Skript zur Vorlesung

# Numerische Lineare Algebra

gehalten von Christian Mehl,
TU Berlin,
Sommersemester 2005

Das Skript ist entstanden aus einer Tex-Vorlesungsmitschrift von Lars Putzig

Version vom 2. September 2005

# Literatur zur Vorlesung

Die Vorlesung orientiert sich teilweise an folgenden Monographien:


- G. Golub, C. Van Loan. *Matrix computations*. Baltimore, 1996.
- Y. Saad. *Numerical methods for large eigenvalue problems*. Manchester, 1992.
- L. Trefethen, D. Bau. *Numerical linear algebra*. Philadelphia, 1997.
- D. Watkins. *Fundamentals of matrix computations*. New York, 2002.


Folgende weitere Werke sind zum Nachschlagen zu empfehlen:


- W. Bunse, A. Bunse-Gerstner. *Numerische Lineare Algebra*. Stuttgart, 1985.
- J. Demmel. *Applied numerical linear algebra*. Philadelphia, 1997.
- N.J. Higham. *Accuracy and stability of numerical algorithms*. Philadelphia, 2002.
- A. Meister. *Numerik linearer Gleichungssysteme*. Braunschweig, 1999.
- G.W. Stewart. *Matrix algorithms*. Philadelphia, 1998-2001, 2 Bände.
- G.W. Stewart, J.G. Sun. *Matrix perturbation theory*. Boston, 1990.

# Kapitel 0

# Motivation

Die Hauptprobleme der Numerischen Linearen Algebra lassen sich i.w. in drei große Klassen teilen:

(a) Eigenwertprobleme:

$$Ax = \lambda x, x \in \mathbb{C}^n, \lambda \in \mathbb{C}$$

(b) Lineare Gleichungssysteme:

$$Ax = b, x \in \mathbb{C}^n, b \in \mathbb{C}^n$$

mit nichtsingulärem $A$.

(c) Überbestimmte LGS und daraus folgend: Least squares-Probleme.

Wir konzentrieren uns in dieser Vorlesung auf die Probleme (a) und (b). Entsprechend bezeichne $A$ für den Rest der Veranstaltung eine Matrix $A \in \mathbb{C}^{n \times n}$, falls nichts anderes gesagt wird.
Anwendungen:

(a) $\dot{x} = Ax$ mit dem Ansatz $x(t) = x_0 e^{\lambda t}$ führt auf das Eigenwertproblem:

$$\lambda x_0 e^{\lambda t} = A e^{\lambda t} x_0$$

(b) viele ... (vgl. Einf. in die Numerische Mathematik)

Wir begegnen hier zwei typischen Situationen:

- $A$ ist klein und voll besetzt ($n = 10^2, 10^3$)
- $A$ groß, aber sparse (schwach besetzt) ($n = 10^6, \ldots$)

| | $A$ klein | $A$ groß |
|---|---|---|
| EWP | QR-Algorithmus | Lanczos, Arnoldi, Jacobi-Davidson |
| LGS | LR-Zerlegung, QR-Zerlegung | CG, GMRES |

# Kapitel 1

# Matrixtheorie

## 1.1 Grundlagen

### 1.1.1 Eigenwerte und Eigenvektoren

Sei $A \in \mathbb{C}^{n\times n}$, dann heißen $v \in \mathbb{C}^n \setminus \{0\}, \lambda \in \mathbb{C}$, die die Gleichung

$$Av = \lambda v$$

erfüllen Eigenvektor bzw. Eigenwert von $A$. Die Menge

$$\sigma(A) := \{\lambda \in \mathbb{C} | \lambda \text{ Eigenwert von } A\}$$

heißt dann Spektrum von $A$.

### 1.1.2 Matrixnormen

$A \in \mathbb{C}^{m\times n}$, dann ist

$$||A||_p := \sup_{x\neq 0} \frac{||Ax||_p}{||x||_p}$$

die Matrix $p$-Norm, $p \in \mathbb{N} \cup \{\infty\}$.

$$\kappa_p(A) := ||A||_p \cdot ||A^{-1}||_p$$

heißt für $m = n$ Konditionszahl.
**Spezialfälle:**

(a) $p = 1 \rightsquigarrow$ Spaltensummennorm:

$$||A||_1 = \max_j \sum_{i=1}^{m} |a_{ij}|$$

(b) $p = \infty \rightsquigarrow$ Zeilensummennorm:

$$||A||_\infty = ||A^T||_1$$

(c) $p = 2 \rightsquigarrow$ Spektralnorm

$$||A||_2 = \text{ Wurzel des größten Eigenwertes von } A^*A$$

mit $A^* = \bar{A}^T$, manchmal auch $A^H$ statt $A^*$.

**Vereinbarung:**

$$||A|| = ||A||_2, \kappa(A) = \kappa_2(A)$$

**Weitere Norm: Frobeniusnorm:**

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

### 1.1.3 Isometrische und unitäre Matrizen

**Definition:** *Sei $U \in \mathbb{C}^{n \times k}, k \leq n$.*

*(a) $U$ heißt isometrisch, falls $U^*U = I_k$ (Einheitsmatrix)*

*(b) $U$ heißt unitär, wenn $U$ isometrisch ist und $n = k$.*

**Satz 1.1** *Sei $U \in \mathbb{C}^{n \times k}, k \leq n$. Dann sind äquivalent:*

*(a) $U$ ist isometrisch*

*(b) die Spalten von $U$ sind orthonormal*

*(c) $\langle Ux, Uy \rangle = \langle x, y \rangle \ \forall x, y \in \mathbb{C}^n$ ($\langle \cdot, \cdot, \rangle$: Standart-Skalarprodukt)*

*(d) $||Ux|| = ||x|| \forall x \in \mathbb{C}^n$*

*Ist $k = n$, so sind (a)-(d) äquivalent zu:*

*(e) $UU^* = I_n$*

*(f) $U^{-1} = U^*$*

*(g) die Zeilen von $U$ sind orthonormal*

*In diesem Fall gilt außerdem:*

$$||U|| = 1 = ||U^{-1}|| = \kappa(U)$$

### 1.1.4 Unterräume

**Definition:** *$\mathcal{U} \subset \mathbb{C}^n$ heißt Untervektorraum, falls für alle $x, y \in \mathcal{U}, \alpha \in \mathbb{C}$ gilt:*

$$x + y \in \mathcal{U}, \alpha x \in \mathcal{U}.$$

**Satz 1.2** *Sei $\mathcal{U} \subset \mathbb{C}^n$ ein Unterraum mit Basis $(x_1, \ldots, x_m)$ und $X = [x_1, \ldots, x_m]$, d.h. rang$(X) = m$. Dann gilt:*

*(a) $\mathcal{U} = \mathcal{R}(X) := \{Xy | y \in \mathbb{C}^n\}$ (Spaltenraum von $X$, Bild von $X$, engl. range)*

*(b) Sei $Y \in \mathbb{C}^{n \times m}$ mit rang$(Y) = m$. Dann gilt:*

$$\mathcal{R}(X) = \mathcal{R}(Y) \Leftrightarrow X = YB, B \in \mathbb{C}^{m \times m}$$

*Insbesondere ist dann $B$ invertierbar und dementsprechend:*

$$XB^{-1} = Y$$

*(c) Das Gram-Schmidt-Verfahren für $(x_1, \ldots, x_m)$ liefert eine Orthonormalbasis $(q_1, \ldots, q_m)$ von $\mathcal{U}$ mit*

$$Span\{q_1, \ldots, q_j\} = Span\{x_1, \ldots, x_j\}$$

*für $j = 1, \ldots, m$. Diese Bedingung ist äquivalent zu:*

*Es gibt eine obere Dreiecksmatrix $R \in \mathbb{C}^{m \times m}$ mit $X = QR$ wobei $Q = [q_1, \ldots, q_m]$. (QR-Zerlegung)*

**Beweis:** Zu (c):
$X = QR$ mit

$$R = \begin{bmatrix} r_{11} & \ldots & r_{1m} \\ 0 & \ddots & \ddots \\ 0 & \ldots & r_{mm} \end{bmatrix}, X = [x_1, \ldots, x_m], Q = [q_1, \ldots, q_m]$$

$x_1 = q_1 r_{11} = r_{11} q_1$ und weiter:

$$\begin{aligned} x_2 &= q_1 r_{12} + q_2 r_{22} \\ &\vdots \\ x_j &= q_1 r_{1j} + \ldots + q_j r_{jj} \end{aligned}$$

□

### 1.1.5 Invariante Unterräume

**Definition:** *Sei $A \in \mathbb{C}^{n\times m}$ und $\mathcal{U} \subset \mathbb{C}^n$. Dann heißt $\mathcal{U}$ A-invariant, falls*

$$x \in \mathcal{U} \Rightarrow Ax \in \mathcal{U}\, \forall x \in \mathbb{C}^n$$

*Falls $\mathcal{U} \neq \{0\}, \mathbb{C}^n$ so heißt $\mathcal{U}$ nicht-trivial.*

**Satz 1.3** *Sei $A \in \mathbb{C}^{n\times n}, X \in \mathbb{C}^{n\times n}$ mit rang$(X) = k$ und $\mathcal{U} = \mathcal{R}(X)$. Dann sind äquivalent:*

*(a) $\mathcal{U}$ ist A-invariant*

*(b) Es gibt eine Matrix $B \in \mathbb{C}^{k\times k}$, so dass erfüllt ist:*

$$AX = XB$$

*Ferner gilt in diesem Fall für $\lambda \in \mathbb{C}$ und $v \in \mathbb{C}^k$:*

$$Bv = \lambda v \Rightarrow AXv = \lambda Xv$$

*Also ist jeder Eigenwert von B auch ein Eigenwert von A.*

**Bemerkung:** $A, X, B$ wie oben, $AX = XB$ ist formal ähnlich zur charakteristischen Gleichung:

$$Ax = x\lambda$$

somit kann man die Matrix $X$ als "verallgemeinerten Eigenvektor" auffassen. $B$ ist dann ein "Eigenwertpaket".

## 1.2 Matrix-Zerlegung

### 1.2.1 Schur-Zerlegung

**Satz 1.4 (Schur, 1909)**
*Sei $A \in \mathbb{C}^{n\times n}$, dann gibt es $U \in \mathbb{C}^{n\times n}$ unitär, so dass*

$$T := U^* AU$$

*eine obere Dreiecksmatrix ist.*

**Beweis:** per Induktion: $n = 1$ trivial.
"$n-1 \Rightarrow n$": Sei $v \in \mathbb{C}^n$ Eigenvektor von $A$ zum Eigenwert $\lambda \in \mathbb{C}$. Sei $q_1 := \frac{v}{||v||}$ und ergänze $q_1$ zu einer Orthonormalbasis $(q_1, \ldots, q_n)$ von $\mathbb{C}^n$. Dann ist $Q = [q_1, \ldots, q_m]$ unitär und es gilt:

$$Q^{-1}AQ = \left[\begin{array}{c|c} \lambda & A_{12} \\ \hline 0 & \\ \vdots & A_{22} \\ 0 & \end{array}\right]$$

Nach Induktionsvoraussetzung gibt es ein $U_{22}$ unitär, so dass $T_{22} := U_{22}^* A_{22} U_{22}$ in oberer Dreiecksform ist. Setze

$$U = Q \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & U_{22} \end{array}\right]$$

Dann ist $T = U^* A U$ eine obere Dreiecksmatrix. □

**Bemerkung:** $U$ kann so gewählt werden, dass die Eigenwerte von $A$ in bel. Reihenfolge auf der Diagonalen erscheinen.

**Definition:** *$T \in \mathbb{R}^{n\times n}$ heißt quasi-obere Dreiecksmatrix, falls $T$ eine Block-obere-Dreiecksmatrix ist und die Diagonalblöcke höchstens die Größe $2 \times 2$ haben.*

**Satz 1.5 (Murnaghan, Wintner, 1931)**
*Sei $A \in \mathbb{R}^{n\times n}$. Dann gibt es eine orthogonale Matrix $Q$, so dass*

$$T = Q^T A Q$$

*eine quasi-obere Dreiecksmatrix ist.*

**Beweis:** analog zum Satz von Schur
**Skizze:** Falls $A$ einen reellen Eigenvektor hat, verfahre wie im komplexen Fall. Andernfalls sei $v = v_1 + iv_2$ ein komplexer Eigenvektor zum komplexen Eigenwert $\lambda = \lambda_1 + i\lambda_2$, wobei $v_1, v_2 \in \mathbb{R}^n$ und $\lambda_1, \lambda_2 \in \mathbb{R}, \lambda_2 \neq 0$. Wegen

$$A\,(v_1 + iv_2) = (\lambda_1 + i\lambda_2)\,(v_1 + iv_2)$$

folgt:

$$\left.\begin{array}{rcl} Av_1 &=& \lambda_1 v_1 - \lambda_2 v_2 \\ Av_2 &=& \lambda_1 v_2 + \lambda_2 v_1 \end{array}\right\} \Rightarrow A\,[v_1\, v_2] = [v_1\, v_2] \left[\begin{array}{cc} \lambda_1 & \lambda_2 \\ -\lambda_2 & \lambda_1 \end{array}\right]$$

Somit ist $\mathrm{Span}\,\{v_1, v_2\}$ ein $A$-invarianter Unterraum. Sei $(q_1, q_2)$ eine ONB von $\mathrm{Span}\,\{v_1, v_2\}$ und $Q = [q_1, q_2, q_3, \ldots, q_n]$ orthogonal, dann gilt:

$$Q^* A Q = \left[\begin{array}{cc|c} \lambda_1 & \lambda_2 & \\ -\lambda_2 & \lambda_1 & A_{12} \\ \hline \multicolumn{2}{c|}{0} & A_{22} \end{array}\right]$$

Weiter wie im komplexen Fall. □

**Definition:** *$A \in \mathbb{C}^{n\times n}$ heißt normal, falls $AA^* = A^*A$.*

**Beispiel:** hermitische Matrizen, schief-hermitische Matrizen, unitäre Matrizen

**Satz 1.6** *Sei $A \in \mathbb{C}^{n\times n}$ normal und $U \in \mathbb{C}^{n\times n}$ unitär, so dass*

$$U^*AU = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$$

*Dann gilt: $A_{12} = 0$*

**Beweis:** Übung □

**Folgerung:** Sei $A \in \mathbb{C}^{n\times n}$ normal, so gibt es $U \in \mathbb{C}^{n\times n}$ unitär, so dass $U^*AU$ diagonal ist.

**Beweis:** Übung □

### 1.2.2 Die Singulärwert-Zerlegung

**Beobachtung:** Das Bild der Einheitsspähre unter einer beliebigen Matrix $A$ ist ein Hyperellipsoid.
$(m = n)$:

$\sigma_1, \ldots, \sigma_n$**:** seien die Längen der Hauptachsen, der Länge nach geordnet $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$.

$u_1, \ldots, u_n$**:** seien die Einheitsvektoren in Richtung der Hauptachsen. $u_i \perp u_j$ für $i \neq j$.

$v_1, \ldots, v_n$**:** seien die Urbilder von $u_1, \ldots, u_n$ unter $A$. Auch hier gilt $v_i \perp v_j$.

Somit gilt also:

$$A\,[v_1, \ldots, v_n] = [u_1, \ldots, u_n] \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \end{bmatrix}$$

**Satz 1.7 (Singulärwertzerlegung, SVD)**
*Sei $A \in \mathbb{C}^{m\times n}$ mit Rang $(A) = r$. Dann gibt es unitäre Matrizen $U \in \mathbb{C}^{m\times m}$ und $V \in \mathbb{C}^{n\times n}$, so dass:*

$$A = U\Sigma V^*, \Sigma = \left[\begin{array}{ccc|c} \sigma_1 & & & \\ & \ddots & & 0 \\ & & \sigma_r & \\ \hline & 0 & & 0 \end{array}\right] \in \mathbb{C}^{m\times n}$$

*Ferner: $\sigma_1 = ||A||_2$ und $\sigma_1, \ldots, \sigma_r$ sind eindeutig bestimmt.*

**Definition:** *Seien* $A, U = [u_1, \ldots, u_m], V = [v_1, \ldots, v_n], \Sigma$ *wie oben und* $\sigma_k := 0$ *für* $k = r+1, \ldots, \min\{m,n\}$.

*(a)* $\sigma_1, \ldots, \sigma_{\min\{m,n\}}$ *heißen die Singulärwerte von A.*

*(b)* $u_1, \ldots, u_m$ *heißen linke Singulärvektoren von A.*

*(c)* $v_1, \ldots, v_n$ *heißen rechte Singulärvektoren von A.*

**Bemerkung:**

(a) Es gilt:

$$A^*A = V\Sigma^*U^*U\Sigma V^* = V\Sigma^*\Sigma V^* = V\Sigma^2 V^*$$

bzw.

$$AA^* = U\Sigma V^*V\Sigma^*U^* = U\Sigma\Sigma^*U^* = U\Sigma^2U^*$$

d.h. $\sigma_1^2, \ldots, \sigma_r^2$ sind die von Null verschiedenen Eigenwerte von $AA^*$ bzw. $A^*A$.

(b) Wegen $AV = U\Sigma$ gilt: Kern $(A) = \text{Span}\{v_{r+1}, \ldots, v_n\}$ und
Bild $(A) = \mathcal{R}(A) = \text{Span}\{u_1, \ldots, u_r\}$.

(c) Neben der "vollen" SVD gibt es noch eine "reduzierte" SVD. z.B. für $m \geq n$:

$$A = \hat{U}\hat{\Sigma}V^*$$

mit $\hat{U} = [u_1, \ldots, u_n] \in \mathbb{C}^{m\times n}$ isometrisch und $\hat{\Sigma} \in \mathbb{C}^{n\times n}$ quadratisch. Analog für $m \leq n$.

(d) Die SVD erlaubt eine optimale Niedrigrang-Approimation an $A$, wegen

$$\begin{aligned} A &= U\Sigma V^* \\ &= U\left(\begin{bmatrix} \sigma_1 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix} + \ldots + \begin{bmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & \sigma_n \end{bmatrix}\right) V^* \\ &= \sum_{j=1}^{r} \sigma_j u_j v_j^* \end{aligned}$$

dabei ist $u_j v_j^*$ jeweils eine Rang-1-Matrix der Größe $m \times n$. Für $0 \leq \nu \leq r$ ist

$$A_\nu := \sum_{i=1}^{\nu} \sigma_i u_j v_i^*$$

die "beste" Rang-$\nu$-Approximation an $A$ in folgendem Sinn:

$$||A - A_\nu|| = \inf_{\substack{B \in \mathbb{C}^{m \times n} \\ \text{Rang } (B) \leq \nu}} ||A - B|| = \sigma_{\nu+1}$$

(dabei ist $\sigma_{r+1} := 0$).

(e) Anwendungen der SVD:

- numerische Rangbestimmung
- Datenkompression z.B. in der Bildverarbeitung

## 1.3 Störungstheorie

**Frage:** Wie "gut" sind berechnete Eigenwerte, Eigenvektoren und invariante Unterräume?

### 1.3.1 Kanonische Winkel und Vektoren

**Problem:** $\mathcal{U}, \mathcal{V} \subset \mathbb{C}^n$ seien Unterräume der Dimension $k$. Wie "nah" sind $\mathcal{U}$ und $\mathcal{V}$?
**Strategie:** Berechne sukzessive die Winkel zwischen $\mathcal{U}$ und $\mathcal{V}$ beginnende mit dem kleinsten. Wähle also normierte Vektoren $x \in \mathcal{U}$ und $y \in \mathcal{V}$, so dass:

$$|\langle x, y \rangle| \stackrel{!}{=} \max.$$

O.B.d.A. können wir $x$ und $y$ so wählen, dass das Skalarprodukt reell und nichtnegativ ist. (Andernfalls wählen wir $z \in \mathbb{C}$ mit $|z| = 1$, so dass $\langle zx, y \rangle = z \langle x, y \rangle$ reell und nicht-negativ ist. Dann gilt

$$|\langle x, y \rangle| = |\langle zx, y \rangle|.)$$

(a) Wähle $x_1 \in \mathcal{U}$ und $y_1 \in \mathcal{V}$ mit $||x_1|| = ||y_1|| = 1$, so dass

$$\langle x_1, y_1 \rangle = \max \{\text{Re} \langle x, y \rangle \,|\, x \in \mathcal{U}, y \in \mathcal{V}, ||x|| = ||y|| = 1\}$$

Dann ist $\langle x_1, y_1 \rangle$ reell, $\vartheta_1 = \langle x_1, y_2 \rangle$ heißt 1-ter kanonischer Winkel und $x_1, y_1$ heißen 1-te kanonische Vektoren.

(b) Angenommen wir haben nun $j-1$-Richtungen und Winkel bestimmt, d.h.:

$$x_1, \ldots, x_{j-1} \in \mathcal{U}, y_1, \ldots, y_{j-1} \in \mathcal{V}$$

sind schon bestimmt, wobei $(x_1, \ldots, x_{j-1})$ und $(y_1, \ldots, y_{j-1})$ orthonormal sind.

Wähle $x_j \in \mathcal{U}$ und $y_j \in \mathcal{V}$ mit $x_j \perp x_1, \ldots, x_{j-1}$ und $y_j \perp, y_1, \ldots, y_{j-1}$ und $||x_j|| = ||y_i|| = 1$, so dass

$$\langle x_j, y_j \rangle$$

maximalen Realteil hat. Dann ist $\langle x_j, y_j \rangle$ reell und

$$\vartheta_j := \arccos \langle x_j, y_j \rangle$$

heißt $j$-ter kanonischer Winkel, $x_j, y_j$ heißen $j$-te kanonische Vektoren. Dies liefert $k$ kanonische Winkel $0 \leq \vartheta_1 \leq \ldots \leq \vartheta_k \leq \frac{\pi}{2}$ und Orthonormalbasen $(x_1, \ldots, x_n)$ und $(y_1, \ldots, y_k)$ von $\mathcal{U}$ bzw. $\mathcal{V}$.

**Lemma 1.8** *Für $i, j = 1, \ldots, k$ und $i \neq j$ gilt: $\langle x_i, y_j \rangle = 0$.*

**Beweis:** Übung □

**Folgerung:** Sei $X = [x_1, \ldots, x_k]$ und $Y = [y_1, \ldots, y_k]$. Dann gilt:

$$X^*Y = (\langle x_i, y_j \rangle) = \begin{bmatrix} \cos\vartheta_1 & & 0 \\ & \ddots & \\ 0 & & \cos\vartheta_k \end{bmatrix}$$

mit $\cos\vartheta_1 \geq \ldots \geq \cos\vartheta_k \geq 0$.
Dies ist eine SVD.

**Praktische Berechnung der kanonischen Winkel und Vektoren:**

(a) Bestimme Orthonormalbasen vn $\mathcal{U}$ und $\mathcal{V}$ bzw. isometrische Matrizen $P, Q \in \mathbb{C}^{n \times k}$ mit

$$\mathcal{R}(P) = \mathcal{U}, \mathcal{R}(Q) = \mathcal{V}$$

(b) Berechne die SVD von $P^*Q$:

$$P^*Q = U\Sigma V^*$$

mit der Diagonalmatrix $\Sigma$.

$$\Rightarrow \Sigma = \underbrace{U^*P^*}_{X^*} \underbrace{QV}_{Y}$$

(c) Setze $U = [u_1, \ldots, u_k]$ und $V = [v_1, \ldots, v_k]$. Dann gilt:

(a) $\vartheta = \arccos\sigma_j, j = 1, \ldots, k$ sind die kanonischen Winkel

(b) $Pu_j, Qv_j, j = 1, \ldots, k$ sind kanonische Vektoren.

### 1.3.2 Abstand von Unterräumen

**Definition:** $\mathcal{U}, \mathcal{V} \in \mathbb{C}^n$ *Unterräume der Dimension* $k$.

*(a) Für* $x \in \mathcal{U}$ *heißt*

$$d(x, \mathcal{V}) := \min_{y \in \mathcal{V}} ||x - y||$$

*der Abstand von* $x$ *zu* $\mathcal{V}$.

*(b)*

$$d(\mathcal{U}, \mathcal{V}) := \max_{\substack{x \in \mathcal{U} \\ ||x||=1}} d(x, \mathcal{V})$$

*heißt Abstand von* $\mathcal{U}$ *und* $\mathcal{V}$.

