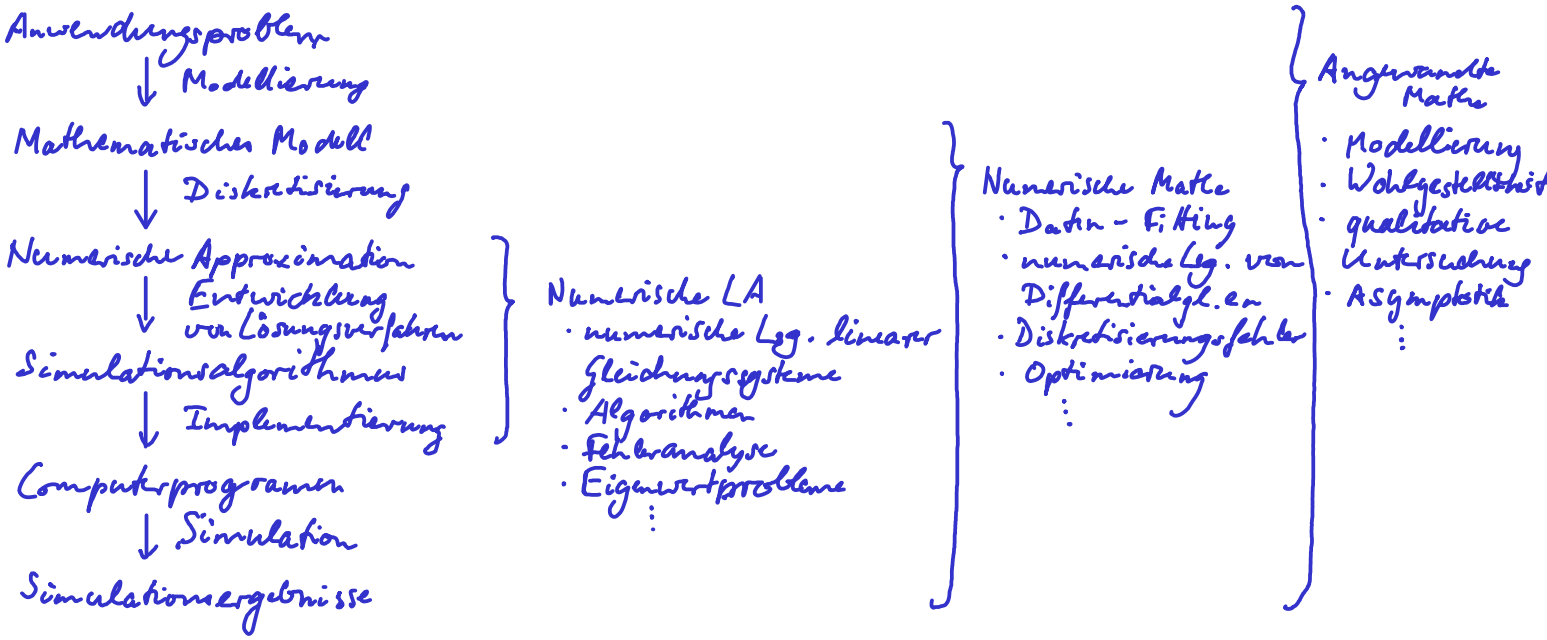
Numerische Lineare Algebra

Organisatorisches:
- Übung beginnt 26.10.15; Zettel & Abgabe Do
- Kursbuchungssystem für Übungsanmeldung wird freigeschaltet 18:00
- jeder soll mind. 2x vorrechnen

Heute: · Einführung

Einführung

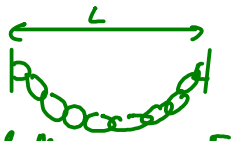
Numerische LA ⊂ Numerische Mathe ⊂ Angewandte Mathe



Numerische Mathematik befasst sich mit der Formulierung und Lösung mathematischer Modelle am Computer. Da Computer nicht beliebig schnell und genau rechnen können, entwickelt die Numerik effiziente Methoden zur Lösung und untersucht ihre analytischen Eigenschaften wie Konvergenz, Konvergenzrate, Fehlerrobustheit, Komplexität usw. als Funktion des Rechenaufwands.

Viele Anwendungen und numerische Verfahren führen auf ein zu lösendes Gleichungssystem. **Numerische LA** liefert & untersucht Lösungsverfahren hierfür. Die zählt zu den grundlegenden numerischen Problemen und Werkzeugen.

Bsp: Hängende Kette



$n+1$ Glieder der Länge 1 & Masse m, Enden im Abstand L

Modellierung: · Position des rechten Gliedendes: $\begin{pmatrix} x_i \\ y_i \end{pmatrix}$, $i = 0, \dots, n$, $\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = 0$, $\begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} L \\ 0 \end{pmatrix}$ (*)

$$(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 = 1 \qquad (**)$$

· potentielle Energie: $E = mg\left[\frac{y_0+y_1}{2} + \frac{y_1+y_2}{2} + \dots + \frac{y_n+y_{n+1}}{2}\right] = mg \sum_{i=1}^{n} y_i$

$\Rightarrow$ minimiere E nach x_i, y_i, $i = 1, \dots, n$, sodass (*), (**)

Diskretisierung: · Modell hat bereits nur endlich viele „diskrete" Variablen

· Bei allgemeinen Problemen gibt es unendlich viele Variablen, z.B. wenn wir statt der Kette ein kontinuierliches Band wählen (kann man auffassen als ∞ viele ∞ kurze Kettenglieder)

· Vorlesungen „Numerische Analysis" & „Numerik partieller Dgl" untersuchen, wie man dies mit diskreten Variablen approximiert

$\Rightarrow$ liefert ähnliche Gleichungen

Lösungsverfahren: · Lagrange-Multiplikator $\lambda = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix}$ für (**)

Lagrange-Funktional $L(x_i, y_i, \lambda_i) = E(y_i) + \sum_{i=1}^{n+1} \lambda_i \left[(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 - 1\right]$

$$\Rightarrow \text{löse} \left\{ \begin{array}{ll} \partial L/\partial y_i = 0 \\ \partial L/\partial x_i = 0 \\ \partial L/\partial \lambda_j = 0 \end{array} \right\}, \; \begin{array}{l} i = 1, \dots, n \\ j = 1, \dots, n+1 \end{array} \quad \text{nach } x_i, y_i, \lambda_j, \; \begin{array}{l} i = 1, \dots, n \\ j = 1, \dots, n+1 \end{array}$$

· in der Vorlesung beschäftigen wir uns u.A. damit, wie man solche Gleichungssysteme löst, z.B. mittels spezieller Fixpunkt-Iteration:

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \\ y_1 \\ \vdots \\ y_n \\ \lambda_1 \\ \vdots \\ \lambda_{n+1} \end{pmatrix}^{(k+1)} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \\ y_1 \\ \vdots \\ y_n \\ \lambda_1 \\ \vdots \\ \lambda_{n+1} \end{pmatrix}^{(k)} - \underbrace{\alpha}_{(0,1]} \underbrace{\left(\frac{\partial^2 L}{\partial (x_i, y_i, \lambda_i)^2} \Big|_{(x_0, \dots, \lambda_{n+1})^{(k)}} \right)^{-1}}_{\text{Hesse-Matrix } D^2 L} \underbrace{\left(\frac{\partial L}{\partial (x_i, y_i, \lambda_i)} \Big|_{(x_0, \dots, \lambda_{n+1})^{(k)}} \right)}_{\text{Gradient } \nabla L}$$

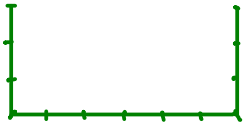
hat Lösung als Fixpunkt und konvergiert (für gut gewähltes $(x_0, \dots, \lambda_{n+1})^{(0)}$) gegen Lösung

· Numerische LA im Speziellen beschäftigt sich mit Methoden zum Lösen des linearen Gleichungssystems $D^2 L \begin{pmatrix} \cdot \\ \cdot \end{pmatrix} = \nabla L$, also mit dem kleinsten & elementarsten Problem aus dem ganzen Ablauf. Für dieses Beispiel genügt uns jedoch erstmal, dass Matlab dies lösen kann.

Implementierung:

```
% Parameter
m = 1; g = 1; L = 10; n = 19;

% Initialisierung von x, y, lambda
x = [0 0 0 0,linspace(0,L,11),L L L L]';
y = -[1:5,5*ones(1,9),5:-1:1]';
l = .1*ones(n+1,1);
```



```
% Fixpunktiteration
for i = 1:100
  xLeft = [0;x(1:end-1)]; xRight = [x(2:end);L];
  yLeft = [0;y(1:end-1)]; yRight = [y(2:end);0];
  lLeft =    l(1:end-1) ; lRight =  l(2:end);

  dL_dx = 2*lLeft.*(x-xLeft) - 2*lRight.*(xRight-x);
  dL_dy = 2*lLeft.*(y-yLeft) - 2*lRight.*(yRight-y) + m*g;
  dL_dl = [(x-xLeft).^2+(y-yLeft).^2;(L-x(end))^2+y(end)^2]-1;

  dL_dxx = diag(2*lLeft+2*lRight);
  dL_dyy = diag(2*lLeft+2*lRight);
  dL_dll = zeros(n+1,n+1);
  dL_dxy = zeros(n,n);
  dL_dxl = spdiags(2*[x-xLeft x-xRight],[0 1],n,n+1);
  dL_dyl = spdiags(2*[y-yLeft y-yRight],[0 1],n,n+1);

  DL = [dL_dx;dL_dy;dL_dl];
  D2L = [dL_dxx  dL_dxy  dL_dxl;
         dL_dxy' dL_dyy  dL_dyl;
         dL_dxl' dL_dyl' dL_dll];
  sol = D2L\DL*.5;
  x = x - sol(1:n); y = y - sol(n+1:2*n); l = l - sol(2*n+1:end);

  plot([0;x;L],[0;y;0],'.-','linewidth',3,'Markersize',10);
  axis equal;
  drawnow;
end
```

Warum konvergiert es für einige α & Initialisierungen, für andere nicht?

Wie schnell konvergiert es?

$\Rightarrow$ typische Numerik-Fragen

Heute: • normierte Räume

Wiederholung Grundlagen

Normierte Räume

Def: (Norm) Sei V ein $\mathbb{K}$-Vektorraum. $\|\cdot\| : V \to \mathbb{R}$ heißt Norm, falls

- $\|v\| > 0 \quad \forall\, v \in V \setminus \{0\}$
- $\|\lambda v\| = |\lambda|\,\|v\| \quad \forall\, \lambda \in \mathbb{K},\ v \in V$
- $\|v + w\| \leq \|v\| + \|w\| \quad \forall\, v, w \in V$

Bsp: • $V = \mathbb{R}^n$, $\|v\|_\infty = \max(v_1, \dots, v_n)$

$\|v\|_1 = |v_1| + \dots + |v_n|$

$\|v\|_p = \left(|v_1|^p + \dots + |v_n|^p\right)^{\frac{1}{p}}$, $p \geq 1$, sind Normen

• $V = C^0([a,b])$, $a < b$, $\|v\|_\infty = \sup_{x \in [a,b]} |v(x)|$

$\|v\|_p = \left(\int_a^b |v(x)|^p \, dx\right)^{\frac{1}{p}}$, $p \geq 1$, sind Normen

Def: (äquivalente Normen) $\|\cdot\|$ und $|||\cdot|||$ sind äquivalent, wenn $\exists\, c, C > 0$ mit $c\|v\| \leq |||v||| \leq C\|v\| \quad \forall v \in V$

Thm: $\dim V < \infty \Rightarrow$ alle Normen sind äquivalent

Def: (Banachraum) • $(u_n)_{n \in \mathbb{N}}$ konvergiert gegen $u \in V :\Leftrightarrow \forall \varepsilon > 0\ \exists N > 0\ \forall n > N : \|u_n - u\| < \varepsilon$

• $(u_n)_{n \in \mathbb{N}}$ heißt Cauchy-Folge $:\Leftrightarrow \forall \varepsilon > 0\ \exists N > 0\ \forall m, n > N : \|u_m - u_n\| < \varepsilon$

• $(V, \|\cdot\|)$ heißt vollständig, wenn alle Cauchyfolgen konvergieren.

$(V, \|\cdot\|)$ heißt dann Banachraum.

Bsp: • $(\mathbb{R}^n, \|\cdot\|)$ ist Banachraum

• $(C^0([a,b]), \|\cdot\|_\infty)$ ist Banachraum

• $(C^0([a,b]), \|\cdot\|_p)$ ist nicht vollständig

Def: (Skalarprodukt) $(\cdot,\cdot) : V \times V \to \mathbb{K}$ heißt Skalarprodukt, wenn

- $(v, v) > 0 \quad \forall\, v \in V \setminus \{0\}$
- $(u, v) = \overline{(v, u)} \quad \forall\, u, v \in V$
- $(\alpha u, v) = \alpha (u, v)$ und $(u + w, v) = (u, v) + (u, w) \quad \forall\, u, v, w \in V,\ \alpha \in \mathbb{K}$

Bsp: • $V = \mathbb{R}^m$, $(v, w) = v^T w$

• $V = \mathbb{C}^m$, $(v, w) = v^T \overline{w}$

Def: (Duale Norm) $\|\cdot\|'$ mit $\|x\|' = \sup_{\|y\| = 1} |(x, y)|$ heißt zu $\|\cdot\|$ duale Norm

Bsp: • $\|\cdot\|_2' = \|\cdot\|_2$, $\|\cdot\|_1' = \|\cdot\|_\infty$, $\|\cdot\|_\infty' = \|\cdot\|_1$ auf $\mathbb{K}^m$

Thm: · Durch $\|v\| = \sqrt{(v,v)}$ wird eine Norm induziert

· $|(u,v)| \leq \|u\|\,\|v\|$ (Cauchy-Schwarz-Ungl.)

Def: (Hilbertraum) Ein Banachraum mit Skalarprodukt-induzierter Norm heißt Hilbertraum

Bsp: $L^2([a,b]) = \{ f\ [a,b] \to \mathbb{R} \text{ messbar}, \int_a^b f^2\,dx < \infty \}$ ist ein Hilbert-Raum mit Skalarprodukt $(f,g) = \int_a^b fg\,dx$.

Operatoren

Def: (linearer Operator) Seien U, V normierte Vektorräume.

· $T: U \to V$ heißt linearer Operator, wenn $T(\alpha x + y) = \alpha T(x) + T(y)\ \forall x,y \in V, \alpha \in \mathbb{K}$

· $\|T\| = \sup_{u \in U, u \neq 0} \frac{\|Tu\|}{\|u\|} = \sup_{\|u\|=1} \|Tu\|$ heißt Operatornorm von T

· T heißt beschränkt, wenn $\|T\| < \infty$

· $L(U,V) = \{ T: U \to V \mid T \text{ linear und beschränkt} \}$

Bem: Duale Norm $\|v\|'$ = Operatornorm von $T: (U, \|\cdot\|) \to (\mathbb{K}, |\cdot|)$, $x \mapsto (x,v)$

Thm: (beschränkte Operatoren)

· T beschränkt $\Leftrightarrow$ T stetig in 0 $\Leftrightarrow$ T stetig

· V vollständig $\Rightarrow$ mit der Operatornorm ist $L(U,V)$ ein Banachraum

Bew: · T beschränkt $\Rightarrow$ für $\varepsilon > 0$ ist $\|T(x) - T(0)\| \leq \|T\|\,\|x\| < \varepsilon\ \forall x$ mit $\|x - 0\| < \frac{\varepsilon}{\|T\|}$

$\Rightarrow$ T stetig in 0, d.h. für $\varepsilon > 0$ existiert δ mit $\|x-0\| < \delta \Rightarrow \|T(x) - T(0)\| < \varepsilon$

$\Rightarrow$ $\|T(x) - T(y)\| = \|T(x-y) - T(0)\| < \varepsilon$ für $\|x - y - 0\| < \delta$

$\Rightarrow$ T stetig

$\Rightarrow$ $\|Tx\| = \|T(x) - T(0)\| = \frac{2\|x\|}{\delta} \left\| T\left(\frac{\delta}{2}\frac{x}{\|x\|}\right) - T(0) \right\| \leq \frac{2\|x\|}{\delta}\varepsilon \quad \forall x \in U$

$\Rightarrow$ T beschränkt ($\|\frac{\delta}{2}\frac{x}{\|x\|} - 0\| < \delta$)

· Normeigenschaften einfach zu zeigen.

Sei $(T_n)_{n \in \mathbb{N}}$ Cauchyfolge $\Rightarrow$ $T_n(x)$ ist Cauchyfolge für alle $x \in U$.

Definiere $T(x) = \lim_{n \to \infty} T_n(x)$. $T(x)$ ist linear wegen Linearität von T_n und lim.

$\|T(x)\| \leq \|T(x) - T_n(x)\| + \|T_n(x)\| = \lim_{n \to \infty} \|T_n(x) - T_m(x)\| + \|T_m(x)\|$;

wähle m groß genug, dass $\|T_n - T_m\| < \varepsilon\ \forall n > m$

$\Rightarrow \|T(x)\| \leq \varepsilon\|x\| + \|T_m\|\,\|x\| \Rightarrow T \in L(U,V)$.

Für $\varepsilon > 0$ wähle N sodass $\|T_m - T_n\| < \varepsilon\ \forall m,n > N$

$\Rightarrow \|T_m(x) - T(x)\| = \lim_{n \to \infty} \|T_m(x) - T_n(x)\| < \varepsilon\|x\|\ \forall m > N \Rightarrow T_m \xrightarrow{m \to \infty} T$ □

Bsp: · Sei $T: U \to V$ linear, U, V endlichdimensional. Dann kann T durch eine Matrix A bezüglich vorgegebener Basen dargestellt werden.

· $x \mapsto Ax$ mit $A \in \mathbb{K}^{m \times n}$ ist linearer Operator $\mathbb{K}^n \to \mathbb{K}^m$

· Aus obigem folgt $L(\mathbb{K}^n, \mathbb{K}^m) \equiv \mathbb{K}^{m \times n}$

Def: (induzierte Matrixnorm) Die Operatornorm von Operatoren $A \in \mathbb{K}^{m \times n}$, $A: (\mathbb{K}^n, \|\cdot\|) \to (\mathbb{K}^m, \|\cdot\|)$, heißt durch $\|\cdot\|$ induzierte Matrixnorm.

Bsp: Durch $\|\cdot\|_1$ wird die maximale Spaltensumme induziert, $\|A\|_1 = \max_{i=1,\dots,n} \|A_{:i}\|_1$.

Def: (adjungierter Operator) Seien $(U, (\cdot,\cdot)_U), (V, (\cdot,\cdot)_V)$ Vektorräume mit Skalarprodukt, $T: U \to V$ linear. Der adjungierte Operator $T^*: V \to U$ ist definiert durch $(Tu, v)_V = (u, T^* v)_U \quad \forall u \in U, v \in V$.

Bsp: · $U = \mathbb{K}^n$, $V = \mathbb{K}^m$, $T: U \to V$, $u \mapsto Au$ für $A \in \mathbb{K}^{m \times n}$

$$\Rightarrow (Tu, v)_U = (Au)^T \overline{v} = u^T A^T \overline{v} = u^T \overline{\bar{A}^T v} = (u, \bar{A}^T v)_V$$

$$\Rightarrow T^*: V \to U, \; v \mapsto \bar{A}^T v$$

Heute: · Singulärwertzerlegung

Endlich-dimensionale Vektorräume & Operatoren

Der Einfachheit halber beschränken wir uns im Folgenden auf Vektoren $\in \mathbb{K}^m$ und Matrizen $\in \mathbb{K}^{m \times n}$

$\|\cdot\|$ bezeichne die Skalarprodukt-induzierte Norm.

Def: (hermitesche Matrizen) · $A \in \mathbb{C}^{m \times n}$ heißt hermitesch, wenn $A^* := \bar{A}^T = A$

· $A \in \mathbb{R}^{m \times n}$ heißt symmetrisch, wenn $A^T = A$

Def: (orthogonale Vektoren) · $u, v \in \mathbb{K}^m$ heißen orthogonal, wenn $(u, v) = 0$.

· $S \subset \mathbb{K}^m \setminus \{0\}$ heißt orthogonal, wenn $u, v \in S \Rightarrow (u, v) = 0$.

· S heißt orthonormal, wenn zusätzlich $\|u\| = 1 \; \forall u \in S$

Thm: S orthogonal $\Rightarrow$ S linear unabhängig

Bew: Sei $S = \{v_1, \dots, v_n\}$ nicht lin. unabh., d. h. $v_k = \sum_{\substack{i=1 \\ i \neq k}}^{n} c_i v_i$, $c_i \in \mathbb{K}$.

$$\Rightarrow (v_k, v_k) = \sum_{\substack{i=1 \\ i \neq k}}^{n} c_i (v_i, v_k) = 0 \quad \lightning \qquad \square$$

Kor: $S = \{q_1, \dots, q_m\} \subset \mathbb{K}^m$ orthogonal $\Rightarrow$ S ist Basis von $\mathbb{K}^m$

Bem: Jeder Vektor $v \in \mathbb{K}^m$ kann zerlegt werden in Komponenten parallel zu orthonormalen $q_1, \dots, q_m$:

$$v = \sum_{i=1}^{m} (q_i, v) q_i = \sum_{i=1}^{m} (q_i q_i^*) v$$

Bew: $v - \sum_{i=1}^{m} (q_i, v) q_i$ ist offenbar orthogonal zu $q_1, \dots, q_m$ und somit gleich 0. $\square$

Def: (unitäre Matrix) · $A \in \mathbb{R}^{m\times m}$ heißt orthogonal, wenn $A^T = A^{-1}$

· $A \in \mathbb{C}^{m\times m}$ heißt unitär, wenn $A^* = A^{-1}$

Bem: Die Spalten & Zeilen unitärer Matrizen A sind orthonormal.

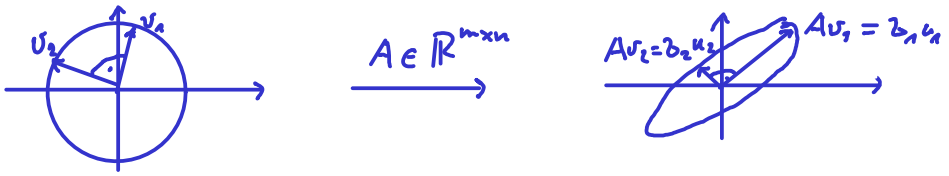
Bew: δ_{ij} = Kronecker delta $= \begin{cases} 1 & i=j \\ 0 & \text{sonst} \end{cases} = (I)_{ij} = (AA^*)_{ij} = (A_{i:}, A_{j:})$

$= (A^*A)_{ij} = (\overline{A_{:i}}, \overline{A_{:j}}) = (A_{:i}, A_{:j})$ □

Thm: $Q \in \mathbb{C}^{m\times m}$ unitär, $x \in \mathbb{C}^m \Rightarrow \|Qx\| = \sqrt{(Qx, Qx)} = \sqrt{x^*Q^*Qx} = \sqrt{x^*x} = \|x\|$

Singulärwertzerlegung

Das Bild der Einheitskugel im $\mathbb{R}^n$ unter jeder $m \times n$ Matrix ist eine Hyperellipse, d.h. eine Oberfläche, die durch Dehnung der Einheitskugel im $\mathbb{R}^m$ in orthogonale Richtungen $u_1, \dots, u_m$ um den Faktor $\sigma_1, \dots, \sigma_m$ entsteht. Die Semiachsen sind $\sigma_1 u_1, \dots, \sigma_m u_m$.



Def: (Singulärwerte) Eine Singulärwertzerlegung einer Matrix $A \in \mathbb{K}^{m\times n}$ ist definiert durch $A = U\Sigma V^*$,

$U = [u_1 | \dots | u_m] \in \mathbb{K}^{m\times m}$ unitär, $V = [v_1 | \dots | v_n] \in \mathbb{K}^{n\times n}$ unitär, $\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \end{bmatrix} \in \mathbb{R}^{m\times n}$

Singulärwerte: $\sigma_1 \geq \sigma_2 \geq \dots \geq 0$

linke Singulärvektoren: $u_1, \dots, u_m$

rechte ——"—— : $v_1, \dots, v_n$

Thm: Jeden $A \in \mathbb{C}^{m\times n}$ besitzt eine Singulärwertzerlegung (SVD). Die Singulärwerte sind eindeutig.

Ist $m = n$ und $\sigma_i \neq \sigma_j \ \forall i \neq j$, sind auch die Singulärvektoren bis auf ein komplexes Vorzeichen eindeutig bestimmt.

Bem: Analog für $A \in \mathbb{R}^{m\times n}$

Bew: Existenz per Induktion nach $\min(m,n)$:

Sehe $\sigma_1 = \|A\| \Rightarrow$ nach dem Satz von Weierstraß gibt es $u_1 \in \mathbb{C}^m$, $v_1 \in \mathbb{C}^n$ mit $\|u_1\| = \|v_1\| = 1$, $Av_1 = \sigma_1 u_1$

Erweitere u_1 und v_1 zu einer Orthonormalbasis $\{u_j\}$ von $\mathbb{C}^m$ und $\{v_j\}$ von $\mathbb{C}^n$.

$U_1 = [u_1 | \dots | u_m]$ & $V_1 = [v_1 | \dots | v_n]$ sind unitär,

$$U_1^* A V_1 = U_1^* [Av_1 | \dots | Av_n] = \begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix}$$ für ein $w \in \mathbb{C}^{n-1}$, $B \in \mathbb{C}^{(m-1)\times(n-1)}$

$$\left\| \begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix} \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\| \geq \sigma_1^2 + w^*w = \sqrt{\sigma_1^2 + w^*w} \left\| \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\| \Rightarrow \sigma_1 = \|A\| = \|U_1^* A V_1\| \geq \sqrt{\sigma_1^2 + w^*w} \Rightarrow w = 0$$ (✗)

Nach Induktionsvoraussetzung $B = U_2 \Sigma_2 V_2^*$ $\Rightarrow$ $A = \underbrace{U_1 \begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix}}_{\text{unitär}} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & V_2 \end{bmatrix}^* V_1^*}_{\text{unitär}}$

Eindeutigkeit:

σ_1 eindeutig. Nimm an, es gebe neben v_1 einen linear unabh. Vektor w mit $\|w\|=1, \|Aw\|=\sigma_1$.

Sei $v_2 = \frac{w-(v_1^* w)v_1}{\|w-(v_1^* w)v_1\|}$, dann gilt auch $\|v_2\|=1$, $v_2 \perp v_1$ und $\|Av_2\| = \sigma_1$. In der Tat gilt

(i) $(Av_1, Av_2) = 0$: (*) & die Beliebigkeit von $v_2,\dots,v_n \Rightarrow 0 = (*)_n = u_1^* Av_2 = \frac{(Av_1)^*}{\sigma_1} Av_2$

(ii) $w = cv_1 + sv_2$ für $c,s \in \mathbb{R}$ mit $c^2+s^2=1$

und somit $\sigma_1^2 = \|Aw\|^2 = c^2 \underbrace{\|Av_1\|^2}_{=\sigma_1^2} + s^2 \|Av_2\|^2 \Rightarrow s^2\sigma_1^2 = s^2\|Av_2\|^2$

Sei nun $A = U\Sigma V^*$, und die Einträge $\sigma_1, \sigma_2, \dots$ von Σ seien paarweise verschieden.

Dann $\sigma_i = \|Av_i\| = \|\Sigma \underbrace{(V^* v_i)}_{=:w_i}\| \le \sigma_1$ mit Gleichheit $\Leftrightarrow w_i = \begin{pmatrix} \alpha_i \\ 0 \\ \vdots \end{pmatrix}$ für $|\alpha_i|=1$, jedoch $v_1 \perp v_2$ ↯ □

Thm: 1) $r = \mathrm{Rang}(A) \Leftrightarrow \sigma_1,\dots,\sigma_r \neq 0$ & $\sigma_s = 0 \ \forall s > r$

2) $\mathrm{Bild}(A) = \langle u_1,\dots,u_r\rangle$, $\mathrm{Kern}(A) = \langle v_{r+1},\dots,v_n\rangle$

3) $\|A\| = \sigma_1$, $\|A\|_F = \sqrt{\sigma_1^2+\dots+\sigma_r^2}$ für die Frobenius-Norm $\|A\|_F = \sqrt{\sum_{i,j} |A_{ij}|^2}$

4) die nichttrivialen Eigenwerte von A^*A und AA^* sind $\sigma_1^2,\dots,\sigma_r^2$

5) für $A^* = A$ sind die Singulärwerte die Beträge der Eigenwerte

6) $|\det A| = \prod_{i=1}^m \sigma_i$ für $A \in \mathbb{K}^{m\times m}$

7) $A = \sum_{i=1}^r \sigma_i (u_i v_i^*)$ = „Summe von r Rang-1-Matrizen"

8) $\|A - A_\nu\| = \inf_{B\in\mathbb{K}^{m\times n},\ \mathrm{Rang}\,B \le \nu} \|A-B\| = \sigma_{\nu+1}$ für $A_\nu = \sum_{i=1}^{\nu} \sigma_i (u_i v_i^*)$

9) $\|A - A_\nu\|_F = \inf_{B\in\mathbb{K}^{m\times n},\ \mathrm{Rang}\,B \le \nu} \|A-B\|_F = \sqrt{\sigma_{\nu+1}^2+\dots+\sigma_r^2}$

Bew: 1) U, V haben vollen Rang $\Rightarrow$ $\mathrm{Rang}\,A = \mathrm{Rang}\,\Sigma$ = # nichttriviale Einträge von Σ

2) folgt aus $\mathrm{Bild}\,\Sigma = \langle e_1,\dots,e_r\rangle \subset \mathbb{K}^m$, $\mathrm{Kern}\,\Sigma = \langle e_{r+1},\dots,e_n\rangle \subset \mathbb{K}^n$

3) $\|A\| = \|\Sigma\| = \max_i |\sigma_i| = \sigma_1$; $\|A\|_F = \sqrt{\mathrm{tr}\,A^*A} = \sqrt{\mathrm{tr}\,V\Sigma^*\Sigma V^*} = \sqrt{\mathrm{tr}\,V^*V\Sigma^*\Sigma} = \sqrt{\mathrm{tr}\,\Sigma^2}$

4) $A^*A = V\Sigma^*\Sigma V^* \Rightarrow A^*A$ ist ähnlich zu $\Sigma^*\Sigma$ mit Eigenwerten $\sigma_1^2,\dots,\sigma_n^2$; analog für AA^*

5) $\exists$ Q unitär, Λ Diagonalmatrix aus Eigenwerten mit $A = Q\Lambda Q^*$, $|\Lambda_{11}| \ge |\Lambda_{22}| \ge \dots$

$\Rightarrow A = Q|\Lambda| \underbrace{\mathrm{sign}(\Lambda)\, Q^*}_{\text{unitär}}$ ist ein SVD von A

6) $|\det A| = |\det U \det\Sigma \det V| = \underbrace{|\det U|}_{=1} |\det\Sigma|\, |\det V| = |\det\Sigma|$ da $1 = \det UU^* = \det U \det U^*$

7) $A = U(\Sigma_1 + \dots + \Sigma_r)V^* = u_1\sigma_1 v_1^* + \dots + u_r\sigma_r v_r^*$ für $\Sigma_i = \mathrm{diag}(0,\dots,0,\sigma_i,0,\dots,0)$

8) Sei $B \in \mathbb{K}^{m\times n}$ mit $\mathrm{Rang}\,B \le \nu$, $\|A-B\| < \sigma_{\nu+1}$.

$\mathrm{Kern}\,B$ ist ein $(n-\nu)$-dimensionaler Unterraum von $\mathbb{K}^n$ mit $\|Aw\| = \|(A-B)w\| < \sigma_{\nu+1}\|w\| \ \forall w \in \mathrm{Kern}\,B$

$W = \langle v_1,\dots,v_{\nu+1}\rangle$ ist ein $(\nu+1)$-dimensionaler Unterraum von $\mathbb{K}^n$ mit $\|Aw\| \ge \sigma_{\nu+1}\|w\| \ \forall w \in W$ ↯

9) ohne Beweis □

Ben: Angenommen, wir können die SVD berechnen, dann kann sie vielfältig numerisch eingesetzt werden, z.B. zum Finden des Rangs, einer Orthonormalbasis des Kerns, der Operatornorm, einer Approximation niedrigen Rangs, ...

```
A = rand(2,2);
[U,S,V] = svd(A); S, U'*U, V'*V, U*S*V'
angle = linspace(0,2*pi,100);
circle = [cos(angle);sin(angle)];
Acircle = A*circle;
subplot(1,2,1);
plot(circle(1,:),circle(2,:),'linewidth',3); axis equal; hold on;
quiver([0 0],[0 0],V(1,:),V(2,:),0,'linewidth',3);
subplot(1,2,2);
plot(Acircle(1,:),Acircle(2,:),'linewidth',3); axis equal; hold on;
quiver([0 0],[0 0],U(1,:)*S,U(2,:)*S,0,'linewidth',3);

img = double(imread('Schloss.png'));
imagesc(img); colormap(gray);
numel(img)
tic; [U,S,V] = svd(img); toc
l = 20; U_ = U(:,1:l); V_ = V(:,1:l); S_ = S(1:l,1:l);
numel(U_) + numel(V_) + numel(diag(S_))
imgCompressed = U_*S_*V_';
imagesc(imgCompressed);
```

Hendrik numerischer Fehler

Numerische Fehler & Stabilität

- Modellierungsfehler: Math. Modell beschreibt Realität nur näherungsweise
  Bsp.: Vernachlässigung der Reibung bei hängender Kette
- Diskretisierungsfehler: Numerische Approximation beschreibt nicht das kontinuierliche Modell mit so vielen Freiheitsgraden
  Bsp.: Approximation eines kontinuierlichen Bandes durch endlich viele „Kettenglieder"
- Messfehler: Eingangsdaten sind fehlerbehaftet
  Bsp.: Ungenau gemessene Position der Kettenaufhängung
- Rechenfehler: Entsteht durch Rundung & Abbruch iterativer Verfahren nach endlicher Zeit
  Bsp: Rundung beim Lösen des Gl.-systems & Abbruch der Fixpunktiteration

In der Vorlesung untersuchen wir Rechenfehler und die Auswirkung von Messfehlern.
Diskretisierungsfehler werden in „Numerischer Analysis" & „Numerik pDgl." behandelt.

