermeiden, ist als Lösung für d_{11} nur $d_{11} = \sqrt{\frac{\beta}{b}}$ zulässig. Um auch d_{22} zu bestimmen, ist als eine weitere Bedingung, die Symplektizität von D zu benutzen, welche die Formel $d_{11}d_{22} = 1$ liefert. Da nun für die komplexen Eigenwerte gilt, dass die Blöcke

$$\begin{bmatrix} -\alpha & \beta \\ -\beta & -\alpha \end{bmatrix}$$

oder

$$\begin{bmatrix} -\alpha & -\beta \\ \beta & -\alpha \end{bmatrix}$$

auf der Hauptdiagonalen im oberen linken Block der Matrix $\tilde{H} = S^{-1}HS$ zu finden sind, wobei S die Matrix aller akkumulierten Transformationen ist, können die Eigenvektoren zu den Eigenwerten mit negativem Realteil direkt aus S abgelesen werden. Dies ist leicht anhand der folgenden Beziehung einzusehen,

$$H[x \;\; y] = [x \;\; y] \begin{bmatrix} -\alpha & -\beta \\ \beta & -\alpha \end{bmatrix}.$$

Leider ist es nicht immer ausreichend, nur die Eigenvektoren zu den Eigenwerten mit negativem Realteil zu kennen. Um auch die restlichen Eigenvektoren zu bestimmen, bieten sich mehrere Verfahren an, die im Folgenden beschrieben werden sollen.

Inverse Iteration

Da die Eigenwerte der Matrix H bereits bekannt sind, bietet die Inverse Iteration, dargestellt in Algorithmus 2.4, die Möglichkeit auf schnellem Wege die gewünschten Eigenvektoren zu bestimmen. Der Hauptaufwand dieses Verfahrens liegt natürlich in der Berechnung von $(H-\mu I)y^{(k)} = y^{(k-1)}$. Die Matrix H ist bekanntlich von schwachbesetzter Natur und kann daher zur Vereinfachung von bereits vorhandenen Algorithmen beitragen. Dazu sei nun die Lösung von $(H - \mu I)y^{(k)} = y^{(k-1)}$ über die LU Zerlegung vorgestellt(vgl.[34], S.196 ff).

Definition 2.5.1. *Eine Zerlegung von $H \in \mathbb{R}^{2n,2n}$ in $H = LU$ heißt LU-Zerlegung, dabei sind L und U untere beziehungsweise obere Dreiecksmatrizen der Größe $2n \times 2n$.*

Algorithmus 2.4 Algorithmus Inverse Iteration

INPUT : $H \in \mathbb{R}^{2n,2n}$ und Eigenwert μ
OUTPUT : Eigenvektor y

Bestimme $y^{(0)}$.
$y^{(0)} = \frac{1}{\|y^{(0)}\|}y^{(0)}$
$k = 1$;
while Konvergenzkriterium nicht erfüllt **do**
  Löse $(H - \mu I)y^{(k)} = y^{(k-1)}$.
  $y^{(k)} = \frac{1}{\|y^{(k)}\|}y^{(k)}$
  $k = k + 1$;
end while

Das Bestimmen einer solchen Zerlegung kann recht teuer sein, wenn man aber bedenkt, dass es sich bei der Matrix H um eine Hamiltonische J-Hessenberg-Matrix handelt, dann ist es möglich die Berechnung von L und U sehr schnell durchzuführen. Ruft man sich die Struktur von

$$\tilde{H} = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} = \begin{bmatrix} \diagdown & \diagdown\!\diagdown\!\diagdown \\ \diagdown & \diagdown \end{bmatrix}$$

ins Gedächtnis, dann muss das Erzeugen der Matrizen L und U mit Hilfe des Gauß'schen Eliminationsverfahrens damit beginnen, den Block H_{21} zu eliminieren. Das Durchführen dieses Prozesses bewirkt, dass im Block H_{22} auf der unteren Nebendiagonalen $n-1$ zusätzlich Elemente entstehen. Diese müssen im nächsten Schritt beseitigt werden, um eine zulässige LU Zerlegung erzeugt zu haben. Die so erhaltenen Matrizen L und U sind dann von folgender Gestalt:

$$L = \begin{bmatrix} 1 & & & \\ \diagdown & 1 & & \\ & \diagdown & 1 & \\ & & \diagdown & 1 \end{bmatrix}$$

und

$$U = \begin{bmatrix} \diagdown & \diagdown\!\diagdown\!\diagdown \\ & \diagdown\!\diagdown \end{bmatrix}.$$

Diese Zerlegung hilft nun die Gleichung $(H - \mu I)y^{(k)} = y^{(k-1)}$ zu lösen, indem zuerst ein Vektor w bestimmt wird, so dass $Lw = y^{(k-1)}$ und anschließend $Uy^{(k)} = w$ das Ergebnis für $y^{(k)}$ liefert. Das Vorwärtslösen ist durch die schwache Besetztheit von L schnell durchführbar. Beim Rückwartslösen von $Uy^{(k)} = w$ ist die geringe Anzahl von Nicht-Null-Elementen ein Garant dafür, dass auch diese Rechnung effizient durchgeführt werden kann. Auf die Möglichkeit der zeilenweisen Pivotisierung sei nun im folgenden kurz eingegangen. Das Durchführen der LU-Zerlegung mit Pivotisierung ist durchaus sinnvoll und wirkt sich keineswegs störend auf die schwache Besetzungsstruktur der Matrizen L und U aus. Beginnt man mit der Eliminierung der Elemente des linken unteren Blockes und es muss im k-ten Schritt die Zeile $n+k$ mit der Zeile k getauscht werden, da das Element ν_k größer ist als das Element δ_k, dann wird mit dem Durchführen dieses Zeilentausches die Tridiagonalstruktur des rechten oberen Blockes unterbrochen. In dieser Zeile ist im nordwestlichen Block nur noch ein Diagonalelement zu finden, so nicht noch ein Zeilentausch beim Eliminieren der Elemente der Nebendiagonalen des rechten unteren Blockes stattfindet. Damit werden nicht nur die Stabilitätsvorteile der Pivotisierung ausgenutzt, sondern es lassen sich, durch das Hinzukommen von Nullen auf der Nebendiagonalen des rechten oberen Blockes, weitere Effizienzsteigerungen beim Rückwärtslösen von $Uy^{(k)} = w$ erreichen. Algorithmus 2.5 stellt eine Umsetzung der LU Zerlegung mit Pivotisierung dar. Mit Hilfe dieses Algorithmus' kann nun eine Variante der Inversen Iteration umgesetzt werden, die die Eigenvektoren zu den Eigenwerten mit positivem Realteil liefert.

Schur-ähnliche Form

Das Bestimmen der Eigenvektoren anhand einer Schur-ähnlichen Form beruht auf den Ideen, die in [16] im Abschnitt 7.6.4 vorgestellt werden. Für den Fall, dass wir den

Algorithmus 2.5 Algorithmus LU Zerlegung

INPUT : $H \in \mathbb{R}^{2n,2n}$
OUTPUT : $L, U \in \mathbb{R}^{2n,2n}$

for $i = 1, \ldots, n$ **do**
  if $|H_{i,i}| < |H_{n+i,i}|$ **then**
    Vertausche Zeile i mit Zeile $n + i$.
  end if
  $L_{n+i,i} = -\frac{1}{H_{i,i}} H_{n+i,i}$
  $U(n + i, :) = L_{n+i,i} U(i, :) + U(n + i, :)$
end for
for $i = 1, \ldots, n - 1$ **do**
  if $|H_{i+1,n+i}| < |H_{n+i+1,n+i}|$ **then**
    Vertausche Zeile $i + 1$ mit Zeile $n + i + 1$.
  end if
  $L_{n+i+1,n+i} = -\frac{1}{H_{i+1,n+i}} H_{n+i+1,n+i}$
  $U(n + i + 1, :) = L_{n+i+1,n+i} U(i + 1, :) + U(n + i + 1, :)$
end for

Eigenvektor x zu einem reellen Eigenwert λ suchen, nehmen wir an, dass wir aus dem SR Algorithmus eine Matrix $\tilde{H} = S^{-1}HS$ erhalten haben mit

$$\tilde{H} = \begin{bmatrix} \tilde{H}_{11} & u & \tilde{H}_{13} \\ 0 & \lambda & v^T \\ 0 & 0 & \tilde{H}_{33} \end{bmatrix}. \tag{2.4}$$

Diese Matrix ist eine quasi obere Dreiecksmatrix. Nehmen wir weiterhin an, dass $\lambda \notin \sigma(\tilde{H}_{11})$, wobei mit $\sigma(\tilde{H}_{11})$ das Spektrum der Matrix $\tilde{H}_{11}$ bezeichnet wird. Löst man das System $(\tilde{H}_{11} - \lambda I)w = -u$, dann ist

$$x = S \begin{bmatrix} w \\ 1 \\ 0 \end{bmatrix}$$

der Eigenvektor zum Eigenwert λ. Das Vorgehen bei komplexen Eigenwerten gestaltet sich ein wenig schwieriger. Wir haben bereits Transformationen S, die gewährleisten, dass $\tilde{H} = S^{-1}HS$ im Falle komplexer Eigenwerte 2×2 Blöcke der Gestalt

$$-\Delta^T = \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} \tag{2.5}$$

oder

$$-\Delta^T = \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix} \tag{2.6}$$

im rechten unteren Bereich auf der Diagonalen besitzt. Wir nehmen an, dass wir am Eigenvektor x zum komplexen Eigenwert $\lambda = \alpha + i\beta$ interessiert sind. Eine einfache Bestimmung des Eigenvektors ist möglich, wenn $\tilde{H}$ die folgende Struktur besitzt:

$$\tilde{H} = \begin{bmatrix} \tilde{H}_{11} & u^{(1)} & u^{(2)} & \tilde{H}_{14} \\ 0 & \lambda & 0 & v_1^T \\ 0 & 0 & \mu & v_2^T \\ 0 & 0 & 0 & \tilde{H}_{44} \end{bmatrix}, \tag{2.7}$$

wobei $\mu = \alpha - i\beta$. Um diese Struktur zu erhalten, ist es notwendig geeignete Transformationen zu finden. Wie sehen geeignete Transformationen aus? Es müssen zwei Fälle unterschieden werden. Findet man zum Eigenwert λ den Fall (2.5) vor, dann erbringt die Transformation mit

$$Q = \frac{1}{\sqrt{2}} \begin{bmatrix} -i & 1 \\ 1 & -i \end{bmatrix}$$

den gewünschten Erfolg. Im Falle (2.6) hilft das Verwenden von

$$Q = \frac{1}{\sqrt{2}} \begin{bmatrix} i & 1 \\ 1 & i \end{bmatrix}.$$

Somit läßt sich eine für die Bestimmung der Eigenvektoren nützliche Form, der nachfolgenden Gestalt, erreichen:

$$Q^{-1}\Delta Q = \begin{bmatrix} \alpha + i\beta & 0 \\ 0 & \alpha - i\beta \end{bmatrix}.$$

Wird diese Transformation in der Matrix S akkumuliert, dann kann die ursprüngliche Matrix H in

$$\tilde{H} = S^{-1}HS = \begin{bmatrix} \tilde{H}_{11} & u^{(1)} & u^{(2)} & \tilde{H}_{14} \\ 0 & \lambda & 0 & v_1^T \\ 0 & 0 & \mu & v_2^T \\ 0 & 0 & 0 & \tilde{H}_{44} \end{bmatrix}$$

umgewandelt werden. Diese Gestalt der Matrix bedeutet eine enorme Vereinfachung des Prozesses zur Berechnung der Eigenvektoren. Wenn wiederum gilt, dass $\lambda \notin \sigma(\tilde{H}_{11})$ und $\mu \notin \sigma(\tilde{H}_{11})$, dann können die Eigenvektoren

$$x = S \begin{bmatrix} w^{(1)} \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

zum Eigenwert λ und

$$y = S \begin{bmatrix} w^{(2)} \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

zum Eigenwert μ berechnet werden. Für $w^{(1)}$ und $w^{(2)}$ gilt, dass $(H_{11} - \lambda I)w^{(1)} = -u^{(1)}$ und $(H_{11} - \mu I)w^{(2)} = -u^{(2)}$. Der Aufwand des gesamten Verfahrens wird durch die Effizienz der Bestimmung von $w^{(1)}$ und $w^{(2)}$ dominiert. Auch an dieser Stelle ist die Hamiltonische Struktur der ursprünglichen Matrix ein guter Verbündeter, da durch den SR Algorithmus und die Transformationen, die zur Berechnung der Eigenvektoren mit negativem Realteil benötigt werden, die Struktur der Matrix $\tilde{H}$ und somit auch

die von $u^{(1)}$ und $u^{(2)}$ von einfacher Gestalt sind. So ist, wie bereits in 2.1 dargestellt,

$$\tilde{H} = \begin{bmatrix} \tilde{H}_{1,1} & & & & \tilde{H}_{1,n+1} & & & \\ & \tilde{H}_{2,2} & & & & \tilde{H}_{2,n+2} & & \\ & & \ddots & & & & \ddots & \\ & & & \tilde{H}_{n,n} & & & & \tilde{H}_{n,2n} \\ 0 & & & & \tilde{H}_{n+1,n+1} & & & \\ & \ddots & & & & \tilde{H}_{n+2,n+2} & & \\ & & \ddots & & & & \ddots & \\ & & & 0 & & & & \tilde{H}_{2n,2n} \end{bmatrix},$$

wobei die Blöcke $\tilde{H}_{j,j}$ und deren Gegenstücke rechts unten und oben von der Größe 1×1 für reelle Eigenwerte oder 2×2 im komplexen Fall sind. Das heißt im reellen Fall vereinfacht sich die Gleichung $(\tilde{H}_{11} - \lambda I)w = -u$, angenommen λ steht im 1×1 Block $\tilde{H}_{n+j,n+j}$, zu

$$w_j = \frac{\tilde{H}_{j,n+j}}{-2\lambda}$$

und $w_k = 0 \; \forall k \neq j$. Das Lösen von $(H_{11} - \lambda I)w^{(1)} = -u^{(1)}$ und $(H_{11} - \mu I)w^{(2)} = -u^{(2)}$ vereinfacht sich in ähnlich angenehmer Weise und macht somit eine effiziente Berechnung der Eigenvektoren möglich. Man erhält, die Struktur von $\tilde{H}_{j,j}$ berücksichtigend,

$$\begin{bmatrix} -2\alpha & \\ & 2(-\alpha - i\beta) \end{bmatrix} \begin{bmatrix} w_j^{(1)} \\ w_{j+1}^{(1)} \end{bmatrix} = \begin{bmatrix} -u_j^{(1)} \\ -u_{j+1}^{(1)} \end{bmatrix}$$

und $w_k^{(1)} = 0 \; \forall k \notin \{j, j+1\}$. Im gleichen Atemzug läßt sich das Gleichungssystem für den Eigenvektor zu μ aufschreiben. Man erhält

$$\begin{bmatrix} 2(-\alpha - i\beta) & \\ & -2\alpha \end{bmatrix} \begin{bmatrix} w_j^{(2)} \\ w_{j+1}^{(2)} \end{bmatrix} = \begin{bmatrix} -u_j^{(2)} \\ -u_{j+1}^{(2)} \end{bmatrix}$$

und auch hier gilt $w_k^{(2)} = 0 \; \forall k \notin \{j, j+1\}$. Es ist nun gelungen die Eigenvektoren

$$x = S \begin{bmatrix} w^{(1)} \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

und

$$y = S \begin{bmatrix} w^{(2)} \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

zu bestimmen. Es kann also im Weiteren davon ausgegangen werden, dass uns alle Eigenvektoren der Hamiltonischen Matrix H bekannt sind.

Kapitel 3

Der symplektische Lanczos-Algorithmus mit impliziten Restarts

3.1 Der Lanczos-Algorithmus für unsymmetrische Matrizen

Wenn man den klassischen unsymmetrischen Lanczos-Prozess für eine Matrix $A \in \mathbb{R}^{n,n}$ und zwei Startvektoren $p_1, q_1 \in \mathbb{R}^n$ betrachtet, dann ergeben sich, nachzulesen in [21], zwei Matrizen $P_k, Q_k \in \mathbb{R}^{n,k}$. Diese genügen

$$AP_k = P_k T_k + \beta_{k+1} p_{k+1} e_k^T \tag{3.1}$$

$$A^T Q_k = Q_k T_k^T + \gamma_{k+1} q_{k+1} e_k^T, \tag{3.2}$$

wobei T_k eine Tridiagonalmatrix ist. Es gilt eine Biorthogonalitätsbeziehung für die Matrizen $P_k = [p_1, \ldots, p_k]$ und $Q_k = [q_1, \ldots, q_k]$, das heißt $Q_k^T P_k = I_k$. Das Multiplizieren von (3.1) mit Q_k^T liefert erfreulicherweise $Q_k^T A P_k = T_k$. Die Eigenwerte der Matrix T_k können, als Ritzwerte der Ausgangsmatrix A, gute Näherungen für deren Eigenwerte sein. Um die Berechnung dieser Ritzwerte noch effizienter zu machen, kann man die Berechnung mit der Matrix A^T für viele Matrizen umgehen. Das bedeutet außerdem, dass eine der beiden Folgen vernachlässigt werden kann, was wiederum eine Reduzierung des Speicherplatzes und des Rechenaufwands nach sich zieht. Die Vereinfachung soll im Folgenden begründet werden, siehe hierzu [15]. Gegeben sei eine Matrix $S \in \mathbb{C}^{n,n}$, für die gilt $A^T S = SA$. Der Startvektor q_1 der zweiten Folge kann beliebig gewählt werden, unsere Wahl fällt auf $q_1 = \frac{Sp_1}{\|Sp_1\|}$. Außerdem gilt $\|p_k\| = \|q_k\| = 1$ für alle k und für die Lanczos-Vektoren $p_k = \varphi_{k-1}(A)p_1$ und $q_k = \gamma_k \varphi_{k-1}(A^T)q_1$, wobei φ_{k-1} ein Polynom vom Grad $k-1$ ist. Im Weiteren vereinfachen wir φ_{k-1} zu $\varphi_{k-1}(A) = \alpha_k A^k$, um das Verstehen des Nachfolgenden leichter zu gestalten. Es ist also

$$q_k = \gamma_k \varphi_{k-1}(A^T) \frac{Sp_1}{\|Sp_1\|}.$$

Das Aufschlüsseln von φ liefert

$$q_k = \gamma_k \alpha_k A^T A^T \cdots A^T A^T S \frac{p_1}{\|Sp_1\|}$$

und mit der Beziehung $A^T S = SA$ folgt leicht

$$q_k = \gamma_k \alpha_k A^T A^T \cdots A^T SA \frac{p_1}{\|Sp_1\|}.$$

Die Matrix S läßt sich durch die Gleichung nach links bewegen und das Ergebnis ist

$$q_k = \gamma_k \alpha_k SAA \cdots AA \frac{p_1}{\|Sp_1\|}$$

und mit der Rücksubstitution von φ bleibt

$$q_k = \gamma_k S \varphi_{k-1}(A) \frac{p_1}{\|Sp_1\|}.$$

Mit dem bereits bekannten Ausdruck für p_k läßt sich der erhaltene Ausdruck umwandeln in

$$q_k = \gamma_k \frac{Sp_k}{\|Sp_1\|}.$$

Ein Ausnutzen von $\|q_k\| = 1$ bringt, dass auch $\|\gamma_k \frac{Sp_k}{\|Sp_1\|}\| = 1$. In [15] ist Nachzulesen, dass $\gamma_k > 0$, was bedeutet $\frac{\|Sp_1\|}{\gamma_k} = \|Sp_k\|$. Dies verwendend ergibt sich

$$q_k = \frac{Sp_k}{\|Sp_k\|}.$$

Diese nützliche Beziehung erleichtert den ganzen Prozess, da die Berechnungen mit A^T aus dem Algorithmus entfallen. Es kann im Verlaufe des Lanczos-Algorithmus' zu einem Versagen des Verfahrens kommen, da in jedem Schritt orthogonalisiert wird und dabei eventuell die Division mit sehr kleinen Größen durchgeführt werden muss. Das Ausfallen kann dadurch verhindert werden, dass man geeignete Startvektoren p_1 und q_1 findet. Bislang ist aber kein Verfahren bekannt, dass es ermöglicht geeignete Startvektoren effizient zu berechnen. Der im Abschnitt 3.3 vorgestellte Lanczos-Algorithmus mit impliziten Restarts bietet eine Möglichkeit, dem Zusammenbruch des Verfahrens beizukommen.

3.2 Der symplektische Lanczos-Algorithmus

Das Entwickeln eines symplektischen Lanczos-Verfahrens, welches die Hamiltonische Struktur berücksichtigt, wird im Folgenden vorgestellt. Das vorgestellte Verfahren wurde von Benner und Faßbender in [5] entwickelt und ist ebenfalls in [14] zu finden. Es werden dazu wieder die permutierten Versionen der Matrizen benötigt, also S_p,H_p und auch J_p. Dieses spezielle strukturerhaltene Verfahren generiert eine Folge von Matrizen

$$S_p^{2n,2k} = [v_1, w_1, v_2, w_2, \ldots, v_k, w_k] \in \mathbb{R}^{2n,2k},$$

die folgender Lanczos-Rekursion genügen

$$H_p S_p^{2n,2k} = S_p^{2n,2k} \tilde{H}_p^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T. \tag{3.3}$$

Bei der Matrix $\tilde{H}_p^{2k}$ handelt es sich um eine permutierte J-Hessenberg-Matrix der Größe $2k \times 2k$. Ohne auf das mögliche Versagen eines Lanczos-Algorithmus' einzugehen, soll nun solch ein Verfahren vorgestellt werden. Sei $S_p = [v_1, w_1, v_2, w_2, \ldots, v_n, w_n]$. Für einen gegebenen Startvektor v_1 gewinnt man aus (3.3) spaltenweise Folgendes:

$$H_p S_p e_m = S_p \tilde{H}_p e_m, \qquad m = 1, 2, \ldots, 2k-1.$$

Dabei gilt für ungerade Spalten

$$\begin{array}{rcl} H_p v_{m+1} & = & \delta_{m+1} v_{m+1} + \nu_{m+1} w_{m+1} \\ & \Longleftrightarrow & \\ \nu_{m+1} w_{m+1} & = & H_p v_{m+1} - \delta_{m+1} v_{m+1} \\ & =: & \tilde{w}_{m+1} \end{array} \tag{3.4}$$

und für die geraden Spalten zeigt sich

$$\begin{array}{rcl} H_p w_m & = & \zeta_m v_{m-1} + \beta_m v_m - \delta_m w_m + \zeta_{m+1} v_{m+1} \\ & \Longleftrightarrow & \\ \zeta_{m+1} v_{m+1} & = & H_p w_m - \zeta_m v_{m-1} - \beta_m v_m + \delta_m w_m \\ & =: & \tilde{v}_{m+1} \end{array} \tag{3.5}$$

Es bleibt ν_{m+1} und ζ_{m+1} so zu wählen, dass $S_p^T J_p S_p = J_p$ gilt. Dieses bedeutet ν_{m+1} und ζ_{m+1} so zu bestimmen, dass $v_{m+1}^T J_p w_{m+1} = 1$. Eine Möglichkeit, dies zu erreichen ist

$$\zeta_{m+1} = \|\tilde{v}_{m+1}\|_2 \quad und \quad \nu_{m+1} = v_{m+1}^T J_p H_p v_{m+1}.$$

Wenn man (3.5) mit $w_m^T J_p$ multipliziert und die Symplektizität von S_p benutzt, dann erhält man für β_m:

$$\beta_m = -w_m^T J_p H_p w_m.$$

Der Parameter δ_m kann frei gewählt werden, möglich sind zum Beispiel $\delta_m = 1$ oder $\delta_m = v_m^T H_p v_m$. Der eben vorgestellte Prozess ist im Algorithmus 3.1 zusammengefasst. Bestimmend für den Aufwand dieses Verfahrens ist, dass nur eine Matrix-Vektor-Multiplikation pro Lanczos-Vektor durchgeführt werden muss. Eine effiziente Implementierung des Verfahrens ist mit dem Aufwand von $6n + (4nz + 32k)k$ flops möglich, wobei nz die Anzahl der Nichtnull-Elemente der Matrix H_p ist und insgesamt $2k$ Lanczos-Vektoren erzeugt werden. Im vorgestellten symplektischen Lanczos-Prozess kann es passieren, dass durch Null oder sehr kleine Größen geteilt werden muss. Sollte dieser Fall für den Parameter $\zeta_{m+1} = \|\tilde{v}_{m+1}\|$ auftreten, dann ist ein symplektischer invarianter Unterraum oder eine gute Approximation an einen solchen gefunden. Das Auffinden eines Vektors $\tilde{v}_{m+1}$, der die beiden Kriterien:

$$\begin{array}{rcl} v_j^T J_p \tilde{v}_{m+1} & = & 0 \\ w_j^T J_p \tilde{v}_{m+1} & = & 0 \end{array}$$

mit $j = 1, \ldots, m$ erfüllt, garantiert eine Fortführung des Algorithmus'. Die Hamiltonische J-Hessenberg-Matrix ist nicht länger unreduziert, sondern das ursprüngliche Problem ist in zwei kleinere zerfallen. Ein weiteres Versagen tritt ein, wenn der Vektor $\tilde{w}_{m+1}$ dem Nullvektor entspricht oder zumindestens vernachlässigbar klein ist. In diesem Fall folgt aus der Beziehung (3.4), dass der Parameter ν_m ebenfalls Null oder vernachlässigbar klein ist. Das Verschwinden von ν_m entspricht einer Deflation der permutierten J-Hessenberg-Matrix. Ruft man sich die Struktur der J-Hessenberg-Matrix in Erinnerung mit $\nu_m = 0$, dann ist klar das ein Links- und ein Rechtseigenvektor aus den bereits akkumulierten Transformationen direkt abgelesen werden können. Als Zusammenspiel der beiden bereits vorgestellten Möglichkeiten des Zusammenbruchs erweist sich der Fall des gleichzeitigen Verschwindens der Vektoren $\tilde{w}_{m+1}$ und $\tilde{v}_{m+1}$, als ebenfalls günstige Variante. Mit den gemachten Argumenten ist klar, dass ein symplektischer invarianter Unterraum gefunden wurde und zusätzlich noch Eigenvektoren abgelesen werden können. Dem Vorgestellten steht der Fall entgegen, dass weder der Vektor $\tilde{w}_{m+1}$, noch der Vektor $\tilde{v}_{m+1}$ vernachlässigbar klein ist, aber der Parameter $\nu_{m+1} = 0$. Bei Eintreten dieser Konstellation ist keine Reduktion der Hamiltonischen Matrix auf eine Hamiltonische J-Hessenbergform mit einer Transformation, deren erste Spalte v_1 ist, möglich.

