7. KRYLOV-VERFAHREN ZUR LÖSUNG LINEARER GLEICHUNGSSYSTEME

In Kapitel 3 haben wir mit der Gauß-Elimination und der QR-Zerlegung verschiedene direkte Verfahren zur Lösung eines linearen Gleichungssystems

$$Ax = b, \qquad A \in \mathbb{C}^{n,n} \text{ nicht singulär}, \quad b \in \mathbb{C}^n \tag{7.1}$$

kennengelernt. Diese Verfahren hatten einen Aufwand von $O(n^3)$ Operationen und einen Speicheraufwand von $O(n^2)$. In der folgenden Tabelle sind die ungefähre Zeit und der Speicherbedarf für die Lösung von (7.1) mit Gauß-Elimination für verschiedene n auf einem Rechner mit Intel Dual Core 3 GHz-Prozessor aufgelistet.

n	Zeit	Speicher
10^2	< 1 msec	80 KByte
10^3	0.129 sec	8 MByte
10^4	1.25 min	800 MByte
10^5	20.8 Stunden	80 GByte
10^6	2.38 Jahre	8 TByte

Offensichtlich sind also direkte Verfahren zur Lösung linearer Gleichungssysteme mit Dimension $n \gg 10^3$, vielleicht 10^4, nicht mehr anwendbar. In der Praxis resultieren lineare Gleichungssysteme jedoch oft aus der Ortsdiskretisierung eines partiellen Differentialoperators. Insbesondere in zwei oder drei Raumdimensionen werden die Probleme dann schnell sehr groß. A ist jedoch in diesen Anwendungen meist nicht voll, sondern **dünn besetzt**, d. h. in jeder Zeile und Spalte von A gibt es nur wenige von Null verschiedene Elemente. Es ist daher billig, einen Vektor mit A zu multiplizieren, jedoch teuer, eine Zerlegung von A zu berechnen, denn diese wird im Allgemeinen voll besetzt sein.

Zur Lösung solcher Probleme verwendet man iterative Verfahren. Wir stellen hier die besonders wichtige Klasse der Krylov-Verfahren vor. Es wird sich herausstellen, dass es fundamentale Unterschiede zwischen Gleichungssystemen mit Hermiteschen Matrizen A und dem allgemeinen Fall gibt. Wir werden in diesem Kapitel jedoch mit einer allgemeinen Konstruktion und Analyse beide Fälle mit denselben Techniken behandeln.

Ohne Einschränkung nehmen wir an, dass der Startwert $x_0 = 0$ ist, denn für einen beliebigen Startwert x_0 kann man das Verfahren mit Startwert Null auf das äquivalente Problem $A(x - x_0) = b - Ax_0$ anwenden.

7.1 Krylov-Verfahren, Arnoldi-Prozess

Erinnerung: Zu $A \in \mathbb{C}^{n,n}$ und $b \in \mathbb{C}^n$, $b \neq 0$ heißt

$$\mathcal{K}_m(A, b) = \operatorname{span}\{b, Ab, \ldots, A^{m-1}b\}$$

der m-te **Krylov-Raum** bezüglich A und b. Eine äquivalente Definition ist

$$\mathcal{K}_m(A,b) = \{p(A)b \mid p \in \mathcal{P}_{m-1}\},$$

wobei $\mathcal{P}_{m-1}$ wie früher den Raum aller Polynome vom Grad höchstens $m-1$ bezeichnet. Daher heißen Krylov-Unterraum-Verfahren auch polynomiale iterative Verfahren. Krylov-Verfahren zur Lösung von (7.1) sind attraktiv, wenn Matrix-Vektormultiplikationen mit A billig sind, denn dies ist die einzige Operation, in die die Matrix A eingeht. In Anwendungen bei partiellen Differentialgleichungen kostet eine Matrix-Vektormultiplikation meist nur $O(n)$ Operationen, im Vergleich zu $O(n^2)$ bei voll besetzten Matrizen.

Sei $V_m = \begin{bmatrix} v_1 & \dots & v_m \end{bmatrix} \in \mathbb{C}^{n,m}$ eine Basis von $\mathcal{K}_m = \mathcal{K}_m(A,b)$. Dann können wir eine approximative Lösung als

$$x_m = V_m y_m, \qquad y_m \in \mathbb{C}^m. \tag{7.2}$$

schreiben. Es stellen sich hier zwei Fragen:

- Wie bestimmt man eine Basis V_m von $\mathcal{K}_m$?
- Wie charakterisiert man den Koeffizientenvektor y_m?

Wir beginnen mit der Konstruktion einer Basis von $\mathcal{K}_m$. Wegen $\mathcal{K}_m \subseteq \mathcal{K}_{m+1}$ genügt es, in jedem Schritt einen neuen Basisvektor zu berechnen. Solche geschachtelten Basen können wie folgt charakterisiert werden:

Lemma 7.1. *Ist $V_j \in \mathbb{C}^{n,j}$ eine Basis von $\mathcal{K}_j$ für $j = 1, \dots, m+1$, wobei $V_j = \begin{bmatrix} V_{j-1} & v_j \end{bmatrix}$, dann gibt es eine eindeutig bestimmte, nicht reduzierte obere Hessenberg-Matrix $\widetilde{H}_m = (h_{ij}) \in \mathbb{C}^{m+1,m}$, so dass*

$$AV_m = V_{m+1}\widetilde{H}_m. \tag{7.3}$$

Zur Erinnerung: Eine Matrix $H = (h_{ij})$ hat **obere Hessenberg-Form** genau dann, wenn $h_{ij} = 0$ für alle $i \geq j+2$ gilt. Eine Hessenberg-Matrix H ist **nicht reduziert**, wenn $h_{j+1,j} \neq 0$ für alle j gilt.

Beweis. Es sei $j \leq m$. Nach Definition von $\mathcal{K}_j$ ist $Av_j \in \mathcal{K}_{j+1}$. Da V_{j+1} eine Basis von $\mathcal{K}_{j+1}$ ist, gibt es eindeutige Koffizienten $h_{1,j}, \dots, h_{j+1,j} \in \mathbb{C}$, so dass

$$Av_j = \sum_{i=1}^{j+1} h_{ij} v_i.$$

Ferner ist $h_{j+1,j} \neq 0$, da $Av_j \in \mathcal{K}_{j+1} \setminus \mathcal{K}_j$, die Hessenberg-Matrix $\widetilde{H}_m$ ist also nicht reduziert. □

Gleichung (7.3) ist äquivalent zu

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T, \tag{7.4}$$

wobei

$$H_m = \begin{bmatrix} I_m & 0 \end{bmatrix} \widetilde{H}_m \in \mathbb{C}^{m,m} \qquad \text{oder} \qquad \widetilde{H}_m = \left[\begin{array}{c|c} H_m & \\ \hline 0 & h_{m+1,m} \end{array}\right].$$

Welche Basis eignet sich nun für ein Krylov-Verfahren? Offensichtlich ist die triviale Basis $\{A^j b\}_{j=0}^{m-1}$ nicht geeignet, denn die Vektoren werden numerisch linear abhängig. (Die Folge

$\{A^j b\}_j$ kennen wir ja bereits aus der Potenzenmethode zur Berechnung eines Eigenvektors von A.) Wir nehmen an, wir hätten bereits eine Orthonormalbasis $\{v_1, \ldots, v_m\}$ von $\mathcal{K}_m$ berechnet. Für $m = 1$ ist dies trivial. Nach (7.4) (letzte Spalte) ist der neue Basisvektor v_{m+1} durch

$$\widetilde{v}_{m+1} = Av_m - \sum_{i=1}^{m} h_{im} v_i, \qquad v_{m+1} = \widetilde{v}_{m+1}/h_{m+1,m}$$

gegeben. Die Forderung $v_j^H v_{m+1} = 0$ führt wegen $v_j^H v_k = 0$ für $j \neq k$, $j, k \leq m$ auf

$$h_{jm} = v_j^H A v_m$$

und die Normierungsbedingung $\|v_{m+1}\| = 1$ auf $h_{m+1,m} = \|\widetilde{v}_{m+1}\|$. Dies ergibt Algorithmus 7.1.

Algorithmus 7.1 Arnoldi-Algorithmus, 1951

Zu gegebenen $A \in \mathbb{C}^{n,n}$, $b \in \mathbb{C}^n$, berechne $\beta = \|b\| > 0$
$v_1 = b/\beta$;
for $m = 1, 2, \ldots$ **do**
  for $j = 1, \ldots, m$ **do**
    $h_{j,m} = v_j^H A v_m$
  end for
  $\widetilde{v}_{m+1} = Av_m - \sum_{j=1}^{m} h_{j,m} v_j$
  $h_{m+1,m} = \|\widetilde{v}_{m+1}\|$
  $v_{m+1} = \widetilde{v}_{m+1}/h_{m+1,m}$
end for

Der Arnoldi-Prozess ähnelt stark dem bekannten Gram-Schmidt-Orthogonalisierungsverfahren. Der einzige Unterschied ist, dass im m-ten Schritt nicht $A^m b$ sondern Av_m zur Orthonormalisierung verwendet wird. Da beide Vektoren in $\mathcal{K}_{m+1} \setminus \mathcal{K}_m$ liegen, jedenfalls solange $\mathcal{K}_{m+1} \neq \mathcal{K}_m$, sind die Verfahren mathematisch äquivalent. Aus Stabilitätsgründen ist das Arnoldi-Verfahren zu bevorzugen. Ebenfalls mathematisch äquivalent, aber numerisch noch stabiler, ist der *modifizierte Gram-Schmidt* Prozess, bei dem die `for`-Schleife über j und die darauf folgende Definition von $\widetilde{v}_{m+1}$ durch

$\widetilde{v}_{m+1} = Av_m$
for $j = 1, \ldots, m$ **do**
  $h_{j,m} = v_j^H \widetilde{v}_{m+1}$
  $\widetilde{v}_{m+1} = \widetilde{v}_{m+1} - h_{j,m} v_j$
end for

ersetzt wird. Die Kosten stimmen für beide Varianten überein.

Einige Eigenschaften des Arnoldi-Verfahrens sind:

Lemma 7.2. *Es seien V_m und $\widetilde{H}_m$ die Matrizen aus dem Arnoldi-Verfahren. Dann gilt:*

(a) $V_m^H V_m = I_m$

(b) $AV_m = V_{m+1}\widetilde{H}_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T$,

(c) $V_m^H A V_m = H_m$,

(d) $V_{m+1}^H A V_m = \widetilde{H}_m$,

(e) *Für $A = A^H$ ist $H_m = H_m^H$ tridiagonal.*

Beweis. (a), (b) sind nach Konstruktion klar.
(c) folgt aus (b) durch Multiplikation mit V_m^H von links, wobei man $V_m^H v_{m+1} = 0$ beachtet.
(d) folgt aus (b) durch Multiplikation mit V_{m+1}^H von links.
(e) folgt direkt aus (c). □

In der folgenden Tabelle haben wir Rechen- und Speicheraufwand für das Arnoldi-Verfahren zusammengestellt, wobei für Hermitesches A Lemma 7.2(e) ausgenutzt wurde.