Def: (absoluter & relativer Fehler) Sei $(V, \|\cdot\|)$ ein Vektorraum, x eine Näherung für $\hat{x}$, $\Delta x = x - \hat{x}$.

- absoluter Fehler $\|\Delta x\|$
- relativer Fehler $\frac{\|\Delta x\|}{\|\hat{x}\|}$ von x.

Bsp: Runden wir π auf 3,1 machen wir einen absoluten Fehler von 0,0415...
und einen relativen von $\frac{0{,}0415\ldots}{\pi} = 0{,}0132\ldots$

Computer speichern und verarbeiten Zahlen in einer endlichen Zahl an Bits

$\Rightarrow$ am Computer gibt es nur endlich viele Zahlen mit „Lücken" dazwischen

Def: (Gleitkommazahldarstellung) Es seien $b \geq 2$ eine Basis, $p \geq 1$ die Anzahl der Mantissenstellen, $r \geq 1$ die Anzahl der Exponentenstellen.

- Eine b-adische Zahl ist eine Folge $x = \pm(x_l x_{l-1} \dots x_0)$ mit $x_i \in \{0, 1, \dots, b-1\}$, ihr Wert ist $\pm \sum_{i=0}^{l} x_i b^i$.
- Menge der Exponenten $E = \{e \in \mathbb{Z} \mid -b^r < e < b^r\}$ = b-adische Zahlen der Länge r
- Menge der Mantissen $L = \{m \in \mathbb{Z} \mid -b^p < m < b^p\}$ = b-adische Zahlen der Länge p
- Menge der Maschinenzahlen $M = \{\pm m b^{-p \pm e} \mid m \in L, e \in E\}$
- normalisierte b-adische Darstellung: $\pm 0.m_1 m_2 \dots m_p b^{\pm e}$ mit $m_1 = 0$ (oder 0)
- Rundung: $rd: \mathbb{R} \to M$, $rd(x) = \operatorname{argmin}_{y \in M} |y - x|$ (nicht eindeutig)
- Maschinengenauigkeit $\varepsilon_m = b^{-p+1}/2 = \frac{1}{2} \frac{0.0\dots01}{0.1a\dots0}$ = maximaler relativer Rundungsfehler

Bem: An modernen Computern typischerweise $b = 2$, $(p, r) = \begin{cases} (23, 7) & \text{IEEE single precision} \\ (52, 10) & \text{IEEE double —"—} \end{cases}$

Def: (Maschinenoperation) Die Operationen $\circledast$ für $* \in \{+, -, \cdot, /\}$ sind definiert durch

$$\circledast: M \times M \to M, \quad m_1 \circledast m_2 = rd(m_1 * m_2)$$

Thm: Der relative Fehler der Maschinenoperationen ist $\left| \frac{m_1 \circledast m_2 - m_1 * m_2}{m_1 * m_2} \right| \leq \varepsilon_m$ für $m_1 * m_2 \neq 0$

Bem: Maschinenoperationen sind i.A. nicht assoziativ, z.B.

$$0 = (10^{-20} \oplus 10) \ominus 10 \neq 10^{-20} \oplus (10 \ominus 10) = 10^{-20}$$

Abstrakt kann man ein Rechenproblem auffassen als eine Funktion $f: X \to Y$ vom normierten Raum der Daten in den normierten Raum der Lösungen. Ein Paar (f, x) mit $x \in X$ wird auch Probleminstanz genannt.

Def: (Kondition) Ein Problem (x, f) heißt

- gut konditioniert, wenn kleine Perturbationen von x zu kleinen Änderungen der Lösung führen
- schlecht konditioniert sonst

Def: (Konditionszahl) Sei Δx eine kleine Perturbation von x und $\Delta f = f(x + \Delta x) - f(x)$.

Absolute Konditionszahl des Problems f an Stelle x $= \hat{\kappa} = \lim_{\delta \to 0} \sup_{\|\Delta x\| \leq \delta} \frac{\|\Delta f\|}{\|\Delta x\|}$

Relative Konditionszahl $= \kappa = \lim_{\delta \to 0} \sup_{\|\Delta x\| \leq \delta} \left(\frac{\|\Delta f\|}{\|f(x)\|} \Big/ \frac{\|\Delta x\|}{\|x\|} \right)$

Bem: Falls f differenzierbar, $\hat{\kappa} = \|Df(x)\|$, $\kappa = \frac{\|Df(x)\|}{\|f(x)\| / \|x\|}$

Bsp: $f: x \mapsto \sqrt{x} \quad \Rightarrow \kappa = \frac{|f'(x)|}{|f(x)|/|x|} = \frac{1}{2}\frac{1}{\sqrt{x}\sqrt{x}/x} = \frac{1}{2} \Rightarrow$ gut konditioniert

- $f: (\mathbb{R}^2, \|\cdot\|_\infty) \to \mathbb{R}, (x_1, x_2) \mapsto x_1 - x_2 \quad \Rightarrow Df(x) = (1 \;\; -1) \quad \Rightarrow \|Df(x)\|_\infty = 2$

$\Rightarrow \kappa = \frac{2}{|x_1 - x_2| / \max\{|x_1|, |x_2|\}} >> 1$ für $x_1 \approx x_2 \quad \Rightarrow$ schlecht konditioniert

Dieses Phänomen heißt „Auslöschung"

- Nullstellenbestimmung von Polynomen, gegeben die Monomkoeffizienten:

z.B. $(x - 1 - \Delta f)(x - 1 + \Delta f) = x^2 - 2x + 1 - \underbrace{\Delta f^2}_{\Delta x}$ $\Rightarrow \kappa = \sup_{\Delta x} \frac{|\Delta f|}{|\Delta x|}\frac{1}{1} = \sup_{\Delta x} \frac{1}{\sqrt{\Delta x}} = \infty$

```
% machine epsilon
eps, (1+eps)-1, (1+eps/2)-1

% a problem
f = @(x1,x2) x1-x2;
x1 = 1; x2 = 1+eps;
% bad conditioning due to extinction
fexact = f(x1,x2)
fpert = f(x1+eps,x2)
abs(fexact-fpert)/abs(fexact)/(eps/abs(x1))
```

Def: (Konditionszahl einer Matrix) Sei $A \in \mathbb{C}^{m \times m}$. Die Konditionszahl von A bzgl. $\|\cdot\|$ ist

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

Bem: Typischerweise betrachten wir $\|\cdot\| = \|\cdot\|_2$, dann $\kappa(A) = \frac{\sigma_1}{\sigma_m}$ für die Singulärwerte.

$\Rightarrow \kappa(A)$ = Exzentrizität der „Hyperellipse".

Thm: Sei $A \in \mathbb{C}^{m \times m}$ regulär.
- $f: \mathbb{C}^m \to \mathbb{C}^m, x \mapsto Ax$, hat Konditionszahl $\kappa \le \kappa(A)$
- $f: \mathbb{C}^m \to \mathbb{C}^m, b \mapsto$ Lsg. von $Ax = b$, hat Konditionszahl $\kappa \le \kappa(A)$

In beiden Fällen gibt es Daten, für die Gleichheit gilt.

Beweis:
- $\kappa = \sup_{\Delta x} \left(\frac{\|A(x+\Delta x) - Ax\|}{\|Ax\|} \Big/ \frac{\|\Delta x\|}{\|x\|} \right) = \sup_{\Delta x} \underbrace{\frac{\|A\Delta x\|}{\|\Delta x\|}}_{\le \|A\|} \cdot \underbrace{\frac{\|x\|}{\|Ax\|}}_{= \frac{\|A^{-1}b\|}{\|b\|} \le \|A^{-1}\| \text{ für } b = Ax}$

Gleichheit gilt für die x mit $\|A^{-1}(Ax)\| = \|A^{-1}\| \, \|Ax\|$.

- zweiter Fall ist $f: b \mapsto A^{-1}b \quad \Rightarrow$ analog ☐

Thm: Sei $b \in \mathbb{C}^m$, $f: \mathbb{C}^{m \times m} \to \mathbb{C}^m$, $A \mapsto A^{-1}b$. Dann ist $\kappa = \kappa(A)$.

Bew: Es ist $\underbrace{(A + \Delta A)(f + \Delta f)}_{b + \Delta A f + A \Delta f + \text{vernachlässigbarer Term}} = b \quad \Rightarrow \Delta f = -A^{-1}\Delta A f$

$\Rightarrow \frac{\|\Delta f\|}{\|\Delta A\|}\frac{\|A\|}{\|f\|} \le \|A\| \, \|A^{-1}\|$ mit Gleichheit für ΔA sodass $\|-A^{-1}\Delta A f\| = \|A^{-1}\| \, \|\Delta A\| \, \|f\|$

Ein solches ΔA existiert: Wähle z, w mit $\|z\| = 1$, $\|A^{-1}z\| = \|A^{-1}\|$ und $|f^* w| = \|f\| \, \|w\|$;

dann gilt für $\Delta A = z w^*$: $\|A^{-1}\Delta A f\| = \|A^{-1}\| \, \|w\|' \, \|f\|$ und

$\|\Delta A\| = \sup_{\|x\|=1} \|\Delta A x\| = \sup_{\|x\|=1} |w^* x| = \|w\|'$ ☐

```
% condition number of random matrix
n = 5;
A = rand(n,n);
cond(A)
[U,S,V] = svd(A); S(1,1)/S(n,n)
% of identity
A = eye(n,n);
cond(A)
% of Vandermonde matrix
A = vander(1:.5:3)
cond(A)
```

Vandermonde Matrix zu $x_0, \dots, x_n$:

$$V = \begin{pmatrix} x_0^0 & x_n^0 & \cdots & x_n^0 \\ x_0^1 & x_n^1 & \cdots & x_n^1 \\ \vdots & \vdots & & \vdots \\ x_0^n & x_n^n & \cdots & x_n^n \end{pmatrix}$$

$V^{-1} \begin{pmatrix} f_0 \\ \vdots \\ f_n \end{pmatrix}$ sind Monomkoeffizienten des $(x_0, f_0), \dots, (x_n, f_n)$ interpolierenden Polynoms

Heute: · Stabilität

Def: (Landau-Symbole) Seien $f, g : D \to \mathbb{R}$ für einen normierten Raum D (z.B. $D = \mathbb{R}$ oder $D = \mathbb{N}$).

- $f \in O(g)\ (x \to a) \quad :\Leftrightarrow \quad g \in \Omega(f)\ (x \to a) \quad :\Leftrightarrow \quad \lim_{x \to a} \frac{\|f(x)\|}{\|g(x)\|} < \infty$
- $f \in o(g)\ (x \to a) \quad :\Leftrightarrow \quad g \in \omega(f)\ (x \to a) \quad :\Leftrightarrow \quad \lim_{x \to a} \frac{\|f(x)\|}{\|g(x)\|} = 0$
- $f(x) \sim g(x) \quad :\Leftrightarrow \quad \exists\, c, C > 0 : c f(x) \le g(x) \le C f(x)$

Bem:

- a kann auch $\pm\infty$ sein
- häufig ist a implizit klar $\Rightarrow$ $(x \to a)$ wird weggelassen
- $O(g) \equiv$ Menge aller Funktionen f mit $\lim_{x \to a} \frac{\|f(x)\|}{\|g(x)\|} < \infty$
- oft schreibt man auch $f(x) = O(g(x))$ statt $f \in O(g)$

Ein Algorithmus zum Lösen eines Problems $f : X \to Y$ kann aufgefasst werden als eine Approximation $\tilde{f} : X \to Y$. I.A. $\tilde{f} \neq f$, da bereits die Rundung am Computer Fehler verursacht. Der relative Fehler kann aufgeteilt werden in einen unvermeidbaren Anteil und einen Beitrag durch den Algorithmus: Sei $\tilde{x}$ etwa Maschinengenauigkeit an x (z.B. $\tilde{x} = \mathrm{rd}\, x$; nur die Größenordnung soll uns interessieren)

$$\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} \le \frac{\|\tilde{f}(x) - f(\tilde{x})\|}{\|f(x)\|} + \frac{\|f(\tilde{x}) - f(x)\|}{\|f(x)\|} = \underbrace{\frac{\|\tilde{f}(x) - f(\tilde{x})\|}{\|f(\tilde{x})\|}}_{\text{durch Algorithmus bestimmt}} \underbrace{\frac{\|f(\tilde{x})\|}{\|f(x)\|}}_{\le 1 + \kappa \frac{\|\tilde{x} - x\|}{\|x\|}} + \kappa \underbrace{\frac{\|\tilde{x} - x\|}{\|x\|}}_{\le \varepsilon_m}$$

Def: (Stabilität) Ein Algorithmus $\tilde{f}$ für ein Problem f heißt (impliziter Limes $\varepsilon_m \to 0$)

- (vorwärts-) stabil, wenn für kleine Änderungen der Eingangsdaten die Änderung seines Ergebnisses die Größenordnung der unvermeidbaren Fehler nicht übersteigt.
  In dieser Vorlesung wollen wir fordern $\frac{\|\tilde{f}(x) - f(\tilde{x})\|}{\|f(\tilde{x})\|} = O(\varepsilon_m)$ für ein $\tilde{x}$ mit $\frac{\|\tilde{x} - x\|}{\|x\|} = O(\varepsilon_m)$
- rückwärts-stabil, wenn $\tilde{f}(x) = f(\tilde{x})$ für ein $\tilde{x}$ mit $\frac{\|\tilde{x} - x\|}{\|x\|} = O(\varepsilon_m)$

Bem: Rückwärtsstabil $\Rightarrow$ Vorwärtsstabil; Rückwärtsstabilität einfacher zu prüfen

Thm: Der Algorithmus $\tilde{f}(x_1, x_2) = rd(x_1) \ominus rd(x_2)$ zur Approximation von $f(x_1, x_2) = x_1 - x_2$ ist rückwärts-stabil.

Bew: · $rd(x_i) = x_i(1+\varepsilon_i)$ für $\varepsilon_i = O(\varepsilon_m)$, $i = 1, 2$

· $rd(x_1) \ominus rd(x_2) = [rd(x_1) - rd(x_2)](1+\varepsilon_3)$ für ein $\varepsilon_3 = O(\varepsilon_m)$

$\Rightarrow \tilde{f}(x_1, x_2) = \underbrace{x_1(1+\varepsilon_1)(1+\varepsilon_3)}_{\tilde{x}_1} - \underbrace{x_2(1+\varepsilon_2)(1+\varepsilon_3)}_{\tilde{x}_2} = f(\tilde{x}_1, \tilde{x}_2)$

für $\frac{|\tilde{x}_i - x_i|}{|x_i|} = \varepsilon_i + \varepsilon_3 + \varepsilon_i\varepsilon_3 = O(\varepsilon_m)$ □

Bsp: · $f: \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$, $f(x,y) = (x,y)$; $\tilde{f}(x,y) = \big((rdx_1 \otimes rdy_1) \oplus (rdx_2 \otimes rdy_2)\big) \oplus \dots (rdx_m \otimes rdy_m)\big)$ ist rückwärtsstabil

Beweis analog zu oben

· $f: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^{n\times n}$, $f(x,y) = x y^T$; $\tilde{f}(x,y) = \begin{pmatrix} rdx_1 \otimes rdy_1 & \dots & rdx_1 \otimes rdy_n \\ \vdots & & \vdots \\ rdx_n \otimes rdy_1 & \dots & rdx_n \otimes rdy_n \end{pmatrix}$

ist stabil, aber nicht rückwärts-stabil

da $\tilde{f}(x,y)$ i.A. keine Rang-1-Matrix ist

Daumenregel: erhöht f die Dimension, ist Rückwärtsstabilität selten.

· $f: \mathbb{R} \to \mathbb{R}$, $f(x) = 1 + x$; $\tilde{f}(x) = 1 \oplus rdx$ ist stabil, aber nicht rückwärts-stabil

z.B. $x = \varepsilon_m/2 \Rightarrow \tilde{f}(x) = 1 = f(\tilde{x})$ mit $\tilde{x} = 0$, aber $\frac{|x - \tilde{x}|}{|x|} = 1$

· $f: \mathbb{R}^{m\times m} \to \mathbb{C}^m$, $f(A) =$ Eigenwerte von A;

$\tilde{f}$: 1. berechne Koeffizienten des charakteristischen Polynoms $p(t) = \det(tI - A)$

2. finde Nullstellen von p

ist instabil (im Gegensatz zu anderen Algorithmen)

Koeffizienten können mit relativem Fehler $O(\varepsilon_m)$ berechnet werden, aber Nullstellenbestimmung anhand der Koeffizienten ist schlecht konditioniert!

z.B. Nullstellen von $t^2 - 2t + 1$ und $t^2 - 2t + 1 + \varepsilon_m$ unterscheiden sich um $\sqrt{\varepsilon_m}$

$\Rightarrow \frac{|\tilde{f}(x) - f(x)|}{|f(x)|} \approx \sqrt{\varepsilon_m}$

```
A = [1+1e-15 0;0 1]; evals = [A(1,1);A(2,2)];
% eigenvalues (stable algorithm)
eig(A)-evals
% characteristic polynomial
p = poly(A)
roots(p)-evals
```

Thm: Sei $\tilde{f}$ ein rückwärts-stabiler Algorithmus für $f: X \to Y$, dann $\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = O(\kappa(x)\varepsilon_m)$ (rel. Fehler)

Bew: $\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = \frac{\|f(\tilde{x}) - f(x)\|}{\|f(x)\|} \leq (\kappa(x) + o(1)) \frac{\|\tilde{x} - x\|}{\|x\|} = O(\kappa(x)\varepsilon_m)$ □

für ein $\tilde{x}$ nah bei x

Bem: · Die Fehlerabschätzung aus Rückwärtsstabilität und Kondition heißt „Rückwärts-Fehleranalyse" und ist häufig einfacher als die „Vorwärts-Fehleranalyse", d.h. das Abschätzen und Summieren der einzelnen Rundungsfehler in jedem Schritt.

Heute: · Gauß-Elimination

Direkte Lösungsverfahren für lineare Gleichungssysteme

Vorwärts- & Rückwärts-Einsetzen

Sei $A \in \mathbb{R}^{m \times m}$ eine obere (untere) Dreiecksmatrix, $A = \left(\begin{smallmatrix} \ddots \\ & \ddots \end{smallmatrix}\right)$ $\left(A = \left(\begin{smallmatrix} \ddots & \\ \ddots & \ddots \end{smallmatrix}\right)\right)$.

Man kann $Ax = b$ lösen mittels Rückwärts- (Vorwärts-) Einsetzen.

Hier Vorwärts-Einsetzen (Rückwärts analog).

Alg: (Vorwärts-Einsetzen) input: $A = \begin{pmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ \vdots & \ddots & \ddots \\ a_{m1} & \cdots & a_{mm} \end{pmatrix}$, $b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$

$x_1 = b_1 / a_{11}$

$x_2 = (b_2 - a_{21} x_1) / a_{22}$

$x_3 = (b_3 - a_{31} x_1 - a_{32} x_2) / a_{33}$

$\vdots$

$x_m = (b_m - \sum_{i=1}^{m-1} a_{mi} x_i) / a_{mm}$

Def: (flop) Ein flop („floating point operation") ist eine Einheit zum Bestimmen der Komplexität eines Algorithmus. Jede elementare Rechenoperation auf einem Computer $(+, -, *, /, \sqrt{})$ zählt als ein flop.

Bem: Mit der Benutzung von flops macht man automatisch die Vereinfachung, dass jede elementare Operation gleich aufwändig ist und dass sie alle seriell ausgeführt werden (was bei modernen Computern nicht der Fall ist → genauere Analyse nötig).

Thm: Aufwand des Vorwärts-Einsetzens $\sim m^2$

Bew: m Schritte, im i-ten Schritt eine Division, $i-1$ Multiplikationen & Additionen

$$\Rightarrow \text{Aufwand} = \sum_{i=1}^{m} (1 + 2(i-1)) = m + 2 \sum_{j=0}^{m-1} j = m + (m-1)m = m^2$$ □

Thm: Vorwärtseinsetzen für $Lx = b$ ist rückwärts-stabil, d.h. die berechnete Lösung $\tilde{x}$ erfüllt $(L+\delta L)\tilde{x} = b$ für eine untere Δ-Matrix $\delta L \in \mathbb{R}^{m\times m}$ mit $\frac{\|\delta L\|}{\|L\|} = O(\varepsilon_m)$

Bew: Vollst. Induktion nach m.

Anfang: $\tilde{x}_1 = rd\, b_1 \oslash rd\, a_{11} = \frac{b_1(1+\varepsilon_1)}{a_{11}(1+\varepsilon_2)}(1+\varepsilon_3)$ für $\varepsilon_1, \varepsilon_2, \varepsilon_3 = O(\varepsilon_m)$

$= b_1 / \tilde{a}_{11}$ mit $\tilde{a}_{11} = a_{11}\frac{(1+\varepsilon_2)(1+\varepsilon_3)}{1+\varepsilon_1} = a_{11}(1+O(\varepsilon_m))$

Schritt: Sei $L' = \begin{pmatrix} l_{11} & & \\ \vdots & \ddots & \\ l_{m-1,1} & \cdots & l_{m-1,m-1} \end{pmatrix}$, $\delta L' \in \mathbb{R}^{(m-1)\times(m-1)}$ untere Δ-Matrix mit

$$(L'+\delta L')\begin{pmatrix}\tilde{x}_1\\ \vdots \\ \tilde{x}_{m-1}\end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_{m-1}\end{pmatrix} \quad \& \quad \frac{\|\delta L'\|}{\|L'\|} = O(\varepsilon_m)$$

$$\tilde{x}_m = \left(rd\, b_m \ominus rd\, l_{m1} \otimes \tilde{x}_1 \ominus rd\, l_{m2} \otimes \tilde{x}_2 \ominus \dots \right) \oslash rd\, l_{mm}$$

$$= \left[\left(b_m(1+\varepsilon_a) - l_{m1}(1+\varepsilon_b)\tilde{x}_1(1+\varepsilon_c)\right)(1+\varepsilon_d) - \dots\right) / \left[l_{mm}(1+\varepsilon')\right]\right](1+\varepsilon'')$$

Nach Ausmultiplizieren ergibt sich

$$\tilde{x}_m = \frac{b_m}{l_{mm}}(1+\varepsilon_0) - \tilde{x}_1\frac{l_{m1}}{l_{mm}}(1+\varepsilon_1) - \dots - \tilde{x}_{m-1}\frac{l_{m,m-1}}{l_{mm}}(1+\varepsilon_{m-1})$$

für $\varepsilon_0, \dots, \varepsilon_{m-1} = O(\varepsilon_m)$

wähle $\delta l_{mm} = l_{mm}\left(\frac{1}{1+\varepsilon_0} - 1\right) = l_{mm}\, O(\varepsilon_m)$

$\delta l_{mi} = l_{mi}\left(\frac{1+\varepsilon_i}{1+\varepsilon_0} - 1\right) = l_{mi}\, O(\varepsilon_m)$

dann ist $\left[L + \underbrace{\begin{pmatrix} \delta L' & \vdots \\ \delta l_{m1} & \cdots \delta l_{mm} \end{pmatrix}}_{=:\delta L}\right]\tilde{x} = b$ und $\frac{\|\delta L\|}{\|L\|} = O(\varepsilon_m)$ □

Bem: Der Beweis liefert sogar $\frac{|\delta L_{ij}|}{|L_{ij}|} = O(\varepsilon_m)$

LU-Zerlegung

Sei $A \in \mathbb{R}^{m\times m}$ regulär. Um $Ax = b$ zu lösen, können wir A zerlegen in eine untere und obere Dreiecksmatrix, $A = LU$, $\left(\square\right) = \left(\triangle\right)\left(\triangledown\right)$,

und dann $x = U^{-1}(L^{-1}b)$ per Vorwärts- und Rückwärts-Einsetzen berechnen.

Die Zerlegung geschieht mittels Gauß-Elimination.

Alg: (Gauß-Elimination) input: $A = A^1 \in \mathbb{R}^{m\times m}$, $B = B^1 = I \in \mathbb{R}^{m\times m}$, output: $U = A^m$, $L = B^m$

for Zeile $i = 1$ bis $m-1$

subtrahiere $l_{ji} = \frac{A^i_{ji}}{A^i_{ii}}$-faches der $i$ten Zeile von A^i bzw B^i von der $j$ten Zeile von A^i bzw B^i

($\Rightarrow$ alle Einträge unter A^i_{ii} werden zu 0)

nenne neue Matrizen A^{i+1}, B^{i+1}

end

Der i-te Schritt kann aufgefasst werden als Linksmultiplikation der u. Δ-Matrix $L_i :=$

i-te Zeile $\rightarrow \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & 1 & & \\ & -l_{i+1,i} & \ddots & \\ & \vdots & & \\ & -l_{m,i} & & 1 \end{pmatrix}$ mit $l_{ji} = \frac{a^i_{ji}}{a^i_{ii}}$

$$\Rightarrow U = A^m = \underbrace{L_{m-1} \cdots L_2 L_1}_{L^{-1}} A$$

$$\Rightarrow L = L_1^{-1} L_2^{-1} \cdots L_{m-1}^{-1} = \begin{pmatrix} 1 & & & \\ l_{2,1} & 1 & & \\ \vdots & l_{3,2} & \ddots & \\ l_{m,1} & l_{m,2} & \cdots & 1 \end{pmatrix} \quad \text{mit } L_i^{-1} = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & 1 & & \\ & l_{i+1,i} & \ddots & \\ & \vdots & & \\ & l_{m,i} & & 1 \end{pmatrix}$$

Bem: Statt B^m mit zu berechnen, werden einfach die l_{ij} gespeichert.

Bsp:
$$A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 3 & 3 \\ 8 & 7 & 9 \end{pmatrix} ; \quad L_1 A = \begin{pmatrix} 1 & & \\ -2 & 1 & \\ -4 & & 1 \end{pmatrix} A = \begin{pmatrix} 2 & 1 & 1 \\ & 1 & 1 \\ & 3 & 5 \end{pmatrix}$$

$$L_2 L_1 A = \begin{pmatrix} 1 & & \\ & 1 & \\ & -3 & 1 \end{pmatrix} L_1 A = \begin{pmatrix} 2 & 1 & 1 \\ & 1 & 1 \\ & & 2 \end{pmatrix} = U$$

$$L = L_1^{-1} L_2^{-1} = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ 4 & & 1 \end{pmatrix} \begin{pmatrix} 1 & & \\ & 1 & \\ & 3 & 1 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ 4 & 3 & 1 \end{pmatrix}$$

Thm: Aufwand der Gauß-Elimination $\sim \frac{2}{3} m^3$ flops

Bew: · $m-1$ Schritte

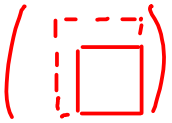
· im i-ten Schritt multipliziere eine Zeile à $m-i+1$ Einträgen mit $m-i$ Skalaren und addiere sie auf $m-i$ Zeilen à $m-i+1$ Einträgen

$$\Rightarrow \text{Aufwand} = \sum_{i=1}^{m-1} (m-i)(m-i+1) 2 \underset{j=m-i}{=} 2 \sum_{j=1}^{m-1} j(j+1) = 2 \sum_{j=1}^{m-1} j^2 + j$$

$$= 2 \left[\frac{(m-1)m(2m-1)}{6} + \frac{(m-1)m}{2} \right] \sim \frac{2}{3} m^3 \qquad \square$$

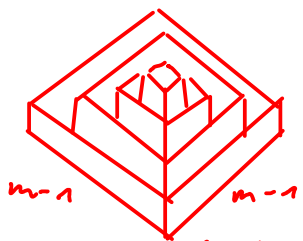
Graphische Visualisierung: In ersten Schritt wird auf allen unteren Matrixeinträgen

operiert:  $\Rightarrow (m-1)^2$ Einträge, 2 flops/Eintrag

2. Schritt:  $\Rightarrow (m-2)^2$ Einträge à 2 flops

$\vdots$

gesamt:



· 2 flops

Volumen = $\frac{1}{3}$ Würfelvolumen = $\frac{1}{3}(m-1)^3$

$\Rightarrow \sim \frac{2}{3} m^3$ flops

Heute: · Pivoting

Gauß-Elimination ist nicht stabil und kann sogar fehlschlagen. Bspw. bricht der Algorithmus für die Matrix $A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ im ersten Schritt zusammen, obwohl $\kappa(A) \approx 1$.

Für $A = \begin{pmatrix} 10^{-20} & 1 \\ 1 & 1 \end{pmatrix}$ funktioniert der Algorithmus wieder und liefert

$$L = \begin{pmatrix} 1 & 0 \\ \text{rd}\,10^{20} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix}, \quad \tilde{U} = \begin{pmatrix} \text{rd}\,10^{-20} & 1 \\ 0 & 1 \ominus \text{rd}\,10^{20} \end{pmatrix} = \begin{pmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{pmatrix}.$$

Vorwärts- und Rückwärts-Einsetzen für rechte Seite $b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ liefert nun $\tilde{x} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ statt der richtigen Lösung $x \approx \begin{pmatrix} -1 \\ 1 \end{pmatrix}$. $\Rightarrow \quad \frac{\|f(A) - f(A+\delta A)\|}{\|f(A+\delta A)\|} \approx \Omega(1) \quad \forall \delta A$ klein

Die Gauß-Elimination kann mittels „Pivotisierung" oder „Pivoting" stabilisiert werden. Statt im k-ten Schritt alle Einträge unter (k,k) zu eliminieren, können wir auch Einträge in anderen Zeilen und Spalten eliminieren:

$$\begin{pmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{pmatrix} \longrightarrow \begin{pmatrix} \times & \times & \times & \times \\ & \times & 0 & \times \\ & \times & \times & \times \\ & \times & 0 & \times \end{pmatrix}$$

Pivot-Element

Dies ist äquivalent dazu, die Spalten und Zeilen von A erst zu permutieren und dann Gauß-Elimination durchzuführen. Für gute Stabilität sollte man ein betragsmäßig möglichst großes Pivot-Element wählen. Würde man im k-ten Schritt aus allen $m-k$ noch nicht zur Elimination benutzten Zeilen (à $m-k$ Nicht-Null-Einträgen) das betragsmäßig größte Element heraussuchen (sog. vollständige Pivotisierung), würde dies $(m-k)^2$ Aufwand bedeuten, insgesamt also $\sum_{k=1}^{m-1} (m-k)^2 \sim \frac{m^3}{3}$. Stattdessen suche nur über Spalte:

Bem: k-ter Schritt äquivalent zu einer Permutation P_k und Multiplikation von L_k

$$\Rightarrow \quad L_{m-1} P_{m-1} \cdots L_2 P_2 L_1 P_1 A = U$$

Bsp: $A = \begin{pmatrix} 2 & 1 & 1 \\ 4 & 3 & 3 \\ 8 & 7 & 9 \end{pmatrix} \Rightarrow$ Pivot

$$P_1 A = \begin{pmatrix} & & 1 \\ & 1 & \\ 1 & & \end{pmatrix} A = \begin{pmatrix} 8 & 7 & 9 \\ 4 & 3 & 3 \\ 2 & 1 & 1 \end{pmatrix}$$

$$L_1 P_1 A = \begin{pmatrix} 1 & & \\ -\frac{4}{2} & 1 & \\ -\frac{1}{4} & & 1 \end{pmatrix} P_1 A = \begin{pmatrix} 8 & 7 & 9 \\ & -\frac{1}{2} & -\frac{3}{2} \\ & -\frac{3}{4} & -\frac{5}{4} \end{pmatrix}$$

$$P_2 L_1 P_1 A = \begin{pmatrix} 1 & & \\ & & 1 \\ & 1 & \end{pmatrix} L_1 P_1 A = \begin{pmatrix} 8 & 7 & 9 \\ & -3/4 & -5/4 \\ & -1/2 & -3/2 \end{pmatrix}$$

$$L_2 P_2 L_1 P_1 A = \begin{pmatrix} 1 & & \\ & 1 & \\ & -2/3 & 1 \end{pmatrix} P_2 L_1 P_1 A = \begin{pmatrix} 8 & 7 & 9 \\ & -3/4 & -5/4 \\ & & -3/6 \end{pmatrix}$$

Bem: $L_{m-1} P_{m-1} \cdots L_1 P_1 = (\overbrace{L'_{m-1} \cdots L'_1}^{L^{-1}})(\overbrace{P_{m-1} \cdots P_1}^{P})$ für

$L'_k = P_{m-1} \cdots P_{k+1} L_k P_{k+1}^{-1} \cdots P_{m-1}^{-1}$. Hier werden nur Zeilen $> k$ permutiert

$\Rightarrow L'_k$ ist immer noch untere Δ-Matrix

$\Rightarrow$ Verfahren liefert $L^{-1} P A = U$ bzw. $PA = LU$

Alg: (Gauß-El. mit partieller Pivotisierung) input: $U = A \in \mathbb{K}^{m \times m}$, $L = P = I$

for $k = 1$ bis $m-1$

wähle $i \geq k$ mit $|U_{ik}|$ maximal

vertausche Zeilen $i \& k$ in A, L und P

for $j = k+1$ bis m

$L_{jk} = U_{jk} / U_{kk}$

$U_{j,k:m} = U_{j,k:m} - L_{jk} U_{k,k:m}$

end

end

Bem: Vollständige Pivotisierung permutiert auch die Spalten, d.h. im k-ten Schritt werden an A eine Permutation P_k und untere Δ-Matrix L_k von links und eine Permutation Q_k von rechts multipliziert,

$$L_{m-1} P_{m-1} \cdots L_1 P_1 A \underbrace{Q_1 \cdots Q_{m-1}}_{Q} = U \quad \Leftrightarrow \quad PAQ = LU$$

Man kann zeigen:

Thm: Sei $A \in \mathbb{K}^{m \times m}$ regulär mit exakter LU-Zerlegung $A = LU$. Dann liefert die Gauß-Elimination ohne Pivotisierung Matrizen $\tilde{L}, \tilde{U}$ mit

$$\tilde{L}\tilde{U} = A + \delta A, \qquad \frac{\|\delta A\|}{\|L\| \|U\|} = O(\varepsilon_m).$$

Bem:
- Für (partielle oder vollständige) Pivotisierung denken wir uns im Theorem einfach A ersetzt durch die Permutation PAQ
- $\frac{\|\delta A\|}{\|A\|} = O(\varepsilon_m)$ würde Rückwärts-Stabilität bedeuten
- bei (partieller) Pivotisierung: Einträge von $\tilde{L}$ im Betrag ≤ 1, Diagonaleinträge $= 1$

$\Rightarrow \|L\| = O(1) \Rightarrow \frac{\|\delta A\|}{\|A\|} = O(\varepsilon_m) \frac{\|U\|}{\|A\|} = O(\rho \varepsilon_m)$ für $\rho = \frac{\max_{ij} |U_{ij}|}{\max_{ij} |A_{ij}|}$

Lemma: Für Gauß-Elimination mit partieller Pivotisierung ist $\rho = \frac{\max_{ij} |U_{ij}|}{\max_{ij} |A_{ij}|} \leq 2^{m-1}$.
Die Schranke ist scharf.

