Direkte Verfahren für lineare Gleichungssysteme

$$Ax = b$$

direkte Verfahren → Lösung nach endlich vielen Schritten

Einschränkung auf (voll besetzte) nicht singuläre Systemmatrizen mit rechter Seite
⇒ A so faktorisieren, dass leichter lösbares Teilproblem entsteht
→ LR-Zerlegung
→ QR-Zerlegung

LR-Zerlegung

$A = L \cdot R$ (R: rechte (obere) Dreiecksmatrix; L: linke (untere) Dreiecksmatrix)

$$Ax = b \Rightarrow \underbrace{LR}_{y} x = b$$
$$Ly = b$$
$$Rx = y$$

⇒ ohne Pivotsuche

$$A = \begin{pmatrix} 1 & 2 & 0 & 0 \\ -3 & -8 & 3 & 0 \\ 0 & -8 & 13 & 3 \\ 0 & 0 & -2 & -4 \end{pmatrix}$$

1. L = Einheitsmatrix, $R = A$

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 1 & 2 & 0 & 0 \\ -3 & -8 & 3 & 0 \\ 0 & -8 & 13 & 3 \\ 0 & 0 & -2 & -4 \end{pmatrix}$$

2. Erste Spalte von R unterhalb der Diagonalen putzen, L „faktorisieren"

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & -2 & 3 & 0 \\ 0 & -8 & 13 & 3 \\ 0 & 0 & -2 & -4 \end{pmatrix}$$

$$L_{2,1} = \frac{R_{2,1}}{R_{1,1}} = -3 \quad \Rightarrow \quad R_{2,i} = R_{2,i} - L_{2,1} \cdot R_{1,i}$$

$i = 1$ bis 4

$-3 + 3 \cdot 1 = 0$
$-8 + 3 \cdot 2 = -2$
$3 + 3 \cdot 0 = 3$
$0 + 3 \cdot 0 = 0$

3. Analog 2. Spalte & 3. Spalte

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 0 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & -2 & 3 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & -2 & -4 \end{pmatrix}$$

$$L_{3,2} = \frac{R_{3,2}}{R_{2,2}} = 4 \quad \Rightarrow \quad R_{3,i} = R_{3,i} - L_{3,2} \cdot R_{2,i}$$

$i = 2$ bis 4

$-8 - 4 \cdot (-2) = 0$
$13 - 4 \cdot 3 = 1$
$3 - 4 \cdot 0 = 3$

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 0 & 4 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & -2 & 3 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

$$\Rightarrow \quad A = L \cdot R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 \\ 0 & 4 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 0 & 0 \\ 0 & -2 & 3 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$

$\Rightarrow L \cdot (Rx) = \vec{b}$
$L \cdot y = \vec{b}$ Vorwärtssubstitution
$Rx = y$ Rückwärtssubstitution

Algorithmus:

$L = \mathbb{1}$

$R = B$

for k von 1 bis $(n-1)$ bei n Spalten, Zeilen

  for j von $k+1$ bis n

$$L_{jk} = \frac{R_{jk}}{R_{kk}}$$

R_{jk} ← Elemente unterhalb der Diagonale in der Spalte

R_{kk} ← Diagonalelement von R

$$R_{j,k:4} = R_{j,k:4} - L_{jk} R_{k,k:4}$$

← Spalte putzen, indem untere Zeile − L * Zeile der Diagonalen

---

R-Zerlegung einer tridiagonalen Matrix

$$A = \begin{pmatrix} & & 0 \\ & & \\ 0 & & \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & & 0 \\ & & \\ 0 & & 1 \end{pmatrix}}_{=: L} \cdot \underbrace{\begin{pmatrix} & & 0 \\ & & \\ 0 & & \end{pmatrix}}_{=: R}$$

$(b_i)_1^n$ auf Diag.

$(a_i)_1^{n-1}$, $(c_i)_1^{n-1}$ } auf Neben-diag.