**Satz 1.9** *Seien* $\mathcal{U}, \mathcal{V} \subset \mathbb{C}^n$ *Unterräume der Dimension* $k$ *mit kanonischen Winkeln* $\vartheta_1 \leq \ldots \leq \vartheta_k$. *Dann gilt:*

$$d(\mathcal{U}, \mathcal{V}) = \sin \vartheta_k$$

**Beweis:** siehe z.B. Stewart/Sun (siehe Website) □

# Kapitel 2

# Eigenwertprobleme mit voll besetzten Matrizen

**Situation:** $A \in \mathbb{C}^{n\times n}$, $n$ "klein", $A$ ist voll besetzt.
Wir können:

- Die Matrix konventionell speichern.
- Die Matrix manipulieren (durch Ähnlichkeitstransformationen)

## 2.1 Die Potenzmethode

Idee: $q \in \mathbb{C}^n \setminus \{0\}$. Bilde $q, Aq, A^2q, \ldots$. Was passiert dann?

**Annahme:** $A$ ist diagonalisierbar. Seien $\lambda_1, \ldots, \lambda_n$ mit $|\lambda_1| \geq \ldots \geq |\lambda_n|$ die Eigenwerte von $A$ und $(v_1, \ldots, v_n)$ eine zugehörige Basis aus Eigenvektoren. Dann gibt es $c_1, \ldots, c_n$ mit:

$$q = c_1 v_1 + \ldots + c_n v_n$$

Weitere Annahme: $c_1 \neq 0$ (denn $c_1 = 0$ wäre ein „großer Zufall"), dann ist:

$$\begin{aligned} Aq &= c_1\lambda_1 v_1 + \ldots + c_n\lambda_n v_n \\ A^k q &= c_1\lambda_1^k v_1 + \ldots + c_n\lambda_n^k v_n \end{aligned}$$

Für $|\lambda_1| > 1$ wird $|\lambda_1^k|$ immer größer, daher skalieren wir:

$$\frac{1}{\lambda_1^k} A^k q = c_1 v_1 + c_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k v_2 + \ldots + c_n \left(\frac{\lambda_n}{\lambda_1}\right)^k v_n$$

3.Annahme: $|\lambda_1| > \lambda_2 \geq \ldots \geq |\lambda_n|$
Dann gilt:

$$\begin{aligned} \left\| \frac{1}{\lambda_1^k} A^k q - c_1 v_1 \right\| &\leq |c_2| \left|\frac{\lambda_2}{\lambda_1}\right|^k ||v_2|| + \ldots + |c_n| \left|\frac{\lambda_n}{\lambda_1}\right|^k ||v_n|| \\ &\leq \left(|c_2|||v_2|| + \ldots + |c_n|||v_n||\right) \left|\frac{\lambda_2}{\lambda_1}\right|^k \overset{k\to\infty}{\longrightarrow} 0 \end{aligned}$$

also $\lim\limits_{k\to\infty} \frac{1}{\lambda_1^k} A^k q = c_1 v_1$ und die Konvergenz ist linear mit Konvergenzrate $r \leq |\frac{\lambda_2}{\lambda_1}|$

**Definition:** *Eine Folge* $(x_k)$ *konvergiert linear gegen* $x$*, falls es* $r$ *mit* $0 < r < 1$ *gibt, so dass*

$$\lim_{k\to\infty} \frac{||x_{k+1} - x||}{||x_k - x||} = r.$$

$r$ *heißt dann Konvergenzrate der Folge.*

In der Praxis is $1/\lambda_1^k$ unbekannt. Daher normieren wir die betragsgrößte Komponente von $A^k q$ auf eins. Dies liefert:

**Algorithmus: (Potenzmethode)**
Berechnet den dominanten Eigenwert $\lambda_1$ und den zugehörigen Eigenvektor $v_1$.

(a) Wähle $q_0 \in \mathbb{C}^n \setminus \{0\}$

(b) Iteriere, für $k = 1, 2, \ldots$ bis Konvergenz:

$$q_k := \frac{1}{\alpha_k} A q_{k-1},$$

wobei $\alpha_k$ die betragsgrößte Komponente von $Aq_{k-1}$ ist.

Wir haben im wesentlichen bewiesen:

**Satz 2.1** *$A \in \mathbb{C}^{n\times n}$ habe die Eigenwerte $\lambda_1, \ldots, \lambda_n$ mit $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|$. Falls $q_0 \in \mathbb{C}^n \setminus \{0\}$ eine Komponente im zu $\lambda_1$ gehörigen invarianten Unterraum hat („$c_1 \neq 0$“), so konvergiert die im obigen Algorithmus definierte Folge $(q_k)$ gegen einen Eigenvektor assoziiert mit $\lambda_1$. Die Konvergenz ist linear mit Rate $r \leq |\frac{\lambda_2}{\lambda_1}|$. Ferner konvergiert die Folge $(\alpha_k)$ gegen den Eigenwert $\lambda_1$.*

**Bemerkung:**

(a) Der Satz gilt auch für nicht-diagonalisierbare Matrizen.

(b) Bilden des Produkts $Aq_k$ kostet $2n^2$ flops. Die Skalierungsoperation kostet $O(n)$ flops. Somit kosten $m$ Iterationen ca. $2n^2m$ flops.

(c) Im Allgemeinen ist $|\frac{\lambda_2}{\lambda_1}| \approx 1$ und die Konvergenz ist extrem langsam.

## 2.2 Shift-and-Invert und Rayleigh-Quotient-Iteration

**Beobachtungen:** Sei $A \in \mathbb{C}^{n\times n}$ und $(\lambda, v) \in \mathbb{C} \times \mathbb{C}^n$ mit $Av = \lambda v$. Dann gilt:

(a) $A^{-1}v = \lambda^{-1}v$ für $A$ invertierbar.

(b) $(A - \varrho I)v = (\lambda - \varrho)v$ für alle $\varrho \in \mathbb{C}$.

Seien $\lambda_1, \ldots, \lambda_n$ die Eigenwerte von $A$ mit $|\lambda_1| \geq \ldots \geq |\lambda_n|$.

**Inverse Iteration:**
Potenzmethode für $A^{-1}$. Falls $|\lambda_n| < |\lambda_{n-1}|$, dann konvergiert die inverse Iteration gegen einen Eigenvektor zu $\lambda_n$ mit Konvergenzrate $|\frac{\lambda_n}{\lambda_{n-1}}|$ (sehr klein, falls $|\lambda_n| \ll |\lambda_{n-1}|$).

**Shift and Invert:** Potenzmethode für $(A - \varrho I)^{-1}$:
Seien $\lambda_j, \lambda_k$ die Eigenwerte, die am dichtesten an $\varrho$ liegen und $|\lambda_j - \varrho| < |\lambda_k - \varrho|$. Dann konvergiert die Potenzmetotde für $(A - \varrho I)^{-1}$ gegen einen Eigenvektor zu $\lambda_j$ mit Rate

$$\left|\frac{\lambda_j - \varrho}{\lambda_k - \varrho}\right|$$

(klein, falls $|\lambda_j - \varrho| \ll |\lambda_k - \varrho|$); optimal wäre: $\lambda_j \approx \varrho$.

**Problem:** Wir brauchen "gute" Shifts! $\rightsquigarrow$ zwei Fragen:

**Frage 1:** Woher wissen wir, wann eine Eigenwertapproximation gut ist?

**Definition:** *Sei $A \in \mathbb{C}^{n\times n}$ und $(\mu, w) \in \mathbb{C} \times \mathbb{C}^n$. Dann heißt $Aw - \mu w$ das Residuum von $(\mu, w)$ bzgl. $A$.*

**Satz 2.2** *Seien $\mu \in \mathbb{C}, \varepsilon > 0$ und $A \in \mathbb{C}^{n\times n}$, sowie $w \in \mathbb{C}^n$ mit $||w|| = 1$. Ist $||Aw - \mu w|| = \varepsilon$, so gibt es eine Matrix $E \in \mathbb{C}^{n\times n}$ mit $||E|| \leq \varepsilon$, so dass*

$$(A + E)w = \mu w.$$

Merksatz: kleines Residuum $\Rightarrow$ kleiner Rückwärtsfehler in Eigenpaaren

**Beweis:** Sei $r := Aw - \mu w$ und $E = -rw^*$ Dann gilt:

$$(A + E)w = Aw - r\underbrace{w^*w}_{=1} = \mu w$$

$$\text{und} \quad ||E|| = ||rw^*|| \leq ||r||||w^*|| = ||r|| = \varepsilon.$$

□

**Frage 2:** Wie bestimmen wir Shifts? oder: Wie bestimmen wir eine gute Eigenwertapproximation aus einer gegebenen Eigenvektorapproximation?

**Idee:** Minimiere $||Aw - \mu w||$. Wir betrachten das überbestimmte LGS:

$$w\mu = Aw$$

mit der $n \times 1$-Matrix $w$, dem unbekannten $1 \times 1$-Vektor $\mu$ und der rechten Seite $Aw$. Aus der Einführung in die Numerik wissen wir (hoffentlich): Wir erhalten die Lösung des Problems „$||Aw - \mu w|| = \min!$“ durch Lösen der Normalengleichung:

$$w^*w\mu = w^*Aw \qquad \text{bzw.} \qquad \mu = \frac{w^*Aw}{w^*w}.$$

**Definition:** *Sei $A \in \mathbb{C}^{n\times n}$ und $w \in \mathbb{C}^n \setminus \{0\}$. Dann heißt*

$$r(w) := \frac{w^*Aw}{w^*w}$$

*der Rayleigh-Quotient von $w$ bzgl. $A$.*

Der folgende Satz liefert eine Abschätzung dafür, wie stark die Abweichung eines Rayleigh-Quotient von einem Eigenwert ist.

**Satz 2.3** *Sei $A \in \mathbb{C}^{n\times n}$ und $(\lambda, v) \in \mathbb{C} \times \mathbb{C}^n$ ein Eigenpaar von $A$ mit $\|v\| = 1$. Dann gilt für $w \in \mathbb{C}^n$ mit $\|w\| = 1$, dass*

$$|\lambda - r(w)| \leq 2\|A\| \cdot \|v - w\|.$$

**Situation:**

approximierter Eigenvektor
Rayleigh-Quotient ↓
approximierter Eigenwert
Shift-and-Invert ↓
approximierter Eigenvektor

Durch Kombination dieser beiden Techniken erhalten wir:

**Algorithmus: Rayleigh-Quotient-Iteration (RQI)**
Dieser Algorithmus berechnet ein Eigenpaar $(\lambda, v) \in \mathbb{C} \times \mathbb{C}^n$ zu einer Matrix $A \in \mathbb{C}^{n\times n}$.

(a) Start: Wähle ein $q_0 \in \mathbb{C}^n$ mit $||q_0|| = 1$ und setze $\lambda_0 := q_0^*Aq_+$.

(b) Iteriere: für $k = 1, 2, \ldots$ bis Konvergenz:

  (a) Löse das LGS $(A - \lambda_{k-1}I)x = q_{k-1}$ für $x$.
  (b) $q_k := \dfrac{x}{||x||}$
  (c) $\lambda_k := q_k^*Aq_k$

**Bemerkung:**

(a) Die Konvergenzanalyse ist schwierig, allerdings lässt sich beobachten, dass der Algorithmus fast immer konvergiert. Im Allgemeinen konvergiert er quadratisch, für Hermitesche Matrizen sogar kubisch.

**Erinnerung:** $(x_k) \to x$ konvergiert von der Ordnung $m \geq 2$ genau dann, wenn gilt:

$$\lim_{k\to\infty} \frac{||x_{k+1} - x||}{||x_k - x||^m} = c \neq 0$$

speziell spricht man bei $m = 2$ bzw. $m = 3$ auch von quadratischer bzw. kubischer Konvergenz.

(b) **Kosten:** $O(n^3)$ flops pro Schritt, da jeweils eine $LR$-Zerlegung berechnet werden muss. ($O(n^2)$ für Hessenbergmatrizen (siehe Kaptiel 2.4.2))

(c) $A - \lambda_{k-1}I$ ist "fast singulär", falls $\lambda_{k-1}$ nahe an einem Eigenwert ist, d.h. $A - \lambda_{k-1}I$ ist i.A. schlecht konditioniert. Wenn wir das LGS rückwärts stabil lösen, dann erfüllt die berechnete Lösung exakt:

$$(A + \Delta A - \lambda_{k-1}I)\hat{x} = q_{k-1}$$

mit $||\Delta A||$ klein. Die Kondition von Eigenwerten und Eigenvektoren hat allerdings i.A. nichts mit der Konditionszahl der Matrix zu tun, d.h. auch Matrizen mit großen Konditionszahlen können gut konditionierte Eigenwerte und Eigenvektoren haben. Falls Eigenwert und Eigenvektor gut konditioniert sind, können wir wegen des kleinen Rückwärtsfehlers also trotzdem durch die Rayleigh-Quotien-Iteration gute Approximation an einen Eigenvektor und einen Eigenwert erwarten.

(Exkurs: Kondition von Eigenwerten: Ist $\lambda$ ein einfacher Eigenwert von $A$, d.h. die algebraische Vielfachheit ist eins, und sind $v, w$ normierte Rechts- bzw. Linkseigenvektoren, d.h.

$$Av = \lambda v, \quad w^*A = \lambda w^*, \quad \|v\| = 1 = \|w\|,$$

so gilt für kleine Störungen $\Delta A$ in erster Näherung, dass $A + \Delta A$ einen Eigenwert $\lambda + \Delta\lambda$ hat mit

$$|\Delta\lambda| \leq \frac{1}{|w^*v|}\|\Delta A\|.$$

Also ist $1/|w^*v|$ eine Konditionszahl für einfache Eigenwerte. Für normale Matrizen gilt $v = w$, also $|w^*v| = 1$. Normale Matrizen haben also gut konditionierte Eigenwerte.)

## 2.3 Simultane Unterraumiteration

$A \in \mathbb{C}^{n\times n}$ mit Eigenwerten $\lambda_1, \ldots, \lambda_n$ mit $|\lambda_1| \geq \ldots \geq |\lambda_n|$.

**Idee:** Statt $q_0 \in \mathbb{C}^n$ betrachte eine linear unabhängige Menge $\{w_1, \ldots, w_m\} \subset \mathbb{C}^n$. Setze

$$W_0 := [w_1, \ldots, w_m] \in \mathbb{C}^{n\times m}$$

Bilde $W_0, AW_0, A^2W_0, \ldots$ bzw.

$$W_k := A^k W_0 = \left[A^k w_0, \ldots, A^k w_m\right], \quad k \geq 1$$

Vermutung: $\mathcal{R}(W_k) \to$ konvergiert gegen den invarianten Unterraum $\mathcal{U}$ assoziert mit den $m$ betragsgrößten Eigenwerten $\lambda_1, \ldots, \lambda_m$, falls kein Ausnahmezustand vorliegt (wie "$c_1 = 0$" bei der Potenzmethode). Dies nennt man simultane Unterraumiteration. (Warum „simultan" später!)

**Satz 2.4** *Sei $A \in \mathbb{C}^{n\times n}$ mit Eigenwerten $\lambda_1, \ldots, \lambda_n$, so dass*

$$|\lambda_1| \geq \ldots \geq |\lambda_m| > |\lambda_{m+1}| \geq \ldots \geq |\lambda_n|.$$

*Seien $\mathcal{U}, \mathcal{V}$ die invarianten Unterräume assoziiert mit $\lambda_1, \ldots, \lambda_m$ bzw. $\lambda_{m+1}, \ldots, \lambda_n$. Ferner sei $W \in \mathbb{C}^{n\times m}$ mit Rang $(W) = m$ und $\mathcal{R}(W) \cap \mathcal{V} = \{0\}$. Dann existiert für die Iteration $W_0 := W$ und $W_{k+1} = AW_k$ für $k \geq 0$ und für jedes $\varrho$ mit $\left|\frac{\lambda_{m+1}}{\lambda_m}\right| < \varrho < 1$ eine Konstante $c$, so dass*

$$d(\mathcal{R}(W_k), \mathcal{U}) \leq c \cdot \varrho^k, \quad k \geq 1.$$

Für den Beweis des Satzes brauchen wir das folgende technische Hilfsmittel:

**Lemma 2.5** *Seien*

$$\mathcal{U} = \mathcal{R}\left(\begin{bmatrix} I_m \\ 0 \end{bmatrix}\right) \quad \text{und} \quad \hat{\mathcal{U}} = \mathcal{R}\left(\begin{bmatrix} I_m \\ X \end{bmatrix}\right)$$

*$m$-dimensionale Unterräume von $\mathbb{C}^n$. ($X \in \mathbb{C}^{(n-m)\times m}$) und seien $\Theta_1, \ldots, \Theta_m$ die kanonischen Winkel zwischen $\mathcal{U}$ und $\hat{\mathcal{U}}$. Dann sind*

$$\tan\Theta_1, \ldots, \tan\Theta_n$$

*die Singulärwerte von $X$. Insbesondere gilt dann $||X|| = \tan\Theta_m$.*

**Beweis:** siehe Stewart/Sun. *Matrix perturbation theory.* Boston, 1990. □

**Beweis:** (des Satzes) für den Spezialfall, dass $A$ diagonalisierbar ist.
Koordinatentransformation:

$$A_{\text{neu}} = S^{-1} A_{\text{alt}} S = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$$

mit $A_1 = \text{diag}\,(\lambda_1, \ldots, \lambda_m)$ und $A_2 = \text{diag}\,(\lambda_{m+1}, \ldots, \lambda_n)$. Dann ist $A_1$ nichtsingulär, da $|\lambda_1| \geq \ldots \geq |\lambda_m| > 0$.

$$\mathcal{U}_{\text{neu}} = S^{-1}\mathcal{U}_{\text{alt}} = \mathcal{R}\left(\begin{bmatrix} I_m \\ 0 \end{bmatrix}\right)$$

und

$$\mathcal{V}_{\text{neu}} = S^{-1}\mathcal{V}_{\text{alt}} = \mathcal{R}\left(\begin{bmatrix} 0 \\ I_{n-m} \end{bmatrix}\right).$$

Weiter sei

$$W_{\text{neu}} = S^{-1}W_{\text{alt}} = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix}$$

für ein $Z_1 \in \mathbb{C}^{m\times m}$ und $Z_2 \in \mathbb{C}^{(n-m)\times m}$. Es gilt dann:

(a) $d(\mathcal{R}(W_{\text{neu}}),\mathcal{U}_{\text{neu}}) \leq \kappa(S)d(\mathcal{R}(W_{\text{alt}}),\mathcal{U}_{\text{alt}})$ (Übung)

(b) $\mathcal{R}(W_{\text{neu}}) \cap \mathcal{V} = \{0\} \Leftrightarrow Z_1$ ist nichtsingulär (Übung)

Im Folgenden lassen wir den Index "neu" weg. Dann gilt:

$$W = \begin{bmatrix} Z_1 \\ Z_2 \end{bmatrix} = \begin{bmatrix} I_m \\ Z_2Z_1^{-1} \end{bmatrix} Z_1 = \begin{bmatrix} I \\ X_0 \end{bmatrix} Z_1$$

mit $X_0 = Z_2Z_1^{-1}$, somit ist

$$\mathcal{R}(W) = \mathcal{R}\left(\begin{bmatrix} I_m \\ X_0 \end{bmatrix}\right)$$

und

$$\mathcal{R}(W_k) = \mathcal{R}(A^kW) = \mathcal{R}(A^k \begin{bmatrix} I \\ X_0 \end{bmatrix}).$$

Nun gilt:

$$A^k \begin{bmatrix} I \\ X_0 \end{bmatrix} = \begin{bmatrix} A_1^k & 0 \\ 0 & A_2^k \end{bmatrix} \begin{bmatrix} I \\ X_0 \end{bmatrix} = \begin{bmatrix} A_1^k \\ A_2^kX_0 \end{bmatrix} = \begin{bmatrix} I_m \\ \underbrace{A_2^kX_0A_1^{-k}}_{=:X_k} \end{bmatrix} A_1^k.$$

Also ist

$$\mathcal{R}(W_k) = \mathcal{R}\left(\begin{bmatrix} I_m \\ X_k \end{bmatrix}\right).$$

Z.z. bleibt:

$$d(\mathcal{R}(W_k),\mathcal{U}) \to 0.$$

Sei $\Theta_m^{(k)}$ der größte kanonische Winkel zwischen $\mathcal{R}(W_k)$ und $\mathcal{U}$. Dann gilt:

$$\begin{aligned} d(\mathcal{R}(W_k),\mathcal{U}) &= \sin\Theta_m^{(k)} \leq \tan\Theta_m^{(k)} = ||X_k|| \leq ||A_2^k||||X_0||||A_1^{-k}|| \\ &= |\lambda_{m+1}^k|||X_0|||\lambda_m^{-k}| \end{aligned}$$

Daraus folgt direkt:

$$d(R(W_k),\mathcal{U}) \leq \tilde{c} \left| \frac{\lambda_{m+1}}{\lambda_m} \right|^k$$

Machen wir nun die Koordinatentransformation rückgängig, dann erhalten wir die Form wie im Satz. (Für den diagonalisierbaren Fall brauchen wir die Schranke $\varrho$ aus dem Satz nicht. Diese kommt aber ins Spiel, wenn man nicht-diagonalisierbare Matrizen betrachtet.) □

**Bemerkung:** Für $W_0 = [w_1, \ldots, w_m]$ ist

$$A^k W_0 = \left[ A^k w_1, \ldots, A^k w_m \right],$$

d.h. wir führen die Unterraumiteration nicht nur für $W_0$ aus, sondern gleichzeitig auch für alle $W_0^{(j)} = [w_1, \ldots, w_j]$, denn

$$A^k W_0^{(j)} = \left[ A^k w_1, \ldots, A^k w_j \right].$$

Unter gewissen Vorraussetzungen (vgl. Sätze) konvergiert also:

$$\text{Span} \left\{ A^k w_1, \ldots, A^k w_j \right\}$$

gegen den invarianten Unterraum assoziiert mit $\lambda_1, \ldots, \lambda_j$ und zwar für alle $j = 1, \ldots, m$. Deswegen spricht man von „simultaner Unterraumiteration“.

**Problem:**

**Theorie:** $A^k W_0 = \left[ A^k w_1, \ldots, A^k w_m \right]$ hat i.A. immer Rang $m$ (falls nicht irgendwelche Ausnahmesituationen vorliegen).

**Praxis:** Für alle $i = 1, \ldots, m$ gilt (unter gewissen Vorrausetzungen):

$$\mathcal{R}(A^k w_i) \to \mathcal{R}(v_1)$$

wobei $v_1$ der dominante Eigenvektor ist. Durch Rundungsfehler fällt $\mathcal{R}(W_k)$ irgendwann zu $\mathcal{R}(v_1)$ zusammen.

Die Grundidee zurr Lösung des Problems ist: Orthonormalisiere die Basis des Spaltenraums in jedem Schritt.

**1. Schritt:** $W_0 = [w_1, \ldots, w_m] = Q_0 R_0$ mit $Q_0 \in \mathbb{C}^{n \times m}$ isometrisch und $R_0 \in \mathbb{C}^{m \times m}$ obere Dreiecksmatrix, dann ist $\mathcal{R}(W_0) = \mathcal{R}(Q_0)$. Wir haben sogar:

$$\text{Span} \{w_1, \ldots, w_j\} = \text{Span} \left\{ q_1^{(0)}, \ldots, q_j^{(0)} \right\}, \; j = 1, \ldots, m,$$

wobei $Q_0 = \left[ q_1^{(0)}, \ldots, q_m^{(0)} \right]$. Dies folgt aus der oberen Dreiecksform von $R_0$.