Bew: · Im k-ten Schritt sei $z_k = \max_{i,j \geq k} |U_{ij}|$.
Zur i-ten Zeile wird das $-L_{jk} \in [-1,1]$ - fache der k-ten addiert
$\Rightarrow$ anschließend kann sich der betragsmäßig maximale Eintrag der $(k:m, k:m)$-Matrix höchstens verdoppelt haben
$\Rightarrow z_{k+1} \leq 2 z_k$
$\Rightarrow$ am Ende $\max_{ij} |U_{ij}| \leq \max(z_1, \dots, z_m) \leq 2^{m-1} z_1 = 2^{m-1} \max_{ij} |A_{ij}|$

· Für $A = \begin{pmatrix} 1 & & & 1 \\ -1 & 1 & & 1 \\ -1 & -1 & 1 & \vdots \\ \vdots & \vdots & \ddots & 1 \\ -1 & -1 & \dots & -1 & 1 \end{pmatrix}$ ergibt sich $U = \begin{pmatrix} 1 & & & & 1 \\ & 1 & & & 2 \\ & & 1 & & 4 \\ & & & \ddots & \vdots \\ & & & & 2^{m-1} \end{pmatrix}$ □

Kor: Die Gauß-Elimination mit partieller Pivotisierung ist rückwärtsstabil.

Bem: Wir wissen $L\tilde{U} = A + \delta A$ mit $\frac{\|\delta A\|}{\|A\|} = O(2^m \varepsilon_m)$. Dies impliziert wegen $O(2^m \varepsilon_m) = O(\varepsilon_m)$ die Rückwärtsstabilität, die Konstante 2^m ist aber bei praktischen Anwendungen nicht vernachlässigbar! Bereits für $m = 160$ kann $\frac{\|\delta A\|}{\|A\|} \approx 2^{160} \varepsilon_m \approx 10^{16} \varepsilon_m \approx 1$ auf typischen Computern passieren. Die Matrizen, für die dies geschieht, sind jedoch statistisch so selten, dass partielle Pivotisierung in der Praxis sehr stabil ist.

```
m = 10;
A = rand(m,m);

% partial pivoting
[L,U,P] = lu(A); norm(L*U-P*A)
subplot(1,2,1); imagesc(L); colorbar;
subplot(1,2,2); imagesc(U); colorbar

[L2,U2] = lu(A); norm(U-U2), norm(P'*L-L2)

% total pivoting
[L,U,P,Q] = lu(A);
[L,U,P,Q] = lu(sparse(A)); norm(L*U-P*A*Q)

% stability for random matrices
dimensions = round(logspace(1,3,5))
for m = dimensions
  for i = 1:100
    A = randn(m,m);
    [L,U] = lu(A);
    loglog( m, max(abs(U(:)))/max(abs(A(:))), 'x' ); hold on;
  end
end
loglog(dimensions,sqrt(dimensions),'linewidth',3);
```

versuche auch A = rand(m,m)
dimensions.^(2/3)

Heute: · Strassen-Algorithmus & Cholesky-Zerlegung

Direkte Löser schneller als $O(m^3)$

Seien $A, B \in \mathbb{R}^{m\times m}$. Der Standard-Algorithmus zur Multiplikation der Matrizen berechnet $(AB)_{ij} = A_{i1}B_{1j} + \dots + A_{im}B_{mj}$ mit insgesamt $m^2(2m-1) \approx 2m^3$ flops. Lange Zeit dachte man, dies wäre optimal. 1969 zeigte Strassen, dass tatsächlich nur $O(m^{\log_2 7})$ Operationen nötig sind! Mittlerweile gibt es noch schnellere Algorithmen mit Exponent noch näher bei 2.

Setze $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$, $B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$, $C = AB = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$

$\Rightarrow C_{ij} = A_{i1}B_{1j} + A_{i2}B_{2j}$

$\Rightarrow$ insgesamt 4 Additionen und 8 Multiplikationen von Matrizen halber Größe

$\Rightarrow$ Aufwand$(m) = 4\left(\frac{m}{2}\right)^2 + 8\,\text{Aufwand}\left(\frac{m}{2}\right) = (2m-1)m^2$ flops (Induktion)

$\Rightarrow$ noch nichts gewonnen

Alg: (Strassen-Algorithmus)

$M_1 := (A_{11} + A_{22})(B_{11} + B_{22})$

$M_2 := (A_{12} + A_{22})B_{11}$

$M_3 := A_{11}(B_{12} - B_{22})$

$M_4 := A_{22}(B_{21} - B_{11})$

$M_5 := (A_{11} + A_{12})B_{22}$

$M_6 := (A_{21} - A_{11})(B_{11} + B_{12})$

$M_7 := (A_{12} - A_{22})(B_{21} + B_{22})$

$C_{11} := M_1 + M_4 - M_5 + M_7$

$C_{12} := M_3 + M_5$

$C_{21} := M_2 + M_4$

$C_{22} := M_1 - M_2 + M_3 + M_6$

Bem: Die Berechnung der Produkte $M_1, \dots, M_7$ geschieht rekursiv wieder mit dem selben Alg.

Thm: Aufwand des Strassen-Alg. $= O(m^{\log_2 7})$

Bew: · $\mu(m)$, $\alpha(m)$, $\beta(m)$ sei Anzahl an flop-Multiplikationen, -Additionen, Gesamt-flops (m: Matrixgröße)

· $\mu(1) = 1$, $\mu(2^k) = 7\mu(2^{k-1}) = \dots = 7^k \mu(1) = 7^k = m^{\log_2 7}$

· $\alpha(1) = 0$, $\alpha(2^k) = 7\alpha(2^{k-1}) + 18\alpha(2^{k-1})^2 = 6(7^k - 4^k)$ (vollst. Ind.)

· $\beta(m) = \beta(2^k) = 7^{k+1} - 6\cdot 4^k = O(m^{\log_2 7})$ □

Analog hat Strassen einen ebenso schnellen Algorithmus zur Matrix-Inversion gefunden.

Alg: (Strassen-Alg zur Inversion) input: $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$, output: $B = A^{-1} = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$

$M_1 = A_{11}^{-1}$ $\quad M_2 = A_{21} M_1$ $\quad M_3 = M_1 A_{12}$

$M_4 = A_{21} M_3$ $\quad M_5 = M_4 - A_{22}$ $\quad M_6 = M_5^{-1}$

$B_{12} = M_3 M_6$ $\quad B_{21} = M_6 M_2$ $\quad M_7 = M_3 B_{21}$

$B_{11} = M_1 - M_7$ $\quad B_{22} = -M_6$

Bem: · Die auftretenden Inversionen und Multiplikationen werden wieder mit dem Strassen-Alg. durchgeführt.

Thm: Aufwand (Strassen-Inversion) $= O(m^{\log_2 7})$

Bew: · Sei wieder $m = 2^k$, $b(m) =$ Aufwand

· $b(1) = 1$, $b(2^{k+1}) = 2\underbrace{b(2^k)}_{\text{Matrix-Inversion}} + \underbrace{6(7^{k+1} - 6 \cdot 4^k)}_{\text{-Multiplikation}} + \underbrace{2 \cdot (2^k)^2}_{\text{-Addition}}$

· $b(2^k) = \frac{42}{5} 7^k + \frac{48}{5} 2^k - 17 \cdot 4^k = O(7^k)$ per vollst. Induktion □

Bem: · Das Lösen von $Ax = b$ mit dem Strassen-Alg. kostet insgesamt $O(m^{\log_2 7})$ flops zur Berechnung von A^{-1} und $O(m^2)$ für $A^{-1}b$, also gesamt $O(m^{\log_2 7})$ flops.

· Der overhead bzw. die Konstanten im Strassen- und noch schnelleren Algorithmen sind so, dass sie erst für sehr große m effizienter als Gauß-Elimination werden. Daher sind sie in der Praxis kaum relevant.

In der Praxis treten oft Matrizen mit spezieller Struktur auf. Für diese lassen sich ebenfalls schnelle Lösungsalgorithmen finden. Ein Beispiel sind Tridiagonalmatrizen $\begin{pmatrix} \diagdown\!\diagdown\!\diagdown \end{pmatrix}$, die typischerweise nach der Diskretisierung elliptischer Dgl. in 1D auftreten. Die Gauß-Elimination für solche Matrizen ist auch als Thomas-Algorithmus bekannt.

Thm: Aufwand des Thomas-Alg. $= O(m)$

Cholesky-Zerlegung

Für $A \in \mathbb{C}^{m \times m}$ hermitesch positiv definit stellt sich heraus, dass Pivotisierung für die LU-Zerlegung nicht nötig ist. Dies ist von Vorteil, da durch Pivotisierung die Symmetrie gestört würde; durch die Symmetrie kann man bei der Berechnung im Vergleich zur Standard-LU-Zerlegung die Hälfte der Operationen einsparen. Tatsächlich liefert die LU-Zerlegung ohne Pivotisierung automatisch eine Zerlegung $A = LU$ mit $U = DL^*$ für eine positive

Diagonalmatrix D. Setzen wir $R = (L\sqrt{D})^*$, so erhalten wir die sog. Cholesky-Zerlegung

$$A = R^* R.$$

Idee für den Algorithmus:

$$A = \begin{pmatrix} A_{nn} & w^* \\ w & K \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{w}{A_{nn}} & I \end{pmatrix} \begin{pmatrix} A_{nn} & w^* \\ 0 & K - \frac{ww^*}{A_{nn}} \end{pmatrix}$$

↑ Gauß-Elimination

Um die Symmetrie zu erhalten, werden nun ganz analog Nullen in der ersten Zeile eingefügt:

$$A = \begin{pmatrix} 1 & 0 \\ \frac{w}{A_{nn}} & I \end{pmatrix} \begin{pmatrix} A_{nn} & 0 \\ 0 & K - \frac{ww^*}{A_{nn}} \end{pmatrix} \begin{pmatrix} 1 & \frac{w^*}{A_{nn}} \\ 0 & I \end{pmatrix}$$

Den Eintrag A_{nn} teilt man typischerweise noch auf: Mit $\alpha = \sqrt{A_{nn}}$ ist

$$A = \underbrace{\begin{pmatrix} \alpha & 0 \\ w/\alpha & I \end{pmatrix}}_{R_1^*} \begin{pmatrix} 1 & 0 \\ 0 & K - \frac{ww^*}{A_{nn}} \end{pmatrix} \underbrace{\begin{pmatrix} \alpha & w^*/\alpha \\ 0 & I \end{pmatrix}}_{R_1}$$

Nun wird analog mit der Elimination der 2. Zeile & Spalte und so weiter fortgefahren, was

$$A = \underbrace{R_1^* R_2^* \cdots R_m^*}_{R^*} \underbrace{R_m \cdots R_1}_{R}$$

ergibt.

Alg: (Cholesky-Zerlegung) input: $R = A$

for $k = 1$ bis m

for $j = k+1$ bis m

$R_{j,j:m} = R_{j,j:m} - R_{k,j:m}\, \overline{R_{kj}} \,/ R_{kk}$

$R_{k,k:m} = R_{k,k:m} / \sqrt{R_{kk}}$

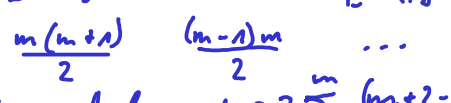
$R \leftarrow$ oberes Δ-Teil von R

Gauß-Elimination, jedoch nur Behandlung des oberen Dreiecks!

Thm: Aufwand Cholesky-Zerlegung $\sim \frac{m^3}{3}$ flops (halb so viel wie Gauß-Elimination)

Bew: bearbeitete Einträge (für jeden Eintrag 2 flops):

$k =$ 1 2 3 ...



$\frac{m(m+1)}{2}$ $\frac{(m-1)m}{2}$ ...

Gesamtvolumen der Pyramide mit Höhe m & Grundfläche

gesamt Aufwand $= 2 \sum_{k=1}^{m} \frac{(m+2-k)(m+1-k)}{2} \sim \sum_{k=1}^{m} k^2 \sim \frac{1}{3} m^3$ □

Bem: · $A \in \mathbb{C}^{m\times m}$ herm. pos. def. $\Rightarrow B^* A B$ herm. pos. def. für alle $B \in \mathbb{C}^{m\times n}$ mit vollem Rang, $n \le m$

Bew: $(B^*AB)^* = B^*A^*B = B^*AB$, $x^*(B^*AB)x = (Bx)^* A (Bx) > 0 \quad \forall x \neq 0$ □

· A herm. pos. def. $\Rightarrow A_{\ell:k,\ell:k} = B^*AB$ mit $B = [e_\ell | e_{\ell+1} | \dots | e_k]$ ist herm. pos. def.

Bem: Der Algorithmus funktioniert, solange im k-ten Schritt an der Stelle (k,k) eine positive Zahl steht, da $\sqrt{R_{kk}}$ berechnet werden muss. Dies ist der Fall, da im 1. Schritt A_{11} positiv ist (da A_{nn} pos. def.), im zweiten Schritt der $(1,1)$-Eintrag von $K - \frac{ww^*}{A_{11}}$, also $(R_1^{-*} A R_1^{-1})_{22}$ positiv ist (da $R_1^{-*} A R_1^{-1} =: \tilde{A}$ pos. def. und somit $\tilde{A}_{22}$ pos. def.) usw.

$\Rightarrow$ keine Pivotisierung nötig.

Thm: Für $A \in \mathbb{C}^{m\times m}$ herm. pos. def. existiert eine eindeutige Cholesky-Zerlegung $A = R^*R$ mit $R_{ii} > 0 \; \forall i$

Bew: · Existenz folgt aus Algorithmus

· Eindeutigkeit: Sei $A = R^*R = S^*S$ für $R \neq S$ obere Δ-Matrizen mit pos. Diagonalen

$$\Rightarrow \quad I = R^{-*} S^* \underbrace{S R^{-1}}_{T} = T^* T$$ für T obere Δ-Matrix mit pos. Diagonale

Da T unitär, d.h. Spalten orthonormal, muss $T = I$ gelten $\square$

Thm: Der Cholesky-Zerlegungs-Algorithmus mit Maschinenoperationen ist rückwärtsstabil, d.h. die Lösung R für gegebenes A erfüllt $R^*R = A + \delta A$ für ein δA mit $\frac{\|\delta A\|}{\|A\|} = O(\varepsilon_m)$.

```
m = 2000;
Z = randn(m,m); A = Z'*Z;
b = randn(m,1);
tic; x = A\b; toc
```
Cholesky

```
A2 = A; A2(m,1) = .1;
tic; x = A2\b; toc
```
Gauß-Elimination (2x so lang)

```
emin = min(eig(A));
A3 = A - .9*emin*eye(m,m);
tic; x = A3\b; toc
```
Cholesky

```
A4 = A - 1.1*emin*eye(m,m);
tic; x = A4\b; toc
```
Cholesky nicht möglich

```
A5 = triu(A);
tic; x = A5\b; toc
```
Rückwärtseinsetzen $\sim m^2$ statt $\frac{1}{3}m^3$

Heute: QR-Zerlegung

QR-Zerlegung

Def: (Projektion) · Eine Projektion ist ein $P \in \mathbb{K}^{m\times m}$ mit $P^2 = P$

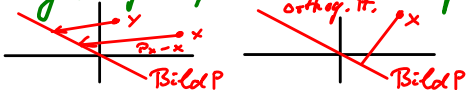
· $I - P$ erfüllt $(I-P)^2 = I - 2P + P^2 = I - P$ und heißt komplementäre Projektion.

· Eine orthogonale Projektion ist eine Projektion P mit $Px \perp (I-P)y \quad \forall x, y \in \mathbb{K}^m$.

Bem: · $P^2 = P$ bedeutet, dass ein projiziertes Element sich durch Projektion nicht mehr ändert, $P(Px) = Px$

· Px kann aufgefasst werden als der von Punkt x geworfene Schatten

· Der Schatten wird in Richtung $x - Px = (I-P)x$ geworfen, d.h. die komplementäre Projektion beschreibt die Schattenrichtung



Thm: · $\text{Bild}(I-P) = \text{Ker}(P)$ & $\text{Bild}(P) = \text{Ker}(I-P)$

· $\text{Bild}(P) \cap \text{Bild}(I-P) = \{0\}$

· $\mathbb{K}^m = \text{Bild}(P) \oplus \text{Bild}(I-P)$ (P zerteilt $\mathbb{K}^m$)

· Sei $\mathbb{K}^m = S_1 \oplus S_2$. $\exists!$ Projektion P mit $\text{Bild}\, P = S_1$, $\text{Ker}\, P = S_2$

· Projektion P ist orthogonale Projektion $\Leftrightarrow$ P ist hermitesch

Bew: · Sei $x \in \text{Ker}\, P$, dann ist $x = (I-P)x \in \text{Bild}(I-P)$.

Sei $y = (I-P)x \in \text{Bild}(I-P)$, dann $Py = Px - P^2x = 0$.

2. Teil folgt durch $\tilde{P} := I - P$

· Sei $x \in \text{Bild}\, P \cap \text{Bild}(I-P) = \text{Ker}\, P \cap \text{Ker}(I-P) \Rightarrow Px = 0$ & $(I-P)x = 0 \Rightarrow x = 0$

· $\mathbb{K}^m = \text{Bild}(P) + \text{Bild}(I-P)$, da $x = Px + (I-P)x \quad \forall x \in \mathbb{K}^m$

außerdem $\dim \text{Bild}\, P + \dim \text{Bild}(I-P) = \dim \text{Bild}\, P + \dim \text{Ker}\, P = m$

· Definiere $Px = s_1$, wobei $x = s_1 + s_2$ für $s_1 \in S_1$, $s_2 \in S_2$.

Eindeutigkeit: Sei B eine andere solche Projektion, dann ist $x = \underbrace{Bx}_{\in S_1} + \underbrace{(I-B)x}_{\in S_2} \Rightarrow Bx = s_1$

· „$\Leftarrow$": $((I-P)y, Px) = x^* P^* (I-P)y = x^* (P - P^2) y = 0$

„$\Rightarrow$": Sei $q_1, \dots, q_n \in \mathbb{K}^m$ Orthonormalbasis von $\text{Bild}\, P$,

$q_{n+1}, \dots, q_m$ Orthonormalbasis von $\text{Ker}\, P = \text{Bild}(I-P)$

$\Rightarrow q_1, \dots, q_m$ ist Orthonormalbasis von $\mathbb{K}^m$ mit $Pq_i = \begin{cases} q_i, & i \le n \\ 0, & i > n \end{cases}$

$\Rightarrow Q^* P Q = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & & 0 \\ & & & & \ddots \\ & & & & & 0 \end{pmatrix} = \Sigma \quad \Rightarrow P = Q \Sigma Q^* = P^*$ □

Bsp: · $P_a = \left(\frac{a}{\|a\|}\right)\left(\frac{a}{\|a\|}\right)^* = \frac{aa^*}{a^*a}$ ist orthogonale Projektion auf $\langle a \rangle$

· $P_{\perp a} = I - \frac{aa^*}{a^*a}$ ist orthogonale Projektion auf $\langle a \rangle^\perp$

· Seien $a_1, \dots, a_n \in \mathbb{K}^m$ linear unabhängig, $A = [a_1 | \dots | a_n]$, P die orthogonale Projektion auf $\langle a_1, \dots, a_n \rangle$. Für $y = Pv$ ist $\text{Bild}\, P = \text{Bild}\, A \perp (y - v)$, d.h. $a_i^* (y - v) = 0 \; \forall i$ oder $A^*(y - v) = 0$. Da $y \in \langle a_1, \dots, a_n \rangle$, $y = Ax$ für ein $x \in \mathbb{K}^n \Rightarrow A^*(Ax - v) = 0$

$\Leftrightarrow x = (A^*A)^{-1} A^* v$, d.h. $y = Ax = A(A^*A)^{-1} A^* v$

$\Rightarrow P = A(A^*A)^{-1} A^*$ (Verallgemeinerung von P_a)

· Für $A = Q$ unitär ergibt sich $P = QQ^*$

Def: (QR-Zerlegung) Sei $A \in \mathbb{K}^{m \times n}$.

· $A = QR$ für $Q \in \mathbb{K}^{m \times m}$ unitär, $R \in \mathbb{K}^{m \times n}$ obere Dreiecksmatrix heißt QR-Zerlegung von A.

· Sei $n < m$, $\hat{Q} = Q_{:,1:n}$, $\hat{R} = R_{1:n,:}$. $A = \hat{Q}\hat{R}$ heißt reduzierte QR-Zerlegung von A.

Bem: Sei $A = [a_1 | \dots | a_n]$, $\hat{Q} = [q_1 | \dots | q_n]$.

Offenbar ist $a_1 = R_{11} q_1$ bzw. $\langle a_1 \rangle = \langle q_1 \rangle$, $\langle a_1, a_2 \rangle = \langle q_1, q_2 \rangle$, ..., d.h. die ersten j Spalten von Q bilden eine Orthonormalbasis des Spaltenraumes $\langle a_1, \dots, a_j \rangle$.

Wir können $q_1, q_2, \dots$ schrittweise berechnen: Im j-ten Schritt suchen wir $q_j \in \langle a_1, \dots, a_j \rangle$ mit $q_j \perp q_1, \dots, q_{j-1}$. Hierzu projizieren wir a_j auf $\langle q_1, \dots, q_{j-1} \rangle^\perp$, d.h.

$$v_j = \underbrace{(I - Q_{j-1} Q_{j-1}^*) a_j}_{= a_j - (q_1^* a_j) q_1 - (q_2^* a_j) q_2 - \dots - (q_{j-1}^* a_j) q_{j-1}} \quad \text{für } Q_{j-1} = [q_1 | \dots | q_{j-1}]; \quad q_j = \frac{v_j}{\|v_j\|} \qquad (*)$$

Durch Vergleich mit $a_j = R_{1j} q_1 + R_{2j} q_2 + \dots + R_{jj} q_j$ ergibt sich

$$R_{ij} = q_i^* a_j, \quad |R_{jj}| = \| a_j - \sum_{i=1}^{j-1} R_{ij} q_i \|$$

Alg. (Gram-Schmidt-Verfahren, instabil) input: $A \in \mathbb{K}^{m \times n}$, output: $\hat{Q}, \hat{R}$

```
for j = 1 bis n
    v_j = a_j
    for i = 1 bis j-1
        R_ij = q_i^* a_j
        v_j = v_j - R_ij q_i
    end
    R_jj = ||v_j||
    q_j = v_j / R_jj
end
```

Thm: Jedes $A \in \mathbb{K}^{m \times n}$ besitzt eine (reduzierte) QR-Zerlegung.

Bew: Gram-Schmidt-Verfahren bricht nur zusammen, wenn R_{jj} null wird und $q_j = v_j / R_{jj}$ berechnet wird. Wenn wir in diesem Schritt jedoch stattdessen ein beliebiges $q_j \perp \langle q_1, \dots, q_{j-1} \rangle$ mit $\|q_j\| = 1$ wählen und dann fortfahren, liefert das Verfahren dennoch eine reduzierte QR-Zerlegung. Ergänzen wir $q_1, \dots, q_n$ zu einer Orthonormalbasis $q_1, \dots, q_m$ des $\mathbb{K}^m$ und setzen $Q = [q_1 | \dots | q_m]$, $R = \begin{pmatrix} \hat{R} \\ 0 \end{pmatrix}$, erhalten wir eine QR-Zerlegung. □

Thm: Jedes $A \in \mathbb{K}^{m \times n}$ mit vollem Rang, $m \geq n$, hat eine eindeutige reduzierte QR-Zerlegung $A = \hat{Q} \hat{R}$ mit $\hat{R}_{ii} > 0$

Bem: „$\hat{R}_{ii} > 0$" ist nötig, da Multiplikation der i-ten Spalte von $\hat{Q}$ und Zeile von $\hat{R}$ mit $z \in \mathbb{K}$, $|z| = 1$, eine alternative reduzierte QR-Zerlegung liefert.

Bew: Folgt aus Eindeutigkeit von $(*)$. □

Heute: · modifiziertes Gram-Schmidt-Verfahren

· Householder-Verfahren

Das Gram-Schmidt-Verfahren berechnet im j-ten Schritt $q_j = \frac{P_j a_j}{\|P_j a_j\|}$ mit $P_j = I - Q_{j-1} Q_{j-1}^*$ für $Q_{j-1} = [q_1 | \dots | q_{j-1}]$. Leider ist dies instabil. Jedoch ist die Projektion P_j mit Rang $m-(j-1)$ gleich $j-1$ Projektionen mit Rang $m-1$,

$$P_j = P_{\perp q_{j-1}} P_{\perp q_{j-2}} \cdots P_{\perp q_1} \quad \text{für} \quad P_{\perp q} = I - qq^*.$$

Der modifizierte Gram-Schmidt-Algorithmus berechnet erst $v_j^{(1)} = a_j$, $v_j^{(2)} = P_{\perp q_1} v_j^{(1)}, \dots$, $v_j^{(j)} = P_{\perp q_{j-1}} v_j^{(j-1)}$, $q_j = \frac{v_j^{(j)}}{\|v_j^{(j)}\|}$, was stabil ist.

Alg. (modifiziertes Gram-Schmidt-Verfahren) input: $V = [v_1 | v_2 | \dots | v_n] = A \in \mathbb{K}^{m \times n}$, output: $\hat{Q}, \hat{R}$

for $i = 1$ bis n

$$\hat{R}_{ii} = \|v_i\|$$

$$q_i = v_i / \hat{R}_{ii}$$

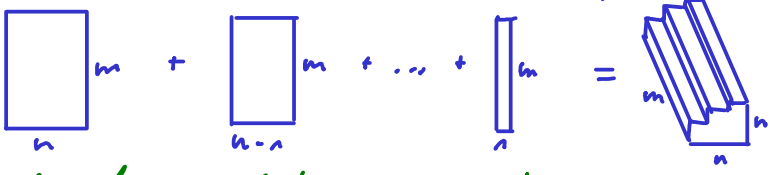
$$\hat{R}_{i,i+1:n} = q_i^* V_{:,i+1:n} \qquad (*)$$

$$V_{:,i+1:n} = V_{:,i+1:n} - q_i (\hat{R}_{i,i+1:n}) \qquad (**)$$

end

Thm: Aufwand modifiziertes Gram-Schmidt-Verfahren $\sim 2mn^2$ flops

Bew: In Schritt i wird in $(*)$ und $(**)$ auf $m(n-i)$ Elementen operiert (je 2 flops)

$\Rightarrow$ 

$\sim \frac{mn^2}{2}$ Einträge à 2·2 flops $\square$

Bem: $Ax = b$ kann gelöst werden durch

1. QR-Zerlegung $A = QR$
2. $y = Q^* b$
3. Rückwärtseinsetzen $x = R^{-1} y$

Das Householder-Verfahren triangulisiert A durch Linksmultiplikation unitärer Matrizen,

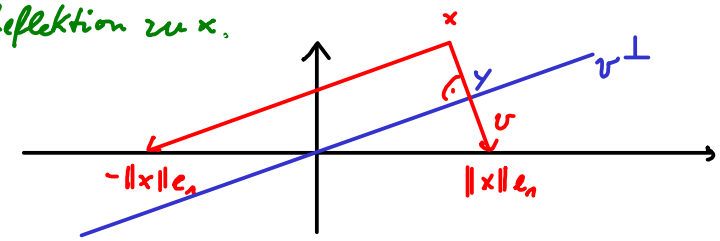
$$\underbrace{Q_n \cdots Q_2 Q_1}_{= Q^*} A = R$$

(während Gram-Schmidt aufgefasst werden kann als $A \underbrace{R_1^{-1} \cdots R_n^{-1}}_{\hat{R}^{-1}} = \hat{Q}$). Dabei führen $Q_1, Q_2, \dots$ sukzessive Nullen in den Spalten von A ein

$$\begin{pmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{pmatrix} \xrightarrow{Q_1} \begin{pmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{pmatrix} \xrightarrow{Q_2} \begin{pmatrix} \times & \times & \times \\ & \times & \times \\ & 0 & \times \\ & 0 & \times \end{pmatrix} \xrightarrow{Q_3} \begin{pmatrix} \times & \times & \times \\ & \times & \times \\ & & \times \\ & & 0 \end{pmatrix}$$

A $\qquad$ R

Def: (Householder-Reflektion) Sei $x \in \mathbb{K}^m$. $F_x = I - 2\frac{vv^*}{v^*v}$ für $v = 2\|x\|e_1 - x$ mit $\|e_1\|=1$ heißt Householder-Reflektion zu x.



Projektion von x auf $v^\perp$: $y = P_{\perp v}\, x = \left(I - \frac{vv^*}{v^*v}\right)x$

$F_x\, x = P_{\perp v}\, x + \left(P_{\perp v}\, x - x\right) = \left(I - 2\frac{vv^*}{v^*v}\right)x$

Typischerweise wählt man $v = \underbrace{-\operatorname{sign}(x_1)\|x\|e_1}_{F_x x} - x$, damit x möglichst weit von $F_x x$ weg ist und somit bei der Berechnung von v keine Auslöschung entsteht.

Bem: $F_x^* F_x = I - 4\frac{vv^*}{v^*v} + 4\frac{vv^*}{v^*v}\frac{vv^*}{v^*v} = I$, d.h. F_x ist unitär und hermitesch.