Algorithmus 3.1 Algorithmus für ein symplektisches Lanczos-Verfahren

INPUT : $H_p \in \mathbb{R}^{2n,2n}$ und $m \in \mathbb{N}$
OUTPUT : $\tilde{H}_p \in \mathbb{R}^{2m,2m}$, $S_p \in \mathbb{R}^{2n,2m}$, ζ_{m+1} und v_{m+1}

Wähle Startvektor $\tilde{v}_1 \neq 0 \in \mathbb{R}^{2n}$.
$v_0 = 0 \in \mathbb{R}^{2n}$
$\zeta_1 = \|\tilde{v}_1\|_2$
$v_1 = \frac{1}{\zeta_1}\tilde{v}_1$
for $m = 1, 2, \ldots$ **do**
  (Berechnung von δ_m)
  $\delta_m = 1$
  (Berechnung von w_m)
  $\tilde{w}_m = H_p v_m - \delta_m v_m$
  $\nu_m = v_m^T J_p H_p v_m$
  $w_m = \frac{1}{\nu_m}\tilde{w}_m$
  (Berechnung von β_m)
  $\beta_m = -w_m^T J_p H_p w_m$
  (Berechnung von v_{m+1})
  $\tilde{v}_{m+1} = H_p w_m - \zeta_m v_{m-1} - \beta_m v_m + \delta_m w_m$
  $\zeta_{m+1} = \|\tilde{v}_{m+1}\|_2$
  $v_{m+1} = \frac{1}{\zeta_{m+1}}\tilde{v}_{m+1}$
end for

3.3 Der symplektische Lanczos-Algorithmus mit impliziten Restarts

Ein Verfahren vom Typ eines Algorithmus' mit impliziten Restarts berechnet aus einer bereits bestehenden Faktorisierung der Ordnung $2m$ eine neue Faktorisierung der gleichen Ordnung, deren Startvektor aus der Anwendung eines Polynoms auf den ursprünglichen Startvektor resultiert, ohne dabei einen expliziten Restart durchgeführt zu haben. Ausgehend von einer Matrix $S_p^{2n,2k}$ und damit einer Folge von Vektoren v_j und w_j, die die folgende symplektische Lanczos-Faktorisierung charakterisieren:

$$H_p S_p^{2n,2m} = S_p^{2n,2m} \tilde{H}^{2m,2m} + \zeta_{m+1} v_{m+1} e_{2m}^T \tag{3.6}$$

erzeugt ein impliziter Restart eine neue Lanczos-Faktorisierung der Ordnung $2m$

$$H_p \check{S}_p^{2n,2m} = \check{S}_p^{2n,2m} \check{H}^{2m,2m} + \check{\zeta}_{m+1} \check{v}_{m+1} e_{2m}^T, \tag{3.7}$$

wobei der Startvektor $\check{v}_1$ sich in folgender Weise aus dem Startvektor v_1 errechnet, ohne dabei einen expliziten Restart durchgeführt zu haben,

$$\check{v}_1 = \rho(H_p - \mu I) v_1.$$

Sorensen stellte 1992 in [31] die Umsetzung eines Algorithmus' mit impliziten Restarts vor, dass die Anwendung des Polynoms unter Zuhilfenahme der QR Zerlegung realisiert. An dieser Stelle soll ein solcher Algorithmus für das Benutzen von Einzel-, Doppel- und Quadrupelshift eingeführt werden, wobei der Einzel- und Doppelshift bereits in [5] diskutiert wurden.

3.3.1 Einzelshift

Sei S_p eine symplektische Matrix der Größe $2m \times 2m$. Wenn man die Gleichung (3.6) von rechts mit S_p multipliziert, dann ergibt sich

$$H_p(S_p^{2n,2m} S_p) = (S_p^{2n,2m} S_p)(S_p^{-1} \tilde{H}^{2m,2m} S_p) + \zeta_{m+1} v_{m+1} e_{2m}^T S_p. \tag{3.8}$$

Definiert man $\check{S}_p^{2n,2m} = S_p^{2n,2m} S_p$ und $\check{H}_p^{2m,2m} = S_p^{-1} \tilde{H}_p^{2m,2m} S_p$, so erhält man

$$H_p \check{S}_p^{2n,2m} = \check{S}_p^{2n,2m} \check{H}^{2m,2m} + \zeta_{m+1} v_{m+1} e_{2m}^T S_p. \tag{3.9}$$

Wenn wir S_p aus der permutierten SR Zerlegung, $H_p - \mu I = S_p R_p$, entnehmen, dann ist aus dem Beweis des Satzes 2.1.2 bekannt, dass S_p eine obere Hessenbergmatrix ist. Somit kann das Residuum von (3.9) wie folgt explizit aufgeschrieben werden,

$$r_m = \zeta_{m+1} v_{m+1} (s_{2m,2m-1} e_{2m-1}^T + s_{2m,2m} e_{2m}^T).$$

Mit s_{ij} sei der Eintrag (i,j) der Matrix S_p bezeichnet. Um einen Residuenterm wie in (3.6), das heißt Vektor mal e_{2m}^T, zu erhalten, muss die Gleichung (3.9) aufgespalten werden. Aus

$$H_p \check{S}_p^{2n,2m} = [\check{S}_p^{2n,2m-2}, \check{v}_m, \check{w}_m, v_{m+1}] \left[\begin{array}{c|cc} \check{H}_p^{2m-2,2m-2} & 0 & \check{\zeta}_m e_{2m-3} \\ \hline \check{\zeta}_m e_{2m-2}^T & \check{\delta}_m & \check{\beta}_m \\ 0 & \check{\nu}_m & -\check{\delta}_m \\ \hline 0 & \zeta_{m+1} S_{2m,2m-1} & \zeta_{m+1} s_{2m,2m} \end{array}\right]$$

kann eine Lanczos-Faktorisierung der Ordnung $2m-2$ gewonnen werden, dies ist

$$H_p \check{S}_p^{2n,2m-2} = \check{S}_p^{2n,2m-2} \check{H}_p^{2m-2,2m-2} + \check{\zeta}_m \check{v}_m e_{2m-2}^T \tag{3.10}$$

Da der Raum, der durch die Spalten von $\check{S}_p^{2n,2m-2}$ aufgespannt wird, symplektisch ist, ist auch die erhaltene Lanczos-Zerlegung zulässig für den Startvektor $\check{v}_1 = \rho(H_p - \mu I)v_1$. Der symplektische Lanczos-Algorithmus generierte Vektoren v_j mit $\|v_j\| = 1$ und ließ die Freiheit $\delta_j = 1$ zu wählen. Dieses gilt nun nicht mehr für die ungeraden Spalten von S_p und auch nicht für die Parameter $\check{\delta}_j$ und somit auch nicht für die neu erhaltene Lanczos-Faktorisierung. Die Normierung der v_j mit $\|v_j\| = 1$ kann durch die Multiplikation von rechts mit einer Diagonalmatrix D der folgenden Gestalt wiederhergestellt werden:

$$D = \operatorname{diag}\left(\frac{1}{\|v_1\|}, 1, \frac{1}{\|v_2\|}, 1, \ldots, \frac{1}{\|v_k\|}, 1\right).$$

3.3.2 Doppelshift

Nehmen wir wieder an, dass eine Lanczos-Zerlegung der folgenden Art vorliegt:

$$H_p S_p^{2n,2m} = S_p^{2n,2m} \tilde{H}_p^{2m,2m} + \zeta_{m+1} v_{m+1} e_{2m}^T. \tag{3.11}$$

Dabei sei $m = k + q$. $2k$ ist die Anzahl der gewünschten Eigenwerte. Die anderen $2q$ Eigenwerte sind nicht von Interesse, weswegen die aus [31] bekannte Vorgehensweise mit exakten Shifts zur Anwendung kommt, ein Polynom anzusetzen, welches den Anteil in Richtung der nichtgesuchten Eigenvektoren minimiert. Ein geeigneter Ansatz hierfür ist

$$\rho(z) = (z - \mu_{2k+1})(z - \mu_{2k+2}) \ldots, (z - \mu_{2k+2q-1})(z - \mu_{2k+2q}).$$

Das Anwenden dieses Polynoms auf die Matrix wird durch das q-malige Durchführen eines SR Doppelshifts realisiert. Wir verwenden hier den im Abschnitt 2.3.2 vorgestellten impliziten Shift. Das Durchführen eines Doppelshifts $(\tilde{H}_p^{2m} - \mu_i I)(\tilde{H}_p^{2m} + \mu_i I) = S_p^{(i)} R_p^{(i)}$ im reellen Fall und von $(\tilde{H}_p^{2m} - \mu_i I)(\tilde{H}_p^{2m} - \bar{\mu}_i I) = S_p^{(i)} R_p^{(i)}$ im komplexen Fall findet q Mal statt und es entsteht $S_p = S_p^{(1)} S_p^{(2)} \cdots S_p^{(q)}$. Diese symplektische Matrix wird verwandt, um die nachstehende Zerlegung zu erhalten:

$$H_p(S_p^{2n,2m} S_p) = (S_p^{2n,2m} S_p)(S_p^{-1} \tilde{H}_p^{2m,2m} S_p) + \zeta_{m+1} v_{m+1} e_{2m}^T S_p. \tag{3.12}$$

Definiert man $\check{S}_p^{2n,2m} = S_p^{2n,2m} S_p$ und $\check{H}_p^{2m,2m} = S_p^{-1} \tilde{H}_p^{2m,2m} S_p$, so erhält man

$$H_p \check{S}_p^{2n,2m} = \check{S}_p^{2n,2m} \check{H}_p^{2m,2m} + \zeta_{m+1} v_{m+1} e_{2m}^T S_p. \tag{3.13}$$

Die Matrizen $S_p^{(i)}$ sind auf Grund der Zerlegung eines Doppelshifts nicht mehr von Hessenberggestalt. Sie besitzen eine zusätzliche Nebendiagonale. Da in S_p die Matrizen $S_p^{(i)}$ akkumuliert sind, besitzt diese $2p$ Nebendiagonalen. Den Residuenterm

$$r_m = \zeta_{m+1} v_{m+1} (s_{2m,2m-2q} e_{2m-2q}^T + \ldots + s_{2m,2m-1} e_{2m-1}^T + s_{2m,2m} e_{2m}^T)$$

betrachtend, ist es nicht sofort möglich eine neue Lanczos-Faktorisierung aufzuschreiben. Es hilft das Aufspalten von (3.13), in eine Matrix

$$[\check{S}_p^{2n,2m-2}, \check{v}_{k+1}, \check{w}_{k+1}, \ldots, v_{m+1}]$$

und eine Matrix

$$\left[\begin{array}{c|cccc} \check{H}_p^{2m-2q,2m-2q} & 0 & \check{\zeta}_{k+1}e_{2k-1} & & \\ \hline \check{\zeta}_k + 1e_{2k}^T & \check{\delta}_{k+1} & \check{\beta}_{k+1} & & \\ 0 & \check{\nu}_{k+1} & -\check{\delta}_{k+1} & & \\ \vdots & & \ddots & \ddots & \\ & & & \ddots & \ddots \\ 0 & & & & \\ \hline \zeta_{m+1}s_{2m,2m-2q}e_{2m-2q}^T & \zeta_{m+1}s_{2m,k+1} & \zeta_{m+1}s_{2m,k+2} & \cdots & \zeta_{m+1}s_{2m,2m} \end{array}\right],$$

dabei sei bemerkt, dass $2m - 2q = 2k$. Wir erhalten eine Lanczos-Faktorisierung der Ordnung $2k$, die wie folgt lautet,

$$H_p\check{S}_p^{2n,2k} = \check{S}_p^{2n,2k}\check{H}^{2k,2k} + r_k e_{2k}^T. \tag{3.14}$$

mit dem Residuenterm

$$r_k = \check{\zeta}_{k+1}\check{v}_{k+1} + \zeta_{m+1}s_{2m,2k}v_{m+1}.$$

Die Umsetzung eines symplektischen Lanczos-Prozess mit impliziten Restarts wird im Algorithmus 3.2 dargestellt.

Algorithmus 3.2 IRSL Algorithmus mit Doppelshift

INPUT : $H \in \mathbb{R}^{2n,2n}$, $p \in \mathbb{N}$ und $k \in \mathbb{N}$
OUTPUT : $\lambda_1, \lambda_2, \ldots, \lambda_{2k} \in \mathbb{C}$

Berechne $m = k + q$
(Erzeuge Lanczos-Faktorisierung der Ordnung $2m$)
$H_pS_p^{2n,2m} = S_p^{2n,2m}\tilde{H}_p^{2m,2m} + \zeta_{m+1}v_{m+1}e_{2m}^T$
while Abbruchkriterium nicht erfüllt **do**
  Berechne $\sigma(\tilde{H}_p^{2m,2m})$
  Festlegen der Shifts $\mu_1, \ldots, \mu_{2q}$
  $S = I_{2m}$
  for $i = 1, \ldots, q$ **do**
    Führe impliziten Doppelshift mit μ_i und μ_{q+i} aus.
    Akkumulierung der Transformationen in S.
  end for
  Berechne $S_p^{2n,2k} = S_p^{2n,2m}S(:, 1:2k)$.
  Berechne das Residuum $r_k = \check{\zeta}_{k+1}\check{v}_{k+1} + \zeta_{m+1}s_{2m,2k}v_{m+1}$.
  (Erzeuge Lanczos-Faktorisierung der Ordnung $2k$)
  $H_pS_p^{2n,2k} = S_p^{2n,2k}\tilde{H}_p^{2k,2k} + r_k e_{2k}^T$
  (Erzeuge Lanczos-Faktorisierung der Ordnung $2m$ mittels
  $2q$ Schritten symplektischer Lanczos-Prozess)
  $H_pS_p^{2n,2m} = S_p^{2n,2m}\tilde{H}_p^{2m,2m} + \zeta_{m+1}v_{m+1}e_{2m}^T$
  Berechne Abbruchkriterium.
end while

3.3.3 Quadrupelshift

Auch im Fall des vierfachen Shifts bildet eine Lanczos-Faktorisierung die Grundlage,

$$H_pS_p^{2n,2m} = S_p^{2n,2m}\tilde{H}_p^{2m,2m} + \zeta_{m+1}v_{m+1}e_{2m}^T. \tag{3.15}$$

Es sei $m = k + 2q$, wobei wiederum $2k$ Eigenwerte gewünscht und die anderen $4q$ nicht von Interesse sind. Der Vierfachshift

$$(\tilde{H}_p^{2m} - \mu_i I)(\tilde{H}_p^{2m} + \mu_i I)(\tilde{H}_p^{2m} - \mu_{q+i} I)(\tilde{H}_p^{2m} + \mu_{q+i} I) = S_p^{(i)} R_p^{(i)}$$

sei auch hier q Mal ausgeführt und die Transformationen $S_p^{(i)}$ sind in $S_p = S_p^{(1)} S_p^{(2)} \dots S_p^{(q)}$ akkumuliert. Zu beachten ist, dass auch hier die Matrizen $S_p^{(i)}$ keine Hessenberggestalt haben, sondern vielmehr drei zusätzliche Nebendiagonalen besitzen und somit das Residuum von folgender Gestalt ist,

$$r_m = \zeta_{m+1} v_{m+1} (s_{2m,2m-4p} e_{2m-4p}^T + s_{2m,2m-3p} e_{2m-3p}^T + \dots + s_{2m,2m} e_{2m}^T)$$

Ein Aufspalten der Matrizen H_p und $\check{S}_p^{2n,2m}$, analog zum Einzel- und Doppelshift, liefert an dieser Stelle die Möglichkeit eine reduzierte Zerlegung der folgenden Art zu erhalten:

$$H_p \check{S}_p^{2n,2k} = \check{S}_p^{2n,2k} \check{H}^{2k,2k} + r_k e_{2k}^T. \tag{3.16}$$

Im Falle des Vierfachshifts kann das Residuum r_k wie folgt aufgeschrieben werden,

$$r_k = \check{\zeta}_{k+1} \check{v}_{k+1} + \zeta_{m+1} s_{2m,2k} v_{m+1}.$$

Die reduzierte Zerlegung wird nun mit Hilfe des symplektischen Lanczos-Algorithmus' zu einer Zerlegung der Größe $2m$:

$$H_p \check{S}_p^{2n,2m} = \check{S}_p^{2n,2m} \check{H}_p^{2m,2m} + \check{\zeta}_{m+1} \check{v}_{m+1} e_{2m}^T. \tag{3.17}$$

3.4 Abbruchkriterien

In [6] liefern Benner und Faßbender Abbruchkriterien für das Lösen des symplektischen Eigenwertproblems. Diese Aussagen sollen in diesem Abschnitt auf die Bearbeitung des Hamiltonischen Eigenwertproblems übertragen werden. Dazu sei mit dem folgenden Lemma begonnen, welches in der weiteren Diskussion von Abbruchschranken hilfreich sein wird.

Lemma 3.4.1. *Sei H eine unreduzierte Hamiltonische J-Hessenberg-Matrix. Wenn $Hx = \lambda x$ mit $x \neq 0$ ist, dann gilt $e_{2k}^T x \neq 0$.*

Beweis. Der Beweis wird mittels Induktion über die Größe der Matrix H geführt. Sei $x = [x_1, x_2, \cdots, x_{2n}]$. Sei nun $n = 2$ und wir betrachten die Zeilen von $Hx = \lambda x$.

$$\begin{aligned}
\delta_1 x_1 + \beta_1 x_3 + \zeta_2 x_4 &= \lambda x_1 && (3.18)\\
\delta_2 x_2 + \zeta_2 x_3 + \beta_2 x_4 &= \lambda x_2 && (3.19)\\
\nu_1 x_1 - \delta_1 x_3 &= \lambda x_3 && (3.20)\\
\nu_2 x_2 - \delta_2 x_4 &= \lambda x_3 && (3.21)
\end{aligned}$$

Sei nun $x_4 = 0$, dann folgt aus der Beziehung (3.21), dass $x_2 = 0$, weil die Unreduziertheit von H liefert, dass $\nu_2 \neq 0$. Ein Betrachten von (3.19) ergibt, dass $\zeta_2 x_3 = 0$ und auch hier leistet die Unreduziertheit der Matrix H gute Dienste, denn $\zeta_2 \neq 0$. Somit wird auch $x_3 = 0$ und ein Ausnutzen der Beziehung (3.20), unter der Bedingung $x_2, x_3, x_4 = 0$, reduziert x_1 auf Null. Dies stellt aber einen Widerspruch zu der

Algorithmus 3.3 IRSL Algorithmus mit Quadrupelshift

INPUT : $H \in \mathbb{R}^{2n,2n}$, $p \in \mathbb{N}$ und $k \in \mathbb{N}$
OUTPUT : $\lambda_1, \lambda_2, \ldots, \lambda_{2k} \in \mathbb{C}$

Berechne $m = k + 2q$
(Erzeuge Lanczos-Faktorisierung der Ordnung $2m$)
$H_p S_p^{2n,2m} = S_p^{2n,2m} \tilde{H}_p^{2m,2m} + \zeta_{m+1} v_{m+1} e_{2m}^T$
while Abbruchkriterium nicht erfüllt **do**
  Berechne $\sigma(\tilde{H}_p^{2m,2m})$.
  Festlegen der Shifts $\mu_1, \ldots, \mu_{4q}$.
  $S = I_{2m}$
  for $i = 1, \ldots, q$ **do**
    Führe impliziten Quadrupelshift mit μ_i und μ_{q+i} aus.
    Akkumulierung der Transformationen in S.
  end for
  Berechne $S_p^{2n,2k} = S_p^{2n,2m} S(:, 1:2k)$.
  Berechne das Residuum $r_k = \check{\zeta}_{k+1} \check{v}_{k+1} + \zeta_{m+1} s_{2m,2k} v_{m+1}$.
  (Erzeuge Lanczos-Faktorisierung der Ordnung $2k$)
  $H_p S_p^{2n,2k} = S_p^{2n,2k} \tilde{H}_p^{2k,2k} + r_k e_{2k}^T$
  (Erzeuge Lanczos-Faktorisierung der Ordnung $2m$ mittels $4q$ Schritten symplektischer Lanczos-Prozess)
  $H_p S_p^{2n,2m} = S_p^{2n,2m} \tilde{H}_p^{2m,2m} + \zeta_{m+1} v_{m+1} e_{2m}^T$
  Berechne Abbruchkriterium.
end while

Annahme $x \neq 0$ dar. Es sei nun angenommen, dass die Aussagen des Lemmas für $2(n-1)$ richtig sind. Der Einfachheit halber werden die Matrizen im bereits bekannten permutierten Fall betrachtet. Es ergibt sich folgende Beziehung:

$$H_p^{2n,2n} x_p = \lambda x_p. \tag{3.22}$$

Ein Aufteilen von H_p und x_p erbringt:

$$H_p = \left[\begin{array}{c|cc} H_p^{2(n-1),2(n-1)} & 0 & \zeta_n e_{2n-3} \\ \hline \zeta_n e_{2(n-1)}^T & \delta_n & \beta_n \\ 0 & \nu_n & -\delta_n \end{array}\right]$$

und

$$x_p = \begin{bmatrix} y \\ \tilde{x}_{2n-1} \\ \tilde{x}_{2n} \end{bmatrix}.$$

Sei nun $x_{2n} = \tilde{x}_{2n} = 0$. Das Betrachten der letzten Zeile von (3.22) liefert $\nu_n \tilde{x}_{2n-1} = 0$. Beachtet man, dass $\nu_n \neq 0$, da H eine unreduzierte Matrix ist, ergibt sich $\tilde{x}_{2n-1} = 0$. Dieses Resultat verwendend erhält man mit Hilfe der vorletzten Zeile von (3.22) die Beziehung $\zeta_n y_{2n-2} + \delta_n \tilde{x}_{2n-1} + \beta_n \tilde{x}_{2n} = \lambda \tilde{x}_{2n-1}$. Daraus folgt, wieder berücksichtigend das H unreduziert ist, dass $y_{2n-2} = 0$, dies stellt aber einen Widerspruch zur Voraussetzung $y_{2n-2} = y e_{2(n-1)}^T \neq 0$ dar.