L und R sind explizit gegeben:

$$L = \begin{pmatrix} 1 & & \\ l_1 & \ddots & \\ & l_{n-1} & 1 \end{pmatrix} \quad \text{und} \quad R = \begin{pmatrix} r_1 & c_1 & \\ & \ddots & \\ & r_{n-1} & c_{n-1} \\ & & r_n \end{pmatrix}$$

mit $l_i = a_i \cdot \frac{\delta_{i-1}}{\delta_i}$ und $r_i = \frac{\delta_i}{\delta_{i-1}}$

$$\delta_i = \begin{cases} 1 & i = 0 \\ b_1 & i = 1 \\ b_i \delta_{i-1} - a_{i-1} c_{i-1} \delta_{i-1} & i \geq 2 \end{cases}$$

$$\Rightarrow L_k = \frac{a_k}{r_k} \qquad r_k = \begin{cases} b_1 & \text{für } k = 1 \\ b_k - c_{k-1} l_{k-1} & \text{für } k \geq 2 \end{cases}$$

⇒ Aufwand für Teilprobleme beim Lösen einer tridiagonalen Matrix A

① LR-Zerlegung $A = L \cdot R$

⇒ $m-1$ Div.
$m-1$ Subtr.
$m-1$ Mult.

② Vorwärtssubstitution: $L \cdot y = b$

$\Rightarrow$ 0 Div.
$m-1$ Subtr.
$m-1$ Mult.

③ Rückwärtssubstitution: $R \cdot x = y$

$\Rightarrow$ m Div
$m-1$ Subtr.
$m-1$ Mult.

$\Rightarrow$ gesamt: $O(n)$

$2m-1$ Div
$3m-3$ Subtr.
$3m-3$ Mult.

Σ: $8m-7$ Operationen $< m^2$ Operationen

$\Rightarrow$ allg. bei Bandbreite b, Matrixordnung n:

$O(b^2 n)$

Aufwand zum Lösen eines linearen GS mit LR-Zerlegung

LR-Zerlegung	$\frac{2}{3}n^3$	$+ O(n^2)$
Vorwärtseinsetzen	n^2	$+ O(0)$
Rückwärtseinsetzen	n^2	$+ O(0)$
Σ:	$\frac{2}{3}n^3$	$+ O(n^2)$

Wann benutze ich LR-Zerlegung <u>ohne</u> Pivot-Suche?

- Permutationsmatrix P gegeben $\Rightarrow$ $P \cdot A$ ausmultiplizieren
- symmetr. positiv definite Ausgangsmatrix

$\rightarrow$ Bestimmung $\det(A) = \det(L) \cdot \det(R) = 1 \cdot \prod_{i=1}^{n} r_{ii}$

$\Rightarrow$ <u>mit Spaltenpivotsuche</u> $\boxed{A = P \cdot L \cdot R}$

Warum braucht man die Pivotisierung?

- numerische Stabilität
- Zeilenvertauschung (dargestellt durch Faktorisierung mit Permutationsmatrix), um Diagonalelemente $a_{ii} = 0$ zu vermeiden

$\Rightarrow$ <u>betragsmäßig</u> größtes Element der betrachteten Spalte auf die Diagonale bringen (mittels Zeilenvertauschungen)

Algorithmus:

$A \in \mathbb{R}^{n \times n}$

$P = \mathbb{1}$, $L = \mathbb{1}$, $R = A$

for $k = 1$ bis $(n-1)$

- $i = \text{argmax}_{i \geq k} |R_{i,k}|$ — Pivotelement wählen
- $P_k \leftrightarrow P_i$ — → Zeilentausch in P
- $R_{k,k:n} \leftrightarrow R_{i,k:n}$ — → Zeilentausch in R_{Rest}, sodass PE auf Diagonale
- $L_{k,1:k-1} \leftrightarrow L_{i,1:k-1}$ — → Zeilentausch in L unterhalb der Diagonalen

  for $j = k+1$ bis n

  - $L_{j,k} = \frac{R_{j,k}}{R_{k,k}}$
  - $R_{j,k:n} = R_{j,k:n} - L_{j,k} R_{k,k:n}$