Im $i$ten Schritt des Householder-Verfahrens wird $Q_{i-1}\cdots Q_1 A = \begin{pmatrix} x & x & \cdots & x \\ & x & \cdots & x \\ & & \ddots & \vdots \\ & & x & \cdots & x \end{pmatrix}$ von links mit $Q_i = \begin{pmatrix} I & \\ & F_x \end{pmatrix}$ multipliziert. ($x \in \mathbb{K}^{m-i}$)

Alg: (Householder-Verfahren) input: $A \in \mathbb{K}^{m\times n}$, output: $R = A$

for $k = 1$ bis n

$x = R_{k:m,k}$

$v_k = -\operatorname{sign}(x_1)\|x\|e_1 - x$

$v_k = v_k / \|v_k\|$

$R_{k:m,k:n} = R_{k:m,k:n} - 2v_k\,(\underbrace{v_k^* R_{k:m,k:n}}_{①})$ (② = $R_{k:m,k:n}$, ③ = $2v_k$)

end

Bem:
- Obiger Algorithmus berechnet Q nicht explizit, da es oft nicht benötigt wird.
- Möchte man $Ax = b$ mit QR-Zerlegung lösen, muss man $y = Q^* b$ berechnen. Hierzu kann $y = b$ und im $k$ten Schritt $y_{k:m} = y_{k:m} - 2v_k(v_k^* y_{k:m})$ gesetzt werden. (*)
- Möchte man Q haben, berechnet man $Q^*e_1, \dots, Q^*e_m$ wie eben und setzt $Q = [Q^*e_1 | \dots | Q^*e_m]^*$

Thm: Aufwand $\sim 2mn^2 - \frac{2}{3}n^3$ flops

Bew: $\sum_{k=1}^{n} \underbrace{\left(2(m-k+1)(n-k+1)\right.}_{①} + \underbrace{\left.2(m-k+1)(n-k+1)\right)}_{②③} = 4\sum_{k=1}^{n}(m-k+1)(n-k+1) \overset{\ell = n-k+1}{=} 4\sum_{\ell=0}^{n-1}(m-n+\ell)\ell \sim 2mn^2 - \frac{2}{3}n^3$ □

Thm: Das Householder-Verfahren mit Maschinenoperationen ist rückwärtsstabil in folgendem Sinn: Für $A \in \mathbb{K}^{m\times n}$ seien R die erhaltene Δ-Matrix und $\hat{v}_1 = (v_1), \hat{v}_2 = \begin{pmatrix} 0 \\ v_2 \end{pmatrix}, \dots, \hat{v}_n = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ v_n \end{pmatrix} \in \mathbb{K}^m$ die erhaltenen Vektoren, sodass $Q = (I - 2\hat{v}_n\hat{v}_n^*)\cdots(I - 2\hat{v}_1\hat{v}_1^*)$. Dann ist $QR = A + \delta A$ für ein δA mit $\frac{\|\delta A\|}{\|A\|} = O(\varepsilon_M)$.

Bsp: $A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}$, $v_1 = -\|x\| e_1 - x = -14 e_1 - x = \begin{pmatrix} -26 \\ -6 \\ 4 \end{pmatrix}$, $v_1^* v_1 = 728$, $Q_1 = \left(I - 2\frac{v_1 v_1^*}{v_1^* v_1}\right)$

$Q_1 A = \begin{pmatrix} -14 & & \\ 0 & & \\ 0 & & \end{pmatrix} \left| \begin{matrix} a_2 - 2\frac{(v_1^* a_2) v_1}{v_1^* v_1} \end{matrix} \right| \left(a_3 - 2\frac{(v_1^* a_3) v_1}{v_1^* v_1} \right) = \left(\begin{matrix} -14 \\ 0 \\ 0 \end{matrix} \left| \begin{matrix} -21 \\ 2261/13 \\ 252/13 \end{matrix} \right| \begin{matrix} 14 \\ -854/13 \\ -553/131 \end{matrix} \right)$

x z

$v_2 = -\|x\| e_1 - x = -175 e_1 - x = \frac{-1}{13}\begin{pmatrix} 4536 \\ 252 \end{pmatrix}$, $Q_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & & \\ 0 & \left(I - 2\frac{v_2 v_2^*}{v_2^* v_2}\right) & \end{pmatrix}$

$\underbrace{Q_2 Q_1}_{Q^*} A = \left(\begin{matrix} -14 \\ 0 \\ 0 \end{matrix} \left| \begin{matrix} -21 \\ -175 \\ 0 \end{matrix} \right| \begin{matrix} 14 \\ z - 2\frac{(v_2^* z) v_2}{v_2^* v_2} \end{matrix} \right) = \begin{pmatrix} -14 & -21 & 14 \\ & -175 & 70 \\ & & -35 \end{pmatrix} = R$

```
m = 80;
[U,~,V] = svd(randn(m));
S = diag(2.^(-1:-1:-m));
A = U*S*V;

[Q,R] = gs(A);
[Qm,Rm] = mgs(A);
[Qh,Rh] = qr(A);
semilogy(1:m,abs(diag(R)),'o','Markersize',20,'Linewidth',5); hold on;
semilogy(1:m,abs(diag(Rm)),'v','Markersize',20,'Linewidth',5);
semilogy(1:m,abs(diag(Rh)),'x','Markersize',20,'Linewidth',5); hold off;

A = [.7      .707;
     .70001 .707];
[Q,R] = qr(A); norm(Q'*Q-eye(2))
[Q,R] = mgs(A); norm(Q'*Q-eye(2))

R = triu(randn(m));
[Q,~] = qr(randn(m));
A = Q*R;
[Q2,R2] = qr(A);
norm(Q2-Q)
norm(R2-R)
norm(A-Q2*R2)/norm(A)
Q3 = Q+1e-4*randn(m);
R3 = R+1e-4*randn(m);
norm(A-Q3*R3)/norm(A)
```

Gram-Schmidt
modifizierter Gram-Schmidt
Householder

Orthogonalität instabil

} schlechte Kondition oder instabiler Algorithmus ?
⇒ rückwärtsstabil

Alg: (löse $Ax=b$ mittels QR-Zerlegung) input: $A \in \mathbb{K}^{m\times m}$, $b \in \mathbb{K}^m$ output: $x \in \mathbb{K}^m$

berechne R und $y = Q^* b$ für QR-Zerlegung von A per Householder-Verfahren

löse $Rx = y$ per Rückwärts-Einsetzen

Kor: Obiger Algorithmus ist rückwärts-stabil, d.h. das Ergebnis x erfüllt $(A+\Delta A)x = b$ für ein ΔA mit $\frac{\|\Delta A\|}{\|A\|} = O(\varepsilon_m)$

Bew: · Berechnung von $y = Q^* b$ mittels (*) ist rückwärts-stabil (Hausaufgabe), d.h. $(Q+\delta Q)y = b$ für ein δQ mit $\frac{\|\delta Q\|}{\|Q\|} = O(\varepsilon_m)$

· Rückwärtseinsetzen ist rückwärts-stabil $\Rightarrow (R+\delta R)x = y$ für ein δR mit $\frac{\|\delta R\|}{\|R\|} = O(\varepsilon_m)$

· $b = (Q+\delta Q)(R+\delta R)x = \underbrace{(QR + \delta QR + Q\delta R + \delta Q \delta R)}_{= A+\delta A \text{ für } \delta A \text{ mit } \|\delta A\|/\|A\| = O(\varepsilon_m)} x = (A+\Delta A)x$

· $\frac{\|R\|}{\|A\|} = \frac{\|Q^*(A+\delta A)\|}{\|A\|} \le \underbrace{\|Q^*\|}_{O(1)} \underbrace{\frac{\|A\| + \|\delta A\|}{\|A\|}}_{= 1 + O(\varepsilon_m)} = O(1)$

$$\left.\begin{aligned} \frac{\|\delta Q R\|}{\|A\|} &\le \underbrace{\frac{\|\delta Q\|}{\|Q\|}}_{O(\varepsilon_m)} \underbrace{\|Q\|}_{O(1)} \underbrace{\frac{\|R\|}{\|A\|}}_{O(1)} = O(\varepsilon_m) \\ \frac{\|Q \delta R\|}{\|A\|} &\le \|Q\| \frac{\|\delta R\|}{\|R\|} \frac{\|R\|}{\|A\|} = O(\varepsilon_m) \\ \frac{\|\delta Q \delta R\|}{\|A\|} &\le \frac{\|\delta Q\|}{\|Q\|} \|Q\| \frac{\|\delta R\|}{\|R\|} \frac{\|R\|}{\|A\|} = O(\varepsilon_m^2) \end{aligned}\right\} \Rightarrow \frac{\|\Delta A\|}{\|A\|} = \frac{\|\delta A + \delta QR + Q\delta R + \delta Q \delta R\|}{\|A\|} = O(\varepsilon_m)$$

□

Heute: · dünn besetzte Matrizen & Graphen

Dünn besetzte Matrizen

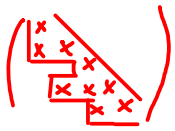
Viele Anwendungsprobleme resultieren in dünn besetzten („sparse") Matrizen, d.h. Matrizen, deren Einträge größtenteils 0 sind. Beispielsweise bei der Diskretisierung von pDgln mit n Freiheitsgraden pro Dimension (d.h. die Lösung $u: [0,1]^n \to \mathbb{R}$ der pDgl. wird approximiert durch ihre Werte an den n^n Stellen $(x_{i_1}^{i_1}, \dots, x_{i_n}^{i_n})$ mit $x_j^{i_j} = \frac{i_j}{n}$) ergeben sich oft Matrizen $A \in \mathbb{R}^{n^n \times n^n}$ mit nur $2n+1$ Einträgen pro Zeile. Wäre für $n=2$, $m=1000$ die Matrix voll besetzt, hätte sie $(n^n)^2 = 10^{12}$ Einträge à 8 Byte (double), d.h. 8 TB!

Man kann dünn besetzte Matrizen auf verschiedene Arten speichern; hiervon hängt u.A. die Effizienz von Matrixoperationen ab.

Def: · Eine symmetrische **Bandmatrix** der **Bandbreite** w ist ein symmetrisches $A \in \mathbb{R}^{m\times m}$ mit $A_{ij} = 0$ für $|i-j| \geq w$. Sie wird oft gespeichert als Matrix $B \in \mathbb{R}^{m\times w}$ mit $B_{ij} = A_{i,1+j-i}$

$A = \left(\begin{array}{c} w \\ \square \end{array}\right) \rightsquigarrow B = \left(\square\right)$

· Die **Hülle** einer symmetrischen Matrix $A \in \mathbb{R}^{m\times m}$ ist $Env(A) = \{(i,j) \mid l_i(A) \leq j \leq i,\ i = 1,\dots,m\}$ für $l_i(A) = \min_{1\leq j\leq i} \{j \mid A_{ij} \neq 0\}$.



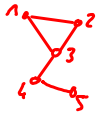
· Eine **Tripletmatrix** ist ein Tripel $(R, C, V) \in \{1,\dots,m\}^k \times \{1,\dots,m\}^k \times \mathbb{R}^k$ und repräsentiert eine Matrix $A \in \mathbb{R}^{m\times m}$, die man aus $0 \in \mathbb{R}^{m\times m}$ erhält, indem man für $i = 1,\dots,k$ an der Stelle (R_i, C_i) den Wert V_i addiert.

$\begin{pmatrix} a & & \\ & b & c \\ & & d \end{pmatrix} \triangleq \begin{array}{l} R = (1\ \ 2\ \ 2\ \ 3) \\ C = (1\ \ 2\ \ 3\ \ 2) \\ V = (a\ \ b\ \ c\ \ d) \end{array}$

· Eine Matrix in CRS (Compressed row storage)-Format ist $(P, C, V) \in \{1,\dots,k\}^m \times \{1,\dots,m\}^k \times \mathbb{R}^k$. $A \in \mathbb{R}^{m\times m}$ wird hier wie folgt repräsentiert: V enthält die Nicht-Null-Einträge zeilenweise aufgelistet, C enthält die entsprechenden Spaltenindizes, P_i ist der Index des ersten Eintrags in V aus Zeile i.

$\begin{pmatrix} a & & \\ & b & c \\ & & d \end{pmatrix} \triangleq \begin{array}{l} P = (1\ \ 2\ \ 4) \\ C = (1\ \ 2\ \ 3\ \ 2) \\ V = (a\ \ b\ \ c\ \ d) \end{array}$

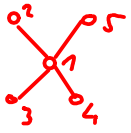
Def: (Graph) · $G = (X, E)$ heißt **Graph** mit Knotenmenge $X = \{1,\dots,m\}$ und Kantenmenge $E \subset X^2$, wobei $(a,b) \in E \Rightarrow (b,a) \in E$. Wir wollen außerdem $(i,i) \in E\ \forall i$ annehmen.



· **Adjazenzmenge** von $Y \subset X$ = $Adj(Y) = \{a \in X \mid (x,a) \in E \text{ für ein } x \in Y\}$

· **Grad** von $x \in X$ = $Grad(x) = |Adj(\{x\}) \setminus \{x\}|$

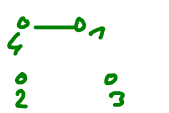
Das Besetzungsmuster einer symmetrischen Matrix A kann man als Graph (X, E) beschreiben. Die Knoten entsprechen den Zeilen, und $(i,j) \in E$ wenn $A_{ij} \neq 0$.



$\triangleq \begin{pmatrix} x & x & x & x & x \\ x & x & & & \\ x & & x & x & \\ x & & & & x \\ x & & & & \end{pmatrix}$

Die zu einem Graphen gehörende Matrix kann bei ungeschickter Knotennumerierung eine große Hülle / Bandbreite haben.

Bsp:



$\triangleq \begin{pmatrix} x & x & & x \\ x & & x & \\ & x & x & \\ x & & & x \end{pmatrix}$

$\triangleq \begin{pmatrix} x & x & & \\ x & x & & \\ & & x & x \\ & & x & x \end{pmatrix}$

Eine Umnumerierung entspricht einer Permutation von Zeilen und Spalten. Die optimale Umnumerierung ist NP-hart, es gibt aber Algorithmen, die eine gute Numerierung liefern:

Alg: (Cuthill-McKee-Algorithmus)

1) Wähle Startknoten x, der neue Nummer 1 erhält,

   setze $Y(1) = x$, $Y = (Y(1))$

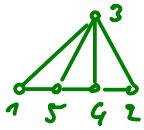
2) Für $i = 1, 2, \ldots$ numeriere die mit neuem i-ten Knoten benachbarten Knoten, die noch nicht neu numeriert sind, in der Reihenfolge ihres Grads, d.h.

   a) $Z_i = Adj(Y(i)) \setminus Y$

   b) ordne Z_i nach Grad $\Rightarrow$ erhalte Vektor $\tilde{Z}_i$

   c) $Y = (Y \;\; \tilde{Z}_i)$

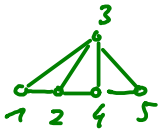
Bsp:



$\hat{=}$

i	$Y(i)$	Z_i	$\tilde{Z}_i$	Y
1	1	$\{3,5\} \setminus \{1\}$	(5, 3)	(1, 5, 3)
2	5	$\{1,4,3\} \setminus \{1,5,3\}$	(4)	(1, 5, 3, 4)
3	3	$\{1,2,4,5\} \setminus \{1,5,3,4\}$	(2)	(1, 5, 3, 4, 2)

$\Rightarrow$



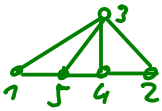
$\hat{=}$

Als Startknoten eignen sich Knoten mit größtmöglichem Abstand zu allen anderen, sog. periphere Knoten, da dann die volle Bandbreite erst in den letzten Zeilen erreicht wird. Ein „pseudo-peripherer" Knoten ist einfacher zu finden.

Def: Eine Schichtung von G mit Wurzel $x \in X$ ist eine Partition $X = \bigcup_{i=0}^{e(x)} L_i(x)$ mit

$$L_0(x) = \{x\}, \quad L_i(x) = Adj(L_{i-1}(x)) \setminus \bigcup_{j=0}^{i-1} L_j(x)$$

Bsp:



Wurzel $x = 1 \Rightarrow L_0(1) = \{1\}, L_1(1) = \{3,5\}, L_2(1) = \{4,2\}$

Bem: $L_i(x)$ beschreibt die Knoten, die von x i Kanten Abstand haben. Die Anzahl an Schichtungen $e(x)$ ist der maximale Abstand, den ein Knoten von x haben kann.

Alg: (Bestimmung pseudo-peripheren Knotens) input: Graph $G = (X, E)$, output: r

1) Wähle $x \in X$ mit Schichtung $L_0(x), \ldots, L_{e(x)}(x)$

2) Wähle Knoten $r \in L_{e(x)}(x)$ mit minimalem Grad

3) Berechne Schichtung $L_0(r), \ldots, L_{e(r)}(r)$

4) Falls $e(r) > e(x)$ ersetze x durch r und gehe zu 2)

Heute: · Symbolische Cholesky-Zerlegung

· Minimum Degree Algorithmus

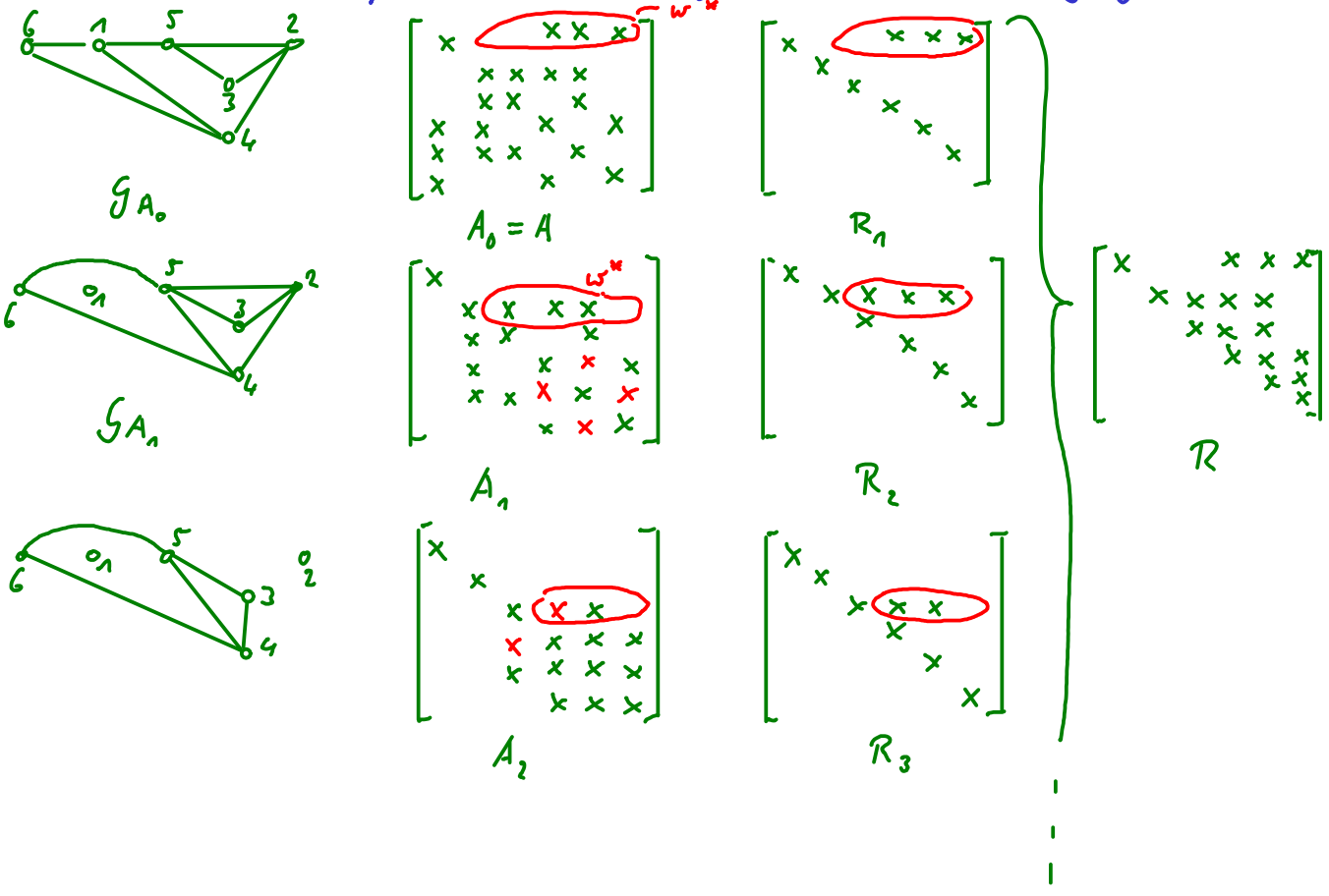
Die Cholesky-Zerlegung $A=R^*R$ funktioniert ohne Pivotisierung. Ist $A\in\mathbb{K}^{m\times m}$ dünn besetzt, kann eine diagonale Pivot-Suche sinnvoll sein, um den „Fill in" (Nicht-Null-Elemente von $R+R^*$ an Stellen, wo A Nulleinträge hat) zu reduzieren.

Der $j$te Schritt der Cholesky-Zerlegung von $A\in\mathbb{K}^{m\times m}$ zerlegt eine Matrix der Form $\begin{pmatrix} I & a & w^* \\ & w & K \end{pmatrix} =: A_{j-1}$ in $R_j^* A_j R_j$ für $R_j = \begin{pmatrix} I & \sqrt{a} & w^*/\sqrt{a} \\ & 0 & I \end{pmatrix}$, $A_j = \begin{pmatrix} I & & \\ & 1 & \\ & & K - \frac{ww^*}{a} \end{pmatrix}$. $\Rightarrow$ I.A. ist $(A_j)_{kl}\neq 0$ wenn $(A_{j-1})_{kl}\neq 0$ oder ww^* an dieser Stelle nicht 0 ist, also $(A_{j-1})_{kj}\neq 0 \wedge (A_{j-1})_{jl}\neq 0$. Auch ist $R_{kl} = (R_m\cdots R_1)_{kl}\neq 0$, wenn $(R_k)_{kl}\neq 0$, also $(A_{k-1})_{kl}\neq 0$. Es folgt:

Thm: Sei $A\in\mathbb{K}^{m\times m}$ positiv definit mit Cholesky-Zerlegung $A=R^*R$. Sei $G_A=(X,E_A)$, $X=\{1,\dots,m\}$ ein zu A und $G_F=(X,E_F)$ ein zu $F=R+R^*$ gehörender Graph. Falls während der Cholesky-Zerlegung keine numerische Kompensation eintritt (d.h. Summen aus Nicht-Null-Einträgen zu Null werden), gilt $(k,l)\in E_F \iff \begin{cases} (k,l)\in E_A \text{ oder} \\ \exists \mu\in\{1,\dots,\min(k,l)\} \text{ mit } (k,\mu)\in E_F \wedge (l,\mu)\in E_F \end{cases}$

Der Graph von A_j (und somit von R_{j+1}) entsteht aus dem Graphen zu A_{j-1} durch Ausschneiden des $j$ten Knoten und Verbinden aller seiner Nachbarknoten untereinander. Die Ermittlung der Besetztheitsstruktur von R auf diese Art heißt symbolische Cholesky-Zerlegung.

Bsp:



Alg: (Symbolische Cholesky-Zerlegung) input: $A \in \mathbb{K}^{m\times m}$, zugehöriger Graph $G_0 = (X, E_0)$, output: G_F

for $i = 1$ bis $m-1$

$$\left.\begin{aligned} \tilde{E}_i &= E_{i-1} \setminus \{(k,l) \in E_{i-1} \mid k = i \text{ oder } l = i\} \\ \tilde{F}_i &= \{(k,l) \mid k,l \in Adj(\{i\})\} \\ E_i &= \tilde{E}_i \cup \tilde{F}_i \\ F_i &= \tilde{F}_i \setminus \tilde{E}_i \end{aligned}\right\} (*)$$

end

$G_F = (X, E_0 \cup F_1 \cup \dots \cup F_{m-1})$

Im i-ten Schritt der Cholesky-Zerlegung suchen wir nun ein diagonales Pivot-Element, um den Fill-in zu reduzieren. Da beim zugehörigen Graphen beim Ausschneiden eines Knotens neue Kanten neur über benachbarte Knoten entstehen können, sollte man zunächst Knoten mit minimalem Grad ausschneiden bzw. als Pivot-Element wählen.

Alg: (Minimum Degree Algorithmus) input: $A \in \mathbb{K}^{m\times m}$ mit Graph $G_0 = (X_0, E_0)$

for $i = 1$ bis m

wähle $z \in X_{i-1}$ mit minimalem Grad ← Pivot-Element

gib z neue Nummer i

$X_i = X_{i-1} \setminus \{i\}$

ermittle E_i wie in $(*)$

end

Bem: Endnumerierung ist unabhängig von Ausgangsnumerierung.

Heute: · Kleinste Fehlerquadrate

Über- und unterbestimmte Gleichungssysteme

Methode der kleinsten Fehlerquadrate

Zur Parameterbestimmung in einem Anwendungsproblem werden oft mehr Messungen als nötig durchgeführt. Dies führt zu überbestimmten Gleichungssystemen.

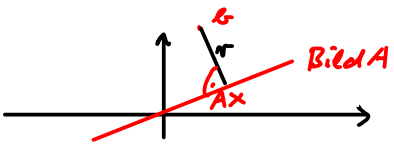
Bsp: gesucht: Monomkoeffizienten $a_0, \dots, a_{n-1}$ des Polynoms $(n-1)$-ten Grades durch die Punkte $(x_1, f_1), \dots, (x_m, f_m) \in \mathbb{R}^2$, $m > n$, $x_1 < \dots < x_m$.

$\Rightarrow$ Löse $A \begin{pmatrix} a_0 \\ \vdots \\ a_{n-1} \end{pmatrix} = \begin{pmatrix} f_1 \\ \vdots \\ f_m \end{pmatrix}$ mit Vandermonde-Matrix $A = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_m & \cdots & x_m^{n-1} \end{pmatrix} \in \mathbb{R}^{m\times n}$

Im Folgenden sei $A \in \mathbb{K}^{m \times n}$ mit $m > n$. I.A. existiert keine Lösung von $Ax = b$; wir suchen daher ein x mit minimalem Residuum.

Def: (Kleinste-Quadrate-Lösung / least squares solution LSS) $x \in \mathbb{K}^n$ heißt LSS von $Ax = b$, wenn

$$\|Ax - b\|_2^2 \leq \|Ay - b\|_2^2 \quad \forall\, y \in \mathbb{K}^n$$



Bem: Die LSS entspricht offenbar dem zu b nächsten Punkt in Bild A.

Thm: x ist LSS zu $Ax = b$ $\Leftrightarrow$ $r := Ax - b \perp$ Bild A $\Leftrightarrow$ $A^*Ax = A^*b$ $\Leftrightarrow$ $Ax = Pb$

für die orthogonale Projektion P auf Bild A. x ist eindeutig $\Leftrightarrow$ A hat vollen Rang.

Bew:
- $r \perp$ Bild A $\Leftrightarrow$ $Ax = Pb$ folgt aus Eigenschaften der orthogonalen Projektion.
- $r \perp$ Bild A $\Leftrightarrow$ $A^*r = 0$ $\Leftrightarrow$ $A^*Ax = A^*b$
- Sei $Ax = Pb$ und y eine LSS. Es ist $\|Ay - b\|^2 \overset{\text{Pythagoras \& } Ax - Ay \perp r}{=} \|Ax - b\|^2 + \|Ax - Ay\|^2$

  $\Rightarrow$ x ist LSS und jede LSS y erfüllt $x - y \in \ker A$, d.h. $Ay = Pb$.
- Eindeutigkeit bei vollem Rang folgt aus $x - y \in \ker A = \{0\}$. □

Im Folgenden habe A vollen Rang. $A^*Ax = A^*b$ heißen die „Normalengleichungen" (NG).

Sei $A = \hat{Q}\hat{R}$ eine reduzierte QR-Zerlegung und $A = \hat{U}\hat{\Sigma}V^*$ eine reduzierte SVD (entsteht aus $U\Sigma V^*$ durch Weglassen der letzten $m-n$ Spalten von U und Zeilen von Σ). Dann kann (NG) auch geschrieben werden als $\hat{R}^*\hat{Q}^*\hat{Q}\hat{R}x = \hat{R}^*\hat{Q}^*b$ bzw.

$$\hat{R}x = \hat{Q}^*b$$

oder als $V\hat{\Sigma}^*\hat{U}^*\hat{U}\hat{\Sigma}V^*x = V\hat{\Sigma}\hat{U}^*b$ bzw.

$$\hat{\Sigma}V^*x = \hat{U}^*b.$$

Es ergeben sich 3 mögliche Algorithmen zur Bestimmung einer LSS:

Alg: (LSS per NG)

1. Berechne A^*A, A^*b
2. Cholesky-Zerlegung $A^*A = R^*R$
3. Löse $R^*Rx = A^*b$ mit Vorwärts- & Rückwärtseinsetzen.

Alg: (LSS per QR)

1. reduzierte QR-Zerlegung $A = \hat{Q}\hat{R}$ via Householder-Verfahren
2. Berechne $\hat{Q}^*b$
3. Löse $\hat{R}x = \hat{Q}^*b$ per Rückwärtseinsetzen

Alg: (LSS per SVD)

1. Berechne reduzierte SVD $A = \hat{U}\hat{\Sigma}V^*$
2. Berechne $\hat{U}^*b$, $w = \hat{\Sigma}^{-1}(\hat{U}^*b)$, $x = Vw$

Thm: Aufwand von LSS per { NG $\sim mn^2 + \frac{n^3}{3}$; QR $\sim 2mn^2 - \frac{2}{3}n^3$; SVD $\sim 2mn^2 + 11n^3$ } flops. } Hausaufgabe

```
format long;
m = 100; n = 15;
t = linspace(0,1,m);
A = fliplr(vander(t)); A = A(1:m,1:n);
b = exp(sin(4*t'))/2006.787453080206;

[Q,R] = qr(A,0);
x = R\(Q'*b); x(15)

x = A\b; x(15)

[Q,R] = mgs(A);
x = R\(Q'*b); x(15)

[Q,R] = mgs([A b]);
x = R(1:n,1:n)\R(1:n,n+1); x(15)

x = (A'*A)\(A'*b); x(15)

[U,S,V] = svd(A,0);
x = V*(S\(U'*b)); x(15)
```

m Messungen, Polynomgrad n-1
m Stützstellen für Polynominterpolation
$\Rightarrow$ exakte $x(15) = 1$

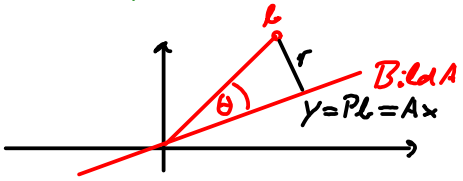
QR-Faktorisierung mit Pivoting

instabil, da Q kaum orthogonal

stabile Berechnung von Q^*b

instabil

Def: (Pseudoinverse) $A^+ = (A^*A)^{-1}A^*$ heißt Pseudoinverse von A.



Sei $\theta = \arccos\frac{\|y\|}{\|b\|}$, $\eta = \frac{\|A\|\,\|x\|}{\|y\|}$.

Thm: Sei $\kappa(A) = \|A\|\,\|A^+\|$. Die relative Konditionszahl (bzgl. $\|\cdot\|_2$) der LSS von $Ax = b$ bzgl. Perturbationen in b ist $\kappa_b = \frac{\kappa(A)}{\eta\cos\theta}$ und in A ist $\kappa_A \le \kappa(A) + \frac{\kappa(A)^2\tan\theta}{\eta}$.

Bew: · Sei $f(b) = A^+b$. $\kappa_b = \frac{\|Df(b)\|}{\|x\|/\|b\|} = \|A^+\|\frac{\|b\|}{\|y\|}\frac{\|y\|}{\|x\|} = \|A^+\|\frac{1}{\cos\theta}\frac{\|A\|}{\eta} = \frac{\kappa(A)}{\eta\cos\theta}$

· Sei $A = U\Sigma V^*$ eine SVD. Da $\|\cdot\|_2$ invariant unter Multiplikation unitärer Matrizen, können wir zur Konditionszahlberechnung A und b von links mit U^*, A und x von rechts mit V multiplizieren und uns somit auf den Fall $A = \begin{pmatrix} A_n \\ 0 \end{pmatrix} = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \\ 0 & & \end{pmatrix}$, $b = \begin{pmatrix} b_n \\ b_2 \end{pmatrix}$, $x = A_n^{-1}b_n$ beschränken.

Eine Perturbation δA kann geschrieben werden als $\delta A = \begin{pmatrix} \delta A_n \\ \delta A_2 \end{pmatrix}$. δA_n ändert nicht Bild A sondern nur x zu $x + \delta x = (A_n + \delta A_n)^{-1}b_n$, d.h. die Kondition ist

$$\frac{\|\delta x\|}{\|x\|} \Big/ \frac{\|\delta A_n\|}{\|A\|} \le \kappa(A_n) = \kappa(A).$$

δA_2 kippt Bild A um einen Winkel $\delta\alpha$, während A_1 unverändert bleibt. Dies ist äquivalent einer Perturbation von b_2 um δb_2 in $x = A_1^{-1} b_2$ mit Konditionszahl

$$\frac{\|\delta x\|}{\|x\|} \Big/ \frac{\|\delta b_2\|}{\|b_2\|} = \frac{\|A_1^{-1}\delta b_2\|}{\|\delta b_2\|} \frac{\|b_2\|}{\|x\|} \le \|A_1^{-1}\| \frac{\|A_1 x\|}{\|x\|} = \frac{\kappa(A_1)}{\eta} = \frac{\kappa(A)}{\eta}.$$

Wie hängen δb_2 und δA_2 zusammen? Die NG zu $\begin{pmatrix} A_1 \\ \delta A_2 \end{pmatrix}(x+\delta x) = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$ sind

$$\left(A_1^* + \delta A_2^* \delta A_2\right)(x+\delta x) = A_1^* b_1 + \delta A_2^* b_2,$$

d.h. mit $(I+\delta B)^{-1} = I - \delta B$ für infinitesimales $\delta B = A_1^{-1} \delta A_2^* \delta A_2 A_1^{-1}$

$$x + \delta x = A_1^{-1}\left[I - \delta B\right]\left(b_1 + A_1^{-1}\delta A_2 b_2\right) \equiv A_1^{-1}(b_1 + \delta b_1).$$

Unter Vernachlässigung von Termen höherer Ordnung ergibt sich

$$\delta b_1 = A_1^{-1} \delta A_2 b_2,$$

also $\dfrac{\|\delta x\|}{\|x\|} \Big/ \dfrac{\|\delta A_2\|}{\|A\|} = \underbrace{\dfrac{\|\delta x\|}{\|x\|} \Big/ \dfrac{\|\delta b_1\|}{\|b_1\|}}_{\le \frac{\kappa(A)}{\eta}} \cdot \underbrace{\dfrac{\|\delta b_1\|}{\|b_1\|} \Big/ \dfrac{\|\delta A_2\|}{\|A\|}}_{\le \|A_1^{-1}\|\|\delta A_2\| \frac{\|b_2\|}{\|b_1\|} = \|A_1^{-1}\|\|\delta A_2\|\tan\theta} \le \kappa(A)^2 \frac{\tan\theta}{\eta}$ □

Thm: LSS per QR und LSS per SVD sind rückwärtsstabil. LSS per NG ist instabil.