Berechnung von	A beliebig	A Hermitesch
Av_m	1 Matrix-Vektormult.	1 Matrix-Vektormult.
$h_{j,m}$	m Skalarprodukte	1 Skalarprodukt ($h_{m-1,m} = \overline{h_{m,m-1}}$)
$\widetilde{v}_{m+1}$	m SAXPYs	2 SAXPYs
$h_{m+1,m}$	1 Skalarprodukt	1 Skalarprodukt
v_{m+1}	n Divisionen	n Divisionen
Speicher	$m+1$ Vektoren	3 Vektoren

Der wesentliche Unterschied zwischen dem Hermiteschen (symmetrischen) und dem nicht-Hermiteschen (nichtsymmetrischen) Fall ist, dass für Hermitesches A Rechen- und Speicheraufwand pro Iteration konstant sind, während sie für allgemeines A mit der Anzahl der Iterationen wachsen. Für sehr große Probleme kann man den Arnoldi-Prozess daher nicht beliebig lange laufen lassen.

Der Arnoldi-Algorithmus 7.1 bricht ab, wenn $h_{m+1,m} = 0$ bzw. $AV_m = V_m H_m$ ist. Dann enthält der bisher konstruierte Krylov-Raum bereits die exakte Lösung $\widehat{x} = A^{-1}b$:

Lemma 7.3. *Ist $AV_m = V_m H_m$, dann gibt es ein $y_m \in \mathbb{C}^m$, so dass $\widehat{x} = A^{-1}b = V_m y_m$.*

Beweis. Wegen $\mathcal{R}(V_1) = \operatorname{span}\{b\}$ ist $\beta v_1 = b$, $\beta \in \mathbb{C}$. Ferner ist jeder Eigenwert von H_m auch Eigenwert von A, für die Spektren gilt also $\lambda(H_m) \subseteq \lambda(A)$, denn ist $\mu \in \lambda(H_m)$ mit Eigenvektor y, dann gilt

$$AV_m y = V_m H_m y = \mu V_m y.$$

Wegen $y \neq 0$ ist $V_m y$ Eigenvektor von A zum Eigenwert μ. Da nach Voraussetzung $0 \notin \lambda(A)$, ist insbesondere H_m nicht singulär und wir erhalten

$$AV_m y_m = V_m H_m y_m = \beta v_1 = b = V_m(\beta e_1),$$

wenn y_m die eindeutige Lösung von $H_m y_m = \beta e_1$ ist. □

In der Praxis wird die in Lemma 7.3 beschriebene Situation nicht auftreten. Wie soll man dann den Koeffizientenvektor y_m wählen? Die beste Approximation y_m würde den Fehler $f_m = \widehat{x} - x_m$, $\widehat{x} = A^{-1}b$ in einer gegebenen Norm $\|\cdot\|$ minimieren:

$$\|f_m\| = \min_{x \in \mathcal{K}_m} \|\widehat{x} - x\|.$$

Leider kann man den Fehler f_m nicht berechnen, solange man die exakte Lösung nicht kennt. Alternativ kann man approximative Lösungen von (7.1) durch Eigenschaften des Residuenvektors

$$r_m = b - Ax_m$$

charakterisieren. Offensichtlich ist $r_m = 0$ genau dann, wenn $x_m = \widehat{x} = A^{-1}b$. Aus $r_m = A(\widehat{x} - x_m) = Af_m$ und $f_m = A^{-1}r_m$ ergeben sich die Ungleichungen

$$\frac{1}{\kappa(A)} \frac{\|r_m\|}{\|r_0\|} \leq \frac{\|f_m\|}{\|f_0\|} \leq \kappa(A) \frac{\|r_m\|}{\|r_0\|}.$$

Die relative Residuennorm und die relative Fehlernorm können also stark differieren, wenn die Konditionszahl von A groß ist. Daher ist es ein wenig riskant, nur die Residuennorm zu betrachten. Ähnliches hatten wir ja auch schon beim Newton-Verfahren in Abschnitt 4.2 beobachtet. Dass dieser Ansatz sich trotzdem durchgesetzt hat, liegt wohl daran, dass das Residuum leicht zu berechnen ist.

Aus (7.2) und Lemma 7.2(b) erhalten wir

$$\begin{aligned} r_m &= b - AV_m y_m \\ &= \beta v_1 - V_{m+1}\widetilde{H}_m y_m \\ &= V_{m+1}(\beta e_1 - \widetilde{H}_m y_m). \end{aligned} \tag{7.5}$$

Dies führt auf

$$\|r_m\| = \|V_{m+1}(\beta e_1 - \widetilde{H}_m y_m)\| \leq \|V_{m+1}\| \, \|\beta e_1 - \widetilde{H}_m y_m\|.$$

Die letzte Ungleichung ist eine Gleichung, wenn V_{m+1} unitär ist, also zum Beispiel wenn V_{m+1} aus dem Arnoldi-Prozess resultiert.

Wir werden hier zwei verschiedene Ansätze zur Charakterisierung von y_m diskutieren:

(M) **Minimiere** $\|\beta e_1 - \widetilde{H}_m y\|$ über alle $y \in \mathbb{C}^m$ (bei reellen Problemen über alle $y \in \mathbb{R}^m$). Dies führt auf ein lineares Ausgleichsproblem der Dimension $(m+1) \times m$. $\widetilde{H}_m$ ist nicht reduziert und hat daher vollen Rang m. Nach Lemma 3.24 ist das lineare Ausgleichsproblem eindeutig lösbar und die Lösung nach Satz 3.22 durch die Normalengleichungen gegeben:

$$y_m = \beta(\widetilde{H}_m^H \widetilde{H}_m)^{-1} \widetilde{H}_m^H e_1 = \beta \widetilde{H}_m^+ e_1,$$

wobei $\widetilde{H}_m^+ := (\widetilde{H}_m^H \widetilde{H}_m)^{-1} \widetilde{H}_m^H$ die **Pseudoinverse** von $\widetilde{H}_m$ bezeichnet.

(G) Wähle y_m so, dass r_m orthogonal zu einem geeigneten Unterraum $\mathcal{W}_m$ der Dimension m ist. Diese Bedingung heißt für $\mathcal{W}_m = \mathcal{K}_m$ **Galerkin-Bedingung** und sonst **Petrov-Galerkin-Bedingung**.

In Lemma 7.3 haben wir gezeigt, dass $AV_m = V_m H_m$ eine hinreichende Bedingung dafür ist, dass der Krylov-Raum $\mathcal{K}_m$ die exakte Lösung enthält. Das folgende Lemma zeigt, dass die Charakterisierungen (M) oder (G) in diesem Fall auch tatsächlich die Lösung liefern.

Satz 7.4. *Ist $AV_m = V_m H_m$ und ist W_m eine Basis von $\mathcal{W}_m$ mit $W_m^H V_m$ nicht singulär, dann ist $x_m = V_m y_m$, wobei y_m durch* (M) *oder* (G) *charakterisiert ist, die exakte Lösung, d. h. $x_m = \widehat{x} = A^{-1}b$.*

Beweis. Wegen (7.5) und $h_{m+1,m} = 0$ ist (G) äquivalent zu

$$W_m^H r_m = W_m^H V_{m+1}(\beta e_1 - \widetilde{H}_m y_m) = W_m^H V_m(\beta e_1 - H_m y_m) = 0.$$

Da $W_m^H V_m$ nicht singulär ist, folgt hieraus $H_m y_m = \beta e_1$, also

$$Ax_m = AV_m y_m = V_m H_m y_m = \beta v_1 = V_m(\beta e_1) = b.$$

Dasselbe y_m liefert $\|\beta e_1 - \widetilde{H}_m y_m\| = 0$, so dass y_m auch (M) genügt. □

Insbesondere liefern alle Krylov-Verfahren, deren Iterierte durch (M) oder (G) bestimmt sind, nach höchstens n Schritten die exakte Lösung.

7.2 Verfahren basierend auf dem Arnoldi-Prozess

In diesem Abschnitt betrachten wir Verfahren, die eine Orthonormalbasis V_{m+1} verwenden, also auf dem Arnoldi-Prozess basieren. Die durch (M) charakterisierte Lösung erfüllt dann eine Minimierungsbedingung, denn sie minimiert die Residuennorm über $\mathcal{K}_m$:

Satz 7.5. *Ist V_{m+1} unitär und $y_m = \beta \widetilde{H}_m^+ e_1$ die Lösung von* (M)*, dann erfüllt $x_m = V_m y_m$*

$$\|r_m\| = \|b - Ax_m\| \le \|b - Ax\| \qquad \text{für alle } x \in \mathcal{K}_m.$$

Beweis. Die Behauptung folgt aus $\|r_m\| = \min_{y\in\mathbb{C}^m} \|\beta e_1 - \widetilde{H}_m y\|$ und $\mathcal{K}_m = \mathcal{R}(V_m)$. □

Bemerkung. Wegen $\|r_m\| = \|\beta e_1 - \widetilde{H}_m y_m\|$ ist es nicht notwendig, $x_m = V_m y_m$ explizit zu berechnen, um die Residuennorm zu kontrollieren. Allerdings muss man beachten, dass diese Gleichung wegen Rundungsfehlern nicht exakt gilt. Da aber für sehr große n die Berechnung von x_m relativ teuer ist, berechnet man das Residuum erst in der Nähe der Toleranzgrenze explizit.

Das resultierende Verfahren (Arnoldi + (M)) heißt **generalized minimal residual method** (GMRES). Eine effiziente Implementierung haben Saad and Schultz 1986 vorgestellt. Sie basiert auf einer QR-Zerlegung von $\widetilde{H}_m$:

$$Q_m \widetilde{H}_m = \widetilde{R}_m = \begin{bmatrix} R_m \\ 0 \end{bmatrix},$$

wobei $Q_m \in \mathbb{C}^{m+1,m+1}$ eine unitäre und $R_m \in \mathbb{C}^{m,m}$ eine obere Dreiecksmatrix ist. R_m ist nicht singulär, da $\widetilde{H}_m$ vollen Rang m hat. Die QR-Zerlegung verwendet man wie in Abschnitt 3.6. Es ist

$$\|\beta e_1 - \widetilde{H}_m y\| = \|\beta Q_m e_1 - \widetilde{R}_m y\| = \|\widetilde{q}_m - \widetilde{R}_m y\| = \left\| \begin{bmatrix} q_m \\ \rho_m \end{bmatrix} - \begin{bmatrix} R_m \\ 0 \end{bmatrix} y \right\|$$

minimal für $y_m = R_m^{-1} q_m$ und das Minimum ist $\|\beta e_1 - \widetilde{H}_m y_m\| = |\rho_m|$. Die QR-Zerlegung kann wie in Abschnitt 3.7 beschrieben durch eine Folge von Householder-Transformationen berechnet werden. Die Hessenberg-Form kann dabei ausgenutzt werden, so dass die relevanten Teilmatrizen der Householder-Matrizen nur die Dimension 2×2 haben. Die Implementierung von Saad und Schultz verwendet jedoch komplexe Givens-Rotationen

$$G_{j,m} = \begin{bmatrix} I_{j-1} & & & \\ & c_j & -\overline{s_j} & \\ & s_j & c_j & \\ & & & I_{m-j-1} \end{bmatrix} \in \mathbb{C}^{m,m}, \quad c_j \in \mathbb{R}, s_j \in \mathbb{C}, \quad c_j^2 + |s_j|^2 = 1.$$

Wir verzichten auf die Details.

Wir erinnern daran, dass $|\rho_m| = \|r_m\|$ die Residuennorm der GMRES-Iterierten ist. Ist also $|\rho_m|$ klein genug, dann lösen wir $R_m y_m = q_m$ durch Rückwärtselimination und berechnen daraus die GMRES-Iterierte $x_m = V_m y_m$.