  } wie LR-Zerlegung

$P = P^T$

Beispiel:

$$A = \begin{pmatrix} 6 & 2 & 6 \\ 8 & 4 & 6 \\ 4 & 8 & 6 \end{pmatrix}$$

1. Initialisiere:

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R = \begin{pmatrix} 6 & 2 & 6 \\ \textcircled{8} & 4 & 6 \\ 4 & 8 & 6 \end{pmatrix}$$

2. Suche Pivotelement → Zeilen tauschen

$1 \leftrightarrow 2$

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad R = \begin{pmatrix} \textcircled{8} & 4 & 6 \\ 6 & 2 & 6 \\ 4 & 8 & 6 \end{pmatrix}$$

3. LR-Zerlegung 1. Spalte

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{6}{8} & 1 & 0 \\ \frac{4}{8} & 0 & 1 \end{pmatrix} \quad R = \begin{pmatrix} 8 & 4 & 6 \\ 0 & -1 & \frac{3}{2} \\ 0 & \textcircled{6} & 3 \end{pmatrix}$$

4. Wiederhole 2 & 3
   → Tausche Zeile 2 & 3

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{3}{4} & 0 & 1 \end{pmatrix} \quad R = \begin{pmatrix} 8 & 4 & 6 \\ 0 & 6 & 3 \\ 0 & -1 & \frac{3}{2} \end{pmatrix}$$

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \quad L = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{3}{4} & -\frac{1}{6} & 1 \end{pmatrix} \quad R = \begin{pmatrix} 8 & 4 & 6 \\ 0 & 6 & 3 \\ 0 & 0 & 2 \end{pmatrix}$$

5. Wenn LR-Zerlegung abgeschlossen, transponiere P

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}^T = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

Anzahl Vertauschungen
→ gerade: $\det(P) = +1$
→ ungerade: $\det(P) = -1$
um $\mathbb{1}$ zu erreichen (*)

$$A = P \cdot L \cdot R = \underbrace{\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}}_{P} \cdot \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{3}{4} & -\frac{1}{6} & 1 \end{pmatrix}}_{L} \cdot \underbrace{\begin{pmatrix} 8 & 4 & 6 \\ 0 & 6 & 3 \\ 0 & 0 & 2 \end{pmatrix}}_{R}$$

Lösen: $\overbrace{PLR}^{A} x = b$

$$LRx = P^{-1} b = \underbrace{P^T b}_{=\tilde{b}} \quad \text{(ausmultiplizieren)}$$

$$LRx = \tilde{b}$$

→ Vorwärts- und Rückwärtssubst.
analog zur normalen LR-Zerlegung

$$\det(A) = \det(P) \cdot \det(L) \cdot \det(R) = \pm 1 \cdot 1 \cdot \prod_{i=1}^{n} r_{i,i} \quad (4)$$

Vorwärtssubstitution

$$L \cdot \underbrace{(Rx)}_{=y} = b$$

$$\Rightarrow y_j = \frac{b_j - \sum_{i=1}^{j-1} L_{j,i} \cdot y_i}{L_{j,j}}$$

Rückwärtssubstitution

$$Rx = y$$

$$\Rightarrow x_j = \frac{y_j - \sum_{i=j+1}^{n} r_{j,i} \cdot x_i}{r_{j,j}}$$

→ Effizientes Lösen mehrerer GS mit gleicher Matrix A

→ bestimme 1x LR von A ($O(n^3)$)

→ für jede rechte Seite mit Vor- & Rückwärtssubstitution lösen

→ alle rechten Seiten zu Beginn bekannt
↳ in m-spaltiger Matrix $B \in \mathbb{R}^{n \times m}$ zsf.
$\Rightarrow AX = B \quad X \in \mathbb{R}^{n \times m}$

LR-Zerlegung von $(A|B)$
→ $L, (R|B^*)$

→ $RX = B^*$
(keine Vorwärtssubst.)

Cholesky-Zerlegung

→ A symmetrisch ($A = A^T$) & positiv definit !