□

Nach dem Durchführen eines Shifts im Impliziten symplektischen Lanczos-Verfahren erhält man, ausgehend von einer Faktorisierung der Ordnung $2k$, eine Beziehung der

folgenden Gestalt:

$$H_p S_p^{2n,2k} = S_p^{2n,2k} \tilde{H}_p^{2k,2k} + r_k e_{2k}^T.$$

Hierbei soll von der Betrachtung der permutierten Matrizen abgewichen werden, dabei entsteht

$$HS^{2n,2k} = S^{2n,2k} \tilde{H}^{2k,2k} + \tilde{r}_k e_{2k}^T. \tag{3.23}$$

Wenn λ ein Eigenwert von $\tilde{H}^{2k,2k}$ ist mit einem Eigenvektor y, dann gilt für $x = S^{2n,2k}y$

$$\begin{aligned} \|Hx - \lambda x\| &= \|HS^{2n,2k}y - \lambda S^{2n,2k}y\| \\ &= \|(HS^{2n,2k} - S^{2n,2k}\tilde{H}^{2k,2k})y\| \\ &= \|(\tilde{r}_k e_{2k}^T)y\| \\ &= \|(\check{\zeta}_k \tilde{\check{v}}_k + \zeta_{k+1} s_{2m,2k} \tilde{v}_{k+1}) e_{2k}^T y\| \\ &= |e_{2k}^T y| \|(\check{\zeta}_k \tilde{\check{v}}_k + \zeta_{k+1} s_{2m,2k} \tilde{v}_{k+1})\| \end{aligned} \quad . \tag{3.24}$$

Nach Lemma 3.4.1 ist $|e_{2k}^T y| \neq 0$, wenn $\tilde{H}^{2k,2k}$ unreduziert ist. Das Paar (λ, x) ist ein Ritzpaar der Matrix H. Wenn die rechte Seite von (3.24) hinreichend klein wird, dann ist (λ, x) eine gute Näherung für ein Eigenpaar der Matrix H. Für die leicht gestörte Matrix $H-E$, kann man zeigen, dass das Paar (λ, x) exakt ist, das heißt $(H-E)x = \lambda x$. E kann mit den folgenden Betrachtungen explizit angegeben werden:

$$\begin{aligned} (H-E)x &= (H-E)S^{2n,2k}y \\ &= HS^{2n,2k}y - ES^{2n,2k}y \\ &= S^{2n,2k}\tilde{H}^{2k,2k}y + \tilde{r}_k e_{2k}^T y - ES^{2n,2k}y \\ &= \lambda x + \tilde{r}_k e_{2k}^T y - ES^{2n,2k}y \end{aligned}$$

Mit $E = \tilde{r}_k e_k^T \left(S^{2n,2k}\right)^T J^n$ erzielt man den gewünschten Erfolg, das heißt, dass $\tilde{r}_k e_{2k}^T y - ES^{2n,2k}y$ verschwindet:

$$\begin{aligned} ES^{2n,2k}y &= \tilde{r}_k e_k^T \left(S^{2n,2k}\right)^T J^n S^{2n,2k} y \\ &= \tilde{r}_k e_k^T J^k y \\ &= \tilde{r}_k e_{2k}^T y \end{aligned} \quad .$$

Es ist also nicht notwendig $(HS^{2n,2k} - \lambda S^{2n,2k})y$ explizit zu berechnen, um eine Abschätzung für die Norm des Residuums zu erhalten, sondern man kann

$$\|Hx - \lambda x\| = |e_{2k}^T y| \|(\check{\zeta}_k \tilde{\check{v}}_k + \zeta_{k+1} s_{2m,2k} \tilde{v}_{k+1})\|$$

verwenden. Leider bedeutet ein kleiner Wert für die Norm des Residuums nicht, dass auch das Ritzpaar eine gute Näherung an ein Eigenpaar der Ausgangsmatrix ist (siehe einleitendes Beispiel in Abschnitt 3 aus [20]). Für nichtnormale Matrizen ist es nicht ausreichend, dass die Norm des Residuums hinreichend klein ist (vgl. [13]). Eine hinreichende Bedingung zur Beurteilung der Qualität des Ritzpaares ist nach [20] das Bestimmen des Abstandes von H zu $(H-E)$, wobei $(H-E)$ die zu H nächstgelegene Matrix ist, für die das Ritztripel (λ, x, y) exakt ist. Das Ritztripel (λ, x, y) ist ein Eigentripel der kleineren Matrix $\tilde{H}^{2k,2k}$, dabei ist λ Eigenwert und x bzw. y Rechts- bzw. Linskeigenvektor. Im Folgenden soll nun die Berechnung der in [20] im Theorem 2' bewiesenen Schranken genauer vorgestellt werden.
Nehmen wir an, dass die Matrix $\tilde{H}^{2k,2k}$ diagonalisierbar ist. Somit existiert ein $Y \in$

$\mathbb{C}^{2k,2k}$ mit folgender Eigenschaft

$$Y^{-1}\tilde{H}^{2k,2k}Y = \begin{bmatrix} -\lambda_1 & & & & & & \\ & \ddots & & & & & \\ & & -\lambda_k & & & & \\ & & & \lambda_1 & & & \\ & & & & \ddots & & \\ & & & & & \lambda_k & \end{bmatrix} = \Lambda,$$

auch kann die Matrix Y symplektisch gewählt werden. Sei $X = S^{2n,2k}Y = [x_1, \cdots, x_{2k}]^T$ und das Residuum $\tilde{r}_k$ sei wieder $\check{\zeta}_k\check{\tilde{v}}_k + \zeta_{k+1}s_{2m,2k}\tilde{v}_{k+1}$. Aus der Gleichung (3.23) erhält man die Beziehung

$$H^{2n,2n}S^{2n,2k}Y = S^{2n,2k}YY^{-1}\tilde{H}^{2k,2k}Y + \tilde{r}_k e_{2k}^T Y$$

oder auch $HX = X\Lambda + \tilde{r}_k e_{2k}^T Y$. Eine Betrachtung der Spalten des letzten Ausdrucks kann für die Abschätzung der Güte des Ritzpaares hilfreich sein. Es gilt

$$Hx_i = -\lambda_i x_i + y_{2k,i}\tilde{r}_k \tag{3.25}$$

und

$$Hx_{k+i} = \lambda_i x_{k+i} + y_{2k,k+i}\tilde{r}_k \tag{3.26}$$

für $i = 1, \ldots, k$. Das Theorem 2' aus [20] liefert eine Berechnunsmöglichkeit für die Norm der Matrix $\|E_{\lambda_i}\|$, die den Rückwärtsfehler bei der Berechnung des Eigenwertes λ_i beschreibt. Dafür werden die Eigentripel der Matrix $(H - E_{\lambda_i})$ benötigt, das heißt zum Eigenwert λ_i einen Linkseigenvektor und einen Rechtseigenvektor. Unter Ausnutzung der Hamiltonischen Struktur von H und den Eigenschaften von J kann die Beziehung (3.25) die folgende Gleichung liefern:

$$\begin{array}{lcl} Hx_i & = & -\lambda_i x_i + y_{2k,i}\tilde{r}_k \\ JHx_i & = & -\lambda_i Jx_i + y_{2k,i}J\tilde{r}_k \\ H^T(Jx_i) & = & \lambda_i Jx_i - y_{2k,i}J\tilde{r}_k \\ (Jx_i)^T H & = & \lambda_i (Jx_i)^T + y_{2k,i}\tilde{r}_k^T J \end{array} .$$

In analoger Weise wird auch die Gleichung (3.26) ausgenutzt um das Nachstehende zu erhalten:

$$\begin{array}{lcl} Hx_{k+i} & = & \lambda_i x_{k+i} + y_{2k,k+i}\tilde{r}_k \\ JHx_{k+i} & = & \lambda_i Jx_{k+i} + y_{2k,k+i}J\tilde{r}_k \\ -H^T Jx_{k+i} & = & \lambda_i Jx_{k+i} + y_{2k,v}J\tilde{r}_k \\ (Jx_{k+i})^T H & = & -\lambda_i (Jx_{k+i})^T - y_{2k,k+i}\tilde{r}_k^T J^T \\ (Jx_{k+i})^T H & = & -\lambda_i (Jx_{k+i})^T + y_{2k,k+i}\tilde{r}_k^T J \end{array} .$$

Die so erhaltene Gleichung zeigt, dass $(-\lambda_i, x_i, (Jx_{k+i})^T)$ ein Eigentripel der Matrix $(H - E_{(-\lambda_i)})$ darstellt. Und auch das Tripel $(\lambda_i, x_{k+i}, (Jx_i)^T)$ erfüllt, dank der oben dargelegten Beziehung, die Bedingungen des Theorems 2' für die Matrix $(H - E_{\lambda_i})$. Nach Normierung der Rechts- und Linkseigenvektoren, lassen sich die Werte für die Normen von $\|E_{\lambda_i}\|$ und $\|E_{(-\lambda_i)}\|$ aufschreiben. Konkret ergibt sich für

$$\|E_{(-\lambda_i)}\| = \max\left\{ \frac{|y_{2k,i}|\|\tilde{r}_k\|}{\|x_i\|}, \frac{|y_{2k,k+i}|\|\tilde{r}_k^T J\|}{\|Jx_{k+i}\|} \right\} \tag{3.27}$$

und $\|E_{\lambda_i}\|$ läßt sich wie folgt berechnen:

$$\|E_{\lambda_i}\| = \max\left\{ \frac{|y_{2k,k+i}|\|\tilde{r}_k\|}{\|x_{k+i}\|}, \frac{|y_{2k,i}|\|\tilde{r}_k^T J\|}{\|Jx_i\|} \right\}. \tag{3.28}$$

Der Berechnung der Norm von $E = \tilde{r}_k e_k^T \left(S^{2n,2k}\right)^T J^n$ gegenüber steht nun die gerade vorgestellte Berechnung des Rückwärtsfehlers über die E_{λ_i}. Ihre Benutzung sollte im Verfahren bevorzugt werden, da die benutzten Größen noch für die Bestimmung anderer Stabilitätsparameter verwendet werden können, beispielsweise die Konditionszahlen der jeweiligen Eigenwerte (vgl. [16]).

Kapitel 4

Deflationstechniken und Krylov-Schur-artige Zerlegung

Um einen effizienten Lanczos-Prozess zu beschreiben, ist es unablässlich, Deflationstechniken einzuführen. Die grundlegenden Ideen wurden dabei von Sorensen und Lehoucq in [22] entwickelt. Im Rahmen dieser Arbeit sollen dabei im Wesentlichen das *Purging*[1] und *Locking*[2] betrachtet werden. Das *Purging* beschäftigt sich mit dem Bereinigen des Spektrums um konvergierte, aber unerwünschte Eigenwerte und das *Locking* kann als Arretierung von bereits konvergierten Eigenwerten aufgefasst werden. Diese beiden Techniken wurden bereits von Sorensen in [29] und [30] beschrieben, allerdings sind die dort vorgestellten Techniken recht aufwändig und kompliziert, besonders im Falle von komplexen Eigenwerten.
Um die einzelnen Techniken, die für das Umsetzen dieser beiden Verfahren notwendig sind, zu beschreiben, ist es hilfreich, sich mit dem von Stewart in [32] eingeführten Krylov-Schur Algorithmus zu beschäftigen. Dieser Algorithmus bietet die Möglichkeit, die Deflation im Sinne von *Purging* und *Locking* auf einfachem Wege umzusetzen. Das folgende Kapitel befasst sich mit der Adaption des Stewart'schen Algorithmus' für den impliziten Arnoldi-Prozess auf den symplektischen Lanczos-Prozess mit impliziten Restarts.

4.1 Die Krylov-Schur-artige Zerlegung

Zur Beginn dieses Abschnittes sei die von Stewart in [33] und [32] beschriebene Krylov Zerlegung eingeführt.

Definition 4.1.1. *Seien die Vektoren $s_1, s_2, \ldots, s_{2k}, s_{2k+1}$ linear unabhängig und sei $S_p^{2k} = (s_1, \ldots, s_{2k})$. Eine Zerlegung der folgenden Gestalt:*

$$H_p S_p^{2k} = S_p^{2k} \tilde{H}_p^{2k} + s_{2k+1} \tilde{h}_{2k+1}^T.$$

heißt Krylov Zerlegung der Ordnung $2k$. *Eine äquivalente Schreibweise dazu ist*

$$H_p S_p^{2k} = S_p^{2k+1} \tilde{H}_p^{2k+1},$$

mit $S_p^{2k+1} = (S_p^{2k} s_{2k+1})$ und

$$\tilde{H}_p^{2k+1} = \begin{bmatrix} \tilde{H}_p^{2k} \\ h_{2k+1}^T \end{bmatrix}.$$

[1] auf Deutsch Bereinigung
[2] im Deutschen Arretierung

Mit geringen Modifikationen der Definition 4.1.1 lässt sich eine Zerlegung gewinnen, die im weiteren Verlauf von großem Nutzen für die Umsetzung von Deflationstechniken ist.

Definition 4.1.2. *Seien die Vektoren $s_1, s_2, \ldots, s_{2k}, s_{2k+1}$ J-orthogonal und sei $S_p^{2k} = (s_1, \ldots, s_{2k})$, dann gilt $(S_p^{2k})^T J_p S_p^{2k} = J_p$. Eine Zerlegung der folgenden Gestalt:*

$$H_p S_p^{2k} = S_p^{2k} \tilde{H}_p^{2k} + s_{2k+1} \tilde{h}_{2k+1}^T.$$

heißt Krylov-Schur-artige Zerlegung der Ordnung $2k$*, wobei $\tilde{H}_p^{2k}$ eine permutierte Hamiltonische Matrix in Schurform ist. Eine äquivalente Schreibweise dazu ist*

$$H_p S_p^{2k} = S_p^{2k+1} \tilde{H}_p^{2k+1},$$

mit $S_p^{2k+1} = (S_p^{2k} s_{2k+1})$ und

$$\tilde{H}_p^{2k+1} = \begin{bmatrix} \tilde{H}_p^{2k} \\ h_{2k+1}^T \end{bmatrix}.$$

Ein wichtiger Schritt auf dem Weg zu effizienten Deflationsstrategien ist die Äquivalenz der bereits bekannten Lanczos-Faktorisierung und der neuen Krylov-Schur-artigen Zerlegung. Beginnen wir mit einer Lanczos-Faktorisierung der Größe $2k$:

$$H_p S_p^{2n,2k} = S_p^{2n,2k} \tilde{H}_p^{2k,2k} + r_k e_{2k}^T.$$

Aus dem SR Algorithmus erhält man eine symplektische Matrix S_p, mit der gilt

$$H_p S_p^{2n,2k} S_p = S_p^{2n,2k} S_p (S_p^{-1} \tilde{H}_p^{2k,2k} S_p) + r_k e_{2k}^T S_p. \tag{4.1}$$

Dabei ist die Matrix $S_p^{-1} \tilde{H}_p^{2k,2k} S_p$ von blockdiagonaler Gestalt, genauer gesagt sind die Blöcke von der Größe 2×2 oder 4×4 (vgl. Beziehung (2.1)). Dabei ist ein Block der Größe 2×2 mit den reellen Eigenwerten von $\tilde{H}_p$ zu identifizieren und die Blöcke der Größe 4×4 repräsentieren die komplexen Eigenwerte der ursprünglichen Matrix. Benutzt man die Bezeichnungen $\breve{S}_p = S_p^{2n,2k} S_p$ und $\breve{H}_p^{2k,2k} = (S_p^{-1} \tilde{H}_p^{2k,2k} S_p)$, sowie $s_p^T = e_{2k}^T S_p$, dann ergibt sich die folgende Zerlegung:

$$H_p \breve{S}_p^{2n,2k} = \breve{S}_p^{2n,2k} \breve{H}_p^{2k,2k} + r_k s_p^T. \tag{4.2}$$

Die so erhaltene Faktorisierung genügt den Anforderungen an eine Krylov-Schur-artige Zerlegung. Dies zeigt, dass eine Lanczos-Faktorisierung mittels symplektischer Transformationen in eine Krylov-Schur-artige Zerlegung überführt werden kann. Die interessante Frage ist nun, ob auch aus einer Krylov-Schur-artigen Zerlegung eine Lanczos-Faktorisierung gewonnen werden kann. Dazu sei

$$H_p S_p^{2k} = S_p^{2k} \tilde{H}_p^{2k} + s_{2k+1} \tilde{h}_{2k+1}^T \tag{4.3}$$

eine solche Zerlegung. Mit Hilfe des Algorithmus' 4.1 erzeugen wir eine symplektische orthogonale Matrix Q_p, so dass $\tilde{h}_{2k+1}^T Q_p = \alpha e_{2k}^T$. Multipliziert man (4.3) mit Q_p, dann erhält man

$$\begin{aligned} H_p S_p^{2k} Q_p &= S_p^{2k} Q_p (Q_p^T \tilde{H}_p^{2k} Q_p) + s_{2k+1} \tilde{h}_{2k+1}^T Q_p \\ &= S_p^{2k} Q_p (Q_p^T \tilde{H}_p^{2k} Q_p) + \alpha s_{2k+1} e_{2k}^T \\ &= \breve{S}_p^{2k} \breve{H}_p^{2k} + \alpha s_{2k+1} e_{2k}^T. \end{aligned} \tag{4.4}$$

Mit (4.4) ist man einer zulässigen Lanczos-Faktorisierung schon recht nahe gekommen. Um den Transformationsprozess zu vervollständigen, muss noch die Matrix $\breve{H}_p^{2k}$, die

Algorithmus 4.1 Symplektischer Algorithmus für $y^TQ = \alpha e_{2k}^T$

INPUT : $y^T \in \mathbb{R}^{2n}$
OUTPUT : $Q \in \mathbb{R}^{2n,2n}$, so dass $y^TQ = \alpha e_{2k}^T$

Q = I;
for $i = 1$ to k **do**
  Erzeuge Givensmatrix G mit c und s, so dass y_{2i-1} verschwindet.
  $Q = QG$;
end for
for $i = 1$ to $k-1$ **do**
  Erzeuge Givensmatrix G mit c und s, so dass y_{2i} verschwindet.
  $Q = QG$;
end for

an dieser Stelle voll besetzt ist, auf Hamiltonische J-Hessenberggestalt transformiert werden, ohne dabei den Residuenausdruck seiner Gestalt $r_k e_{2k}^T$ zu berauben. Der von Mehrmann/Bunse-Gerstner (vgl.[8]) vorgestellte Algorithmus JHESS kann an dieser Stelle nicht direkt verwendet werden, da er zwar eine Hamiltonische J-Hessenberg-Matrix erzeugt, aber auf Grund der Art der Transformationen keinen zulässigen Residuenausdruck für die Lanczos-Faktorisierung zurücklässt. Das Problem des ursprünglichen JHESS Algorithmus' ist das spaltenweise Eliminieren der Elemente. Wenn man also einen Algorithmus konstruieren kann, der die Hamiltonische J-Hessenberg-Matrix zeilenweise herstellt, dann ist die resultierende Transformation S von der Gestalt, dass $e_{2k}^T S = e_{2k}^T$. Der Algorithmus 4.2 zeigt wie ein zeilenweises JHESS-Verfahren umgesetzt werden kann. Die letzte Zeile der Matrix S ist nun ein Vielfaches des Einheitsvektors e_{2k}^T, also $e_{2k}^T S = \beta e_{2k}^T$. Benutzt man die permutierte Matrix S_p, deren letzte Zeile die gleiche Struktur hat wie die unpermutierte Matrix, um damit (4.4) von rechts zu multiplizieren, dann erhält man Folgendes:

$$\begin{aligned} H_p \breve{S}_p^{2k} S_p &= \breve{S}_p^{2k} S_p (S_p^{-1} \breve{H}_p^{2k} S_p) + \alpha s_{2k+1} e_{2k}^T S_p \\ &= \breve{S}_p^{2k} \breve{H}_p^{2k} + \alpha\beta s_{2k+1} e_{2k}^T \\ &= \breve{S}_p^{2k} \breve{H}_p^{2k} + \check{s}_{2k+1} e_{2k}^T. \end{aligned} \tag{4.5}$$

Es ist gelungen, die Krylov-Schur-artige Zerlegung (4.3) in eine Lanczos-Faktorisierung (4.5) zu überführen.
Was ist nun der Vorteil dieser Krylov-Schur-artigen Zerlegung? Betrachtet man die folgende Zerlegung, dabei sei der Index für die Permutierung weggelassen:

$$H(S_1\ S_2) = (S_1\ S_2) \begin{bmatrix} \tilde{H}_{11} & 0 \\ 0 & \tilde{H}_{11} \end{bmatrix} + s_{2k+1}(h_1^T\ h_2^T). \tag{4.6}$$

Die Struktur der Matrix

$$\begin{bmatrix} \tilde{H}_{11} & 0 \\ 0 & \tilde{H}_{22} \end{bmatrix} \tag{4.7}$$

resultiert aus der Entstehung der Zerlegung (4.6), denn wie bereits oben beschrieben, erbringt die Transformation mit der symplektischen Matrix, die im SR Algorithmus entsteht, eine Blockdiagonalmatrix mit Diagonalelementen der Größe 2×2 oder 4×4. Wenn man die Blöcke der Matrix (4.7) so tauschen kann, dass diejenigen mit den gewünschten Eigenwerten links oben stehen und die unerwünschten in der südöstlichen Ecke der Matrix

$$\begin{bmatrix} \breve{H}_{11} & 0 \\ 0 & \breve{H}_{22} \end{bmatrix}, \tag{4.8}$$

dann erhält man eine neue, kleinere Krylov-Zerlegung der folgenden Gestalt:

$$H\breve{S}_1 = \breve{S}_1\breve{H}_{11} + s_{2k+1}\breve{h}_1^T. \tag{4.9}$$

Für (4.9) gilt, dass $\sigma(\breve{H}_{11})$ genau dem Teil des Spektrums entspricht, an dem wir interessiert sind. Es kann also mit der Zerlegung (4.9) weitergearbeitet werden. Im Gegensatz zu oberen Hessenbergmatrizen lässt sich das Tauschen von Eigenblöcken in Blockdiagonalmatrizen einfach realisieren und soll im nächsten Abschnitt vorgestellt werden.