**Situation vor dem k-ten Schritt:** $\mathcal{R}(W_{k-1}) = \mathcal{R}(Q_{k-1})$ mit

$$Q_{k-1} = \left[q_1^{(k-1)}, \ldots, q_m^{(k-1)}\right] \in \mathbb{C}^{n\times m}$$

isometrisch und

$$\text{Span}\left\{A^{k-1}w_1, \ldots, A^{k-1}w_j\right\} = \text{Span}\left\{q_1^{(k-1)}, \ldots, q_j^{(k-1)}\right\}, \quad j = 1, \ldots, m.$$

**k-ter Schritt:** Sei nun $AQ_{k-1} = Q_kR_k$ eine $QR$-Zerlegung mit $Q_k \in \mathbb{C}^{n\times m}$ isometrisch und $R_k \in \mathbb{C}^{m\times m}$ obere $\Delta$-Matrix. Dann ist:

$$\mathcal{R}(W_k) = \mathcal{R}(AW_{k-1}) = \mathcal{R}(AQ_{k-1}) = \mathcal{R}(Q_k),$$

sogar:

$$\text{Span}\left\{A^{k}w_1, \ldots, A^{k}w_j\right\} = \text{Span}\left\{q_1^{(k)}, \ldots, q_j^{(k)}\right\}, \quad j = 1, \ldots, m.$$

**Algorithmus: Unitäre simultane Unterraumiteration**

(a) Start: Wähle $Q_0 \in \mathbb{C}^{n\times m}$ isometrisch.

(b) Iteriere, für $k = 1, 2, \ldots$ bis Konvergenz:

  (a) Berechne $Z_k = AQ_{k-1}$

  (b) Berechne eine $QR$-Zerlegung $Z_k = Q_kR_k$.

**Bemerkung:** Theoretisch hat die unitäre simultane Unterraumiteration dasselbe Konvergenzverhalten wie die simultane Unterraumiteration. Sie leidet aber nicht unter denselben praktischen Problemen.

## 2.4 Der QR-Algorithmus

### 2.4.1 Der einfache QR-Algorithmus ohne Shifts

$A \in \mathbb{C}^{n\times n}$ habe die Eigenwerte $\lambda_1, \ldots, \lambda_n$ wobei $|\lambda_1| \geq \ldots \geq |\lambda_n|$.

**Idee:** $n$-dimensionale unitäre QR-Iteration, d.h. wähle im obigen Algorithmus $m = n$ und $Q_0 = I_n$. Falls $Q_k = \left[q_1^{(k)}, \ldots, q_n^{(k)}\right]$, so konvergiert

$$\text{Span}\left\{q_1^{(k)}, \ldots, q_m^{(k)}\right\}$$

für jedes $1 \leq m \leq n$ gegen den invarianten Unterraum assoziiert mit $\lambda_1, \ldots, \lambda_m$ mit Rate $\left|\frac{\lambda_{m+1}}{\lambda_m}\right|$ falls $|\lambda_{m+1}| < |\lambda_m|$ und falls keine Ausnahmesituation vorliegt.

**Frage 1:** Wie machen wir die Konvergenz sichtbar?

**Antwort:** Bilde $A_k = Q_k^{-1} A Q_k$. Konvergiert $\mathrm{Span}\{q_1^{(k)}, \ldots, q_m^{(k)}\}$ gegen einen invarianten Unterraum, so erwarten wir, dass in der Matrix

$$A_k = \begin{matrix} \\ m \\ n-m \end{matrix} \begin{matrix} \begin{matrix} m & n-m \end{matrix} \\ \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \end{matrix}$$

der Block $A_{21}$ für $k \to \infty$ gegen Null konvergiert. Dies passiert aber für viele $m$ gleichzeitig, d.h. $A_k$ „nähert sich einer oberen Dreiecksmatrix“.

**Frage 2:** Kommen wir direkt von $A_{k-1}$ zu $A_k$?

**Antwort:** Es gilt:

$$\begin{aligned} A_{k-1} &= Q_{k-1}^{-1} A Q_{k-1} \\ A_k &= Q_k^{-1} A Q_k \end{aligned}$$

$$\Rightarrow A_k = Q_k^{-1} Q_{k-1} A_{k-1} \underbrace{Q_{k-1}^{-1} Q_k}_{=:U_k} = U_k^{-1} A_{k-1} U_k$$

Damit können wir den k-ten Schritt der unitären Unterraumiteration

$$A Q_{k-1} = Q_k R_k$$

wie folgt umformulieren:

$$A_{k-1} = Q_{k-1}^{-1} A Q_{k-1} = Q_{k-1}^{-1} Q_k R_k = U_k R_k$$

Dies ist eine $QR$-Zerlegung von $A_{k-1}$. Außerdem gilt:

$$A_k = U_k^{-1} A_{k-1} U_k = U_k^{-1} U_k R_k U_k = R_k U_k$$

Zusammenfassend erhalten wir den folgenden

**Algorithmus: (QR-Algorithmus)**(Francis 1961)
Dieser Algorithmus erzeugt zu einer gegebenen Matrix $A \in \mathbb{C}^{n \times n}$ eine Folge $(A_k)$ von ähnlichen Matrizen, die “sich einer oberen Dreiecksmatrix nähern”.

(a) Starte mit $A_0 = A$

(b) Iteriere, für $k = 1, 2, \ldots$ bis Konvergenz:

  (a) Berechne eine QR-Zerlegung von $A_{k-1}$: $A_{k-1} = U_k R_k$

  (b) Berechne $A_k$ durch: $A_k = R_k U_k$

**Satz 2.6 (Konvergenz des QR-Algorithmus)**
*Sei $A \in \mathbb{C}^{n\times n}$ mit den Eigenwerten $\lambda_1, \ldots, \lambda_n$, wobei $|\lambda_1| \geq \ldots \geq |\lambda_m| > |\lambda_{m+1}| \geq \ldots \geq |\lambda_n|$. Sei $\mathcal{V} \subset \mathbb{C}^n$ der invariante Unterraum assoziiert mit $\lambda_{m+1}, \ldots, \lambda_n$, sowie $(A_k)$ die vom QR-Algorithmus erzeugte Folge von Matrizen. Falls*

$$Span\{e_1, \ldots, e_m\} \cap \mathcal{V} = \{0\}$$

*und*

$$A_k = \begin{matrix} \\ m \\ n-m \end{matrix} \begin{matrix} \begin{matrix} m & n-m \end{matrix} \\ \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ A_{21}^{(k)} & A_{22}^{(k)} \end{bmatrix} \end{matrix}$$

*so existiert zu jedem $\varrho$ mit $\left|\frac{\lambda_{m+1}}{\lambda_m}\right| < \varrho < 1$ eine Konstante $\tilde{c}$, so dass:*

$$||A_{21}^{(k)}|| \leq \tilde{c}\varrho^k.$$

**Beweis:**(Skizze) Sei $\mathcal{U}$ der invariante Unterraum assoziiert mit $\lambda_1, \ldots, \lambda_m$ und

$$\mathcal{U}_k = \text{Span}\left\{q_1^{(k)}, \ldots, q_m^{(k)}\right\}$$

wobei

$$Q_k = \left[q_1^{(k)}, \ldots, q_n^{(k)}\right]$$

die unitäre Matrix mit $Q_k^{-1}AQ_k = A_k$ aus der unitären Unterraumiteration ist. Mit etwas Aufwand zeigt man:

$$||A_{21}^{(k)}|| \leq 2\sqrt{2}||A||d(\mathcal{U}, \mathcal{U}_k)$$

Dann folgt mit dem Satz über die simultane URaumiteration die Existenz einer Konstanten $c > 0$ mit

$$d(\mathcal{U}, \mathcal{U}_k) \leq c\varrho^k.$$

Wähle nun $\tilde{c} := 2\sqrt{2}||A||c$. □

**Spezialfall:** Ist $A$ hermitisch, so konvergiert $(A_k)$ unter den gegebenen Vorraussetzungen gegen eine Diagonalmatrix.

**Bemerkung:** In dieser Form hat der Algorithmus zwei wesentliche Nachteile:

(a) Der Algorithmus ist sehr teuer: $O(n^3)$ flops pro Iterationsschritt:

- QR-Zerlegung $\approx \frac{4}{3}n^3$ flops.
- Matrix/Matrix-Produkt $\approx 2n^3$ flops

Abhilfe: $\rightsquigarrow$ 2.4.2

(b) Die Konvergenz ist sehr langsam, da nur linear.
Abhilfe: $\rightsquigarrow$ 2.4.3

### 2.4.2 Hessenberg-Reduktion

**Definition:** *Eine Matrix $A = (a_{ij})$ heißt Hessenbergmatrix oder in Hessenbergform, falls $a_{ij} = 0$ für $i > j + 1$. Die Hessenbergmatrix $A$ heißt unreduziert, falls $a_{i+1,i} \neq 0$ für alle $i = 1, \ldots, n-1$.*

**Erinnerung: Householder-Transformationen**
haben die Form:

$$P = I - \frac{2}{v^* v} v v^*$$

für $v \in \mathbb{C}^n \setminus \{0\}$ und sind Hermitesch und unitär. (Übung.)
**Wirkung:** $P$ spiegelt $x \in \mathbb{C}^n$ an der Hyperebene $\text{Span}\{v\}^\perp$. (Übung.)
**Ziel:** Spiegele $x \in \mathbb{C}^n \setminus \{0\}$ auf ein Vielfaches des 1. Einheitsvektors, d.h. finde $v$ und berechne $P$, so dass:

$$Px = \pm ||x|| e_1$$

**Berechnung von $v$:** Der Ansatz $v = x + \alpha e_1$ liefert:

$$v = x \pm ||x|| e_1 \qquad \text{und} \qquad Px = \mp ||x|| e_1. \qquad \text{(Übung.)}$$

Aus numerischen Gründen wähle

$$v = \begin{cases} x + ||x|| e_1, & x_1 \geq 0, \\ x - ||x|| e_1, & x_1 < 0. \end{cases}$$

**Vorteil:** "billige Produkte". (Übung.) Für $B \in \mathbb{C}^{n \times m}$ benötigt die Berechnung von $PB$ nur ca. $4mn$ flops (statt $2n^2 m$ flops bei herkömmlicher Matrix/Matrix-Multiplikation). **Anwendung:**

(a) Berechnung der QR-Zerlegung
(siehe `http://www.math.tu-berlin.de/~mehl/lehre/05ss-nla/qr.pdf`).

(b) Hessenberg-Reduktion: Zu $A \in \mathbb{C}^{n \times n}$ berechnen wir $U$ unitär, so dass $U^{-1} A U$ in Hessenbergform ist. Kosten: ca. $\frac{10}{3} n^3$ flops
(siehe `http://www.math.tu-berlin.de/~mehl/lehre/05ss-nla/hessenberg.pdf`).

**Bemerkung:**

(a) Ist $H \in \mathbb{C}^{n \times n}$ eine Hessenbergmatrix, so lässt sich eine QR-Iteration mit Givens-Rotationen in $O(n^2)$ flops durchführen.

Givensrotation:

$$\hat{G}(c, s) = \begin{bmatrix} c & s \\ -\bar{s} & \bar{c} \end{bmatrix}$$

mit $|c|^2 = |s|^2 = 1$. Sind $c$ und $s$ so gewählt, dass $-\bar{s} a_{11} + \bar{c} a_{21} = 0$, so erhalten wir:

$$\hat{G}(c, s) \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} * & * \\ 0 & * \end{bmatrix}$$

(b) Hessenbergmatrizen sind invariant unter QR-Iterationen
(siehe
`http://www.math.tu-berlin.de/~mehl/lehre/05ss-nla/qr_iteration.pdf`)

**Frage:** Ist die Hessenbergreduktion eindeutig? Nein, aber es gilt der folgende Satz:

**Satz 2.7 (Implizites Q-Theorem)**
*Sei $A \in \mathbb{C}^{n\times n}$ und seien $Q = [q_1, \ldots, q_n], U = [u_1, \ldots, u_n]$ unitäre Matrizen, so dass:*

$$H = Q^{-1}AQ = (h_{ij}) \qquad \text{und} \qquad G = U^{-1}AU = (g_{ij})$$

*Hessenbergmatrizen sind. Gilt $q_1 = u_1$ und ist $H$ unreduziert, so gilt:*

$$q_i = c_i u_i$$

*für ein $c_i \in \mathbb{C}$ mit $|c_i| = 1$ und $|h_{i,i-1}| = |g_{i,i-1}|$ für $i = 2, \ldots, n$, d.h. $Q$ ist im wesentlichen durch die erste Spalte $q_1$ schon eindeutig bestimmt.*

**Beweis:** Übung □

### 2.4.3 Der QR-Algorithmus mit Shifts

**Deflation:** Sei $H \in \mathbb{C}^{n\times n}$ in Hessenbergform. Falls $H$ unreduziert ist, d.h. falls $h_{m+1,m} = 0$ für ein $m$, so gilt:

$$H = \begin{matrix} \\ m \\ n-m \end{matrix} \begin{matrix} \begin{matrix} m & n-m \end{matrix} \\ \begin{bmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{bmatrix} \end{matrix}$$

d.h. unser Problem zerfällt in zwei kleinere Teilprobleme: $H_{11}, H_{22}$.

**Ziel:** Führe Deflation herbei mit Hilfe von Shifts.

**Algorithmus (QR-Algorithmus mit Hessenbergreduktion und Shifts):**
Gegeben: $A \in \mathbb{C}^{n\times n}$:

(a) Berechne $U_0$ unitär, so dass:

$$H := U_0^* A U_0$$

in Hessenbergform ist. o.B.d.A. sei $H_0$ unreduziert (andernfalls tritt Deflation ein und wir betrachten ein kleineres Teilproblem).

(b) Iteriere, für $k = 1, 2, \ldots$ bis Deflation eintritt, d.h.:

$$h^{(k)}_{m+1,m} = O(\texttt{eps})$$

für ein $m$:

(i) Wähle einen Shift $\mu_k \in \mathbb{C}$.
(ii) Berechne eine $QR$-Zerlegung $H_{k-1} - \mu_k I = Q_k R_k$ von $H_{k-1} - \mu_k I$.
(iii) Bilde $H_k = R_k Q_k + \mu_k I$.

Dabei entsprechen Schritt (ii) und (iii) gerade einer QR-Iteration für $A - \mu_k I$.

**Bemerkung:**

(a) Der Subdiagonaleintrag $h^{(k)}_{m+1,m}$ in $H_k$ konvergiert mit Rate $\left|\frac{\lambda_{m+1}-\mu_k}{\lambda_m-\mu_k}\right|$ gegen Null.

(b) Falls $h^{(k)}_{m+1,m} = 0$ oder $h^{(k)}_{m+1,m} = O(eps)$, so findet Deflation statt. Fahre mit den zwei kleineren Teilproblemen fort.

(c) Falls $\mu_k$ ein Eigenwert ist, so findet wegen (a) schon nach einem Schritt irgendwo (für ein $m$) Deflation statt.

**Shiftstrategien:**

(a) Rayleigh Quotient-Shift:

Für den Speziallfall, dass $A$ Hermitesch ist, konvergiert $A_k$ im QR-Algorithmus gegen eine Diagonalmatrix. Insbesondere ist dann $q^{(k)}_n$ eine Approximation an einen Eigenvektor. Eine zugehörige gute Eigenwertapproximation ist der Rayleigh-Quotient

$$r(q^{(k)}_n) = (q^{(k)}_n)^* A q^{(k)}_n.$$

Dies ist aber gerade der n-te Diagonaleintrag $a^{(k)}_{n,n}$ von $Q^*_k A Q_k$.

**Heuristisk:** Auch im allgemeinen Fall vermuten wir $a^{(k)}_{n,n}$ als gute Approximation an einen Eigenwert. Wähle also einfach:

$$\mu_k = a^{(k)}_{n,n}$$

Damit konvergiert $h^{(k)}_{n,n-1}$ i.A. quadratisch gegen Null.

(b) Wilkonson-Shift:

Probleme ergeben sich beispielsweise bei der Matrix:

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

Eine QR-Iteration für $A_0 = A$ liefert: $Q_0 = A_0, R_0 = I$ damit ist aber auch:

$$R_0 Q_0 = I A_0 = A_0,$$

d.h. der Algorithmus stagniert. Der Rayleigh Quotient-Shift wäre hier $\mu = 0$, ändert also nichts an der Stagnation des Algortihmus. Darum betrachten wir (für $A \in \mathbb{C}^{n\times n}$) im k-ten Schritt die Submatrix $B$ gegeben durch

$$A_k = \begin{matrix} \\ n-2 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} n-2 & \quad 2 \end{matrix} \\ \begin{bmatrix} * & * \\ * & B \end{bmatrix} \end{matrix}.$$

Nun wähle $\mu_k$ als den Eigenwert von $B$, der näher an $a^{(k)}_{nn}$ liegt.

(c) Weitere Strategien: z.B. Doppelshifts $\rightsquigarrow$ 2.4.4.

**Bemerkung:** Meist werden nur etwa zwei Iterationen benötigt um entweder einen $1 \times 1$ oder $2 \times 2$ Block abzuspalten.

### 2.4.4 Implizite Shifts und "Bulge-Chasing"

Sei $H \in \mathbb{C}^{n\times n}$ eine Hessenbergmatrix und $\mu_1, \ldots, \mu_l \in \mathbb{C}$. Führe $l$ Schritte des QR-Algorithmus mit Shifts $\mu_1, \ldots, \mu_l$ aus:

$$\begin{aligned} H - \mu_1 I &= Q_1 R_1 \\ H_1 &= R_1 Q_1 + \mu_1 I \\ &\vdots \\ H_{l-1} - \mu_l I &= Q_l R_l \\ H_l &= R_l Q_l + \mu_l I \end{aligned}$$

Dann gilt

$$H_l = Q_l^* Q_l R_l Q_l + \mu_l Q_l^* Q_l = Q_l^*(Q_l R_l + \mu_l I) Q_l = Q_l^* H_{l-1} Q_l$$

und somit per Induktion

$$= Q_l^* \ldots Q_1^* H \underbrace{Q_1 \ldots Q_l}_{=:Q} = Q^* H Q.$$

**Frage:** Können wir $Q$ direkt berechnen, ohne $l$ $QR$-Iterationen durchzuführen?

**Lemma 2.8** $M := (H - \mu_l I)\ldots(H - \mu_1 I) = Q_1 \ldots Q_l \underbrace{R_l \ldots R_1}_{=:R} = QR.$

**Beweis:** per Induktion nach $j$ zeigen wir:

$$(H - \mu_j I)\ldots(H - \mu_1 I) = Q_1 \ldots Q_j R_j \ldots R_1, \quad j = 1, \ldots, l.$$

$j = 1$**:** Hier lautet die Aussage des Lemmas $H - \mu_1 I = Q_1 R_1$. Dies ist gerade der erste Schritt des QR-Algorithmus.

$j - 1 \to j$**:**

$$\begin{aligned} & Q_1 \ldots Q_j R_j \ldots R_1 \\ = \;& Q_1 \ldots Q_{j-1}(H_{j-1} - \mu_j I) R_{j-1} \ldots R_1 \\ = \;& Q_1 \ldots Q_{j-1}\Big(Q_{j-1}^* \ldots Q_1^* H Q_1 \ldots Q_{j-1} - \mu_j I\Big) R_{j-1} \ldots R_1 \\ = \;& (H - \mu_j I) Q_1 \ldots Q_{j-1} R_{j-1} \ldots R_1 \\ \overset{I.V.}{=} \;& (H - \mu_j I)(H - \mu_{j-1} I)\ldots(H - \mu_1 I) \end{aligned}$$

□

**Gute Idee:** Berechne $M$ und dann die Householder-QR-Zerlegung von $M$, d.h. $M = QR$, und setze

$$\tilde{H} = Q^* R Q = H_l.$$

Wir benötigen also nur eine QR-Zerlegung statt $l$ QR-Zerlegungen in der QR-Iteration. Andererseits müssen wir $M$ berechnen, d.h. $l - 1$ Matrix-Matrix-Multiplikationen.

**Bessere Idee:** Berechne $\tilde{H}$ direkt aus $H$ unter Benutzung des impliziten $Q$-Theorems.

**Implizite Shift-Strategie:**

(a) Bereche
$$Me_1 = (H - \mu_l I) \dots (H - \mu_1 I) e_1,$$
d.h. die erste Spalte von $M$. Dabei sind nur die ersten $l+1$ Einträge von Null verschieden. Falls $l$ nicht zu groß ist (z.B. $l = 2, \dots, 6$) kostet dies nur $O(1)$ flops.

(b) Bestimme eine Householdermatrix $P_0$, so dass $P_0(Me_1)$ ein Vielfaches von $e_1$ ist. Transformiere $H$ mit $P_0$:
$$\begin{array}{rcl} & & \begin{array}{cc} l+1 & n-l-1 \end{array} \\ P_0 & = & \begin{array}{c} l+1 \\ n-l-1 \end{array} \begin{bmatrix} * & 0 \\ 0 & I \end{bmatrix} \\ & & \begin{array}{cc} l+2 & n-l-2 \end{array} \\ P_0 H P_0 & = & \begin{array}{c} l+2 \\ n-l-2 \end{array} \begin{bmatrix} * & * \\ 0 & \hat{H} \end{bmatrix} \end{array}$$
$P_0$ verändert nur die Zeilen und Spalten $1, \dots, l+1$ von $H$. Daher entsteht eine Hessenbergmatrix mit Buckel (engl. bulge), d.h. $\hat{H}$ ist immer noch in Hessenbergform.

(c) Bestimme Householdermatrizen $P_1, \dots, P_{n-2}$ um die Hessenbergform zu restaurieren: **Bulge chasing:** Wir jagen den Buckel die Diagonale hinab!
(siehe `http://www.math.tu-berlin.de/~mehl/lehre/05ss-nla/bulgechasing.pdf`)
$$\rightsquigarrow \tilde{H} := P_{n-2} \dots P_1 P_0 H P_0 \dots P_{n-2}$$
ist wieder in Hessenbergform und $P_k e_1 = e_1$ für $k = 1, \dots, n-2$.

(d) $P_0$ hat die gleiche erste Spalte wie $Q$, denn wenn wir $Q$ mit der Householder QR-Zerlegung bestimmen, berechnen wir im ersten Schritt $P_0$. Da jeweils $P_k e_1 = e_1$ ist, hat nun auch
$$P_0 P_1 \dots P_{n-2}$$
dieselbe erste Spalte wie $P_0$ bzw. $Q$. Mit dem Impliziten Q-Theorem müssen nun $Q$ und $P_0 \dots, P_{n-2}$ und damit auch $\tilde{H}$ und $Q^* H Q$ im wesentlichen gleich sein. Wir haben also (i.w.) $\tilde{H}$ bzw. $H_l$ direkt aus $H$ berechnet.

**Algorithmus (QR-Algorithmus mit impliziter Doppelshift-Strategie)**: (Francis 1961)
Gegeben $A \in \mathbb{C}^{n \times n}$:

(a) Bestimme $U_0$ unitär, so dass $H_0 := U_0^* A U_0$ in Hessenbergform ist.

(b) Iteriere, für $k = 1, 2, \dots$ bis Konvergenz (Deflation):

  (a) Berechne die Eigenwerte $\mu_1, \mu_2$ des rechten unteren $2 \times 2$ Blocks von $H_{k-1}$.

  (b) Berechne mit der impliziten Shiftstrategie für $l = 2$ die Matrix $\tilde{Q}_k$, die man nach 2 Schritten des QR-Algorithmus mit Shifts $\mu_1, \mu_2$ erhalten würde.

  (c)
$$H_k := \tilde{Q}_k^* H_{k-1} \tilde{Q}_k$$

**Bemerkung:**

(a) Empirische Gesamtkosten für die Berechnung aller Eigenwerte von $A$: ca. $10n^3$ flops, falls auch die Transformationsmatrix $Q$ benötigt wird ca. $25n^3$ flops.

(b) Konvergenzanalyse schwierig, kein globaler Konvergenzbeweis bekannt

(c) Das Verfahren funktioniert auch im Reellen mit reeller Arithmetik.

# Kapitel 3

# Eigenwertprobleme mit großen, schwach-besetzten Matrizen

**Situation**: Gegeben sei eine Matrix $A \in \mathbb{C}^{n\times n}$ mit $n$ sehr groß (z.B.: $n \approx 10^6, 10^7, \ldots$). Dabei sei $A$ "sparse", d.h. schwach-besetzt. $A$ hat also nur wenige von Null verschiedene Einträge. MATLAB verfügt bspw. über die Datenstruktur `sparse`: Eine Matrix $\tilde{A} \in \mathbb{C}^{m\times n}$ wird durch 6 Parameter beschrieben:

(a) $m$: Anzahl der Zeilen

(b) $n$: Anzahl der Spalten

(c) $nnz$: Anzahl der von Null verschiedenen Elemente

(d) $a$: Liste der von Null verschiedenen Einträge

(e) $irow$: Liste der zugehörigen Zeilenindizes

(f) $jcol$: Liste der zugehörigen Spaltenindizes

- Wir können: zu $x \in \mathbb{C}^n$ das Produkt $Ax$ berechnen;
  (Dies geht oft mit $O(n)$ flops statt $O(n^2)$ flops);

- Wir können nicht: Ähnlichkeitstransformationen anwenden (da die transformierte Matrix dann i.A. nicht mehr „sparse“ ist);

Manchmal ist $A$ auch gar nicht explizit gegeben, sondern nur durch eine Subroutine, die zu gegebenem $x \in \mathbb{C}^n$ das Produkt $Ax$ berechnet.