→ A faktorisieren: $A = L \cdot D \cdot L^T$
mit L aus LR-Zerlegung
D Diagonalanteil von R
↳ wg. pos. def.: nur pos. Elemente in der Diagonalen

→ beide Dreiecksfaktoren gleich
↳ nur einen explizit berechnen → spart x Hälfte der Operationen

- auch ohne Pivotisierung stabil
- Aufwand halb so groß wie bei LR (etwa)

QR-Zerlegung

$A = Q \cdot R$
↑ orthogonale Matrix

$Ax = b \Rightarrow QRx = b$
$Rx = Q^T b$

Orthogonale Matrix

- $Q^{-1} = Q^T$ $\quad \det(Q) = \pm 1$
- $Q^T Q = \mathbb{1} = Q Q^T$
- Spalten oder Zeilen von Q bilden Orthonormalbasis
- Abbildung Q ist winkel- und längentreu
- $Qx \circ Qy = x \circ y$

-> im 2D-Fall: Rotationsmatrizen / Spiegelungen

=> Givens-Rotation

$J_{i,j}(\varphi) \in \mathbb{R}^{n \times n}$ Rotationsmatrix an der Achse ij mit dem Winkel φ (gegen den Uhrzeigersinn)

$$J_{i,j}(\varphi) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & c_{jj} & & -s_{ji} & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \\ & & & s_{ij} & & c_{ii} & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow j \\ \\ \\ \\ \leftarrow i \\ \\ \end{matrix}$$

$$= \begin{cases} 1 & \text{for } k = l,\ k \neq i,\ k \neq j \\ c = \cos(\varphi) & \text{for } k = l \neq j,\ k = l = i \\ s = \sin(\varphi) & \text{for } k = i,\ l = j \\ -s = -\sin(\varphi) & \text{for } k = j,\ l = i \\ 0 & \text{sonst} \end{cases}$$

-> $J_{ij}^{-1}(\varphi) = J_{ij}(-\varphi) = J_{ij}^{T}(\varphi)$ (Orthogonalität)

Idee: $J(c,s) \cdot A = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} = \begin{pmatrix} * & * \\ s\alpha + c\gamma & * \end{pmatrix} \overset{!}{=} \begin{pmatrix} * & * \\ 0 & * \end{pmatrix}$

$$\Rightarrow s\alpha + c\gamma \overset{!}{=} 0$$

-> Elimination aller Einträge $a_{ij} \neq 0$ mit $i > j$ mithilfe einer Givens-Rotation

=> allgemein:

$$c = \cos(\varphi) = \frac{\operatorname{sgn}(a_{jj}) \cdot a_{jj}}{\sqrt{a_{jj}^2 + a_{ij}^2}}$$

$$s = \sin(\varphi) = \frac{-\operatorname{sgn}(a_{jj}) \cdot a_{ij}}{\sqrt{a_{jj}^2 + a_{ij}^2}}$$

$$A = \underbrace{J_{21}^T \cdot \ldots \cdot J_{m,n^*}^T}_{=Q} \cdot \underbrace{J_{m,n^*} \cdot \ldots \cdot J_{2,1} \cdot A}_{=R}$$

mit $A \in \mathbb{R}^{m \times n}$ und $n^* = \min\{m-1, n\}$

Algorithmus:

$Q = \mathbb{1}_n$, $R = A$

k = Spalte
j = Zeile

```
for k = 1 bis n*
    for j = k+1 bis n
        J_jk = Givens-Rotation(R, j, k)
        R = J_jk · R
        Q = Q · J_jk^T
```

Beispiel: $A = \begin{pmatrix} 8 & 7 & 7 \\ 6 & 9 & 2 \\ 24 & 16 & 8 \end{pmatrix}$

1. Initialisiere $Q = \mathbb{1}$, $R = A$

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 8 & 7 & 7 \\ 6 & 9 & 2 \\ 24 & 16 & 8 \end{pmatrix}$$