Algorithmus 4.2 Zeilenweiser JHESS Algorithmus

INPUT : $M \in \mathbb{R}^{2n,2n}$ Hamiltonische Matrix
OUTPUT : $T \in \mathbb{R}^{2n,2n}$ und $H \in \mathbb{R}^{2n,2n}$, so dass $H = T^{-1}MT$ in Hamiltonischer J-Hessenbergform ist.

for $j = 1$ to $n-1$ **do**
  for $k = 1$ to $n-j$ **do**
    Erzeuge Givensmatrix G vom Typ I mit c und s, um $M_{2n-j+1,k}$ zu eliminieren.
    $M = G^T MG$;
    $S = SG$;
  end for
  Erzeuge Householdermatrix H, um $M_{2n-j+1,n+2}, \ldots, M_{2n-j+1,2n-j}$ auszulöschen.
  $M = H^T MH$;
  $S = SH$;
  if $M_{2n-j+1,2n-j} \neq 0$ und $M_{2n-j+1,n-j+1} = 0$ **then**
    Zerlegung existiert nicht.
  else
    Erzeuge Gaußmatrix L für $n-j+1$ mit den Parametern c und d, so dass $M_{2n-j+1,2n-j}$ verschwindet.
    $M = L^{-1}ML$;
    $S = SL$;
  end if
  for $k = j+1$ to n **do**
    Erzeuge Givensmatrix vom Typ II für $n-k+1$ mit den Parametern c und $-s$, um $M_{2n-k+1,2n-j+1}$ auszulöschen.
    $M = R^T MR$;
    $S = SR$;
  end for
  Erzeuge Householdermatrix H, um $M(n-j+1, n+2 : 2n-j)$ zu eliminieren.
  $M = H^T MH$;
  $S = SH$;
end for

4.2 Tauschen von Eigenwerten und Eigenblöcken

Das Tauschen von Eigenblöcken spielt für die Deflation von Krylov-Zerlegungen eine bedeutende Rolle (vgl.[32]). Eine allgemeine Diskussion des Tauschens von Eigenblöcken ist dabei in [4] zu finden. Die Grundlage der nachfolgenden Überlegungen bildet die Lanczos-Rekursion

$$H_p S_p^{2n,2k} = S_p^{2n,2k} \tilde{H}_p^{2k,2k} + r_k e_{2k}^T \tag{4.10}$$

der Größe $2k$ und die dazu äquivalente Krylov-Schur-artige Zerlegung

$$H_p \breve{S}_p^{2n,2k} = \breve{S}_p^{2n,2k} \breve{H}_p^{2k,2k} + \breve{r}_k q_p^T. \tag{4.11}$$

Um Aussagen über die Struktur der Matrix $H_p^{2k,2k}$ zu treffen, sei noch einmal auf ihre Entstehung mittels des SR Algorithmus' eingegangen. Die Matrix $\tilde{H}_p^{2k,2k}$ ist eine permutierte Hamiltonische J-Hessenberg-Matrix

$$\tilde{H}_p^{2k,2k} = \left[\begin{array}{cc|cc|cc|cc|cc} \delta_1 & \beta_1 & 0 & \zeta_2 & & & & & & \\ \nu_1 & -\delta_1 & 0 & 0 & & & & & & \\ \hline 0 & \zeta_2 & \delta_2 & \beta_2 & 0 & \zeta_3 & & & & \\ 0 & 0 & \nu_2 & -\delta_2 & 0 & 0 & & & & \\ \hline & & 0 & \zeta_3 & \ddots & & \ddots & & & \\ & & 0 & 0 & & \ddots & & \ddots & & \\ \hline & & & & \ddots & & \ddots & & 0 & \zeta_n \\ & & & & & \ddots & & \ddots & 0 & 0 \\ \hline & & & & & & 0 & \zeta_n & \delta_n & \beta_n \\ & & & & & & 0 & 0 & \nu_n & -\delta_n \end{array}\right].$$

Startet man den SR Algorithmus für diese Matrix, so erhält man im Ergebnis eine Transformationsmatrix S_p für die gilt, dass $\breve{H}_p^{2k,2k} = S_p^{-1} \tilde{H}_p^{2k,2k} S_p$ eine Blockdiagonalmatrix mit Blöcken der Größe 2×2 oder 4×4 ist. Sei nun $\breve{H}_p^{2k,2k}$ von folgender Gestalt:

$$\breve{H}_p^{2k,2k} = \begin{bmatrix} \breve{H}_p^{(11)} & & & & \\ & \breve{H}_p^{(22)} & & & \\ & & \breve{H}_p^{(33)} & & \\ & & & \ddots & \\ & & & & \breve{H}_p^{(ll)} \end{bmatrix}.$$

Ohne Beschränkung der Allgemeinheit sei das Tauschen von Blöcken der Matrix $\breve{H}_p^{2k,2k}$ am Blockdiagonalelement $\breve{H}_p^{(33)} =: \breve{B}$ beschrieben. Die Blockelemente von $\breve{H}_p^{2k,2k}$ können nur von der Größe 2×2 oder 4×4 sein, damit müssen für die Matrix $\breve{B}$ nur vier Fälle betrachtet werden. Sei

$$\breve{B} = \begin{bmatrix} \breve{B}_{11} & 0 \\ 0 & \breve{B}_{22} \end{bmatrix},$$

dann ergeben sich die folgenden vier Möglichkeiten: $\dim(\breve{B}_{11}) = 2$ und $\dim(\breve{B}_{22}) = 2$, $\dim(\breve{B}_{11}) = 2$ und $\dim(\breve{B}_{22}) = 4$, $\dim(\breve{B}_{11}) = 4$ und $\dim(\breve{B}_{11}) = 2$, sowie $\dim(\breve{B}_{22}) = 4$ und $\dim(\breve{B}_{22}) = 4$.

1.Fall: $\dim(\breve{B}_{11}) = 2$ und $\dim(\breve{B}_{22}) = 2$

Um auch die Transformationen, die für das Tauschen der Eigenblöcke notwendig sind, symplektisch zu wählen, sei die folgende Matrix $T_p(n)$ eingeführt:

$$T_p(n) = \begin{bmatrix} & & & & & & 1 \\ & & & & & -1 & \\ & & & & 1 & & \\ & & & \iddots & & & \\ & & & -1 & & & \\ & & 1 & & & & \\ & -1 & & & & & \end{bmatrix} \in \mathbb{R}^{2n,2n}.$$

Diese Matrix erfüllt die Bedingungen für eine permutierte symplektische Transformation, das heiï¿½ $T_p(n)^{-1} J_p T_p(n) = J_p$. Da $T_p(n)$ ebenfalls orthogonal ist, lässt sich die Inverse aus der Beziehung $T_p(n)^T = -T_p(n)$ bestimmen. Betrachten wir nun die Matrix $\breve{B}$, die im ersten Fall folgende Gestalt besitzt:

$$\breve{B} = \begin{bmatrix} -\breve{b}_{11} & \breve{b}_{12} & & \\ 0 & \breve{b}_{11} & & \\ & & -\breve{b}_{33} & \breve{b}_{34} \\ & & 0 & \breve{b}_{33} \end{bmatrix},$$

dann kann mittels der Transformation mit $T_p(2)$ ein Tauschen der Eigenblöcke bewirkt werden. Man erhält

$$-T_p(2)\breve{B}T_p(2) = \begin{bmatrix} \breve{b}_{33} & & & \\ -\breve{b}_{34} & -\breve{b}_{33} & & \\ & & \breve{b}_{11} & \\ & & \breve{b}_{12} & -\breve{b}_{11} \end{bmatrix}.$$

2.Fall: $\dim(\breve{B}_{11}) = 2$ und $\dim(\breve{B}_{22}) = 4$
Der zweite Fall behandelt eine Matrix $\breve{B}$ mit der folgenden Gestalt:

$$\breve{B} = \begin{bmatrix} -\breve{b}_{11} & \breve{b}_{12} & & & & \\ 0 & \breve{b}_{11} & & & & \\ & & -\breve{b}_{33} & \breve{b}_{34} & \breve{b}_{35} & \breve{b}_{36} \\ & & 0 & \breve{b}_{33} & 0 & \breve{b}_{46} \\ & & -\breve{b}_{35} & \breve{b}_{36} & -\breve{b}_{55} & \breve{b}_{56} \\ & & 0 & -\breve{b}_{46} & 0 & \breve{b}_{55} \end{bmatrix}.$$

Um die beiden Blöcke dieser Blockdiagonalmatrix zu tauschen, ist eine Transformation mit $T_p(3)$ erforderlich. Das Ergebnis ist

$$-T_p(3)\breve{B}T_p(3) = \begin{bmatrix} \breve{b}_{55} & 0 & -\breve{b}_{46} & 0 & & \\ -\breve{b}_{56} & -\breve{b}_{55} & -\breve{b}_{36} & -\breve{b}_{35} & & \\ \breve{b}_{46} & 0 & \breve{b}_{33} & 0 & & \\ -\breve{b}_{36} & \breve{b}_{35} & -\breve{b}_{34} & \breve{b}_{33} & & \\ & & & & \breve{b}_{11} & 0 \\ & & & & -\breve{b}_{12} & -\breve{b}_{11} \end{bmatrix}.$$

3.Fall: $\dim(\breve{B}_{11}) = 4$ und $\dim(\breve{B}_{22}) = 2$
In dem nun betrachteten Fall ist die Matrix $\breve{B}$ von folgender Gestalt:

$$\breve{B} = \begin{bmatrix} -\breve{b}_{11} & \breve{b}_{12} & \breve{b}_{13} & \breve{b}_{14} & & \\ 0 & \breve{b}_{11} & 0 & \breve{b}_{24} & & \\ -\breve{b}_{13} & \breve{b}_{14} & -\breve{b}_{33} & \breve{b}_{34} & & \\ 0 & -\breve{b}_{24} & 0 & \breve{b}_{33} & & \\ & & & & -\breve{b}_{55} & \breve{b}_{56} \\ & & & & 0 & \breve{b}_{55} \end{bmatrix}.$$

Mit Hilfe der Matrix $T_p(3)$ ist es auf symplektischem Wege möglich, die Eigenblöcke von $\breve{B}$ zu tauschen. So erhält man

$$-T_p(3)\breve{B}T_p(3) = \begin{bmatrix} \breve{b}_{55} & 0 & & & & \\ -\breve{b}_{56} & -\breve{b}_{55} & & & & \\ & & \breve{b}_{33} & 0 & -\breve{b}_{24} & 0 \\ & & -\breve{b}_{34} & -\breve{b}_{33} & -\breve{b}_{14} & -\breve{b}_{13} \\ & & \breve{b}_{24} & 0 & \breve{b}_{11} & 0 \\ & & -\breve{b}_{14} & \breve{b}_{13} & -\breve{b}_{12} & -\breve{b}_{11} \end{bmatrix}.$$

4.Fall: $\dim(\breve{B}_{11}) = 4$ und $\dim(\breve{B}_{22}) = 4$
Der letzte Fall beschäftigt sich mit der folgenden Matrix $\breve{B}$:

$$\breve{B} = \begin{bmatrix} -\breve{b}_{11} & \breve{b}_{12} & \breve{b}_{13} & \breve{b}_{14} & & & & \\ 0 & \breve{b}_{11} & 0 & \breve{b}_{24} & & & & \\ -\breve{b}_{13} & \breve{b}_{14} & -\breve{b}_{33} & \breve{b}_{34} & & & & \\ 0 & -\breve{b}_{24} & 0 & \breve{b}_{33} & & & & \\ & & & & -\breve{b}_{55} & \breve{b}_{56} & \breve{b}_{57} & \breve{b}_{58} \\ & & & & 0 & \breve{b}_{55} & 0 & \breve{b}_{68} \\ & & & & -\breve{b}_{57} & \breve{b}_{58} & -\breve{b}_{77} & \breve{b}_{78} \\ & & & & 0 & -\breve{b}_{68} & 0 & \breve{b}_{77} \end{bmatrix}.$$

Und auch an dieser Stelle kann das gewünschte Tauschen der Blöcke mit Hilfe der orthogonalen Matrix $T_p(4)$ realisiert werden. Es ergibt sich bei der Transformation die nachstehende Matrix:

$$-T_p(4)\breve{B}T_p(4) = \begin{bmatrix} \breve{b}_{77} & 0 & -\breve{b}_{68} & 0 & & & & \\ -\breve{b}_{78} & -\breve{b}_{77} & -\breve{b}_{58} & -\breve{b}_{57} & & & & \\ \breve{b}_{68} & 0 & \breve{b}_{55} & 0 & & & & \\ -\breve{b}_{58} & \breve{b}_{57} & -\breve{b}_{56} & -\breve{b}_{55} & & & & \\ & & & & \breve{b}_{33} & 0 & -\breve{b}_{24} & 0 \\ & & & & -\breve{b}_{34} & -\breve{b}_{33} & -\breve{b}_{14} & -\breve{b}_{13} \\ & & & & \breve{b}_{24} & 0 & \breve{b}_{11} & 0 \\ & & & & -\breve{b}_{14} & \breve{b}_{13} & -\breve{b}_{12} & -\breve{b}_{11} \end{bmatrix}.$$

Somit ist es in allen vier Fällen gelungen, die Diagonalblöcke der Matrix $\breve{B}$ zu tauschen. Bettet man die Transformationsmatrix T_p in eine Einheitsmatrix der Größe $2k \times 2k$ ein, dann erbringt die Transformation der Ausgangsmatrix $\breve{H}_p^{2k,2k}$ mit der entstandenen Matrix eine Vertauschung der Blöcke von $\breve{B}$. Mit den nun geschaffenen Hilfsmitteln ist es kein Problem, die Eigenblöcke der Matrix $\breve{H}_p^{2k,2k}$ untereinander zu tauschen.

4.3 *Locking* und *Purging*

Aus dem letzten Abschnitt geht die Möglichkeit hervor Eigenblöcke der Matrix $\breve{H}_p^{2m,2m}$ zu tauschen, was zwar die Struktur der (2×2) und (4×4) Blöcke ändert, aber nicht die Verteilung der Eigenwerte auf die einzelnen Blöcke. Dazu sei erwähnt, dass *Locking* und *Purging* zu dem Zeitpunkt ansetzen, an dem sich die Matrix $\breve{H}_p^{2m,2m}$ in Blockdiagonalstruktur befindet. Nach der Durchführung der Operationen zum *Locking* und *Purging* findet die Reduktion von Krylov-Schur-artiger Zerlegung auf eine Lanczos-Faktorisierung statt, so dass nach Abschluss einer Iteration wieder die gewünschte Hamiltonische J-Hessenbergform hergestellt ist.
Nehmen wir einmal an, wir wollen den gewünschten Eigenwert λ_j locken, da das Konvergenzkriterium, welches im nächsten Abschnitt vorgestellt wird, erfüllt ist. Dann soll der zu λ_j gehörende Eigenblock in die linke obere Ecke der Matrix $\breve{H}_p^{2m,2m}$ transportiert werden, was mit den gerade vorgestellten Transformationen problemlos geschehen kann. Das Akkumulieren dieser Transformationen in der Matrix S_p bewirkt, dass die mit λ_j assoziierten Spalten an den Anfang von S_p transportiert werden. Die Lanczos-Faktorisierung kann wie bereits in (4.9) beschrieben verkleinert werden, wobei nicht mehr k Eigenwerte gesucht sind, sondern nur noch $k-1$. Das J-Reorthogonalisieren in den zukünftigen Schritten des Algorithmus muss allerdings gegen die gesamte Matrix S_p geschehen, damit sich der bereits gelockte Eigenwert λ_j nicht wieder in das Spektrum der Matrix drängt. Eine ähnliche Umsetzung empfiehlt sich für das Purgen eines konvergierten, aber unerwünschten Eigenwertes λ_i. Ist auch für diesen Wert das Konvergenzkriterium erfüllt, aber der Eigenwert λ_i erfüllt nicht die Kriterien an die gewünschten Eigenwerte, dann muss das Spektrum um diesen Wert bereinigt werden. Dazu wird der mit λ_i korrespondierende Block mit den bekannten Transformationen in die linke obere Ecke der Matrix $\breve{H}_p^{2m,2m}$ bewegt. Anschließend wird mit einer reduzierten Matrix weitergerechnet, ohne dabei auf das J-Reorthogonalisieren gegen die mit λ_i assoziierten Vektoren zu verzichten.

4.4 Abbruchkriterien

Es sei nun von der folgenden Hamiltonischen Lanczos-Zerlegung ausgegangen:

$$H_p \breve{S}_p^{2n,2k} = \breve{S}_p^{2n,2k} \breve{H}_p^{2k,2k} + r_k q_p^T. \tag{4.12}$$

Hier soll von der Betrachtung der permutierten Matrizen abgewichen werden. Dabei entsteht

$$H \breve{S}^{2n,2k} = \breve{S}^{2n,2k} \breve{H}^{2k,2k} + \breve{r}_k q^T. \tag{4.13}$$

Sei λ ein Eigenwert von $\breve{H}^{2k,2k}$ mit einem Eigenvektor y, dann gilt für $x = \breve{S}^{2n,2k} y$

$$\begin{aligned} \|Hx - \lambda x\| &= \|H\breve{S}^{2n,2k} y - \lambda \breve{S}^{2n,2k} y\| \\ &= \|(H\breve{S}^{2n,2k} - \breve{S}^{2n,2k}\breve{H}^{2k,2k})y\| \\ &= \|(\breve{r}_k q^T)y\| \end{aligned} . \tag{4.14}$$

Es kann nun in Analogie zum Kapitel 3.4 vorgegangen werden, so dass auch hier das Paar (λ, x) für eine leicht gestörte Matrix $(H-E)$ ein exaktes Eigenpaar ist. Die Matrix E hat hier die folgende Gestalt: $E = \breve{r}_k q^T \left(\breve{S}^{2n,2k}\right)^T J^n$. Sei wiederum angenommen, dass die Matrix $\breve{H}^{2k,2k}$ diagonalisierbar ist. Somit existiert ein $Y \in \mathbb{C}^{2k,2k}$ mit folgender

Eigenschaft

$$Y^{-1}\tilde{H}^{2k,2k}Y = \begin{bmatrix} -\lambda_1 & & & & & & \\ & \ddots & & & & & \\ & & -\lambda_k & & & & \\ & & & \lambda_1 & & & \\ & & & & \ddots & & \\ & & & & & \lambda_k & \end{bmatrix} = \Lambda.$$

Auch kann die Matrix Y symplektisch gewählt werden. Mit

$$X = \breve{S}^{2n,2k}Y = [x_1, \cdots, x_{2k}]^T$$

und der Gleichung (4.13) erhält man die Beziehung

$$H^{2n,2n}\breve{S}^{2n,2k}Y = \breve{S}^{2n,2k}YY^{-1}\breve{H}^{2k,2k}Y + \breve{r}_k q^T Y$$

oder auch $HX = X\Lambda + \breve{r}_k q^T Y$. Eine Betrachtung der Spalten des letzten Ausdrucks kann für die Abschätzung der Güte des Ritzpaares (λ, x) hilfreich sein. Es gilt

$$Hx_i = -\lambda_i x_i + \breve{r}_k q^T y_i \tag{4.15}$$

und

$$Hx_{k+i} = \lambda_i x_{k+i} + \breve{r}_k q^T y_{k+i} \tag{4.16}$$

für $i = 1, \ldots, k$. Mit Hilfe des Theorems 2' aus [20] kann analog zum Abschnitt 3.4 $\|E_{\lambda_i}\|$ berechnet werden. Die benötigten Eigentripel lassen sich aus den nachfolgenden Beziehungen erhalten. Die Umformungen werden unter Ausnutzung der Hamiltonischen Struktur von H und den Eigenschaften von J durchgeführt. Die Beziehung (3.25) wird zu:

$$\begin{array}{lcl} Hx_i & = & -\lambda_i x_i + \breve{r}_k q^T y_i \\ JHx_i & = & -\lambda_i Jx_i + J\breve{r}_k q^T y_i \\ H^T(Jx_i) & = & \lambda_i Jx_i - J\breve{r}_k q^T y_i \\ (Jx_i)^T H & = & \lambda_i (Jx_i)^T + \breve{r}_k^T q y_i^T J \end{array} . \tag{4.17}$$

In analoger Weise wird auch die Gleichung (3.26) ausgenutzt um das Nachstehende zu erhalten:

$$\begin{array}{lcl} Hx_{k+i} & = & \lambda_i x_{k+i} + \breve{r}_k q^T y_{k+i} \\ JHx_{k+i} & = & \lambda_i Jx_{k+i} J\breve{r}_k q^T y_{k+i} \\ -H^T Jx_{k+i} & = & \lambda_i Jx_{k+i} + J\breve{r}_k q^T y_{k+i} \\ (Jx_{k+i})^T H & = & -\lambda_i (Jx_{k+i})^T + \breve{r}_k^T q y_{k+1}^T J \end{array} .$$

Man erhält das Eigentripel $(-\lambda_i, x_i, (Jx_{k+i})^T)$ für die Matrix $(H - E_{(-\lambda_i)})$ und $(\lambda_i, x_{k+i}, (Jx_i)^T)$ für $(H - E_{\lambda_i})$. Unter diesen Voraussetzungen lassen sich die Werte für die Norm von E wie folgt aufschreiben:

$$\|E_{(-\lambda_i)}\| = \max\left\{ \frac{\|\breve{r}_k q^T y_i\|}{\|x_i\|}, \frac{\|\breve{r}_k^T q y_{k+i}^T J\|}{\|Jx_{k+i}\|} \right\} \tag{4.18}$$

und $\|E_{\lambda_i}\|$ läßt sich wie folgt bestimmen:

$$\|E_{\lambda_i}\| = \max\left\{ \frac{\|\breve{r}_k q^T y_{k+i}\|}{\|x_{k+i}\|}, \frac{\|\breve{r}_k^T q y_i^T J\|}{\|Jx_i\|} \right\}. \tag{4.19}$$

Mit diesen Abbruchkriterien ist es nun möglich einen kompletten Algorithmus basierend auf der Krylov-Schur-artigen Zerlegung zu programmieren, dargestellt in Algorithmus 4.3.