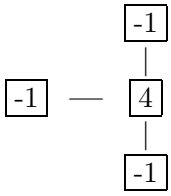
$$x \longrightarrow \boxed{\text{black box}} \rightarrow Ax$$

Dies ist dann die einzige Möglichkeit, um Informationen über die sonst unbekannte Matrix $A$ zu erhalten.

**Beispiel:**

(a) Diskretisierter Laplace-Operator auf einem uniformen Gitter

Zu einer gegebenen Funktion $u(x, y)$ erhalten wir auf einem zweidimensionalen Gitter an jeden Gitterpunkt $(i, j)$ Approximationen $u_{ij} = u(x_i, y_j)$ für, sagen wir, $i = 1, \ldots, m$ und $j = 1, \ldots, n$. Gemäß des Differenzenstern berechnet sich eine Approximation an $v = \Delta u$ im Gitterpunkt $(i, j)$ wie folgt:




$$vij = 4u_{ij} - u_{i,j+1} - u_{i,j-1} - u_{i+1,j} - u_{i-1,j},$$

wobei ggf. noch Randwerte hinzugenommen werden. Dies lässt sich auch als Matrix-Gleichung schreiben:

$$\begin{aligned} v &= [v_{11}, \ldots, v_{1n}, v_{21}, \ldots, v_{2n}, \ldots, v_{m1}, \ldots, v_{mn}] = Au \\ u &= [u_{11}, \ldots, u_{1n}, u_{21}, \ldots, u_{2n}, \ldots, u_{m1}, \ldots, u_{mn}] \end{aligned}$$

Anstatt $A$ explizit anzugeben, reicht es, eine Subroutine zu schreiben, die $v = Au$ aus $u$ berechnet, z.B.:

```
for i=1,...,m
  for j=1,...,n
    v[i,j]=4*u[i,j]-u[i,j+1]-u[i,j-1]-u[i+1,j]-u[i-1,j]
  end
end
```

(b) Toeplitzmatrizen:

$$A = \begin{bmatrix} a_0 & a_{-1} & \ldots & a_{-n} \\ a_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_n & \ldots & a_1 & a_0 \end{bmatrix}$$

$A$ ist eindeutig bestimmt durch den Zeilenvektor

$$[a_n, \ldots, a_1, a_0, a_{-1}, \ldots, a_{-n}]$$

und es reicht, diesen Vektor abzuspeichern. Wir können wieder Subroutinen schreiben, die zu gegebenem $x$ das Produkt $Ax$ berechnen.

**Frage**: Wie berechnen wir Eigenwerte und Eigenvektoren von $A$? (Oder zumindest: Wie berechnen wir *einige* Eigenpaare von $A$? Denn i.A. sind wir nur an einigen Eigenwerten interessiert, z.B. den betragsgrößten, und es wird i.A. auch zu teuer sein, alle Eigenpaare der Matrix zu berechnen.)

**1. Idee**: Simultane unitäre Unterraum-Iteration (populär um 1970)

- Starte mit $m \ll n$ orthonormalen Vektoren $q_1^{(0)}, \ldots, q_m^{(0)}$ und setze $Q_0 = \left[q_1^{(0)}, \ldots, q_m^{(0)}\right]$.
- Berechne $AQ_{k-1} = \tilde{Q}_k$, also $m$-Matrix Vektor Produkte. Dies kostet bei sparsen Matrizen i.A. nur $O(mn)$ flops statt $O(mn^2)$ bei herkömmlichen Matrizen.
- Berechne eine $QR$-Zerlegung:
$$\tilde{Q}_k = Q_k R_k,$$
wobei $Q_k \in \mathbb{C}^{n\times m}$ isometrisch und $R_k \in \mathbb{C}^{m\times m}$ eine obere Dreiecksmatrix ist.

Erfahrungswerte: Diese Methode ist nur bedingt geeignet.

**2. Idee**: Projektionsmethoden: Benutze, das für Paare $(\lambda, v) \in \mathbb{C} \times \mathbb{C}^n \setminus \{0\}$ gilt:

$$(\lambda, v)\,\text{Eigenpaar} \quad \Leftrightarrow \quad Av - \lambda v = 0, \quad \Leftrightarrow \quad Av - \lambda v \perp \mathbb{C}^n$$

(a) Konstruiere einen Unterraum $\mathcal{K} \subseteq \mathbb{C}^n$, den sogenannten Suchraum.

(b) Wähle einen zweiten Unterraum $\mathcal{L} \subseteq \mathbb{C}^n$, den sogenannten Testraum. Dann bestimme Paare $(\tilde{\lambda}, \tilde{v}) \in \mathbb{C} \times \mathcal{K}$ mit

$$A\tilde{v} - \tilde{\lambda}\tilde{v} \perp \mathcal{L}. \qquad (\text{Oft gilt } \mathcal{L} = \mathcal{K}.)$$

Hoffnung: Einige der Paare $(\tilde{\lambda}, \tilde{v})$ sind gute Approximationen an Eigenpaare von $A$.

## 3.1 Krylov-Räume

**Frage**: Wann enthält ein Unterraum $\mathcal{K}$ gute Approximationen an Eigenvektoren von $A$?

Zur Motivation betrachten wir dazu zuerst einen Spezialfall: $A \in \mathbb{R}^{n\times n}$ symmetrisch mit Eigenwerten $\lambda_1 \geq \ldots \geq \lambda_n$. Dann gilt:

$$\lambda_1 = \max_{x\neq 0} r(x) \quad \text{und} \quad \lambda_n = \min_{x\neq 0} r(x), \qquad \text{wobei} \quad r(x) = \frac{x^T Ax}{x^T x} \quad (\text{Übung})$$

Sei nun $\mathcal{R}(Q_k)$ durch $Q_k = [q_1, \ldots, q_k] \in \mathbb{R}^{n\times k}$ gegeben und

$$M_k := \max_{x\in\mathcal{R}(Q_k)\setminus\{0\}} r(x) = \max_{y\neq 0} \frac{y^T Q_k^T A Q_k y}{y^t Q_k^T Q_k y},$$

sowie analog

$$m_k := \min_{x\in\mathcal{R}(Q_k)\setminus\{0\}} r(x).$$

Klar: $\lambda_1 \geq M_k \geq m_k \geq \lambda_k$. Insbesondere ist $M_k$ eine Approximation an $\lambda_1$ und $m_k$ eine Approximation an $\lambda_n$. Unser Ziel ist nun, die Vektoren $q_1, q_2, \ldots, q_k, \ldots$ so zu konstruieren, dass $M_k$ und $m_k$ möglichst schnell $\lambda_1$ und $\lambda_n$ approximieren, d.h. es soll gelten:

$$M_k \to \lambda_1 \quad \text{und} \quad m_k \to \lambda_n \quad \text{möglichst schnell.}$$

Seien $q_1, \ldots, q_k$ schon konstruiert und seien $u_k, w_k \in \text{Span}\{q_1, \ldots, q_k\}$ so gewählt, dass

$$M_k = r(u_k) \quad \text{und} \quad m_k = r(w_k).$$

$r(x)$ wächst am stärksten in Richtung des Gradienten:

$$\nabla r(x) = \frac{2}{x^T x}\Big(Ax - r(x)x\Big) \qquad \text{(Übung 2 Aufgabe 6)}$$

Also gilt $M_{k+1} > M_k$, falls $\nabla r(u_k) \neq 0$ und

$$\nabla r(u_k) \in \text{Span}\{q_1, \ldots, q_k, q_{k+1}\}. \quad (*)$$

Analog: $m_{k+1} < m_k$ falls $\nabla r(w_k) \neq 0$ und

$$\nabla r(w_k) \in \text{Span}\{q_1, \ldots, q_k, q_{k+1}\}. \quad (**)$$

Wegen $\nabla r(x) \in \text{Span}\{x, Ax\}$ sind die Bedingungen $(*)$ und $(**)$ erfüllt, falls:

$$\begin{aligned}
\text{Span}\{q_1, q_2\} &= \text{Span}\{q_1, Aq_1\} \\
\text{Span}\{q_1, q_2, q_3\} &= \text{Span}\{q_1, Aq_1, A^2q_1\} \\
&\vdots \\
\text{Span}\{q_1, \ldots, q_{k+1}\} &= \text{Span}\Big\{q_1, Aq_1, A^2q_1, \ldots, A^kq_1\Big\}
\end{aligned}$$

**Definition:** *Sei $A \in \mathbb{C}^{n\times n}$ und $x \in \mathbb{C}^n$, sowie $l \in \mathbb{N}$.*

*(a)* $K_l(A,x) := \big[x, Ax, A^2x, \ldots, A^{l-1}x\big]$ *heißt* Krylovmatrix *bzgl. $A$ und $x$.*

*(b)* $\mathcal{K}_l(A,x) := \mathcal{R}(K_l(A,x)) = \text{Span}\big\{x, Ax, A^2x, \ldots, A^{l-1}x\big\}$ *heißt* Krylovraum *bzgl. $A$ und $x$.*

Wir haben eben gesehen, dass für symmetrische Matrizen $A \in \mathbb{R}^{n\times n}$ Krylovräume schnell Approximationen an die Eigenwerte $\lambda_1$ und $\lambda_n$, also die Eigenwerte am Rand des Spektrums enthalten. Wir vermuten, dass das auch für beliebige Matrizen $A \in \mathbb{C}^{n\times n}$ gilt:

**Heuristik**: Krylovräume sind „gute“ Suchräume!

Im Folgenden leiten wir einige Eigenschaften von Krylovräumen her. Insbesondere stellen wir Zusammenhänge mit dem Minimalpolynom eines Vektors bzgl. unserer Matrix $A$ und mit Hessenbergmatrizen her.

**Erinnerung**: Seien $A \in \mathbb{C}^{n\times n}, x \in \mathbb{C}^n$, dann gibt es ein eindeutig bestimmtes normiertes Polynom kleinsten Grades mit

$$0 = p(A)\,x = A^m x + \alpha_{m-1}A^{m-1}x + \ldots + \alpha_1 Ax + \alpha_0 x.$$

$p$ heißt dann Minimalpolynom von $x$ bzgl. $A$.

**Lemma 3.1** *Seien $A \in \mathbb{C}^{n\times n}$, $x \in \mathbb{C}^n$ und sei $\nu$ der Grad des Minimalpolynoms von $x$ bzgl. $A$. Dann gilt:*

*(a)* $\dim K_m(A,x) = m \iff m \leq \nu$.

*(b)* $K_\nu(A,x)$ *ist $A$-invariant.*

*(c)* $K_m(A,x) = K_\nu(A,x)$ *für* $m \geq \nu$

**Beweis:** Übung □

**Lemma 3.2** *Seien $A \in \mathbb{C}^{n\times n}$ und $g_1 \in \mathbb{C}^n$, so dass $g_1, Ag_1, \ldots, A^{m-1}g_1$ linear unabhängig sind. Sei $G = [g_1, g_2, \ldots, g_n]$ nicht singulär. Ferner sei $B = G^{-1}AG = (b_{ij})$. Dann sind folgende Aussagen äquivalent:*

*(a)* $b_{jk} = 0$ *für* $k = 1, \ldots, m-1$ *und* $j = k+2, \ldots, n$, *d.h.*

$$B = \begin{bmatrix} b_{11} & \ldots & \ldots & \ldots & \ldots & b_{1n} \\ b_{21} & & & & & \vdots \\ 0 & \ddots & & & & \vdots \\ \vdots & \ddots & b_{m,m-1} & b_{mm} & \ldots & b_{mn} \\ \vdots & & 0 & \vdots & \ddots & \vdots \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & b_{n,m} & \ldots & b_{nn} \end{bmatrix}.$$

*(b)* $Span\{g_1, \ldots, g_l\} = K_l(A, g_1)$ *für* $l = 1, \ldots, m$

*Ist eine der Bedingungen erfüllt, und ist $m = n$, so ist $B$ insbesondere eine unreduzierte Hessenbergmatrix.*

**Beweis:** Übung □

## 3.2 Der Arnoldi-Algorithmus

Gegeben sei $A \in \mathbb{C}^{n\times n}$. Dann haben wir die folgenden Zerlegungen kennen gelernt:

| $QR$-Zerlegung | Hessenbergreduktion |
|---|---|
| $A = QR$ | $A = QHQ^*$ |
| Householder | Householder |
| Gram-Schmidt | *neu: Arnoldi* |

Das Prinzip hinter der Transformation mittels Householder-Matrizen nennen wir „orthogonales Strukturieren“, denn die Struktur der Matrix wird schrittweise erzeugt durch Anwendung von orthogonalen Transformationen. Das Prinzip hinter dem Gram-Schmidt-Verfahren ist dagegen ein „strukturiertes Orthogonalisieren“, denn die orthogonale Transformationsmatrix wird schrittweise erzeugt, indem wir Kenntnisse über die gewünschte Matrixstruktur (hier: obere Dreiecks-Struktur) ausnutzen. Arnoldi entwickelte nun einen Algorithmus, der auf ähnliche Weise die Hessenbergreduktion berechnet, indem auch hier die Transformationsmatrix $Q$, die $A$ auf Hessenbergform transformiert, schrittweise berechnet wird.

**Ansatz**: Sei $Q = [q_1, \ldots, q_n]$ unitär und $H$ eine Hessenbergmatrix. Gelten soll $Q^*AQ = H$ bzw.

$$A\,[q_1, \ldots, q_n] = [q_1, \ldots, q_n] \begin{bmatrix} h_{11} & \ldots & \ldots & h_{1n} \\ h_{21} & & & \vdots \\ & \ddots & & \vdots \\ 0 & & h_{n,n-1} & h_{nn} \end{bmatrix}.$$

Ein Vergleich der k-ten Spalten ergibt:

$$Aq_k = h_{1k}q_1 + \ldots + h_{kk}q_k + h_{k+1,k}q_{k+1}.$$

Also ist

$$q_{k+1} = \frac{1}{h_{k+1,k}} \left( Aq_k - \sum_{i=1}^{k} h_{ik}q_i \right).$$

Weiter gilt wegen der Orthogonalität der $q_i$, dass

$$q_j^* A q_k = h_{jk}, \quad j = 1, \ldots, k$$

Also kann $q_{k+1}$ aus $q_1, \ldots, q_k$ bestimmt werden, falls $h_{k+1,k} \neq 0$. (Dies ist gewährleistet, falls $H$ unreduziert ist).

**Algorithmus** (Arnoldi, 1951):
Berechnet zu $x \in \mathbb{C}^n \setminus \{0\}$ und $A \in \mathbb{C}^{n \times n}$ eine unitäre Matrix $Q = [q_1, \ldots, q_n]$, so dass $Q^{-1}AQ = H$ in Hessenbergform ist.

1) Start: $q_1 = \frac{x}{||x||}$.

2) Für $k = 1, 2, \ldots, n-1$:

   (a) $\tilde{q}_{k+1} := Aq_k - \sum_{i=1}^{k} h_{ik}q_i, h_{ik} = q_i^* A q_k$

   (b) $h_{k+1,k} := ||\tilde{q}_{k+1}||$

   (c) $q_{k+1} = \frac{1}{h_{k+1,k}} \tilde{q}_{k+1}$

**Bemerkung:**

(a) Der Algorithmus bricht ab, falls $h_{m+1,m} = 0$ für ein $m$ (Breakdown). Dann haben wir

$$Aq_k = \sum_{j=1}^{k} h_{jk} q_j$$

für $k = 1, \ldots, m-1$ und

$$Aq_m = \sum_{j=1}^{m} h_{jm} q_j.$$

Also haben wir

$$A\left[q_1, \ldots, q_m\right] = \left[q_1, \ldots, q_m\right] \underbrace{\begin{bmatrix} h_{11} & \ldots & h_{1,m-1} & h_{1m} \\ h_{21} & & & \vdots \\ 0 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \ldots & h_{m,m-1} & h_{mm} \end{bmatrix}}_{=:H_m}.$$

Der Unterraum $\mathrm{Span}\{q_1, \ldots, q_m\}$ ist also $A$-invariant.

(b) Falls $h_{m+1,m} \neq 0$, dann erhalten wir

$$A\left[q_1, \ldots, q_m\right] = \left[q_1, \ldots, q_{m+1}\right] \begin{bmatrix} h_{11} & \ldots & h_{1m} \\ h_{21} & \ddots & \vdots \\ 0 & \ddots & \vdots \\ 0 & \ldots & h_{m+1,m} \end{bmatrix}.$$

Dies ist gleichbedeutend mit

$$\begin{aligned} AQ_m &= Q_m H_m + q_{m+1}\left[0, \ldots, 0, h_{m+1,m}\right] \\ &= Q_m H_m + h_{m+1,m} q_{m+1} e_m^T \end{aligned}$$

Dies nennt man Arnoldi-Konfiguration. Wegen der Orthogonalität der $q_i$ gilt außerdem

$$Q_m^* A Q_m = H_m.$$

**Folgerung**: Wegen des Zusammenhangs von Hessenbergmatrizen und Krylovräumen (siehe Abschnitt 3.1) gilt

$$\mathrm{Span}\left\{q_1, \ldots, q_l\right\} = \mathcal{K}_l\left(A, x\right)$$

für $l = 1, \ldots, m+1$. Der Arnoldi-Algorithmus berechnet also Orthonormalbasen von Krylovräumen.

**Anwendung**: Der Arnoldi-Algorithmus als Projektionsmethode für $A \in \mathbb{C}^{n\times n}$, $n$ groß:

1) Berechne für einen Startvektor $x \neq 0$ die Vektoren $q_1, q_2, \ldots$ mit Arnoldi.

2) Höre nach $m \ll n$ Schritten auf

   (a) wegen Breakdown;

   (b) freiwillig, denn wir können vielleicht nicht $n$ Vektoren speichern. Außerdem wird die Berechnung

   $$\tilde{q}_{k+1} = Aq_k - \sum_{j=1}^{k} h_{jk} q_j$$

   mit jedem Schritt aufwendiger.

   Dies liefert Orthonormalbasen von Krylovräumen:

   $$\mathcal{K}_l(A, x) = \mathrm{Span}\{q_1, \ldots, q_l\} = \mathcal{R}(Q_l), \quad l = 1, \ldots, m$$

3) Falls $h_{m+1,m} = 0$, dann ist $\mathcal{R}(Q_m) = \mathcal{K}_m(A, x)$ invariant.

4) Falls $h_{m+1,m} \neq 0$, so wähle $\mathcal{K}_m(A, x) = \mathcal{R}(Q_m)$ als Such- und Testraum für eine Projektionsmethode, d.h. bestimme $\mu \in \mathbb{C}$ und $v \in \mathcal{R}(Q_m) \setminus \{0\}$ mit

   $$Av - \mu v \perp \mathcal{R}(Q_m).$$

**Hoffnung**: Da $\mathcal{R}(Q_m)$ ein Krylovraum ist, sind einige der berechneten Paare $(\mu, v)$ gute Approximationen an Eigenpaare von $A$.

**Definition:** *Sei $A \in \mathbb{C}^{n\times n}$ und $\mathbb{C}^n \supset \mathcal{K} \neq \{0\}$ ein Unterraum. Dann heißt $(\mu, v) \in \mathbb{C} \times \mathbb{C}^n$ Ritzpaar von $A$ bzgl. $\mathcal{K}$ bzw. $\mu$ Ritzwert und $v$ Ritzvektor, falls*

$$v \in \mathcal{K} \setminus \{0\} \quad \textit{und} \quad Av - \mu v \perp \mathcal{K}.$$

**Frage 1**: Wie berechnen wir Ritzpaare von $A$ bzgl. $\mathcal{K}_m(A, x)$?

**Antwort**: Wir berechnen Eigenpaare von $H_m = Q_m^* A Q_m \in \mathbb{C}^{m\times m}$, denn es gilt allgemein (also nicht nur für den Fall, dass $H_m$ eine Hessenbergmatrix ist):

**Lemma 3.3** *Sei $A \in \mathbb{C}^{n\times n}$, sei $Q_m \in \mathbb{C}^{n\times m}$ isometrisch, sowie $\mu \in \mathbb{C}$, $z \in \mathbb{C}^m$ und $v = Q_m z$. Dann gilt:*

$$Q_m^* A Q_m z = \mu z \quad \Longleftrightarrow \quad Av - \mu v \perp \mathcal{R}(Q_m)$$

**Beweis:**

$$Q_m^* A Q_m z = \mu z = \mu Q_m^* Q_m z \quad \Longleftrightarrow \quad Q_m^*(Av - \mu v) = 0 \quad \Longleftrightarrow \quad Av - \mu v \perp \mathcal{R}(Q_m)$$

□

**Bemerkung:** Die Eigenwerte von $H_m$ können wir z.B. mit dem $QR$-Algorithmus berechnen. Dabei ist $H_m$ sogar schon in Hessenbergform. Zur Berechnung der Eigenvektoren führen wir einen Schritt der inversen Iteration mit Shift $\tilde{\mu}$ aus, wobei $\tilde{\mu}$ einer der berechneten Eigenwerte von $H_m$ ist, d.h. wir wählen einen Startvektor $w_0 \in \mathbb{C}^m$, $||w_0|| = 1$ und lösen

$$(H_m - \tilde{\mu} I_m)\, \tilde{w}_1 = w_0$$

für $\tilde{w}_1$ und setzen $w_1 = ||\tilde{w}_1||$. Meist reicht hier eine Iteration, weil $\tilde{\mu}$ sehr nahe an einem Eigenwert, also auch ein sehr guter Shift ist.

**Frage 2**: Woher wissen wir, ob ein Ritzpaar gute Approximation an ein Eigenpaar ist?

**Antwort**: Ein "kleines Residuum" bedeutet einen "kleinen Rückwärtsfehler" für Eigenpaare, d.h. falls $Av - \mu v$ "klein" ist, dann folgt, dass $(\mu, v)$ eine gute Approximation an ein Eigenpaar von $A$ ist (modulo Kondition). (Vgl. hierzu Abschnitt 2.2.)

**Satz 3.4** *Seien $A, Q_m, H_m, h_{m+1,m}$ wie oben nach $m$ Schritten des Arnoldi-Algorithmus erzeugt. Ferner sei $z = (z_1, \ldots, z_m)^T \in \mathbb{C}^m$ ein Eigenvektor von $H_m$ assoziert mit $\mu \in \mathbb{C}$. Dann ist $(\mu, v)$ mit $v = Q_m z$ ein Ritzpaar von $A$ bzgl. $\mathcal{R}(Q_m)$ und es gilt*

$$||Av - \mu v|| = |h_{m+1,m}||z_m|.$$

**Beweis:** Unter Ausnutzung der Arnoldi-Konfiguration folgt:

$$\begin{aligned} Av - \mu v &= AQ_m z - \mu Q_m z \\ &= \left(Q_m H_m + h_{m+1,m} q_{m+1} e_m^T\right) z - \mu Q_m z \\ &= \underbrace{Q_m \left(H_m z - \mu z\right)}_{=0} + h_{m+1,m} z_m q_{m+1} \\ \Rightarrow ||Av - \mu v|| &= |h_{m+1,m}||z| \end{aligned}$$

□

**Bemerkung:**

(a) Wir brauchen für die Berechnung des Residuums $Av - \mu v$ den Ritzvektor $v$ nicht explizit zu bstimmen.

(b) Im Laufe der Zeit geht in der Praxis die Orthogonalität der $q_i$ verloren. Dies passiert immer, wenn $|h_{m+1,m}|$ klein ist. Abhilfe schafft ein Durchlauf des modifizierten Gram-Schmidt-Verfahrens für $q_1, \ldots, q_m$. Dies nennt man Reorthogonalisierung.

