Zur Numerik großdimensionaler Eigenwertprobleme

Werner Vogt
Technische Universität Ilmenau
Institut für Mathematik
Postfach 100565
98684 Ilmenau

Ilmenau, den 8.11.2004

Zusammenfassung Der Beitrag stellt ausgewählte iterative Verfahren zur numerischen Approximation von Eigenwerten und -vektoren großdimensionaler reeller Matrizen vor. Neben der Vektoriteration und der inversen Iteration für das partielle Eigenwertproblem wird der QR-Algorithmus zur Lösung des vollständigen Eigenwertproblems behandelt. Für sehr große Matrizen ist man an einer Lokalisierung des Spektrums interessiert, wofür Ritz-Werte und Petrov-Werte ermittelt und an Beispielen demonstriert werden.

MSC 2000: 65F15, 65F50, 65F10

Keywords: Matrix eigenvalues, QR method, Krylov subspace methods, Ritz values, Petrov values

1 Einführung

Eigenwerte und Eigenvektoren reeller oder komplexer Matrizen müssen in verschiedensten praktischen Anwendungen berechnet werden, z.B. bei

- mechanischen Knickproblemen (Säulen und Tragwerke, eingespannte belastete Stäbe)
- kritischen Drehzahlen rotierender Wellen
- Schwingungen an Industrierobotern
- Eigenschwingungen elektrischer Schwingkreise (Hochfrequenztechnik, Sicherheit von Großtransformatoren)
- Stabilitätsuntersuchungen dynamischer Systeme (neuronale Netze, Chaosübergang)
- Wellenmechanik und Quantenmechanik.

Die Diskretisierung der Modellgleichungen führt in der Regel auf Eigenwertprobleme für reelle $n \times n$-Matrizen A. Gesucht sind nichttriviale Lösungen $x \in \mathbb{C}^n,\ x \neq 0$, des Gleichungssystems

$$Ax \;=\; \lambda x\,, \quad \lambda \in \mathbb{C}\,, \tag{1}$$

wobei $A \in \mathbb{R}^{n\times n}$ gegeben ist und $x \in \mathbb{C}^n,\ \lambda \in \mathbb{C}$ unbekannt sind. Der Wert λ heißt *Eigenwert* und x ist ein zugehöriger *Eigenvektor*.[1]

Typisch für die angeführten Probleme ist meist eine hohe Matrixdimension $n > 1000$, zu der oft die schwache Besetztheit der Matrix A und deren fehlende Symmetrie kommt. Allein deshalb sind Verfahren der Linearen Algebra (vgl. dazu [15]), die zuerst das charakteristische Polynom von A und anschließend dessen Nullstellen $\lambda_1, \lambda_2, \ldots, \lambda_n$ bestimmen, nicht anwendbar. Anders als bei der Lösung linearer Gleichungssysteme existieren im Allgemeinen keine wirklich *direkten Verfahren*, die die Eigenwerte mit einer endlichen Anzahl von Gleitpunktoperationen bis auf Rundungsfehler genau berechnen können. Alle Ansätze führen auf *indirekte, iterative Verfahren*, indem sie Eigenwerte und Eigenvektoren als Grenzwerte unendlicher Folgen ermitteln. Je nach Bedarf

- liefert ein derartiges Verfahren alle Eigenwerte $\lambda_1, \lambda_2, \ldots, \lambda_n$ und löst so das *vollständige Eigenwertproblem* oder es
- approximiert einen oder mehrere exponierte Eigenwerte, z.B. am Rande des Spektrums, was man als *partielles Eigenwertproblem* bezeichnet.

[1] Das Paar (λ, x) nennt man auch Eigenpaar.

Zunächst bestimmen wir einen extremalen Eigenwert λ_1, insbesondere den betragsgrößten oder betragskleinsten Wert, während anschließend das vollständige Eigenwertproblem behandelt wird. Bei großdimensionalen Matrizen wird man mit einer Approximation von m ausgewählten Eigenwerten für ein $m \ll n$ zufrieden sein, wofür ein moderner Ansatz vorgestellt wird. In der Darstellung beschränken wir uns auf reelle Matrizen, für die keine Symmetrie vorausgesetzt wird. Verallgemeinerungen der vorgestellten Verfahren findet der interessierte Leser u.a. in [33, 34, 4, 9].

2 Vektoriteration und inverse Iteration

2.1 Einfache und verbesserte Vektoriteration

Die Grundform des Verfahrens der Vektoriteration geht auf R. von Mises (1929) zurück und ist zur Approximation des betragsgrößten Eigenwertes geeignet. Wir setzen voraus, dass $A \in \mathbb{R}^{n\times n}$ diagonalisierbar ist und die Eigenwerte in der Anordnung

$$|\lambda_1| \; > \; |\lambda_2| \; \geq \; |\lambda_3| \; \geq \; \ldots \; \geq \; |\lambda_n| \tag{2}$$

indiziert sind. Der einfache Eigenwert $\lambda_1 \in \mathbb{R}$ ist der *dominante Eigenwert* und die zugehörigen Eigenvektoren $x_1, x_2, \ldots, x_n$ bilden eine Basis des $\mathbb{C}^n$. Damit existiert ein Startvektor $y_0 \in \mathbb{R}^n$, der eine nichtverschwindende Koordinate a_1 in Richtung von $x_1 \in \mathbb{R}^n$ besitzt und sich damit in dieser Basis als

$$y_0 = a_1x_1 + a_2x_2 + ... + a_nx_n \quad \text{mit} \quad a_i \in \mathbb{C},\; a_1 \neq 0 \tag{3}$$

darstellen lässt. Bilden wir nun durch sukzessive Multiplikation mit A die Vektorfolge

$$\begin{aligned}
y_1 &= Ay_0 \\
y_2 &= Ay_1 \;=\; A^2y_0 \\
y_3 &= Ay_2 \;=\; A^3y_0 \\
... &= ..... \\
y_k &= Ay_{k-1} \;=\; A^ky_0\,,
\end{aligned}$$

so liefert (3) und Einsetzen der Eigenvektorbeziehung $Ax_i = \lambda_i x_i$ die Darstellung

$$y_k \;=\; A^ky_0 \;=\; a_1\lambda_1^kx_1 + a_2\lambda_2^kx_2 + \cdots + a_n\lambda_n^kx_n\,. \tag{4}$$

Für einen festen Index j können wir mit der Abkürzung $c_i := a_ix_{i,j}$ die j-te Komponente von y_k durch

$$y_{k,j} \;=\; a_1\lambda_1^kx_{1,j} + a_2\lambda_2^kx_{2,j} + \cdots + a_n\lambda_n^kx_{n,j} \;=\; c_1\lambda_1^k + c_2\lambda_2^k + \cdots + c_n\lambda_n^k$$

darstellen. Division zweier aufeinander folgender Iterierter ergibt dann

$$\begin{aligned}
\frac{y_{k+1,j}}{y_{k,j}} &= \frac{c_1\lambda_1^{k+1} + c_2\lambda_2^{k+1} + ... + c_n\lambda_n^{k+1}}{c_1\lambda_1^k + c_2\lambda_2^k + ... + c_n\lambda_n^k} \\
&= \lambda_1\frac{1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^{k+1} + ... + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^{k+1}}{1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^{k} + ... + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^{k}}\,,
\end{aligned} \tag{5}$$

womit wir wegen Voraussetzung (2) den Grenzwert

$$\lim_{k\to\infty} \frac{y_{k+1,j}}{y_{k,j}} \;=\; \lambda_1 \tag{6}$$

erhalten. Wegen der Darstellung $y_k = A^k y_0$ des k-ten Iterationsvektors hat sich auch die Bezeichnung *Potenzmethode, Matrixpotenzierung (engl. power method)* für das Verfahren eingebürgert. Aus Aufwandsgründen sollte man jedoch die Berechnung von A^k tunlichst vermeiden und stattdessen die vorgestellte *einfache Vektoriteration* ausführen.

Beispiel 1 Zur Matrix

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad \text{mit} \quad \lambda_1 = 3,\ \lambda_2 = 1,\ \lambda_3 = 0 \ \text{ und Startvektor } \ y_0 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

bestimmen wir in Tabelle 1 die Iterationsvektoren y_k und die Quotienten der einzelnen Komponenten sowie der Vektornormen $\|y_k\|_1$ und $\|y_k\|_2$. Offenbar konvergieren alle Quotienten

k	A	y_k	y_{k+1}	$\frac{y_{k+1,j}}{y_{k,j}}$	$\frac{\|y_{k+1}\|_1}{\|y_k\|_1}$	$\frac{\|y_{k+1}\|_2}{\|y_k\|_2}$
0	1 1 0 1 2 1 0 1 1	1 0 0	1 1 0	1 - -	2.0	1.414
1	1 1 0 1 2 1 0 1 1	1 1 0	2 3 1	2 3 -	3.0	2.646
2	1 1 0 1 2 1 0 1 1	2 3 1	5 9 4	2.5 3.0 4.0	3.0	2.952
3	1 1 0 1 2 1 0 1 1	5 9 4	14 27 13	2.8 3.0 3.25	3.0	2.9946
4	1 1 0 1 2 1 0 1 1	14 27 13	41 81 40	2.9286 3.0 3.0769	3.0	2.9994
5	1 1 0 1 2 1 0 1 1	41 81 40	122 243 121	2.9756 3.0 3.025	3.0	2.99993

Tabelle 1: Einfache Vektoriteration (Potenzmethode)

gegen $\lambda_1 = 3$, wobei die Iterationsvektoren zwar normmäßig wachsen, sich jedoch wegen

$$y_4 \approx 14 \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \quad y_5 \approx 40 \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}, \quad y_6 \approx 121 \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

rasch gegen den zugehörigen Eigenvektor $x_1 = (1, 2, 1)^T$ „drehen“. □

Um die Größenordnung der Komponenten $y_{k,j}$ zu beschränken, ist eine Normierung aller Vektoren y_k angebracht, womit sich eine *verbesserte Vektoriteration* ergibt:

$$z_k := y_k/\|y_k\|_2 , \qquad y_{k+1} := Az_k , \qquad k = 0, 1, 2, \ldots \tag{7}$$

Satz 2 *$A \in \mathbb{R}^{n\times n}$ sei diagonalisierbar und die Eigenwerte genügen der Anordnung (2). Zu jedem Startvektor y_0 gemäß (3) mit $a_1 \neq 0$ konvergiert die verbesserte Vektoriteration gegen*

$$|\lambda_1| = \lim_{k\to\infty} \|y_k\|_2 \qquad \textit{und} \qquad x_1 = \lim_{k\to\infty} z_k .$$

Beweis: Nach Rückwärtseinsetzen in (7) lässt sich y_k mittels

$$y_k = Az_{k-1} = \frac{1}{\|y_{k-1}\|_2} Ay_{k-1} = \cdots = \frac{1}{\alpha_{k-1}} A^k y_0 \qquad \text{mit} \qquad \alpha_k = \prod_{i=0}^{k} \|y_i\|_2$$

umformen. Benutzen wir die Darstellung (4) für $A^k y_0$, so gewinnen wir damit

$$|\lambda_1| = \lim_{k\to\infty} \frac{\|A^{k+1}y_0\|_2}{\|A^k y_0\|_2} = \lim_{k\to\infty} \frac{\alpha_k \|y_{k+1}\|_2}{\alpha_{k-1}\|y_k\|_2} = \lim_{k\to\infty} \frac{\alpha_{k+1}}{\alpha_k} = \lim_{k\to\infty} \|y_k\|_2 .$$

Wegen $y_{k+1} = Az_k$ erhalten wir $\|y_{k+1}\|_2 z_{k+1} = Az_k$, woraus für $k \to \infty$ die Eigenwertbeziehung $|\lambda_1|z = Az$ mit dem normierten Eigenvektor $z := \lim_{k\to\infty} z_k$ folgt. Dieser ist bis auf das Vorzeichen eindeutig festgelegt. □

Mit der berechneten Eigenvektor-Näherung z_k lässt sich übrigens leicht eine weitere Verbesserung der Eigenwertnäherung ableiten. Multiplizieren wir dazu die Eigenwertgleichung $Ax_1 = \lambda_1 x_1$ von links mit x_1^T und stellen nach λ_1 um, so erhalten wir den so genannten *Rayleigh-Quotienten*

$$\lambda_1 = \frac{x_1^T A x_1}{x_1^T x_1} . \tag{8}$$

Nutzen wir die erhaltene Näherung $z_k \approx x_1$ und die Beziehung $y_{k+1} := Az_k$ aus, so liefert Einsetzen in den Rayleigh-Quotienten die gegenüber $\|y_k\|_2$ verbesserte Approximation (vgl. [20])

$$\lambda_1 \approx \frac{(Az_k)^T z_k}{(z_k)^T z^k} = \frac{(y_{k+1})^T z_k}{(z_k)^T z_k}$$

einschließlich des richtigen Vorzeichens. Algorithmus 3 berechnet λ_1 mittels des Rayleigh-Quotienten.

Beispiel 4 1. Die symmetrische positiv definite Pascal-Matrix der Dimension n kann man in Matlab mittels `A = pascal(n)` generieren. Sie ist von links oben her als Pascalsches Dreieck aufgebaut, d.h. $a_{ik} = a_{i,k-1} + a_{i-1,k}$. Für $n = 5$ lautet sie

```
A =   1     1     1     1     1
      1     2     3     4     5
      1     3     6    10    15
      1     4    10    20    35
      1     5    15    35    70
```

Algorithmus 3 (Verbesserte Vektoriteration)
Function $[\lambda, z] = \text{power}(A, tol, y0)$

1. Setze $\lambda := 0\,,\ z := y0$ und wähle $kmax$
2. For $k = 0, 1, 2, \ldots, kmax$ do
   - 2.1. Speichere $\lambda_{alt} := \lambda$ um
   - 2.2. Berechne $y := Az$ und $\lambda := \|y\|_2$
   - 2.3. Falls $|\lambda - \lambda_{alt}| < tol \cdot \lambda$, so gehe zu Schritt 3
   - 2.4. Normiere $z := y/\lambda$
3. Return $\lambda := (y^T z)/(z^T z)$ und Eigenvektor z

Mit dem Startvektor `y0=rand(n,1)` und Toleranz 10^{-8} liefert das Verfahren schon nach 4 Iterationen die Näherung $\tilde{\lambda_1} = 92.29043482631919$ für λ_1 mit einem Fehler $\tilde{\lambda_1} - \lambda_1 = -3.8339 \cdot 10^{-9}$. Auch der extrem große Spektralradius der 200-reihigen Pascal-Matrix `A=pascal(200)` wird vom Verfahren bereits nach 4 Iterationen zu $\tilde{\lambda_1} = 3.439394178530210 \cdot 10^{118}$ mit Maschinengenauigkeit approximiert. Ursache dafür ist der kleine Konvergenzfaktor der Matrix

$$\varkappa := \max_{i=2(1)n} |\lambda_i/\lambda_1| = 1.117 \cdot 10^{-3}\,.$$

2. Wesentlich schlechter konvergiert das Verfahren hingegen bei der symmetrischen $n \times n$-Poisson-Matrix A (vgl. [15]) mit $m \times m$-Einheitsmatrix I ($m = \sqrt{n}$) und den Matrizen

$$A = \begin{pmatrix} T & -I & & \cdots & O & O \\ -I & T & -I & \cdots & O & O \\ O & -I & T & \cdots & O & O \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ O & O & O & \cdots & T & -I \\ O & O & O & \cdots & -I & T \end{pmatrix}, \quad T = \begin{pmatrix} 4 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 4 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 4 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 4 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 4 \end{pmatrix}.$$

Mit zufällig gewähltem Startvektor $y0$ und Toleranz 10^{-8} ergeben sich die Näherungen in Tabelle 2. Im letzten Fall wurde wegen des großen Konvergenzfaktors $\varkappa$ auch mit 1000 Iterationen die gewünschte Genauigkeit nicht erreicht. □

n	$\tilde{\lambda_1}$	$\tilde{\lambda_1} - \lambda_1$	k	$\varkappa$
16	7.236067797575539	$-1.799 \cdot 10^{-7}$	50	0.8618
100	7.837970678645802	$-1.215 \cdot 10^{-6}$	189	0.9698
900	7.979443586681945	$-3.370 \cdot 10^{-5}$	1000	0.9961