Bem: Die Erklärung für LSS per NG ist folgende: Die Lösung von $A^*Ax = A^*b$ per Cholesky-Zerlegung liefert ein $\tilde{x}$ mit $\frac{\|\tilde{x} - x\|}{\|x\|} = O\left(\kappa(A^*A)\,\varepsilon_m\right)$ (dies ist wegen der Stabilität der Cholesky-Zerlegung die bestmögliche Fehlerschranke). Nun ist $\kappa(A^*A) = \kappa(A)^2$, d.h. der relative Fehler von LSS per NG verhält sich wie $\kappa(A)^2\varepsilon_m$.

Für einen stabilen Algorithmus ist der relative Fehler jedoch $\sim (\kappa_A + \kappa_{b_2})\varepsilon_m$ $\sim \left(\frac{\kappa(A)}{\eta\cos\theta} + \kappa(A) + \frac{\kappa(A)^2\tan\theta}{\eta}\right)\varepsilon_m$, d.h. für kleine $\theta \sim \frac{1}{\kappa(A)}$ oder große $\eta \sim \kappa(A)$ ist der relative Fehler nur $\sim \kappa(A)\,\varepsilon_m$.

Heute:
- Minimum-Norm-Lösung
- Tikhonov-Regularisierung
- Banachscher Fixpunktsatz

Minimum-Norm-Lösung

Im Folgenden sei $A \in \mathbb{R}^{m\times n}$ ohne Annahmen an den Rang oder an m, n. Die LSS ist i.A. nicht eindeutig. Daher wählt man oft diejenige LSS kleinster Norm.

Def: (Moore-Penrose-Inverse MPI) $x^+ \in \mathbb{R}^n$ heißt MPI- oder Minimum-Norm-Lösung von $Ax = b$, wenn x^+ eine LSS mit minimaler Norm ist.

Thm: $\ker A^* = \text{Bild}\, A^\perp \quad (\Leftrightarrow (\ker A^*)^\perp = \text{Bild}\, A)$

Bew: $x \in \ker A^* \Leftrightarrow A^* x = 0 \Leftrightarrow \text{Bild}\, A \perp x$ □

Thm: Die MPI-Lösung x^+ ist eindeutig und charakterisiert durch $A^*A\, x^+ = A^* b$ und $x^+ \in \text{Bild}\, A^*$.

Bew:
- Sei x eine LSS. Die Menge M aller LSS ist $x + \ker A$, denn $y \in M \Leftrightarrow A^*Ay = A^*b = A^*Ax$
  $\Leftrightarrow x - y \in \ker(A^*A) = \ker A$ da $\ker A^* = \text{Bild}\, A^\perp$.
- Sei $x^+ \in M$ mit $x^+ \in \text{Bild}\, A^* = \ker A^\perp$, sei $y \in M$. Nach Pythagoras ist wegen $x^+ - y \in \ker A$
  $\|y\|^2 = \|x^+\|^2 + \|y - x^+\|^2 \Rightarrow x^+$ ist eindeutige MPI-Lösung
- $x^+ \in M \cap \text{Bild}\, A^*$ existiert (wähle $x^+ = x - P_{\ker A} x$ für $P_{\ker A}$ die orthogonale Projektion auf $\ker A$) □

Bsp: gg: Messung (x_i, f_i), ges: a_0, a_1 MPI-Lösung zu $a_0 + a_1 x_i = f_i$

$\Rightarrow \begin{pmatrix} 1 \\ x_1 \end{pmatrix} \begin{pmatrix} 1 & x_1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \end{pmatrix} f_1$ & $\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \in \text{Bild} \begin{pmatrix} 1 \\ x_1 \end{pmatrix}$, d.h. $\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ x_1 \end{pmatrix}$

$\Rightarrow$ suche α mit $\begin{pmatrix} 1 \\ x_1 \end{pmatrix} \begin{pmatrix} 1 & x_1 \end{pmatrix} \alpha \begin{pmatrix} 1 \\ x_1 \end{pmatrix} = \begin{pmatrix} 1 \\ x_1 \end{pmatrix} f_1$

$\Rightarrow \alpha = \frac{f_1}{1 + x_1^2}$, $\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \frac{f_1}{1 + x_1^2} \begin{pmatrix} 1 \\ x_1 \end{pmatrix}$

Def: (MPI) Die Abbildung $A^+ : b \mapsto x^+$ für x^+ die MPI-Lösung zu $Ax = b$ heißt MPI.

Bem:
- $A \in \mathbb{K}^{m \times m}$ regulär $\Rightarrow A^+ = A^{-1}$
- $A \in \mathbb{K}^{m \times n}$ mit vollem Rang und $m > n \Rightarrow A^+ = (A^*A)^{-1} A^*$
- ——— " ——— $m < n \Rightarrow A^+ = A^*(AA^*)^{-1}$

Thm: $\cdot\ (A^+)^+ = A$ $\cdot\ AA^+A = A$ $\cdot\ A^+AA^+ = A^+$ $\cdot\ A^+A$ & AA^+ sind hermitesch

(Hausaufgabe)

Bem: Sei $A = U \Sigma V^*$ eine SVD mit $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_k, 0, \dots, 0)$, dann kann x^+ zu $Ax = b$ berechnet werden durch $w = U^* b$, $v = (w_1/\sigma_1, \dots, w_k/\sigma_k, 0, \dots)$, $x^+ = Vv$.

Die Kondition der MPI-Lösung ist schlecht, wenn $\kappa(A) = \frac{\sigma_1}{\sigma_k}$ groß ist, d.h. kleine Datenfehler werden enorm verstärkt. Dann ist die MPI-Lösung praktisch nicht brauchbar und wird z.B. ersetzt durch Folgendes.

Def: (Tikhonov-Inverse) Sei $\gamma > 0$. $x_\gamma^+ = \underset{x \in \mathbb{K}^n}{\text{argmin}} \|Ax - b\|_2^2 + \gamma^2 \|x\|_2^2$ heißt Tikhonov-regularisierte Lösung von $Ax = b$; die lineare Abbildung $A_\gamma^+ : b \mapsto x_\gamma^+$ heißt Tikhonov-Inverse.

Thm: Die Tikhonov-regularisierte Lösung ist eindeutig und gegeben durch

$$(A^*A + \gamma^2 I)\, x_\gamma^+ = A^* b \quad \text{bzw.} \quad x_\gamma^+ = A_\gamma^+ b = V(\Sigma^2 + \gamma^2 I)^{-1} \Sigma U^* b$$

(für $A = U \Sigma V^*$ die SVD). (Hausaufgabe)

Iterative Lösung von Gleichungssystemen

Häufig sind die direkten Lösungsverfahren trotz Ausnutzung der dünnen Besetztheit zu langsam.
In der Praxis nutzt man daher typischerweise iterative Verfahren, die eine Näherung der Lösung liefern.

Banachscher Fixpunktsatz

Def: (Kontraktion) Sei X normierter Raum, $U \subset X$.

- $f: U \to X$ heißt Kontraktion $\Leftrightarrow \exists\, q \in (0,1)$ mit $\|f(x)-f(y)\| \le q\|x-y\| \;\forall x,y \in U$
- $x \in U$ heißt Fixpunkt von f $\Leftrightarrow f(x) = x$

Thm (Banachscher Fixpunktsatz): Sei X ein Banachraum, $U \subset X$ abgeschlossen, $f: U \to U$ eine Kontraktion, dann gilt:

1) f besitzt genau einen Fixpunkt $\bar{x} \in U$

2) Die „Fixpunktiteration" $x_{k+1} = f(x_k)$, $x_0 \in U$, konvergiert $x_k \to \bar{x}$.

3) Der Fehler erfüllt $\|x_k - \bar{x}\| \le \underbrace{\frac{q}{1-q}\|x_k - x_{k-1}\|}_{\text{„a posteriori-"}} \le \underbrace{\frac{q^k}{1-q}\|x_1 - x_0\|}_{\text{„a priori-Abschätzung"}}$

Beweis: Vorüberlegung:

$$\|x_\ell - x_{\ell-1}\| = \|f(x_{\ell-1}) - f(x_{\ell-2})\| \le q\|x_{\ell-1} - x_{\ell-2}\| \le q^2\|x_{\ell-2} - x_{\ell-3}\| \le \dots \le q^{\ell-k-1}\|x_{k+1} - x_k\| \qquad (*)$$

$$\|x_\ell - x_k\| \le \|x_\ell - x_{\ell-1}\| + \|x_{\ell-1} - x_{\ell-2}\| + \dots + \|x_{k+1} - x_k\|$$

$$\le q^{\ell-k-1}\|x_{k+1} - x_k\| + q^{\ell-k-2}\|x_{k+1} - x_k\| + \dots + q^0\|x_{k+1} - x_k\|$$

$$= \underbrace{\sum_{j=0}^{\ell-k-1} q^j}_{\le \sum_{j=0}^{\infty} q^j = \frac{1}{1-q}} \|x_{k+1} - x_k\| \le \frac{1}{1-q}\|x_{k+1} - x_k\| \overset{(*)}{\le} \frac{q}{1-q}\|x_k - x_{k-1}\| \qquad (**)$$

zu 2) x_k ist eine Cauchyfolge: für $\varepsilon > 0$ wähle $N \in \mathbb{N}$ mit $q^N < \varepsilon \frac{1-q}{\|x_1 - x_0\|}$, dann ist

$$\|x_\ell - x_k\| \overset{(**)}{\le} \frac{q}{1-q}\|x_k - x_{k-1}\| \overset{(*)}{\le} \frac{q^k}{1-q}\|x_1 - x_0\| < \varepsilon \quad \forall\, \ell \ge k \ge N$$

- $\Rightarrow x_k \to \bar{x}$ für ein $\bar{x} \in U$, und $\bar{x}$ ist Fixpunkt, da

$$\|f(\bar{x}) - \bar{x}\| = \lim_{\ell\to\infty} \|f(\bar{x}) - x_{\ell+1}\| = \lim_{\ell\to\infty} \|f(\bar{x}) - f(x_\ell)\| \le \lim_{\ell\to\infty} q\|\bar{x} - x_\ell\| = 0$$

- $\bar{x}$ ist eindeutig, denn $y = f(y) \Rightarrow \|\bar{x} - y\| = \|f(\bar{x}) - f(y)\| \le q\|\bar{x} - y\| \Rightarrow \|\bar{x} - y\| = 0$

3) $\|\bar{x} - x_k\| = \lim_{\ell\to\infty} \|x_\ell - x_k\| \overset{(**)}{\le} \frac{q}{1-q}\|x_k - x_{k-1}\| \overset{(*)}{\le} \frac{q^k}{1-q}\|x_1 - x_0\|$ □

Thm: Sei $f: U \to \mathbb{R}^m$ differenzierbar, $U \subset \mathbb{R}^m$ konvex, $\|\cdot\|$ Norm auf $\mathbb{R}^m$.

f ist Kontraktion mit Konstante q $\Leftrightarrow \|Df(x)\| \le q \quad \forall\, x \in U$

Beweis: „$\Rightarrow$" $Df(x)v = \lim_{h\to 0} \frac{f(x+hv) - f(x)}{h} \quad \forall\, v \in \mathbb{R}^m$

$$\|Df(x)v\| = \lim_{h\to 0} \frac{\|f(x+hv) - f(x)\|}{h} \le q \lim_{h\to 0} \frac{\|hv\|}{h} = q\|v\| \quad \forall v \in \mathbb{R}^m$$

„$\Leftarrow$" Sei $x, y \in U$, $F(s) = f(x + s(y-x)) \Rightarrow F'(s) = Df(x + s(y-x))(y-x)$

$$\|f(y)-f(x)\| = \|F(1)-F(0)\| = \left\|\int_0^1 F'(s)\,ds\right\| \le \sup_s \|F'(s)\| \le \|Df\|\,\|y-x\| \le q\,\|y-x\|$$ □

Bsp: · $f: [-1,1] \to [-1,1]$, $f(x) = \frac{\sin x}{2}$ ist Kontraktion mit Fixpunkt 0, denn $|f'(x)| = \frac{|\cos(x)|}{2} \le \frac{1}{2}$

· $f: \mathbb{R} \to \mathbb{R}$, $f(x) = \tan x$ ist keine Kontraktion, da $|f'(x)| = \frac{1}{\cos^2 x} \ge 1$. Ein Fixpunkt von f kann aber gefunden werden nach der Umformung von $x = \tan x$ in $\underbrace{\arctan x = x}_{\text{Kontraktion}}$

· $f: \mathbb{R}^2 \to \mathbb{R}^2$, $f(x) = Bx - b$ mit $B = O\Lambda O^T$ für

$$O = \frac{1}{5}\begin{pmatrix} 4 & -3 \\ 3 & 4 \end{pmatrix} \text{ orthogonal}, \quad \Lambda = \begin{pmatrix} \frac{13}{14} & 0 \\ & \end{pmatrix}, \quad B = \frac{1}{350}\begin{pmatrix} 208 & 156 \\ 156 & 117 \end{pmatrix}$$

ist kontrahierend bzgl. $\|\cdot\|_2$, da $\|Df\|_2 = \|B\|_2 = \frac{13}{14} < 1$, (*)

aber nicht bzgl. $\|\cdot\|_\infty$, da $\|Df\|_\infty = \max_i \sum_{j=1}^{2} |B_{ij}| = \frac{364}{350} > 1$.

(*) $\Rightarrow$ f hat eindeutigen Fixpunkt $(\bar{x} = (B-I)^{-1}b)$

```
B = [208 156;156 117]/350; b = [0;0];
q = 13/14;
x = [1;1];
xNew=B*x-b; error=norm(xNew,2), aposteriori=q/(1-q)*norm(xNew-x,2), x=xNew;
xNew=B*x-b; error=norm(xNew,2), aposteriori=q/(1-q)*norm(xNew-x,2), x=xNew;
...
```

Heute: · iterativ Gleichungssystemlöser

Iterativ Löser für lineare Gleichungssysteme

Def (Spektralradius): Sei $A \in \mathbb{C}^{m\times m}$. $\rho(A) = |\lambda|$ für den größten Eigenwert λ heißt Spektralradius von A

Thm: $\|A\| \ge \rho(A)$ für alle induzierten Matrixnormen $\|\cdot\|$.

Bew: Sei λ Eigenwert mit Eigenvektor x, $|\lambda| = \rho(A) \Rightarrow \|A\| \ge \frac{\|Ax\|}{\|x\|} = |\lambda|$ □

Thm: Für alle $\varepsilon > 0$ existiert eine Norm $\|\cdot\|$ auf $\mathbb{C}^m$, sodass $\|A\| \le \rho(A) + \varepsilon$ für die induzierte Matrixnorm

Bew: Es existiert $X \in \mathbb{C}^{m\times m}$ regulär, sodass $J = XAX^{-1} = \begin{pmatrix} \lambda_1 & s_1 & & \\ & \lambda_2 & s_2 & \\ & & \ddots & s_{m-1} \\ & & & \lambda_m \end{pmatrix}$ Jordan-Normalform hat, d.h. λ_i sind die Eigenwerte, $s_i \in \{0,1\}$.

Sei $D = \begin{pmatrix} \varepsilon^0 & & & \\ & \varepsilon^1 & & \\ & & \ddots & \\ & & & \varepsilon^m \end{pmatrix}$ $\Rightarrow \|x\| := \|D^{-1}Xx\|_\infty$ definiert eine Norm (Hausaufgabe)

und $\|A\| = \sup_x \frac{\|Ax\|}{\|x\|} = \sup_x \frac{\|X^{-1}DD^{-1}JDD^{-1}Xx\|}{\|x\|} = \sup_x \frac{\|D^{-1}JD\,(D^{-1}Xx)\|_\infty}{\|D^{-1}Xx\|_\infty} \le \|D^{-1}JD\|_\infty$

$$= \left\| \begin{pmatrix} \lambda_1 & \varepsilon s_1 & & \\ & \lambda_2 & \varepsilon s_2 & \\ & & \ddots & \varepsilon s_{m-1} \\ & & & \lambda_m \end{pmatrix} \right\|_\infty = \max_i |\lambda_i| + \varepsilon |s_i| \le \rho(A) + \varepsilon$$ □

Thm: Sei $A = M - N \in \mathbb{K}^{m\times m}$, M regulär, $b, x_0 \in \mathbb{K}^m$, $x_{k+1} = M^{-1}(Nx_k + b)$.

$x_k \to A^{-1}b$ $\forall x_0, b \in \mathbb{K}^m$ $\Leftrightarrow$ $\rho(B) < 1$ für $B = M^{-1}N$.

Bew: „$\Leftarrow$" $f(x) = M^{-1}(Nx + b)$ ist Kontraktion bzgl. einer Norm $\|\cdot\|$, da $\|Df\| = \|B\|$

$\Rightarrow$ x_k konvergiert gegen den eindeutigen Fixpunkt $\bar{x}$ mit $M\bar{x} = N\bar{x} + b \Leftrightarrow A\bar{x} = 0$.

„$\Rightarrow$" Sei $\rho(B) > 1$, λ Eigenvektor und x_0 Eigenvektor mit $|\lambda| = \rho(B)$, $b = 0$, dann ist $x_k = \lambda^k x_0 \not\to 0$. □

Def: Sei $A \in \mathbb{K}^{m\times m}$ regulär, $b \in \mathbb{K}^m$, $A = L + D + R$ für eine invertierbare Diagonalmatrix D und eine untere und obere Dreiecksmatrix L und R ohne Diagonale. Die Iteration $x_{k+1} = M^{-1}(Nx_k + b)$ heißt

- **Jakobi- oder Gesamtschritt-Verfahren** für $M = D$, $N = -L-R$, d.h.

$$(x_{k+1})_i := \frac{1}{A_{ii}}\left(b_i - \sum_{j\neq i} A_{ij}(x_k)_j\right) = \text{Lsg von } A_{i:}\begin{pmatrix}(x_k)_1\\ \vdots\\ \times\\ (x_k)_{i+1}\\ \vdots\\ (x_k)_m\end{pmatrix} = b_i$$

- **Gauß-Seidel- oder Einzelschritt-Verfahren** für $M = D+L$, $N = -R$, d.h.

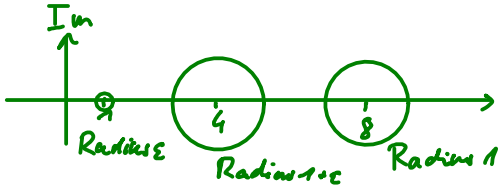
$$(x_{k+1})_i := \frac{1}{A_{ii}}\left(b_i - \sum_{j>i} A_{ij}(x_k)_j - \sum_{j<i} A_{ij}(x_{k+1})_j\right)$$

Bem:
- Beim JV können alle Einträge von x simultan aktualisiert werden, beim GSV werden alle Einträge nacheinander aktualisiert (genau wie bei JV, nur dass immer die bisher aktuellsten Einträge genutzt werden).
- geringer Aufwand, wenn A dünn besetzt

Thm (Satz von Gerschgorin): Jeder Eigenwert von $A \in \mathbb{C}^{m\times m}$ liegt in mindestens einer der m Kreisscheiben der komplexen Ebene mit Zentrum A_{ii} und Radius $\sum_{j\neq i} |A_{ij}|$. Wenn n Kreisscheiben ein zusammenhängendes, von den anderen Scheiben disjunktes Gebiet bilden, liegen darin genau n Eigenwerte (gezählt mit algebraischer Vielfachheit).

Bew: siehe Übung.

Bsp: $A = \begin{pmatrix} 8 & 1 & 0\\ 1 & 4 & \varepsilon\\ 0 & \varepsilon & 1\end{pmatrix}$ hat Gerschgorinkreise



die die drei Eigenwerte enthalten.

Bem: Die Abschätzung kann oft verschärft werden, indem man zu einer ähnlichen Matrix XAX^{-1} übergeht, z.B. mit diagonalem X.

Def: (Diagonaldominanz) $A \in \mathbb{K}^{m\times m}$ heißt (strikt) diagonaldominant, wenn $\sum_{j\neq i} |A_{ij}| \leq (<) |A_{ii}| \ \forall i$

Thm: A strikt diagonaldominant $\Rightarrow$ A regulär und JV, GSV konvergieren.

Bew:
- kein Gerschgorinkreis enthält 0 $\Rightarrow$ 0 ist kein Eigenwert $\Rightarrow$ A regulär
- JV konvergiert $\Leftrightarrow \rho(M^{-1}N) = \rho(\underbrace{-D^{-1}(L+R)}_{=:B}) < 1$; dies gilt, da $B_{ii}=0$ und $\sum_{j\neq i} |B_{ij}| = \sum_{j\neq i} |A_{ij}|/|A_{ii}| < 1 \Rightarrow$ Gerschgorinkreise haben Zentrum 0 & Radius < 1
- GSV konvergiert $\Leftrightarrow \rho(\underbrace{-(D+L)^{-1}R}_{=:B}) < 1$.

Angenommen, $(D+L)^{-1}R$ habe einen Eigenwert $|\lambda| \geq 1$, dann ist $R + \lambda(D+L)$ strikt

diagonaldominant und somit invertierbar. Weiterhin ist das charakteristische Polynom $\chi_B(\lambda) = \det(B - \lambda I) = 0$, dies widerspricht jedoch

$$\det(B-\lambda I) = \det\left(-(D+L)^{-1}\left[R + \lambda(D+L)\right]\right) = \det\left(-(D+L)^{-1}\right)\det\left(R+\lambda(D+L)\right) \neq 0 \quad \square$$

Bem: Der Satz gilt sogar wenn A nur diagonaldominant und zusätzlich irreduzibel ist, d.h. es gibt keine disjunkte Zerlegung $\{1,\dots,m\} = I \cup J$ mit $A_{ij} = 0 \ \forall i \in I, j \in J$.

Heute: · SOR

· Landweberverfahren

· Gradientenverfahren

Def: Sei $A \in \mathbb{K}^{m\times m}$, $\omega \in [0,2]$, $A = L + D + R$ wie zuvor, $b \in \mathbb{K}^m$

· $x_{k+1} = (1-\omega)\, x_k + \omega D^{-1}(b - (L+R)x_k)$ heißt relaxiertes JV.

· $(x_{k+1})_i = (1-\omega)(x_k)_i + \frac{\omega}{A_{ii}}\left(b_i - \sum_{j<i} A_{ij}(x_{k+1})_j - \sum_{j>i} A_{ij}(x_k)_j\right)$

bzw. $x_{k+1} = (D+\omega L)^{-1}\left[(1-\omega)Dx_k + \omega(b - Rx_k)\right]$ heißt relaxiertes GSV.

Für $\omega > 1$ spricht man auch von „successive overrelaxation" (SOR).

Bem: $\omega = 1$ liefert JV und GSV.

Thm: Sei $A = L + D + L^* \in \mathbb{R}^{m\times m}$ symmetrisch positiv definit, $\omega \in (0,2)$, dann konvergiert SOR.

Bew: Sei $B = (D+\omega L)^{-1}((1-\omega)D - \omega L^*)$; wir brauchen $\rho(B) < 1$.

Sei λ Eigenwert von B mit Eigenvektor x, $d = x^*Dx$, $l = x^*Lx = x^*L^*x$

$$\Rightarrow \quad ((1-\omega)D - \omega L^*)x = \lambda(D+\omega L)x$$

$$\Rightarrow \quad (1-\omega)d - \omega l = \lambda(d+\omega l) \quad \Rightarrow \quad \lambda = \frac{(1-\omega)d - \omega l}{d+\omega l}, \text{ d.h.}$$

$|\lambda| < 1$ wenn $d + \omega l > \begin{cases} (1-\omega)d - \omega l \\ -(1-\omega)d + \omega l \end{cases} \Leftrightarrow \begin{cases} \omega(d+2l) > 0 \\ (2-\omega)\, d > 0 \end{cases}$.

Dies gilt wegen $\omega < 2$, $d = \sum_{i=1}^{m} \underbrace{A_{ii}}_{e_i^* A e_i > 0} |x_i|^2 > 0$, $d + 2l = x^*(L + D + L^*)x = x^*Ax > 0$ $\square$

Bem: Für $\omega \notin (0,2)$ ist jedoch $\rho(B) \geq 1$, denn

$$\rho(B)^m \geq |\lambda_1|\cdot|\lambda_2|\cdot\ldots\cdot|\lambda_m| = |\det B| = \left|\det \underbrace{(I+\omega D^{-1}L)^{-1}}_{\begin{pmatrix}1 & & 0\\ \times & \ddots & \\ \times & \times & 1\end{pmatrix}} \underbrace{((1-\omega)I - \omega D^{-1}L^*)}_{\begin{pmatrix}1-\omega & & \\ 0 & \ddots & \\ & & 1-\omega\end{pmatrix}}\right| = (1-\omega)^m.$$

Def: (Landweberverfahren) Das Landweberverfahren zum Finden der MPI-Lösung zu $Ax = b$, $A \in \mathbb{K}^{m\times n}$, $b \in \mathbb{K}^m$, für $\omega > 0$ fest ist $x_0 \in \mathbb{K}^n$, $x_{k+1} = f(x_k) := x_k - \omega A^*(Ax_k - b)$

Bem: Das Landweberverfahren entsteht aus der Fixpunktform der NG, $\bar{x} = \bar{x} + A^*(b - A\bar{x})$, durch Relaxieren, $x_{k+1} = (1-\omega)x_k + \omega\left[x_k + A^*(b - Ax_k)\right]$

Thm: Sei $\omega \in (0, \frac{2}{\|A\|_2^2})$, $x_0 \in \text{Bild} A^*$. Das Landweber-Verfahren konvergiert gegen die MPI Lösung

Bew: $U = \text{Bild} A^* \subset \mathbb{R}^n$ ist abgeschlossen, $f: U \to U$. Es ist zu zeigen, dass f eine Kontraktion ist, also $\rho(Df|_U) < 1$, dann konvergiert x_k gegen einen Fixpunkt $\bar{x} \in U$, also gegen die MPI-Lösung.

Die Eigenwerte λ_i von $Df|_U = (I - \omega A^* A)|_U$ sind genau $1 - \omega b_i^2$: für b_i die Singulärwerte von $A|_U$. Es ist $\|A\|_2 = b_1 \geq b_2 \geq \dots > 0$ (alle strikt positiv, da $A|_U$ auf $U = \text{Ker} A^\perp$ operiert). $\Rightarrow \lambda_i \in (-1, 1) \quad \forall i$. □

Bem: Die vorgestellten Verfahren lassen sich auch wie folgt interpretieren:

- Wir suchen die Lösung zu $Ax = b$ mittels Fixpunktiteration.
- Der einfache Ansatz $x = (I - A)x + b$ ist nicht immer zielführend, da die rechte Seite typischerweise keine Kontraktion ist.
- Stattdessen konditionieren wir das System zunächst mit einer Matrix Q vor, $QAx = Qb$, und betrachten dann die Fixpunktiteration

$$x_{k+1} = (I - QA)x_k + Qb.$$

- Konvergenz ist umso schneller, je stärker die Kontraktion ist, d.h. je kleiner $\rho(I - QA)$
  $\Rightarrow$ Q sollte eine möglichst gute, aber auch einfache Approximation an A^{-1} sein.
- JV: $Q = D^{-1}$ (für $A = L + D + R$)
  GSV: $Q = (D + L)^{-1}$

Krylov-Raum-Verfahren

Statt eines linearen Gleichungssystems kann man oft auch ein Minimierungsproblem lösen:
Sei $A \in \mathbb{R}^{m \times m}$ symmetrisch positiv definit, $b \in \mathbb{R}^m$ und $f: \mathbb{R}^m \to \mathbb{R}$, $x \mapsto \frac{1}{2} x^* A x - x^* b$.
Wir suchen das Minimum $\bar{x}$ von $f \Rightarrow 0 = \nabla f(\bar{x}) = A\bar{x} - b$, d.h. das Minimum löst $Ax = b$.
Ein mögliches iteratives Verfahren erzeugt eine Folge $x_0, x_1, \dots \in \mathbb{R}^m$ mit $f(x_0) > f(x_1) > \dots$, indem es im k-ten Schritt in eine Richtung d_k geht, in die f abnimmt.

Alg (Liniensuchverfahren) input: $f: \mathbb{R}^m \to \mathbb{R}$, $x_0 \in \mathbb{R}^m$; output: Approximationen x_k an $\text{argmin} f$

for $k = 0, 1, 2, \dots$
  wähle Abstiegsrichtung d_k (d.h. $Df(x_k) d_k < 0$)
  wähle Schrittweite α_k mit $f(x_k + \alpha_k d_k) < f(x_k)$
  $x_{k+1} = x_k + \alpha_k d_k$
end

Bem: Der Algorithmus funktioniert auch für allgemeine Minimierungsprobleme einer Fkt. f.

Thm: Sei $A \in \mathbb{R}^{m\times m}$ symmetrisch positiv definit, dann ist $(x,y)_A := x^* A y$ ein Skalarprodukt auf $\mathbb{R}^m$. Die induzierte Norm sei $\|x\|_A = \sqrt{x^* A x}$.

Bew: Hausaufgabe

Für $Ax = b$ bzw. $f(x) = \frac{1}{2} x^* A x - x^* b \to \min!$, A symmetrisch positiv definit, seien das Residuum und der Fehler

$$r_k = b - A x_k, \qquad e_k = x_k - A^{-1} b = -A^{-1} r_k.$$

Wir wollen die Richtung des steilsten Funktionsabfalls wählen, $d_k = -\nabla f(x_k) = -Ax_k + b = r_k$, und die optimale Schrittweite, $\alpha_k = \operatorname{argmin}_\alpha f(x_k + \alpha d_k) = \operatorname{argmin}_\alpha f(x_k) + \alpha d_k^*(Ax_k - b) + \frac{\alpha^2}{2} d_k^* A d_k$

Alg: (Gradientenverfahren für lineares Gleichungssystem) input: A, b, x_0; output: Approximationen x_k an $A^{-1}b$

for $k = 0, 1, 2, \dots$

$$d_k = r_k = b - A x_k$$

$$\alpha_k = \frac{-d_k^*(Ax_k - b)}{d_k^* A d_k} = \frac{\|r_k\|_2^2}{\|r_k\|_A^2}$$

$$x_{k+1} = x_k + \alpha_k d_k$$

end

Def: (Krylov-Raum) Sei $A \in \mathbb{R}^{m\times m}$, $y \in \mathbb{R}^m$. Der k-te Krylovraum ist $K^k(A,y) = \langle y, Ay, \dots, A^{k-1} y\rangle$.