(c) Wir wissen bisher nur, wie wir gute Approximationen erkennen können. Wir wissen jedoch noch nicht, ob auch welche auftauchen! $\rightsquigarrow$ Kapitel 3.5

## 3.3 Der symmetrische Lanczos-Algorithmus

Spezialfall: $A = A^* \in \mathbb{C}^{n\times n}$ ist Hermitesch.
Sei $H = Q^*AQ$ in Hessenbergform, dann ist $T := H$ tridiagonal. Angenommen $T$ ist mit dem Arnoldi-Algorithmus berechnet. Dann ist

$$T = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \dots & 0 \\ \beta_1 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \beta_{n-1} \\ 0 & \dots & 0 & \beta_{n-1} & \alpha_n \end{bmatrix} \in \mathbb{R}^{n\times n}$$

sogar reell, denn die Diagonale einer Hermiteschen Matrix ist reell und außerdem haben wir $\beta_m = h_{m+1,m} = ||\tilde{q}_{m+1}||$ für $m = 1, \dots, n-1$. Ist $Q = [q_1, \dots, q_m]$, so erhalten wird durch Vergleich der Spalten in $AQ = QT$, dass

$$\begin{aligned} Aq_1 &= \alpha_1 q_1 + \beta_1 q_2 \\ Aq_k &= \beta_{k-1} q_{k-1} + \alpha_k q_k + \beta_k q_{k+1}, \quad k = 2, \dots, n \end{aligned}$$

Dies bezeichnet man als 3-Term Rekursion. Da $q_1, \dots, q_n$ orthogonal sind, gilt ferner

$$\alpha_k = q_k^* A q_k.$$

**Algorithmus** (Lanczos, eigentlich Lánczos, 1950)
Der Algorithmus entspricht i.w. dem Arnoldi-Algorithmus für den Spezialfall $A = A^*$. Gegeben sei also $A^* = A \in \mathbb{C}^{n\times n}$.

1) Start: Wähle $x \neq 0$. Dann setze $q_0 := 0, q_1 := \dfrac{x}{||x||}$ und $\beta_0 := 0$

2) Für $k = 1, 2, \dots, m$:

   (a) $\alpha_k := q_k^* A q_k$

   (b) $r_k = Aq_k - \beta_{k-1} q_{k-1} - \alpha_k q_k$

   (c) Falls $r_k = 0$ STOP. Andernfalls setze $\beta_k := ||r_k||$ und $q_{k+1} := \dfrac{1}{\beta_k} r_k$.

**Bemerkung:**

(a) Wie bei Arnoldi gilt nach $m$ Schritten:

$$\text{Span}\{q_1, \dots, q_l\} = \mathcal{K}_l(A, x), \quad l = 1, \dots, m$$

Die Eigenpaare der Matrix

$$T_m = \begin{bmatrix} \alpha_1 & \beta_1 & & 0 \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ 0 & & \beta_{m-1} & \alpha_m \end{bmatrix}$$

liefern Ritzpaare für $A$ bzgl. $\mathcal{R}(Q_m)$.

(b) Wie bei Arnoldi: In der Praxis verlieren wir wieder die Orthogonalität der $q_1, \ldots, q_m \rightsquigarrow$ Reorthogonalisierung.

(c) Wegen der 3-Term Rekursion brauchen wir die "alten" $q_i$ nicht zu speichern

$\rightsquigarrow$ wir können $m$ viel größer wählen als bei Arnoldi

Aber: Wenn wir die "alten" $q_i$ nicht speichern, können wir nicht reorthogonalisieren.

$\rightsquigarrow$ Verlust der Orthogonalität

$\rightsquigarrow$ Geistereigenwerte ($T_m$ kann mehrfache Eigenwerte haben, die einem einfachen Eigenwert von $A$ entsprechen.)

## 3.4 Der unsymmetrische Lanczos-Algorithmus

Wir haben wieder eine große Matrix $A \in \mathbb{C}^{n\times n}$ gegeben. Das Problem bei Arnoldi ist: Die Rekursion wird schnell immer teurer. Gibt es auch für $A \neq A^*$ eine 3-Term Rekursion?

**Idee**: Bringe $A$ auf Tridiagonalform und verzichte dabei auf die Orthogonalität der Transformationsmatrix. Der Ansatz ist nun also:

$$X^{-1}AX = T$$

mit $T$ tridiagonal, bzw.

$$AX = X \begin{bmatrix} \alpha_1 & \gamma_1 & & 0 \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \gamma_{n-1} \\ 0 & & \beta_{n-1} & \alpha_n \end{bmatrix}$$

Betrachten wir $X$ nun als

$$X = [x_1, \ldots, x_k],$$

so erhalten wir:

$$Ax_k = \gamma_{k-1}x_{k-1} + \alpha_k x_k + \beta_k x_{k+1}$$

für $k = 1, \ldots, n-1$ und mit $\gamma_0 := 0, x_0 := 0$. Außerdem gilt:

$$T^* = X^*A^*X^{-*} = Y^{-1}A^*Y$$

mit $Y := X^{-*}$. Dann gilt natürlich $Y^*X = I$, d.h. $y_j^*x_i = \delta_{ij}$ mit

$$Y = [y_1, \ldots, y_n].$$

Diese Familien von Vektoren $(x_1, \ldots, x_n)$ und $(y_1, \ldots, y_n)$ heißen deswegen biorthogonal. Wir nehmen nun in der Gleichung $A^*Y = YT^*$ auch wieder einen Spaltenvergleich vor:

$$A^*y_k = \bar{\beta}_{k-1}y_{k-1} + \bar{\alpha}_k y_k + \bar{\gamma}_k y_{k+1}$$

für $k = 1, \ldots, n-1$ und $\beta_0 := 0, y_0 := 0$. Wegen $Y^*X = I$ folgt damit

$$\alpha_k = y_k^*Ax_k.$$

Weiter erhalten wir

$$\beta_k x_{k+1} = Ax_k - \gamma_{k-1}x_{k-1} - \alpha_k x_k =: r_k,$$

sowie

$$\bar{\gamma} y_{k+1} = A^* y_k - \bar{\beta}_{k-1} y_{k-1} - \bar{\alpha}_k y_k =: s_k.$$

Ferner muss gelten:

$$1 = y_{k+1}^* x_{k+1} = \frac{1}{\beta_k \gamma_k} s_k^* r_k$$

für alle $k \geq 1$. Damit haben wir im Prinzip alles was wir brauchen. Allerdings haben wir in der Berechnung von $\beta_k, \gamma_k$ noch Freiheit. Wir könnten nun also eine der folgenden Varianten ansetzen:

$$\beta_k = ||r_k||, \gamma_k = ||s_k||, \beta_k = \gamma_k, \ldots$$

**Algorithmus**: (unsymmetrischer Lanczos, mit Wahl $\beta_k = ||r_k||$)
Gegeben: $A \in \mathbb{C}^{n \times n}$, $x_1, y_1 \in \mathbb{C}^n$ mit $y_1^* x_1 = 1$.

1) Start: $\beta_0 := 0, \gamma_0 := 0, x_0 := 0, y_0 := 0$

2) Für $k = 1, 2, \ldots$:

$$\begin{aligned} \alpha_k &:= y_k^* A x_k \\ r_k &:= Ax_k - \gamma_{k-1}x_{k-1} - \alpha_k x_k \\ s_k &:= A^* y_k - \bar{\beta}_{k-1} y_{k-1} - \bar{\alpha}_k y_k \end{aligned}$$

Falls $s_k^* r_k = 0$, dann STOP. Andernfalls:

$$\begin{aligned} \beta_k &= ||r_k|| \\ \gamma_k &= \frac{1}{\beta_k} s_k^* r_k \\ x_{k+1} &= \frac{1}{\beta_k} r_k \\ y_{k+1} &= \frac{1}{\gamma_k} s_k \end{aligned}$$

**Bemerkung:**

1) Nach $m$ Schritten ohne Abbruch haben wir

$$A \underbrace{[x_1, \ldots, x_m]}_{:= X_m} = [x_1, \ldots, x_m, x_{m+1}] \begin{bmatrix} \alpha_1 & \gamma_1 & & 0 \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \gamma_{m-1} \\ & & \beta_{m-1} & \alpha_m \\ 0 & & & \beta_m \end{bmatrix}.$$

Bezeichnen wir die aus den ersten $m$ Zeilen der obigen Matrix bestehende Submatrix mit $T_m$, dann haben wir

$$AX_m = X_m T_m + \beta_m x_{m+1} e_m^T$$

und analog

$$A^* Y_m = Y_m T_m^* + \gamma_m^* y_{m+1} e_m^T.$$

2) Wegen des Zusammenhangs von Krylovräumen und "partiellen Hessenbergmatrizen" (siehe Abschnitt 3.1) gilt

$$\text{Span}\{x_1, \ldots, x_l\} = \mathcal{K}_l(A, x_1), \quad l = 1, \ldots, m,$$

und

$$\text{Span}\{y_1, \ldots, y_l\} = \mathcal{K}_l(A^*, y_1), \quad l = 1, \ldots, m.$$

3) Für $A = A^*$ und $x_1 = y_1$, d.h. $||x_1|| = 1$, ist der Algorithmus mit dem symmetrischen Lanczos-Algorithmus identisch.

4) Der Algorithmus bricht ab, falls $s_k^* r_k = 0$:

   (a) Falls $r_k = 0$, so ist Span$\{x_1, \ldots, x_k\}$ ein $A$-invarianter Unterraum.

   (b) Falls $s_k = 0$, so ist Span$\{y_1, \ldots, y_k\}$ ein $A^*$-invarianter Unterraum.

   (c) Falls $s_k^* r_k = 0$, aber $s_k, r_k \neq 0$ so bricht der Algorithmus ohne Information über invariante Unterräume ab. Dies nennt man "serious Breakdown", da wir keine Informationen gewinnen, den Algorithmus aber dennoch nicht fortführen können.

5) Der unsymmetrische Lanczos-Algorithmus ist *nicht stabil*. Immer wenn $s_k^* r_k \approx 0$ kommt es zu Instabilitäten.

6) **Lanczos als Projektionsmethode:** Nach $m \ll n$ Schritten haben wir

$$AX_m = X_m T_m + \beta_m \left(x_{m+1} e_m^T\right).$$

Wegen $Y_m^* X_m = I_m$ gilt:

$$T_m = Y_m^* A X_m$$

Einige Eigenwerte dieser Matrix sind (hoffentlich) gute Approximationen an Eigenwerte von $A$. Seien nun $\mu \in \mathbb{C}$ und $z \in \mathbb{C}^m$, $v = X_m z$´. Ist $(\mu, v)$ ein Eigenpaar von $T_m$, so gilt

$$Y_m^* A X_m z = T_m z = \mu z = \mu Y_m^* X_m z$$

und dies ist äquivalent zu

$$Y_m^* (A X_m z - \mu X_m z) = 0.$$

Wir haben also $Y_m^*(Av - \mu v) = 0$ bzw.

$$Av - \mu v \perp \mathcal{R}(Y_m).$$

Ein solches Paar $(\mu, v)$ heißt dann Petrovpaar: Hier nutzen wir $\mathcal{R}(X_m) = \mathcal{K}_m(A, x_1)$ als Suchraum und $\mathcal{R}(Y_m) = \mathcal{K}(A^*, y_1)$ als Testraum.

7) **Lanczos in der Praxis:**

   (a) Betrachte Look-ahead-Varianten: Abschwächung der Biorthogonalität, um um Breakdowns herumzukommen. Dabei wird an den Breakdown-Stellen einfach die Tridiagonal-Struktur der Matrix aufgegeben und um zusätzliche Parameter ergänzt.

   (b) Biorthogonalität geht im Laufe der Zeit durch Rundungsfehler verloren.

## 3.5 Konvergenz von Krylovraumverfahren

**Zusammenfassung**: Krylovraumverfahren konstruieren zu $A \in \mathbb{C}^{n\times n}$ und $x \in \mathbb{C}^n$ ein $X_m = [x_1, \ldots, x_m] \in \mathbb{C}^{n\times m}$, so dass

$$\operatorname{Span}\{x_1, \ldots, x_l\} = \mathcal{K}_l(A, x_1), \quad l = 1, \ldots m$$

Wir brechen nach $m \ll n$ Schritten ab und wählen $\mathcal{K} = \mathcal{K}_m(A, x_1)$ als Suchraum und einen Testraum $\mathcal{L} \subset \mathbb{C}^n$ und bestimmen Paare

$$\mu \in \mathbb{C}, \quad v \in \mathcal{K} \setminus \{0\},$$

so dass

$$Av - \mu v \perp \mathcal{L}.$$

Bei Arnoldi und dem symmetrischen Lanczos wählen wir $\mathcal{L} = \mathcal{K}$, beim unsymmetrische Lanczos $\mathcal{L} = \mathcal{K}_m(A^*, y_1)$.

**Frage**: Gibt es unter den berechneten Paaren $(\mu, v)$ gute Approximationen an Eigenpaare? Wie sieht der Krylovraum eigentlich genau aus?

**Lemma 3.5** *Bezeichnet $\Pi_{m-1}$ die Menge der Polynome vom Grad kleiner gleich $m-1$, so gilt:*

$$\mathcal{K}_m(A, x) = \{p(A)x \mid p \in \Pi_{m-1}\}$$

**Beweis:** Sei $w \in \mathcal{K}_m(A, x)$. Dann gibt es $\alpha_0, \ldots, \alpha_{m-1} \in \mathbb{C}$ mit

$$w = \alpha_0 x + \alpha_1 Ax + \ldots + \alpha_{m-1} A^{m-1} x = p(A)\, x,$$

wobei $p(t) = \alpha_0 + \alpha_1 t + \ldots + \alpha_{m-1} t^{m-1}$. □

**Präzisere Frage**: Enthält der Krylovraum gute Approximationen an Eigenvektoren?

Sei $A$ nun vorerst diagonalisierbar. (Da die Menge der diagonalisierbaren Matrizen dicht in der Menge der Matrizen liegt, ist dies keine bedeutende Einschränkung.) Dann gibt es eine Basis aus Eigenvektoren $(v_1, \ldots, v_n)$, d.h. es gilt

$$Av_i = \lambda_i v_i$$

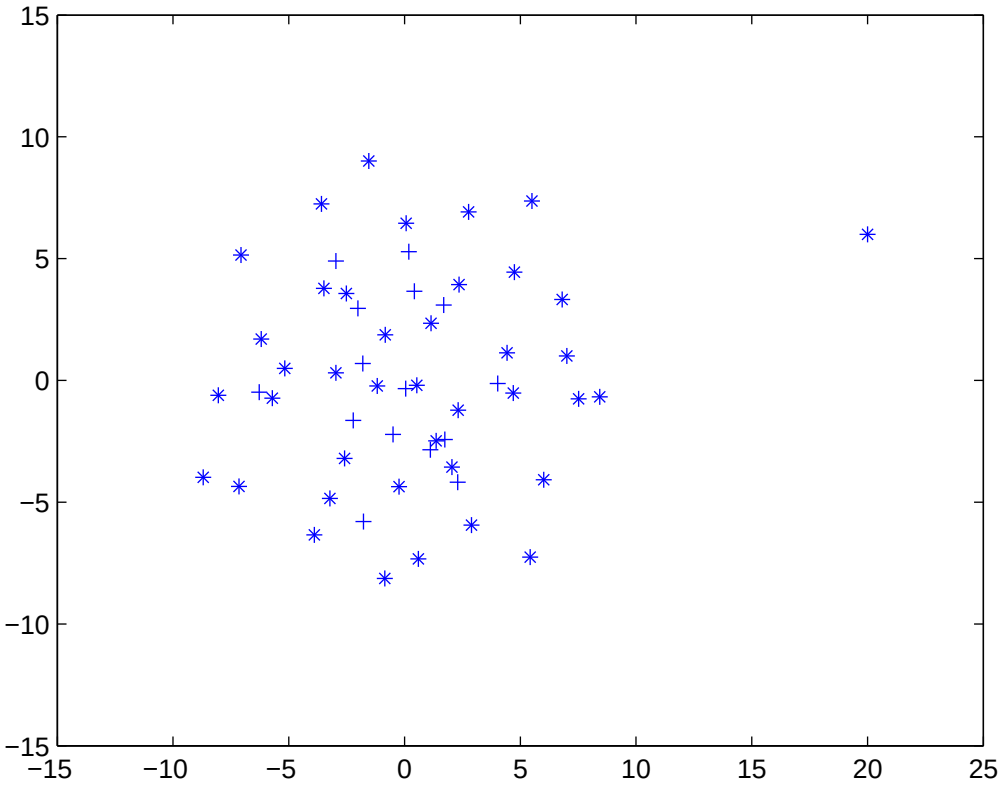
für $i = 1, \ldots, n$ und $\lambda_1, \ldots, \lambda_n \in \mathbb{C}$. Sei $x \in \mathbb{C}^n$ dann gibt es $c_1, \ldots, c_n \in \mathbb{C}$ mit

$$x = c_1 v_1 + \ldots + c_n v_n.$$

Ist $x$ zufällig gewählt, so ist wahrscheinlich, dass keines der $c_i$ sehr kleinen Betrag hat (im Vergleich zu den anderen $c_j$). Dann hat $w \in \mathcal{K}_m(A, x)$ für ein $p \in \Pi_{m-1}$ die Form

$$\begin{aligned} w &= p(A)\, x = p(A)\,(c_1 v_1 + \ldots + c_n v_n) \\ &= c_1 p(\lambda_1)\, v_1 + \ldots + c_n p(\lambda_n)\, v_n. \end{aligned}$$

Die Eigenwerte von $A$ sind i.A. über die komplexe Ebene verteilt. Das Spektrum einer Matrix könnte z.B. wie folgt aussehen.




Dabei steht jedes $*$ für einen Eigenwert der Matrix. Sei nun $\lambda_{17}$ der Ausreißer am rechten Rand des Spektrums und $\lambda_{31}$ einer der Eigenwerte mittendrin. Wähle ein Polynom $\tilde{p} \in \Pi_{m-1}$ mit Nullstellen nahe den Eigenwerten, z.B. mit Nullstellen in $+$. Dann ist $|\tilde{p}(\lambda_{17})|$ sehr wahrscheinlich sehr groß im Vergleich zu $|\tilde{p}(\lambda_i)|$, $i \neq 17$. Definieren wir nun

$$p(t) := \frac{\tilde{p}(t)}{\tilde{p}(\lambda_{17})},$$

dann ist $p(\lambda_{17}) = 1$ und $|p(\lambda_i)|$ klein für $i \neq 17$. Das bedeutet aber, dass

$$w = p(A)\, x$$

eine gute Approximation an $v_{17}$, den Eigenvektor assoziiert mit $\lambda_{17}$ ist. Umgekehrt ist es für $m \ll n$ nahezu unmöglich, ein Polynom $\hat{p}$ zu finden, so dass

$$\hat{p}(\lambda_{31}) = 1 \quad \text{und} \quad |\hat{p}(\lambda_i)| \text{ klein für } i \neq 31,$$

da in der Nähe von $\lambda_{31}$ viele andere Eigenwerte liegen.

**Fazit**: Wir finden zunächst Eigenwerte am Rand des Spektrums.

Für quantitative Aussagen benutzt man z.B. Tschebyscheff-Polynome.

## 3.6 Der implizit neugestartete Arnoldi-Algorithmus

**Ziel**: Berechnung einiger Eigenwerte von $A \in \mathbb{C}^{n\times n}$, $n$ groß.

**Problem** (immer noch): Arnoldi wird im Laufe der Zeit immer teurer. Der unsymmetrische Lanczos ist eine Alternative, ist aber auch instabil. Können wir vielleicht den Arnoldi-Algorithmus verbessern?

**Ideen**:

1) **Neustarts**: Nach $m$ Schritten Arnoldi mit Startvektor $x$ wähle ein $p \in \Pi_{m-1}$ mit

$$|p(\lambda_i)| = \begin{cases} \text{groß} & \text{für interessante } \lambda_i \\ \text{klein} & \text{für uninteressante } \lambda_i \end{cases}$$

Wähle nun $p(A)x$ als neuen Startvektor und beginne erneut mit Arnoldi. Da der Startvektor in den Komponenten in Richtung der interessierenden Eigenvektoren verstärkt wurde, erwarten wir, dass die erzeugten Krylovräume insbesondere gute Approximationen an die interessanten Eigenvektoren enthalten.

Nachteil: Dadurch, dass wir mit einem einzigen Vektor neustarten, verlieren wir jede Menge bereits gewonnene Informationen. Wie wählen wir also den neuen Startvektor $p(A)x$ optimal in dem Sinn, dass wir möglichst viel "Information" behalten, d.h. möglichst viele Approximationen an interessante Eigenwerte erhalten?

2) Behalte mehr Information: Falls $k$ Eigenwerte gewünscht sind, lasse den Arnoldi-Algorithmus $m = k + l$ Schritte laufen. Dann behalte $k$ Vektoren und wirf $l$ Vektoren weg.

$$\rightsquigarrow \text{ IRA (implicitly restarted Arnoldi method), Sorensen, 1992.}$$

Die Strategie des implizit neugestarteten Arnoldi-Algorithmus ist die folgende:

1) Nach $m$ Schritten Arnoldi ohne Abbruch haben wir eine Arnoldi-Konfiguration

$$AQ_m = Q_m H_m + h_{m+1,m} q_{m+1} e_m^T.$$

$Q_m = [q_1, \ldots, q_m]$ ist dabei isometrisch und $H_m \in \mathbb{C}^{m\times m}$ ist in Hessenbergform.

2) Wähle $l$ Shifts $\nu_1, \ldots, \nu_l \in \mathbb{C}$. (Wie? Später!)

3) Führe $l$ Schritte des QR-Algorithmus mit den Shifts $\nu_1, \ldots, \nu_l$ aus:

$$\begin{aligned} H^{(1)} &= H_m \\ \text{für } j &= 1, \ldots, l: \\ &\quad H^{(j)} - \nu_j I = U_j R_j \quad \text{(QR-Zerlegung)} \\ &\quad H^{(j+1)} = R_j U_j + \nu_j I \\ \text{end} & \\ \hat{H}_m &= H^{(l+1)} \\ U &= U_1 \ldots U_l \end{aligned}$$

Dann gilt $\hat{H}_m = U^* H_m U$. Außerdem hat jedes $U_i$ die Form $U_i = G_{12}^{(i)} \ldots G_{m-1,m}^{(i)}$ mit Givens-Matrizen $G_{12}^{(i)}, \ldots, G_{m-1,m}^{(i)}$. Dann ist jedes $U_i$ eine Hessenbergmatrix, d.h. $U$ ist als Produkt von $l$ Hessenbergmatrizen eine Bandmatrix mit unterer Bandweite $l$, d.h. eine obere Dreiecksmatrix mit $l$ zusätzlichen unteren Nebendiagonalen.

4) Es gilt: $AQ_m = Q_m H_m + f_{m+1} e_m^T$ mit $f_{m+1} = h_{m+1,m} q_{m+1}$. Somit ist:

$$A \underbrace{Q_m U}_{=:\hat{Q}_m} = Q_m U \underbrace{U^* H_m U}_{\hat{H}_m} + f_{m+1} \underbrace{e_m^T U}_{=:u_m^T},$$

wobei $u_m^T$ die letzte Zeile von $U$ ist. Also erhalten wir

$$A\hat{Q}_m = \hat{Q}_m \hat{H}_m + f_{m+1} u_m^T$$

mit

$$u_m^T = [\underbrace{0, \ldots, 0}_{k}, \alpha, \underbrace{*, \ldots, *}_{k}].$$

für ein $\alpha \in \mathbb{C}$. Teile nun $\hat{Q}_m$ entsprechend auf:

$$\hat{Q}_m = \begin{array}{c} \\ \end{array} \begin{array}{cc} k & l \\ \end{array} \left[ \hat{Q}_k \quad \tilde{Q}_l \right] = [\hat{q}_1, \ldots, \hat{q}_m]$$

und setze $\hat{Q}_j = [\hat{q}_1, \ldots, \hat{q}_j]$ für $j = 1, \ldots, m$. Analog teilen wir auch $\hat{H}_m$ auf:

$$\hat{H}_m = \begin{array}{c} k \\ l \end{array} \begin{array}{c} \begin{array}{cc} k & \quad l \end{array} \\ \begin{bmatrix} \hat{H}_k & * \\ \beta e_1 e_k^T & \tilde{H}_k \end{bmatrix} \end{array}$$

Dann gilt

$$A \left[ \hat{Q}_k \, \tilde{Q}_k \right] = \left[ \hat{Q}_k \, \tilde{Q}_l \right] \begin{bmatrix} \hat{H}_k & * \\ \beta e_1 e_k^T & \tilde{H}_l \end{bmatrix} + f_{m+1} \left[ 0, \ldots, 0, \alpha, *, \ldots, * \right].$$

Wirf nun die letzten $l$ Spalten weg. Dann erhalten wir

$$\begin{aligned} A\hat{Q}_k &= \hat{Q}_k \hat{H}_k + \beta \tilde{Q}_l e_1 e_k^T + f_{m+1} [0, \ldots, 0, \alpha] \\ &= \hat{Q}_k \hat{H}_k + \underbrace{\left( \beta \tilde{Q}_l e_1 + \alpha f_{m+1} \right)}_{=:\hat{f}_{m+1}} e_k^T \\ &= \hat{Q}_k \hat{H}_k + \hat{f}_{m+1} e_k^T. \end{aligned}$$

Dies ist eine Arnoldi-Konfiguration! Und zwar diejenige, die wir nach $k$ Schritten erhalten hätten, wenn wir den Arnoldi-Algorithmus mit dem Vektor $\hat{q}_1$ neu gestartet hätten Deswegen spricht man hier von einem "impliziten Neustart".