Tabelle 2: Verbesserte Vektoriteration für Poisson-Matrix

2.2 Inverse Iteration

Besteht die Aufgabe darin, den betragskleinsten Eigenwert λ_n einer regulären Matrix zu approximieren, so gehen wir zu ihrer inversen Matrix A^{-1} über. Mit der Annahme

$$0 \;<\; |\lambda_n| \;<\; |\lambda_{n-1}| \;\leq\; |\lambda_{n-2}| \;\leq\; \ldots \;\leq\; |\lambda_1| \tag{9}$$

erfüllen deren Eigenwerte $\beta_i = 1/\lambda_i$ die Voraussetzung (2). Die Vektoriteration $y_{k+1} := A^{-1}z_k$ mit der aufwändigen Matrixinversion vermeiden wir durch Umstellung und Lösung des linearen Systems $Ay_{k+1} = z_k$ mittels einer LU-Zerlegung von A. Die so erhaltene *inverse Vektoriteration*

$$z_k \;=\; \frac{y_k}{\|y_k\|_2}\,, \qquad LRy_{k+1} \;=\; z_k\,, \qquad k = 0, 1, 2, \ldots \tag{10}$$

verlangt zwar einmalig $\frac{2}{3}n^3$ arithmetische Operationen für die LU-Zerlegung von A; allerdings bedeuten die $2n^2$ Operationen für eine Vorwärts- und Rückwärts-Substitution keinen höheren Aufwand als für die verbesserte Vektoriteration 3 im k-ten Schritt. Aus der Darstellung

Algorithmus 5 (Inverse Iteration)
Function $[\lambda, z] =$ inverse_iteration(A, tol, λ_0, y_0)

1. Setze $\beta := 0\,,\; z := y_0$ und wähle $kmax$
2. LU-Zerlegung von $B := A - \lambda_0 I \;=\; LR$
3. For $k = 0, 1, 2, \ldots, kmax$ do
   - 3.1. Speichere $\beta_{alt} := \beta$ um
   - 3.2. Vorwärtselimination $Lw = z$ liefert w
   - 3.3. Rückwärtselimination $Ry = w$ liefert y
   - 3.4. Berechne $\beta := \|y\|_2$
   - 3.5. Falls $|\beta - \beta_{alt}| < tol \cdot \beta$, so gehe zu Schritt 4
   - 3.6. Normiere $z := y/\beta$
4. Berechne Rayleigh-Quotient $\beta := (z^T z)/(y^T z)$
5. Return $\lambda := \lambda_0 + \beta$ und Eigenvektor z

(5) erkennen wir unschwer, dass die Konvergenz der einfachen Vektoriteration durch den Quotienten $|\lambda_1/\lambda_2| < 1$ und damit die Geschwindigkeit der inversen Iteration durch den Konvergenzfaktor $|\lambda_n/\lambda_{n-1}| < 1$ bestimmt wird. Mit einer *Spektralverschiebung* wollen wir deshalb erreichen, dass $|\lambda_n| \ll |\lambda_{n-1}|$ wird und benutzen dazu folgendes leicht beweisbare

Lemma 6 *Das Spektrum von A sei $\sigma(A) = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. Zu festem $\lambda_0 \in \mathbb{C}$ besitzt die Matrix $B = A - \lambda_0 I$ mit der Einheitsmatrix I das Spektrum $\sigma(B) = \{\lambda_1 - \lambda_0, \lambda_2 - \lambda_0, \ldots, \lambda_n - \lambda_0\}$ und dieselben Eigenvektoren wie A.*

Sei nun λ_n ein beliebiger gesuchter Eigenwert von A, für den wir eine gute Anfangsnäherung λ_0 kennen. Wir führen eine Spektralverschiebung mittels $B = A - \lambda_0 I$ durch und bestimmen den betragskleinsten Eigenwert β_n von B mittels inverser Iteration. Der gesuchte Eigenwert von A ergibt sich zu $\lambda_n = \lambda_0 + \beta_n$. Mit geeigneter Wahl von λ_0 lässt sich nun der Konvergenzfaktor

$$q := \frac{|\lambda_n - \lambda_0|}{\min_{j \neq n} |\lambda_j - \lambda_0|}$$

beliebig klein machen. Dieses auf H. Wielandt (1944) zurückgehende Verfahren 5 benötigt neben dem Startvektor y_0 auch eine gute Eigenwertnäherung λ_0.

Bemerkung 7 In der Praxis treten oft parameterabhängige Probleme $A(\mu)x = \lambda x$ mit $\mu \in \mathbb{R}$ auf, z.B. bei der Stabilitätsanalyse dynamischer Systeme. Das für einen Parameterwert μ_0 berechnete Eigenpaar (λ_0, z_0) kann oft als gute Startnäherung für die Matrix $A(\mu_1)$ mit benachbartem Parameterwert $\mu_1 = \mu_0 + \Delta\mu$ genutzt werden.

Beispiel 8 Die 16-reihige Poisson-Matrix A aus Beispiel 4 besitzt neben dem betragsgrößten Eigenwert $\lambda_1 = 7.236067977...$ auch mehrfache Eigenwerte λ_i der Vielfachheiten m_i. Tabelle 3 zeigt, dass die inverse Iteration auch den 4-fachen Eigenwert $\lambda_7 = 4$ mit vorgegebener Toleranz 10^{-8} gut approximiert, falls eine geeignete Startnäherung λ_0 vorliegt.

λ_0	$\tilde{\lambda}_i$	$\tilde{\lambda}_i - \lambda_i$	m_i	k
2.0	1.763932022309071	$-1.911 \cdot 10^{-10}$	2	9
6.0	6.236067978058331	$+5.585 \cdot 10^{-10}$	2	9
4.4	3.999999990091857	$-9.908 \cdot 10^{-9}$	4	21

Tabelle 3: Inverse Iteration für Poisson-Matrix

3 QR-Zerlegung und QR-Verfahren

Um sämtliche Eigenwerte von A zu bestimmen und damit das vollständige Eigenwertproblem $Ax = \lambda x$ zu lösen, ist eine Ähnlichkeitstransformation in die *Jordan-Form* (vgl. [15]) wünschenswert. Deren numerische Berechnung ist allerdings hochgradig instabil, denn kleine Störungen der Koeffizienten können zu Jordan-Formen mit stark abweichenden Eigenwerten und Eigenvektoren führen. Deshalb haben sich *orthogonale Ähnlichkeitstransformationen* bewährt. Sie überführen die gegebene Matrix $A \in \mathbb{R}^{n \times n}$ mit Hilfe einer orthogonalen Matrix $Q \in \mathbb{R}^{n \times n}$ (im komplexwertigen Fall mit einer unitären Matrix) in eine obere Quasi-Dreiecksmatrix

$$R = Q^T A Q \,, \tag{11}$$

deren Eigenwerte λ_i, $i = 1(1)n$, und Eigenvektoren z_i leicht bestimmt werden können. Die λ_i sind als Invarianten der Ähnlichkeitstransformation zugleich die gesuchten Eigenwerte von A. Die Eigenvektoren von A ergeben sich zu $x_i = Qz_i$. Orthogonale Matrizen Q erfordern wegen $Q^T = Q^{-1}$ keine Matrixinversion, weshalb sie bereits von C. Jacobi (1846) erfolgreich

verwendet wurden. Das QR-Verfahren als leistungsfähigstes Verfahren dieser Klasse wurde in Arbeiten von H. RUTISHAUSER (1958), J.F.G. FRANCIS (1961) und V.N. KUBLANOVSKAJA (1961) entwickelt und stellt gegenwärtig mit den der schnellen Transformationen von W. GIVENS und A. HOUSEHOLDER das Standardverfahren für das vollständige Eigenwertproblem dar. Ziel des Verfahrens ist die Überführung von A in die obere *Quasi-Dreiecksgestalt* (auch als *reelle Schur-Form* bezeichnet)

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} & \cdots & R_{1m} \\ 0 & R_{22} & R_{23} & \cdots & R_{2m} \\ 0 & 0 & R_{33} & \cdots & R_{3m} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & R_{mm} \end{pmatrix} \tag{12}$$

mit den ein- oder zweireihigen quadratischen Matrizen R_{ii}, $i = 1(1)m$. Denn dann bilden die einreihigen Diagonalblöcke R_{ii} die reellen Eigenwerte, während sich aus den zweireihigen R_{ii} auch die Paare konjugiert komplexer Eigenwerte von R ergeben. Eine solche Darstellung kann stets garantiert werden:

Satz 9 (I. Schur) *Zu jeder Matrix $A \in \mathbb{R}^{n\times n}$ existiert eine orthogonale Matrix $Q \in \mathbb{R}^{n\times n}$, so dass $R = Q^T AQ$ Quasi-Dreiecksform (12) besitzt.*

Den umfangreichen Beweis findet man in [28], S. 265f. Bevor der iterative Aufbau der Matrizen R und Q behandelt wird, soll die besondere Eignung orthogonaler Matrizen aus numerischer Sicht herausgestellt werden. Da zum Aufbau von Transformationsmatrizen zahlreiche Matrixoperationen nötig sind, treten ständig Rundungsfehler in Erscheinung. Matrixmultiplikationen QA verschlechtern dabei im Allgemeinen die Kondition von A. Ist jedoch Q orthogonal, so tritt dieser Fall nicht ein.

Satz 10 *Für Matrizen $A \in \mathbb{R}^{n\times n}$ und orthogonale Matrizen $Q \in \mathbb{R}^{n\times n}$ gilt:*
(i) $\|QA\|_2 = \|A\|_2$ *und* $\|QA\|_F = \|a\|_F$

(ii) $cond_2(QA) = cond_2(A)$, $cond_F(QA) = cond_F(A)$, *falls A regulär ist.*

BEWEIS: Wegen $(QA)^T(QA) = A^TQ^TQA = A^TA$ haben diese Matrizen dieselben Eigenwerte und damit dieselbe Spektralnorm. Bei regulärem A erhält man dieselbe Aussage für $(QA)^{-1}$ und A^{-1}. Also stimmen auch die Spektralkonditionen $cond_2 = \|A\|_2\|A^{-1}\|_2$ überein. Mit der Spaltendarstellung $A = (a_1, a_2, \ldots, a_n)$ lautet die Frobeniusnorm

$$\|A\|_F^2 = \sum_{i=1}^{n}\sum_{k=1}^{n} a_{ik} = \sum_{i=1}^{n} \|a_i\|_2^2$$

unter Benutzung der Euklidischen Norm. Damit notiert man die Frobenius-Norm für QA

$$\begin{aligned} \|QA\|_F^2 &= \sum_{i=1}^{n} \|Qa_i\|_2^2 = \sum_{i=1}^{n} (Qa_i)^T(Qa_i) \\ &= \sum_{i=1}^{n} a_i^TQ^TQa_i = \sum_{i=1}^{n} \|a_i\|_2^2 = \|A\|_F^2 ; \end{aligned}$$

analoges gilt für die Inversen $(QA)^{-1}$ und A^{-1}. □

3.1 QR-Zerlegung

Die Faktorisierung $A = QR$ einer im Allgemeinen rechteckigen Matrix $A \in \mathbb{R}^{m\times n}$ mit $m \geq n$ in eine orthogonale quadratische Matrix $Q \in \mathbb{R}^{m\times m}$ und eine Matrix $R \in \mathbb{R}^{m\times n}$, die unterhalb der Hauptdiagonalen nur Nullen besitzt, heißt (volle) *QR-Zerlegung*. Zur Lösung des Eigenwertproblems benötigen wir nur den Spezialfall quadratischer $n \times n$-Matrizen mit der oberen Dreiecksmatrix R.

3.1.1 Householder-Reflektor

Ähnlich wie bei der LU-Zerlegung transformieren wir die Ausgangsmatrix A so, dass sie nach n Schritten in die obere Dreiecksmatrix $R = (r_{ik})$ übergeht. Multiplikation der im k-ten Schritt vorliegenden Matrix A_k mit einer geeigneten orthogonalen Matrix Q_k soll die neue Matrix

$$A_{k+1} \;=\; Q_k A_k \;=\; \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1k} & r_{1,k+1} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2k} & r_{2,k+1} & \cdots & r_{2n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & r_{kk} & r_{k,k+1} & \cdots & r_{kn} \\ 0 & 0 & \cdots & 0 & a^{k+1}_{k+1,k+1} & \cdots & a^{k+1}_{k+1,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & a^{k+1}_{n,k+1} & \cdots & a^{k+1}_{nn} \end{pmatrix} \tag{13}$$

liefern, die bereits k Nullspalten unterhalb der Hauptdiagonalen aufweist. Die Matrix Q_k hat offenbar die Aufgabe, die Subdiagonalelemente der k-ten Spalte zu Null zu machen. Damit die $k-1$ ersten Nullspalten erhalten bleiben, sollen die Matrizen den allgemeinen Aufbau

$$Q_k \;=\; \begin{pmatrix} I & 0 \\ 0 & H \end{pmatrix} \qquad \text{mit der Einheitsmatrix} \quad I \in \mathbb{R}^{(k-1)\times(k-1)}$$

und einer orthogonalen Matrix $H \in \mathbb{R}^{(n-k+1)\times(n-k+1)}$ besitzen. A. Householder (1958) gelang es, derartige Spiegelungsmatrizen H zu konstruieren, die mit einer einzigen Multiplikation eine komplette Subdiagonalspalte von A_k in den Nullvektor transformieren. Sei $x \in \mathbb{R}^{n-k+1}$ die k-te Restspalte in A_k, so soll sie der *Householder-Reflektor* H in die k-te Spalte von A_{k+1} mit $\|Hx\| = \|x\|$ transformieren

$$x \;=\; \begin{pmatrix} a^k_{k,k} \\ a^k_{k+1,k} \\ a^k_{k+2,k} \\ \cdots \\ a^k_{n,k} \end{pmatrix} \qquad \Rightarrow \qquad Hx \;=\; \begin{pmatrix} \|x\| \\ 0 \\ 0 \\ \cdots \\ 0 \end{pmatrix} \;=\; \|x\| e_1 \,. \tag{14}$$

In Abb. 1 ist der Vektor $u = Hx - x = \|x\|e_1 - x \in \mathbb{R}^2$ dargestellt sowie die Hyperebene E^+ durch den Nullpunkt mit Normalenvektor u. Bestimmen wir H so, dass zu beliebigem Punkt y das Bild Hy durch Spiegelung an E^+ erzeugt wird, so überführt H speziell den Vektor x in den multiplizierten Einheitsvektor $Hx = \|x\|e_1$, d.h. alle Komponenten von Hx mit Ausnahme der ersten Komponente verschwinden. Man überlegt sich leicht, dass die Matrix

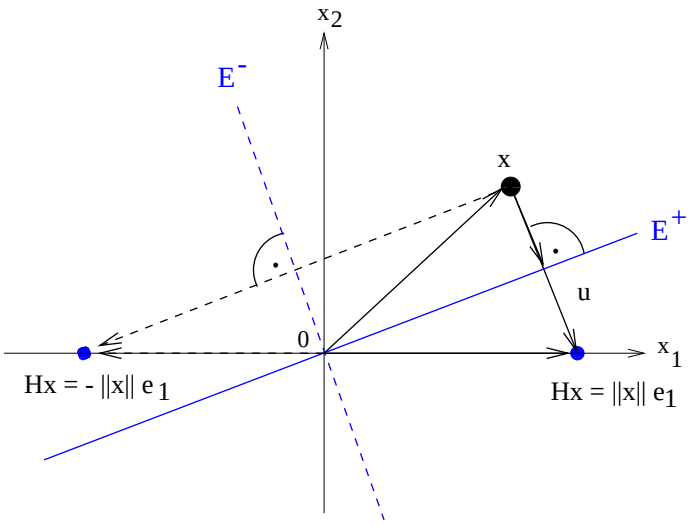

Abbildung 1: Householder-Spiegelung mittels H

$$H \;=\; I - 2\frac{uu^T}{u^Tu} \;=\; I - 2\left(\frac{u}{\|u\|}\right)\left(\frac{u}{\|u\|}\right)^T \tag{15}$$

genau diese Eigenschaft besitzt. Der Householder-Reflektor H ist zudem symmetrisch und orthogonal

$$H^T \;=\; H\,, \qquad H^TH \;=\; H^2 \;=\; I\,.$$

Diese beiden Eigenschaften übertragen sich auch auf Q_k. Anwendung von H auf einen Vektor $y \in \mathbb{R}^{m-k+1}$ ergibt die Transformationsformel

$$Hy \;=\; \left(I - 2\frac{uu^T}{u^Tu}\right)y \;=\; y - 2v(v^Ty) \quad \text{mit} \quad v := \frac{u}{\|u\|}\,. \tag{16}$$

Wie Abb. 1 zeigt, ist auch eine Spiegelung mittels der Hyperebene E^- möglich, indem $u = -\|x\|e_1 - x$ gebildet wird. Um eine Subtraktion der 1. Vektorkomponenten und damit einhergehende Stellenauslöschungen zu vermeiden, wählen wir u nach der Regel

$$u \;:=\; sign(x_1)\|x\|_2 \cdot e_1 + x \quad \text{mit} \quad sign(x_1) = 1\,,\ \text{falls } x_1 = 0$$

womit $\|u\|$ nie kleiner als $\|x\|$ wird.