Algorithmus 4.3 Krylov-Schur-artiger Algorithmus

INPUT : $H \in \mathbb{R}^{2n,2n}$ und $k \in \mathbb{N}$
OUTPUT : $\lambda_1, \ldots, \lambda_{2k}$

Erzeuge eine Lanczos-Faktorisierung der Größe $2k$.
$H_p S_p^{2n,2k} = S_p^{2n,2k} \tilde{H}_p^{2k,2k} + \zeta_{k+1} v_{k+1} e_{2k}^T$
while $lockindex < 2 * k$ **do**
  Vergrößere bestehende Lanczos-Faktorisierung auf Größe $2m$.
  $H_p S_p^{2n,2m} = S_p^{2n,2m} \tilde{H}_p^{2m,2m} + \zeta_{m+1} v_{m+1} e_{2m}^T$
  Berechne die Eigenwerte und Eigenvektoren von $\tilde{H}_p^{2m,2m}$.
  Erzeugen der Krylov-Schur-artigen Faktorisierung.
  $H_p \breve{S}_p^{2n,2m} = \breve{S}_p^{2n,2m} \breve{H}_p^{2m,2m} + \zeta_{m+1} v_{m+1} s_p^T$
  Sortieren der Eigenwerte.
  Tauschen der Eigenblöcke von $\breve{H}_p^{2m,2m}$.
  Berechnen der $\|E_{\lambda_i}\|$ und $\|E_{-\lambda_i}\|$.
  if $\|E_{\lambda_i}\| \leq \varepsilon$ und $\|E_{-\lambda_i}\| \leq \varepsilon$ **then**
    if λ_i gewünschter Eigenwert **then**
      Locken des zu λ_i gehörigen Blockes.
    else
      (λ_i unerwünschter Eigenwert)
      Purgen des zu λ_i gehörigen Blockes.
    end if
    Erhöhen des lockindexes.
  end if
  Verkleinere die Krylov-Schur-artige Zerlegung:$\breve{H}_p^{2k,2k}$ und $\breve{S}_p^{2n,2k}$.
  (Krylov-Schur-artige Zerlegung auf Lanczos-Gestalt bringen)
  Berechne Q_p mit $s_p^T Q_p = \alpha e_k^T$.
  Akkumulieren der Transformation.
  $\breve{H}_p^{2k,2k} = Q_p \breve{H}_p^{2k,2k} Q_p^T$
  $\breve{S}_p^{2n,2k} = \breve{S}_p^{2n,2k} Q_p^T$
  Transformieren $\breve{H}_p^{2k,2k}$ auf Hamiltonische J-Hessenbergform (Algorithmus 4.2).
  $\breve{H}_p^{2k,2k} = \tilde{S}_p^{-1} \breve{H}_p^{2k,2k} \tilde{S}_p$
  $\breve{S}_p^{2n,2k} = \breve{S}_p^{2n,2k} \tilde{S}_p$
  $r_k = \alpha(\tilde{S}_p)_{2k,2k} r_k$
end while

4.5 Shift-und-Invert Strategien

Um einen vollständig funktionalen Algorithmus zu besitzen, reicht es nicht aus, nur die Eigenwerte mit größtem Betrag zu berechnen. Es ist auch notwendig, Eigenwerte in bestimmten Bereichen des Spektrums zu approximieren. Eine bewährte Methode, dies zu erreichen ist das Betrachten eines neuen Eigenwertproblems der Struktur

$$(H - \sigma I)^{-1} x = \nu x$$

, wobei $H \in \mathbb{R}^{2n,2n}$ die Ausgangsmatrix ist und σ den Wert darstellt, in dessen Nähe wir Eigenwerte suchen. Die Eigenwerte ν der Matrix $(H - \sigma I)^{-1}$ geben Aufschluss über die Eigenwerte der ursprünglichen Matrix H. Wenn ν ein Eigenwert von $(H - \sigma I)^{-1}$ ist, dann ist $\lambda = \frac{1}{\nu} + \sigma$ ein Eigenwert der Matrix H. Eine genaue Diskussion des Vorgehens im Shift-und-Invert Fall ist in [10] oder auch in [33] zu finden. Dieses Vorgehen ist bei Hamiltonischen Matrizen leider nicht anwendbar, da die Matrix $(H - \sigma I)^{-1}$ nicht von Hamiltonischer Struktur ist. Um dennoch Eigenwerte in bestimmten Bereichen des Spektrums zu berechnen, wurden in [23] und [3] spezielle Shift-und-Invert Probleme untersucht, die einen Ausweg aus diesem Dilemma bieten und im Folgenden kurz vorgestellt werden. Ist man an den kleinsten Eigenwerten der Matrix H interessiert, also einem Shift $\sigma = 0$, dann wird mit der inversen Matrix H^{-1} gerechnet. Im numerischen Sinne werden natürlich Gleichungssysteme der Gestalt $Hx = y$ gelöst. Die Matrix H^{-1} ist Hamiltonisch und somit sind die vorgestellten strukturerhaltenen Verfahren anwendbar. Liegt allerdings ein echter Shift $\sigma \in \mathbb{R}$ oder $\sigma \in i\mathbb{R}$ vor, dann kann die folgende Hamiltonische Matrix hilfreich sein:

$$H_2(\sigma) = (H - \sigma I)^{-1}(H + \sigma I)^{-1} H^{-1}.$$

Die Eigenwerte θ dieser Matrix lassen sich mittels des Polynoms $\lambda^3 - \sigma^2\lambda - \frac{1}{\theta} = 0$ auf die Eigenwerte λ der Ausgangsmatrix transformieren. Die Qualität der Lösungen λ_i dieses Polynoms, kann über die Bestimmung des jeweiligen Rückwärtsfehlers angegeben werden. Im Falle eines komplexen Shifts σ, ist es möglich mit Hilfe der Matrix

$$H_4(\sigma) = (H - \sigma I)^{-1}(H + \sigma I)^{-1}(H - \bar{\sigma} I)^{-1}(H + \bar{\sigma} I)^{-1} H^{-1}.$$

die Eigenwerte λ in der Nähe des Shifts σ zu erhalten. Dazu müssen die Wurzeln des Polynoms $\lambda^5 - (\sigma^2 + \bar{\sigma}^2)\lambda^3 + \sigma^2\bar{\sigma}^2\lambda - \frac{1}{\theta} = 0$ berechnet und ihre Güte über den Rückwärtsfehler bestimmt werden. Es ist also gelungen mit Hilfe geeigneter Matrizen eine Hamiltonische Shift-und-Invert Methode zu erhalten.

4.5.1 Das quadratische Eigenwertproblem

Ein Hamiltonisches Eigenwertproblem, dass sich gut mittels Shift-und-Invert Strategien behandeln lässt, ist das quadratische Eigenwertproblem. Für ausführliche Betrachtungen sei [3] und [23] empfohlen. Im quadratischen Eigenwertproblem werden ein $\lambda \in \mathbb{C}$ und ein $u \in \mathbb{R}^n$ gesucht, so dass gilt

$$(\lambda^2 M + \lambda G + K)u = 0, \tag{4.20}$$

mit Matrizen M, G und $K \in \mathbb{C}^{n,n}$. Für die weiteren Ausführungen beschäftigen wir uns mit einem speziellen quadratischen Eigenwertproblem in dem $M = M^T > 0$, $G = -G^T$ und $-K = -K^T > 0$. Üblicherweise wird dieses Eigenwertproblem mit Hilfe einer Linearisierung gelöst, diese sei hier $y = \lambda x$. Man erhält daraus die folgende Beziehung:

$$\left(\begin{bmatrix} 0 & -K \\ M & 0 \end{bmatrix} - \lambda \begin{bmatrix} M & G \\ 0 & M \end{bmatrix} \right) \begin{bmatrix} y \\ x \end{bmatrix} = 0 \tag{4.21}$$

Das so entstandene Matrizenbüschel ist ein Hamiltonisches/Schief-Hamiltonisches Matrizenbüschel, da die Matrix $\begin{bmatrix} 0 & -K \\ M & 0 \end{bmatrix}$ Hamiltonisch ist und die Matrix $N = \begin{bmatrix} M & G \\ 0 & M \end{bmatrix}$ Schief-Hamiltonisch, dass heißt $(JN)^T = -JN$. Die Matrix N kann in folgender Weise zerlegt werden:

$$N = Z_1 Z_2 = \begin{bmatrix} I & \frac{1}{2}G \\ 0 & M \end{bmatrix} \begin{bmatrix} M & \frac{1}{2}G \\ 0 & I \end{bmatrix}.$$

Da M symmetrisch positiv definit ist, gilt

$$Z_1^{-1}(H - \lambda N)Z_2^{-1} = H - \lambda I, \tag{4.22}$$

wobei H Hamiltonisch ist und die folgende Gestalt hat:

$$H = \begin{bmatrix} I & -\frac{1}{2}G \\ 0 & I \end{bmatrix} \begin{bmatrix} 0 & -K \\ M^{-1} & 0 \end{bmatrix} \begin{bmatrix} I & -\frac{1}{2}G \\ 0 & I \end{bmatrix}. \tag{4.23}$$

4.5.2 Shift-und-Invert für das quadratische Eigenwertproblem

Sei $\sigma \in \mathbb{R}$ oder $\sigma \in \mathbb{C}$ die Zahl, in deren Nähe wir Eigenwerte der Matrix H (vgl. (4.23)) suchen. Wir benutzen die im Abschnitt 4.5 vorgestellten Matrizen $H_2(\sigma)$ im reellen oder rein imaginären Fall und $H_4(\sigma)$ im Falle eines komplexen Shifts. Das numerische Lösen dieser Ausdrücke ist umständlich und nicht zu empfehlen, da eine Vielzahl von Gleichungssystemen gelöst werden muss. Mit Hilfe der Definition von H lassen sich aber Vereinfachungen herbeiführen. So gilt für H^{-1}

$$H^{-1} = \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} 0 & M \\ -K^{-1} & 0 \end{bmatrix} \cdot \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix} \tag{4.24}$$

und für $(H - \sigma I)^{-1}$, unter Benutzung der Abkürzung $Q(\sigma) = \sigma^2 M + \sigma G + K$,

$$(H - \sigma I)^{-1} = \begin{bmatrix} I & \frac{1}{2}G + \sigma M \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} 0 & M \\ -Q(\sigma)^{-1} & 0 \end{bmatrix} \cdot \begin{bmatrix} I & \frac{1}{2}G + \sigma M \\ 0 & I \end{bmatrix}. \tag{4.25}$$

Benutzt man (4.24) und (4.25), so lässt sich $H_2(\sigma)$ folgendermaßen aufschreiben:

$$\begin{aligned} H_2(\sigma) \;=\; & \begin{bmatrix} I & \frac{1}{2}G + \sigma M \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} 0 & M \\ -Q(\sigma)^{-1} & 0 \end{bmatrix} \cdot \begin{bmatrix} I & G \\ 0 & I \end{bmatrix} \cdot \\ & \begin{bmatrix} 0 & M \\ -Q(-\sigma)^{-1} & 0 \end{bmatrix} \cdot \begin{bmatrix} I & G - \sigma M \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} 0 & M \\ -K^{-1} & 0 \end{bmatrix} \cdot \\ & \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix} \end{aligned} \tag{4.26}$$

und $H_4(\sigma)$ wird zu:

$$\begin{aligned} H_4(\sigma) \;=\; & \begin{bmatrix} I & \frac{1}{2}G+\sigma M \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} 0 & M \\ -Q(\sigma)^{-1} & 0 \end{bmatrix} \cdot \begin{bmatrix} I & G \\ 0 & I \end{bmatrix} \cdot \\ & \begin{bmatrix} 0 & M \\ -Q(-\sigma)^{-1} & 0 \end{bmatrix} \cdot \begin{bmatrix} I & \frac{1}{2}G-\sigma M \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} I & \frac{1}{2}G+\bar{\sigma} M \\ 0 & I \end{bmatrix} \cdot \\ & \begin{bmatrix} 0 & M \\ -Q(\bar{\sigma})^{-1} & 0 \end{bmatrix} \cdot \begin{bmatrix} I & G \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} 0 & M \\ -Q(-\bar{\sigma})^{-1} & 0 \end{bmatrix} \cdot \\ & \begin{bmatrix} I & G-\bar{\sigma} M \\ 0 & I \end{bmatrix} \cdot \begin{bmatrix} 0 & M \\ -K^{-1} & 0 \end{bmatrix} \cdot \begin{bmatrix} I & \frac{1}{2}G \\ 0 & I \end{bmatrix} . \end{aligned} \tag{4.27}$$

Die Effizienz im Lösen dieses Problems wird durch das Lösen der Gleichungssysteme für $Q(\sigma)^{-1}$ und für K^{-1} bestimmt. Glücklicherweise gilt für die verschiedenen $Q(\sigma)$, dass $Q(-\sigma)) = Q(\sigma)^T$ und $Q(\bar{\sigma}) = \overline{Q(\sigma)}$. Wenn also eine LU-Zerlegung für die Matrix $Q(\sigma)$ berechnet ist, dann können auch die anderen Gleichungssysteme ohne neue Berechnung einer Zerlegung gelöst werden. Zur Behandlung des Problems für $Q(\sigma)$ wird in [3] und in [25] die SuperLU Zerlegung — beschrieben in [11] — empfohlen. Als vergleichbar mit der SuperLU Zerlegung, wird in [3] die Verwendung von UMFPack (vgl.[9]) vorgestellt. Um das Gleichungssystem für K^{-1} effizient zu lösen, muss die symmetrisch positiv definite Struktur dieser Matrix berücksichtigt werden, dazu ist das Verwenden einer Cholesky Zerlegung zu bevorzugen, zur Lösung dieses Problems empfiehlt C. Pester in [25] die Verwendung des TAUCS Paketes (vgl.[19]). Neueste Erkenntnisse legen die Verwendung des PARDISO Paketes (vgl. [24]) sowohl für das Lösen mit $Q(\sigma)$, als auch für die Bearbeitung von K^{-1} nahe.

4.5.3 "Positive-Real-Balancing"

Eine weitere Klasse von Hamiltonischen Matrizen, die sich gut zum Anwenden der Shift-und-Invert Strategien eignet, entsteht beim "Positive-Real-Balancing", mehr dazu in [18]. In diesem speziellen Fall hat die Hamiltonische Matrix H die folgende Struktur:

$$H = \begin{bmatrix} A+BC^T & BB^T \\ -CC^T & -(A^T+CB^T) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & -A^T \end{bmatrix} + \begin{bmatrix} B \\ -C \end{bmatrix} \begin{bmatrix} C^T & B^T \end{bmatrix}. \tag{4.28}$$

Mit Hilfe der *Sherman-Morrison-Woodbury* Formel kann die Inverse zur Hamiltonischen Matrix H berechnet werden, siehe [16]. Die Anwendung der Gleichung von *Sherman-Morrison-Woodbury* ergibt:

$$\begin{aligned} H^{-1} \;&=\; \begin{bmatrix} A^{-1} & \\ & A^{-T} \end{bmatrix} - \begin{bmatrix} A^{-1} & \\ & A^{-T} \end{bmatrix} \begin{bmatrix} B \\ -C \end{bmatrix} \left(I_m + F - F^T\right)^{-1} \begin{bmatrix} C^T & B^T \end{bmatrix} \begin{bmatrix} A^{-1} & \\ & A^{-T} \end{bmatrix} \\ &=\; \begin{bmatrix} A^{-1} & \\ & A^{-T} \end{bmatrix} - \begin{bmatrix} \tilde{B} \\ -\tilde{C} \end{bmatrix} \left(I_m + F - F^T\right)^{-1} \begin{bmatrix} \tilde{C}^T & \tilde{B}^T \end{bmatrix} \\ &=\; \begin{bmatrix} A^{-1} & \\ & A^{-T} \end{bmatrix} - \begin{bmatrix} \tilde{B} \\ -\tilde{C} \end{bmatrix} \begin{bmatrix} \check{C}^T & \check{B}^T \end{bmatrix} \end{aligned}$$

mit $F = C^T A^{-1} B \in \mathbb{R}^{m,m}$. Dabei ist zu beachten, dass die eben beschriebene Formel für H^{-1} nur einmal am Anfang des Lanczos-Prozesses berechnet werden muss. Im weiteren

Verlauf muss lediglich mit dieser Matrix multipliziert werden. Es ist leicht zu sehen, dass der Hauptaufwand im Lösen der Gleichungssyteme mit A und $\left(I_m + F - F^T\right)$ liegt. Dabei birgt gerade das Lösen mit $\left(I_m + F - F^T\right)$ enorme Effizienzvorteile gegenüber dem Rechnen mit der Ausgangsmatrix H, da die Dimension des Gleichungssystems nur noch von der Größenordnung $(m \times m)$ ist.

Kapitel 5

Numerische Eigenschaften

5.1 Zusammenbruch der symplektischen Lanczos-Verfahrens

Bislang wurde immer angenommen, dass die SR Zerlegung der gegebenen Matrizen existiert. Startet man den Lanczos-Prozess mit einem Vektor $\tilde{v}_1$ und es kommt mit diesem Startvektor zu einem Zusammenbruch des Algorithmus', dann kann auch die SR Zerlegung von $(H-\mu I)$, $(H-\mu I)(H+\mu I)$ oder $(H-\mu_1 I)(H+\mu_1 I)(H-\mu_2 I)(H+\mu_2 I)$ nicht existieren. Wenn man den SR Algorithmus genauer studiert (vgl. [8]), dann ist klar, dass ein Zusammenbruch des Verfahrens nur durch die nichtorthogonalen Gaußmatrizen hervorgerufen werden kann. Der folgende Satz stellt einen Zusammenhang zwischen dem Zusammenbruch des symplektischen Lanczos-Algorithmus' und der Existenz der Gaußtransformationen her.

Satz 5.1.1. *1. Sei $\tilde{H}^{2k,2k}$ eine unreduzierte J-Hessenberg-Matrix, die (3.3) genügt. Weiterhin sei $\mu \in \mathbb{R}$ und $G_p(j,y)$ die j-te permutierte symplektische Gaußmatrix, die zum Erzeugen der SR Zerlegung von $(\tilde{H}_p^{2k,2k} - \mu I)$ benötigt wird. Wenn die vorhergehenden $(j-1)$ Gaußmatrizen existieren, dann schlägt das Erzeugen von $G_p(j,y)$ genau dann fehl, wenn $\breve{v}_j^T J_p H_p \breve{v}_j = 0$ mit $\breve{v}_j$ wie in (3.10).*

2. Für den Shift gelte $\mu \in \mathbb{R}$ und $G_p(j,y)$ sei die j-te permutierte symplektische Gaußmatrix, die zum Erzeugen der SR Zerlegung von $(\tilde{H}_p^{2k,2k} - \mu I)(\tilde{H}_p^{2k,2k} + \mu I)$ benötigt wird. Existieren die vorhergehenden $(j-1)$ Gaußmatrizen, schlägt das Erzeugen von $G_p(j,y)$ genau dann fehl, wenn $\breve{v}_j^T J_p H_p \breve{v}_j = 0$ mit $\breve{v}_j$ wie in (3.14).

3. Es gelte $\mu_1, \mu_2 \in \mathbb{C}$ oder $\mu_1, \mu_2 \in \mathbb{R}$. Ist $G_p(j,y)$ die j-te permutierte symplektische Gaussmatrix, die zum Erzeugen der SR Zerlegung von

$$(\tilde{H}_p^{2k,2k} - \mu_1 I)(\tilde{H}_p^{2k,2k} + \mu_1 I)(\tilde{H}_p^{2k,2k} - \mu_2 I)(\tilde{H}_p^{2k,2k} + \mu_2 I)$$

benötigt wird, dann schlägt das Erzeugen von $G_p(j,y)$ genau dann fehl — angenommen die vorhergehenden $(j-1)$ Gaussmatrizen existieren — wenn $\breve{v}_j^T J_p H_p \breve{v}_j = 0$ mit $\breve{v}_j$ wie in 3.14.

Beweis. Der Beweis des Satzes folgt dem Beweis des Theorems 3 aus [17].

1. Nehmen wir an, dass die ersten $j-1$ Gaußmatrizen mit $G_p(\frac{i+1}{2}, y_i)$, $i = 3, 5, \ldots, 2j-1$ existieren und es gelte

$$\begin{bmatrix} \hat{S}_p^{2j} & 0 \\ 0 & I \end{bmatrix} = J_p(1, c_1, s_1) \prod_{i=2}^{j} G_p(i, y_i) J_p(i, c_i, s_i).$$

Aus

$$H_p S_p^{2n,2j} = S_p^{2n,2j} \tilde{H}_p^{2j,2j} + \zeta_{j+1} v_{j+1} e_{2j}^T$$

erhält man

$$H_p \breve{S}_p^{2n,2j} = \breve{S}_p^{2n,2j} \breve{H}_p^{2j,2j} + \zeta_{j+1} v_{j+1} e_{2j}^T,$$

wobei $\breve{S}_p^{2n,2j} = S_p^{2n,2j} \hat{S}_p^{2n,2j}$ und $\breve{H}_p^{2j,2j} = \left(\hat{S}_p^{2n,2j}\right)^{-1} \tilde{H}_p^{2j,2j} \hat{S}_p^{2n,2j}$. Da gilt

$$\left(\breve{S}_p^{2n,2j}\right)^T J_p^n \breve{S}_p^{2n,2j} = J_p^n, \tag{5.1}$$

erhält man

$$-J_p^j \left(\breve{S}_p^{2n,2j}\right)^T J_p^n H_p \breve{S}_p^{2n,2j} = \breve{H}_p^{2j,2j}. \tag{5.2}$$

Betrachtet man den führenden $(2j+2) \times (2j+2)$ Block von

$$\begin{bmatrix} \hat{S}_p^{2j} & 0 \\ 0 & I \end{bmatrix}^{-1} \tilde{H}_p^{2k,2k} \begin{bmatrix} \hat{S}_p^{2j} & 0 \\ 0 & I \end{bmatrix}, \tag{5.3}$$

dann erhält man

$$\begin{bmatrix}
\breve{\delta}_1 & \breve{\beta}_1 & 0 & \breve{\zeta}_2 & & & & \\
\breve{\nu}_1 & -\breve{\delta}_1 & 0 & 0 & & & & \\
0 & \breve{\zeta}_2 & \ddots & & \ddots & & & \\
0 & 0 & & \ddots & & \ddots & & \\
 & & \ddots & & \breve{\delta}_j & \breve{\beta}_j & 0 & x_2 \\
 & & & \ddots & \breve{\nu}_j & -\breve{\delta}_j & 0 & -x_1 \\
 & & & & x_1 & x_2 & \breve{\delta}_{j+1} & \breve{\beta}_{j+1} \\
 & & & & 0 & 0 & \breve{\nu}_{j+1} & -\breve{\delta}_{j+1}
\end{bmatrix}.$$

Die leichte Störung der Hamiltonischen J-Hessenbergform durch x_1 und x_2 ist auf Grund der besonderen Gestalt von $\hat{S}_p^{2j}$ erklärbar. Aus dem Beweis des Satzes 2.1.2 ist bekannt, dass $\hat{S}_p^{2j}$ eine obere Hessenbergmatrix ist. Auf der anderen Seite kann der obere $(2j+2) \times (2j+2)$ Block dank

$$-J_p^j \left[\breve{S}_p^{2n,2j} \mid v_{j+1} \mid w_{j+1}\right]^T J_p^n H_p \left[\breve{S}_p^{2n,2j} \mid v_{j+1} \mid w_{j+1}\right]$$

wie folgt geschrieben werden:

$$\left[\begin{array}{c|cc}
\breve{H}_p^{2j,2j} & -J_p^j \left(\breve{S}_p^{2n,2j}\right)^T J_p^n H_p v_{j+1} & -J_p^j \left(\breve{S}_p^{2n,2j}\right)^T J_p^n H_p w_{j+1} \\
\hline
-w_{j+1}^T J_p^n H_p \breve{S}_p^{2n,2j} & \delta_{j+1} & \beta_{j+1} \\
v_{j+1}^T J_p^n H_p \breve{S}_p^{2n,2j} & \nu_{j+1} & -\delta_{j+1}
\end{array}\right].$$

Es gilt also für die Parameter x_1 und x_2

$$\begin{array}{rcl} x_1 & = & -w_{j+1}^T J_p^n H_p \breve{v}_j \\ x_2 & = & -w_{j+1}^T J_p^n H_p \breve{w}_j \end{array}.$$

Der nächste Schritt im JHESS Algorithmus eliminiert x_1 mittels einer Gaußtransformation G_p. Diese Transformation kann nur erzeugt werden, wenn der Parameter $\breve{\nu}_j \neq 0$. Ist also der Parameter $\breve{\nu}_j$ gleich Null, so kann die SR Zerlegung von $(\tilde{H}_p^{2k,2k} - \mu I)$ nicht generiert werden, was einen Zusammenbruch im symplektischen Lanczos-Prozess nach sich zieht. Die umgekehrte Aussage wird durch die Eindeutigkeit des symplektischen Lanczos-Verfahrens impliziert.