Falls der Speicher nicht ausreicht, GMRES bis zur Konvergenz laufen zu lassen, muss man zwischendurch neu starten. Diese Möglichkeit ist in Algorithmus 7.2 vorgesehen. Der Neustart-Index k kann abhängig vom verfügbaren Speicher so klein wie nötig gewählt werden. Die Originalversion von GMRES kann als GMRES(∞) interpretiert werden, wobei die letzte Zeile entfällt. Ein gravierender Nachteil von Neustarts ist, dass die Minimierungseigenschaft

aus Satz 7.5 verloren geht und häufig die Konvergenz erheblich verlangsamt wird. Ein Beispiel mit

$$A = T \otimes I_{30} + I_{30} \otimes T^T, \qquad T = \text{tridiag}(-2, 3, 1) \tag{7.6}$$

und einem Zufallsvektor b ist in Abbildung 7.1 dargestellt.

Algorithmus 7.2 GMRES(k), GMRES mit Restarts

Wähle $x_0 \in \mathbb{C}^n$, setze $r_0 = b - Ax_0$;
Initialisiere $m = 0$, $\beta = \|r_0\|$, $v_1 = r_0/\beta$;
repeat
  while noch nicht konvergent und $m \leq k$ {Konvergenzkriterium} **do**
    {Arnoldi-Prozess}
    $m = m + 1$
    for $j = 1, \ldots, m$ **do**
      $h_{j,m} = v_j^H A v_m$;
    end for
    $\widetilde{v}_{m+1} = Av_m - \sum_{j=1}^{m} h_{j,m} v_j$;
    $h_{m+1,m} = \|\widetilde{v}_{m+1}\|$;
    $v_{m+1} = \widetilde{v}_{m+1}/h_{m+1,m}$;
    Berechne R_m und $\widetilde{q}_m$;
  end while
  Berechne die Lösung $y_m = \beta \widetilde{H}_m^+ e_1$ von $\|\beta e_1 - \widetilde{H}_m y\| = \min$;
  Setze $x_m = x_0 + V_m y_m$;
  {Neustart}
  Berechne $r_m = b - Ax_m$; Setze $x_0 = x_m$, $\beta = \|r_m\|$, $v_1 = r_m/\beta$; $m = 0$
until Konvergenz erreicht

Das Arnoldi-Verfahren kombiniert mit der Galerkin-Bedingung (G) wurde unter dem Namen **Full Orthogonalization Method** (FOM) von Saad 1981 vorgeschlagen. Auch diese Variante kann neugestartet werden, wenn sie zu teuer wird. Sie heißt dann FOM(k).

Während eine Lösung von (M) für jedes m existiert, ist das für (G) nicht mehr richtig; nicht einmal im Fall $\mathcal{W}_m = \mathcal{K}_m$.

Satz 7.6. *Sei V_{m+1} unitär und $\mathcal{W}_m = \mathcal{K}_m$. Dann existiert eine Lösung von* (G) *genau dann, wenn H_m nicht singulär ist. In diesem Fall ist $x_m = V_m y_m$, wobei $H_m y_m = \beta e_1$.*

Beweis. (G) mit $\mathcal{W}_m = \mathcal{K}_m$ und $W_m = V_m$ ist äquivalent zu

$$\begin{aligned} V_m^H r_m &= V_m^H V_{m+1}(\beta e_1 - \widetilde{H}_m y_m) \\ &= \beta e_1 - [\ I_m \quad 0\ ] \widetilde{H}_m y_m \\ &= \beta e_1 - H_m y_m. \end{aligned}$$

Daher erfüllt die Lösung, wenn sie existiert, $H_m y_m = \beta e_1$.

Angenommen, eine Lösung existiert und H_m ist singulär, d. h.

$$\text{Rang}\,([\ e_1 \quad H_m\ ]) = \text{Rang}\,(H_m) < m.$$

H_m ist eine nicht reduzierte obere Hessenberg-Matrix. Dann ist $[\ e_1 \quad H_m\ ]$ eine $m \times (m+1)$ obere Dreiecksmatrix mit Diagonalelementen 1 und $h_{j+1,j} \neq 0$, $j = 1, \ldots, m-1$, hat also den Rang m im Widerspruch zur Annahme. □

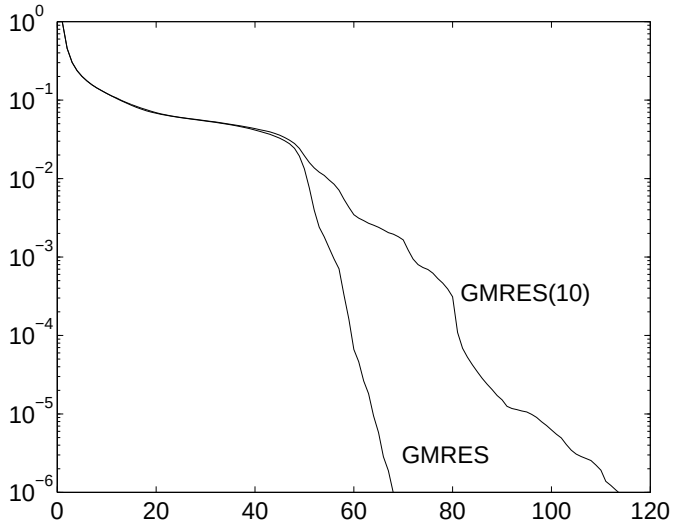

Abb. 7.1: Residuennormen über Iterationen von GMRES und GMRES(10).

Unter gewissen Voraussetzungen kann die Existenz aller Galerkin-Iterierten garantiert werden.

Lemma 7.7. *Ist V_m unitär, dann gilt $\mathcal{F}(H_m) \subseteq \mathcal{F}(A)$. Insbesondere folgt aus $0 \notin \mathcal{F}(A)$, dass H_m für alle m nicht singulär ist.*

Beweis. Nach Definition des Wertebereichs gilt

$$\begin{aligned} \mathcal{F}(H_m) &= \{y^H V_m^H A V_m y \mid y \in \mathbb{C}^m, \|y\| = 1\} \\ &= \{x^H A x \mid x \in \mathcal{K}_m(A, b), \|x\| = 1\} \\ &\subseteq \{x^H A x \mid x \in \mathbb{C}^n, \|x\| = 1\} = \mathcal{F}(A). \end{aligned}$$

Die zweite Aussage folgt aus $\lambda(H_m) \subseteq \mathcal{F}(H_m)$ nach Lemma 6.15(b). □

Bemerkung. Aus Satz 7.6 schließen wir für die Residuen der Galerkin-Iterierten

$$\begin{aligned} r_m &= V_{m+1}(\beta e_1 - \widetilde{H}_m y_m) \\ &= V_m(\beta e_1 - H_m y_m) - h_{m+1,m} v_{m+1} e_m^T y_m \\ &= -h_{m+1,m} v_{m+1} e_m^T y_m. \end{aligned} \tag{7.7}$$

Damit steht auch hier

$$\|r_m\| = h_{m+1,m} |e_m^T y_m|$$

zur Verfügung, ohne vorher x_m zu berechnen.

Der Arnoldi-Prozess wird für große Iterationszahlen m sehr teuer, während das cg-Verfahren für symmetrische und positiv definite Matrizen A in jedem Schritt gleich wenig kostet. Es stellt sich die Frage, ob es nicht möglich ist, auch für beliebiges A ein Verfahren mit folgenden Eigenschaften zu konstruieren:

(a) Minimierung des Fehlers (oder des Residuums) in einer festen Norm unabhängig vom Startwert und

(b) kurze Rekursionen (also gleicher Aufwand in jedem Schritt).

Die negative Antwort gaben Faber and Manteufel 1984 sowie Voevodin 1983: Es gibt kein Verfahren, welches gleichzeitig (a) und (b) für beliebiges A erfüllt. Solche Verfahren gibt es nur für spezielle Matrizen, insbesondere Hermitesche (symmetrische).

Bei einer der beiden Forderungen müssen wir also Abstriche machen. GMRES erfüllt (a) jedoch nicht (b), FOM erfüllt weder (a) noch (b). Im nächsten Abschnitt diskutieren wir das nichtsymmetrische Lanczos-Verfahren, mit dem man eine Basis konstruieren kann, die (b) erfüllt. Anschließend kombinieren wir diese Basis mit einer Charakterisierung der Koeffizientenvektoren y_m, die zu Iterierten führt, die (a) in abgeschwächter Form erfüllen.

7.3 Nichtsymmetrisches Lanczos-Verfahren

In diesem Abschnitt leiten wir eine Alternative zum Arnoldi-Verfahren her, die eine Basis von $\mathcal{K}_m(A,b)$ mit kurzen (3-Term) Rekursionen in jedem Schritt berechnet. Die Grundidee klingt abwegig: Obwohl schon die Berechnung einer Basis $\{v_j\}_j$ von $\mathcal{K}_m$ zu schwierig war, berechnet man nun eine weitere Basis $\{w_j\}_j$ eines Krylov-Raumes, jetzt aber bezüglich A^H statt A, $\mathcal{K}_m(A^H,c)$ für einen geeigneten Vektor $c \in \mathbb{C}^n$. Im Gegensatz zum Arnoldi-Verfahren verlangt man nicht die Orthogonalität der einzelnen Basis, sondern Biorthogonalität zwischen den beiden Basen:

$$w_j^H v_k = \begin{cases} 0, & \text{für } j \neq k, \\ \delta_k \neq 0, & \text{für } j = k, \end{cases} \tag{7.8}$$

oder

$$v_m \perp \mathcal{K}_{m-1}(A^H,c) \qquad \text{und} \qquad w_m \perp \mathcal{K}_{m-1}(A,b). \tag{7.9}$$

Wir werden zeigen, dass ein solches Paar von Basen mit kurzen Rekursionen konstruiert werden kann.