5) Mache nun $l$ weitere Arnoldi-Schritte und beginne wieder bei 1) bis genügend Eigenwerte gefunden wurden.

**Fragen**:

1) Wie wählen wir die Shifts $\nu_1, \ldots, \nu_l$?

2) Wie hängen $q_1$ und $\hat{q}_1$ zusammen?

3) Wie hängen die zugehörigen Krylov-Räume zusammen?

**Lemma 3.6** *Sei $p(t) = (t-\nu_1)\dots(t-\nu_l)$. Dann gilt mit obigen Bezeichnungen und Voraussetzungen*

$$p(A)\,Q_m = Q_m U R + F_m,$$

*wobei* $F_m = \overset{k}{\begin{bmatrix}0 & \overset{l}{\tilde{F}_m}\end{bmatrix}}$ *und* $R = R_k \dots R_1$.

**Beweis:** Wir haben bereits gezeigt (vgl. Abschnitt 2.4.4):

$$p(H_m) = (H_m - \nu_l I)\dots(H_m - \nu_1 I) = UR$$

Jetzt zeigen wir per Induktion nach $l$, dass

$$p(A)\,Q_m = Q_m p(H_m) + F_m.$$

'$l = 1$': Wir haben die Arnoldi-Konfiguration $AQ_m = Q_m H_m + h_{m+1,m} q_{m+1} e_m^T$. Dann gilt

$$(A - \nu_1 I)\,Q_m = Q_m\,(H_m - \nu_1 I) + \underbrace{h_{m+1,m} q_{m+1} e_m^T}_{=:F_1}.$$

'$l-1 \Rightarrow l$': Es gilt:

$$\begin{aligned}
& p(A)\,Q_m \\
= \;& (A-\nu_1 I)\,(A-\nu_2 I)\dots(A-\nu_l I)\,Q_m \\
\overset{\text{I.V.}}{=} \;& (A-\nu_1 I)\left(Q_m\,(H_m-\nu_2 I)\dots(H_m-\nu_l I) + F_{m-1}\right) \\
= \;& \left[Q_m\,(H_m-\nu_1 I) + h_{m+1,m} q_{m+1} e_m^T\right](H_m-\nu_2 I)\dots(H_m-\nu_l I) + (A-\nu_1 I)\,F_{m-1} \\
= \;& Q_m p(H_m) + h_{m+1,m} q_{m+1}\, e_m^T \underbrace{(H_m-\nu_2 I)\dots(H_m-\nu_l I)}_{\text{hat Bandweite } l-1} + \overset{m-l+1}{\begin{bmatrix}0\end{bmatrix}}\overset{l-1}{\begin{bmatrix}(A-\nu_1 I)\,\tilde{F}_{m-1}\end{bmatrix}}, \\
& \qquad\qquad \underbrace{\qquad\qquad = \begin{bmatrix}0,\dots,0,\underbrace{*,\dots,*}_{l}\end{bmatrix} \qquad\qquad}_{=:F_m}
\end{aligned}$$

wobei $F_m = \overset{k}{\begin{bmatrix}0 & \overset{l}{\tilde{F}_m}\end{bmatrix}}$ die gewünschte Form hat. □

**Satz 3.7** *Mit obigen Bezeichnungen und Vorraussetzungen gilt:*

*1)* $\hat{q}_1 = \alpha p(A)\,q_1$ *für ein* $\alpha \in \mathbb{C}$.

*2)* $\mathcal{R}(\hat{Q}_j) = p(A)\,\mathcal{R}(Q_j) = \mathcal{R}(p(A)\,Q_j)$, *für* $j = 1, \dots, m$.

**Beweis:**

1) Es gilt

$$\begin{aligned} p(A)\,Q_m &= \underbrace{Q_m U}_{=\hat{Q}_m} R + F_m = \hat{Q}_m R + F_m \\ &= \hat{Q}_m \begin{bmatrix} r_{11} & \dots & r_{1m} \\ & \ddots & \vdots \\ 0 & & r_{mm} \end{bmatrix} + \overset{\begin{matrix} m-l & l \end{matrix}}{\begin{bmatrix} 0 & \tilde{F}_m \end{bmatrix}} \end{aligned}$$

Vergleich der ersten Spalten ergibt

$$p(A)\,q_1 = \hat{q}_1 r_{11}.$$

Wähle also $\alpha = 1/r_{11}$. Dabei ist $r_{11} \neq 0$, denn sonst wäre $p(A)\,q_1 = 0$, d.h. das Minimalpolynom von $q_1$ bzgl. $A$ hätte Grad $\leq l$. Dann ist aber der Krylovraum $\mathcal{K}_l(A, q_1)$ invariant, d.h. Arnoldi wäre nach $l$ Schritten abgebrochen. Dies ist ein Widerspruch zu $m > l$ Schritten ohne Abbruch.

2) Könnte man analog zu 1) zeigen. Stattdessen gehen wir folgenden Weg:

Wir wissen:

$$\mathcal{R}(Q_j) = \mathcal{K}_j(A, q_1), \quad j = 1, \dots, m$$

und

$$\mathcal{R}(\hat{Q}_j) = \mathcal{K}_j(A, \hat{q}_1), \quad j = 1, \dots, m.$$

Somit folgt

$$\begin{aligned} \mathcal{R}(\hat{Q}_j) &= \operatorname{Span}\left\{p(A)\,q_1, Ap(A)\,q_1, \dots, A^{j-1}p(A)\,q_1\right\} \\ &= \operatorname{Span}\left\{p(A)\,q_1, p(A)\,Aq_1, \dots, p(A)\,A^{j-1}q_1\right\} \\ &= p(A)\operatorname{Span}\left\{q_1, Aq_1, \dots, A^{j-1}q_1\right\} \\ &= p(A)\,\mathcal{R}(Q_j) \end{aligned}$$

□

**Folgerung**: $\mathcal{R}(\hat{Q}_j) = (A - \nu_1 I) \dots (A - \nu_l I)\,\mathcal{R}(Q_j), \quad j = 1, \dots, m.$

Dies entspricht $l$ Schritten simultaner Unterraumiteration mit Shifts $\nu_1, \dots, \nu_l$. Daher können wir davon ausgehen, dass $\mathcal{K}_j(A, q_1)$ i.A. bessere Approximationen an Eigenvektoren enthält als $K_j(A, q_1)$.

**Shiftwahl**: Angenommen $A$ ist diagonalisierbar und $(v_1, \dots, v_n)$ ist eine Basis aus Eigenvektoren zu den Eigenwerten $\lambda_1, \dots, \lambda_n$. Dann gilt

$$q_1 = c_1 v_1 + \dots + c_n v_n \quad \text{mit } c_i \in \mathbb{C}.$$

Wegen $\hat{q}_1 = \alpha p(A) q_1$ folgt

$$\hat{q}_1 = \alpha c_1 p(\lambda_1)\,v_1 + \dots + \alpha c_n p(\lambda_n)\,v_n.$$

Dann ist aber $|p(\lambda_i)|$ groß bzw. klein, falls $\lambda_i$ weit weg von allen $\nu_j$ bzw. nahe einem $\nu_j$ ist. Die Komponenten zu Eigenwerten weit weg von $\nu_1, \dots, \nu_l$ werden in $\hat{q}_1$ verstärkt. Also wähle $\nu_1, \dots, \nu_l$ weit weg von Eigenwerten, die interessieren.

**Beispiel:** Gesucht: $k$ Eigenwerte nahe $\mu \in \mathbb{C}$. Berechne die $m = k + l$ Eigenwerte von $H_m$ und wähle dann die $l$ am weitesten von $\mu$ entfernten Eigenwerte von $H_m$ als Shifts.

**Bemerkung:**

1) Locking- und Purging-Techniken

   Falls $(\lambda, v)$ ein konvergiertes Ritzpaar ist, so wollen wir verhindern, dass weitere Ritzpaare gegen dieses bereits gefundene Paar konvergieren. Dies kann man auf verschiedene Art und Weisen verhindern:

   (a) $\lambda$ ist gewünscht: $v$ wird "geblockt" (locking)

   (b) $\lambda$ ist unerwünscht: $v$ wird "gestrichen" (purging)

   Dazu gibt es viele Details, nachzulesen z.B. bei Lehoucq/Sorensen 1996 "Deflation techniques for an IRA iteration".

2) Die Funktion `eigs` in MATLAB benutzt IRA.

## 3.7 Der Jacobi-Davidson-Algorithmus

**Ziel**: Berechnung eines gezielt ausgewählten Eigenpaars $(\lambda, u)$ einer großen, sparsen Matrix $A \in \mathbb{C}^{n \times n}$ (z.B. könnte $\lambda$ der betragsgrößte Eigenwert, der Eigenwert mit dem geringsten Abstand zu $\tilde{\mu} \in \mathbb{C}$, etc. sein).

**1. Idee**: Arnoldi-Algorithmus mit Shift-and-Invert-Techniken, d.h. wir wenden den Arnoldialgorithmus auf $(A - \nu I)^{-1}$ an. Dies hat den Nachteil, dass wir in jedem Schritt ein LGS mit $A - \nu I$ lösen müssen. Die Praxis zeigt, dass dieses LGS *sehr genau* gelöst werden muss. Wir brauchen also eine (sparse) LR-Zerlegung von $(A - \nu I)$. Diese zu berechnen ist extrem teuer.

**2. Idee**: Wir benutzen wieder eine Projektionsmethode: Wir wählen eine isometrische Matrix $Q_m \in \mathbb{C}^{n \times m}$ mit $m \ll n$ und benutzen $\mathcal{R}(Q_m)$ als Such- und Testraum. Dann gilt wieder

$$Q_m^* A \underbrace{Q_m z}_{=:v} = \mu z \quad \Leftrightarrow \quad Av - \mu v \perp \mathcal{R}(Q_m),$$

d.h. wir erhalten nach wie vor Ritzpaare $(\mu, v)$ aus Eigenpaaren von $Q_m^* A Q_m$.
Neu: $\mathcal{R}(Q_m)$ braucht kein Krylov-Raum zu sein.

**Aufgabe**: Zu $q_1 \in \mathbb{C}^n$, $||q_1|| = 1$ konstruiere eine isometrische Matrix

$$Q_m = [q_1, \ldots, q_m],$$

so dass das Eigenpaar $(\lambda, u)$ möglichst schnell durch Ritzpaare von $A$ bzgl. $\mathcal{R}(Q_m)$ approximiert wird. Seien $q_1, \ldots, q_k$ schon konstruiert. Berechne nun die Ritzpaare von $A$ bzgl. $\mathcal{R}(Q_k)$. Sei $(\mu_k, v_k)$ das Ritzpaar mit $||v_k|| = 1$, das $(\lambda, u)$ am besten approximiert.

**Ansatz**:

$$\begin{aligned} u &= v_k + x, \qquad \text{mit } x \perp v_k \\ \lambda &= \mu_k + \eta \end{aligned}$$

Dabei sei o.B.d.A. das (ohnehin unbekannte) $u$ so skaliert, dass tatsächlich $(u - v_k) \perp v_k$ gilt.

**Idee**: Approximiere $x$ und berechne $q_{k+1}$ aus $x$ durch Orthornomalisieren gegen $q_1, \ldots, q_k$.

**Berechnung von** $x$: Wir suchen orthogonal zu $v_k$. Wir betrachten daher

$$P = I - v_k v_k^*.$$

Dies ist der Orthoprojektor auf $\text{Span}\,(v_k)^\perp$. Ferner sei

$$r_k := Av_k - \mu_k v_k$$

das Residuum von $(\mu_k, v_k)$ bzgl. $A$. Dann gilt

$$Pv_k = 0, \quad Px = x, \quad Pr_k = r_k,$$

denn da $(\mu_k, v_k)$ ein Ritzpaar ist, gilt $r_k \perp \mathcal{R}\,(Q_k) \ni v_k$, also $r_k \perp v_k$. Ferner gilt

$$\begin{aligned} & Au &&= \lambda u \\ \iff \quad & A\,(v_k + x) &&= \lambda\,(v_k + x) \\ \iff \quad & (A - \lambda I)\,x &&= -\,(A - \lambda I)\,v_k \\ & &&= -\,(A - \mu_k I)\,v_k + \eta v_k = -r_k + \eta v_k \\ \iff \quad & P\,(A - \lambda I)\,x &&= -Pr_k = -r_k \\ \iff \quad & P\,(A - \lambda I)\,Px &&= -r_k \quad \text{ und } x \perp v_k \end{aligned}$$

Wir können nun leider die letzte Gleichung nicht direkt lösen, da wir $\lambda$ nicht kennen. Daher ersetzen wir $\lambda$ einfach durch den Ritzwert $\mu_k$:

$$P\,(A - \mu_k I)\,Px = -r_k, \quad x \perp v_k. \tag{3.1}$$

Diese Gleichung nennt man die "Jacobi-Korrektor-Gleichung".

**Bemerkung:** $P\,(A - \mu_k I)\,P$ ist die orthogonale Projektion von $A - \mu_k I$ auf $\text{Span}\,\{v_k\}^\perp$.

**Algorithmus**: (Jacobi-Davidson-Algorithmus, Slijpen/van der Vorst, 1996)
Dieser Algorithmus berechnet zu einer gegebenen Matrix $A \in \mathbb{C}^{n\times n}$ ein $Q_m \in \mathbb{C}^{n\times m}$ isometrisch, so dass $M_m = Q_m^* A Q_m$ gute Approximationen an ein vorgegebenes Eigenpaar $(\lambda, u)$ enthält.

1) Start: Wähle $q_1 \in \mathbb{C}^n, ||q_1|| = 1$.

2) Iteriere, für $k = 1, 2, \ldots$ bis Konvergenz:

   (a) $Q_k = [q_1, \ldots, q_k]\,,\ w_k = Aq_k,\ W_k = [w_1, \ldots, w_k] = AQ_k$ und $M_k = Q_k^* W_k$.

(b) Berechne die Ritzpaare (bzw. die Eigenpaare von $M_k$) und wähle das Ritzpaar $(\mu_k, v_k)$, dass $(\lambda, u)$ am besten approximiert. Es gilt $v_k = Q_k z$ für ein $z \in \mathbb{C}^k$.

(c) Berechne $r_k = W_k z - \mu Q_k z$. (Dies ist das Residuum, denn $r_k = W_k z - \mu Q_k z = AQ_k z - \mu_k Q_k z = Av_k - \mu_k v_k$.)

(d) Ist $||r_k||$ hinreichend klein, dann STOP, wir haben ein konvergiertes Eigenpaar gefunden. Andernfalls löse die Jacobi-Korrektor Gleichung (3.1) nach $x$.

(e) Berechne $q_{k+1}$ aus $x$ durch Orthnormalisieren gegen $q_1, \ldots, q_k$.

**Interpretation**: Wegen $Px = x$ folgt aus (3.1):

$$(A - \mu_k I)\, x = -r_k + \alpha v_k$$

mit $\alpha = v_k^* (A - \mu_k I)\, x$. Also folgt:

$$\begin{aligned} x &= -(A - \mu_k I)^{-1} r_k + \alpha (A - \mu_k I)^{-1} v_k \\ &= -v_k + \alpha (A - \mu_k I)^{-1} v_k \end{aligned}$$

Da $v_k$ schon in dem Suchraum enthalten ist, haben wir $\mathcal{R}(Q_k)$ also effektiv um den skalierten Vektor $(A - \mu_k I)^{-1} v_k$ erweitert. Da

$$v_k^* (Av_k - \mu v_k) = 0, \quad \mu_k = \frac{v_k^* A v_k}{v_k^* v_k}$$

entspricht das dem Vektor, den man erhält, wenn man einen Schritt der Rayleigh-Quotient-Iteration auf $v_k$ anwendet. Diese konvergiert für $A = A^*$ kubisch, bzw. allgemein mindestens quadratisch.

**Folgerung:** (heuristisch) Jacobi-Davidson konvergiert quadratisch gegen ein Eigenpaar $(\lambda, u)$, falls (3.1) in jedem Schritt exakt gelöst wird.

**Bemerkung:**

1) I.A. ist es unmöglich (zu teuer) die Gleichung (3.1) exakt zu lösen. Deswegen wird (3.1) üblicherweise nur approximativ gelöst. Mögliche Approximationen sind:

   (a) $P(A - \mu_k I)\, P \approx I$ damit folgt $x = -r_k$. Dies ist formal äquivalent zum Arnoldi-Algorithmus (Übung).

   (b) $P(A - \mu_k I)\, P \approx (D - \mu_k I)$ wobei $D$ der Diagonalteil von $A$ ist. ($\rightsquigarrow$ Davidson-Methode in der Quantenchemie) Dies funktioniert gut für diagonal-dominante Matrizen.

   (c) Löse (3.1) mit iterativen Verfahren (siehe Kapitel 4).

2) Warum heißt der Algorithmus "Jacobi-Davidson"-Algorithmus? Der Algorithmus hat im Wesentlichen mit dem Davidson-Algorithmus das schrittweises Vergrößern eines Suchraums gemeinsam. Für die Approximation $P(A - \mu_k I)\, P \approx (D - \mu_k I)$ erhalten wir den Davidson-Algorithmus zurück.

Andererseits stammt der folgende Ansatz von Jacobi:

$$A \begin{bmatrix} 1 \\ z \end{bmatrix} = \begin{bmatrix} \alpha & c^T \\ b & F \end{bmatrix} \begin{bmatrix} 1 \\ z \end{bmatrix} = \lambda \begin{bmatrix} 1 \\ z \end{bmatrix}$$

Dies ist äquivalent zu

$$\begin{aligned} \lambda &= \alpha + c^T z \\ (F - \lambda I)\, z &= -b. \end{aligned}$$

Jacobi empfahl, das LGS $(F - \lambda I)z = 0 - b$ iterativ zu lösen und aus der Lösung gemäß $\lambda = \alpha + c^T z$ eine verbesserte Eigenwertapproximation zu berechnen. Die Grundidee ist hier das orthogonale Suchen nach Korrekturen, denn der Korrekturvektor $[0, z^T]^T$ ist orthogonal zu dem Ansatzvektor $[1, 0]^T$. Im Jacobi-Davidson-Algorithmus suchen wir nun ebenfalls in jedem Schritt wieder orthogonal zu unserer letzten Approximation.

3) Vergleich: Jacobi-Davidson und Arnoldi mit Shift-and-Invert für die Aufgabe, ein ausgewähltes Eigenpaar zu berechnen.

Jacobi-Davidson besteht i.w. aus zwei Schritten. Durch Lösen der Jacobi-Korrektor-Gleichung $P\,(A - \mu_k I)\,Px = -r_k$ bestimmen wir die neue Suchrichtung. Dann müssen wir für die Berechnung der Eigenwerte von $M_k$ den Operator $A$ anwenden in dem Schritt $W_k = AQ_k$. Wir können also die Jacobi-Korrektor-Gleichung approximativ lösen, denn hier bestimmen wir nur die Suchrichtung. Die spektrale Information des Operators $A$ wird in dem Schritt $W_k = AQ_k$ injiziert, denn hier berechnen wir schließlich die Matrix $M_k$ und deren Eigenwerte. Wenn wir die Jacobi-Korrektor-Gleichung approximativ lösen wird vielleicht die Konvergenz verlangsamt, aber die spektrale Information des Operators $A$ wird nicht verfälscht, da sie an anderer Stelle eingeht.

Bei Arnoldi sind diese beiden Schritte "Wahl der neuen Suchrichtung" und "Anwenden des Operators" kombiniert in der Gleichung

$$\tilde{q}_{k+1} = (A - \nu I)^{-1} q_k - \sum_{i=1}^{k} h_{ik} q_i.$$

Die Lösung des LGS muss hier exakt erfolgen, denn dies ist die einzige Stelle, wo wir den Operator $A$ anwenden, also spektrale Information injizieren. Wenn wir das LGS nur approximativ lösen wird die spektrale Information des Operators $A$ verfälscht.

Weiter lässt sich sagen, dass sich Neustarts, Locking und Purging bei Jacobi-Davidson einfacher bewerkstelligen lassen, da wir keine Arnoldi-Konfiguration aufrecht erhalten müssen.

Andererseits approximiert Jacobi-Davidson i.A. zunächst nur das ausgewählte Eigenpaar $(\lambda, u)$, während Arnoldi gleichzeitig auch andere Eigenpaare approximiert.

# Kapitel 4

# Iterative Verfahren zur Lösung von Linearen Gleichungssystemen

**Situation**: $A \in \mathbb{C}^{n\times n}$ schwach besetzt, $n$ groß, $b \in \mathbb{C}^n$.

**Ziel**: Bestimme $x \in \mathbb{C}^n$ mit $Ax = b$.

## 4.1 Spliting-Methoden

Die Grundidee ist hier die Matrix in zwei Summanden aufzuteilen: $A = M - N$, so dass man das Problem in ein Fixpunktproblem umwandeln kann:

$$Mx = Nx + b$$

Dadurch ergibt sich ein Iterationsverfahren vermöge der Rekursion

$$Mx^{(k+1)} = Nx^{(k)} + b$$

**Bemerkung:** Sind $A, M$ nichtsingulär und $\varrho\left(M^{-1}N\right) < 1$, wobei

$$\varrho\left(B\right) := \max_{\lambda\in\sigma(B)} |\lambda|,$$

dann konvergiert die Iteration für jeden Startwert $x_0$ gegen $A^{-1}b$ (siehe Übung).

**Beispiel:** Wir zerlegen $A = \underbrace{\begin{bmatrix} 0 & \dots & 0 \\ * & \ddots & \vdots \\ * & * & 0 \end{bmatrix}}_{=:L} + \underbrace{\begin{bmatrix} * & & 0 \\ & \ddots & \\ 0 & & * \end{bmatrix}}_{=:D} + \underbrace{\begin{bmatrix} 0 & * & * \\ \vdots & \ddots & * \\ 0 & \dots & 0 \end{bmatrix}}_{=:R}$.

(a) Setze $M = D$ und $N = -L - R$. DIes entspricht der *Jacobi-Iteration.*

(b) Setze $M = L + D$ und $N = -R$ entsprechend. Dies entspricht dem *Gauß-Seidel-Verfahren.*

## 4.2 Die Methode der konjugierten Gradienten (CG - „Conjugate Gradients“)

**Spezialfall**: $A \in \mathbb{R}^{n\times n}$ symmetrischen und positiv definit, $b \in \mathbb{R}^n$.

**Grundidee**: Betrachte

$$\varphi : \mathbb{R}^n \to \mathbb{R}, x \mapsto \varphi(x) = \frac{1}{2}x^T Ax - x^T b.$$

Dann gilt:

$$\nabla\varphi(x) = Ax - b, \quad \operatorname{Hess}\varphi(x) = A,$$

d.h. unser Lineares Gleichungssystem entspricht einem Extremwertproblem, denn in $\hat{x} = A^{-1}b$ ist das eindeutig bestimmte globale Minimum von $\varphi$:

$$\varphi(\hat{x}) = -\frac{1}{2}b^T A^{-1}b$$

Wir minimieren also $\varphi$ einfach schrittweise und hoffen, dass wir uns dadurch auch der Lösung des linearen Gleichungssystems annähern.