Bem: $r_{k+1} = b - A x_{k+1} = b - A x_k - \alpha_k A r_k = (I - \alpha_k A) r_k = (I - \alpha_k A)(I - \alpha_{k-1} A) r_{k-1} = \dots$

$$= (I - \alpha_k A) \cdots (I - \alpha_0 A) r_0 \in K^{k+1}(A, r_0),$$

- $x_{k+1} \in x_k + K^{k+1}(A, r_0)$, $x_k \in x_{k-1} + K^k(A, r_0) \subset x_{k-1} + K^{k+1}(A, r_0)$, …

$$x_{k+1} \in x_0 + K^{k+1}(A, r_0)$$

Lemma: Im k-ten Schritt minimiert das Gradientenverfahren sowohl f als auch die A-Norm des Fehlers auf $x_0 + \langle r_k\rangle =: V_k$, d.h.

a) $f(x_{k+1}) = \min_{x \in V_k} f(x)$, b) $\|e_{k+1}\|_A = \min_{x\in V_k} \|x - A^{-1} b\|_A$, bzw.

c) $\alpha_k = \operatorname{argmin}_\alpha f(x_k + \alpha r_k)$, d) $\alpha_k = \operatorname{argmin}_\alpha \|x_k + \alpha r_k - A^{-1} b\|_A = \operatorname{argmin}_\alpha \|e_k + \alpha r_k\|_A$

Bew:

- a) $\Leftrightarrow$ c) & b) $\Leftrightarrow$ d)
- c) per definitionem von α_k
- d): $\|(I - \alpha A) e_k\|_A = e_k^* A e_k - 2 \underbrace{e_k^* A (\alpha A e_k)}_{\alpha r_k^* r_k} + \underbrace{(\alpha A e_k)^* A (\alpha A e_k)}_{\alpha^2 r_k^* A r_k}$ wird von $\frac{\|r_k\|_2^2}{\|r_k\|_A^2}$ minimiert □

$$= \operatorname{argmin}_\alpha \|(I - \alpha A) e_k\|_A$$

Thm: Sei $A \in \mathbb{R}^{m\times m}$ symmetrisch positiv definit, $\kappa = \kappa(A) = \|A\|_2 \|A^{-1}\|_2$. Das Gradientenverfahren konvergiert mit

$$\|e_k\|_A \le \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|e_0\|_A.$$

Bew: · Sei $A = U^* \Sigma U$ SVD, $\Sigma = \mathrm{diag}(\sigma_1, \dots, \sigma_m)$, dann ist

$$\|(I-\alpha A)\|_A^2 = \sup_x \frac{x^*(I-\alpha A)^* A (I-\alpha A)x}{x^* A x} = \sup_x \frac{x^* U^* (I-\alpha\Sigma)\Sigma(I-\alpha\Sigma)Ux}{x^* U^* \Sigma U x}$$

$$= \sup_{z = \sqrt{\Sigma}\, U x} \frac{z^*(I-\alpha\Sigma)(I-\alpha\Sigma)z}{z^* z} = \sup_z \frac{\|(I-\alpha\Sigma)z\|_2^2}{\|z\|_2^2} = \max_i |1-\alpha\sigma_i|^2$$

· $\|e_{k+1}\|_A = \|(I-\alpha_k A)e_k\|_A \overset{\text{Lemma}}{=} \min_\alpha \|(I-\alpha A)e_k\|_A \le \min_\alpha \|I-\alpha A\|_A \|e_k\|_A = \min_\alpha \max_i |1-\alpha\sigma_i| \|e_k\|_A$

Für $\alpha = \frac{2}{\sigma_1+\sigma_m}$ ist $1-\alpha\sigma_i = \frac{\sigma_1+\sigma_m-2\sigma_i}{\sigma_1+\sigma_m} \in \left[\frac{-\sigma_1+\sigma_m}{\sigma_1+\sigma_m}, \frac{\sigma_1-\sigma_m}{\sigma_1+\sigma_m}\right] = \left[-\frac{\kappa-1}{\kappa+1}, \frac{\kappa-1}{\kappa+1}\right]$ mit $\kappa = \sigma_1/\sigma_m$

$\Rightarrow \|e_{k+1}\|_A \le \frac{\kappa-1}{\kappa+1}\|e_k\|_A$ □

Bem: Konditionieren wir $Ax=b$ mit Q vor, $QAQ^*y = Qb$ für $x = Q^*y$, so ist $\|e_k\|_{QAQ^*} \le \left(\frac{\kappa(QAQ^*)-1}{\kappa(QAQ^*)+1}\right)^k \|e_0\|_{QAQ^*}$. Dabei wünschen wir $\kappa(QAQ^*)$ nahe bei 1, also $Q \approx \sqrt{A^{-1}}$, um schnelle Konvergenz zu erhalten. Als Vorkonditionierer können z.B. ähnliche wie für FV oder GSV gewählt werden.

Heute: · CG-Verfahren

Sei $A \in \mathbb{R}^{m\times m}$ symmetrisch positiv definit. Das Gradientenverfahren minimiert im $k$ten Schritt $f(x) = \frac{1}{2}x^*Ax - x^*b$ und $\|e\|_A$ über $x \in x_k + \langle r_k\rangle \subset x_0 + K^{k+1}(A, r_0)$. Können wir es verbessern zu einem Verfahren, das f und $\|e\|_A$ im $k$ten Schritt über $x_0 + K^{k+1}(A, r_0)$ minimiert? Ja, mit konjugierten Suchrichtungen d_k!

Def: Sei $A \in \mathbb{R}^{m\times m}$ symmetrisch positiv definit. $x, y \in \mathbb{R}^m$ heißen A-konjugiert $\Leftrightarrow (x,y)_A = 0$.

Thm: Sei H Hilbertraum mit Skalarprodukt $(\cdot,\cdot)$, $v_0, v_1, \dots, v_{k-1}$ orthonormal, $x \in H$. Dann ist

$$x_k = \sum_{j=0}^{k-1} (x, v_j) v_j$$ die Bestapproximation von x in $V = \langle v_0, \dots, v_{k-1}\rangle$, d.h. $\|x_k - x\| = \min_{y\in V} \|y - x\|$.

Bew: · $(x - x_k, v_l) = (x, v_l) - (x_k, v_l) = (x, v_l) - (x, v_l) = 0 \quad \forall\, l$

· $\|x - x_k\|^2 = (x - x_k, x - x_k) = (x - x_k, x - \underbrace{x_k + z}_{\in V}) \quad \forall\, z \in V$, da z = Linearkombination der v_l

· $\|x - x_k\|^2 = (x - x_k, x - y) \le \|x - x_k\| \|x - y\| \quad \forall y \in V \Rightarrow \|x - x_k\| \le \|x - y\| \quad \forall y \in V$ □

Thm: Sei $A \in \mathbb{R}^{m\times m}$ symmetrisch positiv definit, $f(x) = \frac{1}{2}x^*Ax - x^*b$. Wähle im Liniensuchverfahren $d_0, d_1, \dots$ paarweise A-konjugiert und $\alpha_k = \operatorname{argmin}_\alpha f(x_k + \alpha d_k)$. Dann

1) $\alpha_k = r_k^* d_k \,/\, (d_k, d_k)_A$
2) $r_k^* d_j = 0 \quad \forall\, j < k$
3) x_k minimiert $\|x - A^{-1}b\|_A$ und f auf $x_0 + \langle d_0, \dots, d_{k-1}\rangle =: X_k$
4) $x_m = A^{-1}b$

Bew: 1) $f(x_k+\alpha_k d) = f(x_k) + \alpha \underbrace{d_k^*(Ax_k-b)}_{=-r_k^* d_k} + \frac{\alpha^2}{2}\underbrace{d_k^* A d_k}_{=(d_k,d_k)_A}$, nun setze $\frac{d}{d\alpha}$ zu 0

2) Induktion: $r_{j+1}^* d_j = (b-Ax_{j+1})^* d_j = (b - Ax_j - A\frac{r_j^* d_j}{(d_j,d_j)_A} d_j)^* d_j = r_j^* d_j - r_j^* d_j \frac{(d_j,d_j)_A}{(d_j,d_j)_A} = 0$

Sei $r_k^* d_j = 0$, dann ist $r_{k+1}^* d_j = (b-Ax_{k+1})^* d_j = (b - Ax_k - A\frac{r_k^* d_k}{(d_k,d_k)_A} d_k)^* d_j = r_k^* d_j - \frac{r_k^* d_k}{(d_k,d_k)_A}(d_k,d_j)_A = 0$

3a) zz: $y = \sum_{j=0}^{k-1} \alpha_j d_j$ minimiert $\|x_0 + y - A^{-1}b\|_A = \|e_0 + y\|_A$ auf $\langle d_0,\dots,d_{k-1}\rangle$

· Sei $v_j = d_j / \|d_j\|_A$, dann $V := \langle v_0,\dots,v_{k-1}\rangle = \langle d_0,\dots,d_{k-1}\rangle$, $v_0,\dots,v_{k-1}$ orthonormal bzgl. $(\cdot,\cdot)_A$

· $y = \sum_{j=0}^{k-1} (r_j^* v_j) v_j = -\sum_{j=0}^{k-1} (v_j, e_j)_A v_j = -\sum_{j=0}^{k-1} (v_j, e_0)_A v_j$

$r_j = -Ae_j$ $\qquad e_j - e_0 \in \langle d_0,\dots,d_{j-1}\rangle = \langle v_0,\dots,v_{j-1}\rangle$

$\Rightarrow -y$ ist Bestapproximation an e_0 auf V $\Rightarrow \|e_0+y\|_A$ ist minimal auf V

3b) $x_k = \operatorname{argmin}_{x\in X_k} \|x - A^{-1}b\|_A = \operatorname{argmin}_{x\in X_k} \frac{1}{2}\|x-A^{-1}b\|_A^2$

$= \operatorname{argmin}_{x\in X_k} \frac{1}{2}x^*Ax - \frac{(x,A^{-1}b)_A + (A^{-1}b,x)_A}{2} + \frac{(A^{-1}b,A^{-1}b)_A}{2} = \operatorname{argmin}_{x\in X_k} f(x) + \frac{b^*A^{-1}b}{2}$

4) folgt aus 3) □

Bem: Eine alternative Interpretation ist folgende: $\mathbb{R}^m = \langle d_0,\dots,d_{m-1}\rangle$, d.h. $A^{-1}b = x_0 + \sum_{i=0}^{m-1}\alpha_i d_i$ für bestimmte $\alpha_i \in \mathbb{R}$. Diese α_j erhält man durch multiplizieren von $d_j^* A$ von links $\Rightarrow$ $d_j^* b = d_j^* A x_0 + \alpha_j d_j^* A d_j$ bzw. $\alpha_j = \frac{d_j^*(b-Ax_0)}{d_j^* A d_j} = \frac{d_j^* r_0}{(d_j,d_j)_A} = \frac{d_j^* r_j}{(d_j,d_j)_A}$. Die Approximation x_k an $A^{-1}b$ erhält man nun durch Abschneiden der Summe, $x_k = x_0 + \sum_{i=0}^{k-1}\alpha_i d_i$.

Die Idee des konjugierten Gradienten- (CG-) Verfahrens ist, die Suchrichtungen d_k so zu wählen, dass sie konjugiert sind und den Krylovraum aufspannen. Dies geschieht mit Gram-Schmidt-Orthogonalisierung der r_k: Setze $d_0 = r_0$ und

$$d_{k+1} = r_{k+1} - \sum_{j=0}^{k}\left(r_{k+1}, \frac{d_j}{\|d_j\|_A}\right)_A \frac{d_j}{\|d_j\|_A} \qquad (*)$$

Thm: Die so erzeugten r_k und d_k spannen den Krylov-Raum auf, d.h.

$$K^{k+1}(A, r_0) = \langle r_0,\dots,r_k\rangle = \langle d_0,\dots,d_k\rangle$$

Bew: Vollständige Induktion: Induktionsanfang: $d_0 = r_0 \in K^1(A,r_0)$

Induktionsschritt: Sei $K^{j+1}(A,r_0) = \langle r_0,\dots,r_j\rangle = \langle d_0,\dots,d_j\rangle$ $\forall j \le k$:

„$K^{k+2} \supset \langle\dots\rangle$": $r_{k+1} = r_k - \alpha_k A d_k \in K^{k+2}(A,r_0)$, $d_{k+1} \in \langle r_{k+1}, d_0,\dots,d_k\rangle \subset K^{k+2}(A,r_0)$

„$K^{k+2} \subset \langle\dots\rangle$": angenommen $\not\subset$, dann wäre $A^{k+1}r_0 \notin \langle r_0,\dots,r_{k+1}\rangle$ $\Rightarrow$ r_{k+1} ist Linearkombination nur aus $r_0, Ar_0,\dots,A^k r_0$, hat aber keine Komponente $A^{k+1}r_0$ $\Rightarrow$ $r_{k+1} \in K^{k+1}(A,r_0)$

$\Rightarrow r_{k+1} - r_k = -\alpha_k A d_k \in K^{k+1}(A,r_0) \Rightarrow d_k \in K^k(A,r_0)$

$\Rightarrow K^{k+1}(A,r_0) = \langle d_0,\dots,d_k\rangle = \langle K^k(A,r_0), d_k\rangle = K^k(A,r_0)$

$\Rightarrow K^{k+2}(A,r_0) = \langle A K^{k+1}(A,r_0), r_0\rangle = \langle A K^k(A,r_0), r_0\rangle = K^{k+1}(A,r_0) = \langle d_0,\dots,d_k\rangle$ ↯ □

Thm: Diese r_k, d_k erfüllen $(r_{k+1}, d_j)_A = 0 \quad \forall j < k$.

Bew: Wir wissen $r_{k+1} \perp d_0, \dots, d_k \Rightarrow r_{k+1} \perp v \ \forall v \in K^{k+1}(A, r_0) \Rightarrow r_{k+1} \perp Av \ \forall v \in K^k(A, r_0)$

$\Rightarrow (r_{k+1}, v)_A = 0 \ \forall v \in \langle d_0, \dots, d_{k-1} \rangle$ □

Es folgt, dass in (*) alle Summanden bis auf den letzten wegfallen:

$$d_{k+1} = r_{k+1} - \frac{(r_{k+1}, d_k)_A}{(d_k, d_k)_A} d_k = r_{k+1} - \alpha_k \frac{(r_{k+1}, d_k)_A}{r_k^* d_k} d_k = r_{k+1} + \frac{r_{k+1}^* r_{k+1}}{r_k^* r_k} d_k,$$

da $r_{k+1}^* r_{k+1} = (r_k - \alpha_k A d_k)^* r_{k+1} = r_k^* r_{k+1} - \alpha_k (r_{k+1}, d_k)_A = -\alpha_k (r_{k+1}, d_k)_A$ und

$r_k^* d_k = r_k^* (r_k - (\dots) d_{k-1}) = r_k^* r_k$. ($\perp \langle d_0, \dots, d_{k-1} \rangle$)

Alg (CG-Verfahren) input: A sym. pos. def., b; output: Approximation x_k an $A^{-1}b$

$x_0 = 0$, $r_0 = b$, $d_0 = r_0$

for $k = 0, 1, 2, \dots$

$\alpha_k = r_k^* r_k / (d_k, d_k)_A$

$x_{k+1} = x_k + \alpha_k d_k$

$r_{k+1} = r_k - \alpha_k A d_k$

$\beta_{k+1} = r_{k+1}^* r_{k+1} / r_k^* r_k$

$d_{k+1} = r_{k+1} + \beta_{k+1} d_k$

end

Heute: · CG Konvergenz

Bem: Wir haben bereits gezeigt:

· $x_k = \underset{x \in K^k(A,b)}{\operatorname{argmin}} \|x - A^{-1}b\|_A = \underset{x \in K^k(A,b)}{\operatorname{argmin}} f(x)$

· $x_m = A^{-1}b$, daher kann CG auch als direkte Methode aufgefasst werden. Wir möchten jedoch nicht m Schritte ausführen!

Der größte Aufwand in jedem Schritt ist die Matrix-Vektor-Multiplikation Ad_k. Ist A dünn besetzt mit $O(m)$ Einträgen (typischer Fall), kostet jede Iteration $O(m)$ Aufwand, die exakte Lösung x_m würde also mit $O(m^2)$ Aufwand erreicht.

Ein Vorteil von Krylovraum-Verfahren ist, dass sie mit Polynomen von Matrizen zusammenhängen. Sei $p(\lambda) = \sum_{i=0}^{k} a_i \lambda^i$, dann ist $p(A)$ für $A \in \mathbb{R}^{m \times m}$ definiert durch $\sum_{i=0}^{k} a_i A^i$.

Sei A symmetrisch mit Spektralzerlegung $A = U^* \Lambda U$, $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_m)$ den Eigenwerten, U orthogonal, dann ist $\sum_{i=0}^{k} a_i A^i = U^* \left(\sum_{i=0}^{k} a_i \Lambda^i \right) U = U^* \operatorname{diag}(p(\lambda_1), \dots, p(\lambda_m)) U$.

Sei $P_k = \{ p \text{ Polynom vom Grad} \le k \}$, $\hat{P}_k = \{ p \in P_k \mid p(0) = 1 \}$.

Thm: Im CG-Verfahren gilt $\|e_k\|_A = \min_{x \in K^k(A,b)} \|x - A^{-1}b\|_A = \min_{\tilde{p} \in \tilde{P}_k} \|\tilde{p}(A) e_0\|_A$.

Bew: „$\geq$" Für alle $x \in K^k(A,b)$ ist $x = p(A)b = Ap(A)A^{-1}b = -Ap(A)e_0$ für ein $p \in P_{k-1}$

$\Rightarrow x - A^{-1}b = (I - Ap(A))e_0 = \tilde{p}(A)e_0$ für ein $\tilde{p} \in \tilde{P}_k$

„$\leq$" Sei $\tilde{p} \in \tilde{P}_k$, dann ist $\tilde{p}(A)e_0 = -p(A)Ae_0 + e_0 = \underbrace{p(A)b}_{\in K^k(A,b)} + e_0$ für ein $p \in P_{k-1}$. □

Thm: $\|e_k\|_A / \|e_0\|_A \leq \inf_{p \in \tilde{P}_k} \max_{\lambda \text{ Eigenwert von } A} |p(\lambda)|$.

Bew: $\|e_k\|_A^2 / \|e_0\|_A^2 = \inf_{p \in \tilde{P}_k} \|p(A)e_0\|_A^2 / \|e_0\|_A^2 = \inf_{p \in \tilde{P}_k} \underbrace{(e_0^* p(A) A p(A) e_0)}_{B} / \underbrace{(e_0^* A e_0)}_{C}$

mit $B = e_0^* U^* \operatorname{diag}(\lambda_1 p(\lambda_1)^2, \dots, \lambda_m p(\lambda_m)^2) U e_0$, $C = e_0^* U^* \operatorname{diag}(\lambda_1, \dots, \lambda_m) U e_0$

und $B/C \leq \max_i |\lambda_i p(\lambda_i)^2 / \lambda_i|$. □

Kor: Hat A nur n verschiedene Eigenwerte λ_i, liefert das CG-Verfahren nach n Schritten die Lösung $x_n = A^{-1}b$.

Bew: $p(\lambda) = \frac{(\lambda_1 - \lambda) \cdots (\lambda_n - \lambda)}{\lambda_1 \cdots \lambda_n} \in \tilde{P}_n$ erfüllt $p(\lambda_i) = 0 \ \forall i$

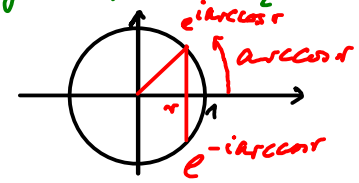
$\Rightarrow \|e_n\|_A / \|e_0\|_A \leq \max_{i=1,\dots,n} |p(\lambda_i)| = 0$. □

Def: $T_k(x) = \cos(k \arccos x)$, $x \in [-1,1]$, heißt Tschebyscheff-Polynom vom Grad k.

Bem: $T_k \in P_k$ (Übung)

- $|T_k(x)| \leq 1 \quad \forall x \in [-1,1]$.
- $T_k(x) = \frac{1}{2}(e^{ik \arccos x} + e^{-ik \arccos x})$
- Für die Variablentransformation $\mathbb{C} \to \mathbb{C}$, $x = \frac{z + z^{-1}}{2}$, gilt $T_k(x) = \frac{1}{2}(z^k + z^{-k})$.

In der Tat, für $r \in [-1,1]$ gilt $r = \frac{e^{i \arccos r} + (e^{i \arccos r})^{-1}}{2}$.



Sei $x \in [-1,1]$, $T_k(x) = \cos(k \arccos x) \in [-1,1]$

$\Rightarrow z = e^{i \arccos x}$, $e^{i \arccos T_k(x)} = e^{ik \arccos x} = z^k$

$\Rightarrow T_k(x) = \frac{1}{2}(z^k + z^{-k})$

Thm: $\|e_k\|_A / \|e_0\|_A \leq 2 \Big/ \left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-k}\right] \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k$ für $\kappa = \kappa(A)$.

Bew: Seien λ_1, λ_m größter und kleinster Eigen-/Singulärwert von A, d.h. $\kappa = \frac{\lambda_1}{\lambda_m}$

$p(\lambda) = T_k\left(1 - 2\frac{\lambda - \lambda_m}{\lambda_1 - \lambda_m}\right)$, $\tilde{p}(\lambda) = p(\lambda)/p(0)$ mit $p(0) = T_k\left(\frac{\lambda_1 + \lambda_m}{\lambda_1 - \lambda_m}\right) = T_k\left(\frac{\kappa+1}{\kappa-1}\right)$.

$\Rightarrow |\tilde{p}(\lambda_i)| = |p(\lambda_i)| / |p(0)| \leq 1 / \left|T_k\left(\frac{\kappa+1}{\kappa-1}\right)\right|$.

Sei nun $\frac{\kappa+1}{\kappa-1} = \frac{1}{2}(z + z^{-1})$, d.h. $z^2 - 2\frac{\kappa+1}{\kappa-1}z + 1 = 0$, d.h. $z = \frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}$ (Hausaufgabe)

$\Rightarrow T_k\left(\frac{\kappa+1}{\kappa-1}\right) = \frac{1}{2}\left[\left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^k + \left(\frac{\sqrt{\kappa}+1}{\sqrt{\kappa}-1}\right)^{-k}\right]$ □

Bem: Um die Konvergenz zu verbessern, ist wieder Vorkonditionierung nötig, d.h. $Ax = b$ wird umgeformt zu $K^*AK\,y = K^*b$ mit $x = Ky$, sodass $\kappa(K^*AK) \approx 1$, d.h. $KK^* \approx A^{-1}$.

```
m = 2000;
B = rand(m,m);
b = rand(m,1);

thres = .01;
A = eye(m,m) + triu(B.*(B<thres),1); A = (A+A')/2;
spy(A);
kappa = cond(A)

tic; [~,~,~,~,r] = pcg(A,b,0,20); toc
A = sparse(A);
tic; [x,~,~,~,r] = pcg(A,b,0,20); toc
semilogy(r,'.-','Linewidth',3,'Markersize',10); hold on;
semilogy(1:20,((sqrt(kappa)-1)/(sqrt(kappa)+1)).^(1:20),'r','Linewidth',3);
norm(A*x-b,2)

thres = .05;
A = eye(m,m) + triu(B.*(B<thres),1); A = sparse(A+A')/2;
cond(A)
tic; [~,~,~,~,r] = pcg(A,b,0,20); toc
semilogy(r,'.-','Linewidth',3,'Markersize',10); hold on;

thres = .1;
A = eye(m,m) + triu(B.*(B<thres),1); A = sparse(A+A')/2;
cond(A)
tic; [~,~,~,~,r] = pcg(A,b,0,20); toc
semilogy(r,'.-','Linewidth',3,'Markersize',10); hold on;

thres = .2;
A = eye(m,m) + triu(B.*(B<thres),1); A = sparse(A+A')/2;
cond(A)
tic; [~,~,~,~,r] = pcg(A,b,0,20); toc
semilogy(r,'.-','Linewidth',3,'Markersize',10); hold on;
```

Heute: · Obere Hessenberg-form

Nun betrachte $Ax=b$, $A\in\mathbb{K}^{n\times n}$ regulär, ohne Annahme $A^*=A$. Eine zu CG verwandte Idee ist, im k-ten Schritt das Residuum über $K^k(A,b)$ zu minimieren, $x_k = \operatorname{argmin}_{x\in K^k(A,b)} \|Ax-b\|_2$, die Grundlage von GMRES. Hierzu sei $K_k = \left(A^{k-1}b \,|\, \dots \,|\, Ab \,|\, b\right)$ die Matrix, deren Spalten $K^k(A,b)$ aufspannen, dann ist $x_k = K_k y_k$ mit $y_k = \operatorname{argmin}_y \|AK_k y - b\|_2$. Dieser Ansatz ist jedoch sehr instabil, da K_k sehr schlecht konditioniert ist (wir werden sehen, dass alle Spalten von K_k ungefähr gleich dem Eigenvektor zum größten Eigenwert sind). Besser wäre statt K_k eine Matrix, deren Spalten $K^k(A,b)$ mit Orthonormalvektoren aufspannen (wie bei CG bzgl. $(\cdot,\cdot)_A$) – hierzu brauchen wir etwas Vorbereitung.

Manchmal ist es wünschenswert, eine Matrix in eine obere Dreiecksmatrix zu transformieren, allerdings nicht wie bei der QR oder LU Zerlegung, sondern mit Ähnlichkeitstransformationen.

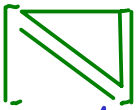
Erste Idee: Householder-Verfahren, nur nach jedem Schritt multipliziere Q_j auch von rechts:

$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}$ Householder $\longrightarrow$ $\begin{bmatrix} x & x & x & x \\ & x & x & x \\ & x & x & x \\ & x & x & x \end{bmatrix}$ $\xrightarrow{\cdot Q_1 \text{ für Ähnlichkeitstransformation}}$ $\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ & x & x & x \\ & x & x & x \end{bmatrix}$

A $\qquad$ $Q_1^* A$ $\qquad$ $Q_1^* A Q_1$

Leider werden die Nulleinträge wieder gefüllt. I. A. ist dies nicht auf direktem Weg möglich, aber man kann zumindest die obere Hessenbergform erreichen.

Def: (Hessenberg-Form) $H \in \mathbb{K}^{m \times n}$ hat <u>obere Hessenbergform</u>, wenn $H_{ij} = 0 \; \forall j < i-1$.



Zweite Idee: Wähle im k-ten Schritt eine Householder-Reflektion, die nur auf $A_{k+1:m,\,k+1:m}$ operiert, also die k-te Zeile in Ruhe lässt, $Q_k = \begin{pmatrix} I_{k \times k} & \\ & I - 2vv^* \end{pmatrix}$

$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix}$ Householder mit $v = \frac{\operatorname{sign}(x_1)\|x\|e_1 + x}{\| \cdots \|_2}$ $\longrightarrow$ $\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ & x & x & x \\ & x & x & x \end{bmatrix}$ $\xrightarrow{\cdot Q_1 \text{ für Ähnlichkeitstransformation}}$ $\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ & x & x & x \\ & x & x & x \end{bmatrix}$ usw.

x $\quad$ A $\qquad$ $Q_1^* A$ $\qquad$ $Q_1^* A Q_1$

Wir erhalten $\underbrace{Q_{m-2}^* \cdots Q_2^* Q_1^*}_{Q^*} A \underbrace{Q_1 Q_2 \cdots Q_{m-2}}_{Q} = H$ mit oberer Hessenbergform.

Alg: (Hessenbergform via Householder) input: $H = A \in \mathbb{K}^{m \times m}$; output: ähnliche Hessenbergmatrix H

for $k = 1$ bis $m-2$

$x = H_{k+1:m,\,k}$

$v_k = \operatorname{sign}(x_1)\|x\|_2 e_1 + x$

$v_k = v_k / \|v_k\|_2$

$H_{k+1:m,\,k:m} = H_{k+1:m,\,k:m} - 2 v_k (v_k^* H_{k+1:m,\,k:m})$

$H_{1:m,\,k+1:m} = H_{1:m,\,k+1:m} - 2 (H_{1:m,\,k+1:m} v_k) v_k^*$

end

Bem: · Dies ist nur eine Variation der QR-Zerlegung; Aufwand und Stabilität lassen sich auf gleiche Weise untersuchen.

· Wenn A hermitesch ist, ist dies auch $H \Rightarrow H$ ist tridiagonal.

Thm: Sei $A = QHQ^* \in \mathbb{K}^{m \times m}$ regulär für H obere Hessenbergmatrix, Q unitär mit Spalten $q_1, \ldots, q_m$.

Wenn $H_{i+1,i} \neq 0 \; \forall i$, dann ist $\langle q_1, \ldots, q_n \rangle = K^n(A, q_1) \; \forall n \leq m$.

$H_{i+1,i} = 0 \Rightarrow K^j(A, q_1) = K^i(A, q_1) \; \forall j \geq i$

Beweis: Induktion: · $\langle q_1 \rangle = K^1(A, q_1)$

· Sei $\langle q_1, \ldots, q_i \rangle = K^i(A, q_1)$; $AQ = QH \Rightarrow A q_i = H_{1i} q_1 + \ldots + H_{ii} q_i + H_{i+1,i} q_{i+1}$

$H_{i+1,i} \neq 0 \Rightarrow q_{i+1} \in \langle q_1, \ldots, q_i, A q_i \rangle \in K^{i+1}(A, q_1)$

$H_{i+1,i} = 0 \Rightarrow A q_i \in \langle q_1, \ldots, q_i \rangle \Rightarrow K^{i+1}(A, q_1) = \langle q_1, A K^i(A, q_1) \rangle \subset K^i(A, q_1)$ $\square$

Bem: Sei $b \in \mathbb{K}^m$, $\tilde{Q} = (b | \tilde{q}_2 | \dots | \tilde{q}_m)$ unitär, $\hat{Q} H \hat{Q}^* = \tilde{Q}^* A \tilde{Q}$ die Hessenbergreduktion von $\tilde{Q}^* A \tilde{Q}$ per Householder, $Q = \tilde{Q}\hat{Q} = (q_1 | \dots | q_m) = (b | q_2 | \dots | q_m)$. Es ist $q_1, \dots, q_n$ eine Orthonormalbasis von $K^n(A, b)$ (sofern nicht $K^{n-1}(A,b) = K^n(A,b)$).

Ein Nachteil des Householder-Verfahrens ist, dass bei der Berechnung einer Orthonormalbasis von $K^n(A,b)$ die gesamte Zerlegung $\hat{Q} H \hat{Q}^* = \tilde{Q}^* A \tilde{Q}$ berechnet wird. Die Alternative ist ein modifiziertes Gram-Schmidt-Verfahren.

Alg: (Arnoldi-Iteration) input: $A \in \mathbb{K}^{m \times m}$, $b \in \mathbb{K}^m$; output: $q_1, \dots, q_k$ orthonormal mit $\langle q_1, \dots, q_k \rangle = K^k(A,b)$

$q_1 = b / \|b\|_2$

for $k = 1, 2, \dots$

$v = A q_k$

for $j = 1$ bis k } Gram-Schmidt-Orthonormalisierung

$H_{jk} = q_j^* v$

$v = v - H_{jk} q_j$

end

$H_{k+1,k} = \|v\|$

$q_{k+1} = v / H_{k+1,k}$

end

Bem: Die ersten k Schritte liefern eine obere Hessenbergmatrix $\tilde{H}_k \in \mathbb{K}^{k+1,k}$ und $Q_i = (q_1 | \dots | q_i)$ mit $A Q_k = Q_{k+1} \tilde{H}_k$ und $H_k = (\tilde{H}_k)_{1:k,1:k} = Q_k^* A Q_k$.

Nun möchten wir $Ax = b$ iterativ lösen durch Minimieren des Residuums $\|Ax - b\|_2$ über $K^k(A,b)$

$$\Rightarrow \min_{x \in K^k(A,b)} \|Ax - b\|_2 = \min_y \|A Q_k y - b\|_2 = \min_y \|Q_{k+1} \tilde{H}_k y - b\|_2 = \min_y \|\tilde{H}_k y - Q_{k+1}^* b\|_2$$
$$= \min_y \|\tilde{H}_k y - \|b\|_2 e_1\|_2 .$$

Alg: (generalised minimal residual, GMRES) input: $A \in \mathbb{K}^{m \times m}$, $b \in \mathbb{K}^m$; output: Approximation x_k an $A^{-1}b$

Arnoldi-Iteration k zusätzlich am Ende der for-Schleife:

$y_k = \operatorname{argmin}_y \|\tilde{H}_k y - \|b\|_2 e_1\|_2$ mittels QR-Zerlegung (*)

$x_k = Q_k y_k$

Bem:
- Wegen der Hessenberg-Struktur von $\tilde{H}_k$ ist der Aufwand von (*) nur $O(k^2)$. Benutzt man statt Householder-Reflektionen Givens-Rotationen und verwendet die QR-Zerlegung von $\tilde{H}_{k-1}$, lässt sich dies zu $O(k)$ reduzieren (Übung).
- Ist A hermitesch, lassen sich symmetrische Operationen einsparen; dies führt zu MINRES.

Thm: Für GMRES gilt: $\|Ax_k - b\|_2 = \min_{\tilde{p} \in \tilde{P}_k} \|\tilde{p}(A) b\|_2$

Bew: $\|Ax_k - b\|_2 = \min_{x \in K^k(A,b)} \|Ax - b\|_2$

$\cdot\ x \in K^k(A,b) \Leftrightarrow x = p(A)b$ für ein $p \in P_{k-1} \Leftrightarrow Ax - b = \tilde{p}(A)b$ für ein $\tilde{p} \in \tilde{P}_k$ □

Thm: Wenn (A, b) zu $(\beta A, \beta b)$ geändert wird, $\beta \in \mathbb{K}$, ändert sich das GMRES Residuum r_k zu βr_k.

$\cdot$ — " — (UAU^*, Ub) — " —, U unitär, — " — $U^* r_k$.