3.1.2 Der Algorithmus

Führen wir n Spiegelungen gemäß $A_{k+1} = Q_kA_k$ aus, so erhalten wir die obere Dreiecksmatrix

$$R \;=\; Q_nQ_{n-1}\cdots Q_2Q_1A \;=\; Q^TA \quad \text{mit} \quad Q \;:=\; Q_1Q_2\cdots Q_n \tag{17}$$

d.h. die gewünschte Zerlegung $A = QR$ mit der orthogonalen Matrix Q. Um die Matrix Q explizit zu ermitteln, kann z.B. $Q^T = Q_nQ_{n-1}\cdots Q_2Q_1I$ mit der Einheitsmatrix $I \in \mathbb{R}^{n\times n}$ gemäß Transformationsformel (17) parallel berechnet werden, womit die Speicherung der Vektoren v vermieden wird. Alternativen dazu findet man in [27].

Algorithmus 11 (QR-Zerlegung)
Function $[Q, R] = \text{QRdecomposition}(A, n)$

1. Setze $Q := I$ (Einheitsmatrix)
2. For $k = 1, 2, \ldots, n$ do
   1. Speichere $x := A(k : n, k)$ um
   2. Berechne $v := sign(x_1)\|x\|_2 \cdot e_1 + x$
   3. Falls $v \neq 0$, so
      3.1. Normiere $v := v/\|v\|_2$
      3.2. $A(k : n, k : n) := A(k : n, k : n) - 2v[v^T A(k : n, k : n)]$
      3.3. $Q(k : n, k : n) := Q(k : n, k : n) - 2v[v^T Q(k : n, k : n)]$
4. Return $R := A$ und $Q := Q^T$

Zur effizienten Darstellung des Algorithmus 11 nutzen wir die in MATLAB übliche Notation für Teilmatrizen: Mit `A(i:n,k:n)` wird die Teilmatrix von A mit oberem linken Element a_{ik} und unterem rechten Element a_{nn} bezeichnet. Für den Spaltenvektor, beginnend mit a_{ik}, notieren wir dann abkürzend `A(i:n,k)`. Algorithmus 11 berücksichtigt auch den Fall eines verschwindenden Vektors x. Nur dann wird $\|v\|_2 = 0$ und damit $r_{kk} = 0$ in der Matrix R, womit die Spiegelung unterbleiben kann. Damit ist bereits die Durchführbarkeit des Algorithmus bei beliebigen Matrizen garantiert:

Satz 12 *Für jede Matrix $A \in \mathbb{R}^{n\times n}$ existiert die QR-Zerlegung $A = QR$.*

Bemerkung 13 1. Die QR-Zerlegung kann auch mittels Rotationsmatrizen durch die *schnelle Givens-Transformation* erreicht werden (vgl. [28]). Jedoch zeichnet sich Algorithmus 11 durch beste Performance und hohe numerische Stabilität aus.

2. Die QR-Zerlegung kann wegen des Satzes 10 mit Vorteil zur stabilen Lösung schlecht konditionierter linearer Gleichungssysteme $Ax = a$ angewendet werden. Allerdings verdoppelt sich die Rechenzeit gegenüber der LU-Zerlegung wegen des größeren arithmetischen Aufwandes $T(n) \approx \frac{4}{3}n^3$.

3. MATLAB gestattet die QR-Zerlegung beliebiger reeller und komplexer $m \times n$-Matrizen A. Die Funktion

```
[Q,R] = qr(A)
```

liefert eine unitäre (im reellen Fall eine orthogonale) $m \times m$-Matrix Q und eine obere Dreiecksmatrix R der Dimension $m \times n$ mit $A = Q * R$. Weitere Aufrufvarianten existieren, besonders für sparse Matrizen.

Beispiel 14 Für die 4×4-Matrix

$$A = \begin{pmatrix} 20 & -7 & 3 & -2 \\ -7 & 5 & 1 & 4 \\ 3 & 1 & 3 & 1 \\ -2 & 4 & 1 & 2 \end{pmatrix}$$

liefert das Kommando `[Q,R] = qr(A)` die QR-Zerlegung $A = QR$ mit

```
Q =
   -0.9305   -0.1734    0.0943   -0.3086
    0.3257   -0.4975   -0.2263   -0.7715
   -0.1396   -0.4747   -0.7354    0.4629
    0.0930   -0.7050    0.6317    0.3086

R =
  -21.4942    8.3744   -2.7915    3.2102
         0   -4.5684   -3.1470   -3.5279
         0         0   -1.5180   -0.5657
         0         0         0   -1.3887
```

Wegen $\|Q^TQ - I\|_2 = 3.9157 \cdot 10^{-16}$ und $\|A - QR\|_2 = 1.5646 \cdot 10^{-15}$ können die Ergebnisse mittels `format long e` in hoher Genauigkeit dargestellt werden. □

3.2 QR-Verfahren

3.2.1 Der Basisalgorithmus

Wie bereits bemerkt wurde, ist im Gegensatz zu linearen Gleichungssystemen das Matrixeigenwertproblem im Allgemeinen nicht durch endlich viele arithmetische Operationen lösbar. Deshalb kann die gesuchte orthogonale Matrix Q in (11) nur durch einen iterativen Prozess ermittelt werden. Hat man also eine transformierte Matrix A_k aus A ermittelt, so wird man eine *elementare orthogonale Ähnlichkeitstransformation* mit der orthogonalen Matrix Q_k durchführen, womit die Folge

$$A_{k+1} := Q_k^T A_k Q_k\,, \quad k = 1, 2, \ldots \quad \text{mit} \quad A_1 := A \tag{18}$$

ähnlicher Matrizen entsteht. Rückwärtseinsetzen liefert dann die Darstellung

$$A_{k+1} = Q_k^T Q_{k-1}^T \cdots Q_1^T A_1 Q_1 \cdots Q_{k-1} Q_k = (Q_1 \cdots Q_{k-1} Q_k)^T A (Q_1 \cdots Q_{k-1} Q_k)\,.$$

Die Matrizen Q_k bestimmen wir so, dass eine Quasi-Dreiecksform $R = Q^TAQ$ entsteht, also die Grenzwerte

$$\lim_{k\to\infty} A_{k+1} = R\,, \qquad \lim_{k\to\infty} Q_1 Q_2 \cdots Q_{k-1} Q_k = Q$$

angenähert werden. Dies bewerkstelligen wir mit der eingeführten QR-Zerlegung $A_k = Q_k R_k$ und anschließender Multiplikation der beiden Matrizen in umgekehrter Reihenfolge $A_{k+1} :=$

Algorithmus 15 (QR-Algorithmus)
Function $[Q, R] = \text{QRbasis}(A, tol)$

1. Setze $U := I$ (Einheitsmatrix)
2. For $k = 1, 2, \ldots$ do
   - 2.1. Zerlege $A = QR$ nach Algorithmus 11
   - 2.2. Berechne $A := RQ$
   - 2.3. Datiere auf: $U := UQ$
   - 2.4. Falls in A gilt: $|a_{i+1,i} \cdot a_{i+2,i+1}| < tol\ ,\ \ i = 1(1)n-2\ ,$ so gehe zu Schritt 3
3. Return $R := A$ und $Q := U$

R_kQ_k . Denn wegen $R_k = Q_k^{-1}A_k = Q_k^TA_k$ ergibt sich daraus die als *QR-Transformation* bezeichnete orthogonale Ähnlichkeitstransformation

$$A_{k+1} \;=\; R_kQ_k \;=\; Q_k^TA_kQ_k\ . \tag{19}$$

Iterative Wiederholung der zwei Schritte liefert die Grundform 15 des QR-Algorithmus. Schritt 2.3 kann entfallen, wenn nur die Eigenwerte benötigt werden. In Schritt 2.4 wird geprüft, ob A im Rahmen der Genauigkeit *tol* bereits Quasi-Dreiecksgestalt besitzt.

Beispiel 16
1. Wir bestimmen die Eigenwerte der symmetrischen Matrix aus Beispiel 14

$$A \;=\; \begin{pmatrix} 20 & -7 & 3 & -2 \\ -7 & 5 & 1 & 4 \\ 3 & 1 & 3 & 1 \\ -2 & 4 & 1 & 2 \end{pmatrix}$$

mit dem Aufruf `[Q,R] = qrbasis(A,1e-8)` und erhalten nach 12 Iterationen die Quasi-Dreiecksgestalt

```
R =
   23.5274   -0.0000    0.0000    0.0000
   -0.0000    6.4605    0.0000    0.0000
    0.0000    0.0000    0.6345   -0.9833
   -0.0000    0.0000   -0.9833   -0.6224
```

Die beiden Eigenwert-Näherungen liegen damit (in format long) mit $\tilde{\lambda_1} = 23.52738620165210$ und $\tilde{\lambda_2} = 6.46051471995713$ vor, während λ_3 und λ_4 aus der 2×2- Matrix

$$B \;=\; \begin{pmatrix} 0.6345 & -0.9833 \\ -0.9833 & -0.6224 \end{pmatrix} \quad \text{zu} \qquad \begin{aligned} \tilde{\lambda_3} \;&=\; 1.17304887031690 \\ \tilde{\lambda_4} \;&=\; -1.16094979192615 \end{aligned}$$

leicht ermittelt werden können. Der Fehler dieser Approximationen beträgt

$$\|\tilde{\lambda} - \lambda\|_\infty \;=\; \max_{i=1(1)4} |\tilde{\lambda}_i - \lambda_i| \;=\; 7.105 \cdot 10^{-14}\,.$$

2. Die unsymmetrische 6×6- Matrix

```
A =
     7     3     4   -11    -9    -2
    -6     4    -5     7     1    12
    -1    -9     2     2     9     1
    -8     0    -1     5     0     8
    -4     3    -5     7     2    10
     6     1     4   -11    -7    -1
```

besitzt das Spektrum $\sigma(A) = \{5+6i, 5-6i, 1+2i, 1-2i, 3, 4\}$ mit 4 konjugiert komplexen Eigenwerten. Das Verfahren `[Q,R] = qrbasis(A,1e-8)` bricht nach 50 Iterationen mit der oberen Schur-Form (in format short)

```
R =
    5.0000   -6.0000   -6.1588  -11.0353   -8.9058    5.8406
    6.0000    5.0000    3.8819   15.7233   14.1821   10.4615
   -0.0000   -0.0000    4.0000    3.0000   12.6647   -1.2671
   -0.0000   -0.0000   -0.0000    3.0000   -0.2809   11.3983
    0.0000    0.0000    0.0000    0.0000    1.0000   -2.0000
    0.0000    0.0000    0.0000   -0.0000    2.0000    1.0000
```

ab. Die komplexwertigen Eigenwertnäherungen erhält man aus den 2×2-Matrizen

$$B_{12} \;=\; \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} \quad \text{und} \qquad B_{56} \;=\; \begin{pmatrix} r_{55} & r_{56} \\ r_{65} & r_{66} \end{pmatrix}$$

zusammen mit den reellen Werten zu

```
lambda =
  5.00000000000005 + 6.00000000000012i
  5.00000000000005 - 6.00000000000012i
  1.00000134476773 + 1.99999841773378i
  1.00000134476773 - 1.99999841773378i
  4.00000045213392
  2.99999685833052
```

Die moderate Genauigkeit $\|\tilde{\lambda} - \lambda\|_\infty \;=\; 3.141 \cdot 10^{-6}$ wird anders als beim ersten Beispiel nun mit hohem Aufwand erreicht. □

Die Konvergenztheorie des Verfahrens ist insbesondere im Falle mehrfacher und/oder komplexer Eigenwerte kompliziert. Sind die Eigenwerte jedoch reell und verschieden, so gilt (vgl. [34]) der folgende

Satz 17 (Konvergenz des QR-Algorithmus) *$A \in \mathbb{R}^{n\times n}$ habe Eigenwerte mit $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. Dann konvergieren die Matrizen $A_k = (a_{ik}^{(k)})$ gegen eine obere Dreiecksmatrix R mit*

$$\lim_{k\to\infty} a_{ii}^{(k)} \;=\; \lambda_i\,, \quad i = 1(1)n\,.$$

Für die Subdiagonalelemente von A_k gilt asymptotisch bei $k \to \infty$

$$|a_{i-1,i}^{(k)}| \;=\; O\left(\left|\frac{\lambda_i}{\lambda_{i-1}}\right|^k\right) \;, \quad i = 2(1)n \;. \tag{20}$$

Bemerkung 18

1. Ist A zusätzlich symmetrisch, so sind wegen (19) offenbar alle A_k symmetrisch und konvergieren gegen die Diagonalmatrix $R = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

2. Hat A komplexe Eigenwertpaare, deren Beträge allerdings verschieden von den Beträgen der übrigen Eigenwerte sind, so lässt sich unter weiteren Voraussetzungen zeigen, dass die Matrizen A_k gegen eine Quasi-Dreiecksmatrix R konvergieren.

3. Wegen (20) konvergiert das Verfahren *linear* und besonders langsam, falls $|\lambda_i/\lambda_{i-1}|$ nahe 1 liegt. Für ein konjugiert komplexes Paar λ_{i-1}, λ_i konvergieren die Werte $a_{i-1,i}^{(k)}$ nicht gegen Null.

Beispiel 19 1. Mit `A = GALLERY('CHOW',n)'` gewinnen wir obere Hessenberg-Matrizen der Form

$$A \;=\; \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \;,$$

für die der Fehler $error := \max_{i=1(1)n-2} |a_{i+1,i} \cdot a_{i+2,i+1}|$ des QR-Basisalgorithmus eine gewünschte Entwicklung gemäß Abb. 2 zeigt.

2. Für die Poisson-Matrix aus Beispiel 4 mit reellen, aber mehrfachen Eigenwerten versagt der QR-Basisalgorithmus jedoch bereits bei moderaten Dimensionen, wie dies in Abb. 3 erkennbar wird. □

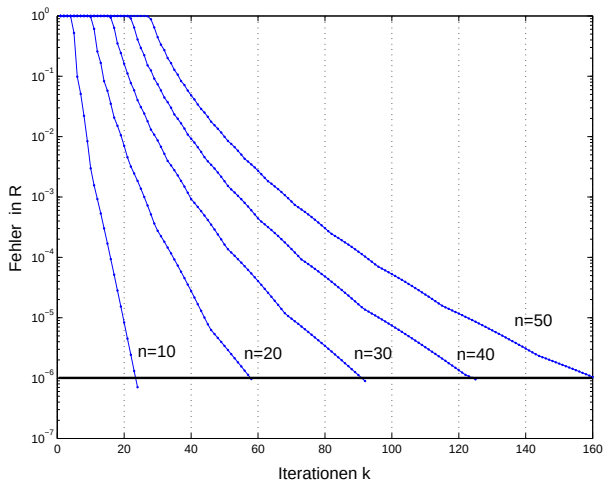

Abbildung 2: Konvergenz des QR-Algorithmus für Hessenberg-Matrix

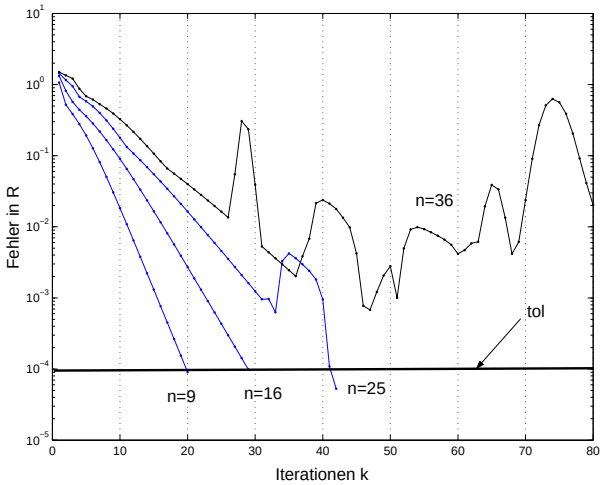

Abbildung 3: Konvergenz des QR-Algorithmus für Poisson-Matrix

3.2.2 Numerische Realisierung

Der Rechenaufwand des Algorithmus 15 mit $\mathcal{O}(n^3)$ Operationen pro Schritt kann beträchtlich reduziert werden, wenn wir die Ausgangsmatrix mittels einer orthogonalen Ähnlichkeitstransformation zuerst in die *obere Hessenbergform*

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1,n-1} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2,n-1} & a_{2n} \\ 0 & a_{23} & a_{33} & \cdots & a_{3,n-1} & a_{3n} \\ 0 & 0 & a_{43} & \cdots & a_{4,n-1} & a_{4n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & a_{n,n-1} & a_{nn} \end{pmatrix} \tag{21}$$

transformieren. Speziell für symmetrische Matrizen A erhalten wir wegen $A_1 := Q^T A Q$ damit eine symmetrische *Tridiagonalmatrix*. Der QR-Algorithmus lässt die Struktur dieser Matrizen invariant, d.h. hat A obere Hessenbergform oder Tridiagonalform, so besitzen alle erzeugten Matrizen A_k des Algorithmus 15 ebenfalls diese Eigenschaft. Für eine QR-Zerlegung einer Hessenbergmatrix brauchen wir jedoch nur $\mathcal{O}(n^2)$ arithmetische Operationen, während voll besetzte Matrizen $\mathcal{O}(n^3)$ Operationen benötigen.