### 4.2.1 Steilster Abstieg, Gradientensuchrichtung

**Idee**: $\varphi$ fällt am stärksten in Richtung des negativen Gradienten ab:

$$-\nabla\varphi(x) = b - Ax$$

**Definition:** *Seien $A \in \mathbb{C}^{n\times n}$ und $x, b \in \mathbb{C}^n$. Dann heißt*

$$r = b - Ax$$

*das Residuum von $x$ bzgl. $A$ und $b$.*

Für $r \neq 0$ ist $\varphi(x + \alpha r) < \varphi(x)$ für ein $\alpha > 0$. Daher können wir $\varphi$ verkleinern, indem wir so einen Parameter $\alpha$ bestimmen. Dabei minimieren wir in jedem Schritt über $\alpha$:

**Lemma 4.1** *Das Minimum von $\alpha \mapsto \varphi(x + \alpha r)$ ist in*

$$\alpha = \frac{r^T r}{r^T Ar}.$$

**Beweis:** Übung. □

Damit erhalten wir den folgenden Algorithmus.

**Algorithmus** („Steepest Descent“ bzw. „Steilster Abstieg“)
Berechnet für $A \in \mathbb{R}^{n\times n}$ symmetrisch positiv definit und $b \in \mathbb{R}^n$ die Lösung $x = A^{-1}b$ des linearen Gleichungssystems $Ax = b$.

1) Start: Wähle $x_0 \in \mathbb{R}^n$

2) Iteriere für $k = 1, 2, \ldots$ bis Konvergenz:

   (a) $r_{k-1} = b - Ax_{k-1}$

   (b) Falls $r_{k-1} = 0$ STOP! ($x_{k-1} = A^{-1}b$). Andernfalls setze $\alpha_k = \dfrac{r_{k-1}^T r_{k-1}}{r_{k-1}^T A r_{k-1}}$.

   (c) $x_k = x_{k-1} + \alpha_k r_{k-1}$

**Bemerkung:** Man kann zeigen, dass:

$$\varphi(x_{k+1}) + \frac{1}{2}b^T A^{-1} b \leq \left(1 - \frac{1}{\kappa_2(A)}\right)\left(\varphi(x_k) + \frac{1}{2}b^T A^{-1} b\right)$$

Wir erhalten also globale Konvergenz für alle Startwerte. Dabei müssen wir aber folgende Nachteile in Kauf nehmen:

(a) Die Konvergenz ist sehr langsam, falls $\kappa_2(A)$ groß ist.

(b) Die Konvergenzaussage bezieht sich auf $\varphi$, aber wenn $\varphi$ schnell klein wird, muss dies nicht auch automatisch für das Residuum gelten.

Diese Nachteile entstehen uns im wesentlichen aus folgenden Gründen:

1) Wir minimieren nur über eine Suchrichtung $r_k$. Wir haben aber mehr Richtungen zur Verfügung (nämlich $r_0, \ldots, r_k$).

2) Die Suchrichtungen sind „nicht verschieden genug“.

### 4.2.2 A-konjugierte Suchrichtungen

Wir versuchen nun das Konvergenzverhalten des Algorithmus aus Abschnitt 4.2.1 durch eine kleine Modifikation zu verbessern. Die Grundidee dabei ist: Wählen wir in jedem Schritt statt des negativen Gradienten als Suchrichtung ein $p \in \mathbb{R}^n$, mit $p \not\perp r$, so finden wir auch in dieser Richtung (oder der Gegenrichtung) einen Abfall von $\varphi$. Wir wählen nun also in jedem Schritt statt $r_k$ eine Suchrichtung $p_k$ mit $p_k^T r_k \neq 0$.
Dazu stellen wir folgende **Forderungen** an die Wahl von $p_{k+1}$ und $x_{k+1}$:

1) $p_1, \ldots, p_{k+1}$ sind linear unabhängig.

2) $\varphi(x_{k+1}) = \min\limits_{x \in \mathcal{R}_{k+1}} \varphi(x)$, wobei $\mathcal{R}_{k+1} := x_0 + \operatorname{Span}\{p_1, \ldots, p_{k+1}\}$.

3) $x_{k+1}$ kann „leicht“ aus $x_k$ berechnet werden.

Die Bedingungen 1) und 2) garantieren zusammen Konvergenz nach spätestens $n$ Schritten, denn dann minimieren wir $\varphi$ über den gesamten Raum $\mathbb{R}^n$.

Wir diskutieren im folgenden die Berechnung von $p_{k+1}$ und $x_{k+1}$. Dazu seien die Suchrichtungen $p_1, \ldots, p_k \in \mathbb{R}^n$ und $x_k$ mit $\varphi(x_k) = \min\limits_{x \in \mathcal{R}_k} \varphi(x)$ bereits bestimmt.

**Gesucht**: $p_{k+1}$ und $x_{k+1}$ mit $\varphi(x_{k+1}) = \min\limits_{x \in \mathcal{R}_{k+1}} \varphi(x)$, so dass 1)–3) erfüllt sind.

Dazu schreiben wir $x_k = x_0 + P_k y_k$ mit $P_k = [p_1, \ldots, p_k]$ und $y_k \in \mathbb{R}^k$ und machen den Ansatz

$$x_{k+1} = x_0 + P_k y + \alpha p_{k+1}$$

für $y \in \mathbb{R}^k, \alpha \in \mathbb{R}$. Unser Ziel ist dann die Bestimmung der Parameter $y$ und $\alpha$. Nun gilt:

$$\begin{aligned}
\varphi(x_{k+1}) &= \frac{1}{2}(x_0 + P_k y + \alpha p_{k+1})^T A (x_0 + P_k y + \alpha p_{k+1}) - \left(x_0 + P_k y + \alpha p_{k+1}^T\right) b \\
&= \varphi(x_0 + P_k y) + \alpha p_{k+1}^T A (x_0 + P_k y) - \alpha p_{k+1}^T b + \frac{1}{2}\alpha^2 p_{k+1}^T A p_{k+1} \\
&= \underbrace{\varphi(x_0 + P_k y)}_{\text{nur } y} + \alpha p_{k+1}^T A P_k y + \underbrace{\frac{1}{2}\alpha^2 p_{k+1}^T A p_{k+1} - \alpha p_{k+1}^T r_0}_{\text{nur } \alpha}
\end{aligned}$$

Wäre der störende Mischterm nicht, dann könnten wir getrennt über die beiden Variablen minimieren. Also wählen wir $p_{k+1}$ so, dass gilt:

$$p_{k+1}^T A P_k = 0.$$

Damit erhalten wir:

$$\min_{x \in \mathcal{R}_{k+1}} \varphi(x) = \underbrace{\min_{y \in \mathbb{R}^k} \varphi(x_0 + P_k y)}_{\text{Lsg: } y = y_k} + \underbrace{\min_{\alpha \in \mathbb{R}} \left(\frac{1}{2}\alpha^2 p_{k+1}^T A p_{k+1}^T - \alpha p_{k+1}^T r_0\right)}_{\text{Lsg: } \alpha_{k+1} = \frac{p_{k+1}^T r_0}{p_{k+1}^T A p_{k+1}}}$$

Die erste Minimierungsaufgabe wird durch $y = y_k$ gelöst, denn $x_k = x_0 + P_k y_k$ erfüllt ja gerade

$$\varphi(x_k) = \min_{x \in \mathcal{R}_k} \varphi(x).$$

Die zweite Minimierungsaufgabe ist eine Minimierungsaufgabe über den reellen Zahlen und wird durch $\alpha_{k+1} = \frac{p_{k+1}^T r_0}{p_{k+1}^T A p_{k+1}}$ gelöst. Durch diese Vorgehensweise haben wir die Forderungen 2) und 3) erfüllt.

**Fazit**: Wähle A-konjugierte Suchrichtungen $p_k$, d.h. wähle

$$p_{k+1} \in \operatorname{Span}\{Ap_1, \ldots, Ap_k\}^\perp, k = 1, 2, \ldots$$

Dann folgt:

$$p_i^T A p_j = 0, \quad i \neq j, \quad i, j = 1, \ldots, k$$

d.h. $p_1, \ldots, p_k$ sind orthogonal bzgl. des Skalarprodukts:

$$\langle x, y \rangle_A := y^T A x$$

Es stellt sich nun die Frage, ob sich auch immer $A$-konjugierte Suchrichtungen finden lassen. Die Antwort erhalten wir aus dem folgenden Lemma.

**Lemma 4.2** *Ist $r_k = b - Ax_k \neq 0$, so gibt es $p_{k+1} \in Span\{Ap_1, \ldots, Ap_k\}^\perp$ mit $p_{k+1}^T r_k \neq 0$.*

**Beweis**: Für $k = 0$ ist dies klar (wähle z.B. $p_1 = r_0$). Für $k \geq 1$ folgt wegen $r_k \neq 0$:

$$A^{-1}b \notin \mathcal{R}_k = x_0 + \text{Span}\{p_1, \ldots, p_k\},$$

d.h. insbesondere ist das Minimum von $\varphi$ noch nicht erreicht. Somit ist dann auch

$$b \notin Ax_0 + \text{Span}\{Ap_1, \ldots, Ap_k\}$$

bzw.

$$r_0 = b - Ax_0 \notin \text{Span}\{Ap_1, \ldots, Ap_k\}.$$

Also gibt es $p_{k+1} \in \text{Span}\{Ap_1, \ldots, Ap_k\}^\perp$ mit $p_{k+1}^T r_0 \neq 0$. Wegen $x_k \in x_0 + \text{Span}\{p_1, \ldots, p_k\}$ gilt:

$$r_k = b - Ax_k \in r_0 + \text{Span}\{Ap_1, \ldots, Ap_k\}$$

also ist auch

$$p_{k+1}^T r_k = p_{k+1}^T r_0 \neq 0. \quad \square$$

**Bemerkung:** Wir halten folgende Beobachtung aus dem obigen Beweis fest: Wegen $p^T r_k = p^T r_0$ für $p \in \text{Span}\{Ap_1, \ldots, Ap_k\}^T$ gilt speziell $p_{k+1}^T r_k = p_{k+1}^T r_0$, also auch

$$\alpha_{k+1} = \frac{p_{k+1}^T r_0}{p_{k+1}^T A p_{k+1}} = \frac{p_{k+1}^T r_k}{p_{k+1}^T A p_{k+1}}$$

Wir zeigen nun, dass durch unsere Vorgehensweise auch die Forderung 1) erfüllt ist:

**Lemma 4.3** *Die Suchrichtungen $p_1, \ldots, p_k$ sind linear unabhängig.*

**Beweis:** $P_k^T A P_k = \text{diag}\left(p_1^T A p_1, \ldots, p_k^T A p_k\right)$ ist insbesondere invertierbar (da $A$ pos. def.). Also hat $P_k$ vollen Rang, d.h. die Spalten $p_1, \ldots, p_k$ sind linear unabhängig. $\square$

Zusammenfassend erhalten wir folgenden Algorithmus:

**Algorithmus** ($A$-konjugierte Suchrichtungen)
Berechnet für $A \in \mathbb{R}^{n \times n}$ symmetrisch positiv definit und $b \in \mathbb{R}^n$ die Lösung $x = A^{-1}b$ des linearen Gleichungssystems $Ax = b$.

1) Start: Wähle $x_0 \in \mathbb{R}^n$

2) Iteriere für $k = 1, 2, \ldots$ bis Konvergenz:

   (a) $r_k = b - Ax_k$

   (b) Falls $r_k = 0$ STOP! ($x_k = A^{-1}b$). Andernfalls wähle $p_{k+1} \in \text{Span}\{Ap_1, \ldots, Ap_k\}^\perp$ mit $p_{k+1}^T r_k \neq 0$ und berechne

   $$\alpha_{k+1} = \frac{p_{k+1}^T r_k}{p_{k+1}^T A p_{k+1}}.$$

   (c) $x_{k+1} = x_k + \alpha_{k+1} p_{k+1}$

Man beachte, dass wir noch Freiheit in der Wahl von $p_{k+1}$ haben.

### 4.2.3 „CG = steilster Abstieg + $A$-konjugierte Suchrichtungen“

Wie wir gesehen haben, bietet die Wahl von $A$-konjugierten Suchrichtungen einige Vorteile (einfache Berechnung von $x_{k+1}$ aus $x_k$, Garantie der Konvergenz nach $n$ Schritten). Andererseits möchten wir auch die Idee des „steilsten Abstiegs“nicht aufgeben, denn unsere Funktion $\varphi$ fällt ja in Richtung des negativen Gradienten besonders schnell ab und wir sehen diese Richtung daher heuristisch als eine „gute Suchrichtung“an. Die Idee ist nun, die Freiheit in der Wahl von $p_{k+1}$ demhingehend zu benutzen, d.h. wir wählen das $p_{k+1}$, welches „am nächsten“an $r_k$, also in Richtung des negativen Gradienten liegt. Wir wählen also das $p_{k+1}$, für das gilt

$$||p_{k+1} - r_k|| = \min_{p \in \mathrm{Span}\{Ap_1,\ldots,Ap_k\}^\perp} ||p - r_k|| \qquad (*)$$

Dies mutet zunächst eigenartig an, denn im Abschnitt 4.2.2. hatten wir uns Mühe gegeben, die Suchrichtungen so zu wählen, dass das zugehörige Optimierungsproblem besonders einfach gelöst werden kann. Und nun bestimmen wir die jeweilige Suchrichtung über eine neue Optimierungsaufgabe. Macht das überhaupt Sinn? Wir werden im Folgenden sehen, dass sich die neue Optimierungsaufgabe $(*)$ mit überraschender Einfachheit lösen lässt, denn es wird sich herausstellen, dass die neue Suchrichtung $p_{k+1}$ einfach nur eine Linearkombination der vorhergehenden Suchrichtung $p_k$ und des Residuums $r_k$ ist.

**Grundvoraussetzung**: Im Folgenden seien mit denselben Bezeichnungen und Voraussetzungen wie in 4.2.2 die $A$-konjugierten Suchrichtungen so gewählt, dass (*) erfüllt ist für $k = 0, \ldots, m$. Ferner sei $P_k = [p_1, \ldots, p_k]$.

**Ziel**: Zeige $p_{k+1} \in \mathrm{Span}\,\{p_k, r_k\}$.

**Lemma 4.4** *Sei $k \in \{1, \ldots, m\}$ und $z_k \in \mathbb{R}^k$ so, dass*

$$||r_k - AP_k z_k|| = \min_{z \in \mathbb{R}^k} ||r_k - APz||.$$

*Dann gilt: $p_{k+1} = r_k - AP_k z_k$.*

**Beweis:** Sei $\hat{p} := r_k - AP_k z_k$, dann ist durch die Voraussetzung des Lemmas $\hat{p}$ gerade die orthogonale Projektion von $r_k$ auf $\mathcal{R}\,(AP_k)^\perp$, also ist

$$||\hat{p} - r_k|| = \min_{p \in \mathcal{R}(AP_k)^\perp} ||p - r_k||.$$

Damit folgt: $\hat{p} = p_{k+1}$. □

**Satz 4.5** *Ist $r_k \neq 0$ für $k = 0, \ldots, m$, so gilt für $k = 0, \ldots, m$:*

1) $r_{k+1} = r_k - \alpha_{k+1} A p_{k+1}$
2) $\mathrm{Span}\,\{p_1, \ldots, p_{k+1}\} = \mathrm{Span}\,\{r_0, \ldots, r_k\} = \mathcal{K}_{k+1}\,(A, r_0)$
3) $r_{k+1} \perp r_j$ für $j = 0, \ldots, k$
4) $p_{k+1} \in \mathrm{Span}\,\{p_k, r_k\}$

**Beweis**:

1) Wegen $x_{k+1} = x_k + \alpha_{k+1} p_{k+1}$ gilt

$$r_{k+1} = b - Ax_{k+1} = \underbrace{b - Ax_k}_{= r_k} - \alpha_{k+1} A p_{k+1}.$$

2) Durch wiederholtes Anwenden von 1) folgt:

$$\operatorname{Span}\{Ap_1, \ldots, Ap_k\} \subseteq \operatorname{Span}\{r_0, \ldots, r_k\}, \quad k = 1, \ldots, m.$$

Im Lemma haben wir gezeigt, dass für alle $k = 0, \ldots, m$ gilt:

$$p_{k+1} = r_k - AP_k z_k \in \operatorname{Span}\{r_0, \ldots, r_k\}.$$

Damit erhalten wir

$$\operatorname{Span}\{p_1, \ldots, p_{k+1}\} \subseteq \operatorname{Span}\{r_0, \ldots, r_k\}$$

für $k = 0, \ldots, m$. Ferner gilt mit 1):

$$r_{k+1} \in \operatorname{Span}\{r_k, Ap_{k+1}\} = \operatorname{Span}\{r_k, Ar_0, \ldots, Ar_k\}$$

für $k = 0, \ldots, m$. Dann ist also:

$$\begin{array}{rcl} r_1 & \in & \operatorname{Span}\{r_0, Ar_0\}, \\ r_2 & \in & \operatorname{Span}\{r_0, Ar_0, Ar_1\} \subseteq \operatorname{Span}\left\{r_0, Ar_0, A^2 r_0\right\}, \\ usw. & \vdots & usw. \end{array}$$

Mit Induktion erhalten wir schließlich

$$\operatorname{Span}\{p_1, \ldots, p_{k+1}\} \subseteq \operatorname{Span}\{r_0, \ldots, r_k\} \subseteq \mathcal{K}_{k+1}(A, r_0).$$

Die Gleichheit folgt aus Dimensionsgründen.

3) Wir zeigen $P_k^T r_k = 0$ d.h. $p_1, \ldots, p_k \perp r_k$ für alle $k = 1, \ldots, m$. Wegen 2) gilt dann auch $r_0, \ldots, r_{k-1} \perp r_k$ wie gewünscht. Nun gilt $x_{k+1} = x_0 + P_k y_k$, wobei $y_k$ die Funktion

$$\begin{aligned} \varphi(x_0 + P_k y) &= \frac{1}{2}(x_0 + P_k y)^T A(x_0 + P_k y) - (x_0 + P_k y)^T b \\ &= \varphi(x_0) + y^T P_k^T (Ax_0 - b) + \frac{1}{2} y^T P_k^T A P_k y \end{aligned}$$

minimiert. Der Gradient von $y \mapsto \varphi(x_0 + P_k y)$ wird also an der Stelle $y = y_k$ gleich Null, d.h. es gilt

$$P_k^T A P_k y_k + P_k^T (Ax_0 - b) = 0.$$

Dies ist gleichbedeutend mit $0 = P_k^T (b - Ax_0 - AP_k y_k) = P_k^T (b - Ax_k) = P_k^T r_k$.

4) Ist $k = 1$, so folgt mit 2), dass $p_2 \in \text{Span}\{r_0, r_1\}$. Wegen $p_1 = r_0$ gilt dann $p_2 \in \text{Span}\{p_1, r_1\}$. Für $k > 1$ partitionieren wir den Vektor $z_k$ aus Lemma 4.4 als

$$z_k = \begin{bmatrix} w \\ \mu \end{bmatrix}, \quad w \in \mathbb{R}^{k-1},\ \mu \in \mathbb{R}.$$

Mit $r_k = r_{k-1} - \alpha_k A p_k$ wegen 1) erhalten wir aus Lemma 4.4:

$$\begin{aligned} p_{k+1} &= r_k - AP_k z_k \\ &= r_k - AP_{k-1} w - \mu A p_k \\ &= r_k - AP_{k-1} w + \frac{\mu}{\alpha_k}(r_k - r_{k-1}) \\ &= \left(1 + \frac{\mu}{\alpha_k}\right) r_k + s_k, \end{aligned}$$

wobei

$$\begin{aligned} s_k &= -\frac{\mu}{\alpha_k} r_{k-1} - AP_{k-1} w \\ &\in \text{Span}\{r_{k-1}, AP_{k-1} w\} \\ &\subseteq \text{Span}\{r_{k-1}, Ap_1, \ldots, Ap_{k-1}\} \\ &\subseteq \text{Span}\{r_0, \ldots, r_{k-1}\}. \end{aligned}$$

(Man beachte, dass $\alpha_k$ nach Konstruktion von Null verschieden ist!) Wegen 3) sind $r_k$ und $s_k$ dann orthogonal. Damit können wir das Optimierungsproblem in Lemma 4.4 lösen, indem wir $w$ und $\mu$ bestimmen, so dass

$$\|p_{k+1}\|^2 = \left(1 + \frac{\mu}{\alpha_k}\right)^2 \|r_k\|^2 + \|s_k\|^2$$

minimal wird. Dann ist aber insbesondere $s_k$ so, dass auch $\|s_k\|$ (bei festem $\mu$ und variablen $w$) minimal ist. Nun wird $\|r_{k-1} - AP_{k-1} z\|$ aber nach Lemma 4.4 durch $z = z_{k-1}$ minimiert und es ergibt sich $p_k = r_{k-1} - AP_{k-1} z_{k-1}$. Folglich ist $s_k$ ein Vielfaches von $p_k$. Damit haben wir aber

$$p_{k+1} \in \text{Span}\{r_k, s_k\} = \text{Span}\{r_k, p_k\}. \quad \square$$

**Folgerung:** Gegebenenfalls nach Skalierung von $p_{k+1}$ haben wir

$$p_{k+1} = r_k + \beta_k p_k.$$

Wegen $p_k^T A p_{k+1} = 0$ gilt außerdem:

$$\beta_k = -\frac{p_k^T A r_k}{p_k^T A p_k}.$$

Damit lässt sich $p_{k+1}$ unmittelbar aus $p_k$ und $r_k$ konstruieren, ohne dass wir die Minimierungsaufgabe $(*)$ explizit lösen müssen.

Wir fassen nun die erzielten Ergebnisse in folgendem Algorithmus zusammen:

**Algorithmus**: (CG, Konjugierte-Gradienten-Methode - Hestenes/Stiefels, 1952)
Berechnet für $A \in \mathbb{R}^{n\times n}$ symmetrisch, positiv definit und $b \in \mathbb{R}^n$ die Lösung $x = A^{-1}b$ des LGS $Ax = b$.

1) Start: $x_0 \in \mathbb{R}^n$, $r_0 = b - Ax_0$, $p_1 = r_0$

2) Iteriere, für $k = 1, 2, \ldots$ bis $n$ oder Konvergenz:

   (a) $\alpha_k = \dfrac{p_k^T r_k - 1}{p_k^T A p_k}$

   (b) $x_k = x_{k-1} + \alpha_k p_k$

   (c) $r_k = b - Ax_k$

   (d) $\beta_{k+1} = -\dfrac{p_k^T A r_k}{p_k^T A p_k}$

   (e) $p_{k+1} = r_k + \beta_{k+1} p_k$

**Bemerkung:** Die Kürze und Einfachheit des Algorithmus lässt vergessen, wie viele theoretische Resultate sich in seinem Hintergrund verstecken. So ist z.B. Konvergenz des Algorithmus nach spätestens $n$ Schritten garantiert, denn der *CG*-Algorithmus ist ja ein Spezialfall des Algorithmus mit $A$-konjugierten Suchrichtungen aus Abschnitt 4.2.2. Die Iterierte $x_k$ erfüllt daher die Bedingung

$$\varphi(x_k) = \min_{x \in x_0 + \mathcal{R}_k} \varphi(x),$$

wobei $\varphi(x) = \frac{1}{2}x^T Ax - x^T b$ und $\mathcal{R}_k = x_0 + \mathrm{Span}\{p_1, \ldots, p_k\}$. Nun ist aber $\mathrm{Span}\{p_1, \ldots, p_k\} = \mathcal{K}_k(A, r_0)$ nach Satz 4.5, d.h. wir minimieren $\varphi$ über dem affinen Krylovraum $x_0 + \mathcal{K}_k(A, r_0)$. Unsere Iterierte $x_k$ erfüllt also

$$\varphi(x_k) = \min_{x \in x_0 + \mathcal{K}_k(A, r_0)} \varphi(x).$$

Aus diesem Grund nennt man den *CG*-Algorithmus ein **Krylovraumverfahren**.