2. Nulleinträge unterhalb der Diagonalen erzeugen mithilfe der Givens-Rotationsmatrix

$$c = \frac{a_{11}}{\sqrt{a_{11}^2 + a_{21}^2}} = \frac{8}{10}$$

$$s = -\frac{\operatorname{sgn}(a_{11}) \cdot a_{21}}{\sqrt{a_{11}^2 + a_{21}^2}} = -\frac{6}{10}$$

$$\Rightarrow J_{21} = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{8}{10} & \frac{6}{10} & 0 \\ -\frac{6}{10} & \frac{8}{10} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

→ multipliziere: $R = J_{21} \cdot R$

$Q = J_{21}^T \cdot \mathbb{1}$

$$R = \begin{pmatrix} \frac{8}{10} & \frac{6}{10} & 0 \\ -\frac{6}{10} & \frac{8}{10} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 8 & 7 & 7 \\ 6 & 9 & 2 \\ 24 & 16 & 8 \end{pmatrix} = \begin{pmatrix} 10 & 11 & \frac{34}{5} \\ 0 & 3 & -\frac{13}{5} \\ 24 & 16 & 8 \end{pmatrix}$$

$$Q = J_{21}{}^T = \begin{pmatrix} \frac{8}{10} & -\frac{6}{10} & 0 \\ \frac{6}{10} & \frac{8}{10} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Wiederhole:

$$c = \frac{a_{11}}{\sqrt{a_{11}^2 + a_{31}^2}} = \frac{5}{13}$$

$$s = -\frac{sgn(a_{11}) \cdot a_{31}}{\sqrt{a_{11}^2 + a_{31}^2}} = -\frac{12}{13}$$

$$\Rightarrow J_{31} = \begin{pmatrix} c & 0 & -s \\ 0 & 1 & 0 \\ s & 0 & c \end{pmatrix} = \begin{pmatrix} \frac{5}{13} & 0 & \frac{12}{13} \\ 0 & 1 & 0 \\ -\frac{12}{13} & 0 & \frac{5}{13} \end{pmatrix}$$

$$R = J_{31} \cdot R$$

$$= \begin{pmatrix} \frac{5}{13} & 0 & \frac{12}{13} \\ 0 & 1 & 0 \\ -\frac{12}{13} & 0 & \frac{5}{13} \end{pmatrix} \begin{pmatrix} 10 & 11 & \frac{34}{5} \\ 0 & 3 & -\frac{13}{5} \\ 24 & 16 & 8 \end{pmatrix} = \begin{pmatrix} 26 & 19 & 10 \\ 0 & 3 & -\frac{13}{5} \\ 0 & -4 & -\frac{16}{5} \end{pmatrix}$$

$$Q = Q \cdot J_{31}{}^T = \begin{pmatrix} \frac{8}{10} & -\frac{6}{10} & 0 \\ \frac{6}{10} & \frac{8}{10} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{5}{13} & 0 & -\frac{12}{13} \\ 0 & 1 & 0 \\ \frac{12}{13} & 0 & \frac{5}{13} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{4}{13} & -\frac{3}{5} & -\frac{48}{65} \\ \frac{3}{13} & \frac{4}{5} & -\frac{36}{65} \\ \frac{12}{13} & 0 & \frac{5}{13} \end{pmatrix}$$

Wiederhole für die nächste Spalte:

$$c = \frac{a_{22}}{\sqrt{a_{22}^2 + a_{32}^2}} = \frac{3}{5}$$

$$s = -\frac{sgn(a_{32}) \cdot a_{32}}{\sqrt{a_{22}^2 + a_{32}^2}} = \frac{4}{5}$$

$$J_{32} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{3}{5} & -\frac{4}{5} \\ 0 & \frac{4}{5} & \frac{3}{5} \end{pmatrix}$$

$$R = J_{32} \cdot R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{3}{5} & -\frac{4}{5} \\ 0 & \frac{4}{5} & \frac{3}{5} \end{pmatrix} \begin{pmatrix} 26 & 19 & 10 \\ 0 & 3 & -\frac{13}{5} \\ 0 & -4 & -\frac{16}{5} \end{pmatrix} = \begin{pmatrix} 26 & 19 & 10 \\ 0 & 5 & 1 \\ 0 & 0 & -4 \end{pmatrix}$$