Wie bei der Vektoriteration lässt sich die Konvergenz des Verfahrens durch eine *Spektralverschiebung (engl. shift)* der Matrizen A_k um σ_k weiter verbessern. Betrachten wir den Fall reeller Eigenwerte λ_i, die den Voraussetzungen des Satzes 17 genügen. Zuerst führen wir die QR-Zerlegung einer geeignet verschobenen Matrix $B_k = A_k - \sigma_k I$ mit Einheitsmatrix I durch und „korrigieren“ anschließend die Verschiebung bei der auszuführenden Multiplikation

$$A_{k+1} \;=\; R_k Q_k + \sigma_k I \;=\; Q_k^T (A_k - \sigma_k I) Q_k + \sigma_k I \;=\; Q_k^T A_k Q_k \,, \tag{22}$$

so dass wiederum eine Folge orthogonal-ähnlicher Matrizen A_k entsteht. Für σ_k ist wegen (20) eine möglichst gute Eigenwertnäherung zu verwenden, z.B. die so genannte *Rayleigh-Quotienten-Verschiebung*

$$\sigma_k \;:=\; a_{nn}^{(k)} \,, \quad k = 1, 2, 3, \ldots .$$

Die Spektralverschiebung nach J.H.Wilkinson wählt σ_k hingegen als den zum n-ten Diagonalelement $a_{nn}^{(k)}$ nächstgelegenen reellen Eigenwert σ der 2-reihigen Untermatrix

$$\begin{pmatrix} a_{n-1,n-1}^{(k)} & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{nn}^{(k)} \end{pmatrix} \qquad \text{mittels} \qquad \begin{vmatrix} a_{n-1,n-1}^{(k)} - \sigma & a_{n-1,n}^{(k)} \\ a_{n,n-1}^{(k)} & a_{nn}^{(k)} - \sigma \end{vmatrix} \;=\; 0$$

und erreicht damit eine noch bessere Konvergenz in Algorithmus 20.

Mit der gewählten Verschiebung ist eine Konvergenz des Verfahrens gegen λ_n zu erwarten, so dass $a_{n,n-1}^{(k)}$ zu Null wird. Damit zerfällt die Matrix A_k wegen der Darstellung

$$A_k = \left(\begin{array}{ccccccc} a_{11} & a_{12} & a_{13} & \cdots & a_{1,n-2} & a_{1,n-1} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2,n-2} & a_{2,n-1} & a_{2n} \\ 0 & a_{23} & a_{33} & \cdots & a_{3,n-2} & a_{3,n-1} & a_{3n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & & \cdots & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} \\ \hline 0 & 0 & 0 & \cdots & 0 & 0 & a_{nn} \end{array}\right) \tag{23}$$

in die Eigenwertnäherung $\lambda_n := a_{nn}$ und die $(n-1)$-reihige Hessenbergmatrix A, die durch Streichung der letzten Zeile und Spalte aus A_k entsteht. Sie enthält das Restspektrum $\lambda_1, \lambda_2, \ldots, \lambda_{n-1}$ mit den Voraussetzungen des Satzes 17 und kann analog mit Algorithmus 20 behandelt werden usw. Mit jedem gefundenen Eigenwert reduziert sich somit die Dimension der Matrix A. Tritt bei diesem Vorgehen ein Paar konjugiert komplexer Eigenwerte λ_{n-1}, λ_n

Algorithmus 20 (QR-Algorithmus mit Shift)
Function $[Q, R] =$ QRshift(A, tol)

1. Setze $U := I$ (Einheitsmatrix)
2. For $k = 1, 2, \ldots$ do
   - 2.1. Berechne den Shift σ
   - 2.2. Zerlege $A - \sigma I = QR$ nach Algorithmus 11
   - 2.3. Berechne $A := RQ + \sigma I$
   - 2.4. Datiere auf: $U := UQ$
   - 2.5. Falls in A gilt: $|a_{i+1,i} \cdot a_{i+2,i+1}| < tol \; \forall \, i$, so gehe zu Schritt 3
3. Return $R := A$ und $Q := U$

auf, so würde ein komplexer Wilkinson-Shift entstehen. Die Rechnung im Komplexen kann vermieden werden, wenn wir zwei Schritte mit reellen Verschiebungen durchführen und mit diesem *QR-Doppelschritt* eine QR-Transformation der Form

$$A_{k+2} \;=\; Q_{k+1}^T Q_k^T A_k Q_k Q_{k+1}$$

ausführen. Details dazu und weitere Hinweise zur effizienten Umsetzung des Verfahrens findet der interessierte Leser in [28].

3.2.3 Zeitkomplexität

Bei diesem iterativen Verfahren hängt die Zahl der Operationen nicht nur von der Dimension n, sondern auch von der Iterationszahl k und der konkreten Eigenwertverteilung ab. Für vollbesetzte Matrizen mit zufällig verteilten Einträgen beträgt der *arithmetische Aufwand* im Durchschnitt $10n^3 \ldots 20n^3$ Operationen, falls nur die Eigenwerte berechnet werden. In Abb. 4 wird dies im Zeitvergleich mit 1.5 LU-Zerlegungen bestätigt (n^3 Operationen). Wegen der Aufmultiplikation der Q_k erhöht sich dieser Wert bei Eigenvektorberechnung auf ungefähr $35n^3 \ldots 50n^3$. Dagegen reduziert sich im Falle symmetrischer Matrizen die Zeitkomplexität um den Faktor 2.

MATLAB stellt die beiden Funktionen `schur` und `eig` zur Lösung des vollständigen Eigenwertproblems bereit. Beide Funktionen benutzen intern den QR-Algorithmus in verschie-

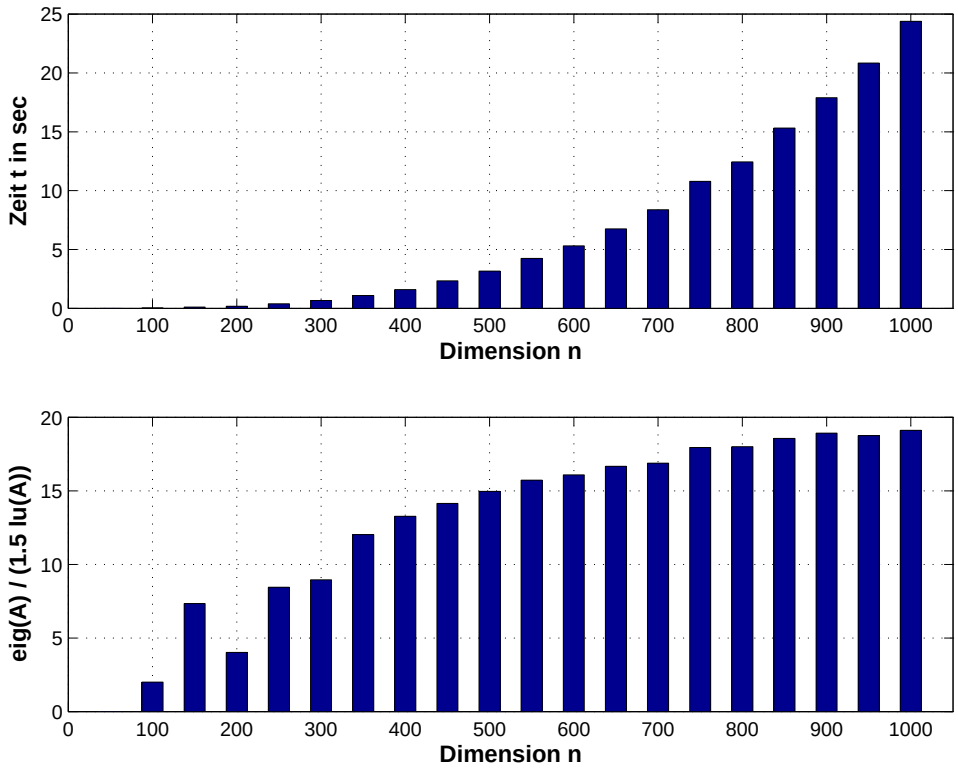

Abbildung 4: $R = eig(A)$ für zufällig erzeugte Matrizen

denen Varianten, angepasst an die konkreten Eigenschaften der Matrix A (reell, komplex, symmetrisch, hermitesch, voll besetzt, sparse).

1. Die reelle Schur-Zerlegung (12) mit $R = Q^T A Q$ wird durch
   `R = schur(A)` und `[Q,R] = schur(A)`
   geliefert. Ist A komplex, so steht in R die komplexe Schur-Form mit den Eigenwerten auf der Diagonalen.
2. Das Kommando `d = eig(A)` liefert im reellen und im komplexen Fall den Vektor der Eigenwerte, während
   `[V,D] = eig(A)`
   eine Diagonalmatrix D der Eigenwerte und eine Matrix V der zugehörigen Eigenvektoren (Spalten) gemäß der Darstellung $AV = VD$ erzeugt.
3. Das verallgemeinerte Eigenwertproblem $Ax = \lambda Bx\,,\ x \neq 0\,,$ mit den zwei Matrizen $A, B \in \mathbb{R}^{n\times n}$ wird durch die analogen Befehle
   `d = eig(A,B)` und `[V,D] = eig(A,B)`
   gelöst.

Beispiel 21 1. Die Matrix A besitzt neben dem einfachen Eigenwert $\lambda_1 = -1$ den algebraisch 2-fachen Eigenwert $\lambda_{2,3} = 1$. Dessen zugehöriger Eigenraum ist allerdings nur eindimensional und wird von $v_2 = (1, -2, 1)^T$ aufgespannt.

$$A = \begin{pmatrix} 6 & 12 & 19 \\ -9 & -20 & -33 \\ 4 & 9 & 15 \end{pmatrix}.$$

Das Kommando `[V,D] = eig(A)` liefert die Eigenwert-Näherungen

```
lambda = diag(D) =   -0.99999999999999
                      1.00000008190354
                      0.99999991809645
```

und die fast identischen Eigenvektoren v_2 und v_3 in

```
V =
  -0.47409982303502  -0.40824831461279  -0.40824826631493
   0.81274255377432   0.81649657349729   0.81649658835817
  -0.33864273073930  -0.40824828117581  -0.40824829975191
```

so dass V mit $cond(V) = 9.506 \cdot 10^7$ keine numerisch brauchbare Basis des $\mathbb{R}^3$ bildet.

2. Die reelle Schur-Zerlegung `[Q,R] = schur(A)` mittels des QR-Algorithmus

```
Q =
  -0.47409982303502   0.66475256462620   0.57735031447212
   0.81274255377432   0.07820614937187   0.57735027451698
  -0.33864273073930  -0.74295883198419   0.57735021857978
R =
  -1.00000000000001  20.78461273540836 -44.69483542654644
                   0   1.00000004152261  -0.60955691533074
                   0                   0   0.99999995847740
```

ergibt dagegen mit den Spaltenvektoren von Q eine stabile Orthonormalbasis des $\mathbb{R}^3$ mit $cond(Q) = 1.0$. □

4 Krylov-Unterraum-Methoden

Für Matrizen mit Tausenden von Zeilen ist auch das QR-Verfahren mit Spektralverschiebungen (vgl. Algorithmus 20) nicht mehr effizient. Andererseits approximiert das Verfahren der verbesserten Vektoriteration (vgl. Algorithmus 3) nur einen einzigen Eigenwert. Wir beobachten dabei, dass es zu gewähltem Startvektor v auch eine Basis des sogenannten Krylov-Unterraumes

$$\mathcal{K}_m \;=\; \mathcal{K}_m(A, v) \;:=\; span\{v, Av, \ldots, A^{m-1}v\} \tag{24}$$

erzeugt. Davon werden jedoch stets nur die beiden zuletzt berechneten Vektoren genutzt. Nun wollen wir den gesamten Unterraum speichern und damit eine Approximation von m der n Eigenwerte gewinnen. Bei großen Matrizen A kann man damit das Gesamtspektrum $\sigma(A)$ mitunter gut lokalisieren.

4.1 Krylov-Unterräume und Arnoldi-Verfahren

4.1.1 Projektionsverfahren

Projektionsverfahren werden in [15] zur iterativen Lösung großdimensionaler reeller Gleichungssysteme

$$Ax \;=\; a\,, \quad A \in \mathbb{R}^{n\times n},\; a, x \in \mathbb{R}^n \tag{25}$$

mit regulärer Koeffizientenmatrix A eingeführt. Die dort betrachteten Verfahren werden auch für das Eigenwertproblem (1) bedeutsam sein, weshalb sie hier kurz vorgestellt werden. Ist eine Näherungslösung x_m von (25) bekannt, so heißt

$$r_m \;:=\; a - Ax_m \tag{26}$$

Residuenvektor (das Residuum) von x_m. Das Ziel iterativer Verfahren besteht darin, ausgehend von einem Startvektor x_0, eine Folge von Näherungen (x_m) zu erzeugen, deren Residuen r_m rasch gegen Null konvergieren. Ein Projektionsverfahren sucht eine Näherungslösung x_m in einem m-dimensionalen affinen Unterraum $x_0 + \mathcal{K}_m$ von $\mathbb{R}^n$. Dabei soll m sehr klein gegenüber der Dimension n sein und x_0 eine geeignete Startnäherung darstellen. Um ein Element δ_m des „Suchraumes“ (engl. *search subspace*) $\mathcal{K}_m$ bestimmen zu können, sind m Bedingungen aufzustellen, die in der Regel als Orthogonalitätsbedingungen formuliert werden. Dazu definiert man einen m-dimensionalen Test-Unterraum (engl. *subspace of constraints*) $\mathcal{L}_m$ und fordert, dass das Residuum r_m orthogonal auf $\mathcal{L}_m$ steht. Es ist also ein solches $x_m \in x_0 + \mathcal{K}_m$ zu finden, für das gilt

$$r_m \;=\; a - Ax_m \;\perp\; \mathcal{L}_m\,. \tag{27}$$

Wegen $x_m = x_0 + \delta_m\,, \delta_m \in \mathcal{K}_m$, kann die Approximationsaufgabe auch durch

$$\langle r_0 - A\delta_m, w\rangle_2 \;=\; 0 \qquad \forall\, w \in \mathcal{L}_m \tag{28}$$

mit dem Euklidischen Skalarprodukt in $\mathbb{R}^n$ beschrieben werden (vgl. Abb. 5).

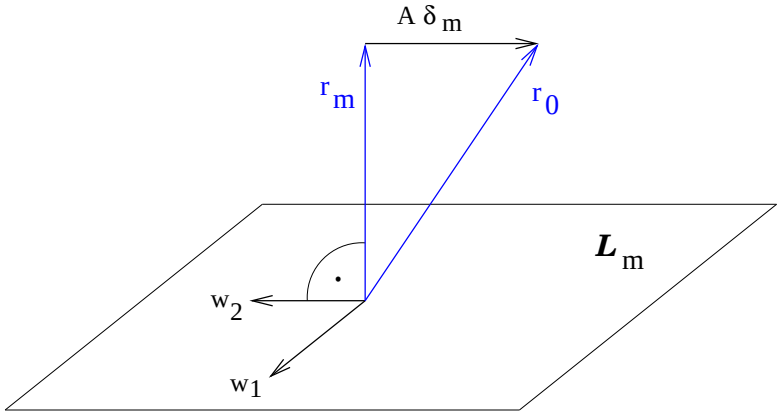

Abbildung 5: Projektion des Residuums auf $\mathcal{L}_m$

Definition 22 (Projektionsverfahren)

(*i*) *Ein Projektionsverfahren bestimmt zu* $x_0 \in \mathbb{R}^n$ *eine Folge von Näherungslösungen* $x_m = x_0 + \delta_m\,,\ \delta_m \in \mathcal{K}_m\,,$ *mit den* m*-dimensionalen Unterräumen* $\mathcal{K}_m$ *und* $\mathcal{L}_m\,,$ *die die Bedingung (27) erfüllen.*

(*ii*) *Ist* $\mathcal{K}_m = \mathcal{L}_m\,,$ *so heißt (27) Galerkin-Bedingung und das Projektionsverfahren ist orthogonal.*

(*iii*) *Ist* $\mathcal{K}_m \neq \mathcal{L}_m\,,$ *so heißt (27) Petrov-Galerkin-Bedingung und das Projektionsverfahren ist schief.*

Sei $V = (v_1, v_2, \ldots, v_m)$ die $n \times m-$Matrix, deren Spaltenvektoren eine Basis des Unterraumes $\mathcal{K}_m$ bilden und $W = (w_1, w_2, \ldots, w_m)$ eine entsprechende Matrix für eine Basis von $\mathcal{L}_m$. Dann kann x_m offensichtlich in der Gestalt

$$x_m \;=\; x_0 + Vy \quad \text{mit} \quad y \in \mathbb{R}^m \tag{29}$$

dargestellt werden und die Orthogonalitätsbedingung (27) lässt sich in der Form

$$W^T r_m \;=\; W^T(a - Ax_m) \;=\; W^T(r_0 - AVy) \;=\; 0$$

notieren. Mit den Abkürzungen $A_m := W^T AV$ und $a_m := W^T r_0$ reduziert sich das großdimensionale System $Ax = a$ so auf das i.A. kleine m-dimensionale Gleichungssystem

$$A_m y \;=\; a_m$$

dessen Lösung y mit geringem Aufwand bestimmbar ist, wenn wir die Regularität von A_m voraussetzen. Formale Auflösung nach y und Einsetzen in (29) liefert dann die kompakte Lösungsdarstellung

$$x_m \;=\; x_0 + V(W^T AV)^{-1} W^T r_0\,. \tag{30}$$

Alternativ zur Orthogonalitätsforderung (27) kann man auch eine Minimierungsbedingung

$$\|r_m\|_2 \;=\; \|a - Ax_m\|_2 \;\Longrightarrow\; Min! \quad \text{für} \quad x_m \in x_0 + \mathcal{K}_m \tag{31}$$

mit der Euklidischen Norm für das Residuum aufstellen. Setzen wir die Darstellung (29) für x_m ein, so ergibt sich ein freies Minimierungsproblem. Die notwendige Bedingung für ein lokales Extremum liefert die Gleichungen

$$(AV)^T(r_0 - AVy) \;=\; 0\,.$$

Wir setzen also speziell den Unterraum $\mathcal{L}_m = A\mathcal{K}_m$ an, so dass wir $W = AV$ wählen können. Dann hat AV den Rang m, womit $A_m = (AV)^T(AV)$ regulär ist und diese Bedingung auch als Projektionsverfahren in Definition 22 eingeordnet werden kann. Eine detaillierte Darstellung zahlreicher Projektionsverfahren findet man in [27].