### 4.2.4 Konvergenzeigenschaften von CG

Der Zusammenhang des *CG*-Algorithmus mit Krylovräumen erlaubt eine detaillierte Konvergenzanalyse. Dazu führen wir zunächst eine spezielle Norm ein:

**Definition:** *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Dann heißt die durch*

$$||x||_A := \sqrt{x^T A x}$$

*definierte Norm auf $\mathbb{R}^n$ die A-Norm.*

**Ziel**: Abschätzung des Fehlers

$$e_k := A^{-1}b - x_k = A^{-1}(b - Ax_k) = A^{-1}r_k$$

wobei $(x_k)$ die durch *CG* erzeugte Iterationsfolge ist.

**Satz 4.6 (Optimalität von CG im Sinne der A-Norm)** *Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit und $(x_k)$ die für einen Startwert $x_0$ durch CG erzeugte Folge. Ist $r_{k-1} \neq 0$, so gilt:*

$$||e_k||_A = ||A^{-1}b - x_k||_A < ||A^{-1}b - x||_A$$

*für alle $x \in x_0 + \mathcal{K}_k(A, r_0)$ mit $x_k \neq x$.*

**Beweis**: Wir wissen: $x_k \in x_0 + \mathcal{K}_k(A, r_0)$. Sei nun $x \in x_0 + \mathcal{K}_k(A, r_0)$ beliebig und $\Delta x = x_k - x$, d.h. $\Delta x \in \mathcal{K}_k(A, r_0)$, sowie

$$\hat{e} := A^{-1}b - x = A^{-1}b - (x_k - \Delta x) = e_k + \Delta x$$

Dann gilt:

$$\begin{aligned} ||\hat{e}||_A^2 &= \hat{e}^T A\hat{e} = (e_k + \Delta x)^T A(e_k + \Delta x) \\ &= e_k^T A e_k + 2e_k^T A\Delta x + \Delta x^T A\Delta x \end{aligned}$$

und

$$2e_k^T A\Delta x = 2r_k^T A^{-1} A\Delta x = 2r_k^T \Delta x = 0$$

da $\Delta x \in \mathcal{K}_k(A, r_0) = \text{Span}\{r_0, \ldots, r_{k-1}\}$ und $r_k \perp r_j$ für $j = 0, \ldots, k-1$ gemäß Satz 4.5. Wir erhalten damit:

$$||\hat{e}||_A^2 = ||e_k||_A^2 + ||\Delta x||_A^2 > ||e_k||_A^2, \quad \text{falls } \Delta x \neq 0. \quad \square$$

**Korollar 4.7** *Sei $\widetilde{\Pi}_k := \{p \in \Pi_k | p(0) = 1\}$. Mit den Bezeichnungen und Voraussetzungen aus Satz 4.6 (insbesondere $r_{k-1} \neq 0$), gibt es genau ein Polynom $p_k \in \widetilde{\Pi}_k$ mit*

$$||p_k(A)\, e_0||_A = \min_{p\in\widetilde{\Pi}_k} ||p(A)\, e_0||_A$$

*Ferner gilt: $e_k = p_k(A)\, e_0$ und*

$$\frac{||e_k||_A}{||e_0||_A} = \min_{p\in\widetilde{\Pi}_k} \frac{||p(A)\, e_0||_A}{||e_0||_A} \leq \inf_{p\in\widetilde{\Pi}_k} \max_{\lambda\in\sigma(A)} |p(\lambda)| \qquad (*)$$

**Beweis**: Es gilt: $x_k \in x_0 + \mathcal{K}_k(A, r_0)$, d.h.

$$x_k = x_0 + \hat{p}_{k-1}(A)\, r_0$$

für ein $\hat{p}_{k-1} \in \Pi_{k-1}$. Außerdem gilt:

$$r_k = b - Ax_k = \underbrace{b - Ax_0}_{=:r_0} - A\hat{p}_{k-1}(A)\, r_0$$

Damit erhalten wir

$$e_k = A^{-1}r_k = \underbrace{A^{-1}r_0}_{=e_0} - \hat{p}_{k-1}(A)\, r_0 = e_0 - \hat{p}_{k-1}(A)\, Ae_0 = \underbrace{(I - \hat{p}_{k-1}(A)\, A)}_{=p_k(A)\in\tilde{\Pi}_k} e_0$$

Damit folgt die Eindeutigkeit von $p_k$, sowie die Gleichheit in $(*)$ aus dem vorigen Satz. Für die Ungleichung in $(*)$ sei $(v_1, \ldots, v_n)$ eine Orthonormalbasis aus Eigenvektoren von $A$ zu den Eigenwerten $\lambda_1, \ldots, \lambda_n$. Ferner sei $p \in \tilde{\Pi}_k$, sowie

$$e_0 = c_1 v_1 + \ldots + c_n v_n \qquad \text{mit } c_1, \ldots, c_n \in \mathbb{R}.$$

Dann gilt:

$$p(A)\, e_0 = c_1 p(\lambda_1)\, v_1 + \ldots + c_n p(\lambda_n)\, v_n.$$

Wegen der Orthogonalität der $v_i$ erhalten wir

$$||e_0||_A^2 = e_0^T A e_0 = \sum_{i=1}^{n} c_i^2 \lambda_i$$

und

$$||p(A)\, e_0||_A^2 = \sum_{i=1}^{n} c_i^2 p(\lambda_i)^2\, \lambda_i \leq \max_{\lambda \in \sigma(A)} p(\lambda)^2 \sum_{i=1}^{n} c_i^2 \lambda_i.$$

Daraus folgt aber

$$\frac{||p(A)\, e_0||_A^2}{||e_0||_A^2} \leq \max_{\lambda \in \sigma(A)} |p(\lambda)|^2. \quad \square$$

**Bemerkung:**

1) Aus Korollar 4.7 können wir folgern, dass $CG$ schnell konvergiert, falls $A$ ein „gutes“Spektrum hat, d.h. für das Polynome $p$ mit $p(0) = 1$ und kleinem Grad existieren, so dass $|p(\lambda)|$ für alle $\lambda \in \sigma(A)$ klein ist. Dies ist z.B. der Fall, falls

   (a) die Eigenwerte in Clustern auftreten,

   (b) alle Eigenwerte weit weg vom Ursprung liegen.
   (Dann ist $\kappa_2(A) = \frac{\lambda_{max}}{\lambda_{min}}$ nicht zu groß.)

2) Mit Hilfe von Tschebyscheffpolynomen kann man die folgende quantitive Abschätzung beweisen:

$$\frac{||e_k||_A}{||e_0||_A} \leq 2 \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k,$$

   wobei $\kappa := \kappa_2(A)$ und

$$\frac{||e_k||_2}{||e_0||_2} \leq 2\sqrt{\kappa} \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k.$$

3) Verbesserung der Konvergenzrate von CG erreicht man durch Vorkonditionierung:

   (a) Für allgemeine LGS: $Ax = b$ betrachte:

$$M^{-1} A x = M^{-1} b$$

   wobei $M^{-1}A$ ein “gutes” Spektrum hat und $Mz = c$ leicht zu lösen ist.

   (b) Für LGS $Ax = b$ mit $A$ symmetrisch und positiv definit betrachte:

$$\left(C^{-1} A C^{-T}\right)(Cx) = C^{-1} b$$

   wobei $C^{-1}AC^{-T}$ ein “gutes” Spektrum hat und $Cz = d$ leicht gelöst werden kann. $C^{-1}AC^{-T}$ ist wieder symmetrisch und positiv definit.

### 4.2.5 $CG$ und Lanczos

In diesem Abschnitt verwenden wir dieselben Bezeichnungen wie in den vorangegangenen Abschnitten. Betrachten wir dann einmal die folgenden Matrizen:

$$R_k = [r_0, \ldots, r_{k-1}], \quad P_k = [p_1, \ldots, p_k], \quad B_k = \begin{bmatrix} 1 & -\beta_2 & & 0 \\ & 1 & \ddots & \\ & & \ddots & -\beta_n \\ 0 & & & 1 \end{bmatrix}$$

Aus den Gleichungen $p_1 = r_0$ und $p_i = r_{i-1} + \beta_i p_{i-1}$ für $i = 2, \ldots, n$ (siehe Abschnitt 4.2.3) erhalten wir

$$R_k = P_k B_k.$$

Dann ist die Matrix $R_k^T A R_k$ aber tridiagonal, denn

$$R_k^T A R_k = B_k^T P_k^T A P_k B_k = B_k^T \begin{bmatrix} p_1^T A p_1 & & 0 \\ & \ddots & \\ 0 & & p_k^T A p_k \end{bmatrix} B_k.$$

Außerdem wissen wir aus Satz 4.5, dass die $r_0, \ldots, r_{k-1}$ orthogonal sind und einen Krylovraum aufspannen, d.h. $\frac{r_0}{\|r_0\|}, \ldots, \frac{r_{k-1}}{\|r_{k-1}\|}$ ist eine Orthonormalbasis von $\mathcal{K}_k(A, r_0)$.
Daraus ergibt sich eine sehr interessante Folgerung. Ist nämlich $q_1 := \frac{r_0}{\|r_0\|}$ und sind $q_1, \ldots, q_k$ die durch den Lanczos-Algorithmus erzeugten Vektoren, so gilt wegen des impliziten $Q$-Theorems

$$q_j = \pm \frac{r_{j-1}}{\|r_{j-1}\|}, \quad j = 1, \ldots, k.$$

Die beim Lanczosalgorithmus erzeugte Tridiagonalmatrix $T_k$ entspricht also (bis auf einige Vorzeichen) der Matrix $R_k^T A R_k$. Wir merken uns also:

$$\boxed{\text{„}CG = \text{Lanczos“}}$$

**Anwendung**: Im Laufe des $CG$-Algorithmus können wir die Tridiagonalmatrix $R_k^T A R_k$ berechnen und erhalten damit Informationen über extreme Eigenwerte von $A$. Insbesondere lässt erhalten wir dadurch Information über die Konditionzahl $\kappa_2(A) = \frac{\lambda_{max}}{\lambda_{min}}$.

### 4.2.6 GMRES

**Situation**: $A \in \mathbb{C}n \times n$ invertierbar, $n$ groß, $A$ schwach besetzt, $b \in \mathbb{C}^n$
($A$ kann also Hermitesch und i.A. indefinit oder auch nicht-Hermitesch sein.)

**Ziel**: Bestimme $x \in \mathbb{C}^n$ mit $Ax = b$.

Im Abschnitt 4.2 haben wir festgestellt, dass (gewisse affine) Krylovräume „gute Suchräume“ sind, d.h. wir finden dort gute Approximationen an die gesuchte Lösung. Es liegt daher nahe, auch für den allgemeinen Fall ein Krylovraumverfahren zu verwenden. Im $CG$-Algorithmus haben wir benutzt, dass die gesuchte Lösung $\hat{x} + A^{-1}b$ das eindeutig bestimmte Minimun der Funktion $\varphi = \frac{1}{2}x^T Ax - x^T b$. Dies gilt aber i.A. nur unter der Voraussetzung, dass $A \in \mathbb{R}^{n \times n}$ symmetrisch positiv definit ist.

**Idee**: Zu einem gegebenen Startvektor $x_0 \in \mathbb{C}^n$ und $r_0 := b - Ax_0$ bestimme $x_k$ mit

$$\|b - Ax_k\|_2 = \min_{x \in x_0 + \mathcal{K}_k(A, r_0)} \|b - Ax\|_2.$$

| | | |
|---|---|---|
| $A$ Hermitesch | $\rightsquigarrow$ | MINRES (mininmal residuals), Paige/Saunders 1975 |
| $A$ allgemein | $\rightsquigarrow$ | GMRES (generalized minimal residuals), Saad/Schultz 1986 |

**Frage**: Wie lösen wir das Least-Squares-Problem $\|b - Ax_k\|_2 = \min_{x \in x_0 + \mathcal{K}_k(A, r_0)} \|b - Ax\|_2$?

**Antwort**: In Abschnitt 4.2.5. haben wir festgestellt, dass $CG$ im wesentlichen dem Lanczos-Algorithmus entspricht. Nun befassen wir uns mit unsymmetrischen Matrizen, also erwarten wir:

$$\boxed{\text{„GMRES} = \text{Arnoldi“}}$$

Nach $k$ Schritten des Arnoldi-Algorithmus haben wir die Arnoldi-Konfiguration

$$AQ_k = Q_k H_k + h_{k+1,k} q_{k+1} e_k^T = Q_{k+1} H_{k+1}$$

mit $Q_k = [q_1, \dots q_k]$, $Q_{k+1} = [Q_k, q_{k+1}]$ isometrisch und

$$H_{k+1,k} = \begin{bmatrix} h_{11} & \dots & \dots & h_{1k} \\ h_{21} & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & h_{k,k-1} & h_{kk} \\ 0 & \dots & 0 & h_{k+1,k} \end{bmatrix} \in \mathbb{C}^{(k+1)\times k}.$$

Ist $q_1 = \frac{r_0}{\|r_0\|}$, so gilt $\text{Span}\{q_1, \dots, q_k\} = \mathcal{K}_k(A, r_0)$. Sei nun $x \in x_0 + \mathcal{K}_k(A, r_0)$, d.h. $x = x_0 + q_k y$ für ein $y \in \mathbb{C}^k$. Dann gilt

$$\begin{aligned} \|b - Ax\| &= \|b - A(x_0 + q_k y)\| \\ &= \|r_0 - AQ_k y\| \\ &= \|r_0 - Q_{k+1} H_{k+1,k} y\| \\ &= \|Q_{k+1}^* r_0 - H_{k+1,k} y\| \qquad \text{da } Q_{k+1} \text{ isometrisch} \\ &= \left\| \|r_0\| \cdot e_1 - H_{k+1,k} y \right\| \qquad \text{da } q_2, \dots, q_{k+1} \perp q_1 = \tfrac{r_0}{\|r_0\|}. \qquad (**) \end{aligned}$$

**Erinnerung**: Lösung von Least-Squares-Problemen $\|c - My\| \stackrel{!}{=} \min$, $M \in \mathbb{C}^{k\times n}$, $k \le n$:

1) berechne eine $QR$-Zerlegung von $M$:

$$M = QR, \quad Q \in \mathbb{C}^{n\times n} \text{ unitär}, \quad R = \begin{matrix} k \\ n-k \end{matrix} \overset{k}{\begin{bmatrix} R_1 \\ 0 \end{bmatrix}};$$

2) Da $Q$ unitär ist, gilt

$$\|c - My\|^2 = \|Q^*c - Ry\|^2 = \left\| \begin{bmatrix} c_1 - R_1 y \\ c_2 \end{bmatrix} \right\|^2, \qquad \text{wobei} \quad Q^*c = \begin{matrix} k \\ n-k \end{matrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix}.$$

Falls $R_1$ invertierbar ist, so wird dies minimal, wenn $R_1 y = c$. Löse also das lineare Gleichungssystem $R_1 y = c$.

Kommen wir zurück zu unserem Least-Squares-Problem $(**)$. Die Matrix $H_{k+1,k}$ ist in Hessenbergform und wir wollen das LS-Problem für alle $k$ lösen. Angenommen, wir haben das Problem bereits für $k-1$ gelöst, d.h. wir haben eine $QR$-Zerlegung für $H_{k,k-1}$ berechnet:

$$H_{k,k-1} = \widetilde{Q}_k \widetilde{R}_k, \quad \widetilde{Q}_k \text{ unitär}, \quad \widetilde{R}_{k-1} = \begin{bmatrix} R_{k-1} \\ 0 \end{bmatrix}, \quad R_{k-1} \text{ obere Dreiecksmatrix.}$$

Dann gilt:

$$\begin{bmatrix} \widetilde{Q}_k^* & 0 \\ 0 & 1 \end{bmatrix} \cdot H_{k+1,k} = \begin{bmatrix} \widetilde{Q}_k^* & 0 \\ 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} H_{k,k-1} & h_{kk} \\ \hline 0 & h_{k+1,k} \end{array}\right] = \left[\begin{array}{c|c} \widetilde{R}_{k-1} & \widetilde{Q}_k^* h_{kk} \\ \hline 0 & h_{k+1,k} \end{array}\right]$$

$$= \begin{matrix} \\ k-1 \\ 1 \\ 1 \end{matrix} \begin{matrix} \begin{matrix} k-1 & 1 \end{matrix} \\ \begin{bmatrix} R_{k-1} & * \\ 0 & * \\ 0 & h_{k+1,k} \end{bmatrix} \end{matrix}.$$

Das Element $h_{k+1,k}$ kann nun durch eine einzige Givens-Rotation eliminiert werden. Wir erhalten also eine $QR$-Zerlegung von $H_{k+1,k}$ aus der bereits berechneten von $H_{k,k-1}$ durch Anwenden einer Givens-Rotation und durch Berechnung von $\widetilde{Q}_k^* h_{kk}$ (kostet $\mathcal{O}(n)$ flops). Zusammenfassend erhalten wir den folgenden Algorithmus.

**Algorithmus** (GMRES) Berechnet für $A \in \mathbb{C}^{n \times n}$ invertierbar, $b \in \mathbb{C}^n$ und einen Startvektor $x_0 \in \mathbb{C}^n$ die Lösung $\hat{x} = A^{-1}b$ von $Ax = b$.

1) Start: $r_0 = b - Ax_0$, $h_{10} = \|r_0\|$.

2) Iteriere: für $k = 1, 2, \ldots$ bis Konvergenz:

   a) $q_k = \dfrac{r_k}{h_{k,k-1}}$

   b) $r_k = Aq_k - \displaystyle\sum_{j=1}^{k} h_{jk} q_j$ mit $h_{jk} = Q_j^* r_k$

   c) $h_{k+1,k} = \|r_k\|$

   d) bestimme $y_k$, so dass $\left\| \|r_0\| \cdot e_1 - H_{k+1,k} y_k \right\|$ minimal wird

   e) $x_k = x_0 + Q_k y_k$

**Bemerkung:** Wie *CG* lässt sich auch GMRES auf polynomiale Approximation in $\widetilde{\Pi}_k = \{p \in \Pi_k | p(0) = 1\}$ zurückführen:

$$x = x_0 + \hat{p}(A)r_0 \qquad \text{für ein } \hat{p} \in \Pi_{k-1},$$

da $x = x_0 + \mathcal{K}_k(A, r_0)$. Damit folgt

$$r_k := b_a x_k = b - Ax_0 - A\hat{p}(A)r_0 = \Big(I - A\hat{p}(A)\Big)r_0 = p(A)r_0 \qquad \text{für ein } p \in \widetilde{\Pi}_k.$$

Damit lässt sich GMRES umfornulieren zu der Aufgabe:

$$\text{Finde } p \in \widetilde{\Pi}_k, \text{ so dass } \|p(A)r_0\| \text{ minimal wird.}$$

Denn ist $p_k \in \widetilde{\Pi}_k$, so dass $r_k = p_k(A)r_0$, so gilt

$$\|r_k\| = \|p_k(A)r_0\| \leq \|p(A)r_0\|$$

für alle $p \in \widetilde{\Pi}_k$.

**Satz 4.8** *Sei $A \in \mathbb{C}^{n\times n}$ diagonalisierbar und $V^{-1}AV = \Lambda$ diagonal. Dann gilt:*

$$\frac{\|r_k\|}{\|r_0\|} \leq \kappa(V) \inf_{p\in\widetilde{\Pi}_k} \max_{\lambda\in\sigma(A)} \Big|p(\lambda)\Big|.$$

**Beweis**: Für jedes Polynom $p \in \widetilde{\Pi}_k$ gilt:

$$\begin{aligned} \|p(A)\| &= \|p(V\Lambda V^{-1}\| = \|Vp(\Lambda)V^{-1}\| \\ &\leq \|V\| \cdot \|p(\Lambda)\| \cdot \|V^{-1}\| = \kappa(V) \cdot \|p(\Lambda)\|, \end{aligned}$$

sowie

$$\|p(\Lambda)\| = \max_{\lambda\in\sigma(A)} |p(\lambda)|,$$

da $\Lambda$ diagonal ist. Damit erhalten wir

$$\begin{aligned} \|r_k\| &= \|p_k(A)r_0\| \leq \inf_{p\in\widetilde{\Pi}_k} \|p(A)r_0\| \leq \inf_{p\in\widetilde{\Pi}_k} \|p(A)\| \cdot \|r_0\| \\ &\leq \|r_0\| \cdot \kappa(V) \inf_{p\in\widetilde{\Pi}_k} \max_{\lambda\in\sigma(A)} |p(\lambda)|. \quad \square \end{aligned}$$

**Folgerung**: GMRES konvergiert schnell, falls

1) sich dass Spektrum von $A$ „vernünftig“verhält;

2) $\kappa(V)$ klein ist, d.h. wenn $A$ nicht zu weit von einer normalen Matrix entfernt ist (denn ist $A$ normal, so kann die diagonalisierende Matrix $V$ unitär gewählt werden, hat also Konditionszahl eins).

**Bemerkung:** Konvergenzbeschleunigung erhalten wir wieder durch Präkonditionierung, d.h. statt $Ax = b$ lösen wir das System $M^{-1}Ax = M^{-1}b$, wobei sich das LGS $My = c$ leicht lösen lassen muss.

**Bemerkung:** Methoden zur Lösung von $Ax = b$, $A$ invertierbar mit $A \neq A^*$:

1) *CGN* (das *N* steht hier für „Normalengleichung“)
Statt $Ax = b$ betrachte die Normalengleichung, also das LGS $A^*Ax = A^*b$ mit der positiv definiten Matrix $A^*A$ und löse dieses mit dem *CG*-Algorithmus.

   **Nachteil**: Quadrierung der Konditionszahl: $\kappa(A^*A) = \kappa(A)^2$.

   **Vorteil**: Die Eigenwerte von $A^*A$ sind gerade die Quadrate der Singulärwerte von $A$. Daher ist *CGN* sinnvoll für Matrizen $A$ mit „schlechtem Spektrum“, aber „guten Singulärwerten“.

2) *BiCG* (Biconjugate gradients)

   *CG*: das berechnete $x_k \in x_0 + \mathcal{K}_k(A, r_0)$ liefert $r_k \perp r_0, \ldots, r_{k_1}$,
   also $r_k \perp \mathcal{K}_k(A, r_0)$

   *BiCG*: wähle $s_0$ mit $s_0^* r_0 = 1$ und bestimme das $x_k \in x_0 + \mathcal{K}_k(A, r_0)$
   mit $r_k \perp \mathcal{K}_k(A, s_0)$

   Diese Vorgehensweise entspricht dem unsymmetrischen Lanczos-Algorithmus. Wir erhalten damit folgende Tabelle von Entsprechungen:

| | $Ax = \lambda x$ | $Ax = b$ |
|---|---|---|
| $A = A^*$ | Lanczos | CG |
| $A \neq A^*$ | Arnoldi | GMRES |
| | Lanczos | BiCG |

3) Übersicht über verschiedene Klassen von Krylovraummethoden

   gemeinsamer Nenner: $\mathcal{K} = \mathcal{K}_k(A, r_0)$ Krylovraum

   entscheidende Größe: $r_k = b - Ax_k$ (Residuum)

   a) Ritz-Galerkin-Ansatz: wähle $x_k \in x_0 + \mathcal{K}$, so dass $r_k \perp \mathcal{K}$
      $\rightsquigarrow$ CG, FOM, GENCG

   b) Minimale-Residuen-Ansatz: wähle $x_k \in x_0 + \mathcal{K}$, so dass $\|r_k\|$ minimal ist
      $\rightsquigarrow$ MINRES, GMRES, ORTHODIR

   c) Petrov-Galerkin-Ansatz: wähle $x_k \in x_0 + \mathcal{K}$, so dass $r_k \perp \mathcal{L}$, $\mathcal{L} \subseteq \mathbb{C}^n$, $\dim \mathcal{L} = k$
      $\rightsquigarrow$ BiCG, QMR

   d) Minimalfehler-Ansatz: wähle $x_k \in x_0 + \mathcal{K}$, so dass $\|x_k - A^{-1}b\|$ minimal ist
      $\rightsquigarrow$ SYMMLQ, GMERR

   Weiter gibt es noch hybride Methoden, wie (CGS,Bi-CGSTAB,...)

Zum Abschluss sei noch einmal auf das Zauberwort hingewiesen, um das niemand herumkommt, der sich intensiv mit der Lösung von linearen Gleichungssystemen beschäftigt:

**Präkonditionierung**