Da die Bedingung (7.9) für $m = 1$ leer ist, können wir annehmen, dass wir bereits Basisvektoren $v_1,\dots,v_m$ und $w_1,\dots,w_m$ kennen, für die

$$\begin{aligned} \operatorname{span}\{v_1,\dots,v_m\} &= \mathcal{K}_m(A,b), \\ \operatorname{span}\{w_1,\dots,w_m\} &= \mathcal{K}_m(A^H,c) \end{aligned}$$

und (7.8) für ein $m \geq 1$ gilt. Wegen Lemma 7.1 gibt es Koeffizienten $h_{jm} \in \mathbb{C}, j = 1,\dots,m$, so dass

$$\widetilde{v}_{m+1} = Av_m - \sum_{j=1}^{m} h_{jm} v_j, \qquad v_{m+1} = \widetilde{v}_{m+1}/\gamma_{m+1}.$$

Multiplikation mit w_l^H von links liefert wegen (7.8) für $k = m+1$

$$0 = w_l^H \widetilde{v}_{m+1} = w_l^H A v_m - h_{lm} w_l^H v_l = (A^H w_l)^H v_m - h_{lm} w_l^H v_l, \qquad \text{für} \quad l \leq m.$$

Aus $A^H w_l \in \mathcal{K}_{l+1}(A^H,c)$ schließen wir mit (7.9), dass $(A^H w_l)^H v_m = 0$ für $l+1 < m$ und daraus $h_{lm} = 0$ für $l < m-1$. Schreiben wir

$$\widetilde{v}_{m+1} = Av_m - \alpha_m v_m - \beta_m v_{m-1}, \qquad v_{m+1} = \widetilde{v}_{m+1}/\gamma_{m+1},$$

dann folgt

$$\alpha_m = \frac{w_m^H A v_m}{w_m^H v_m}, \qquad \beta_m = \frac{w_{m-1}^H A v_m}{w_{m-1}^H v_{m-1}}.$$

Analog erhalten wir aus

$$\widetilde{w}_{m+1} = A^H w_m - \widetilde{\alpha}_m w_m - \widetilde{\beta}_m w_{m-1}, \qquad w_{m+1} = \widetilde{w}_{m+1}/\widetilde{\gamma}_{m+1}$$

die Koeffizienten

$$\widetilde{\alpha}_m = \frac{v_m^H A^H w_m}{v_m^H w_m} = \overline{\alpha}_m, \qquad \widetilde{\beta}_m = \frac{v_{m-1}^H A^H w_m}{v_{m-1}^H w_{m-1}}.$$

Aus der Rekursion für w_m ergibt sich

$$A^H w_{m-1} = \widetilde{\gamma}_m w_m + \widetilde{\alpha}_{m-1} w_{m-1} + \widetilde{\beta}_{m-1} w_{m-2},$$

also $v_m^H A^H w_{m-1} = \widetilde{\gamma}_m \overline{\delta_m}$. Auf dieselbe Weise zeigt man $w_m^H A v_{m-1} = \gamma_m \delta_m$, woraus sich

$$\beta_m = \frac{\overline{\widetilde{\gamma}_m}}{\delta_{m-1}} \delta_m \qquad \text{und} \qquad \overline{\widetilde{\beta}_m} = \gamma_m \frac{\delta_m}{\delta_{m-1}} = \frac{\gamma_m}{\overline{\widetilde{\gamma}_m}} \beta_m \tag{7.10}$$

ergibt. Daraus resultiert Algorithmus 7.3

Algorithmus 7.3 Nichtsymmetrisches Lanczos-Verfahren, 1952

Wähle b, $c \in \mathbb{C}^n$ mit $\delta_1 = c^H b \neq 0$;
Setze $v_1 = b$, $w_1 = c$, $v_0 = w_0 = 0$ und $\beta_1 = \widetilde{\beta}_1 = 0$;
for $m = 1, 2, \ldots$ **do**
  $\alpha_m = w_m^H A v_m / \delta_m$, $\widetilde{\alpha}_m = \overline{\alpha_m}$;
  if $m > 1$ **then**
    $\beta_m = \overline{\widetilde{\gamma}_m} \delta_m / \delta_{m-1}$,
    $\widetilde{\beta}_m = \overline{\gamma_m \delta_m / \delta_{m-1}} = \overline{\beta_m \gamma_m} / \widetilde{\gamma}_m$;
  end if
  $\widetilde{v} = A v_m - v_m \alpha_m - v_{m-1} \beta_m$;
  $\widetilde{w} = A^H w_m - w_m \widetilde{\alpha}_m - w_{m-1} \widetilde{\beta}_m$;
  $\widetilde{\delta} = \widetilde{w}^H \widetilde{v}$;
  if $\widetilde{\delta} = 0$ **then**
    Setze $L = m$, Stopp;
  else
    Wähle $\gamma_{m+1} \neq 0$, $\widetilde{\gamma}_{m+1} \neq 0$ und δ_{m+1}, so dass $\gamma_{m+1} \overline{\widetilde{\gamma}_{m+1}} \delta_{m+1} = \widetilde{\delta}$;
    Setze $v_{m+1} = \widetilde{v}/\gamma_{m+1}$, $w_{m+1} = \widetilde{w}/\widetilde{\gamma}_{m+1}$,
  end if
end for

Die Kosten für einen Schritt des Lanczos-Verfahrens sind vier SAXPYs, zwei Skalarprodukte und zwei Matrix-Vektormultiplikationen (eine mit A und eine mit A^H).
Bemerkungen.

(a) Häufig werden die Skalierungsfaktoren γ_m und $\widetilde{\gamma}_m$ so gewählt, dass $\|v_m\| = \|w_m\| = 1$ gilt.

(b) Wählt man $\widetilde{\gamma}_m = \overline{\gamma_m}$, so folgt $\widetilde{\beta}_m = \overline{\beta_m}$. Die Rekursionskoeffizienten für v's und w's stimmen dann bis auf Konjugation überein.

(c) Ist $A = A^H$ und $b = c$, dann gilt $w_m = v_m$ für alle m, wenn wir $\gamma_m = \widetilde{\gamma}_m$ wählen. Der Algorithmus wird dann symmetrisches Lanczos-Verfahren genannt und ist äquivalent zum Arnoldi-Verfahren, wenn die Vektoren v_m auf Norm eins normiert werden.

Wir schreiben wieder $V_m = [\ v_1 \ \ldots \ v_m\ ]$ und $W_m = [\ w_1 \ \ldots \ w_m\ ]$. Die Koeffizienten des Lanczos-Verfahrens fassen wir in einer Tridiagonalmatrix zusammen:

$$\widetilde{T}_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & \\ \gamma_2 & \alpha_2 & \beta_3 & & \\ & \gamma_3 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_m \\ & & & \gamma_m & \alpha_m \\ & & & & \gamma_{m+1} \end{bmatrix}, \qquad T_m = [\ I_m \ \ 0\ ]\widetilde{T}_m. \tag{7.11}$$

Satz 7.8. *Für das nichtsymmetrische Lanczos-Verfahren gilt*

(a) $W_m^H V_m = D_m = \operatorname{diag}(\delta_1, \ldots, \delta_m)$, $m = 1, \ldots, L$;

(b) $AV_m = V_{m+1}\widetilde{T}_m = V_m T_m + \gamma_{m+1} v_{m+1} e_m^T$;

(c) $A^H W_m = W_{m+1}\widetilde{S}_m = W_m S_m + \widetilde{\gamma}_{m+1} w_{m+1} e_m^T$, *wobei* $\widetilde{S}_m$ *analog zu* $\widetilde{T}_m$ *definiert ist, jedoch mit Koeffizienten* $\widetilde{\alpha}_j$, $\widetilde{\beta}_j$, $\widetilde{\gamma}_j$;

(d) $W_m^H A V_m = D_m T_m = S_m^H D_m$;

(e) *In exakter Arithmetik bricht der Lanczos-Prozess nach* $m = L - 1$ *Schritten ab, wobei*

$$1 \le L \le \min\{\dim \mathcal{K}_n(A, b), \dim \mathcal{K}_n(A^H, c)\}.$$

Beweis. (a), (b) und (c) folgen aus der Konstruktion und den Rekursionsformeln.

(d) Nach (b) gilt $W_m^H A V_m = W_m^H V_{m+1}\widetilde{T}_m = [\ D_m \ \ 0\ ]\widetilde{T}_m = D_m T_m$ und nach (c) ist $W_m^H A V_m = \widetilde{S}_m^H W_{m+1}^H V_m = S_m^H D_m$.

(e) Sei $k = \dim \mathcal{K}_n(A, b)$. Angenommen, der Lanczos-Prozess stoppt nicht früher als für $m = k$, insbesondere ist also $\delta_j \neq 0$ für $j = 1, \ldots, k$. Dann gilt $\mathcal{K}_k(A, b) = \mathcal{K}_{k+1}(A, b) = \mathcal{K}_n(A, b)$ und daher ist $v_{k+1} \in \mathcal{K}_k(A, b)$. Es gibt also ein $z \in \mathbb{C}^k$, so dass $v_{k+1} = V_k z$. Aus der Biorthogonalitätsbedingung folgt dann

$$0 = W_k^H v_{k+1} = W_k^H V_k z = D_k z.$$

Nach Voraussetzung ist D_k nicht singulär, also $z = 0$. Der Lanczos-Algorithmus bricht wegen $v_{k+1} = 0$ ab.

Analog folgert man aus $k = \dim \mathcal{K}_n(A^H, c)$ einen Abbruch wegen $w_{k+1} = 0$, so dass $L \le \min\{\dim \mathcal{K}_n(A, b), \dim \mathcal{K}_n(A^H, c)\}$. □

Wegen (d) nennt man w_m einen **linken Lanczos-Vektor** und v_m einen **rechten Lanczos-Vektor**.

Bemerkungen. Der Algorithmus bricht ab, wenn $\widetilde{\delta} = \widetilde{w}_L^H \widetilde{v}_L = 0$. Hierfür gibt es zwei verschiedene Ursachen:

(a) (Regulärer Zusammenbruch) $\widetilde{w}_L = 0$ oder $\widetilde{v}_L = 0$. Ist $\widetilde{w}_L = 0$, dann spannen die linken Lanczos-Vektoren einen A^H-invarianten Unterraum auf, ist $\widetilde{v}_L = 0$, dann spannen die rechten Lanczos-Vektoren einen A-invarianten Unterraum auf.

(b) (Ernsthafter Zusammenbruch) $\widetilde{w}_L \neq 0$ und $\widetilde{v}_L \neq 0$. Die Lanczos-Vektoren spannen weder einen A-invarianten noch einen A^H-invarianten Unterraum auf.

In endlicher Arithmetik ist ein ernsthafter Zusammenbruch unwahrscheinlich. Stabilitätsprobleme treten durch *Fastzusammenbrüche* auf, die durch $|\widetilde{\delta}| \approx 0$ verursacht werden, wobei weder $\widetilde{w}_L$ noch $\widetilde{v}_L$ kleine Norm haben. Exakte und Fastzusammenbrüche können durch **look-ahead** vermieden werden. Ganz grob gesprochen, schwächt man bei look-ahead-Verfahren die Biorthogonalitätsbedingung ab und fordert nur noch Blockbiorthogonalität. Das bedeutet, dass D_m blockdiagonal statt diagonal wird, wobei in der Praxis nur wenige Blöcke Dimension größer eins, etwa zwei bis vier haben. Als Folgerung wird T_m blocktridiagonal, das Verfahren kann also immer noch mit kurzen Block-3-Term-Rekursionen implementiert werden. Aus Zeitgründen verzichten wir hier auf die Details.

7.4 Verfahren basierend auf dem Lanczos-Prozess

Wir zeigen jetzt, wie man durch die Minimierungsbedingung (M) oder die Galerkin-Bedingung (G) charakterisierte Lösungen mit Hilfe der Lanczos-Basen berechnen kann. Dazu nehmen wir an, dass linke und rechte Lanczos-Vektoren auf Norm eins normiert sind und dass kein vorzeitiger Zusammenbruch auftritt.

Wir beginnen mit der Minimierungsbedingung (M). Für $x_m = V_m y_m$ gilt nach (7.5)

$$r_m = V_{m+1}(\beta e_1 - \widetilde{T}_m y_m) =: V_{m+1} t_m.$$

Da im Gegensatz zum Arnoldi-Verfahren V_m nicht unitär ist, ist die Minimierung von $\|r_m\|$ zu teuer (man müsste ein voll besetztes lineares Ausgleichsproblem der Dimension $n \times m$ lösen). Die Bedingung (M) bedeutet hier nicht die Minimierung von $\|r_m\|$, sondern nur die Minimierung des Koeffizientenvektors $\|t_m\|$ von r_m in der Lanczos-Basis V_{m+1}. Das ergibt

$$\|r_m\| \le \|V_{m+1}\| \cdot \|t_m\| \le \sqrt{m+1}\|t_m\|.$$

Das daraus resultierende Verfahren wurde von Freund und Nachtigal 1991 vorgeschlagen und **quasi minimal residual method** (QMR) genannt.