4.1.2 Krylov-Unterräume und Arnoldi-Verfahren

Eine effiziente Bestimmung der Lösungsnäherungen x_m erfordert einen schrittweisen Aufbau der Unterräume $\mathcal{K}_m$ für $m = 1, 2, 3, \ldots$ mit möglichst wenigen Operationen, z.B. mit wenigen Matrix-Vektor-Multiplikationen Av. Dafür haben sich *Krylov-Unterräume* bewährt.

Definition 23 (Krylov-Unterraum) *Zu gegebenem $v \in \mathbb{R}^n$ hat ein Krylov-Unterraum die Form*

$$\mathcal{K}_m \;=\; \mathcal{K}_m(A, v) \;:=\; span\{v, Av, \ldots, A^{m-1}v\}\,. \tag{32}$$

Speziell sei zu einer Lösungsnäherung x_0 der Krylov-Unterraum

$$\mathcal{K}_m \;=\; \mathcal{K}_m(A, r_0) \;:=\; span\{r_0, Ar_0, \ldots, A^{m-1}r_0\} \tag{33}$$

mit dem Residuum $r_0 = b - Ax_0$ definiert.

Offenbar ist $\mathcal{K}_m$ der Unterraum aller Vektoren in $\mathbb{R}^n$, die man als $x = p(A)v$ darstellen kann, wobei p ein Polynom vom maximalen Grad $m-1$ ist. Als *Minimalpolynom* eines gegebenen Vektors v bezeichnet man das Nichtnullpolynom p mit minimalem Grad, mit dem $p(A)v = 0$ gilt. Der Grad μ dieses Polynoms von v in Bezug auf A wird *Grad von v (bezüglich A)* genannt. Als Folgerung aus dem bekannten Cayley-Hamilton-Theorem kann der Grad von v die Raumdimension n nicht übersteigen. Die folgenden Eigenschaften von Krylov-Unterräumen sind nun leicht zu beweisen:

Satz 24

(*i*) *Sei μ der Grad von v. Dann ist $\mathcal{K}_\mu$ invariant unter A und $\mathcal{K}_m = \mathcal{K}_\mu \ \forall m \geq \mu$.*

(*ii*) *Der Krylov-Unterraum $\mathcal{K}_m$ hat genau dann die Dimension m, wenn der Grad μ von v nicht kleiner als m ist, d.h.*

$$dim(\mathcal{K}_m) = m \quad \Longleftrightarrow \quad grad(v) \geq m. \tag{34}$$

Damit gilt $dim(\mathcal{K}_m) = \min\{m, grad(v)\}$.

Beweis: Die Vektoren $v, Av, \ldots, A^{m-1}v$ bilden genau dann eine Basis von $\mathcal{K}_m$, wenn für jede Menge von m Skalaren α_i, $i = 0, \ldots, m-1$ unter denen mindestens ein $\alpha_i \neq 0$ ist, die Linearkombination $\sum_{i=0}^{m-1} \alpha_i A^i v \neq 0$ ist. Dies ist äquivalent der Bedingung, dass das einzige Polynom mit Grad $\leq m-1$, für dass $p(A)v = 0$ gilt, das Nullpolynom ist. □

Zuerst wollen wir eine Orthonormalbasis des Krylov-Raumes $\mathcal{K}_m$ berechnen. Dazu wenden wir das aus der Linearen Algebra bekannte Gram-Schmidt-Orthogonalisierungsverfahren auf die Krylov-Basis $\{v, Av, \ldots, A^{m-1}v\}$ an und erhalten das *Arnoldi-Verfahren* in Algorithmus 25. Die Vektoren v_i, $i = 1, \ldots, m$ sind per Konstruktion orthonormal. Dass sie auch den

Algorithmus 25 (Arnoldi-Verfahren)

Function $[V, H] =$ Arnoldi(v, A, m)

1. Normiere $v_1 = v/\|v\|_2$
2. For $j = 1(1)m$ do
   1. Berechne $h_{ij} = (Av_j, v_i)$ für $i = 1(1)j$
   2. Berechne $w_j = Av_j - \sum_{i=1}^{j} h_{ij} v_i$
   3. Berechne $h_{j+1,j} = \|w_j\|_2$
   4. Falls $h_{j+1,j} = 0$, so STOP
   5. Normiere $v_{j+1} = w_j / h_{j+1,j}$
3. Return $V = (v_j)$ und $H = (h_{i,j})$

Raum $\mathcal{K}_m$ aufspannen, folgt aus der Tatsache, dass jeder Vektor v_j der Form $q_{j-1}(A)v_1$

genügt, wobei q_{j-1} ein Polynom vom Grade $j-1$ ist. Denn für $j=1$ ist das Ergebnis offensichtlich, da $v_1 = q_0(A)v_1$ mit $q_0(A) \equiv 1$. Angenommen, das Ergebnis gilt für alle $n \leq j$, d.h. $v_n = q_{n-1}(A)v_1$. Für $j+1$ erhält man dann

$$h_{j+1}v_{j+1} = Av_j - \sum_{i=1}^{j} h_{ij}v_i = Aq_{j-1}(A)v_1 - \sum_{i=1}^{j} h_{ij}q_{i-1}(A)v_1 \,,$$

woraus sich die Darstellung des Polynoms

$$q_j(A) = \underbrace{Aq_{j-1}(A)}_{Grad\ j} - \underbrace{\sum_{i=1}^{j} h_{ij}q_{i-1}(A)}_{Grad\ j-1}$$

ergibt. Also kann der Vektor v_{j+1} durch ein Polynom der Form $q_j(A)v_1$ dargestellt werden, wobei q_j vom Grad j ist. Induktiv haben wir damit das Verfahren verifiziert:

Satz 26 *Angenommen, Algorithmus 25 stoppt nicht vor dem m-ten Schritt. Dann bilden die Vektoren $v_1, v_2, \ldots, v_m$ eine Orthonormalbasis des Krylov-Unterraums $\mathcal{K}_m(A, v)$.*

Für die folgenden Verfahren ist die Matrixdarstellung des Algorithmus wesentlich.

Satz 27 *V_m sei die $n \times m$-Matrix mit den Spaltenvektoren $v_1, \ldots, v_m$. Weiterhin sei $\bar{H}_m$ die $(m+1) \times m$-Hessenberg-Matrix, deren Einträge $h_{ij} \neq 0$ durch den Algorithmus 25 definiert sind und H_m die Matrix, die entsteht, wenn man in der Matrix $\bar{H}_m$ die letzte Zeile weglässt. Dann gelten die folgenden Gleichungen:*

$$AV_m = V_{m+1}\bar{H}_m = V_m H_m + w_m e_m^T \tag{35}$$

$$V_m^T A V_m = H_m \,. \tag{36}$$

Beweis: Aus den Schritten 2.2, 2.3 und 2.5 des Algorithmus 25 folgt die Gleichung

$$Av_j = \sum_{i=1}^{j+1} h_{ij}v_i, \qquad j = 1, 2, \ldots, m, \tag{37}$$

die in Matrixdarstellung mit dem m-ten Einheitsvektor e_m die Beziehung (35) liefert. Gleichung (36) erhält man, wenn man beide Seiten von (35) mit V_m^T multipliziert und die Orthonormalität von $\{v_1, \ldots, v_m\}$ berücksichtigt. □

Bemerkung 28 1. Der Algorithmus stoppt, wenn die Norm von w_j im Schritt j verschwindet und so v_{j+1} nicht berechnet werden kann. In [27] wird gezeigt, dass das Arnoldi-Verfahren genau dann im Schritt j abbricht, wenn das Minimalpolynom von v_1 den Grad j hat.

2. Bei größerer Dimension m empfiehlt sich die Anwendung der *modifizierten Gram-Schmidt-Orthogonalisierung*, welche auf die Variante des Arnoldi-Verfahrens in Algorithmus 29 führt.

Noch robustere Varianten des Arnoldi-Verfahrens, z.B. unter Benutzung der Householder-Orthogonalisierung, findet man in [27].

Algorithmus 29 (Arnoldi-Verfahren modifiziert)
Function $[V, H] =$ Arnoldimod(v, A, m)

1. Normiere $v_1 = v/\|v\|_2$
2. For $j = 1(1)m$ do
   1. Berechne $w_j = Av_j$
   2. For $i = 1(1)j$ do
      1. $h_{ij} = (w_j, v_i)$
      2. $w_j = w_j - h_{ij}v_i$
   3. Berechne $h_{j+1,j} = \|w_j\|_2$
   4. Falls $h_{j+1,j} = 0$, so STOP
   5. Normiere $v_{j+1} = w_j/h_{j+1,j}$
3. Return $V = (v_j)$ und $H = (h_{i,j})$

4.2 Lanczos-Biorthogonalisierung

Für symmetrische positiv definite Systeme $Ax = a$ haben M.R.Hestenes und E.Stiefel die leistungsfähige Methode der konjugierten Gradienten (CG-Verfahren) entwickelt (vgl. [21]). Wir wollen den von C. Lanczos verallgemeinerten Zugang betrachten, mit dem beliebige reguläre Systeme gelöst werden können. Das so gewonnene *bikonjugierte Gradientenverfahren (engl. biconjucate gradient method (BiCG)* löst dabei nicht nur das Originalsystem $Ax = a$, sondern bei Bedarf auch duale (transponierte) Systeme $A^\top x^* = a^*$ mit beliebiger rechter Seite a^* . Es stellt ein Projektionsverfahren auf den Krylov-Unterraum

$$\mathcal{K}_m := \mathcal{K}_m(A, v_1) = \text{span}\{v_1, Av_1, \ldots, A^{m-1}v_1\}\,, \tag{38}$$

dar, der orthogonal zum *transponierten Krylov-Unterraum*

$$\mathcal{L}_m := \mathcal{K}_m(A^\top, w_1) = \text{span}\{w_1, A^\top w_1, \ldots (A^\top)^{m-1}w_1\} \tag{39}$$

ist. Dabei sei $v_1 = r_0/\|r_0\|_2$. Der Vektor w_1 ist beliebig, vorausgesetzt $\langle v_1, w_1\rangle \neq 0$, er wird jedoch oft gleich v_1 gewählt. Falls auch das duale System $A^\top x^* = a^*$ gelöst werden soll, so erhält man w_1 analog zu v_1 durch die Normierung des Anfangsresiduums $r_0^* = a^* - A^\top x_0^*$ des dualen Systems.

Im ersten Teil des Verfahrens wird ein Paar biorthogonaler Basen $V = (v_1, v_2, \ldots, v_m)$ und $W = (w_1, w_2, \ldots, w_m)$ für die beiden Krylov-Unterräume $\mathcal{K}_m$ und $\mathcal{L}_m$ mittels der *Lanczos-Biorthogonalisierung* erzeugt (vgl. Algorithmus 30). Die berechneten Größen δ_{j+1} und β_{j+1} stellen Skalierungsfaktoren für die Vektoren v_{j+1} und w_{j+1} dar. Aus den Schritten 5–7 des Algorithmus folgt, dass die Bedingung $\langle v_{j+1}, w_{j+1}\rangle = 1$ stets erfüllt ist. Die α_j, β_j und δ_j

Algorithmus 30 (Lanczos-Biorthogonalisierung)
Function $[V, W, T] =$ Biorthogonalisierung(A, m)

1. Wähle Vektoren v_1 und w_1 mit $(v_1, w_1) = 1$
2. Setze $\beta_1 = \delta_1 := 0$ und $w_0 = v_0 := 0$
3. For $j = 1(1)m$ do
   1. $\alpha_j = \langle Av_j, w_j \rangle$
   2. $\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}$
   3. $\hat{w}_{j+1} = A^\top w_j - \alpha_j w_j - \delta_j w_{j-1}$
   4. $\delta_{j+1} = |\langle \hat{v}_{j+1}, \hat{w}_{j+1} \rangle|^{1/2}$. Falls $\delta_{j+1} = 0$ ist, so STOP.
   5. $\beta_{j+1} = \langle \hat{v}_{j+1}, \hat{w}_{j+1} \rangle / \delta_{j+1}$
   6. $w_{j+1} = \hat{w}_{j+1} / \beta_{j+1}$
   7. $v_{j+1} = \hat{v}_{j+1} / \delta_{j+1}$
4. Return $V = (v_j)$, $W = (w_j)$ und $T = tridiag(\delta_j, \alpha_j, \beta_j)$

fassen wir zur Tridiagonalmatrix

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix} \tag{40}$$

zusammen. Wir können feststellen, dass für die Vektoren $v_i \in \mathcal{K}_m(A, v_1)$ ist, während $w_j \in \mathcal{K}_m(A^\top, w_1)$ gilt. Der gesamte Algorithmus wird durch folgenden Satz begründet:

Satz 31 *Falls der Algorithmus nicht vor dem m-ten Schritt abbricht, so bilden die Vektoren v_i, $i = 1, \ldots, m$ und w_j, $j = 1, \ldots, m$ ein biorthonormales System, d.h.*

$$\langle v_j, w_i \rangle = \delta_{ij} \qquad i = 1(1)m,\ j = 1(1)m, \quad (\delta_{ij} - \textit{Kronecker-Symbol}) \,. \tag{41}$$

Darüber hinaus ist $(v_i)_{i=1,2,\ldots,m}$ eine Basis von $\mathcal{K}_m(A, v_1)$ und $(w_i)_{i=1,2,\ldots,m}$ eine Basis von $\mathcal{K}_m(A^\top, w_1)$. Es gelten folgende Gleichungen:

$$AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^\top \tag{42}$$

$$A^\top W_m = W_m T_m^\top + \beta_{m+1} w_{m+1} e_m^\top \tag{43}$$

$$W_m^\top A V_m = T_m \tag{44}$$

Beweis: Die Biorthogonalität der Vektoren v_i und w_i kann induktiv gezeigt werden. Es gilt $\langle v_1, w_1 \rangle = 1$ nach Schritt 1. Seien bereits die Vektoren $v_1, \ldots, v_j$ und $w_1, \ldots, w_j$ biorthogonal.