$$Q = Q \cdot J_{32}^T = \begin{pmatrix} \frac{4}{13} & -\frac{3}{5} & -\frac{48}{65} \\ \frac{3}{13} & \frac{4}{5} & -\frac{36}{65} \\ \frac{12}{13} & 0 & \frac{5}{13} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{3}{5} & \frac{4}{5} \\ 0 & -\frac{4}{5} & \frac{3}{5} \end{pmatrix} = \begin{pmatrix} \frac{4}{13} & \frac{3}{13} & -\frac{12}{13} \\ \frac{3}{13} & \frac{12}{13} & \frac{4}{13} \\ \frac{12}{13} & -\frac{4}{13} & \frac{3}{13} \end{pmatrix}$$

$$\Rightarrow A = \underbrace{\frac{1}{13} \begin{pmatrix} 4 & 3 & -12 \\ 3 & 12 & 4 \\ 12 & -4 & 3 \end{pmatrix}}_{Q} \cdot \underbrace{\begin{pmatrix} 26 & 19 & 10 \\ 0 & 5 & 1 \\ 0 & 0 & -4 \end{pmatrix}}_{R}$$

$\frac{n(n-1)}{2}$ Schritte zu R

Eigenschaften:
- auch ohne Pivotsuche stabil
- bessere Konditionierung als LR-Zerlegung

$$K_2(A) = K_2(R) \quad , \quad K_2(Q) = 1$$

Kondition = Abhängigkeit der Lösung von Störung der Eingangsdaten

- leichter parallelisierbar, aber schwerer zu implementieren
- Hauptproblem: Aufwand etwa 4x so hoch wie LR-Zerlegung
  - -> Verbesserung: durch Skalierung alle Diagonalelemente = 1, Skalierungen separat multiplizieren (spart ≈ Hälfte des Rechenaufwands) „rationale Givensrotation"

=> Householderspiegelungen

Ziel: mithilfe einer Spiegelung die notwendigen Nullen in einer Spalte erzeugen

H ist symmetrisch und orthogonal: $H^{-1} = H^T = H$

Aufwand: $O(n^3)$ (etwa doppelt so hoch wie LR)

Algorithmus: mit $A \in \mathbb{R}^{n \times m}$, $n \leq m$

$Q = \mathbb{1}_n$, $R = A$

for $k = 1$ bis $\min(m, n-1)$

- $x = R_{k:n,\,k}$ — x ist Vektor der k-ten Spalte ab dem Diagonalelement
- $e_1^k = (1, 0, \dots, 0)^T$ ← Länge: $n-k+1$
- $v = x - \|x\|_2 \cdot e_1^k$
- $H_k' = \mathbb{1}_{n-k+1} - 2 \cdot \frac{vv^T}{\|v\|_2^2}$
- $H_k = \left(\begin{array}{c|c} \mathbb{1}_{k-1} & 0 \\ \hline 0 & H_k' \end{array}\right)$
- $Q = Q \cdot H_k$
- $R = H_k \cdot R$

Beispiel: $A = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}$

1. $Q = \mathbb{1}$, $R = A$

$$Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} 12 & -51 & 4 \\ 6 & 167 & -68 \\ -4 & 24 & -41 \end{pmatrix}$$

2. Bestimme H_1 (erste Spalte)

$$x = \begin{pmatrix} 12 \\ 6 \\ -4 \end{pmatrix} \qquad \|x\|_2 = \sqrt{12^2 + 6^2 + (-4)^2} = 14 \qquad e_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

$$v = x - \|x\|_2 \cdot e_1 = \begin{pmatrix} 12 \\ 6 \\ -4 \end{pmatrix} - 14 \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} -2 \\ 6 \\ -4 \end{pmatrix}$$