Bemerkung. Falls die Lanczos-Vektoren nicht auf eins normiert sind, führt man eine Gewichtsmatrix $\Omega_m = \mathrm{diag}(\|v_1\|, \ldots, \|v_m\|)$ ein und erhält aus

$$r_m = V_{m+1}\Omega_{m+1}^{-1} t_m, \qquad t_m = \Omega_{m+1}(\beta e_1 - \widetilde{T}_m y_m)$$

ebenfalls obige Abschätzung.

Wie bei GMRES erfolgt die Lösung des linearen Ausgleichsproblems mit einer QR-Zerlegung der Tridiagonalmatrix $\widetilde{T}_m$:

$$Q_m \widetilde{T}_m = \widetilde{R}_m = \begin{bmatrix} R_m \\ 0 \end{bmatrix}.$$

Weil die Householder-Transformationen zur Berechnung der QR-Zerlegung immer nur in 2×2-Diagonalblöcken von der Einheitsmatrix abweichen, impliziert die Tridiagonalform von $\widetilde{T}_m$ eine Bandstruktur von R_m (Übung). Angenommen, wir hätten bereits

$$Q_{m-1}\widetilde{T}_{m-1} = G_{m-1,m} G_{m-2,m} \cdots G_{1,m} \widetilde{T}_{m-1} = \begin{bmatrix} R_{m-1} \\ 0 \end{bmatrix}$$

mit $m \times m$ Householder-Matrizen $G_{j,m}$. Um hieraus die Faktorisierung von $\widetilde{T}_m$ zu erhalten, geht man von

$$\widetilde{T}_m = \left[\begin{array}{c|c} & 0_{m-2} \\ \widetilde{T}_{m-1} & \beta_m \\ & \alpha_m \\ \hline 0 & \gamma_{m+1} \end{array}\right]$$

aus und berechnet eine 2×2 Householder-Matrix $\widetilde{G}_m$ derart, dass

$$\widetilde{G}_m \begin{bmatrix} \alpha'_m \\ \gamma_{m+1} \end{bmatrix} = \begin{bmatrix} \zeta_m \\ 0 \end{bmatrix}, \qquad \widetilde{G}_m = \begin{bmatrix} c_m & \overline{s_m} \\ s_m & -c_m \end{bmatrix}, \qquad \alpha'_m = s_{m-1}\beta_m - c_{m-1}\alpha_m$$

abgebildet wird. Es ergibt sich

$$R_m = \begin{bmatrix} \zeta_1 & \eta_2 & \theta_3 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \theta_m \\ & & & \ddots & \eta_m \\ & & & & \zeta_m \end{bmatrix},$$

wobei $\zeta_m \neq 0$, falls $\operatorname{Rang} \widetilde{T}_m = m$. Wie früher definieren wir

$$\widetilde{q}_m = \beta Q_m e_1 = \begin{bmatrix} q_m \\ \rho_m \end{bmatrix},$$

so dass $y_m = R_m^{-1} q_m$ die Lösung von (M) ist. Obwohl R_m eine Bandmatrix mit nur drei von Null verschiedenen Diagonalen ist, ist R_m^{-1} im Allgemeinen eine voll besetzte obere Dreiecksmatrix. Zur Berechnung von $x_m = V_m y_m$ wären damit *alle* bisher berechneten rechten Lanczos-Vektoren erforderlich. Der wesentliche Vorteil des Lanczos-Verfahrens war aber, dass man nur drei rechte und drei linke Lanczos-Vektoren speichern muss. Auf keinen Fall will man daher wesentlich mehr Vektoren zur Berechnung von x_m verwenden. Dies ist möglich, wenn man zusätzlich Richtungsvektoren abspeichert. Es gilt

$$x_m = V_m R_m^{-1} q_m, \qquad \text{wobei} \quad \widetilde{q}_m = G_{m,m+1} \begin{bmatrix} \widetilde{q}_{m-1} \\ 0 \end{bmatrix} = \begin{bmatrix} q_m \\ \rho_m \end{bmatrix}.$$

Da $G_{m,m+1}$ sich nur in der unteren rechten 2×2 Ecke von der Einheitsmatrix unterscheidet, stimmen die ersten $m-1$ Komponenten von q_m mit denen von q_{m-1} überein und für die letzte Komponente gilt

$$\tau_m = e_m^T q_m = c_m \rho_{m-1}, \tag{7.12}$$

wobei c_m das $(1,1)$-Element der 2×2 Householder-Matrix bezeichnet (was dem (m,m)-Element von $G_{m,m+1}$ entspricht). Dies ergibt die folgende Update-Formel für x_m

$$x_m = x_{m-1} + \tau_m V_m R_m^{-1} e_m. \tag{7.13}$$

Der Trick besteht nun in der Definition von Suchrichtungen

$$P_m = V_m R_m^{-1} = [\ p_1 \quad \dots \quad p_m\ ].$$

Betrachten wir $V_m = P_m R_m$ spaltenweise, so ergibt sich

$$v_m = \theta_m p_{m-2} + \eta_m p_{m-1} + \zeta_m p_m$$

oder äquivalent

$$p_m = \frac{1}{\zeta_m}(v_m - \eta_m p_{m-1} - \theta_m p_{m-2}).$$

Die Suchrichtungen können daher ebenfalls mit dreigliedrigen Rekursionsformeln berechnet werden und daraus ergibt sich für die Iterierten

$$x_m = x_{m-1} + \tau_m p_m. \tag{7.14}$$

Algorithmus 7.4 QMR

Wähle $x_0 \in \mathbb{C}^n$, setze $r_0 = b - Ax_0$, $\beta = \rho_0 = \|r_0\|$,
Setze $p_0 = p_{-1} = 0$;
Starte den nichtsymmetrischen Lanczos-Prozess 7.3 mit geeignetem linken Startvektor c und $v_1 = r_0/\beta$.
Am Ende des m-ten Schritts ergänze die folgenden Zeilen:
    Berechne $\theta_m, \eta_m, \zeta_m, \rho_m, \tau_m$;
    Setze $p_m = (v_m - \eta_m p_{m-1} - \theta_m p_{m-2})/\zeta_m$;
    Berechne $x_m = x_{m-1} + \tau_m p_m$.

Die Kosten sind sieben SAXPYs, zwei Skalarprodukte und zwei Matrix-Vektormultiplikationen (eine mit A und eine mit A^H) pro Schritt.

Als nächstes betrachten wir die Petrov-Galerkin-Bedingung (G) mit dem Unterraum $\mathcal{W}_m = \mathcal{K}_m(A^H, c)$ und den linken Lanczos-Vektoren als Basis W_m. Dann ist (G) äquivalent zu

$$W_m^H r_m = W_m^H V_{m+1}(\beta e_1 - \widetilde{T}_m y_m) = D_m(\beta e_1 - T_m y_m) = 0.$$

Da D_m nicht singulär ist, folgt $T_m y_m = \beta e_1$. Das daraus resultierende Verfahren heißt **biconjugate gradient method** (BiCG) und wurde von Lanczos 1952 und Fletcher 1976 vorgeschlagen.

Satz 7.9. *Seien V_m, W_m die Matrizen aus dem Lanczos-Prozess und $x_m = V_m y_m$. Dann existiert eine Lösung von* (G) *genau dann, wenn T_m nicht singulär ist. In diesem Fall ist $T_m y_m = \beta e_1$.*

Beweis. Der Beweis von Satz 7.6 gilt hier ebenso. □

Wie bei QMR kann auch BiCG mit kurzen Rekursionen für die Iterierten implementiert werden. Die wesentliche Idee ist hier, die LDU-Zerlegung von T_m, $T_m = L_m E_m U_m$, zu verwenden. Setzt man in

$$T_m = \begin{bmatrix} 1 & & & \\ \nu_2 & \ddots & & \\ & \ddots & \ddots & \\ & & \nu_m & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_1 & & & \\ & \ddots & & \\ & & \ddots & \\ & & & \varepsilon_m \end{bmatrix} \begin{bmatrix} 1 & \mu_2 & & \\ & \ddots & \ddots & \\ & & \ddots & \mu_m \\ & & & 1 \end{bmatrix}$$

den $(m, m-1)$-ten und den $(m-1, m)$-ten Eintrag auf beiden Seiten gleich, so folgt

$$\gamma_m = \nu_m \varepsilon_{m-1}, \qquad \beta_m = \varepsilon_{m-1}\mu_m. \tag{7.15}$$

Hieraus kann man μ_m und ν_m berechnen. Durch Vergleich der (m, m)-ten Einträge ergibt sich

$$\alpha_m = \begin{bmatrix} \nu_m & 1 \end{bmatrix} \begin{bmatrix} \varepsilon_{m-1}\mu_m \\ \varepsilon_m \end{bmatrix}.$$

Die LDU-Zerlegung einer Tridiagonalmatrix haben wir bereits in Algorithms 6.11 dargestellt. Algorithmus 7.5 entspricht diesem in der hier verwendeten Notation.

Algorithmus 7.5 LDU-Zerlegung von T_m (vgl. Algorithmus 6.11)

$\varepsilon_1 = \alpha_1$;
for $j = 2, \ldots, m$ **do**
  $\nu_j = \gamma_j / \varepsilon_{j-1}$;
  $\mu_j = \beta_j / \varepsilon_{j-1}$;
  $\varepsilon_j = \alpha_j - \nu_j \beta_j$;
end for

Nach Satz 7.9 sind die Galerkin-Iterierten durch

$$x_m = V_m y_m, \qquad y_m = \beta T_m^{-1} e_1$$

gegeben. Obwohl für die LDU-Zerlegung lediglich ε_m, ν_m und μ_m im m-ten Schritt neu berechnet werden müssen, unterscheiden sich im Allgemeinen alle Einträge in y_m von denen in y_{m-1}. Führen wir jedoch wie bei QMR Suchrichtungen ein, so erreichen wir kurze Rekursionen für x_m: Wegen $T_m^{-1} = U_m^{-1} E_m^{-1} L_m^{-1}$ gilt

$$x_m = P_m z_m, \qquad P_m = V_m U_m^{-1}, \qquad z_m = \beta E_m^{-1} L_m^{-1} e_1.$$

Da U_m bidiagonal ist, kann p_m mit einer zweigliedrigen Rekursionsformel berechnet werden:

$$p_1 = v_1, \qquad p_m = v_m - \mu_m p_{m-1}, \quad m = 2, 3, \ldots \tag{7.16}$$

Außerdem ist $z_m = \begin{bmatrix} z_{m-1} \\ \tau_m \end{bmatrix}$, denn

$$L_m E_m z_m = \beta e_1 = \left[\begin{array}{cc|c} & L_{m-1} & 0 \\ 0 & \nu_m & 1 \end{array}\right] \left[\begin{array}{cc|c} & E_{m-1} & 0 \\ 0 & & \varepsilon_m \end{array}\right] \begin{bmatrix} z_{m-1} \\ \tau_m \end{bmatrix} = \begin{bmatrix} \beta \\ 0_{m-1} \end{bmatrix},$$

falls $\varepsilon_{m-1}\tau_{m-1}\nu_m + \varepsilon_m \tau_m = 0$ oder

$$\tau_m = -\frac{\varepsilon_{m-1}\tau_{m-1}\nu_m}{\varepsilon_m}, \qquad \tau_1 = \frac{\beta}{\epsilon_1}. \tag{7.17}$$

Damit kann x_m gemäß

$$x_m = x_{m-1} + \tau_m p_m \tag{7.18}$$

berechnet werden. Zusammengefasst ergibt sich Algorithmus 7.6.