Wir wollen nun zeigen, dass die Vektoren $v_1, \ldots, v_{j+1}$ und $w_1, \ldots, w_{j+1}$ biorthogonal sind. Zuerst weisen wir nach, dass $\langle v_{j+1}, w_i \rangle = 0$ für $i \leq j$ ist. Denn falls $i = j$, so gilt

$$\langle v_{j+1}, w_j \rangle = \delta_{j+1}^{-1} [\langle Av_j, w_j \rangle - \alpha_j \langle v_j, w_j \rangle - \beta_j \langle v_{j-1}, w_j \rangle].$$

Das letzte Skalarprodukt verschwindet aufgrund der Induktionsannahme. Die beiden anderen Terme fallen wegen der Definition von $\alpha_j = \langle Av_j, w_j \rangle$ und der Tatsache weg, dass $\langle v_j, w_j \rangle = 1$ ist. Betrachten wir also das Skalarprodukt $\langle v_{j+1}, w_i \rangle$ mit $i < j$,

$$\begin{aligned} \langle v_{j+1}, w_i \rangle &= \delta_{j+1}^{-1} [\langle Av_j, w_i \rangle - \alpha_j \langle v_j, w_i \rangle - \beta_j \langle v_{j-1}, w_i \rangle] \\ &= \delta_{j+1}^{-1} [\langle v_j, A^\top w_i \rangle - \beta_j \langle v_{j-1}, w_i \rangle] \\ &= \delta_{j+1}^{-1} [\langle v_j, \beta_{i+1} w_{i+1} + \alpha_i w_i + \delta_i w_{i-1} \rangle - \beta_j \langle v_{j-1}, w_i \rangle] \,. \end{aligned}$$

Für $i < j - 1$ verschwinden wegen der Induktionsannahme alle Skalarprodukte in der obigen Gleichung. Für den Fall $i = j - 1$ gilt

$$\begin{aligned} \langle v_{j+1}, w_{j-1} \rangle &= \delta_{j+1}^{-1} [\langle v_j, \beta_j w_j + \alpha_{j-1} w_{j-1} + \delta_{j-1} w_{j-2} \rangle - \beta_j \langle v_{j-1}, w_{j-1} \rangle] \\ &= \delta_{j+1}^{-1} [\beta_j \langle v_j, w_j \rangle - \beta_j \langle v_{j-1}, w_{j-1} \rangle] \\ &= 0 \,. \end{aligned}$$

Analog kann gezeigt werden, dass $\langle v_i, w_{j+1} \rangle = 0$ für $i \leq j$ ist. Zudem gilt $\langle v_{j+1}, w_{j+1} \rangle = 1$. Damit ist die Induktionsbehauptung bewiesen. Der zweite Teil des Satzes kann analog zu Satz 27 und den Gleichungen (35), (35) für das Arnoldi-Verfahren bewiesen werden. □

Algorithmus 32 (Bi-Lanczos-Verfahren)

Function $[x_m] =$ Bi_Lanczos(A, a, m, x_0)

1. Berechne $r_0 = a - Ax_0$, $\beta = \|r_0\|_2$ und $v_1 = r_0/\beta$
2. Bestimme die Lanczos-Basen V_m, W_m und die Tridiagonalmatrix T_m mit Algorithmus 30
3. Löse $T_m y_m = \beta e_1$ nach y_m auf
4. Return $x_m = x_0 + V_m y_m$

Mit den ermittelten Lanczos-Basen der Krylov-Räume $\mathcal{K}_m$ und $\mathcal{L}_m$ können wir nun das lineare Gleichungssystem und das duale System leicht lösen. Der *zweiseitige Lanczos-Algorithmus (Bi-Lanczos)* startet dazu speziell mit $v_1 = r_0/\|r_0\|_2$ und $\beta = \|r_0\|_2$. Die Petrov-Galerkin-Bedingung (27) liefert in Matrixnotation für $x_m = x_0 + V_m y_m$ unter Benutzung von (44)

$$\begin{aligned} 0 &= W_m^T (a - Ax_m) = W_m^T (r_0 - AV_m y_m) \\ &= W_m^T \beta v_1 - W_m^T A V_m y_m \\ &= \beta e_1 - T_m y_m \end{aligned}$$

woraus die Darstellung

$$y_m \;=\; T_m^{-1}\beta e_1\;, \quad e_1 \;-\; \text{Einheitsvektor}$$

mit der Tridiagonalmatrix T_m folgt. Das Verfahren kann so in Form des Algorithmus 32 notiert werden.

Betrachten wir den Schritt 3 genauer. Wenn wir eine LU-Zerlegung der Tridiagonalmatrix in der Form $T_m \;=\; L_m R_m$ vornehmen, so lassen sich damit die beiden $n \times m-$Matrizen

$$P_m \;=\; V_m R_m^{-1} \;=\; (p_1, p_2, \ldots, p_m) \quad \text{und} \quad P_m^* \;=\; W_m L_m^{-1} \;=\; (p_1^*, p_2^*, \ldots, p_m^*)$$

mit den Spalten p_j und p_j^* bestimmen. Das *bikonjugierte Gradientenverfahren (BiCG)* konstruiert sukzessive diese Matrizen und die Näherungslösungen x_j und x_j^* für das Originalsystem $Ax = a$ und für das duale lineare System $A^T x^* = a^*$ samt den Residuen r_j und r_j^*. Ausgehend vom Bi-Lanczos-Verfahren liefert dieses Toleranz-gesteuerte Verfahren 33 die Lösungen und Residuen, bis eine gewünschte Genauigkeit erreicht wird (vgl. [27]).

Algorithmus 33 (Bi-CG-Verfahren)
Function $[x_j, x_j^*] = \mathsf{BiCG}(A, a, a^*, x_0, x_0^*, tol)$

1. Berechne $r_0 := a - Ax_0$ und $r_0^* := a^* - A^\top x_0^*$
2. Setze $p_0 := r_0$ und $p_0^* := r_0^*$
3. For $j = 0, 1, 2, \ldots$ while $\|r_j\|_2 > tol \cdot \|a\|_2$ do
   1. $\alpha_j := \langle r_j, r_j^* \rangle / \langle Ap_j, p_j^* \rangle$
   2. $x_{j+1} := x_j + \alpha_j p_j$
   3. $x_{j+1}^* := x_j^* + \alpha_j p_j^*$
   4. $r_{j+1} := r_j - \alpha_j A p_j$
   5. $r_{j+1}^* := r_j^* - \alpha_j A^\top p_j^*$
   6. $\beta_j := \langle r_{j+1}, r_{j+1}^* \rangle / \langle r_j, r_j^* \rangle$
   7. $p_{j+1} := r_{j+1} + \beta_j p_j$
   8. $p_{j+1}^* := r_{j+1}^* + \beta_j p_j^*$
4. Return x_j und x_j^*

Bemerkung 34 1. Ist A symmetrisch und positiv definit, so reduziert sich der Algorithmus auf das *konjugierte Gradientenverfahren (CG)*, das z.B. in [20], Band 1, S. 128ff, ausführlich behandelt wird.

2. In jedem Verfahrensschritt treten im Gegensatz zum GMRES-Verfahren (vgl. [27]) nun zwei Matrix-Vektor-Multiplikationen Ap_j und $A^\top p_j^*$ auf; zudem ist mit $\langle r_j, r_j^* \rangle = 0$ ein unerwünschter Verfahrensabbruch möglich. Allerdings liefert das Verfahren neben x_j auch die Näherungslösung x_j^* des dualen Systems.

3. Weitere Verbesserungen, wie das CGS-Verfahren und das BiCGSTAB-Verfahren, vermeiden die Multiplikation mit der Transponierten $A^\top$ und zeigen mitunter schnellere Konvergenz. Unerwünschte Verfahrensabbrüche werden durch eine so genannte look-ahead-Strategie vermieden, die der interessierte Leser in [27, 21] findet.

4.3 Ritz-Werte und Ritz-Vektoren

Die Krylov-Basis (24) erweist sich numerisch als wenig geeignet, denn deren Vektoren „drehen sich“ zunehmend gegen den dominierenden Eigenvektor und verlieren so ihre lineare Unabhängigkeit auf dem Computer (vgl. Beispiel 1). Hier bietet sich eine Orthogonalisierung mit dem Arnoldi-Verfahren 29 an. Zu gegebenem Vektor v und natürlicher Zahl m liefert das *modifizierte Arnoldi-Verfahren* 29 die $n \times m$-Matrix V_m mit den orthonormalen Spaltenvektoren $v_1, \ldots, v_m$. Weiterhin ist H_m die quadratische $m \times m$-Hessenberg-Matrix, deren Einträge $h_{ij} \neq 0$ ebenfalls durch den Algorithmus definiert sind. Nach Satz 27 genügen sie den folgenden Gleichungen

$$V_m^T A V_m \;=\; H_m \qquad \text{und} \qquad V_m^T V_m \;=\; I_m \tag{45}$$

mit der Einheitsmatrix I_m. Die Matrix H_m ist offenbar die Projektion von A auf den Krylov-Raum $\mathcal{K}_m(A, v)$. Sei (θ, y) ein *Eigenpaar* der Matrix H_m, d.h. θ ist ein (im Allgemeinen komplexer) Eigenwert mit zugehörigem m-dimensionalen Eigenvektor $y \neq 0$. Anwendung

Algorithmus 35 (Ritz-Werte und Ritz-Vektoren)
Function $[\theta, X] =$ ritzvalues (A, m)

1. Wähle Startvektor v
2. Bestimme die Arnoldi-Basis V_m sowie H_m mit Algorithmus 29
3. Bestimme die Eigenwerte θ_i und Eigenvektoren y_i von H_m mit QR-Algorithmus 20
4. Berechne die Ritz-Vektoren $x_i = V_m y_i,\ i = 1(1)m$
5. Return $\theta = (\theta_1, \theta_2, \ldots, \theta_m)$ und $X = (x_1, x_m, \ldots, x_m)$

von (45) ergibt

$$\begin{aligned} H_m y \;=\; \theta y \quad &\Leftrightarrow \quad V_m^T A V_m y - \theta V_m^T V_m y \;=\; 0 \\ &\Leftrightarrow \quad V_m^T (Ax - \theta x) \;=\; 0 \qquad \text{mit} \quad x = V_m y\,. \end{aligned} \tag{46}$$

Zwar erfüllt das Paar (θ, x) im Allgemeinen nicht die Eigenwertbeziehung für A, aber das Residuum $Ax - \theta x$ ist nun senkrecht zum Krylov-Raum $\mathcal{K}_m(A, v)$. Es genügt somit einer Galerkin-Bedingung 22, so dass (θ, x) als Approximation eines Eigenpaares von A aufgefasst werden kann.

Definition 36 (Ritz-Wert) *Der Wert θ heißt Ritz-Wert von A bezüglich des Krylov-Raumes $\mathcal{K}_m(A, v)$ und $x = V_m y$ ist der entsprechende Ritz-Vektor.*

Zur Bestimmung von m Ritz-Werten und -Vektoren ergibt sich damit der einfache Algorithmus 35. Als Startvektor ist $v = (1, 1, \ldots, 1)^T$ oder ein zufällig erzeugter Vektor üblich.

Beispiel 37 1. Mit `A = GALLERY('CHOW',500)` gewinnen wir die untere Hessenberg-Matrix der Dimension 500

$$A = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix},$$

für die das Spektrum ($\times$) und die Ritz-Werte mit $m = 15$, 25 und $m = 50$ in Abb. 6 dargestellt sind.

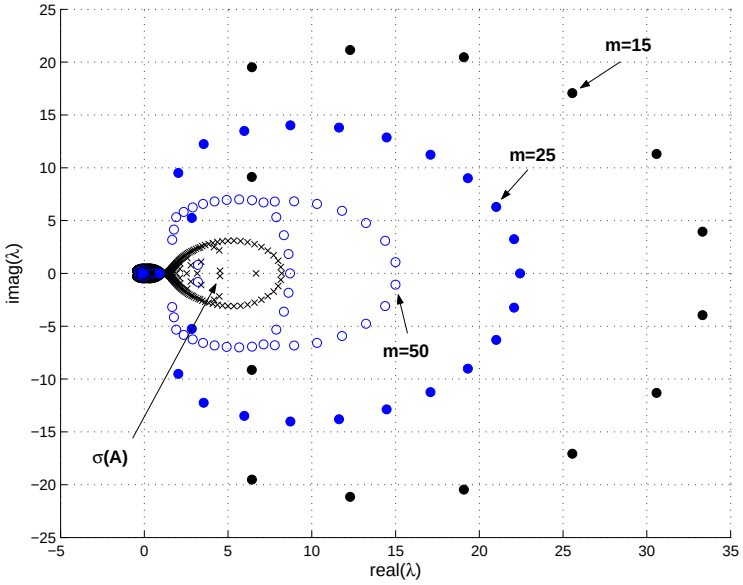

Abbildung 6: Ritz-Werte für A = GALLERY('CHOW',500)

2. Für die 1000-reihige Matrix `A = GALLERY('CHEBVAND',1000)` liefert das Verfahren bei $m = 20$ eine ungenaue Approximation des Randes des Spektrums von A (vgl. Abb. 7), allerdings mit nur 1.5 % der Rechenzeit von `eig(A)`. Mit 3.9 % der Rechenzeit erhalten wir dagegen in Abb. 8 die 50 Ritz-Werte bereits am Rande von $\sigma(A)$. □

Die Lage der Ritz-Werte hängt stark von den Eigenschaften der betreffenden Matrix A ab. Ritz-Werte finden sich oft in der Umgebung der „extremalen“ Eigenwerte λ_i der Matrix A, d.h. am Rande des Spektrums $\sigma(A)$ sowie am Rande des numerischen Wertebereichs.

Definition 38 (Numerischer Wertebereich einer Matrix) *Der numerische Wertebereich von $A \in \mathbb{C}^{n\times n}$ (Wertevorrat der Bilinearform, engl. numerical range)*[2] *ist die Menge*

[2]Der numerische Wertebereich $\mathcal{W}(A) \subset \mathbb{C}$ geht auf F. Hausdorff und O. Toeplitz zurück. Er darf nicht mit dem Wertebereich $R(A)$ (engl. range) der linearen Abbildung A verwechselt werden!

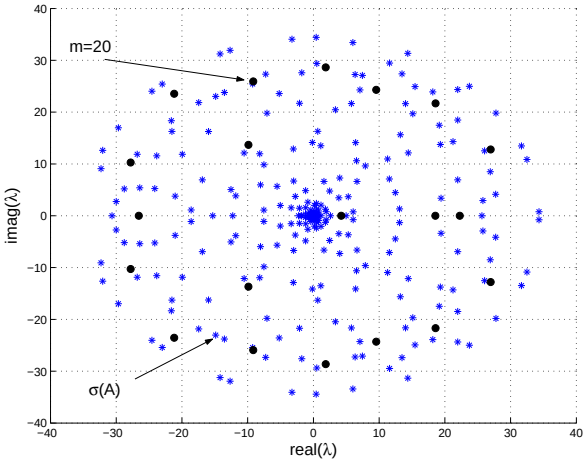

Abbildung 7: 20 Ritz-Werte

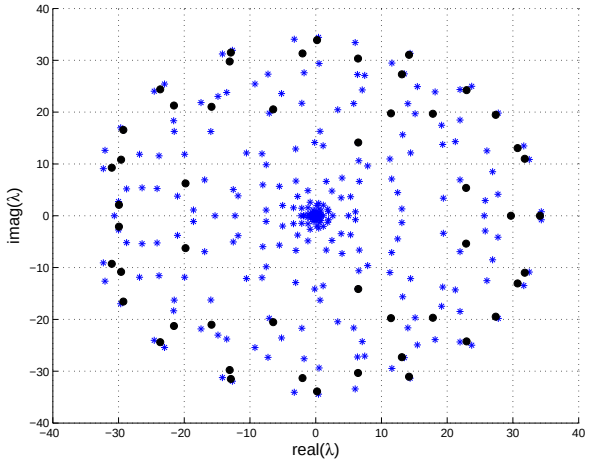

Abbildung 8: 50 Ritz-Werte

aller Rayleigh-Quotienten mit $x \in \mathbb{C}^n,\ x \neq 0$

$$\mathcal{W}(A) \ = \ \{\varrho \mid \varrho = \frac{x^*Ax}{x^*x}\ ,\ x \in \mathbb{C}^n,\ x \neq 0\} \ \subset \ \mathbb{C}\,. \tag{47}$$

Bei der Bildung von $\mathcal{W}(A)$ sind – auch im Falle reeller Matrizen A – alle komplexwertigen Vektoren x zu berücksichtigen, womit auch die äquivalente Darstellung

$$\mathcal{W}(A) \ = \ \{\varrho \mid \varrho = x^*Ax\ ,\ x \in \mathbb{C}^n,\ \|x\|_2 = 1\} \tag{48}$$

möglich ist. Mit dem Skalarprodukt $\langle x, y\rangle := y^*x = \sum_{i=1}^n x_i\bar{y}_i$ und dem Vektorraum $H = \mathbb{C}^n$ findet man die allgemeine Form (vgl. [11])

$$\mathcal{W}(A) \ = \ \{\langle Ax, x\rangle \mid x \in H,\ \|x\| = 1\}\ , \tag{49}$$

womit eine Verallgemeinerung für lineare Operatoren in Hilberträumen H möglich ist. Der numerische Wertebereich besitzt eine Reihe bemerkenswerter Eigenschaften:

Satz 39 (Eigenschaften des numerischen Wertebereichs)

(*i*) $\mathcal{W}(A)$ *ist kompakt und konvex (Toeplitz-Hausdorff).*

(*ii*) $\mathcal{W}(A)$ *enthält das Spektrum der Matrix* A.