$\cdot\ x_m = A^{-1}b$ für A regulär

$\cdot$ Ist A diagonalisierbar zu $\Lambda = X^{-1}AX$, dann ist $\|r_k\|_2 / \|b\|_2 \le \kappa(X) \min_{\tilde{p} \in \tilde{P}_k} \max_i |\tilde{p}(\lambda_i)|$

Bew: $\cdot$ Invarianz unter β bzw. U folgt aus Algorithmus

$\cdot\ x_m = \operatorname{argmin}_{x \in K^m(A,b) = \mathbb{K}^m} \|Ax - b\|_2$ und $K^m(A,b) = \mathbb{K}^m$

oder $b, Ab, \dots, A^{m-1}b$ sind lin. abh., d.h. $0 = \sum_{i=0}^{m-1} \alpha_i A^i b$ für $\alpha_i \in \mathbb{K}$;

sei $j \ge 0$ am kleinsten mit $\alpha_j \neq 0 \Rightarrow A^j b = -\sum_{i=j+1}^{m-1} \frac{\alpha_i}{\alpha_j} A^i b$

$\Rightarrow A^{-1}b = -\sum_{i=j+1}^{m-1} \frac{\alpha_i}{\alpha_j} A^{i-j-1} \in K^m(A,b)$

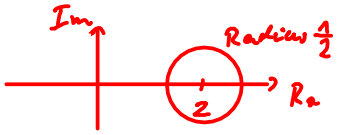
$\cdot\ \|r_k\|_2 / \|b\|_2 \le \min_{\tilde{p} \in \tilde{P}_k} \|\tilde{p}(A)\|_2 = \min_{\tilde{p} \in \tilde{P}_k} \|X \tilde{p}(\Lambda) X^{-1}\|_2 \le \|X\|_2 \|X^{-1}\|_2 \min_{\tilde{p} \in \tilde{P}_k} \|\tilde{p}(\Lambda)\|_2$ □

```
m = 200;
A = 2*eye(m) + 0.5*randn(m)/sqrt(m);
b = rand(m,1);

[~,D] = eig(A); plot(real(D),imag(D),'.','Markersize',10);

[x,~,~,~,r_b] = gmres(A,b,29,0,1);
semilogy(1:30,r_b,'b','Linewidth',3); hold on;
semilogy(1:30,1./4.^(1:30),'r','Linewidth',3)
```

⇒ Eigenwerte



⇒ optimales Polynom $\sim \tilde{p}(z) = \left(1 - \frac{z}{2}\right)^k$

⇒ $\|\tilde{p}(\Lambda)\|_2 \sim \left(\frac{1}{4}\right)^k$

Heute: · Bisektion

· Newton-Verfahren

Iterative Lösung nichtlinearer Gleichungssysteme

Neben der Banachschen Fixpunktiteration (z.B. angewandt auf $x = x + f(x)$) gibt es weitere Verfahren zum Finden einer Nullstelle einer Funktion $f: \mathbb{R}^m \to \mathbb{R}^m$.

Def: (Asymptotische Konvergenzrate) Sei $\lim_{k \to \infty} x_k = x^*$. Die Konvergenz heißt asymptotisch

· linear, falls $\|x_{k+1} - x^*\| \le q \|x_k - x^*\|$ für ein $q < 1$ & alle k groß genug

· sublinear sonst

· superlinear, falls für alle $q \in (0,1)$ gilt $\|x_{k+1} - x^*\| \le q \|x_k - x^*\|$ für k groß genug

· von Ordnung $p > 0$ falls $\|x_{k+1} - x^*\| \le C \|x_k - x^*\|^p$ für ein $C > 0$ & alle k groß genug

Bem: · „von Ordnung 2" heißt auch „quadratisch"

· Die Konvergenz $x_k \to x^*$ nennt man (super-/sub-)linear bzw. von Ordnung p, wenn $\|x_k - x^*\| \le \delta_k$ für eine entsprechend asymptotisch konvergente Nullfolge δ_k.

Bisektion

Sei $f : [a,b] \to \mathbb{R}$ stetig mit $f(a) < 0, f(b) > 0$. Nach dem Zwischenwertsatz gibt es $x^* \in [a,b]$ mit $f(x^*) = 0$.

Alg. (Bisektion/Intervallschachtelung) input: $f, a_0 = a, b_0 = b$; output: untere & obere Approximationen a_k & b_k an Nullstelle x^*

for $k = 0, 1, \dots$

wähle $c \in (a_k, b_k)$

if $f(c) < 0$: $a_{k+1} = c$, $b_{k+1} = b_k$

else : $a_{k+1} = a_k$, $b_{k+1} = c$

end

Bem: · Für die Wahl $c = \frac{a_k + b_k}{2}$ sind $|a_k - x^*|, |b_k - x^*| \le |b_k - a_k| = 2^{-k} |b_0 - a_0| \Rightarrow$ lineare Konvergenz

· Bessere Wahl durch lineare Approximation:

$$f(x) \approx \frac{(x-\alpha) f(\beta) + (\beta - x) f(\alpha)}{\beta - \alpha} \quad \Rightarrow \quad f(c) \approx 0 \text{ für } c = \frac{\alpha f(\beta) - \beta f(\alpha)}{f(\beta) - f(\alpha)}$$

wähle z.B. $\alpha = a_k$, $\beta = b_k$ oder $\alpha = a_{k-1}$, $\beta = a_k$ bzw. alterniere dazwischen

Newton-Verfahren

Sei $f : \mathbb{R}^m \to \mathbb{R}^m$ differenzierbar. Linearisieren um x_k liefert $f(x) \approx f(x_k) + Df(x_k)(x - x_k)$

$\Rightarrow f(x) \approx 0$ für $x = x_k - Df(x_k)^{-1} f(x_k)$

Alg: (Newton-Verfahren) input: $f : \mathbb{R}^m \to \mathbb{R}^m$, $x_0 \in \mathbb{R}^m$; output: Approximation x_k an Nullstelle

for $k = 0, 1, 2, \dots$

$x_{k+1} = x_k - Df(x_k)^{-1} f(x_k)$

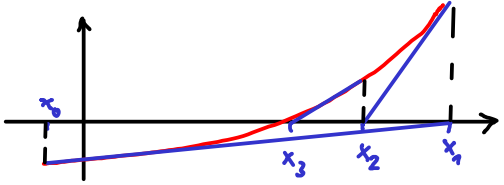
end

Bem: · Das Newtonverfahren kann aufgefasst werden als ein Liniensuchverfahren zur Minimierung von $g(x) := \|f(x)\|_2^2$ mit Suchrichtung $d_k = -Df(x_k)^{-1} f(x_k)$ (dies ist Abstiegsrichtung, da $Dg(x_k) d_k = -2 \|f(x_k)\|_2^2$) und Schrittweite $\alpha_k = 1$.

· Der Newtonschritt kann auch um eine Schrittweite α_k erweitert werden, $x_{k+1} = x_k - \alpha_k Df(x_k)^{-1} f(x_k)$, z.B. so dass $\|f(x_{k+1})\|^2$ sinkt.

· I.A. konvergiert das Verfahren nur, wenn man nah genug bei x^* startet.

In 1D ist $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$



Thm: Sei $f: \mathbb{R}^m \to \mathbb{R}^m$ differenzierbar mit Lipschitz-stetiger Ableitung und $f(x^*)=0$, $Df(x^*)$ invertierbar.
Für x_0 nah genug an x^* konvergiert das Newton-Verfahren quadratisch gegen x^*.

Bew:
- $Df(x)$ ist invertierbar auf einer Umgebung U von x^* und $Df(x)^{-1}$ ist stetig auf U
- Setze $g(x) = x - Df(x)^{-1} f(x)$.
- Sei L die Lipschitz-Konstante von Df, $x \in U$.

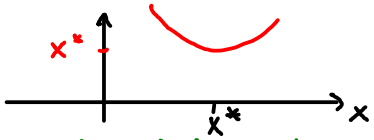
$$\begin{aligned}
\|g(x) - x^*\| &= \|Df(x)^{-1}(f(x) - f(x^*)) - x + x^*\| = \left\| Df(x)^{-1} \int_0^1 Df(x^* + t(x-x^*))(x-x^*)\,dt - x + x^* \right\| \\
&\le \|x - x^*\| \int_0^1 \| Df(x)^{-1}[Df(x) + Df(x^*+t(x-x^*)) - Df(x)] - I \|\,dt \\
&\le \|x - x^*\| \int_0^1 \|Df(x)^{-1}\| L(1-t)\|x - x^*\|\,dt \\
&\le C \|x - x^*\|^2
\end{aligned}$$

□

Bsp: Heron-Verfahren zur Berechnung von $x^* = \sqrt[n]{a}$ als Nullstelle von $f(x) = a - x^n$

$$\Rightarrow x_{k+1} = x_k + \frac{a - x_k^n}{n x_k^{n-1}} = \frac{n-1}{n} x_k + \frac{a}{n} x_k^{1-n} =: g(x_k)$$

$$g'(x) = \frac{n-1}{n}\left(1 - \frac{a}{x^n}\right)$$



$\Rightarrow$ globales Minimum bei x^*

$\Rightarrow g(x) \ge g(x^*)\ \forall x > 0 \quad \Rightarrow g(x_n) \ge x^*$

Für $x \ge x^*$ ist $|g'(x)| = \frac{n-1}{n}\left(1 - \frac{a}{x^n}\right) < \frac{n-1}{n}$

$\Rightarrow$ g ist Kontraktion auf $[x^*, \infty) \Rightarrow$ Newton-Verfahren konvergiert

Heute: · Eigenwerte

Eigenwertprobleme

Wiederholung Eigenwerte

Def: Sei $A \in \mathbb{C}^{m \times m}$. $\lambda \in \mathbb{C}$ heißt Eigenwert und $x \in \mathbb{C}^m$ Eigenvektor, wenn $Ax = \lambda x$, $x \neq 0$.
Die Menge $\Lambda(A)$ aller Eigenwerte heißt Spektrum von A.

Thm:
- Eigenwerte von A = Nullstellen des charakteristischen Polynoms $\chi_A(\lambda) = \det(\lambda I - A)$
- Die Eigenvektoren zum Eigenwert λ bilden mit 0 einen Untervektorraum $E_\lambda \subset \mathbb{C}^m$.

Def:
- geometrische Vielfachheit von λ = $\dim E_\lambda$
- algebraische Vielfachheit von λ = Vielfachheit der Nullstelle λ von χ_A

Thm:
- algebraische Vielfachheit $\ge$ geometrische Vielfachheit
- A hat m Eigenwerte, gezählt mit algebraischer Vielfachheit

Def: · $A, B \in \mathbb{C}^{m\times m}$ heißen *ähnlich*, wenn $A = XBX^{-1}$ für ein $X \in \mathbb{C}^{m\times m}$

· A heißt *diagonalisierbar*, wenn es ähnlich zu einer Diagonalmatrix Λ ist

· A heißt *unitär diagonalisierbar*, wenn $A = Q\Lambda Q^*$ für Λ diagonal, Q unitär

· A heißt *normal* wenn $A^*A = AA^*$

Thm: · Sei $\Lambda(A) = \{\lambda_1, \dots, \lambda_m\}$. $\det A = \prod_{i=1}^{m} \lambda_i$, $\operatorname{tr} A = \sum_{i=1}^{m} \lambda_i$

· ähnliche Matrizen haben gleiche Eigenwerte und Vielfachheiten

· A diagonalisierbar $\Leftrightarrow$ geometrische = algebraische Vielfachheit $\forall \lambda \in \Lambda(A)$

· A ist unitär diagonalisierbar $\Leftrightarrow$ A ist normal

Bsp: · (schief-) hermitesche Matrizen sind normal

· unitäre Matrizen sind normal

Def: $A = QTQ^*$ mit Q unitär, T obere Dreiecksmatrix, heißt *Schur-Zerlegung* von A

Thm: Jedes $A \in \mathbb{C}^{m\times m}$ besitzt eine Schur-Zerlegung

Bew: Induktion nach m; Induktionsanfang $m = 1$ klar.

Induktionsschritt: · Sei $m > 1$, x ein Eigenvektor von A mit Eigenwert λ

· wähle Orthonormalbasis $q_1 = \frac{x}{\|x\|}, q_2, \dots, q_m$; $U = [q_1 | \dots | q_m]$

· $U^*AU = \begin{bmatrix} \lambda & b^* \\ 0 & C \end{bmatrix}$ für $b \in \mathbb{C}^m$, $C \in \mathbb{C}^{m\times m}$

· nach Induktionsvoraussetzung $\exists$ Schur-Zerlegung $C = VTV^*$

· $A = Q\begin{bmatrix} \lambda & b^*V \\ 0 & T \end{bmatrix}Q^*$ für $Q = U\begin{bmatrix} 1 & 0 \\ 0 & V \end{bmatrix}$ ist Schur-Zerlegung. □

Bem: Die Eigenwerte einer Matrix A können gefunden werden als

· Nullstellen von χ_A (für kl. Matrizen instabil, ähnlich wie Nullstellensuche in Monom-Darstellung)

· Diagonaleinträge der Diagonalisierung (nur diagonalisierbare Matrizen)

· der Jordan-Normalform

· der Schur-Zerlegung (stabil, da unitäre Ähnlichkeitstransformationen genutzt werden)

Thm: (Abel) Für jedes $m \geq 5$ existiert ein Polynom $p(\lambda)$ mit Grad m, rationalen Koeffizienten und einer reellen Nullstelle r, die nicht geschrieben werden kann als ein Ausdruck aus rationalen Zahlen, $+, -, \cdot, /$ und $\sqrt[k]{\ }$.

Kor: Es gibt keine direkte Methode, die zu einer beliebigen Matrix A nach endlich vielen Schritten nur mit $+, -, \cdot, /, \sqrt[k]{\ }$ die Eigenwerte findet.

Bew: Zu beliebigem $p(\lambda) = \lambda^m + a_{m-1}\lambda^{m-1} + \dots + a_1\lambda + a_0$ ist $p = \chi_A$ für $A = \begin{bmatrix} 0 & & & -a_0 \\ 1 & \ddots & & -a_1 \\ & \ddots & 0 & \vdots \\ & & 1 & -a_{m-1} \end{bmatrix}$ □

Eigenwerte spielen in vielen physikalischen Problemen eine Rolle. Das einfachste Beispiel betrifft die Stabilität von dynamischen Systemen: Sei $\frac{d}{dt}x = F(x) \in \mathbb{R}^m$ eine gewöhnliche Differentialgleichung (die z.B. aus einer partiellen entsteht durch räumliche Diskretisierung) mit stationärem Zustand $\bar{x}$ (d.h. $F(\bar{x}) = 0$). In einer Umgebung von $\bar{x}$ kann man F durch Linearisierung approximieren, $\frac{dy}{dt} = \frac{dx}{dt} = F(x) \approx Ay$ für $y = x - \bar{x}$, $A = DF(\bar{x})$ mit Lösung $y(t) = \exp(tA) y(0) = X \operatorname{diag}(\exp t\lambda_i) X^{-1} y(0)$ für die Diagonalisierung $A = X \operatorname{diag}(\lambda_i) X^{-1}$. Sind alle $\operatorname{Re}(\lambda_i) < 0$, so ist $\lim_{t \to \infty} y(t) = 0$, d.h. $y = 0$ (bzw. $x = \bar{x}$) ist stabil. Für „>" gilt $\lim_{t \to \infty} \|y(t)\| = \infty$, wenn man nicht exakt mit $x = \bar{x}$ startet oder dies perturbiert $\Rightarrow \bar{x}$ ist instabil.

Schneechaos Münsterland 2005

pDgl: nichtlineare Elastizität

Knick-Instabilität

(Schummelei: Eigenwerte des linearisierten Systems sind nicht > 0, sondern maximal = 0)

Tacoma Narrows Bridge Collapse
https://www.youtube.com/watch?v=XggxeuFDaDU

lineare Elastizität
mit Anregung

Heute: · Vektoriteration

· Inverse Iteration

Iterative Verfahren zur Eigenwertbestimmung

Um Rechenoperationen einzusparen, wird eine Matrix $A \in \mathbb{C}^{m \times m}$ typischerweise erst in obere Hessenbergform transformiert, bevor iterativ die Eigenwerte und -vektoren bestimmt werden. Damit wir nur mit reellen Eigenwerten arbeiten müssen, wollen wir im Folgenden $A \in \mathbb{R}^{m \times m}$, $A^* = A$ annehmen.

Def: Der **Rayleigh-Quotient** eines Vektors $x \in \mathbb{R}^m$ ist $r(x) = \frac{x^* A x}{x^* x}$.

Bem: $Dr(x) = \frac{1}{x^* x}\left(x^* A + x^* A^T\right) - \frac{x^* A x}{(x^* x)^2} 2x^* = \frac{2}{x^* x}\left(x^* A - r(x) x^*\right)$

· Sei x Eigenvektor zum Eigenwert λ, dann ist $r(x) = \lambda$, $Dr(x) = 0$.

$\Rightarrow$ Taylorentwicklung um x liefert $r(\tilde{x}) - r(x) = O\left(\|x - \tilde{x}\|^2\right)$

$\Rightarrow$ Zu einer Schätzung des Eigenvektors liefert der Rayleigh-Quotient eine quadratisch genaue Schätzung des Eigenwerts

Eine Schätzung des Eigenvektors erhält man z.B. mit folgendem (selten genutztem) Algorithmus.

Alg: (Vektor-Iteration) input: $A \in \mathbb{R}^{m\times m}$, $v_0 \in \mathbb{R}^m$; output: Approximation v_k an Eigenvektor zu betragsmäßig größtem Eigenwert

for $k = 1, 2, \dots$

$w = A v_{k-1}$

$v_k = w / \|w\|_2$

end

Thm: A habe Eigenwerte $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_m|$ mit Eigenvektoren $q_1, \dots, q_m$, $\|q_i\|_2 = 1$, $q_1^* v_0 \neq 0$.
Für die Vektoriteration gilt $\| \pm v_k - q_1 \| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$, $|r(v_k) - \lambda_1| = O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^{2k}\right)$ (lineare Konvergenz).

Bew: · Sei $v_0 = a_1 q_1 + \dots + a_m q_m$, $c_k = \frac{1}{\|Av_0\|_2 \cdots \|Av_{k-1}\|_2}$.

· $v_k = c_k A^k v_0 = c_k \left(a_1 \lambda_1^k q_1 + \dots + a_m \lambda_m^k q_m\right) = c_k \lambda_1^k \left(a_1 q_1 + a_2 \left(\frac{\lambda_2}{\lambda_1}\right)^k q_2 + \dots + a_m \left(\frac{\lambda_m}{\lambda_1}\right)^k q_m\right)$

$= \pm q_1 + O\left(\left|\frac{\lambda_2}{\lambda_1}\right|^k\right)$

· $|r(v_k) - \lambda_1| = O\left(\|\pm v_k - q_1\|^2\right)$ □

Vektoriteration findet nur den größten Eigenwert/-vektor. Da $(A - \mu I)^{-1}$ für $\mu \in \mathbb{R}$ die gleichen Eigenvektoren hat wie A, können wir auch Vektoriteration auf $(A-\mu I)^{-1}$ anwenden und erhalten den Eigenvektor zum betragsmäßig größten Eigenwert $(\lambda_i - \mu)^{-1}$ von $(A-\mu I)^{-1}$, also einen Eigenvektor von A zum Eigenwert λ_i am nächsten bei μ. Dies ist der Standard-Algorithmus zum Finden eines Eigenvektors für gegebenen Eigenwert μ.

Alg: (Inverse Iteration) input: A, μ, v_0; output: Approximation v_k an Eigenvektor von A zu Eigenwert am nächsten bei μ

for $k = 1, 2, \dots$

Löse $(A - \mu I) w = v_{k-1}$ $\quad (*)$

$v_k = w / \|w\|$

end

$O(m)$ für A tridiagonal

Bem: · Man kann $A - \mu I$ vorher LU- oder QR-zerlegen $\Rightarrow$ Aufwand $O(m^2)$ pro Iteration

· Für μ nahe einem Eigenwert ist $(*)$ sehr schlecht konditioniert, dies beeinflusst jedoch nicht die Stabilität des Algorithmus (Übung)

Thm: Seien λ_j & λ_i Eigenwerte von A am nächsten/zweitnächsten zu μ, $q_j^* v_0 \neq 0$. Für die Inverse Iteration gilt $\| \pm v_k - q_j \| = O\left(\left|\frac{\mu - \lambda_i}{\mu - \lambda_j}\right|^k\right)$, $|r(v_k) - \lambda_j| = O\left(\left|\frac{\mu - \lambda_i}{\mu - \lambda_j}\right|^{2k}\right)$ (lineare Konvergenz)

Bew: Inverse Iteration = Vektoriteration für $(A-\mu I)^{-1}$ mit betragsmäßig größtem/zweitgrößtem Eigenwert $(\lambda_j - \mu)^{-1}$ / $(\lambda_i - \mu)^{-1}$. □

Ein Eigenwert/-vektor-Paar erhält man durch Alternieren zwischen Rayleigh Quotient & Inverser Iteration:

Alg: (Rayleigh-Quotienten-Iteration) input: A, λ^0, v_0 ; output: Approximation λ^k, v_k an Eigenwert/-vektor-Paar von A

for $k = 1, 2, \ldots$

Löse $(A - \lambda^{k-1} I) w = v_{k-1}$

$v_k = w / \|w\|$

$\lambda^k = v_k^* A v_k$

end

Thm: Wenn die Iteration gegen ein Eigenwert/-vektor-Paar (λ, q) konvergiert (dies passiert für fast alle v_0), dann kubisch: $\|\pm q - v_k\| = O(\|\pm q - v_{k-1}\|^3)$, $|\lambda_k - \lambda| = O(|\lambda_{k-1} - \lambda|^3)$.

Bew: · Sei $\|\pm q - v_k\| \le \varepsilon \Rightarrow |\lambda - \lambda_k| = C\varepsilon^2$ für ein C (von Rayleigh-Quotienten-Funktion r abhängig) $\Rightarrow \|\pm q - v_{k+1}\| = O\left(\left|\frac{\lambda - \lambda_k}{\tilde{\lambda} - \lambda_k}\right| \|\pm q - v_k\|\right) = O(\varepsilon^3)$ für $\tilde{\lambda}$ zweitnächster Eigenwert an λ_k

· Analog für Eigenwert □

Bem: · Wenn A tridiagonal (schon in Hessenbergform transformiert), benötigt jede Iteration $O(n)$ Flops ; sei e_k der Fehler der $k$ten Iteration $\Rightarrow e_k = O(e_{k-1}^3) = O(e_0^{3^k})$

$\Rightarrow \varepsilon_m$ Genauigkeit erreicht nach ca. $\log_3\left(\frac{\log \varepsilon_m}{\log e_0}\right) \ll m$ Iterationen.

```
format long;
m = 100;
A = rand(m,m); A = A + A';
w = rand(m,1);
v = w/norm(w,2);

for k = 1:6
  lambda = v'*A*v
  w = (A-lambda*eye(m))\v;
  v = w/norm(w,2);
end
```

kubische Konvergenz!

Heute: QR-Iteration

Eine Variante der Vektoriteration liefert Eigenvektoren zu den n betragsmäßig größten Eigenwerten:

Alg: (Simultane Vektoriteration) input: $A \in \mathbb{R}^{m \times m}$, $V_0 \in \mathbb{R}^{m \times n}$; output: Approximation $Q_k = [q_1^k | \ldots | q_n^k]$ an Eigenvektoren

for $k = 1, 2, \ldots$

$V_k = A V_{k-1}$

berechne QR-Zerlegung $Q_k R_k = V_k$

end

Thm: Für die Eigenwerte von A gelte $|\lambda_1| > \dots > |\lambda_n| > |\lambda_{n+1}| \geq \dots \geq |\lambda_m|$, die orthonormalen Eigenvektoren seien $q_1, \dots, q_m$. Seien $Q = [q_1 | \dots | q_m]$ und V_0 sodass $(Q^* V_0)_{1:i,1:i}$ regulär ist für alle $i \leq n$. Es gilt $\| \pm q_j^k - q_j \| = O(C^k)$ für $C = \max_{i \leq n} |\lambda_{i+1}| / |\lambda_i|$.

Bew: · Sei $U_i = [q_1 | \dots | q_i]$, $\Lambda_i = \operatorname{diag}(\lambda_1, \dots, \lambda_i)$.

· $V_k = A^k V_0 = Q \Lambda_m^k Q^* V_0 = U_i \Lambda_i^k U_i^* V_0 + O(|\lambda_{i+1}|^k) = (U_i \Lambda_i^k + O(|\lambda_{i+1}|^k)) \, U_i^* V_0$

($U_i^* V_0 = (Q^* V_0)_{1:i,1:i}$ regulär)

$\Rightarrow \langle q_1^k, \dots, q_i^k \rangle = \langle \underbrace{v_1^k, \dots, v_i^k}_{\text{Spalten von } V_k} \rangle = \langle \underbrace{\lambda_1 q_1 + O(|\lambda_{i+1}|^k), \dots, \lambda_i q_i + O(|\lambda_{i+1}|^k)}_{\text{Spalten von } U_i \Lambda_i^k + O(|\lambda_{i+1}|^k)} \rangle = \langle q_1 + O(|\tfrac{\lambda_{i+1}}{\lambda_1}|^k), \dots, q_i + O(|\tfrac{\lambda_{i+1}}{\lambda_i}|^k) \rangle$

· da dies für alle $i \leq n$ gilt, ist $\pm q_i^k = q_i + O(C^k)$ □

Von der Vektoriteration wissen wir bereits $v_i^k = A^k v_i^0 \approx c \, q_1$ für $k \to \infty$ $\Rightarrow$ Rundungsfehler nivellieren den Unterschied zwischen den v_i^k $\Rightarrow$ simultane Vektoriteration ist nicht stabil.

Damit $\langle v_1^k, \dots, v_i^k \rangle$ für alle i stabil konvergiert, amplifizieren wir in jeder Iteration die Unterschiede, ohne den Spann zu verändern: durch Orthonormalisierung.

Alg: (Simultane Vektoriteration II bzw. QR-Iteration I) input: $A \in \mathbb{R}^{m \times m}$, $V_0 \in \mathbb{R}^{m \times n}$ mit orthonormalen Spalten

for $k = 1, 2, \dots$ output: Approximation $V_k = [q_1^k | \dots | q_n^k]$ an Eigenvektoren

$\tilde{V}_k = A V_{k-1}$ (1)

berechne reduzierte QR-Zerlegung $V_k \tilde{R}_k = \tilde{V}_k$ (2)

end

Bem: Die ersten i Spalten von V_k haben den gleichen Spann wie von $\tilde{V}_k$. $\Rightarrow$ Die ersten i Spalten von V_k haben den gleichen Spann wie in der simultanen Vektoriteration I. $\Rightarrow$ Es gilt das gleiche Konvergenz-Theorem.

Die QR-Iteration ist eine äquivalente Umformulierung. Sie geht ähnlich vor wie die Hessenberg-Transformation mittels Householder: Dort wird in jedem Schritt $A = QH$ zerlegt (wobei H viele Einträge eliminiert hat) und dann $HQ = Q^* A Q$ berechnet. Bei der QR-Iteration werden in A ebenfalls viele Einträge eliminiert (alle unter der Diagonalen) mittels QR-Zerlegung $A = QR$, und dann $RQ = Q^* A Q$ berechnet. Hierdurch werden die Nulleinträge wieder gefüllt, aber nach und nach konvergiert die Matrix gegen eine Diagonalmatrix.

Alg: (QR-Iteration II) input: $A_0 = A$, output: Approximation an Spektralzerlegung

for $k = 1, 2, \dots$

$Q_k R_k = A_{k-1}$ (3)

$A_k = R_k Q_k$ (4)

end

Thm: · QR-Iteration II $\Leftrightarrow$ QR-Iteration I mit $V_0 = I$; insbesondere ist $V_k = Q_1 \cdots Q_k$, $R_k = \tilde{R}_k$

· Es ist $A^k = V_k R^{(k)}$ mit $R^{(k)} = R_k \cdots R_1$ und $A_k = V_k^* A V_k$.

Bew: Induktionsanfang: · $V_1 \tilde{R}_1 = A = Q_1 R_1$ sind gleiche QR-Zerlegung $\Rightarrow V_1 = Q_1$, $R_1 = \tilde{R}_1$

· außerdem $A^1 = Q_1 R_1 = V_1 R^{(1)}$ & $A_1 = R_1 Q_1 = Q_1^* A Q_1 = V_1^* A V_1$

Induktionsschritt: · $V_{k+1} \tilde{R}_{k+1} \overset{(2)}{=} \tilde{V}_{k+1} \overset{(1)}{=} A V_k \overset{IV}{=} A V_{k-1} Q_k \overset{(1)}{=} \tilde{V}_k Q_k \overset{(2)}{=} V_k \tilde{R}_k Q_k \overset{IV \& (4)}{=} V_k A_k \overset{(3)}{=} V_k Q_{k+1} R_{k+1}$

ist gleiche QR-Zerlegung $\Rightarrow \tilde{R}_{k+1} = R_{k+1}$, $V_{k+1} = V_k Q_{k+1}$

· $A^{k+1} \overset{IV}{=} A V_k R^{(k)} \overset{(1)}{=} \tilde{V}_{k+1} R^{(k)} \overset{(2)}{=} V_{k+1} \tilde{R}_{k+1} R^{(k)} = V_{k+1} R^{(k+1)}$

· $A_{k+1} \overset{(4)}{=} R_{k+1} Q_{k+1} \overset{(3)}{=} Q_{k+1}^* A_k Q_{k+1} \overset{IV}{=} Q_{k+1}^* V_k^* A V_k Q_{k+1} = V_{k+1}^* A V_{k+1}$ □

Bem: · Da die Spalten von V_k für $k \to \infty$ gegen die Eigenvektoren konvergieren, konvergiert $A = V_k A_k V_k^* = (Q_1 \cdots Q_k) A_k (Q_1 \cdots Q_k)^*$ gegen die Spektralzerlegung von A.

· Die Spalten v_i^k von V_k konvergieren gegen $\pm q_i$ mit einem Fehler $O\left(\left(\max_j \left|\frac{\lambda_j}{\lambda_{j+1}}\right|\right)^k\right)$, die Diagonaleinträge von A_k sind die Rayleigh-Quotienten zu den v_i^k und konvergieren somit mit der doppelten Potenz gegen die Eigenwerte λ_i.

Bem: · Ist in der simultanen Vektoriteration $V_0 \neq I$, so liefert der gleiche Beweis $A^k V_0 = V_k R^{(k)}$, d.h. die k-te Iteration berechnet die QR-Zerlegung von $A^k V_0$.

· Die QR-Iteration kann auch aufgefasst werden als simultane Vektoriteration II angewandt auf A^{-1} (also als „simultane inverse Iteration") mit Anfangsmatrix $V_0 = P = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}$. Tatsächlich ist $A^{-k} P \underset{A=A^*}{=} (A^k)^{-*} P = (V_k R^{(k)})^{-*} P = V_k R^{(k)-*} P = \underbrace{(V_k P)}_{\text{orthogonal}} \underbrace{(P R^{(k)-*} P)}_{\text{obere } \Delta\text{-Matrix}}$ die QR-Zerlegung von $(A^{-1})^k V_0$.

Die Inverse Iteration kann durch Shifts μI verbessert werden, und die richtige Wahl der Shifts liefert sogar kubische Konvergenz. Da die QR-Iteration auch aufgefasst werden kann als Inverse Iteration, gilt dies auch für die QR-Iteration.

Alg: (QR-Iteration mit Shifts) input, output wie zuvor

for $k = 1, 2, \ldots$

wähle Shift μ_k

$Q_k R_k = A_{k-1} - \mu_k I$ (5)

$A_k = R_k Q_k + \mu_k I$ (6)

falls $(A_k)_{1:i, i+1:m}$ nahe genug bei 0, setze $A^{(1)} = (A_k)_{1:i, 1:i}$, $A^{(2)} = (A_k)_{i+1:m, i+1:m}$, und wende von nun an QR-Iteration auf $A^{(1)}$ & $A^{(2)}$ separat an

end

Bem: · Typischerweise wird A zuvor in Tridiagonalform überführt $\Rightarrow A_k$ ist tridiagonal
$\Rightarrow$ Aufwand jeder Iteration ist $O(m)$

· Als Shift μ_k können verschiedene Approximationen an einen Eigenwert genutzt werden (z.B. ein Rayleigh-Quotient oder $(A_{k-1})_{mm}$)

· Ein Shift nahe dem $i$ten Eigenwert erzeugt Nullen in der $i$ten Zeile; für die resultierenden $A^{(1)}$ und $A^{(2)}$ möchte man nun verschiedene Shifts anwenden
$\Rightarrow$ führe für beide separate QR-Iteration durch.

Thm: Es ist $A_k = V_k^* A V_k$ und $(A-\mu_k I)\cdots(A-\mu_2 I)(A-\mu_1 I) = V_k R^{(k)}$ mit $V_k = Q_1 \cdots Q_k$, $R^{(k)} = R_k \cdots R_1$

Bew: Wie zuvor per Induktion.

Anfang: $A_1 \overset{(6)}{=} R_1 Q_1 + \mu_1 I \overset{(5)}{=} Q_1^*(A_0 - \mu_1 I) Q_1 + \mu_1 I = V_1^* A V_1$ & $A - \mu_1 I \overset{(5)}{=} Q_1 R_1 = V_1 R^{(1)}$

Schritt: · $A_k \overset{(6)}{=} R_k Q_k + \mu_k I \overset{(5)}{=} Q_k^*(A_{k-1} - \mu_k I) Q_k + \mu_k I \overset{IV}{=} Q_k^* V_{k-1}^* A V_{k-1} Q_k = V_k^* A V_k$

· $(A-\mu_k I)\cdots(A-\mu_1 I) \overset{IV}{=} (A-\mu_k I) V_{k-1} R^{(k-1)} \overset{IV}{=} V_{k-1} \underbrace{(A_{k-1} - \mu_k I)}_{Q_k R_k} \underbrace{V_{k-1}^* V_{k-1}}_{I} R^{(k-1)} \overset{(5)}{=} V_k R^{(k)}$ □

Bem: Wie die Version ohne Shifts kann die QR-Iteration mit konstantem Shift μ aufgefasst werden als simultane Vektoriteration II angewandt auf $(A-\mu I)^{-1}$ mit $V_0 = P = \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix}$, wobei sich ein V_k ergibt mit $V_k = Q_1 \cdots Q_k P$ (siehe oben). Insbesondere ist die erste Spalte von V_k (= letzte Spalte von $Q_1 \cdots Q_k$) gleich der Eigenvektor-Approximation v_k der Vektoriteration zu $(A-\mu I)^{-1}$ (= Inverse Iteration zu A mit Shift μ) mit Anfangsvektor $v_0 = e_m = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$.