2. Der Beweis für den Doppelshift verläuft in analoger Weise. Die ersten $(j-1)$ Gaußmatrizen mögen existieren und es gelte

$$\begin{bmatrix} \hat{S}_p^{2j} & 0 \\ 0 & I \end{bmatrix} = R_p(1,c,s) \prod_{i=2}^{j} J_p(i+1,c,s) G_p(i,y) J_p(i,c,s) R_p(i,c,s).$$

Ein Vorgehen wie im Beweis der Einzelshiftaussage bringt für den oberen $(2j+2) \times (2j+2)$ Block von (5.3) zum einen

$$\begin{bmatrix} \breve{\delta}_1 & \breve{\beta}_1 & 0 & \breve{\zeta}_2 & & & & \\ \breve{\nu}_1 & -\breve{\delta}_1 & 0 & 0 & & & & \\ 0 & \breve{\zeta}_2 & \ddots & & \ddots & & & x_1 \\ 0 & 0 & & \ddots & & \ddots & & 0 \\ & & \ddots & & \breve{\delta}_j & \breve{\beta}_j & 0 & x_3 \\ & & & \ddots & \breve{\nu}_j & -\breve{\delta}_j & 0 & -x_2 \\ & & & x_1 & x_2 & x_3 & \breve{\delta}_{j+1} & \breve{\beta}_{j+1} \\ & & & & 0 & 0 & \breve{\nu}_{j+1} & -\breve{\delta}_{j+1} \end{bmatrix}$$

und zum anderen auf Grund von (5.2) folgendes:

$$\left[\begin{array}{c|cc} \breve{H}_p^{2j,2j} & -J_p^j \left(\breve{S}_p^{2n,2j}\right)^T J_p^n H_p v_{j+1} & -J_p^j \left(\breve{S}_p^{2n,2j}\right)^T J_p^n H_p w_{j+1} \\ \hline -w_{j+1}^T J_p^n H_p \breve{S}_p^{2n,2j} & \delta_{j+1} & \beta_{j+1} \\ v_{j+1}^T J_p^n H_p \breve{S}_p^{2n,2j} & \nu_{j+1} & -\delta_{j+1} \end{array}\right].$$

Es gilt für die Werte von x_1,x_2 und x_3

$$\begin{aligned} x_1 &= -w_{j+1}^T J_p^n H_p \breve{w}_{j-1} \\ x_2 &= -w_{j+1}^T J_p^n H_p \breve{v}_j \\ x_3 &= -w_{j+1}^T J_p^n H_p \breve{w}_j. \end{aligned}$$

Das JHESS Verfahren muss nun die zusätzlichen Einträge x_j eliminieren. Dazu werden die bereits bekannten symplektischen Transformationsmatrizen verwendet. Beispielsweise wird mit der Elimination des Elementes x_1 durch eine symplektische Givensrotation des Typs R_p begonnen. Im Verlaufe des Beseitigens der x_j ist es möglich, dass zusätzliche Elemente entstehen. Bis auf x_2 können alle Einträge durch orthogonale symplektische Transformationen beseitigt werden. Das Element x_2 allerdings muss mit einer symplektischen Gausstransformation beseitigt werden, was nur unter der Voraussetzung, dass $\breve{\nu}_j \neq 0$ ist, funktioniert. Die Argumentation aus dem ersten Teil des Beweises liefert den Schluss für den Doppelshiftfall.

3. In Analogie zu den ersten beiden Teilen, wird auch der Fall des Vierfachshifts bewiesen. Wiederum sei die Existenz der ersten $(j-1)$ Gaußtransformationen vorausgesetzt und es gelte

$$\begin{bmatrix} \hat{S}_p^{2j} & 0 \\ 0 & I \end{bmatrix} = W_p(1,3,w) \prod_{i=2}^{j} J_p(i+1) J_p(i) R_p(i) G_p(i) J_p(i+1) J_p(i) R_p(i+1) R_p(i),$$

die Parameter der Transformationsmatrizen seien der Überschaubarkeit wegen vernachlässigt. Ebenso wie im Doppel- und Einzelshiftfall benutzt man den oberen $(2j+2) \times (2j+2)$ Block von (5.3) und erhält

$$\left[\begin{array}{cc|cc|cc|cc|cc} \breve{\delta}_1 & \breve{\beta}_1 & 0 & \breve{\zeta}_2 & & & & & & \\ \breve{\nu}_1 & -\breve{\delta}_1 & 0 & 0 & & & & & & \\ \hline 0 & \breve{\zeta}_2 & \ddots & & \ddots & & & & & x_1 \\ 30 & 0 & & \ddots & & \ddots & & & & 0 \\ \hline & & \ddots & & \ddots & & 0 & \breve{\zeta}_j & & x_3 \\ & & & \ddots & & \ddots & 0 & 0 & & -x_2 \\ \hline & & & & 0 & \breve{\zeta}_j & \breve{\delta}_j & \breve{\beta}_j & 0 & x_5 \\ & & & & 0 & 0 & \breve{\nu}_j & -\breve{\delta}_j & 0 & -x_4 \\ \hline & & & x_1 & x_2 & x_3 & x_4 & x_5 & \breve{\delta}_{j+1} & \breve{\beta}_{j+1} \\ & & & 0 & 0 & 0 & 0 & 0 & \breve{\nu}_{j+1} & -\breve{\delta}_{j+1} \end{array}\right],$$

außerdem ergibt sich mit Hilfe von (5.2):

$$\left[\begin{array}{c|cc} \breve{H}_p^{2j,2j} & -J_p^j \left(\breve{S}_p^{2n,2j}\right)^T J_p^n H_p v_{j+1} & -J_p^j \left(\breve{S}_p^{2n,2j}\right)^T J_p^n H_p w_{j+1} \\ \hline -w_{j+1}^T J_p^n H_p \breve{S}_p^{2n,2j} & \delta_{j+1} & \beta_{j+1} \\ v_{j+1}^T J_p^n H_p \breve{S}_p^{2n,2j} & \nu_{j+1} & -\delta_{j+1} \end{array}\right].$$

Aus diesen beiden Darstellungen lassen sich folgende Werte ablesen:

$$\begin{aligned} x_1 &= -w_{j+1}^T J_p^n H_p \breve{w}_{j-2} \\ x_2 &= -w_{j+1}^T J_p^n H_p \breve{v}_{j-1} \\ x_3 &= -w_{j+1}^T J_p^n H_p \breve{w}_{j-1} \\ x_4 &= -w_{j+1}^T J_p^n H_p \breve{v}_j \\ x_5 &= -w_{j+1}^T J_p^n H_p \breve{w}_j \end{aligned}.$$

Der JHESS Algorithmus soll auch hier die störenden Einträge der Matrix beseitigen. Begonnen wird dabei mit der Elimination des Elements x_1 mittels einer symplektischen Givensrotation vom Typ I. Eine solche Transformation kann wiederum das Entstehen neuer Elemente bedeuten. Die so hinzugekommenen Elemente und die Elemente x_j können mit Hilfe orthogonaler symplektischer Transformationen eliminiert werden. Die einzige Ausnahme bildet das Element x_4. Zu seiner Beseitigung ist eine Gaußmatrix zu verwenden, die nur unter der Bedingung $\breve{\nu}_j \neq 0$ existiert. An dieser Stelle ist mit der bekannten Argumentation die Aussage des Quadrupelshifts bewiesen.

□

5.2 Stabilität

Die Lanczos-Rekursion ist so konstruiert, dass die entstehenden Vektoren alle J-orthogonal sind. Aus praktischer Sicht macht die endliche Arithmetik dieser Beziehung einen Strich durch die Rechnung. Ohne das J-Reorthogonalisieren der Vektoren ist das

Verfahren nicht mehr zu gebrauchen und instabil. Eine Möglichkeit, die bereits erzeugten Vektoren v_{j+1} und w_j zu J-Reorthogonalisieren, wird im folgenden dargestellt (vgl. [5]):

$$w_j = w_j + S_p^{2n,2j-2} J_p^{j-1} \left(S_p^{2n,2j-2}\right)^T J_p^n w_j \tag{5.4}$$

und

$$v_{j+1} = v_{j+1} + S_p^{2n,2j} J_p^j \left(S_p^{2n,2j}\right)^T J_p^n v_{j+1}. \tag{5.5}$$

Das Ausführen dieser J-Reorthogonalisierung ist teuer. Für den Vektor w_j sind es $16n(j-1)$ flops und $16nj$ für v_{j+1}. Die Gleichungen (5.5) und (5.4) beschreiben symplektische Gram-Schmidt-ähnliche Orthogonalisierungsverfahren, näher erläutert an der Beziehung (5.5). Betrachtet man die Beziehung $\left(S_p^{2n,2j}\right)^T J_p^n v_{j+1}$ genauer mit dem schief-symmetrischen Skalarprodukt $y^T J_p x =< y, x >_{J_p}$, dann erhält man:

$$\begin{bmatrix} v_1^T \\ w_1^T \\ \vdots \\ v_j^T \\ w_j^T \end{bmatrix} J_p^n v_{j+1} = \begin{bmatrix} < v_1, v_{j+1} >_{J_p} \\ < w_1, v_{j+1} >_{J_p} \\ \vdots \\ < v_j, v_{j+1} >_{J_p} \\ < w_j, v_{j+1} >_{J_p} \end{bmatrix}. \tag{5.6}$$

Das Analysieren von $S_p^{2n,2j} J_p^j$ ergibt

$$\begin{bmatrix} v_1 & w_1 & \cdots & v_j & w_j \end{bmatrix} J_p^j = \begin{bmatrix} -w_1 & v_1 & -w_2 & v_2 & \cdots & -w_j & v_j \end{bmatrix}. \tag{5.7}$$

Mit Hilfe der Formeln (5.6) und (5.7) ergibt sich die folgende, dem Gram-Schmidt-Verfahren ähnliche, Beziehung:

$$v_{j+1} = v_{j+1} - \sum_{i=1}^{j} w_i < v_i, v_{j+1} >_{J_p} + \sum_{i=1}^{j} v_i < w_i, v_{j+1} >_{J_p}. \tag{5.8}$$

Dieses symplektische Gram-Schmidt-ähnliche Verfahren wird im Algorithmus 5.1 dargestellt. Da das klassische Gram-Schmidt-Verfahren numerisch nicht sehr stabil ist, wird eine modifizierte Variante des Gram-Schmidt-Verfahrens verwendet, bei der die J-Orthogonalität der vorhergehenden Vektoren ausgenutzt wird, dargestellt in Algorithmus 5.2. Für den Vektor w_j kann analog vorgegangen werden. In [27] werden von Salam ebenfalls Techniken zum J-Orthogonalisieren und J-Reorthogonalisieren vorgestellt, deren Ergebnisse aber bei ersten numerischen Tests keine Verbesserungen zu den hier vorgestellten Verfahren zeigten. Der Aufwand dieser Verfahren war sogar höher, da die Vektoren in den von Salam präsentierten Algorithmen in unpermutiertem Zustand bearbeitet werden, die Matrix S_p der Lanczos-Vektoren aber in permutierter Form vorliegt.

Ein weiterer Punkt ist die Stabilität des SR Schrittes. Betrachtet man eine $(2k \times 2k)$ Hamiltonische J-Hessenberg-Matrix und deren SR Zerlegung, dann benutzt man nur $k-1$ Transformationen, die nichtorthogonal sind. Für diese nichtorthogonalen Transformationen gilt, dass es keine andere nichtorthogonale Transformation gibt, die den gewünschten Zweck erfüllt und eine kleinere Konditionszahl besitzt (vgl.[8]).

Algorithmus 5.1 Klassisches symplektisches Gram-Schmidt-ähnliches Verfahren

INPUT : $S_p^{2n,2j}$ und v_{j+1}
OUTPUT : v_{j+1}

$\tilde{v}_{j+1} = v_{j+1}$
for $i = 1 : j$ **do**
$\tilde{v}_{j+1} = \tilde{v}_{j+1} - w_i < v_i, v_{j+1} >_{J_p} + v_i < w_i, v_{j+1} >_{J_p}$
end for
$v_{j+1} = \tilde{v}_{j+1}$

Algorithmus 5.2 Modifiziertes symplektisches Gram-Schmidt-ähnliches Verfahren

INPUT : $S_p^{2n,2j}$ und v_{j+1}
OUTPUT : v_{j+1}

$\tilde{v}_{j+1} = v_{j+1}$
for $i = 1 : j$ **do**
$\tilde{v}_{j+1} = \tilde{v}_{j+1} - w_i < v_i, \tilde{v}_{j+1} >_{J_p} + v_i < w_i, \tilde{v}_{j+1} >_{J_p}$
end for
$v_{j+1} = \tilde{v}_{j+1}$

Kapitel 6

Numerische Ergebnisse

6.1 SR Algorithmus

Mit dem entwickelten SR Algorithmus können die Eigenwerte einer Hamiltonischen J-Hessenberg-Matrix auf effiziente Weise berechnet werden. Zur Analyse des Verfahrens soll daher die Genauigkeit der Ergebnisse mit den Resultaten der *eig* Routine aus Matlab verglichen werden. Außerdem wird die Iterationszahl im Bezug auf die Matrixgröße analysiert, sowie die maximale Konditionszahl der verwendeten Gaußtransformationen.

Beispiel 6.1.1. Das erste Beispiel ist dem CAREX Paket, siehe Beispiel 16 in [7], entlehnt. Dabei wird die Hamiltonische Matrix H mit dem JHESS Algorithmus (vgl. Algorithmus 4.2) auf Hamiltonische J-Hessenberggestalt transformiert. Es werden nun die Eigenwerte der Ausgangsmatrix mit der Matlab *eig* Routine bestimmt, sowie die Eigenwerte der J-Hessenberg Matrix mit dem SR Algorithmus. Die Ergebnisse für ein Beispiel der Größe 34 sind in der Abbildung 6.1 dargestellt, dabei werden auf Grund der Hamiltonischen Struktur nur die Eigenwerte mit negativem Realteil dargestellt.

eig	SR Algorithmus
-4.09007692547218	-4.09007692547218
-4.09007692547217	-4.09007692547217
-3.83317281079134	-3.83317281079134
-3.83317281079134	-3.83317281079134
-3.35764070604493	-3.35764070604494
-3.35764070604493	-3.35764070604493
-2.73657991827709	-2.73657991827709
-2.73657991827709	-2.73657991827702
-2.07265697232460	-2.07265697232467
-2.07265697232460	-2.07265697232460
-1.49292460661253	-1.49292460661264
-1.49292460661253	-1.49292460661253
-1.12803607263302	-1.12803607263302
-1.12803607263302	-1.12803607263302
-1.00907878741279	-1.00907878741279
-1.00907878741279	-1.00907878741279
-1.00000000000000	-1.00000000000000

Tabelle 6.1: Eigenwertberechnung mittels *eig* Routine und dem SR Algorithmus (maximale Konditionszahl 51.75)

Beispiel 6.1.2. Das folgende Beispiel stammt aus der zweiten Version des CAREX Paketes, siehe dazu Beispiel 2.9 in [1]. Auch hier soll ein Vergleich der Genauigkeit der erzielten Ergebnisse von Matlabs *eig* Routine mit dem programmierten SR Algorithmus durchgeführt werden. Da die Matlab Routine einen "Balancing" Algorithmus verwendet, wird zur Vergleichbarkeit der Ergebnisse für den SR Algorithmus ein "Balancing" Verfahren vorgeschaltet. Hierbei handelt es sich um die *habalance* Routine aus dem HAPACK[1] Paket. Die Matlab Routine wird auf die Hamiltonische Matrix

$$H = \begin{bmatrix} A & -G \\ -Q & -A^T \end{bmatrix}$$

angewandt, wobei für den SR Algorithmus zuerst ein "Balancing" angewendet wird, gefolgt von einer J-Hessenberg Reduktion. Die entstandene Matrix wird nun mit den Ergebnissen der *eig* Routine verglichen, dargestellt in Abbildung 6.2. Dabei ist zu beachten, dass der Übersichtlichkeit halber nur die Eigenwerte mit negativem Real- und Imaginärteil dargestellt werden. Für den SR Algorithmus bedeutet das keinerlei Informationsverlust, da er die Hamiltonische Symmetrie der Eigenwerte berücksichtigt, hingegen können bei der *eig* Routine durchaus kleine Abweichungen in den Hamiltonischen Eigenquadrupeln auftreten. Eine wichtige Größe, um die Qualität des SR Algorithmus zu bewerten ist das Bestimmen des relativen Fehlers $\frac{1}{|\lambda_{eig}|}|\lambda_{eig} - \lambda_{sr}|$. Dargestellt wird dieser Wert in der Abbildung 6.1, hierzu wird der Fehler aufgetragen gegen die Eigenwerte mit negativem Real- und Imaginärteil und dabei wird mit dem betragskleinsten Eigenwert begonnen.

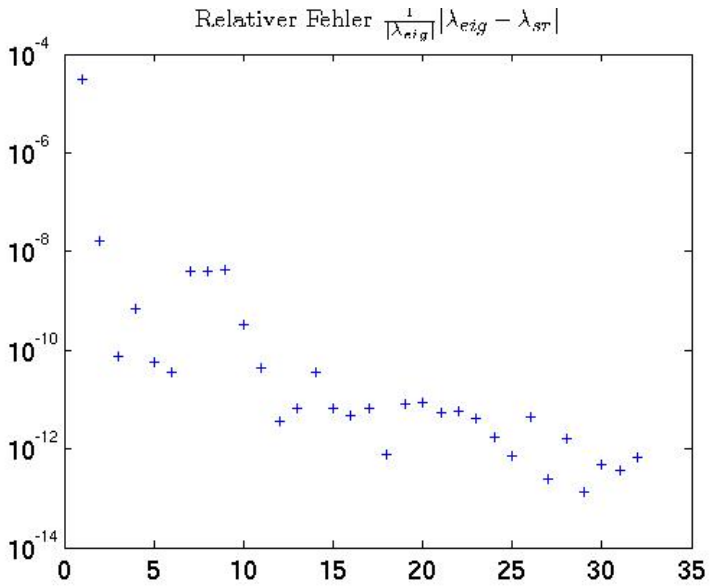

Abbildung 6.1: Relativer Fehler der Eigenwerte

[1]Softwarepaket und Dokumentation unter: *http://www.tu-chemnitz.de/mathematik/hapack/*

Matlab *eig* Routine	SR Algorithmus
1.0e+03*	1.0e+03*
-0.00002919299438 - 0.00004687016286i	-0.00002919327692 - 0.00004686800857i
-0.00051650000000 - 0.00000526782688i	-0.00051650000855 - 0.00000526779698i
-0.00308543080956 - 0.00412128377130i	-0.00308543080957 - 0.00412128377177i
-0.00530100000000	-0.00530099999637
-0.00034953881984 - 0.01434374704328i	-0.00034953881869 - 0.01434374704411i
-0.00147852876060 - 0.01984293097859i	-0.00147852876080 - 0.01984293097928i
-0.02000000000000	-0.02000000007751 - 0.00000000014357i
-0.02000000000000	-0.02000000007751 - 0.00000000014357i
-0.02003805181291 - 0.00006063302789i	-0.02003805172857 - 0.00006063301555i
-0.00040984539530 - 0.02221833347612i	-0.00040984539746 - 0.02221833346866i
-0.00243759251665 - 0.02216225512464i	-0.00243759251678 - 0.02216225512363i
-0.01276428442535 - 0.02115761031344i	-0.01276428442538 - 0.02115761031352i
-0.00307778652683 - 0.02714366804376i	-0.00307778652664 - 0.02714366804396i
-0.03327000000000	-0.03327000000115
-0.00099437738524 - 0.03616603716412i	-0.00099437738544 - 0.03616603716436i
-0.03999990517162	-0.03999990517143
-0.00631013144935 - 0.04003217558965i	-0.00631013144947 - 0.04003217558991i
-0.04260504103501	-0.04260504103504
-0.00566007295995 - 0.05080922555256i	-0.00566007296047 - 0.05080922555293i
-0.00544826592170 - 0.05317411493530i	-0.00544826592147 - 0.05317411493485i
-0.00333015750585 - 0.05637076782292i	-0.00333015750600 - 0.05637076782260i
-0.00283775147618 - 0.06376949407385i	-0.00283775147623 - 0.06376949407347i
-0.00441189434758 - 0.06944064603172i	-0.00441189434726 - 0.06944064603203i
-0.00344483694286 - 0.08867483953893i	-0.00344483694260 - 0.08867483953909i
-0.00577022639487 - 0.09295999844256i	-0.00577022639479 - 0.09295999844250i
-0.01172013359241 - 0.10930053857059i	-0.01172013359179 - 0.10930053857115i
-0.00822102146274 - 0.13909999369878i	-0.00822102146251 - 0.13909999369883i
-0.01035001483622 - 0.16299999878790i	-0.01035001483631 - 0.16299999878764i
-0.22120000000000	-0.22119999999997
-0.01179000000892 - 0.30459999995738i	-0.01179000000899 - 0.30459999995723i
-1.00000000023843	-1.00000000023879
-1.00000001783526	-1.00000001783596

Tabelle 6.2: Eigenwertberechnung mittels *eig* Routine und dem SR Algorithmus (maximale Konditionszahl 767)

Ein wichtiger Aspekt bei der Bewertung des SR Algorithmus' ist die Iterationszahl in Abhängigkeit von der Matrixgröße. Um diesen Aspekt für unterschiedliche Matrizen zu testen, werden im Folgenden Berechnungen für verschiedene zufällig erzeugte Matrizen durchgeführt.

Beispiel 6.1.3. In diesem Beispiel werden mit Hilfe des SR Algorithmus' die Eigenwerte einer Hamiltonischen J-Hessenberg-Matrix bestimmt. Die Eigenwerte der zufällig erzeugten Matrizen können sowohl reell als auch komplex sein, eine beispielhafte Verteilung ist in der Abbildung 6.2 dargestellt. Als Resultate werden die durchschnittliche maximale Konditionszahl der Gaußtransformationen, die durchschnittliche Iterationszahl, sowie die durchschnittliche Iterationszahl pro Eigenwert, sowie die Matrixgröße angegeben. Es werden pro Matrixgröße jeweils 10 Matrizen verwendet. Die Ergebnisse sind in der Tabelle 6.3 dargestellt. Bei den 100 Berechnungen mit dem SR-Verfahren kam es lediglich zu einem Zusammenbruch des Algorithmus', das heißt die Konditionszahl der Gaußmatrix war größer als die vorgegebene Schranke.

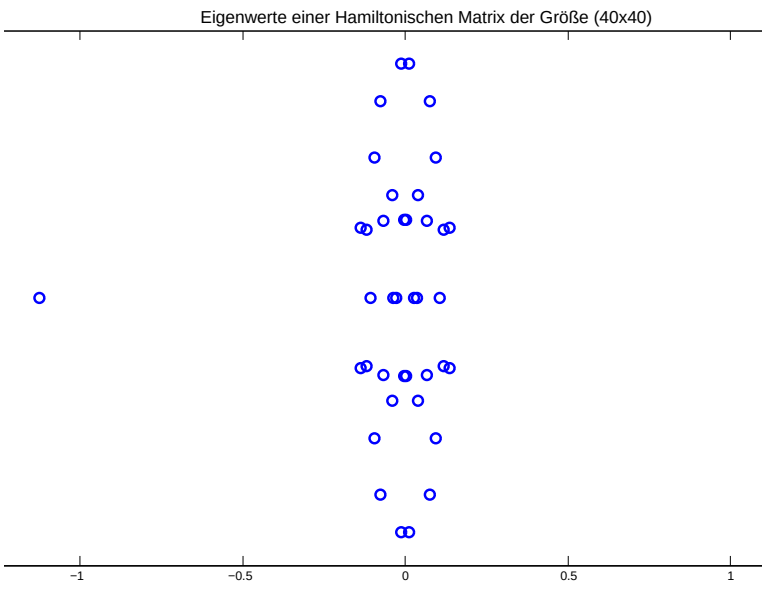

Abbildung 6.2: Eigenwerte einer Hamiltonischen Matrix der Größe (40×40)

Matrixgröße	Iterationszahl	Konditionszahl	Iteration pro Eigenwert
20	14.8	5.41e+003	0.740
40	29.6	9.30e+004	0.740
60	43.7	3.70e+005	0.728
80	53.5	1.18e+006	0.669
100	70.4	1.30e+005	0.704
120	83	5.36e+004	0.692
140	87.6	2.21e+005	0.626
160	108.3	6.99e+005	0.677
180	122.5	1.42e+006	0.681
200	137.9	1.76e+006	0.690

Tabelle 6.3: Ergebnisse des SR Algorithmus'

6.2 Krylov-Schur-artiger Algorithmus

6.2.1 Das quadratische Eigenwertproblem und die Fichera-Ecke

Im folgenden Beispiel werden spröde elastische Körper, bei denen in der Umgebung von Rissspitzen das lineare Materialgesetz (Lamé Gleichung) hinreichend gut die Wirklichkeit approximiert, betrachtet. Eine ausführliche Diskussion dieses Problems ist in [26] zu finden. Die Variationsformulierung für das zur Fichera-Ecke[2] korrespondierende 3D Elastizitätsproblem führt auf ein quadratisches Eigenwertproblem der folgenden Form:

$$(\lambda^2 M + \lambda G + K)u = 0, \tag{6.1}$$

wobei $M = M^T > 0$, $G = -G^T$ und $-K = -K^T > 0$. Dieses Eigenwertproblem wird durch einige spezifische Parameter bestimmt, deren Erläuterung ebenfalls [26] überlassen wird. Dies sind das Elastizitätsmodul E, die *Laméschen Konstanten* λ und ν, sowie der Öffnungswinkel ξ der Fichera-Ecke. Die Berechnung zur Fichera-Ecke werden auf Grund der sich anbietenden Transformation in Kugelkoordinaten auf dem Schnittgebilde von ursprünglicher Fichera-Ecke und Einheitskugel, siehe 6.3, durchgeführt. Die folgenden Daten zur Fichera-Ecke wurden freundlicherweise von C. Pester zur Verfügung gestellt. Dabei sind die *Laméschen Konstanten* wie folgt gewählt: $\mu = 0.5$ und $\nu = 0.3$ und der Öffnungswinkel betrage 90°.Es wird der in dieser Arbeit entwickelte Krylov-Schur-artige Algorithmus mit dem ebenfalls von C.Pester zur Verfügung gestellten Matlab Code zum Shira Algorithmus (vgl. [23]) und der Matlab Routine *eigs* verglichen. Ein großer Teil des Rechenaufwands entsteht beim Vergrößern der Faktorisierung, da an dieser Stelle mit der Ausgangsmatrix der Größe $(2n \times 2n)$ gearbeitet werden muss oder mit einer Zerlegung dieser, um Gleichungssysteme zu lösen. Dazu soll im folgenden der Aufwand der beiden Verfahren pro Iterationsschritt dargestellt werden, der durch das Vergrößern der Faktorisierung hervorgerufen wird. Der Fall $\sigma \in \mathbb{R}$ oder $\sigma \in i\mathbb{R}$ wird in der Tabelle 6.4 dargestellt. Das Hauptaugenmerk liegt hierbei auf den *saxpy*[3] Operationen, den Skalarprodukten, dem Lösen von schwachbesetzten Gleichungssystemen und Matrix-Vektor-Multiplikationen. Ein Aufwandsvergleich auf Grundlage eines Doppelshifts ist in der Tabelle 6.4 dargestellt.