Der Aufwand pro Schritt beläuft sich auf sechs SAXPYs, zwei Skalarprodukte und zwei Matrix-Vektormultiplikationen (eine mit A und eine mit A^H).

Algorithmus 7.6 BiCG basierend auf 3-Term-Rekursionen

Wähle $x_0 \in \mathbb{C}^n$, setze $r_0 = b - Ax_0$, $\beta = \rho_0 = \|r_0\|$,
Setze $p_1 = r_0$;
Starte den nichtsymmetrischen Lanczos-Prozess mit geeignetem linken Startvektor c und $v_1 = r_0$.
Am Ende des m-ten Schritts ergänze die folgenden Zeilen:
if $m = 1$ **then**
  $\varepsilon_1 = \alpha_1$, $\tau_1 = \beta/\varepsilon_1$;
else
  {Berechne den Update der LDU-Zerlegung von T_m}
  $\nu_m = \gamma_m/\varepsilon_{m-1}$, $\mu_m = \beta_m/\varepsilon_{m-1}$, $\varepsilon_m = \alpha_m - \nu_m\beta_m$;
  Setze $\tau_m = -\varepsilon_{m-1}\tau_{m-1}\nu_m/\varepsilon_m$;
  Berechne $p_m = v_m - \mu_m p_{m-1}$;
end if
Berechne $x_m = x_{m-1} + \tau_m p_m$.

Während die QMR-Iterierten für jedes m existieren und mit Hilfe einer QR-Zerlegung stabil berechnet werden können, kann es bei BiCG genau wie bei FOM vorkommen, dass zu gewissen Indizes keine Iterierten existieren, die (G) erfüllen. Dies ist nach Satz 7.9 genau dann der Fall, wenn T_m singulär ist. In der Praxis wesentlich problematischer ist, dass eine schlecht konditionierte Matrix T_m zu numerischen Instabilitäten im BiCG-Algorithmus führt. Da im Gegensatz zu FOM die Iterierten von Schritt zu Schritt durch eine Update-Formel erzeugt werden, trägt eine "schlechte" Iterierte zu allen folgenden Iterierten bei. Wir werden daher als nächstes zeigen, wie man dieses Problem umgehen kann, indem man die BiCG-Iterierten – falls sie existieren – aus QMR-Iterierten berechnet.

Hierzu stellen wir T_m mit Hilfe der QR-Zerlegung von $\widetilde{T}_m$ dar:

$$\begin{aligned} T_m = [\ I_m \quad 0\ ]\widetilde{T}_m &= [\ I_m \quad 0\ ]G_{1,m+1}^H \cdots G_{m-1,m+1}^H G_{m,m+1}^H \begin{bmatrix} R_m \\ 0 \end{bmatrix} \\ &= G_{1,m}^H \cdots G_{m-1,m}^H \begin{bmatrix} I_{m-1} & 0 \\ 0 & c_m \end{bmatrix} R_m \\ &= Q_{m-1}^H \begin{bmatrix} I_{m-1} & 0 \\ 0 & c_m \end{bmatrix} R_m. \end{aligned}$$

Lemma 7.10. *T_m ist genau dann singulär, wenn $c_m = 0$.*

Beweis. R_m ist nicht singulär und $Q_{m-1}^H = G_{1,m}^H \cdots G_{m-1,m}^H$ ist unitär. □

Die BiCG-Iterierten sind durch $x_m^{\mathrm{BiCG}} = V_m y_m^{\mathrm{BiCG}}$ gegeben, wobei

$$y_m^{\mathrm{BiCG}} = \beta T_m^{-1} e_1 = R_m^{-1} \begin{bmatrix} I_{m-1} & 0 \\ 0 & c_m^{-1} \end{bmatrix} (\beta Q_{m-1} e_1).$$

Es gilt

$$\beta Q_{m-1} e_1 = \widetilde{q}_{m-1} = \begin{bmatrix} q_{m-1} \\ \rho_{m-1} \end{bmatrix}$$

und daher

$$y_m^{\mathrm{BiCG}} = R_m^{-1} \begin{bmatrix} q_{m-1} \\ \rho_{m-1}/c_m \end{bmatrix}.$$

Für die QMR-Iterierten haben wir

$$y_m^{\mathrm{QMR}} = R_m^{-1} q_m = R_m^{-1} \begin{bmatrix} q_{m-1} \\ \tau_m \end{bmatrix},$$

woraus

$$y_m^{\mathrm{BiCG}} = y_m^{\mathrm{QMR}} + R_m^{-1} \begin{bmatrix} 0_{m-1} \\ \rho_{m-1}/c_m - \tau_m \end{bmatrix}$$

folgt.

Satz 7.11. *Wenn die m-te BiCG-Iterierte existiert, dann kann sie gemäß*

$$x_m^{\mathrm{BiCG}} = x_m^{\mathrm{QMR}} + \frac{\tau_m (1 - c_m^2)}{c_m^2} p_m^{\mathrm{QMR}}$$

aus der m-ten QMR-Iterierten und der m-ten QMR-Suchrichtung berechnet werden.

Beweis. Lemma 7.10 zeigt, dass die m-te BiCG-Iterierte genau dann existiert, wenn $c_m \neq 0$. Dann gilt

$$\begin{aligned} x_m^{\mathrm{BiCG}} &= V_m y_m^{\mathrm{QMR}} + V_m R_m^{-1} \begin{bmatrix} 0_{m-1} \\ \rho_{m-1}/c_m - \tau_m \end{bmatrix} \\ &= x_m^{\mathrm{QMR}} + (V_m R_m^{-1} e_m) \left(\frac{\rho_{m-1}}{c_m} - \tau_m \right). \end{aligned}$$

Aus $\tau_m = c_m \rho_{m-1}$ erhalten wir

$$\frac{\rho_{m-1}}{c_m} - \tau_m = \frac{\tau_m}{c_m^2} (1 - c_m^2)$$

Die Behauptung folgt aus $V_m R_m^{-1} e_m = P_m^{\mathrm{QMR}} e_m = p_m^{\mathrm{QMR}}$. □

Obwohl die von QMR erzeugten Residuennormen im Allgemeinen nicht monoton fallend sind, beobachtet man in der Regel eine relativ glatte, fast monoton fallende Kurve. Im Gegensatz dazu variieren die Residuennormen von BiCG meist sehr stark, wobei große Spitzen durchaus auftreten. Ein Beispiel für das Problem (7.6) ist in Abbildung 7.2 dargestellt.

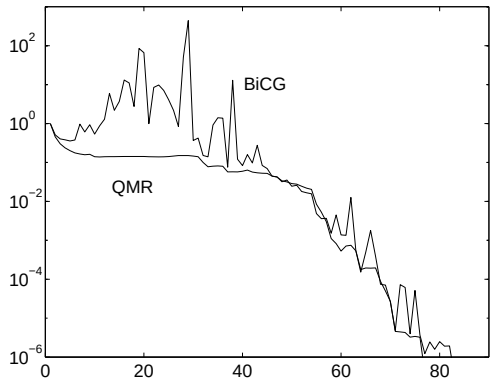

Abb. 7.2: Residuennormen von QMR und BiCG aufgetragen über die Anzahl der Iterationen.

Bemerkung. Ist $A = A^H$, so wählt man im Lanczos-Prozess den linken Startvektor $c = b$, denn dann stimmen rechte und linke Lanczos-Vektoren überein. Das BiCG-Verfahren ist dann mathematisch äquivalent zum Verfahren der konjugierten Gradienten, kurz dem cg-Verfahren. Allerdings wird das cg-Verfahren in der Regel nicht in dieser Form (d. h. mit dreigliedrigen Rekursionsformeln) implementiert, sondern mit gekoppelten 2-Term-Rekursionen. Diese stellen wir im nächsten Abschnitt vor. Eliminiert man im QMR-Algorithmus die Rekursionen für die linken Lanczos-Vektoren, so ergibt sich die **minimal residual method** (MINRES), die von Paige und Saunders 1975 vorgeschlagen wurde. Während wir hier MINRES als Spezialfall von QMR aufgefasst haben, ist es historisch genau umgekehrt gewesen: QMR ist als Verallgemeinerung von MINRES entwickelt worden. BiCG und cg hingegen wurden etwa gleichzeitig vorgeschlagen.

7.5 Das Verfahren der konjugierten Gradienten

Als Spezialfall betrachten wir lineare Gleichungssysteme (7.1) mit Koeffizientenmatrix

$$A = A^H \qquad \text{positiv definit.}$$

Wir wissen bereits, dass Arnoldi- und Lanczos-Verfahren dann (bis auf Skalierung) äquivalent sind, also eine Orthogonalbasis V_m des Krylov-Raumes $\mathcal{K}_m(A, b)$ bzw, $\mathcal{K}_m(A, r_0)$ berechnen. Linke Lanczos-Vektoren werden dann nicht benötigt.

Für die Galerkin-Näherungen (BiCG bzw. FOM) erhalten wir das Verfahren der konjugierten Gradienten (cg-Verfahren). Die Näherungen des cg-Verfahrens existieren nach Lemma 7.7 für jedes m, da A positiv definit ist und deswegen $0 \notin \mathcal{F}(A)$ gilt.

Das so definierte Galerkin-Verfahren ist für $A = A^H$ positiv definit tatsächlich ein Minimierungsverfahren, allerdings in der sogenannten Energienorm $\|x\|_A = \sqrt{x^H A x}$.

Satz 7.12. *Ist $\widehat{x} = A^{-1}b$ die Lösung von (7.1), $A = A^H$ positiv definit, $x_m \in x_0 + \mathcal{K}_m(A, r_0)$ die Galerkin-Iterierte ($r_m \perp \mathcal{K}_m(A, r_0)$), dann gilt für x_m (also die cg-Iterierten aus Algorithmus 7.7)*

$$\|\widehat{x} - x_m\|_A \le \|\widehat{x} - x\|_A$$

und

$$\|b - Ax_m\|_{A^{-1}} \le \|b - Ax\|_{A^{-1}}$$

für alle $x = x_0 + \mathcal{K}_m(A, r_0)$.

Beweis. Es sei $e_m = \widehat{x} - x_m$ der Fehler im m-ten Schritt. Wir betrachten einen beliebigen Punkt $x = x_m - \Delta x \in x_0 + \mathcal{K}_m$ mit Fehler $e = \widehat{x} - x = e_m + \Delta x$. Wegen $x_m \in x_0 + \mathcal{K}_m$ ist $\Delta x \in \mathcal{K}_m$. Für diesen gilt wegen $r_m = Ae_m$

$$\begin{aligned}\|e\|_A^2 &= (e_m + \Delta x)^H A(e_m + \Delta x)\\ &= e_m^H A e_m + (\Delta x)^H A(\Delta x) + e_m^H A\Delta x + (\Delta x)^H A e_m\\ &= e_m^H A e_m + (\Delta x)^H A(\Delta x) + 2\mathrm{Re}\,(r_m^H \Delta x).\end{aligned}$$

Wegen $\Delta x \in \mathcal{K}_m \perp r_m$ ist $r_m^H \Delta x = 0$. Es verbleibt

$$\|e\|_A^2 = e_m^H A e_m + (\Delta x)^H A(\Delta x).$$

Hierbei hängt nur der zweite Term von Δx ab und da A positiv definit ist, ist dieser nicht negativ, wobei der Wert 0 genau dann angenommen wird, wenn $\Delta x = 0$ ist. Dies ist genau für $x = x_m$ der Fall.