Durch neue Wahl des Shifts in jeder Iteration als eine Eigenwertapproximation zur Eigenvektorapproximation v_k wurde aus der Inversen Iteration die kubisch konvergente Rayleigh-Iteration
$\Rightarrow$ Wähle hier als Shift eine Eigenwertapproximation passend zur letzten Spalte z_k von $Q_1 \cdots Q_k$, z.B. den Rayleigh-Quotienten.

Def: In der QR-Iteration mit Shift sei $(A_{k-1})_{m-1:m,\, m-1:m} = \begin{bmatrix} a_{m-1} & b_{m-1} \\ b_{m-1} & a_m \end{bmatrix}$. Die Wahl

· $\mu_k = a_m$ ist der Rayleigh-Shift

· $\mu_k = a_m - \operatorname{sign}(\delta)\, b_{m-1}^2 \,/\, \left(|\delta| + \sqrt{\delta^2 + b_{m-1}^2}\right)$ mit $\delta = \frac{a_{m-1} - a_m}{2}$ ist der Wilkinson-Shift

Bem: · Der Rayleigh-Shift μ_{k+1} ist der Rayleigh-Quotient von $z_k = Q_1 \cdots Q_k e_m$:

$(A_k)_{mm} = e_m^* A_k e_m = e_m^* (Q_1 \cdots Q_k)^* A\, Q_1 \cdots Q_k e_m = \frac{z_k^* A z_k}{z_k^* z_k}$

· Der Wilkinson-Shift ist der Eigenwert von $(A_{k-1})_{m-1:m,\, m-1:m}$ näher bei $(A_k)_{mm}$; die angegebene Formel ermöglicht eine stabile Berechnung; bei $A \neq A^*$ liefert dies komplex konjugierte Eigenwertpaare

· Beide liefern normalerweise kubische Konvergenz; nur der Wilkinson-Shift sichert Konvergenz

Bsp: $A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ mit Eigenwerten ± 1

- QR-Iteration ohne Shift: $A_0 = Q_1 R_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $A_1 = R_1 Q_1 = A_1$, $A_2 = A$ usw.
- mit Rayleigh-Shift: Shift ist $(A_0)_{22} = 0 \Rightarrow$ gleiche Iteration „unentschieden zwischen ± 1"
- mit Wilkinson-Shift: $\mu_1 = -1$; $A_0 - \mu_1 I = Q_1 R_1 = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\frac{\sqrt{2}}{2}\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$

$$A_1 = R_1 Q_1 + \mu_1 I = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix} - \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

$\Rightarrow$ konvergiert nach einem Schritt, $A = Q_1 A_1 Q_1^*$

Heute: Bisektionsverfahren

Im Prinzip können Eigenwerte auch durch Finden der Nullstellen des charakteristischen Polynoms χ_A mittels Bisektion (oder einem anderen Verfahren) gefunden werden. Natürlich wird χ_A hierfür nicht in Monomdarstellung ausgerechnet – dann wäre das Verfahren instabil.

Stattdessen werden ein paar Tricks genutzt.

Thm: Sei $A = \begin{bmatrix} a_1 & b_1 & & \\ b_1 & \ddots & \ddots & \\ & \ddots & \ddots & b_{m-1} \\ & & b_{m-1} & a_m \end{bmatrix} \in \mathbb{R}^{m\times m}$, $b_i \neq 0\ \forall i$ und $A^{(k)} = A_{1:k,1:k}$.

- Die Eigenwerte $\lambda_1^{(k)}, \dots, \lambda_k^{(k)}$ von $A^{(k)}$ sind paarweise verschieden für $k \in \{1, \dots, m\}$. (Übung)
- $\lambda_j^{(k+1)} < \lambda_j^{(k)} < \lambda_{j+1}^{(k+1)} \quad \forall\, j, k$

Bem: Wenn $b_i = 0$, können separat die Eigenwerte von $A_{1:i,1:i}$ & $A_{i+1:m,i+1:m}$ berechnet werden.

Thm:

- $\det(A^{(k)}) = a_k \det(A^{(k-1)}) - b_{k-1}^2 \det(A^{(k-2)})$ (alte Übung)
- Anzahl ℓ^m negativer Eigenwerte von A = Anzahl z^m Vorzeichenwechsel in Folge $1, \det A^{(1)}, \det A^{(2)}, \dots, \det A^{(m)}$

Bew: Induktion in m; Fall $m=1$ trivial. Induktionsschritt:

Fall $\lambda_1^{(m+1)} < \lambda_1^{(m)} < \lambda_2^{(m+1)} < \lambda_2^{(m)} < \dots < \lambda_k^{(m+1)} < 0 < \lambda_k^{(m)} < \dots < \lambda_m^{(m)} < \lambda_{m+1}^{(m+1)}$:

$$\operatorname{sign}(\det A^{(m+1)}) = (-1)^{\ell^{m+1}} \neq (-1)^{\ell^m} = \operatorname{sign}(\det A^{(m)}),\ \text{d.h. } z^{m+1} = z^m + 1 = \ell^m + 1 = \ell^{m+1}$$

Fall $\lambda_k^{(m+1)} < \lambda_k^{(m)} < 0 < \lambda_{k+1}^{(m+1)}$ analog. $\square$

Kor:

- $p^{(k)}(x) = \det(A^{(k)} - xI)$ erfüllt $p^{(k)}(x) = (a_k - x)p^{(k-1)}(x) - b_{k-1}^2 p^{(k-2)}(x)$
- Anzahl Eigenwerte zwischen $[a, b) = z^m(A - bI) - z^m(A - aI) = t^m(b) - t^m(a)$

für $t^m(x)$ = Anzahl Vorzeichenwechsel in $1, p^{(1)}(x), p^{(2)}(x), \dots, p^{(m)}(x)$

Alg: (Bisektionsverfahren) input: a_0, b_0 mit $t^m(a_0) + 1 = t^m(b_0)$

for $k = 1, 2, \dots$

$$c = \frac{a_{k-1} + b_{k-1}}{2}$$

if $t^m(c) = t^m(a_{k-1})$, $a_k = c$, $b_k = b_{k-1}$

else $a_k = a_{k-1}$, $b_k = c$

end

Bem: · $O(m)$ Aufwand pro Iteration $\Rightarrow O(m|\log \varepsilon_m|)$ Aufwand zum Finden eines Eigenwerts

· Eigenvektor am Schluss per 1 Schritt Inverse Iteration

· Tridiagonalisierung + Bisektionsverfahren = Standard zum Finden aller Eigenwerte in einem Intervall oder der 20 kleinsten o. Ä.

Heute: · Divide & Conquer - Verfahren

· Jacobi - Verfahren

Divide & Conquer - Ansatz: Divide - teile Problem in kleine, gleichartige Teilprobleme auf

Conquer - löse Teilprobleme (rekursiv auf gleiche Weise)

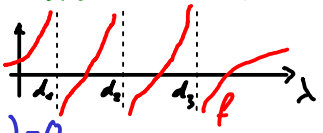
Sei $A \in \mathbb{R}^{m\times m}$ tridiagonal, $A_{i,i+1} \neq 0 \; \forall i$, $A^* = A$. Schreibe $A = \left[\begin{array}{c|c} A_{1:\frac{m}{2},1:\frac{m}{2}} & \beta \\ \hline \beta & A_{\frac{m}{2}+1:m,\frac{m}{2}+1:m} \end{array}\right] = \left[\begin{array}{c|c} \hat{A}_1 & \\ \hline & \hat{A}_2 \end{array}\right] + \left[\begin{array}{c|c} \beta & \beta \\ \hline \beta & \beta \end{array}\right]$.

Seien $\hat{A}_1 = Q_1 D_1 Q_1^*$, $\hat{A}_2 = Q_2 D_2 Q_2^*$, q_1 = letzte Zeile von Q_1, q_2 = erste Zeile von Q_2. Dann ist

$$A = \begin{pmatrix} \hat{A}_1 & \\ & \hat{A}_2 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix} (0 \dots 0\,1\,1\,0 \dots) = \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} \left(\begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix} + \beta z z^* \right) \begin{pmatrix} Q_1^* & \\ & Q_2^* \end{pmatrix} \text{ für } z = \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}.$$

Was sind Eigenwerte von $\begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix} + \beta z z^*$? Wir dürfen $z_i \neq 0 \; \forall i$ annehmen (sonst kann man das Problem reduzieren (Hausaufgabe)) und $D_{ii} \neq D_{jj} \; \forall i \neq j$, da $A_{i,i+1} \neq 0 \; \forall i$.

Thm: Sei $D \in \mathbb{R}^{m\times m}$ diagonal mit paarweise verschiedenen Einträgen, $w \in \mathbb{R}^m$ mit $w_i \neq 0 \; \forall i$. $D + ww^*$ habe paarweise verschiedene Eigenwerte. Diese sind die Nullstellen von $f(\lambda) = 1 + \sum_{j=1}^{m} \frac{w_j^2}{d_j - \lambda}$



Bew: $(D + ww^*)q = \lambda q \iff (D - \lambda I)q + w(w^*q) = 0 \iff q + (D-\lambda I)^{-1} w (w^* q) = 0$

$\Rightarrow w^*q + w^*(D-\lambda I)^{-1} w (w^*q) = 0 \iff f(\lambda) \cdot w^* q = 0$

· $w^*q \neq 0$ (also $f(\lambda) = 0$), denn sonst wäre λ Eigenwert von $D \Rightarrow q_i \neq 0$ für genau ein $i \Rightarrow w^* q \neq 0$

· f hat m Nullstellen und $(D + ww^*)$ hat m verschiedene Eigenwerte $\Rightarrow$ Nullstellen = Eigenwerte □

Bem: Analoges gilt für $D - ww^*$.

· $f(\lambda) = 0$ heißt „secular equation" und kann für jeden Eigenwert mit Newton-Verfahren in $O(1)$ Schritten gelöst werden (bzw. ähnlichen Verfahren)

Alg: (Divide & Conquer - Verfahren)

```
function [Q, D] = dac(A)
  if A ∈ ℝ then Q = 1, D = A
  else
      A = (Â₁  Â₂) + β(  ⋱  )
      [Q₁, D₁] = dac(Â₁),  [Q₂, D₂] = dac(Â₂),  D̃ = (D₁  D₂)
```

for $k = 1$ bis $m = \dim(A)$

D_{kk} = Lösung von $f(\lambda) = 0$ mit Startwert $\tilde{D}_{kk}$ (1)

q_k = ein Schritt Inverse Iteration mit Shift D_{kk} (2)

end

$Q = [q_1 | \dots | q_m]$

Bem: · Vergleichbar gut wie QR-Iteration, aber numerisch stabile Implementierung kompliziert

· Aufwand: (1) $\hat{=}$ $O(m)$ ($O(1)$ Iterationen, $O(m)$ flops für Auswertung f)

(2) $\hat{=}$ $O(m)$ (da Matrix tridiagonal)

$\Rightarrow$ for-Schleife $\hat{=}$ $O(m^2)$

Gesamtaufwand: $O(m^2 + 2(\frac{m}{2})^2 + 4(\frac{m}{4})^2 + \dots + m(\frac{m}{m})^2) = O(m^2 \sum_{i=0}^{\log m} 2^{-i}) = O(m^2)$

Das QR-Verfahren (ohne Shifts) berechnet in jeder Iteration eine QR-Zerlegung $A = QR$ und multipliziert dann Q noch einmal von rechts: $A_{neu} = RQ = Q^* A Q$. Wird die QR-Zerlegung mit Householder-Verfahren gemacht, so ist $Q = Q_1 \cdots Q_{m-1}$ mit Q_i der Householder-Reflektion, die die $i$te Spalte eliminiert. D.h., man kann das QR-Verfahren folgendermaßen auffassen:

$A_0 = A \xrightarrow{Q_1^* \cdot \text{ eliminiere 1. Spalte}} Q_1^* A_0 \xrightarrow{\cdot Q_1 \text{ für Ähnlichkeit}} Q_1^* A_0 Q_1 \xrightarrow{Q_2^* \cdot \text{ eliminiere 2. Spalte}} Q_2^* Q_1^* A_0 Q_1 \xrightarrow{\cdot Q_2} Q_2^* Q_1^* A_0 Q_1 Q_2$

$\dots \longrightarrow A_1 = Q_3^* Q_2^* Q_1^* A_0 Q_1 Q_2 Q_3 \longrightarrow$ beginne wieder von vorn

$\Rightarrow$ in jeder unitären Ähnlichkeitstransformation $B \mapsto Q_i^* B Q_i$ werden die neuen Nulleinträge wieder gefüllt. Wir wissen: die Nulleinträge können nicht bestehen bleiben, sonst gäbe es ein direktes Eigenwertbestimmungsverfahren; allerdings werden die neuen Einträge immer kleiner. Versucht man jedoch, nicht eine ganze Spalte zu eliminieren, sondern z.B. nur einen Eintrag, so kann dies durch eine Ähnlichkeitstransformation gemacht werden:

$$A = \begin{bmatrix} \ddots & \vdots & \\ \cdots & a & \cdots & d & \cdots \\ & \vdots & & \vdots & \\ \cdots & d & \cdots & b & \cdots \\ & \vdots & & \vdots & \ddots \end{bmatrix} \begin{matrix} \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \end{matrix}, \quad Q_{ij} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & c & & s \\ & & & \ddots & \\ & & -s & & c \end{bmatrix} \text{ mit } \begin{matrix} c = \cos\theta \\ s = \sin\theta \\ \theta = (\arctan \frac{2d}{b-a})/2 \end{matrix} \Rightarrow Q_{ij}^* A Q_{ij} = \begin{bmatrix} \ddots & \vdots & \\ \cdots & \cdots & 0 \\ & \vdots & \\ \cdots & 0 & \cdots \\ & & \ddots \end{bmatrix}$$

d.h. die $\{i,j\} \times \{i,j\}$ Untermatrix $\begin{pmatrix} a & d \\ d & e \end{pmatrix}$ wird diagonalisiert (dies wäre im Prinzip für bis zu 4×4 Untermatrizen möglich, darüber gibt es keine direkte Diagonalisierungsformel). Das Jacobi-Verfahren iteriert diesen Ansatz.

Alg: (Jacobi-Verfahren) input: $A_0 = A \in \mathbb{R}^{m\times m}$ symmetrisch, output: Approximation A_k an Diagonalisierung

for $k = 1, 2, \ldots$

wähle $(A_{k-1})_{ij} \neq 0$, $i < j$

$A_k = Q_{ij}^* A_{k-1} Q_{ij}$

end

Bem: · Aufwand einer Iteration $= O(m)$

· Nulleinträge werden wieder gefüllt, werden aber kleiner.

· In jeder Iteration kann man den betragsmäßig größten Nichtdiagonaleintrag $(A_k)_{ij}$ wählen (totale Pivotisierung, Suche kostet $O(m^2)$ Aufwand),

aber auch zeilenweises Vorgehen $(i,j) = (1,2), (1,3), \ldots, (1,m), (2,3), (2,4), \ldots$ liefert Konvergenz.

· Nicht erst tridiagonalisieren, da die Struktur zerstört wird.

· Vergleichbar gut wie Tridiagonalisierung + QR-Iteration.

Thm: Das Jacobi-Verfahren mit totaler Pivotisierung konvergiert bis auf Maschinengenauigkeit ε_m zu einer Diagonalmatrix in $O(m^2 |\log \varepsilon_m|)$ Iterationen.

Bew: Für $S_k = \sum_{i\neq j} (A_k)_{ij}^2$ gilt $S_{k+1} \leq \left(1 - \frac{2}{m^2-m}\right) S_k$ (Übung)

$\Rightarrow S_{k+m^2} \leq \left(1 - \frac{2}{m^2}\right)^{m^2} S_k \leq C S_k$ für $C = \max_{z \geq 2} \left(1 - \frac{2}{z}\right)^z$ (Übung)

$\Rightarrow S_{\alpha m^2} \leq C^\alpha S_0 \leq C^\alpha \|A\|_F^2 \leq \varepsilon_m^2$ für $\alpha \geq \frac{2}{\log C} \log \frac{\varepsilon_m}{\|A\|_F} = O(|\log \varepsilon_m|)$ □

Bem: Das Theorem benutzt lineare Konvergenz; tatsächlich ist die Konvergenz jedoch quadratisch.

Heute: · Lanczos-Iteration

Für (dünn besetzte) Matrizen hoher Dimension sind die bisherigen Verfahren (z.B. die anfängliche Tridiagonalisierung oder auch die Jacobi-Iteration) zu aufwändig. Stattdessen kann man versuchen, iterativ mehr und mehr Eigenwerte zu approximieren. Dies geschieht mit der Lanczos-Iteration, der Version der Arnoldi-Iteration für symmetrische reelle Matrizen. Alles Folgende gilt jedoch auch für $A \neq A^*$ mit der Arnoldi- statt Lanczos-Iteration.

Def: Die Lanczos-Iteration ist die Arnoldi-Iteration für hermitesche Matrizen.

Kor: · Im $k$ten Schritt erzeugt die Lanczos-Iteration mittels Gram-Schmidt-Orthonormalisierung eine QR-Zerlegung der Krylovmatrix $K_k = [b | Ab | \ldots | A^{k-1}b]$, $K_k = Q_k R_k$ mit $Q_k = [q_1 | \ldots | q_k]$ ein Orthonormalbasis des $k$ten Krylovraums $K^k(A,b)$ und $R_k = \begin{bmatrix} \|b\|_2 & \\ 0 & \tilde{H}_k \end{bmatrix}$ für eine obere Hessenbergmatrix $\tilde{H}_k$.

· $AQ_k = Q_{k+1}\tilde{H}_k$ und $H_k = (\tilde{H}_k)_{1:k,1:k} = Q_k^* A Q_k \Rightarrow H_k, \tilde{H}_k$ sind tridiagonal!

Bem: Aus der Gram-Schmidt-Orthonormalisierung $v - \underbrace{q_1^* v}_{H_{k1}} q_1 - \dots \underbrace{q_k^* v}_{H_{kk}} q_k$ im k-ten Schritt wird durch die Symmetrie nun $v - q_{k-1}^* v q_{k-1} - q_k^* v q_k \Rightarrow$ Aufwand pro Schritt wächst nicht mehr mit k

· In obiger Gleichung nutzt man exakte mathematische Identitäten aus; numerisch fallen die Terme evtl. nicht alle weg $\Rightarrow$ die q_i verlieren nach vielen Iterationen ihre Orthogonalität

$\longrightarrow$ allgemeines Problem von Algorithmen mit tridiagonalen Termen.

Alg: (Lanczos-Iteration für Eigenwerte) input: $A \in \mathbb{R}^{n\times n}$ symmetrisch; $b \in \mathbb{R}^n$; output: Approximation $\lambda_1^k, \lambda_2^k, \dots$ an Eigenwerte

for $k = 1, 2, \dots$

berechne H_k wie in Lanczos-Iteration

finde Eigenwerte $\lambda_1^k, \lambda_2^k, \dots, \lambda_k^k$ von H_k mit Standardverfahren, z.B. QR-Iteration

end

$\lambda_1^k, \dots, \lambda_k^k$ heißen „Ritz-Schätzungen". Sie approximieren typischerweise Eigenwerte am Rande des Spektrums. Die Intuition hierzu ist wie folgt.

Thm: Solange K_k vollen Rang hat, ist χ_{H_k} die eindeutige Lösung von $\min_{p\in\hat{P}_k} \|p(A)b\|_2$ für

$\hat{P}_k = \{\text{Polynome } p \mid p(\lambda) = \lambda^k + \gamma_{k-1}\lambda^{k-1} + \dots + \gamma_0\}$.

Bem: $p(A)b$ kann für jedes $p \in \hat{P}_k$ geschrieben werden als $A^k b + K_k y$, wobei $y = \begin{pmatrix}\gamma_0\\ \vdots \\ \gamma_{k-1}\end{pmatrix}$ die Koeffizienten von p sind

$\Rightarrow \min_{p\in\hat{P}_k} \|p(A)b\|_2 = \min_{y\in\mathbb{R}^k} \|A^k b - K_k y\|_2$ ist die Projektion von $A^k b$ auf Bild K_k

$\Rightarrow K_k y$ ist eindeutig $\Rightarrow$ y eindeutig $\Rightarrow$ p eindeutig

Bew: · Für die Hessenberg-Zerlegung $A = QHQ^*$ gilt $Q = [Q_k \; U]$, $H = \begin{bmatrix} H_k & Y \\ X & Z\end{bmatrix}$ mit $X_{ij} = 0$ für $(i,j) \neq (1,k)$

· Lösung y ist charakterisiert durch Normalengleichungen & Orthogonalität $K_k y - A^k b \perp \text{Bild} K_k$

$\Rightarrow p(A)b \perp \text{Bild} K_k \Leftrightarrow p(A)b \perp Q_k \Leftrightarrow Q_k^* p(A) b = 0$

$\Leftrightarrow \underbrace{Q_k^* Q}_{(I\,|\,0)\in\mathbb{R}^{k\times n}} p(H) \underbrace{Q^* b}_{=\|b\|e_1} = 0 \Leftrightarrow p(H)_{1:k,1} = 0 \Leftrightarrow p(H_k)_{1:k,1} = 0$

· $\chi_{H_k}(H_k) = 0$, also ist auch $\chi_{H_k}(A)b \perp K^k(A,b)$

$\Rightarrow q := p - \chi_{H_k} \in P_{k-1}$ mit $\underbrace{q(A)b}_{\in K^k(A,b)} \perp K^k(A,b) \Rightarrow q(A)b = 0 \Rightarrow q = 0$ □

Thm: Sei $A \in \mathbb{R}^{m \times m}$ symmetrisch mit nur n verschiedenen Eigenwerten, $b \in \mathbb{R}^m$ habe Komponenten in Richtung von n zugehörigen Eigenvektoren. Nach n Iterationen findet das Lanczos-Verfahren alle Eigenwerte.

Bew: Sei $b = \sum_{i=1}^{n} \alpha_i q_i$, $\alpha_i \neq 0$, für orthonormale Eigenvektoren q_i zu den Eigenwerten $\lambda_1, \dots, \lambda_n$

- K_n hat vollen Rang: $A^k b = \sum_{i=1}^{n} \alpha_i \lambda_i^k q_i$

  daher $K_n y = 0 \Leftrightarrow 0 = \sum_{i=1}^{n} \alpha_i (\hat{\lambda}_i \cdot y) q_i$ für $\hat{\lambda}_i = \begin{pmatrix} \lambda_i^0 \\ \vdots \\ \lambda_i^{n-1} \end{pmatrix}$ $\Leftrightarrow \hat{\lambda}_i \cdot y = 0 \; \forall i \Leftrightarrow y = 0$

- $\chi_{K_n} = p$ für $p(\lambda) = \prod_{i=1}^{n} (\lambda - \lambda_i)$: Sei $A = Q^* \Lambda Q$ die Spektralzerlegung.

  $p(A) b = Q^* p(\Lambda) Q b = 0$, d.h. $p = \underset{p \in \hat{P}_n}{\operatorname{argmin}} \| p(A) b \|_2$ □

```
m = 100;
b = randn(m,1);
A = randn(m,m)/sqrt(m); A = A + A';
A(1,1) = 3;

E = eig(A); plot(E,zeros(m,1),'.','Markersize',5);

n = 20;
Q = zeros(m,n);
H = zeros(n,n);

Q(:,1) = b/norm(b);
for k = 1:n-1
  % compute Lanczos step
  v = A * Q(:,k);
  for j = max(1,k-1):k
    H(j,k) = v'*Q(:,j);
    v = v - H(j,k)*Q(:,j);
  end
  H(k+1,k) = norm(v);
  Q(:,k+1) = v / H(k+1,k);

  % compute Ritz estimates
  lambda = eig(H(1:k,1:k));

  % compute Lanczos polynomial
  p = @(x) prod(lambda-x);

  % plot results
  plot(E,zeros(m,1),'.','Markersize',5); hold on;
  x = linspace(-2,4,30);
  c = x; for i = 1:30, c(i) = p(x(i)); end
  plot(x,c,'r','Linewidth',3); hold off;
  axis([x(1),x(end),-200,200]);
  pause;
end
```

Heute: SVD

Thm: Sei $A = A^* \in \mathbb{K}^{m\times m}$, $\delta A \in \mathbb{K}^{m\times m}$. Für den k-ten Eigenwert und δA klein genug gilt

$$|\lambda_k(A+\delta A) - \lambda_k(A)| = O(\|\delta A\|_2),$$ d.h. die Kondition ist $\kappa \sim \frac{\|\delta A\|_2}{|\lambda_k|}\frac{\|A\|_2}{\|\delta A\|_2} = \frac{\|A\|_2}{|\lambda_k|}$

Beweis: Sei $Q\Lambda Q^* = A$ die Spektralzerlegung, $\delta\tilde{A} := Q^*\delta A Q$. Gerschgorin-Radius

$$\Rightarrow |\lambda_k(A+\delta A) - \lambda_k(A)| = |\lambda_k(\Lambda + Q^*\delta A Q) - \lambda_k(\Lambda)| \le |\delta\tilde{A}_{kk}| + \underbrace{\sum_{i\neq k}|\delta\tilde{A}_{ik}|}$$

$$\le \|\delta\tilde{A}\|_2 + \|\delta\tilde{A}\|_\infty \le \|\delta\tilde{A}\|_2 + \sqrt{n}\,\|\delta\tilde{A}^*\|_2 = O(\|\delta\tilde{A}\|_2) = O(\|\delta A\|_2) \qquad \square$$

Sei $A \in \mathbb{K}^{m\times n}$, $m \ge n$ (sonst betrachte einfach A^*), mit SVD $A = U\Sigma V^*$. Folgender Algorithmus zur Berechnung der reduzierten SVD ist naheliegend, aber instabil:

1) Berechne Spektralzerlegung $A^*A = V\Lambda V^*$, $\Lambda = \mathrm{diag}(\lambda_1,\dots,\lambda_n) \in \mathbb{R}^{n\times n}$

2) $\Sigma = \mathrm{diag}(\sqrt{\lambda_1},\dots,\sqrt{\lambda_n}) \in \mathbb{R}^{n\times n}$

3) $U = AV\Sigma^{-1}$

In der Tat, wenn 1) stabil berechnet wird, erwarten wir Fehler $\frac{|\tilde{\lambda}_i - \lambda_i|}{|\lambda_i|} = O(\kappa\varepsilon_m)$

$= O\left(\frac{\|A^*A\|}{|\lambda_i|}\varepsilon_m\right)$, d.h. $|\tilde{\lambda}_i - \lambda_i| = O(\varepsilon_m\|A\|^2) \Rightarrow |\tilde{\sigma}_i - \sigma_i| = O(\varepsilon_m\|A\|^2/\sqrt{\lambda_i}) = O(\varepsilon_m\|A\|^2/\sigma_i)$.

Ein stabiler Algorithmus würde $\frac{|\tilde{\sigma}_i - \sigma_i|}{\sigma_i} = O(\kappa_{\sigma_i}\varepsilon_m)$ liefern, wobei $\kappa_{\sigma_i} \sim \frac{\|A\|}{\sigma_i}$ die Konditionszahl der Berechnung des i-ten Singulärwerts σ_i ist, d.h. $|\tilde{\sigma}_i - \sigma_i| = O(\varepsilon_m\|A\|)$.

Tatsächlich kann man die SVD umschreiben als das Problem, die Spektralzerlegung der hermiteschen Matrix $B = \begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}$ zu finden. Aus dieser kann man wegen

$$\begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix} = \begin{bmatrix} V & V \\ U & -U \end{bmatrix}\begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix}\begin{bmatrix} V & V \\ U & -U \end{bmatrix}^*$$

die Singulärvektoren und Singulärwerte als Eigenwerte ablesen, somit $\kappa_{\sigma_i} \sim \frac{\|B\|_2}{\sigma_i} \sim \frac{\|A\|}{\sigma_i}$.

Der Fehler der naiven Methode ist somit um den Faktor $\frac{\|A\|_2}{\sigma_i}$ zu groß – für den größten Singulärwert ist dies 1, für den kleinsten jedoch $\kappa(A)$.

In der Praxis wird die SVD mittels Spektralzerlegung von B berechnet, jedoch wird A zunächst bidiagonalisiert.

Alg: (Golub-Kahan-Bidiagonalisierung) input: $A \in \mathbb{K}^{m\times n}$, output: Bidiagonalisierung B

$B = A$

for $k = 1$ bis n

eliminiere $B_{k+1:m,k}$ mittels Householder-Reflektion U_k^* (*)

eliminiere $B_{k,k+2:n}$ mittels Householder-Reflektion V_k (**)

end

Bem: Der Algorithmus liefert $U_1, \dots, U_n \in \mathbb{K}^{m\times m}$ unitär, $V_1, \dots, V_n \in \mathbb{K}^{n\times n}$ unitär, $B \in \mathbb{K}^{m\times n}$ mit

$B_{ij} = 0$ für $j \notin \{i, i+1\}$ und $A = U_1 \cdots U_n \, B \, (V_1 \cdots V_n)^*$

$$\begin{bmatrix} x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \\ x & x & x & x \end{bmatrix} \xrightarrow{U_1^*\cdot} \begin{bmatrix} x & x & x & x \\ & x & x & x \\ & x & x & x \\ & x & x & x \\ & x & x & x \end{bmatrix} \xrightarrow{\cdot V_1} \begin{bmatrix} x & x & & \\ & x & x & x \\ & x & x & x \\ & x & x & x \\ & x & x & x \end{bmatrix} \xrightarrow{U_2^*\cdot} \begin{bmatrix} x & x & & \\ & x & x & x \\ & & x & x \\ & & x & x \\ & & x & x \end{bmatrix} \xrightarrow{\cdot V_2} \begin{bmatrix} x & x & & \\ & x & x & \\ & & x & x \\ & & x & x \\ & & x & x \end{bmatrix} \cdots \longrightarrow \begin{bmatrix} x & x & & \\ & x & x & \\ & & x & x \\ & & & x \\ & & & \end{bmatrix}$$

A $\quad U_1^* A$ $\quad U_1^* A V_1$ $\quad U_2^* U_1^* A V_1$ $\quad U_2^* U_1^* A V_1 V_2$ $\quad B$

- insgesamt n Reflektionen von links und $n-2$ von rechts ($V_{n-1} = V_n = I$);

Schritte (*) genau wie in QR-Zerlegung von $A \in \mathbb{K}^{m\times n}$, (**) wie in QR-Zerlegung von A^*

$\Rightarrow$ Aufwand $\sim 2mn^2 - \frac{2}{3}n^3 + 2mn^2 - \frac{2}{3}n^3 = 4mn^2 - \frac{4}{3}n^3$ flops

Alg: (Lawson-Hanson-Chan-Bidiagonalisierung) input: $A \in \mathbb{K}^{m\times n}$, output: Bidiagonalisierung B

berechne reduzierte QR-Zerlegung $A = QR$ mittels Householder-Verfahren

berechne Golub-Kahan-Bidiagonalisierung von $R_{1:n,1:n} = \tilde{U}\tilde{B}\tilde{V}^*$

setze $U = Q \begin{bmatrix} \tilde{U} & 0 \\ 0 & I \end{bmatrix}$, $V = \tilde{V}$, $B = \begin{bmatrix} \tilde{B} \\ 0 \end{bmatrix}$

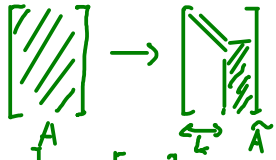
Bem: $\cdot$ $A = UBV^*$

$\cdot$ Aufwand $\sim 2mn^2 - \frac{2}{3}n^3 + \left(4n^3 - \frac{4}{3}n^3\right) = 2mn^2 + 2n^3$ flops

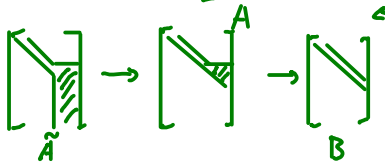
$<$ Golub-Kahan-Aufwand genau dann wenn $m > \frac{5}{3}n$

Alg: (Drei-Schritt-Bidiagonalisierung) input/output wie oben

1) führe k Schritte Golub-Kahan-Bidiagonalisierung durch



2) Lawson-Hanson-Chan-Bidiagonalisierung auf $\tilde{A}_{k+1:m,\,k+1:n}$



Bem: Aufwand $\sim \underbrace{\left[\left(4mn^2 - \frac{4}{3}n^3\right)}_{\text{GK gesamt}} - \underbrace{\left(4(m-k)(n-k)^2 - \frac{4}{3}(n-k)^3\right)}_{\text{GK für } (m-k)\times(n-k) \text{ Matrix}}\right] + \underbrace{\left[2(m-k)(n-k)^2 + 2(n-k)^3\right]}_{\text{LHC}}$ flops

$\Rightarrow$ optimale Wahl $k = 2n - m$ liefert Aufwand $\sim 4mn^2 - \frac{4}{3}n^3 - \frac{2}{3}(m-n)^3$ flops