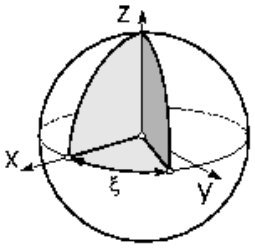

Abbildung 6.3: Schnitt zwischen Fichera-Ecke und Einheitskugel

Beispiel 6.2.1. Die Größe der betrachteten Matrizen M, G, K liegt hier bei $n = 5139$. Die jeweils verglichenen Verfahren werden mit dem selben Startvektor bedient. Der erste Vergleich findet zwischen dem aus [23] stammenden SHIRA Algorithmus und dem Krylov-Schur-artigen (KRSCHUR) Verfahren mit dem reellen Shift $\sigma = 1$ statt.

[2]Die Fichera-Ecke ist der Würfel $(-1, 1) \times (-1, 1) \times (-1, 1)$ ohne den Würfel $[0, 1] \times [0, 1] \times [0, 1]$

[3]ein *saxpy* entspricht dem Aufwand der Operation $ax + b$

	SHIRA	krschur
Skalarprodukte	$2(m-k)(m-k+1)$	$2(m-k)(m-k+1)+3(m-k)$
saxpys	$2(m-k)(m-k+1)+5(m-k)$	$2(m-k)(m-k+1)+8(m-k)$
Sparses Lösen	2	3

Tabelle 6.4: Aufwand von SHIRA und Krylov-Schur-artigem Algorithmus

Es werden 12 Eigenwerte gesucht und die Größe des Suchraums liegt in beiden Fällen bei 24 mit einer Abbruchschranke von 10^{-8}. Das Ergebnis ist in den Tabellen 6.5 und 6.6 dargestellt. Die Residuen in den beiden vorangestellten Verfahren werden über das Errechnen eines Eigenvektors mittels inverser Iteration, siehe Algorithmus 6.1 bestimmt.

Algorithmus 6.1 Berechnung des Residuums mittels Inverser Iteration

INPUT : $Q(\lambda) \in \mathbb{R}^{n,n}$ mit λ Eigenwertnäherung
OUTPUT : Residuum $res \in \mathbb{R}$

Erzeuge Startvektor $w \in \mathbb{R}^n$ mit $w \neq 0$.
Berechne $x = Q(\lambda)^{-1}w$.
Berechne $x = \frac{1}{\|x\|}x$.
Berechne $res = \frac{1}{\|Q(\lambda)\|_1}\|Q(\lambda)x\|$.

Ein Vergleich des KRSCHUR Verfahrens mit der Matlab *eigs*-Routine soll nun am Beispiel des Nullshifts vorgestellt werden, um 14 Eigenwerte des quadratischen Eigenwertproblems zu bestimmen. Dabei wird auch hier für beide Verfahren der gleiche Startvektor verwendet und die Größe des Suchraums betrage 22. Hier ist die Abbruchschranke mit 10^{-10} festgelegt. Die beiden Verfahren liefern die in den Tabellen 6.7 und 6.8 aufgezeigten Ergebnisse. Schaut man sich die Ergebnisse des Vergleichs von SHIRA und KRSCHUR an, dann wird deutlich, dass die Iterationszahlen und auch die Residuen der bestimmten Eigenwertnäherungen sehr ähnlich sind. Im Vergleich mit der *eigs* Funktion besticht der KRSCHUR Algorithmus durch geringere Iterationszahlen und in etwa gleiche Residuen.

Eigenwerte	Residuen
-0.90592878886122	6.945e-17
-0.90634686034999	1.325e-16
-1.07560224930036	3.933e-17
-1.60332758477172	1.639e-15
-1.65786577098970	6.311e-16
-1.66121735256372	2.157e-16

Tabelle 6.5: SHIRA Ergebnisse nach 3 Iterationen.

Eigenwerte	Residuen
-0.90592878885719	2.813e-16
-0.90634686034789	1.752e-16
-1.07560224930035	2.970e-16
-1.60332758476210	2.576e-15
-1.65786577098053	3.323e-15
-1.66121735256369	2.576e-15

Tabelle 6.6: KRSCHUR Ergebnisse nach 3 Iterationen und maximaler Konditionszahl 21178.

Eigenwerte	Residuen
-0.90592878886324	1.864e-16
-0.90634686035071	1.648e-16
-1.07560224930012	1.823e-16
-1.60332758476362	5.733e-17
-1.65786577099183	5.241e-15
-1.66121735256949	3.922e-15
-1.75605775408379	9.367e-14

Tabelle 6.7: Ergebnisse nach 11 Iteration mit *eigs*

Eigenwerte	Residuen
-0.90592878885661	3.244e-16
-0.90634686034754	2.242e-16
-1.07560224930008	1.591e-16
-1.60332758476411	7.370e-16
-1.65786577098821	2.488e-15
-1.66121735257659	1.169e-14
-1.75605775398702	2.057e-15

Tabelle 6.8: KRSCHUR Ergebnisse nach 8 Iterationen und maximaler Konditionszahl 2518.

Beispiel 6.2.2. Im folgenden Beispiel zum quadratischen Eigenwertproblem sind die Matrizen M, G und K von der Größe $N = 12828$. Es wird analog zum vorhergehenden Beispiel verfahren. Beim Vergleich von Shira und KRSCHUR werden wiederum 12 Eigenwerte berechnet, die Suchraumgröße betrage ebenfalls 24 und die Toleranz liegt bei 10^{-8}. Die in den Tabellen 6.9 und 6.10 dargestellten Ergebnisse sind durch die Berechnungen mit dem Shift $\sigma = 1$ erhalten worden. Auch für diese Größe soll

Eigenwerte	Residuen
-0.90510929898159	1.810e-16
-0.90529568786501	2.778e-16
-1.07480595544986	6.573e-17
-1.60117345104856	1.414e-15
-1.65765608689995	4.930e-15
-1.65914529725448	1.994e-15

Tabelle 6.9: SHIRA Ergebnisse nach 3 Iterationen.

Eigenwerte	Residuen
-0.90510929901264	7.174e-16
-0.90529568788285	7.145e-16
-1.07480595545160	7.563e-16
-1.60117345105306	1.149e-14
-1.65765608693113	9.270e-15
-1.65914529732631	2.094e-14

Tabelle 6.10: KRSCHUR Ergebnisse nach 2 Iterationen und maximaler Konditionszahl 59676.

ein Vergleich der beiden Verfahren *eigs* und KRSCHUR durchgeführt werden. Dabei werden 14 Eigenwerte gesucht in einem Suchraum der Dimension 22 mit der Fehlertoleranz 10^{-10}. Dargestellt sind die Ergebnisse in den Tabellen 6.11 und 6.12. Auch hier zeigt sich, dass der KRSCHUR Algorithmus dem SHIRA Verfahren ähnliche Ergebniss erzielt. Im Vergleich zur *eigs* Routine ist die Iterationszahl geringer, aber der Residuenfehler größer. Das Verwenden einer größeren Abbruchschranke empfiehlt sich hier, um eventuell den Residuenfehler zu verkleinern.

Eigenwerte	Residuen
-0.90510929898901	2.370e-16
-0.90529568786703	2.289e-16
-1.07480595545014	5.546e-16
-1.60117345102236	2.865e-16
-1.65765608688654	3.826e-16
-1.65914529726278	2.028e-16
-1.75187759476509	3.033e-14

Tabelle 6.11: Ergebnisse nach 12 Iteration mit *eigs*

Eigenwerte	Residuen
-0.90510929895565	4.337e-16
-0.90529568769521	5.042e-15
-1.07480595545042	5.909e-16
-1.60117345085658	6.147e-14
-1.65765606214944	5.009e-12
-1.65914519388901	3.001e-11
-1.75187759437868	1.322e-13

Tabelle 6.12: KRSCHUR Ergebnisse nach 9 Iterationen und maximaler Konditionszahl 29142.

6.2.2 Wärmeleitungsgleichung

Das folgende Beispiel stammt aus dem autonomen linear-quadratischen Optimalsteuerungsproblem der eindimensionalen Wärmeleitungsgleichung. Benutzt man den Standard Finite-Elemente Ansatz mit einer B-Spline Basis, dann kann eine endlichdimensionale Approximation durch das Lösen einer algebraischen Riccatigleichung erhalten werden. Eine genauere Diskussion des Beispiels kann in [7] nachgelesen werden. Die hier betrachtete Hamiltonische Matrix H hat die folgende Gestalt:

$$H = \begin{bmatrix} M_N^{-1}K_N & -BB^T \\ -C^TC & -\left(M_N^{-1}K_N\right)^T \end{bmatrix},$$

mit

$$M_N = \frac{1}{6N}\begin{bmatrix} 4 & 1 & & & \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & 4 \end{bmatrix}$$

und

$$K_N = -aN\begin{bmatrix} 2 & -1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix},$$

sowie $B = M_N^{-1}b_N$ und $C = c_N^T$. Die Vektoren b_N und c_N lassen sich aus der Integration der B-Spline Basisfunktionen erschließen. Für die praktische Anwendung ist das Bestimmen der Eigenwerte mit kleinem Betrag von großer Bedeutung. Um diese Eigenwerte zu erhalten, ist es notwendig die Shift-und-Invert Strategie mit dem Nullshift durchzuführen. Diese Aufgabe wirft das Problem des Rechnens mit der inversen Matrix H^{-1} auf, was bedeutet, dass eine Zerlegung für die Matrix H gefunden werden muss, mit deren Hilfe sich das Gleichungssystem $Hx = y$ lösen lässt. Ein Ansatz, der sich an dieser Stelle bietet, ist das Ausnutzen der Grundbeziehung für Hamiltonische Matrizen $(JH)^T = JH$. Aus dieser Gleichung folgt $H = -JH^TJ$ und es ist (H^TJ) eine symmetrische Matrix. Für diese Klasse von Matrizen empfiehlt es sich (vgl. [16]) eine LDL^T-Zerlegung zu benutzen. Dies benutzend ergibt sich $H^TJ = LDL^T$ und somit $H = -JH^TJ = -JLDL^T$. Diese Zerlegung kann beim Lösen des Gleichungssystems $Hx = y$ benutzt werden.

Beispiel 6.2.3. Die Matrix H ist für die folgenden Berechnungen von der Größe (2000×2000). Dabei werden die Verfahren *eigs* und KRSCHUR verglichen. Der Startvektor ist für beide Verfahren identisch, ebenso wie die Abbruchschranke von 10^{-10} der Algorithmen. Auch in diesem Fall werden, bei einer Suchraumgröße von 24, 12 Eigenwerte der Matrix H gesucht. Die Ergebnisse der beiden Routinen sind in den Tabellen (6.13) und (6.14) dargestellt. In diesem Beispiel sind die Ergebnisse von *eigs* und KRSCHUR im Hinblick auf Iterationszahl und die Größe der Residuen nahezu identisch.

Eigenwerte	Residuen
-0.09976767973694	1.287e-16
-0.39597717994449	1.646e-16
-0.88863485943190	5.616e-16
-1.57915744339631	4.771e-15
-2.46761444895309	1.802e-15
-3.55339069140684	6.253e-16

Tabelle 6.13: KRSCHUR Ergebnisse nach 3 Iterationen und maximaler Konditionszahl 74.

Eigenwerte	Residuen
-0.09976767967664	1.426e-16
-0.39597717993198	1.623e-16
-0.88863485906871	1.397e-16
-1.57915744337173	9.453e-14
-2.46761444935482	1.825e-16
-3.55339069160076	2.472e-16

Tabelle 6.14: *eigs* Ergebnisse nach 3 Iterationen

Zusammenfassung

Das Lösen von Eigenwertproblemen ist ein integraler Bestandteil der Numerischen Mathematik. Dabei werden die Anforderungen an die Qualität der numerischen Methoden immer höher, da immer größere Systeme gelöst werden müssen. Den Stand der Dinge repräsentiert im Bereich der Krylov-Unterraum-Verfahren das Arnoldi-Algorithmus mit impliziten Restarts. Da für viele Anwendungen die korrespondierenden Matrizen von einer bestimmten Struktur sind, ist es sinnvoll die numerischen Vorteile des impliziten Arnoldi-Verfahrens mit der Strukturerhaltung der Ausgangsmatrix zu koppeln. Dazu wurde von Benner und Faßbender ein symplektischer Lanczos-Algorithmus entwickelt, der analog zum Arnoldi-Verfahren arbeitet. Im Rahmen der Arbeit wurde der SR Algorithmus, der die Rolle des QR Algorithmus aus dem Arnoldi-Verfahren übernimmt, weiterentwickelt und in der Programmiersprache MATLAB umgesetzt. Der SR Algorithmus musste um die Bestimmung der Eigenvektoren mit positivem Realteil erweitert werden, der für die Hamiltonische Gestalt optimale Quadrupelshift musste entwickelt und implementiert werden und eine zeilenweise Form des ursprünglichen Algorithmus musste für die Bearbeitung des symplektischen Lanczos-Algorithmus theoretisch erschlossen und ebenso programmiertechnisch umgesetzt werden.
Nach Fertigstellung des SR Algrorithmus wurde in Analogie zu dem von Stewart im Jahre 2001 entwickelten Krylov-Schur Algorithmus ein Krylov-Schur-ähnliches Verfahren entwickelt, welches die bisher da gewesenen Schwierigkeiten bei der Deflation des Eigenwertproblems auf elegante Weise löst. Die dabei entstandenen Algorithmen zum Tauschen von Eigenblöcken wurden ebenfalls optimiert und in MATLAB-Routinen umgesetzt. Im Rahmen der theoretischen Vorarbeiten zum Krylov-Schur-ähnlichen Algrorithmus wurde die Äquivalenz zum bereits von Benner und Faßbender entwickelten Algorithmus bewiesen und die Transformationen zur Überführung der beiden verwandten Verfahren ineinander entwickelt und implementiert.
Nach Implementierung beider Algorithmen mussten zur Vergleichbarkeit mit anderen Softwarepaketen Shift-und-Invert-Strategien entwickelt und umgesetzt werden. Die Hamiltonische Struktur erfordert hier im Vergleich zu Nichthamiltonischen Matrizen ein spezielles Vorgehen, bei dem das Problem der Invertierung mehrerer Matrizen gelöst werden musste. Für das quadratische Eigenwertproblem wurden Algorithmen zur LU- und Cholesky-Zerlegung verwendet und implementiert. Zusätzlich wurden Shift-und-Invert-Strategien entwickelt, die Probleme des 'Positive-Real Balancing' bearbeiten.
Den Abschluss der Arbeit bilden die numerischen Ergebnisse der entwickelten und implementierten Algorithmen. Die Ergebnisse zum quadratischen Eigenwertproblem und zur Wärmeleitungsgleichung waren vielversprechend und zeigten sich im Bezug auf Iterationszahl und Genauigkeit der Ergebnissse mehr als konkurrenzfähig zu den vorhandenen Softwarepaketen.

Ausblick

Ein wesentlicher Punkt in der Zukunft sollte die Implementierung der vorgestellten Algorithmen in einer maschinennahen Programmiersprache – beispielsweise Fortran oder C – sein. Dieses ist notwendig, um Laufzeitanalysen im Vergleich mit anderen Algorithmen durchzuführen. Mit der Fortran Implementierung sollte das Erzeugen von Matlab mex Interfaces einhergehen, damit auch auf dieser Oberfläche ein leistungsfähiger strukturerhaltender Algorithmus für Hamiltonische Matrizen zur Verfügung steht. Im Rahmen der Einbettung in Matlab sollten Laufzeitvergleiche des SR Algorithmus' mit der *eig* Routine durchgeführt werden.

Anhang A

MATLAB Routinen

A.1 Grundlegende Routinen

A.1.1 perm

Aufruf

A=perm(A)

Input

A unpermutierte Matrix der Größe $(n \times m)$

Output

A permutierte Matrix der Größe $(n \times m)$

Beschreibung

Diese Funktion berechnet die 'perfect shuffle' Permutation der Matrix A, vorgestellt in der Einleitung dieser Arbeit.

A.1.2 unperm

Aufruf

A=unperm(A

Input

A permutierte Matrix der Größe $(n \times m)$

Output

A unpermutierte Matrix der Größe $(n \times m)$

Beschreibung

Diese Routine berechnet die Umkehrung zur 'perfect shuffle' Permutation.

A.1.3 housegen

Aufruf

v = housegen(a,n)

Input

a	Eingangsvektor der Dimension n
n	Dimension des Vektors a

Output

v	Householdervektor

Beschreibung

Im Verlaufe dieser Routine wird ein Vektor v berechnet für den gilt, dass mittels $H = I - vv^T$ der folgende Vektor erzeugt werden kann:

$$Ha = \alpha e_1 = \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

A.1.4 orthov

Aufruf

v_kplus1 = orthov(S_p,v_kplus1,kplus1)

Input

S_p	Symplektische Transformationsmatrix
v_kplus1	zu J-orthogonalisierender Vektor
kplus1	Index des Vektor v_{k+1}

Output

v_kplus1	zu S_p J-orthogonaler Vektor

Beschreibung

Diese Funktion führt eine J-Orthogonalisierung (vgl. Algorithmus 5.2). 5.2

A.1.5 orthow

Autor

Martin Stoll

Aufruf

w_k = orthow(S_p,w_k,k)

Input

S_p	Symplektische Transformationsmatrix
w_k	zu J-orthogonalisierender Vektor
k	Index des Vektor w_k

Output

w_k	zu S_p J-orthogonaler Vektor

Beschreibung

Diese Funktion führt eine J-Orthogonalisierung (vgl. Algorithmus 5.2).

A.2 SR Algorithmus

A.2.1 hsr

Aufruf

[H,ev,S,S2,cond_max,flag,iter] = hsr(H,S,max_cond,witheigvec,withschur,witheigvec2)

Input

H	Hamiltonische J-Hessenberg-Matrix der Größe $(2n \times 2n)$ $\{eye(2n)\}$
S	Symplektische Matrix der Größe $(2n \times 2n)$ $\{eye(2n)\}$
max_cond	maximale Konditionszahl $\left\{\frac{1}{\sqrt{eps}}\right\}$
witheigvec	Boolean für Eigenvektorenberechnung {1}
withschur	Boolean für die Ausgabe in Kylov-Schur-artigen Form {1}
witheigvec2	Boolean für die zweite Hälfte der Eigenvektoren {1}

Output

H	Hamiltonische Matrix $(2n \times 2n)$
ev	Vektor mit den berechneten Eigenwerten
S	Matrix mit berechneten Eigenvektoren
S2	Matrix mit der zweiten Hälfte Eigenvektoren
cond_max	maximal erreichte Konditionszahl
flag	Boolean für aufgetrene Fehler
iter	benötigte Iterationszahl

Beschreibung

In Verwendung der Resultate aus Kapitel 2 reduziert diese Funktion die Ausgangsmatrix auf eine Hamiltonische Matrix, deren Eigenwertprobleme sich auf die Größe (2×2) oder (4×4) beschränken. Zusätzlich zu den Eigenwerten der Matrix besteht die Möglichkeit Eigenvektoren zu erhalten, sowohl für Eigenwerte mit negativem Realteil, als auch für solche mit postivem Realteil.

A.2.2 implicit_sr_qs

Aufruf

[H_neu,S,err,cond_max] = implicit_sr_qs(H,S,shiftmatrix,n,witheigvec,cond_max,maxcond)

Input

H	Hamiltonische J-Hessenberg-Matrix der Größe $(2n \times 2n)$
S	Symplektische Matrix der Größe $(2n \times 2n)$
shiftmatrix	Matrix der Größe (4×4)
n	Größe der Ausgangsmatrix $2n$
witheigvec	Boolean für die Eigenwertberechnung
cond_max	maximal erreichte Konditionszahl
maxcond	maximale Konditionszahl

Output

H_neu	geshiftete Hamiltonische J-Hessenberg-Matrix der Größe $(2n \times 2n)$
S	Symplektische Matrix der Größe $(2n \times 2n)$ die verwendete Transformationen enthält
err	Boolean für aufgetretene Fehler
cond_max	maximal erreichte Konditionszahl

Beschreibung

Diese Funktion repräsentiert eine Umsetzung des Algorithmus' 2.3.

A.2.3 implicit_sr_qs3

Aufruf

[H_neu,S,err,cond_max] = implicit_sr_qs3(H,S,shiftmatrix,n,witheigvec,cond_max,maxcond)

Input

H	Hamiltonische J-Hessenberg-Matrix der Größe (6×6)
S	Symplektische Matrix der Größe $(2n \times 6)$
shiftmatrix	Matrix der Größe (4×4)
n	Größe der Ausgangsmatrix $2n$
witheigvec	Boolean für die Eigenwertberechnung
cond_max	maximal erreichte Konditionszahl
maxcond	maximale Konditionszahl

Output

H_neu	geshiftete Hamiltonische J-Hessenberg-Matrix der Größe (6×6)
S	Symplektische Matrix der Größe $(2n \times 6)$ die verwendete Transformationen enthält
err	Boolean für aufgetretene Fehler
cond_max	maximal erreichte Konditionszahl

Beschreibung

Dieses Unterprogramm entspricht einer Umsetzung des Algorithmus' 2.3 für Matrizen der Größe (6×6). Diese Matrizen sind die kleinsten noch zu shiftenden Matrizen im Verlaufe des SR Algorithmus'. Eine Bearbeitung mit der Routine A.2.2 würde viele Fallabfragen bedeuten, diese entfallen bei der Einführung einer eigenen Routine für den (6×6) Fall.

A.2.4 complev

Aufruf

[H,S_ev,S_schur]=complev(H,lambda);

Input

H	(2×2) Matrix
lambda	zur Matrix gehörender komplexer Eigenwert

Output

H	Transformierte $(2 \times 2$ Matrix
S_ev	Matrix mit Transformationen auf Eigenwertgestalt
S_schur	Matrix mit Transformationen auf Schurgestalt

Beschreibung

In Analogie zur Lapack Routine slanv2, (vgl. [2]) wird mit der vorgestellen Routine eine Matrix A auf Schurform H gebracht.