Die zweite Behauptung folgt für $r = b - Ax = Ae$ aus

$$\|e\|_A^2 = e^H Ae = (A^{-1}r)^H A(A^{-1}r) = r^H A^{-1} r = \|r\|_{A^{-1}}.$$

Damit ist der Satz bewiesen. □

Eine Folge dieses Satzes ist, dass der Algorithmus spätestens nach n Schritten mit $r_n = 0$ abbricht, denn die Residuen bilden eine Orthogonalbasis von $\mathcal{K}_k(A, b)$ und $\dim \mathcal{K}_k(A, r_0) \leq k \leq n$. Das 1952 von Hestenes und Stiefel vorgeschlagene cg-Verfahren wurde zunächst auch als direktes Verfahren zur Lösung von $Ax = b$ interpretiert. In der Praxis ist es jedoch nur als iteratives Verfahren (Reid 1971) relevant, denn brauchbare Näherungen werden häufig bereits nach wenigen Schritten erreicht.

Üblicherweise wird das cg-Verfahren nicht über den "Umweg" des Lanczos-Verfahrens hergeleitet und implementiert. Der Zusammenhang zwischen der herkömmlichen Herleitung und dem Lanczos-Verfahren soll im Folgenden hergestellt werden.

Mit der Notation aus dem letzten Abschnitt gilt

$$V_m^H V_m = D_m = \operatorname{diag}(\delta_1, \ldots, \delta_m)$$

und

$$AV_m = V_{m+1}\widetilde{T}_m, \qquad V_m^H A V_m = D_m T_m,$$

wobei

$$T_m = T_m^H = L_m^H E_m U_m, \qquad \widetilde{T}_m = \widetilde{L}_m^H E_m U_m, \qquad \widetilde{L}_m = L_{m+1} \begin{bmatrix} I_m \\ 0 \end{bmatrix}.$$

Für die Suchrichtungen $P_m = V_m U_m^{-1}$ folgt daraus

$$AP_m = AV_m U_m^{-1} = V_{m+1}\widetilde{L}_m E_m \tag{7.19}$$

und

$$P_m^H A P_m = U_m^{-H} V_m^H V_{m+1} \widetilde{L}_m E_m = U_m^{-H} D_m L_m E_m =: D'_m. \tag{7.20}$$

Da $P_m^H A P_m$ Hermitesch und D'_m eine untere Dreiecksmatrix ist, muss D'_m diagonal sein,

$$D'_m = \operatorname{diag}(\delta'_1, \ldots, \delta'_m), \qquad \delta'_m = p_m^H A p_m.$$

Des Weiteren sind die Residuenvektoren $r_m = b - Ax_m$ wegen (7.7) Vielfache der Lanczos-Vektoren und damit orthogonal (aber i. A. nicht orthonormal). Dies legt nahe, statt mit den Lanczos-Vektoren v_j direkt mit den Residuenvektoren r_j zu rechnen. Hierzu muss man im Lanczos-Verfahren die Skalierungsfaktoren γ_m so wählen, dass $v_{m+1} = r_m$ gilt. Die Initiatlisierung $v_1 = r_0$ führt auf $\beta = 1$. Um herauszufinden, wie diese Skalierungsfaktoren berechnet werden, leiten wir zunächst einige Rekursionsformeln her.

Wegen

$$x_m = x_{m-1} + \tau_m p_m$$

gilt für die Residuenvektoren die Update-Formel

$$r_m = b - Ax_m = b - Ax_{m-1} - \tau_m A p_m = r_{m-1} - \tau_m A p_m. \tag{7.21}$$

Außerdem folgt aus $P_m U_m = V_m$ die Rekursionsformel

$$p_m = v_m - \mu_m p_{m-1}. \tag{7.22}$$

Multiplizieren wir diese Gleichung mit A, so ergibt sich mit (7.21) und (7.19)

$$Ap_m = \frac{1}{\tau_m}(r_{m-1} - r_m) = \epsilon_m(v_m + \nu_{m+1}v_{m+1}).$$

Hieraus erkennt man, dass $\nu_{m+1} = -1$ die Skalierung ist, die $v_{m+1} = r_m$ liefert, denn dann gilt $\epsilon_m\tau_m = 1$ für jedes $m = 1, 2, \ldots$, wie man aus (7.17) mit $\beta = 1$ erkennt. Wegen (7.15) folgt schließlich $\gamma_{m+1} = -\epsilon_m$. Ebenfalls aus (7.15) erhalten wir mit $\delta_m = v_m^H v_m = r_{m-1}^H r_{m-1}$ und (7.10)

$$\mu_m = \frac{\beta_m}{\epsilon_{m-1}} = \frac{\gamma_m}{\epsilon_{m-1}}\frac{\delta_m}{\delta_{m-1}} = -\frac{\delta_m}{\delta_{m-1}},$$

sowie aus (7.20)

$$\epsilon_m = \frac{\delta'_m}{\delta_m} \qquad \text{also} \qquad \tau_m = \frac{\delta_m}{\delta'_m}.$$

Damit ergibt sich Algorithmus 7.7 (mit $\mu'_m = -\mu_m$).

Satz 7.13. *Für die Residuenvektoren $r_0, \ldots, r_m$ und die Suchrichtungen $p_1, \ldots, p_m$ des cg-Verfahrens 7.7 gilt*

$$\operatorname{span}\{r_0, \ldots, r_m\} = \mathcal{K}_{m+1}(A, r_0), \qquad \operatorname{span}\{p_1, \ldots, p_m\} = \mathcal{K}_m(A, r_0).$$

Die Residuenvektoren sind orthogonal, $r_k^H r_j = 0$ für $j \neq k$, $j, k = 0, \ldots, m$ und die Suchrichtungen sind A-konjugiert, $p_k^H A p_j = 0$ für $j \neq k$, $j, k = 1, \ldots, m$.

Algorithmus 7.7 Verfahren der konjugierten Gradienten

Wähle $x_0 \in \mathbb{C}^n$, setze $r_0 = b - Ax_0$ und wähle eine Toleranz $\texttt{tol} > 0$;
Initialisiere $m = 1$, $p_1 = r_0$, $\delta_1 = r_0^H r_0$;
while $\delta_m \geq \texttt{tol}^2$ **do**
  Berechne $\delta'_m = p_m^H A p_m$;
  Berechne $\tau_m = \delta_m/\delta'_m$;
  Setze $x_m = x_{m-1} + \tau_m p_m$;
  Berechne $r_m = r_{m-1} - \tau_m A p_m$;
  Berechne $\delta_{m+1} = r_m^H r_m$, $\mu'_{m+1} = \delta_{m+1}/\delta_m$;
  Setze $p_{m+1} = r_m + \mu'_{m+1} p_m$;
  Ersetze m durch $m + 1$.
end while

Für das cg-Verfahren für A Hermitesch und positiv definit können Fehlerabschätzungen angeben, die auf der Lösung eines polynomialen Approximationsproblems beruhen. In Satz 7.12 haben wir gezeigt, dass die Iterierten den Fehler in der A-Norm minimieren. Dies führt auf die folgende Fehlerabschätzung:

Satz 7.14. *Ist $A = A^H$ positiv definit, so gilt für den Fehler der m-ten Iterierten des cg-Verfahrens*

$$\|x_m - \widehat{x}\|_A \leq \min_{\substack{p_m \in \mathcal{P}_m \\ p_m(0)=1}} \|p_m(A)\| \|x_0 - \widehat{x}\|_A.$$

Beweis. Für $A = A^H$ existiert $A^{1/2}$, denn ist $A = Q\Lambda Q^H$ eine unitäre Transformation auf Diagonalform, so kann man wegen $\lambda_j > 0$ für $j = 1, \ldots, n$ in eindeutiger Weise

$$A^{1/2} = Q\Lambda^{1/2}Q^H \quad \text{mit} \quad \Lambda^{1/2} = \operatorname{diag}(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_n})$$

definieren. Es gilt dann $\|x\|_A^2 = x^H Ax = \|A^{1/2}x\|^2$.

Ohne Einschränkung nehmen wir wieder $x_0 = 0$ an. Nach Satz 7.12 gilt

$$\|x_m - \widehat{x}\|_A \le \|x - \widehat{x}\|_A \qquad \text{für alle } x \in \mathcal{K}_m(A, b).$$

Jedes $x \in \mathcal{K}_m(A, b)$ kann als $x = q_{m-1}(A)b = q_{m-1}(A)A\widehat{x}$ geschrieben werden, wobei $q_{m-1} \in \mathcal{P}_{m-1}$. Also ist $\widehat{x} - x = p_m(A)\widehat{x}$ mit $p_m(\lambda) = 1 - \lambda q_{m-1}(\lambda)$ und es folgt die Abschätzung

$$\|x_m - \widehat{x}\|_A \le \|p_m(A)\widehat{x}\|_A \qquad \text{für alle } p_m \in \mathcal{P}_m \text{ mit } p_m(0) = 1.$$

Ferner folgt aus $A^{1/2}p_m(A) = p_m(A)A^{1/2}$

$$\|p_m(A)\widehat{x}\|_A^2 = \|p_m(A)A^{1/2}\widehat{x}\|_2^2 \le \|p_m(A)\|_2^2 \, \|\widehat{x}\|_A^2.$$

Dies schließt den Beweis. □

Mit Hilfe der Tschebyscheff-Polynome kann man eine explizite obere Schranke für das polynomiale Minimierungsproblem angeben.

Satz 7.15. *Ist $A = A^H$ positiv definit, dann gilt*

$$\min_{\substack{p_m \in \mathcal{P}_m \\ p_m(0)=1}} \|p_m(A)\| \le 2\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^m,$$

wobei $\kappa = \kappa(A)$ die Konditionszahl von A bezeichnet.