(*iii*) $\mathcal{W}(A)$ *ist invariant unter unitären Transformationen, d.h.* $\mathcal{W}(U^*AU) \ = \ \mathcal{W}(A)$, *falls* U *unitär ist.*

(*iv*) $\mathcal{W}(A^*) \ = \ \{\varrho \mid \bar{\varrho} \in \mathcal{W}(A)\}$

(*v*) $\mathcal{W}(A+B) \ \subset \ \mathcal{W}(A) + \mathcal{W}(B) \ = \ \{\lambda + \mu \mid \lambda \in \mathcal{W}(A),\ \mu \in \mathcal{W}(B)\}$

(*vi*) *Ist* A *hermitesch, so ist* $\mathcal{W}(A)$ *das kleinste reelle Intervall, das alle Eigenwerte enthält.*

(*vii*) *Ist* A *schiefhermitesch, h.h.* $A^* = -A$, *so ist* $\mathcal{W}(A)$ *das kleinste rein imaginäre Intervall, das alle Eigenwerte enthält.*

BEWEIS: (i) wird in [11], S.3-4, aufwändig bewiesen, weshalb wir hier darauf verzichten. Behauptung (ii) ist offensichtlich, da für jedes Eigenpaar (λ, x) gilt

$$\frac{x^*Ax}{x^*x} \;=\; \frac{x^*\lambda x}{x^*x} \;=\; \lambda \;\in\; \mathcal{W}(A)\,.$$

(iii) $\varrho \in \mathcal{W}(U^*AU) \;\Leftrightarrow$ es existiert ein $x \in \mathbb{C}^n$, $x \neq 0$, mit $(x^*U^*AUx)/(x^*x) \in \mathcal{W}(A)$. Wegen

$$\frac{x^*U^*AUx}{x^*x} \;=\; \frac{(Ux)^*A(Ux)}{(Ux)^*(Ux)} \;=\; \frac{y^*Ay}{y^*y} \;\in\; \mathcal{W}(A) \quad \text{mit} \quad y = Ux \neq 0$$

ist die Behauptung erfüllt.

(iv) $\varrho \;=\; x^*A^*x \;\in\; \mathcal{W}(A^*)$ mit $\|x\|_2 = 1 \;\Leftrightarrow\; \bar{\varrho} \;=\; \bar{x}^*\bar{A}^*\bar{x} \;=\; x^*Ax \;\in\; \mathcal{W}(A)\,.$

(v) $\varrho \;=\; \dfrac{x^*(A+B)x}{x^*x} \;\in\; \mathcal{W}(U^*AU) \;\Rightarrow$

$$\varrho \;=\; \frac{x^*Ax}{x^*x} + \frac{x^*Bx}{x^*x} \;=\; \lambda + \mu \quad \text{mit} \quad \lambda \;=\; \frac{x^*Ax}{x^*x} \in \mathcal{W}(A)\,,\; \mu \;=\; \frac{x^*Bx}{x^*x} \in \mathcal{W}(B)\,.$$

Den Beweis von (vi) und (vii) findet man in [13], S.208. □

Mit diesen Eigenschaften kann das Spektrum einer Matrix A mitunter grob lokalisiert werden. Unitäre Ähnlichkeitstransformationen (iii) lassen dabei nicht nur $\sigma(A)$, sondern auch $\mathcal{W}(A)$ invariant. Die Aussage (v) des Satzes gilt i.A. für Spektren nicht, weshalb die Inklusion

$$\sigma(A+B) \;\subset\; \mathcal{W}(A+B) \;\subset\; \mathcal{W}(A) + \mathcal{W}(B)$$

oft nützlich ist.

Satz 40 (Bendixson) *Das Spektrum einer Matrix $A \in \mathbb{C}^{n\times n}$ liegt in dem Rechteck*

$$R \;=\; \mathcal{W}\left(\frac{A+A^*}{2}\right) \;+\; \mathcal{W}\left(\frac{A-A^*}{2}\right)\,. \tag{50}$$

BEWEIS: Wir zerlegen die Matrix A in ihren hermiteschen und ihren schiefhermiteschen Anteil

$$A \;=\; \frac{A+A^*}{2} \;+\; \frac{A-A^*}{2}$$

und wenden Teil (v) des Satzes 39 auf die beiden Matrizen an. Mit den Eigenschaften (vi) und (vii) für die beiden Summanden ergibt sich, dass R ein Rechteck ist. □

Für hinreichend großes m erhält man mit den m Ritz-Werten $\theta_1, \theta_2, \ldots, \theta_m$ eine Approximation von $\mathcal{W}(A)$. Denn man beweist leicht

Satz 41 *Alle Ritz-Werte θ einer Matrix $A \in \mathbb{C}^{n\times n}$ gehören zum numerischen Wertebereich $\mathcal{W}(A)$.*

BEWEIS: Sei (θ, y) ein Ritz-Paar nach Algorithmus 35, d.h. $H_m y = \theta y$. Mit $x := V_m y$ notieren wir den Rayleigh-Quotienten bezüglich A

$$\varrho = \frac{x^* A x}{x^* x} = \frac{y^* V_m^T A V_m y}{y^* V_m^T V_m y} = \frac{y^* H_m y}{y^* y} = \theta \in \mathcal{W}(A) ,$$

womit die Behauptung folgt. □

Die Zusammenhänge zwischen Ritz-Werten und „zugehörigen“ Eigenwerten sind für hermitesche Matrizen A weitgehend geklärt; Abschätzungen hierzu findet man z.B. in [13]. Das Verhalten der Ritz-Werte allgemeiner Matrizen wird untersucht (vgl. [4]).

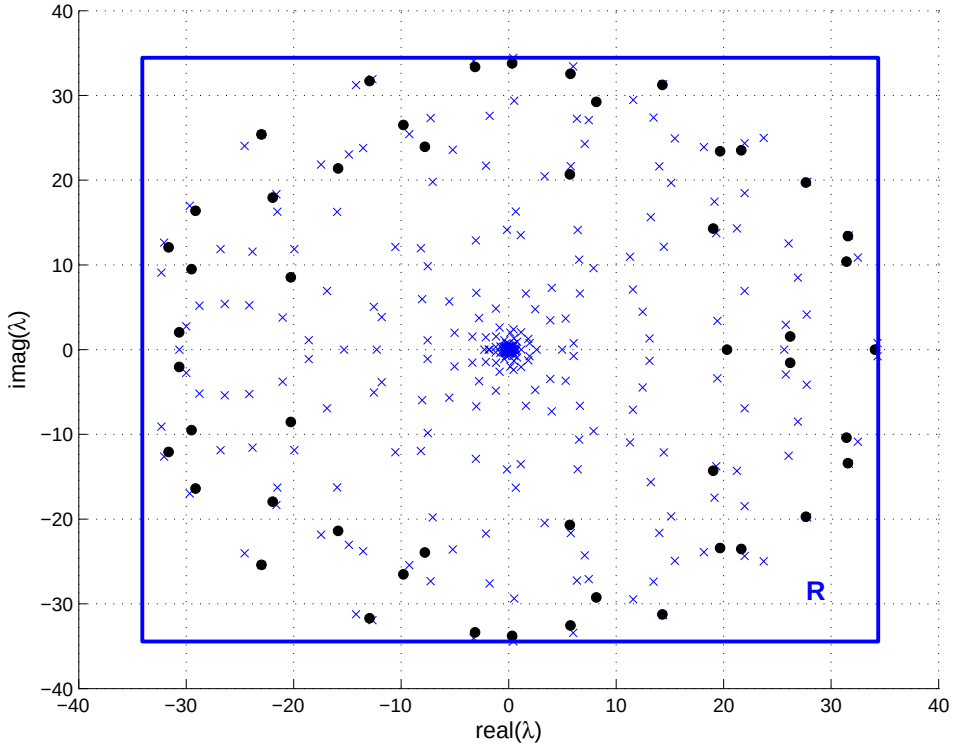

Abbildung 9: 50 Ritz-Werte und R für A = GALLERY('CHEBVAND',1000)

Beispiel 42 Für die 1000-reihige Matrix `A = GALLERY('CHEBVAND',1000)` des Beispiels 37 wird das Spektrum (×) zusammen mit den 50 Ritz-Werten (•) in Abb. 9 dargestellt. Das Rechteck R gemäß Satz 40 enthält sowohl $\sigma(A)$ als auch alle Ritz-Werte und grenzt – in diesem Falle – das Spektrum scharf ein. □

In jedem Schritt des Arnoldi-Verfahrens ist zwar nur eine Matrix-Vektor-Multiplikation erforderlich, so dass bei kleinem m der Algorithmus 35 effektiv arbeitet. Bei größerem m wird das Verfahren allerdings mit zunehmender Rekursionslänge ineffektiv.

4.4 Bi-Lanczos-Verfahren und Petrov-Werte

Das Verfahren der Lanczos-Biorthogonalisierung 30 betrachtet außer dem Krylov-Unterraum $\mathcal{K}_m := \mathcal{K}_m(A, v)$ auch den transponierten Krylov-Unterraum $\mathcal{L}_m := \mathcal{K}_m(A^\top, w)$ mit $\langle w, v \rangle \neq 0$. Satz 31 garantiert, dass die mit dem Verfahren erzeugten Basen $V = (v_1, v_2, \ldots, v_m)$ von

$\mathcal{K}_m$ und $W = (w_1, w_2, \ldots, w_m)$ von $\mathcal{L}_m$ ein biorthonormales System bilden. Darüber hinaus gilt die Gleichung

$$W_m^T A V_m \;=\; T_m \tag{51}$$

mit der (im Allgemeinen unsymmetrischen) Tridiagonalmatrix

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \delta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & \delta_m & \alpha_m \end{pmatrix} \tag{52}$$

Zwar sind pro Algorithmusschritt nun 2 Matrix-Vektor-Multiplikationen nötig, aber da ansonsten nur *3-Term-Rekursionen* auftreten, nimmt der Aufwand im Gegensatz zum Arnoldi-Verfahren mit wachsensem m nicht überproportional zu.

Sei (θ, y) ein *Eigenpaar* der Tridiagonalmatrix T_m, d.h. θ ist ein (im Allgemeinen komplexer) Eigenwert mit zugehörigem m-dimensionalen Eigenvektor $y \neq 0$. Wir wenden (51) an, nutzen die Biorthonormalität der Basen V und W aus und erhalten

$$\begin{aligned} T_m y \;=\; \theta y \quad &\Leftrightarrow \quad W_m^T A V_m y - \theta W_m^T V_m y \;=\; 0 \\ &\Leftrightarrow \quad W_m^T (Ax - \theta x) \;=\; 0 \qquad \text{mit} \quad x = V_m y\,. \end{aligned} \tag{53}$$

Offenbar steht nun das Residuum $Ax - \theta x$ senkrecht zum Krylov-Raum $\mathcal{L}_m$ und genügt einer *Petrov-Galerkin-Bedingung* 22. Für die Werte θ ist deshalb folgender Begriff üblich:

Definition 43 (Petrov-Wert) *Der Wert θ heißt Petrov-Wert von A bezüglich der Krylov-Räume $\mathcal{K}_m$ und $\mathcal{L}_m$. Der Vektor $x = V_m y$ ist der entsprechende rechte Petrov-Vektor, während $z = W_m y$ linker Petrov-Vektor heißt.*

Algorithmus 44 (Petrov-Werte und -Vektoren)

Function $[\theta, X, Z] =$ petrov_values (A, m)

1. Bestimme $V = (v_j)$, $W = (w_j)$ und $T = tridiag(\delta_j, \alpha_j, \beta_j)$ mit Algorithmus 30
2. Bestimme die Eigenwerte θ_i und Eigenvektoren y_i von T_m mit QR-Algorithmus 20
3. Berechne die rechten Petrov-Vektoren $x_i = V_m y_i,\ i = 1(1)m$
4. Berechne die linken Petrov-Vektoren $z_i = W_m y_i,\ i = 1(1)m$
5. Return $\theta = (\theta_1, \theta_2, \ldots, \theta_m)$, $X = (x_i)$ und $Z = (z_i)$

Algorithmus 44 bestimmt m Petrov-Werte und zugehörige Petrov-Vektoren von A. Diese approximieren entsprechende Eigenwerte von A sowie rechte und linke Eigenvektoren, d.h. Eigenvektoren von A und A^T. Den Startvektor v erzeugt man in der Regel zufällig und bestimmt w so, dass $\langle v, w \rangle = 1$ gilt.

Beispiel 45

1. Die 500-reihige Matrix `A = GALLERY('cycol',500)+GALLERY('CHOW',500)` besitzt neben einem Eigenwert-Cluster einige „Ausreißer“ (vgl. Abb. 10). Die Petrov-Werte mit $m = 25$ approximieren auch die isolierten Eigenwerte mit guter Genauigkeit.

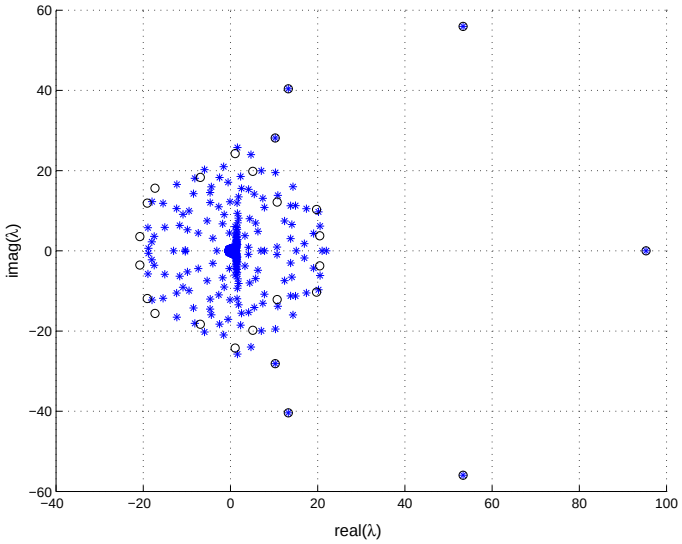

Abbildung 10: Petrov-Werte für A=GALLERY('cycol',500)+GALLERY('CHOW',500)

2. Für die 1000-reihige Matrix `A = GALLERY('CHEBVAND',1000)` approximiert das Bi-Lanczos-Verfahren bei $m = 20$ die extremalen Eigenwerte besser als das Arnoldi-Verfahren 35 (vgl. Abb. 11), allerdings braucht es 3.25 % der Rechenzeit von `eig(A)`. Mit 8.07 % der Rechenzeit erhalten wir die in Abb. 12 dargestellten 50 Petrov-Werte. □

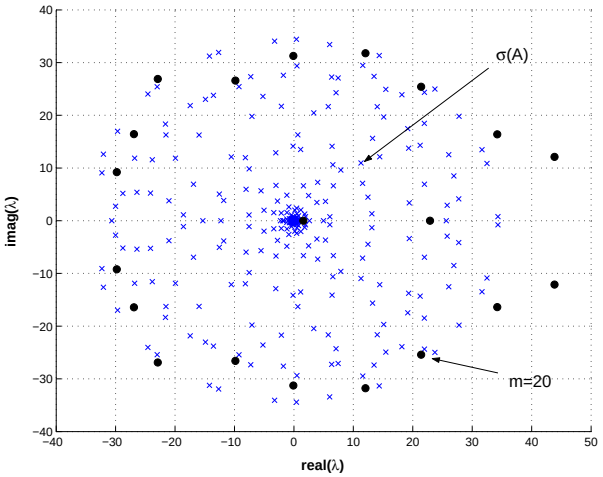

Abbildung 11: 20 Petrov-Werte

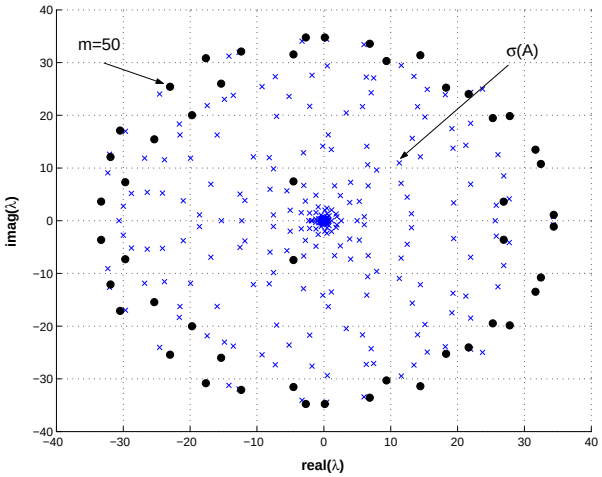

Abbildung 12: 50 Petrov-Werte

Schnelle Algorithmen für großdimensionale Eigenwertprobleme werden gegenwärtig zunehmend eingesetzt. Wir betrachten beispielhaft eine großdimensionale unsymmetrische Bandmatrix aus [4].