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$$

Auf der anderen Seite ist es möglich die Transformationen zu akkumulieren die notwendig sind um mittels des Schur-ähnlichen Verfahrens alle Eigenvektoren zu bestimmen.

A.2.5 evquasischur

Aufruf

[y1,y2]=evquasischur(H,S,lambda,p,q,isreal,n,indicator)

Input

H	Hamiltonische Matrix
S	Symplektische Matrix
lambda	Eigenwert, der auf der Diagonalen stehen soll
p	Startindex, des zu transformierenden Blockes
q	Endindex, des zu transformierenden Blockes
isreal	Boolean für reelle oder komplexe Eigenwerte
n	Größe der Matrix H
indicator	Vorzeichenindikator für die zwei verschiedenen Varianten

Output

y1	Erster Eigenvektor im komplexen Fall, im reellen Fall einziger Eigenvektor
y2	Zweiter Eigenvektor im komplexen Fall

Beschreibung

Diese Routine stellt die Eigenvektorberechnungen für die Schur-ähnliche Form dar, beschrieben im Teil 2.5 dieser Arbeit.

A.2.6 invlr

Aufruf

[lhs] = invlr(L,R,rhs,n)

Input

L	Untere Dreickesmatrix aus LR Zerlegung der ursprünglichen Matrix
R	Obere Dreiecksmatrix aus LR Zerlegung der ursprünglichen Matrix
rhs	Vektor, rechte Seite des Gleichungssystems
n	Größe der Matrizen $2n$

Output

lhs	Lösungsvektor des Gleichunssystems

Beschreibung

Anhand dieser Funktion wird für die Hamiltonische Ausgangsmatrix H mit deren LU Zerlegung $H = LU$, das Gleichunssystem $LUrhs = lhs$ gelöst.

A.2.7 hamlr

Aufruf

[L,R] = hamlr(H,n)

Input

H	Hamiltonische J-Hessenberg-Matrix der GRöße $(2n \times 2n)$
n	Größe der Matrix H $(2n \times 2n)$

Output

L	Untere Dreiecksmatrix aus LR Zerlegung von H
R	Untere Dreiecksmatrix aus LR Zerlegung von H

Beschreibung

Die vorgestellte Routine berechnet für die Hamiltonische J-Hessenberg-Matrix H eine LU Zerlegung, ohne Pivotisierung. Eine Diskussion des Verfahrens ist in 2.5 zu finden.

A.3 Symplektischer Lanczos-Prozess mit impliziten Restarts

A.3.1 irsl

Aufruf

[ev,S_p]=irsl(H,k);

Input

H	Hamiltonische J-Hessenberg-Matrix H
k	$2k$ ist die Anzahl der gesuchten Eigenwerte

Output

ev	Vektor, der die Eigenwertnäherungen enthält
S_p	Matrix, die akkumulierte Transformationen enthält

Beschreibung

Diese Routine entspricht der Umsetzung des symplektischen Lanczos-Algorithmus', wie er in 3.3 vorgestellt wird.

A.3.2 lanczosstart

Aufruf

[H_til_p,S_p,res_coeff,res_vec]=lanczosstart(H,k)

Input

H	Hamiltonische J-Hessenberg-Matrix
k	Parameter für Größe der zu erzeugenden Lanczos-Faktorisierung

Output

H_til_p	Hamiltonische J-Hessenbergmatrix der Größe $(2k \times 2k$
S_p	Transformationsmatrix
res_coeff	Residuenvektor
res_vec	Residuenkoeffizient

Beschreibung

Diese Funktion berechnet mit Hilfe des symplektischen Lanczos-Algorithmus' (vgl.3.1) eine Lanczos-Faktorisierung der Größe $(2k \times 2k)$.

A.3.3 lanczosblow

Aufruf

[H_til_p,S_p,res_coeff,res_vec] = lanczosblow(H,H_neu,S,n,start,m,residuum_coeff,residuum_vec)

Input

H	Hamiltonische J-Hessenberg-Matrix
H_neu	Hamiltonische J-Hessenberg-Matrix der Größe $(2k \times 2k)$
S	symplektische Transformationsmatrix
n	Größe der Ausgangsmatrix $2n$
start	Größenparameter der bereits vorhandenen Lanczos-Faktorisierung
m	Größenparameter der zu erzeugenden Lanczos-Faktorisierung
residuum_coeff	aktueller Residuenvektor
residuum_vec	aktueller Residuenkoeffizient

Output

H_til_p	Hamiltonische J-Hessenberg-Matrix der Größe $(2m \times 2m)$
S_p	symplektische Transformationsmatrix
res_coeff	neuer Residuenkoeffizient
res_vec	neuer Residuenvektor

Beschreibung

Die vorgestellte Routine vergrößert eine vorhandene Lanczos-Faktorisierung der Größe $(2k \times 2k)$ auf eine neue Faktorisierung der Größe $(2m \times 2m)$.

A.3.4 implicitsrds

Aufruf

[H_neu,S,err,condmax] = implicitsrds(H,S,shift1,shift2,n,witheigvec,condmax,maxcond)

Input

H	Hamiltonische J-Hessenberg-Matrix
S	Symplektische Transformationsmatrix
shift1	Erster Shift
shift2	Zweiter Shift
n	Größenparameter
witheigvec	Boolean für die Eigenwertberechnung
condmax	Bisher größte aufgetretene Konditionszahl
maxcond	Obere Schranke für die Konditionszahlen

Output

H_neu	Hamiltonische J-Hessenberg-Matrix
S	Symplektische Transformationsmatrix
err	Boolean für Fehlerauftreten
condmax	Bisher größte aufgetretene Konditionszahl

Beschreibung

Dieses Programm führt einen Doppelshift gemäß Kapitel 2.3.2 durch, dabei erfolgt keine effiziente Berechnung der Shiftgrößen, wie im genannten Kapitel angesprochen. Es werden explizit die Shiftgrößen verwendet.

A.3.5 implicitsrqs

Aufruf

[H_neu,S,err,cond_max] = implicitsrqs(H,S,shift1,shift2,n,witheigvec,cond_max,maxcond)

Input

H	Hamiltonische J-Hessenberg-Matrix
S	Symplektische Transformationsmatrix
shift1	Erster Shift
shift2	Zweiter Shift
n	Größenparameter
witheigvec	Boolean für die Eigenwertberechnung
condmax	Bisher größte aufgetretene Konditionszahl
maxcond	Obere Schranke für die Konditionszahlen

Output

H_neu	Hamiltonische J-Hessenberg-Matrix
S	Symplektische Transformationsmatrix
err	Boolean für Fehlerauftreten
condmax	Bisher größte aufgetretene Konditionszahl

Beschreibung

Mittel dieser Funktion wird ein Quadrupelshift gemäß Kapitel 2.3.3 durch. Auch hier wird nicht der effiziente Ansatz aus dem benannten Kapitel gewählt, sondern die explizite Variante mit den eingegebenen Shiftgrößen.

A.3.6 implicitsrqs3

Aufruf

[H_neu,S,err,cond_max] = implicitsrqs3(H,S,shift1,shift2,n,witheigvec,cond_max,maxcond)

Input

H	Hamiltonische J-Hessenberg-Matrix
S	Symplektische Transformationsmatrix
shift1	Erster Shift
shift2	Zweiter Shift
n	Größenparameter
witheigvec	Boolean für die Eigenwertberechnung
condmax	Bisher größte aufgetretene Konditionszahl
maxcond	Obere Schranke für die Konditionszahlen

Output

H_neu	Hamiltonische J-Hessenberg-Matrix
S	Symplektische Transformationsmatrix
err	Boolean für Fehlerauftreten
condmax	Bisher größte aufgetretene Konditionszahl

Beschreibung

Dies ist eine analoge Funktion zu der eben beschriebenen, siehe Anhang A.3.5. Allerdings arbeitet diese auf Hamiltonischen Matrizen der Größe (6×6).

A.4 Krylov-Schur-artiger Algorithmus

A.4.1 krschur

Aufruf

[ev,eigvec,S_p,Elambda] = krschur(H,k,sigma,n)

Input

H	Hamiltonische J-Hessenberg-Matrix oder function handle
k	Anzahl der gewünschten Eigenwerte $2k$
sigma	Shift für Shift-und-Invert Methode
n	Matrixgröße

Output

ev	$2k$ Ritzwerte
eigvec	Eigenvektoren zu den Ritzwerten $\mathbb{R}^{2k,2k}$
S_p	Transformationsmatrix
Elambda	Feld mit den erreichten Rückwärtsfehlern

Beschreibung

Diese Funktion berechnet $2k$ Ritzwerte zur Hamiltonischen Matrix H. Dabei handelt es sich um eine Umsetzung des in Kapitel 4 vorgestellten Algorithmus' 4.3.

A.4.2 lanczosstart_h

Aufruf

[H_til,S_p,res_coeff,res_vec]=lanczosstart_h(H,k,sigma,tol,ishandle,n)

Input

H	Hamiltonische J-Hessenberg-Matrix oder fucntion handle
k	Größe der zu erzeugenden Lanczos-Faktorisierung $2k$
sigma	Shiftgröße
tol	Abbruchparameter
ishandle	Boolean, ob H function handle oder explizite Matrix
n	Größenparameter

Output

H_til	Erzeugte Hamiltonischen J-Hessenberg-Matrix der Größe $(2k \times 2k)$
S_p	Symplektische Transformationsmatrix
res_coeff	Residuenkoeffizient
res_vec	Residuenvektor

Beschreibung

Diese Funktion berechnet mit Hilfe des symplektischen Lanczos-Algorithmus' (vgl.3.1) eine Lanczos-Faktorisierung der Größe $(2k \times 2k)$.

A.4.3 lanczosblow_h

Aufruf

[H_til,S_p,res_coeff,res_vec] =
lanczosblow_h(H,H_neu,S,n,start,m,residuum_coeff,residuum_vec,sigma,tol,ishandle)

Input

H	Hamiltonische J-Hessenberg-Matrix oder function handle
H_neu	Bereits erzeugt Hamiltonische J-Hessenberg-Matrix
S	Symplektische Transformationen
n	Größenparameter
start	Größenparameter der bereits erzeugten Lanczos-Faktorisierung
m	Endparameter der zu erzeugenden Lanczos-Faktorisierung
residuum_coeff	Aktueller Residuenkoeffizient
residuum_vec	Aktueller Residuenvektor
sigma	Shiftgröße
tol	Abbruchparameter
ishandle	Boolean, ob H function handle oder explizite Matrix

Output

H_til	Erzeugte Hamiltonischen J-Hessenberg-Matrix der Größe $(2m \times 2m)$
S_p	Symplektische Transformationsmatrix
res_coeff	Residuenkoeffizient
res_vec	Residuenvektor

Beschreibung

Diese Funktion vergrößert mit Hilfe des symplektischen Lanczos-Algorithmus' (vgl.3.1) eine Lanczos-Faktorisierung der Größe $(2k \times 2k)$ auf eine der Größe $(2m \times 2m)$.

A.4.4 orthQ

Aufruf

Q = orthQ(y,k);

Input

y	Vektor in $\mathbb{R}^{2k}$
k	Größenparameter für den Vektor y

Output

Q	Orthogonale symplektische Matrix

Beschreibung

Diese Funktion entspricht einer Umsetzung des Algorithmus' 4.1, es wird eine symplektische orthogonale Matrix Q bestimmt, so dass $y^T Q = \alpha e_{2k}^T$.

A.4.5 upswap

Aufruf

[H,S_p] = upswap(H,S_p,chindex,n,tol)

Input

H	Hamiltonische Matrix in blockdiagonaler Gestalt
S_p	Symplektische Transformationsmatrix
chindex	Feld, dass die Indizes der zu tauschenden Blöcke enthält
n	Größenparameter der Ausgangsmatrix
tol	Abbruchparameter

Output

H	Hamiltonische Matrix in blockdiagonaler Gestalt mit getauschten Eigenblöcken
S_p	Symplektische Transformationsmatrix

Beschreibung

Diese Funktion realisiert das Tauschen von Eigenblöcken, vorgestellt im Abschnitt 4.2. Dabei werden die durch chindex definierten Blöcke in der Matrix in die linke obere Ecke getauscht.

A.4.6 downswap

Aufruf

[H,S_p] = downswap(H,S_p,chindex,n,tol)

Input

H	Hamiltonische Matrix in blockdiagonaler Gestalt
S_p	Symplektische Transformationsmatrix
chindex	Feld, dass die Indizes der zu tauschenden Blöcke enthält
n	Größenparameter der Ausgangsmatrix
tol	Abbruchparameter

Output

H	Hamiltonische Matrix in blockdiagonaler Gestalt mit getauschten Eigenblöcken
S_p	Symplektische Transformationsmatrix

Beschreibung

Diese Funktion realisiert das Tauschen von Eigenblöcken, vorgestellt im Abschnitt 4.2. Es werden die in chindex benannten Blöcke in die rechte untere Ecke der Matrix verschoben.

A.4.7 invhamquadr

Aufruf

x = invhamquadr(y)

Input

y	Vektor

Output

x	Vektor

Beschreibung

Die Routine berechnet in Abhängigkeit vom globalen Parameter σ die Lösung x des Gleichungssystems

$$Hx = y,$$

$$(H - \sigma I)(H + \sigma I)Hx = y$$

oder

$$(H - \sigma I)(H + \sigma I)(H - \bar{\sigma} I)(H + \bar{\sigma} I)Hx = y.$$

A.4.8 jhessrow

Aufruf

[S,H,condmax] = jhessrow(M,isHamilton)

Input

M	Matrix
isHamilton	Boolean der angibt, ob die Matrix H Hamiltonisch ist

Output

S	Symplektische Transformationen
H	Matrix in J-Hessenberg Form oder im Fall einer Hamiltonischen Ausgangsmatrix in Ha
condmax	Maximale Konditionszahl der verwendeten Transformationen

Beschreibung

Diese Funktion berechnet nach dem Algorithmus 4.2 eine zeilenweise Variante des JHESS Algorithmus'.

A.5 Externe Routinen

A.5.1 gauss1

Aufruf

[c,d,condmax,err] = gauss1(a,b,condmax,badcond)

Input

a	Parameter 1
b	Parameter 2
condmax	maximal erreichte Konditionszahl
badcond	maximale Konditionszahl

Output

c	Ausgabeparameter 1
d	Ausgabeparameter 2
condmax	maximal erreichte Konditionszahl
err	Boolean für Fehler in der Parametererzeugung

Beschreibung

Diese Routine erzeugt die beiden Parameter c und d so, dass gilt

$$\begin{bmatrix} c & & & d \\ & c & d & \\ & & c^{-1} & \\ & & & c^{-1} \end{bmatrix} \begin{bmatrix} \otimes \\ a \\ b \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ r \\ 0 \end{bmatrix}$$

A.5.2 givens

Aufruf

[c,s] = givens(a,b)

Input

a	Parameter 1
b	Parameter 2

Output

c	Ausgabeparameter 1
s	Ausgabeparameter 2

Beschreibung

Diese Routine bestimmt die Parameter c und s so, dass gilt $c^2 + s^2 = 1$ und

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \end{bmatrix}$$

A.5.3 sinvert

Aufruf

M_inv = sinvert(M)

Input

M	symplektische Matrix der Dimension $(2n \times 2n)$

Output

M_inv Inverse der symplektischen Matrix M

Beschreibung

Mit Hilfe dieser Routine wird effzient die Inverse einer symplektischen Matrix erzeugt.

Literaturverzeichnis

[1] J. Abels and P. Benner. CAREX – a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0). SLICOT Working Note 1999-14, November 1999.

[2] 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 users' guide, 1999.

[3] Th. Apel, V. Mehrmann, and D. Watkins. Structured eigenvalue methods for the computation of corner singularities in 3D anisotropic elastic structures. *Computer Methods in Applied Mechanics and Engineering*, 191:4459–4473, 2002.

[4] Z. Bai and J.W. Demmel. On swapping diagonal blocks in real Schur form. *Linear Algebra Appl.*, 186:73–95, 1993.

[5] P. Benner and H. Faßbender. An implicitly restarted symplectic Lanczos method for the Hamiltonian eigenvalue problem. *Linear Algebra and its Applications*, 263:75–111, 1997.

[6] P. Benner and H. Faßbender. An implicitly restarted symplectic Lanczos method for the symplectic eigenvalue Problem. *SIAM J. Matrix Anal.Appl.*, 22:682–713, 2000.

[7] P. Benner, A.J. Laub, and V. Mehrmann. A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: Continuous-time case. Technical Report SPC 95_22, Fakultät für Mathematik, TU Chemnitz–Zwickau, 1995.

[8] A. Bunse-Gerstner and V. Mehrmann. A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation. *IEE Trans. Automat. Control*, 12:1104–1113, 1986.

[9] T.A. Davis. Umfpack version 4.4 user guide. Technical report, Dept. of Computer and Information Science and Engineering Univ. of Florida, Gainesville, FL, 2005.

[10] J.W. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.

[11] J.W. Demmel, J.R. Gilbert, and X.S. Li. SuperLU Users' Guide. Technical report, Lawrence Berkeley National Laboratory, 2003.

[12] L. Elsner. On some algebraic problems in connection with the general eigenvalue algorithms. *Linear Algebra Appl.*, 26:123–138, 1979.

[13] H. Fassbender. Error analysis of the symplectic Lanczos method for the symplectic eigenvalue problem. *Bit Numerical Mathematics*, 40(3):471–496, 2000.

[14] W. R. Ferng, W. W. Lin, and C. S. Wang. The shift-inverted J-Lanczos algorithm for the numerical solutions of large sparse algebraic Riccati equations. *Comp. Math. Appl.*, 33(10):23?40, 1997.

[15] R. Freund. Transpose-free quasi-minimal residual methods for non-Hermitian linear systems. *Numerical analysis manuscript 92-97*, 14:470 – 482, 1992. AT&T Bell Labs.

[16] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins UniversityPress, 1996.

[17] E.J. Grimme, D.C. Sorensen, and P.V. Dooren. Model reduction of state space systems via an implcitly restarted Lanczos method. Technical report, Department of Computational and Applied Mathematics, Rice University, Houston,Texas, USA, 1994.

[18] S. Gugercin and A.C. Antoulas. A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, 77(8):748–766, 2004.

[19] D. Irony, G. Shklarski, and S. Toledo. Parallel and fully recursive multifrontal supernodal sparse Cholesky. *Future Generation Computer Systems*, 20(3):425–440, 2004.

[20] W. Kahan, B.N. Parlett, and E. Jiang. Resiudal bounds on approximate eigensystems of normal matrices. *SIAM J. Numer.Anal.*, 19:470–484, 1982.

[21] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J.Res.Nat.Bur.Standards*, 45:255–282, 1950.

[22] R. Lehoucg and D.C. Sorensen. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM J. Matrix Anal. Appl*, 17(4):789–821, 1996.

[23] V. Mehrmann and D. Watkins. Structure-preserving methods for computing eigenpairs of large sparse skew-Hamiltonian/Hamiltonian pencils. *SIAM Journal on Scientific Computing*, 22:1905–1925, 2001.

[24] O.Schenk and K. Gärtner. *PARDISO User Guide Version 1.2.3*. Computer Science Department, University of Basel, Switzerland, 1.2.3 edition, 2005.

[25] C. Pester. Cocos, computation of corner singularities. Technical report, TU Chemnitz, 2005.

[26] J. Rosam. Berechnung der Rissgeometrie bei spröden elastischen Körpern, 2004. Diplomarbeit.

[27] A. Salam. On theoretical and numerical aspects of symplectic Gram-Schmidt-like algorithms. *Numerical Algorithms*, 39:437–462, 2005.

[28] V. Sima. *Algorithms for Linear-Quadratic Optimization*. Marcel Dekker, inc, New York, 1996.

[29] D. Sorensen. Deflation for implicitly restarted Arnoldi methods. Technical report, CAAM at Rice University, 1998.

[30] D. Sorensen. Numerical methods for large eigenvalue problems. *Acta Numerica*, pages 519–584, 2002.

[31] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl*, 13:357–385, 1992.

[32] G.W. Stewart. A Krylov-Schur algorithm for large eigenproblems. *SIAM J. on Matrix Analysis and Applications*, 23(4):601–614, 2001.

[33] G.W. Stewart. *Matrix Algorithms, Volume II: Eigensystems*. SIAM, Philadelphia, 2001.

[34] J. Stoer. *Numerische Mathematik 1*. Springer Verlag, 2002.

[35] N. Wong, V. Balakrishnan, and C.-K. Koh. A fast Newton/Smith algorithm for solving algebraic Riccati equations and its application in model order reduction. In *Proc. Design Automation Conference*, San Diego, CA, June 2004.

Thesen

1. Diese Arbeit beschäftigt sich mit einem symplektischen Verfahren zum Lösen von Hamiltonischen Eigenwertproblemen. Dabei beruht dieser Algorithmus auf dem Durchführen von impliziten Restarts.

2. Im ersten Teil der Arbeit wurde der von Mehrmann/Bunse-Gerstner entwickelte SR Algorithmus untersucht, implementiert und erweitert. Dazu konnte ein Verfahren zur Durchführung eines impliziten Quadrupelshifts entwickelt und ebenfalls implementiert werden. Desweiteren wurde der SR Algorithmus um die Bestimmung der Eigenvektoren zu den Eigenwerten mit positivem Realteil ergänzt. Dafür konnte ein Verfahren basierend auf der Inversen Iteration mit einer speziellen LU-Zerlegung für Hamiltonische Matrizen entwickelt werden, ebenso wie ein Krylov-Schur-ähnlicher Algorithmus.

3. Im weiteren Verlauf wurde der von Benner und Faßbender eingeführte symplektische Lanczos-Algorithmus untersucht und implementiert. Dazu zählte das Finden und Beweisen von Konvergenzkriterien für das Verfahren mit impliziten Restarts. Die Abbruchschranken konnten in den Programmcode zum Hamiltonischen Lanczos-Prozess integriert und getestet werden.

4. Es wurde in Analogie zum Verfahren von Stewart eine Krylov-Schur-artige Zerlegung entwickelt, dazu gehörte der Beweis der Äquivalenz dieser Zerlegung zu der ursprünglichen Lanczos-Faktorisierung. Im Rahmen dieses Beweises wurde ein spaltenweiser JHESS-Algorithmus entwickelt und programmtechnisch umgesetzt.

5. Mit Hilfe der Krylov-Schur-artigen Zerlegung war es möglich, *Purging* und *Locking* umzusetzen. Dazu mussten Transformationen gefunden werden, die das Tauschen von Eigenblöcken realisieren und gleichfalls symplektisch sind.

6. Für den Krylov-Schur-artigen Algorithmus konnten Abbruchkriterien aufgestellt und bewiesen werden, die auf der Basis des Rückwärtsfehlers arbeiten.

7. Die Integration von Shift-und-Invert Strategien in den Programmcode wurde vorgenommen. Dabei lagen die Schwerpunkte auf dem Lösen des quadratischen Eigenwertproblems. Auch wurden theoretischen Untersuchungen für das "Positive-Real-Balancing"durchgeführt.

8. Für die Fälle des Doppel- und Quadrupelshifts wurden Analogiebeweise zum Zusammenbruch der SR Zerlegung durchgeführt. Das Analysieren der Stabilität war mit dem Implementieren einer J-Reorthogonalisierung verbunden, dabei konnte der vorhandene Algorithmus Gram-Schmidt-ähnliche Verfahren zu einer modifizierten Variante ausgebaut werden.

9. Die Numerischen Ergebnisse zeigten im Vergleich mit dem SHIRA Algorithmus und der Matlab *eigs* Routine die Konkurrenzfähigkeit des entwickelten Algorithmus' im Bezug auf Genauigkeit der Ergebnisse und Iterationszahl der Verfahren.

Erklärung

Ich erkläre an Eides Statt, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Chemnitz, den 30. April 2009