Beweis. Es sei $\mathcal{F}(A) = [\alpha, \beta]$ mit $0 < \alpha < \beta$. Dann ist nach Lemma 6.15 $\kappa = \beta/\alpha$ und es gilt

$$\|p_m(A)\| = \max_{\lambda \in \lambda(A)} |p_m(\lambda)| \le \max_{\lambda \in [\alpha, \beta]} |p_m(\lambda)|.$$

Wir transformieren das Problem vom Intervall $[\alpha, \beta]$ nach $[-1, 1]$:

$$\begin{aligned} \ell : [\alpha, \beta] &\to [-1, 1] \\ \lambda &\mapsto t = \frac{2}{\beta - \alpha}\lambda - \frac{\beta + \alpha}{\beta - \alpha}. \end{aligned}$$

Durch diese Transformation wird 0 auf

$$\ell(0) = -\frac{\beta + \alpha}{\beta - \alpha} = -\frac{\kappa + 1}{\kappa - 1} =: t_0 < -1$$

abgebildet. Damit ist

$$\min_{\substack{p_m \in \mathcal{P}_m \\ p_m(0)=1}} \|p_m(A)\| \le \min_{\substack{p_m \in \mathcal{P}_m \\ p_m(t_0)=1}} \max_{\lambda \in [-1,1]} |p_m(\lambda)|.$$

Aus Abschnitt 2.3, Lemma 2.11 wissen wir, dass die rechte Seite für das skalierte Tschebyscheff-Polynom

$$\frac{T_m(t)}{T_m(t_0)}$$

minimal wird, so dass

$$\min_{\substack{p_m \in \mathcal{P}_m \\ p_m(0)=1}} \|p_m(A)\| \leq \frac{1}{|T_m(t_0)|}.$$

Es gilt

$$T_m(t) = \frac{1}{2}\left[\left(t + \sqrt{t^2-1}\right)^m + \left(t + \sqrt{t^2-1}\right)^{-m}\right],$$

denn für $t = \cos x$ ist $T_m(t) = \cos mx$ und dasselbe gilt für die rechte Seite (Übung). Daraus erhalten wir die Abschätzung

$$|T_m(t_0)| = T_m(|t_0|) \geq \frac{1}{2}\left[|t_0| + \sqrt{t_0^2-1}\right]^m.$$

Einsetzen von t_0 liefert

$$\begin{aligned}
|t_0| + \sqrt{t_0^2-1} &= \frac{\kappa+1}{\kappa-1} + \left(\frac{\kappa^2+2\kappa+1-(\kappa^2-2\kappa+1)}{(\kappa-1)^2}\right)^{1/2} \\
&= \frac{1}{\kappa-1}\left(\kappa+1+2\sqrt{\kappa}\right) \\
&= \frac{(\sqrt{\kappa}+1)^2}{(\sqrt{\kappa}+1)(\sqrt{\kappa}-1)} \\
&= \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}.
\end{aligned}$$

Aus

$$|T_m(t_0)| \geq \frac{1}{2}\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^m$$

ergibt sich schließlich die Behauptung. □

Das cg-Verfahren bewirkt eine schnelle Fehlerreduktion, wenn die Kondition von A klein ist oder die Eigenwerte von A in einigen wenigen Clustern liegen. Letzteres gilt auch für Krylov-Verfahren für nichtsymmetrische Matrizen, während hier die Kondition kaum eine Rolle spielt.

Haben die Eigenwerte diese Eigenschaft nicht, so kann man versuchen, zu einem äquivalenten linearen Gleichungssystem überzugehen, welches bessere Eigenschaften hat. Eine Möglichkeit der sogenannten *Vorkonditionierung* lernen wir im nächsten Abschnitt kennen.

7.6 Vorkonditionierung

Ein lineares Gleichungssystem $Ax = b$, wobei A groß und dünn besetzt ist, kann mit Hilfe von Krylov-Verfahren näherungsweise gelöst werden. Die Konvergenz hängt wesentlich von der Lösung eines polynomialen Approximationsproblems und damit von der Lage der Eigenwerte oder des Wertebereichs ab.

Exemplarisch zeigen wir für $A \in \mathbb{R}^{n,n}$ symmetrisch und positiv definit, wie die Konvergenz durch Übergang zu einem äquivalenten System beschleunigt werden kann. Für B nicht singulär ist

$$Ax = b \Longleftrightarrow BAx = Bb.$$

Für das Verfahren der konjugierten Gradienten wäre $B = A^{-1}$ die optimale Wahl, denn dies würde zu Konvergenz in einem Schritt führen. Aus offensichtlichen Gründen ist diese Wahl

jedoch nicht praktikabel. Daher verlangen wir für die Vorkonditionierungsmatrix B folgende Eigenschaften:

(a) $B \approx A^{-1}$;

(b) $B \in \mathbb{R}^{n,n}$ symmetrisch und positiv;

(c) $x \mapsto Bx$ leicht zu berechnen.

Falls (a) gilt, kann man $\kappa(BA) \ll \kappa(A)$ und damit nach Satz 7.15 wesentlich schnellere Konvergenz erwarten. (b) ist sinnvoll, da A^{-1} diese Eigenschaften hat. In der Regel wird man zwischen den widersprüchlichen Forderungen (a) und (c) einen Kompromiss finden müssen.

Eine Möglichkeit ist eine **unvollständige Cholesky-Zerlegung** von A. Bei der Berechnung der Cholesky-Zerlegung wie wir sie in Abschnitt 3.5 kennengelernt haben, entstehen im Allgemeinen Cholesky-Faktoren, die nicht mehr dünn besetzt sind. Bei der unvollständigen Cholesky-Zerlegung berechnet man hingegen eine Faktorisierung

$$A = \widetilde{L}\widetilde{L}^T + R,$$

wobei $\widetilde{L}$ eine untere Dreiecksmatrix mit derselben Besetzungsstruktur wie A auf und unterhalb der Diagonalen und R eine Restmatrix ist, vgl. Algorithmus 7.8. Varianten lassen auch andere (dünne) Besetzungsstrukturen zu, insbesondere bei der unvollständigen LU-Zerlegung für nichtsymmetrische Matrizen. Wir gehen hier nicht näher darauf ein.

Algorithmus 7.8 Unvollständige Cholesky-Zerlegung

for $j = 1, \ldots, n$ **do**
  $l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 \right)^{1/2}$
  for $i = j+1, \ldots, n$ **do**
    if $a_{ij} \neq 0$ **then**
      $l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk} \right) / l_{jj}$ {summiert wird nur über k mit $a_{ik} a_{jk} \neq 0$}
    else
      $l_{ij} = 0$
    end if
  end for
end for

Häufig bewirkt $B = (\widetilde{L}\widetilde{L}^T)^{-1}$ eine Clusterung der Eigenwerte von BA (ohne Beweis). Leider kann man das cg-Verfahren nur auf symmetrische und positiv definite Matrizen anwenden, BA ist jedoch im Allgemeinen nicht symmetrisch. Zu einem symmetrischen System mit denselben Eigenschaften gelangt man, wenn eine Faktorisierung $B = CC^T$ von B bekannt ist, etwa $C = \widetilde{L}^{-T}$ bei der unvollständigen Cholesky-Zerlegung. Es ist nämlich

$$CC^T Ax = CC^T b \Longleftrightarrow C^T AC \cdot C^{-1} x = C^T b \Longleftrightarrow \widetilde{A}\widetilde{x} = \widetilde{b}$$

mit

$$\widetilde{A} = C^T AC, \quad \widetilde{x} = C^{-1} x, \quad \widetilde{b} = C^T b.$$

Die Matrix $\widetilde{A}$ ist wieder symmetrisch und positiv definit, so dass man dieses System mit dem cg-Verfahren lösen kann. Wir zeigen jetzt noch, dass man das cg-Verfahren so umformulieren kann, dass die Matrix C nicht benötigt wird. Das ist wichtig bei Vorkonditionierungsmatrizen B, bei denen eine solche Zerlegung nicht oder zumindest nicht billig berechnet werden kann. Dazu betrachten wir Algorithmus 7.7, wobei alle Größen mit $\widetilde{\ }$ versehen sind. Aus

$$\widetilde{x}_m = \widetilde{x}_{m-1} + \tau_m \widetilde{p}_m \Longleftrightarrow C^{-1}x_m = C^{-1}x_{m-1} + \tau_m \widetilde{p}_m$$

folgt durch Multiplikation mit C

$$x_m = x_{m-1} + \tau_m p_m \qquad \text{mit} \quad p_m = C\widetilde{p}_m.$$

Für die Residuen gilt

$$\widetilde{r}_m = \widetilde{b} - \widetilde{A}\widetilde{x}_m = C^T b - C^T ACC^{-1}x_m = C^T(b - Ax_m) = C^T r_m,$$

so dass

$$\widetilde{r}_m = \widetilde{r}_{m-1} - \tau_m \widetilde{A}\widetilde{p}_m \Longleftrightarrow C^T r_m = C^T r_{m-1} - \tau_m C^T ACC^{-1}p_m$$

durch Multiplikation mit C^{-T} zu

$$r_m = r_{m-1} - \tau_m A p_m$$

wird. Für die Suchrichtungen ist schließlich

$$\widetilde{p}_m = \widetilde{r}_{m-1} - \mu_m \widetilde{p}_{m-1} \Longleftrightarrow C^{-1}p_m = C^T r_{m-1} - \mu_m C^{-1}p_{m-1}.$$

Hier erhält man durch Multiplikation mit C

$$p_m = Br_{m-1} - \mu_m p_{m-1}.$$

Für die Koeffizienten ergibt sich

$$\begin{aligned}\delta'_m &= \widetilde{p}_m^T \widetilde{A}\widetilde{p}_m = (C^{-1}p_m)^T C^T ACC^{-1}p_m = p_m^T A p_m \\ \delta_{m+1} &= \widetilde{r}_m^T \widetilde{r}_m = (C^T r_m)^T C^T r_m = r_m^T B r_m.\end{aligned}$$

Mit $\mu'_m = -\mu_m$ ergibt sich Algorithms 7.9. Auch hier genügen Unterprogramme zur Berechnung von $x \mapsto Ax$ und $x \mapsto Bx$. Es ist nicht nötig, die Matrizen A und B explizit zu kennen.

Satz 7.16. *Angenommen, es existieren* $0 < \gamma \le \Gamma$ *mit*

$$\gamma v^T B^{-1} v \le v^T A v \le \Gamma v^T B^{-1} v \qquad \textit{für alle } v \in \mathbb{R}^n.$$

Ist $\widehat{x} = x^* = A^{-1}b$*, dann gilt die Fehlerabschätzung*

$$\|x_m - x^*\|_A \le 2\left(\frac{\sqrt{\widetilde{\kappa}} - 1}{\sqrt{\widetilde{\kappa}} + 1}\right)^m \|x_0 - x^*\|_A \qquad \textit{mit } \widetilde{\kappa} = \frac{\Gamma}{\gamma}.$$

Beweis. Für den Fehler gilt

$$\begin{aligned}\|x_m - x^*\|_A^2 &= (x_m - x^*)^T A(x_m - x^*) \\ &= \big(C(\widetilde{x}_m - \widetilde{x}^*)\big)^T AC(\widetilde{x}_m - \widetilde{x}^*) \\ &= (\widetilde{x}_m - \widetilde{x}^*)^T C^T AC(\widetilde{x}_m - \widetilde{x}^*) \\ &= (\widetilde{x}_m - \widetilde{x}^*)\widetilde{A}(\widetilde{x}_m - \widetilde{x}^*) \\ &= \|\widetilde{x}_m - \widetilde{x}^*\|_{\widetilde{A}}^2.\end{aligned}$$

Die Behauptung folgt damit aus Satz 7.14 und Satz 7.15 (angewendet auf $\widetilde{A}\widetilde{x} = \widetilde{b}$) unter Beachtung von $\kappa(\widetilde{A}) \le \widetilde{\kappa}$ (Übung). □

Algorithmus 7.9 Vorkonditioniertes Verfahren der konjugierten Gradienten

Wähle $x_0 \in \mathbb{R}^n$ und $B \in \mathbb{R}^{n,n}$ symmetrisch und positiv mit $B \approx A^{-1}$.
Setze $r_0 = b - Ax_0$ und wähle eine Toleranz $\texttt{tol} > 0$;
Initialisiere $p_1 = Br_0$, $\delta_1 = r_0^T Br_0$, $m = 1$;
while $\delta_m \geq \texttt{tol}^2$ **do**
  Berechne $\delta'_m = p_m^T A p_m$;
  Berechne $\tau_m = \delta_m / \delta'_m$
  Update: $x_m = x_{m-1} + \tau_m p_m$;
  Setze $r_m = r_{m-1} - \tau_m A p_m$;
  Berechne $\delta_{m+1} = r_m^H B r_m$; $\mu'_{m+1} = \delta_{m+1}/\delta_m$;
  Setze $p_{m+1} = Br_m + \mu'_m p_m$;
  $m = m + 1$;
end while