Beispiel 46 Die reellen pentadiagonalen $n \times n$-Bandmatrizen (vgl. [4], S.89)

$$A = \begin{pmatrix} 2 & 1 & -0.4 & 0 & \cdots & 0 & 0 \\ 0 & 2 & 1 & -0.4 & \cdots & 0 & 0 \\ 2 & 0 & 2 & 1 & \cdots & 0 & 0 \\ 0 & 2 & 0 & 2 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & 2 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 2 \end{pmatrix}.$$

werden in sparser Darstellung gespeichert. Für $n = 200$ zeigt Abb.13 das Spektrum ($\times$)

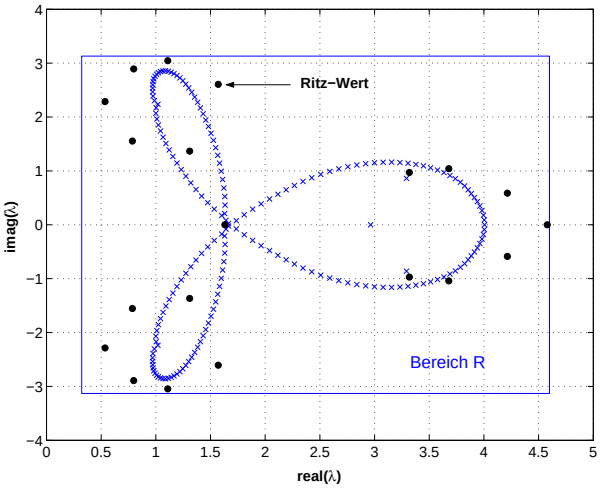

Abbildung 13: 20 Ritz-Werte

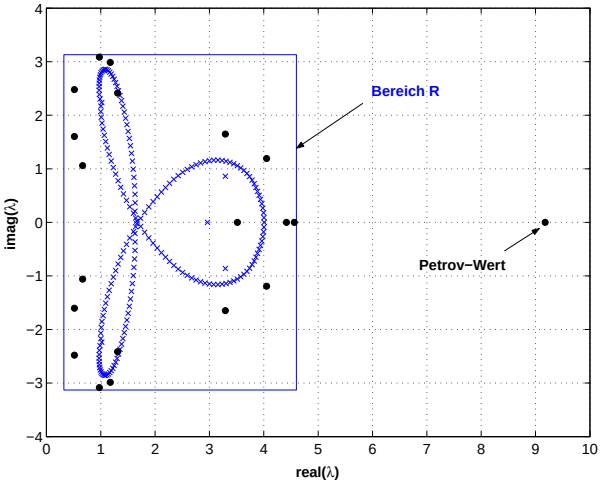

Abbildung 14: 20 Petrov-Werte

von A zusammen mit den 20 Ritz-Werten ($\bullet$) und den Bereich R gemäß (50). Bei den 20 Petrov-Werten in Abb.14 treten wie im vorigen Beispiel „Ausreißer“ auf.

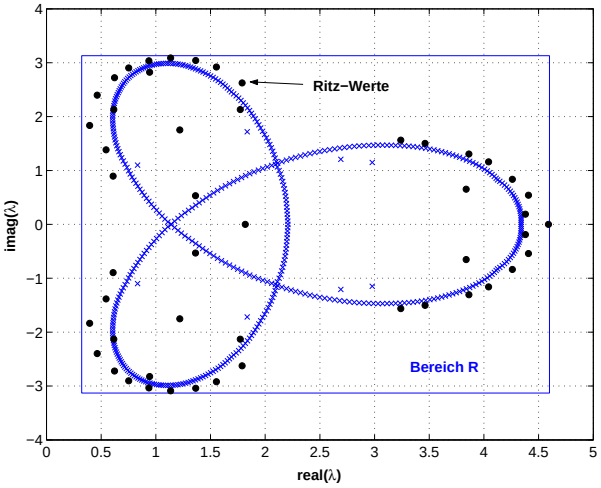

Abbildung 15: 50 Ritz-Werte

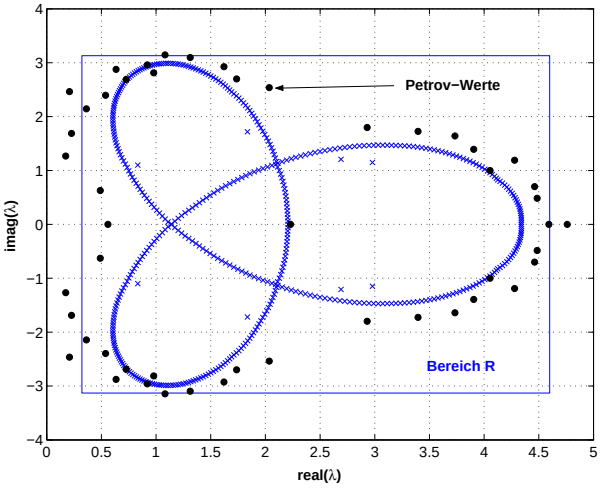

Abbildung 16: 50 Petrov-Werte

Im Falle der 500-reihigen Matrix A werden in Abb.15 die 50 Ritz-Werte gezeichnet, während

Abb.16 die ersten 50 Petrov-Werte zeigt. Während die Ritz-Werte – wie auch alle Eigenwerte λ_i – gemäß Satz 41 stets im Rechteck R liegen, trifft dies bei den Petrov-Werten auch in diesem Falle nicht zu.

Für die großdimensionale Matrix mit $n = 50\,000$ Zeilen wurden die Ritz-Werte θ_i und damit die Spektralradien

$$\varrho(H_m) \;:=\; \max_{i=1(1)m} |\theta_i|$$

der $m \times m$-Matrizen H_m für $m = 5, 10, 15, 20$ berechnet. Abb. 17 stellt deren Entwicklung dar. Die entsprechenden Spektralradien $\varrho(T_m)$ der Tridiagonalmatrizen T_m gemäß (52) aus

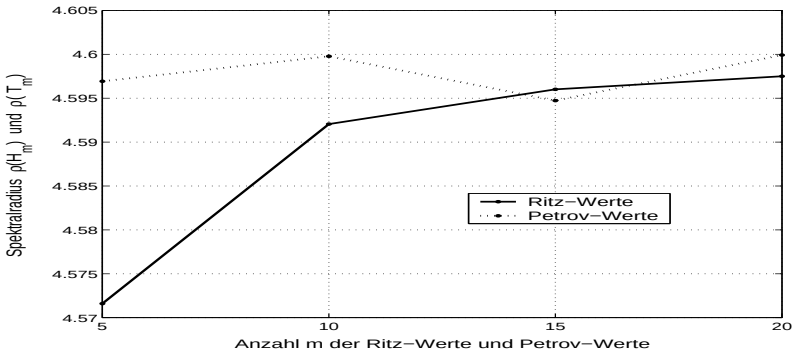

Abbildung 17: Spektralradien der Ritz- und Petrov-Werte für $n = 50\,000$

den Petrov-Werten sind ebenfalls in diesem Fall eingezeichnet; allerdings sind sie oft weniger geeignet, um eine gute Schätzung des unbekannten Spektralradius $\varrho(A)$ zu gewinnen (vgl. dazu Abb. 14). In Tabelle 4 wird die Entwicklung der Ritz-Werte für die Matrix-

m	$n = 100\,000$		$n = 200\,000$	
	$\varrho(H_m)$	Zeit (sec)	$\varrho(H_m)$	Zeit (sec)
5	4.57149633408765	0.38	4.57147234650181	0.75
10	4.59164523171915	0.89	4.59158223657048	1.79
15	4.59613696898894	1.59	4.59614671008193	3.23
20	4.59774511348431	2.44	4.59769375095428	4.85
25	4.59846615499577	3.44	4.59844448385815	7.64
30	4.59890091633856	4.65	4.59893953530372	10.08

Tabelle 4: Spektralradius $\varrho(H_m)$ zu m Ritz-Werten von A

dimensionen $n = 100\,000$ und $n = 200\,000$ zusammen mit den Rechenzeiten in Sekunden[3] dargestellt. Weitere verallgemeinerte Problemstellungen und moderne Verfahren zur Eigenwert-Approximation findet man in der Monografie [4].

[3]Fujitsu Siemens Computers Mobile, Intel(R) Pentium 4 CPU 1.60 GHz

5 Zusammenfassung

Eigenwerte und -vektoren großer Matrizen müssen mit iterativen Verfahren ermittelt werden. Das trifft sowohl auf das QR-Verfahren für das vollständige Eigenwertproblem, als auch auf die Varianten der Vektoriteration und die Krylov-Verfahren für das partielle Eigenwertproblem zu.

1. Die *Methode der Vektoriteration*, auch Potenzmethode genannt, bestimmt unter geeigneten Voraussetzungen den betragsgrößten (dominanten) Eigenwert λ_1 und den zugehörigen Eigenvektor. Mit dem Rayleigh-Quotienten kann diese Näherung weiter verbessert werden.
2. Mittels *inverser Iteration und Spektralverschiebung* kann ein beliebiger Eigenwert λ_i ermittelt werden, falls eine grobe Näherung λ_0 dafür vorliegt. Das trifft oft bei parameterabhängigen Eigenwertproblemen zu, z.B. bei der Parameterfortsetzung dynamischer Systeme.
3. *Orthogonale Ähnlichkeitstransformationen* haben sich zur Lösung des vollständigen Eigenwertproblems besonders bewährt. Mit Hilfe orthogonaler Matrizen, deren Anwendung die Kondition nicht verschlechtert, gelingt für jede $n \times n$-Matrix A die Transformation in obere Quasi-Dreiecksform (auch reelle Schur-Form genannt) mit leicht bestimmbaren Eigenwerten.
4. Der *QR-Algorithmus* gewinnt eine orthogonale Ähnlichkeitstransformation mittels QR-Zerlegung der Matrix A und Bildung des Produktes $A' = R \cdot Q$. Iterative Wiederholung dieses Prozesses führt – zumindest im Falle reeller und verschiedener Eigenwerte – auf eine obere Dreiecksmatrix mit den gesuchten Eigenwerten.
5. Durch geeignete *Spektralverschiebungen* lässt sich überlineare Konvergenz des QR-Algorithmus erreichen. Weitere Konvergenzverbesserungen, wie die vorherige Transformation auf Hessenbergform und der QR-Doppelschritt, machen den QR-Algorithmus zu einem leistungsfähigen Verfahren für die Lösung des vollständigen Eigenwertproblems.
6. Die *QR-Zerlegung* einer $n \times n$-Matrix A in eine orthogonale (Q) und eine obere Dreiecksmatrix (R) lässt sich effizient und numerisch stabil mittels Householder-Spiegelungen durchführen. Sie ist nicht nur Hauptbestandteil des QR-Algorithmus, sondern kann auch zur numerisch stabilen Lösung normal- und überbestimmter linearer Gleichungssysteme genutzt werden.
7. *Krylov-Unterraum-Methoden* sind auch für Eigenwert-Approximationen sehr großer Matrizen wesentlich. Mit ihrer Hilfe gewinnt man m komplexe Werte, die den Rand des numerischen Wertebereichs – die Menge aller Rayleigh-Quotienten – annähern und damit grob das Spektrum $\varrho(A)$ von A eingrenzen.
8. Die *Ritz-Werte* lassen sich mittels des (modifizierten) Arnoldi-Verfahrens, die *Petrov-Werte* mittels des Bi-Lanczos-Verfahrens ermitteln. Sie approximieren das Spektrum von A in vertretbarer Rechenzeit. Schnelle Algorithmen für großdimensionale Eigenwertprobleme werden zunehmend in praktischen Problemstellungen eingesetzt.
9. Eine *Darstellung moderner Eigenwertverfahren* findet man in der Übersicht [4], während die Spezialliteratur [34, 20, 13, 25, 32] weitere bekannte Verfahren enthält. MATLAB bietet mit `eig(A)` und `schur(A)` die Lösung des vollständigen Eigenwertproblems

auf Basis des QR-Algorithmus an. Für das partielle Eigenwertproblem sollten eigene MATLAB-Funktionen entwickelt werden.

Literatur

[1] Allgower, E. L.; Georg, K.: *Numerical Continuation Methods*, Springer Verlag, Berlin 1990

[2] Carroll, T.; Pecora, L. (Hrsg.): *Nonlinear Dynamics in Circuits. Yoshinaga, T.; Kawakami, H., S.89-120*, World Scientific, Singapore 1995

[3] Ciarlet, P.G.; Lions, J.L. (Hrsg.): *Handbook of Numerical Analysis, Vol. V. Allgower, E.L.; Georg, K.: Numerical Path Following, S. 3-207*, Elsevier, Amsterdam 1997

[4] Ciarlet, P.G.; Lions, J.L. (Hrsg.): *Handbook of Numerical Analysis, Vol. VIII. Van der Vorst, H.A.: Computational Methods for Large Eigenvalue Problems, S.3-179*, Elsevier, Amsterdam 2002

[5] Deuflhard, P.; Hohmann, A.: *Numerische Mathematik I*, 2. Auflage, W. de Gruyter, Berlin 1993

[6] Deuflhard, P.: *Newton Methods for Nonlinear Problems*, Springer Verlag, Berlin 2004

[7] Eisenstat, S.C.; Walker, H.F.: *Choosing the Forcing Terms in an Inexact Newton Method*, SIAM J. Scientific Comput. 17, No.1, S.16-32

[8] Engeln-Müllges, G.; Reutter, F.: *Numerik–Algorithmen mit ANSI C-Programmen*, Wissenschaftsverlag, Mannheim 1993

[9] Golub, G.; Loan, C.V.: *Matrix Computations*, John Hopkins University Press, Baltimore 1989

[10] Gramlich, G.; Werner, W.: *Numerische Mathematik mit MATLAB*, dpunkt.verlag GmBH, Heidelberg 2000

[11] Gustafson, K.E.; Rao, D.K.M.: *Numerical Range. The Field of Values of Linear Operators and Matrices*, Springer Verlag, New York 1997

[12] Hackbusch, W.: *Iterative Lösung großer schwachbesetzter Gleichungssysteme*, B.G.Teubner, Stuttgart 1991

[13] Hanke-Bourgeois, M.: *Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens*, B.G. Teubner, Stuttgart 2002

[14] Hermann, M.: *Numerische Mathematik*, Oldenbourg Verlag, München 2001

[15] Hoffmann, A.; Marx, B.; Vogt, W.: *Mathematik für Ingernieure. Lineare Algebra, Analysis – Theorie und Numerik*, Pearson Stuidium, München, erscheint 2005

[16] Isaacson, E.; Keller, H. B.: *Analyse numerischer Verfahren.* Verlag Harry Deutsch, Frankfurt 1972.

[17] Kelley, C.T.: *Iterative Methods for Linear and Nonlinear Equations*, SIAM Publications, Philadelphia 1995

[18] Kosmol, P.: *Methoden zur numerischen Behandlung nichtlinearer Gleichungen und Optimierungsaufgaben*, B.G.Teubner, Stuttgart 1993

[19] Latussek, P.; Seidel, H.U.; Vogt, W.; Wagner, E.: *Lehr-und Übungsbuch Mathematik. Band V*, Fachbuchverlag, Leipzig–Köln 1992

[20] Maess, G.: *Vorlesungen über numerische Mathematik. Band 1 und 2*, Akademie – Verlag, Berlin 1984

[21] Meister, A.: *Numerik linearer Gleichungssysteme*, Vieweg–Verlag, Braunschweig 1999

[22] Ortega, J.M.; Rheinboldt, W.C.: *Iterative Solution of Nonlinear Equations in Several Variables.* Academic Press, New York 1970

[23] Philippow, E.S.; Büntig, W.G.: *Analyse nichtlinearer dynamischer Systeme der Elektrotechnik.* Carl Hanser Verlag, München 1992

[24] Piegl, L.; Tiller, W.: *The NURBS Book.* 2nd edition, Springer Verlag, Berlin 1997

[25] Quarteroni, A.; Sacco, R.; Saleri, F.: *Numerische Mathematik. Band 1 und 2*, Springer Verlag, Berlin 2002

[26] Rheinboldt, W.C.: *Methods for Solving Systems of Nonlinear Equations.* 4th ed., SIAM Publications, Philadelphia 1994

[27] Saad, Y.: *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston 1995

[28] Schwarz, H. R.: *Numerische Mathematik.* B.G. Teubner, Stuttgart 1988

[29] Schwetlick, H.: *Numerische Lösung nichtlinearer Gleichungen.* Deutscher Verlag der Wissenschaften, Berlin 1979

[30] Schwetlick, H.; Kretzschmar, H.: *Numerische Verfahren für Naturwissenschaftler und Ingenieure.* Fachbuchverlag, Leipzig 1991

[31] Stoer, J.: *Numerische Mathematik 1*, 7., neubearbeitete und erweiterte Auflage, Springer Verlag, Berlin 1994

[32] Törnig, W.; Spellucci, P.: *Numerische Mathematik für Ingenieure und Physiker. Band 1 und 2.* 2. Auflage, Springer Verlag, Berlin 1988.

[33] Trefethen, L.N.; Bau, D.: *Numerical Linear Algebra*, SIAM Publications, Philadelphia 1997

[34] Wilkinson, J.H.: *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford 1992