$$\|v\|_2 = \sqrt{(-2)^2 + 6^2 + (-4)^2} = \sqrt{56}$$

$$H_1' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - 2 \cdot \frac{vv^T}{\|v\|_2^2}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - 2 \cdot \frac{\begin{pmatrix} -2 \\ 6 \\ -4 \end{pmatrix} \cdot \begin{pmatrix} -2 & 6 & -4 \end{pmatrix}}{56}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{28} \cdot \begin{pmatrix} 4 & -12 & 8 \\ -12 & 36 & -24 \\ 8 & -24 & 16 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \frac{1}{7} \begin{pmatrix} 1 & -3 & 2 \\ -3 & 9 & -6 \\ 2 & -6 & 4 \end{pmatrix} = \begin{pmatrix} \frac{6}{7} & \frac{3}{7} & -\frac{2}{7} \\ \frac{3}{7} & -\frac{2}{7} & \frac{6}{7} \\ -\frac{2}{7} & \frac{6}{7} & \frac{3}{7} \end{pmatrix} = H_1$$

$$Q = Q \cdot H_1 = \begin{pmatrix} \frac{6}{7} & \frac{3}{7} & -\frac{2}{7} \\ \frac{3}{7} & -\frac{2}{7} & \frac{6}{7} \\ -\frac{2}{7} & \frac{6}{7} & \frac{3}{7} \end{pmatrix}$$

$$R = H_1 \cdot R = \begin{pmatrix} 14 & 21 & -14 \\ 0 & -49 & -14 \\ 0 & 168 & -77 \end{pmatrix}$$

3. Wiederhole für die 2. Spalte

$$x = \begin{pmatrix} -49 \\ 168 \end{pmatrix} \qquad \|x\|_2 = \sqrt{30625} \qquad e_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$u = \begin{pmatrix} -224 \\ 168 \end{pmatrix} \qquad \|u\|_2 = 280$$

$$H_2' = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - 2 \cdot \frac{uu^T}{\|u\|_2^2} = \begin{pmatrix} \frac{7}{25} & -\frac{24}{25} \\ -\frac{24}{25} & -\frac{7}{25} \end{pmatrix}$$

→ Fülle H' auf $n \times n$ auf

$$H_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{7}{25} & -\frac{24}{25} \\ 0 & -\frac{24}{25} & -\frac{7}{25} \end{pmatrix}$$

$$Q = Q \cdot H_2 = \begin{pmatrix} \frac{6}{7} & \frac{69}{175} & -\frac{58}{175} \\ \frac{3}{7} & -\frac{158}{175} & \frac{6}{175} \\ -\frac{2}{7} & -\frac{6}{35} & -\frac{33}{35} \end{pmatrix} \qquad R = H_2 \cdot R = \begin{pmatrix} 14 & 21 & -14 \\ 0 & -175 & 70 \\ 0 & 0 & 35 \end{pmatrix}$$

$$A = \underbrace{\frac{1}{175} \cdot \begin{pmatrix} 150 & 69 & -58 \\ 75 & -158 & 6 \\ -50 & -30 & -165 \end{pmatrix}}_{Q} \cdot \underbrace{\begin{pmatrix} 14 & 21 & -14 \\ 0 & -175 & 70 \\ 0 & 0 & 35 \end{pmatrix}}_{R}$$

Orthonormalisierung

→ Gram-Schmidt-Verfahren
- klassisch: numerisch instabil

besser: QR-Zerlegung

$$\begin{pmatrix} | & | & & | \\ b_1 & b_2 & \cdots & b_n \\ | & | & & | \end{pmatrix} = B = Q \cdot R = \begin{pmatrix} | & | & | & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & | & | \end{pmatrix} \cdot R$$

↑ lin. unabh. Spaltenvekt.

↑ orthonormale Vektoren q_i

Shermann-Morrison-Woodbury-Formel

Bsp. $A = \begin{pmatrix} -1 & 0 & 4 \\ 2 & 4 & -3 \\ 2 & 5 & 2 \end{pmatrix}$ v_1 v_2 v_3

Gram-Schmidt-Orthogonalisierung

$$q_n = v_n - \sum_{i=1}^{n-1} \frac{v_n^T q_i}{|q_i|^2} \cdot q_i$$

v_n: Spaltenvektoren von A; $|q_i|^2$: Länge ohne Wurzel

$$q_1 = \begin{pmatrix} -1 \\ 2 \\ 2 \end{pmatrix}$$

$$q_2 = \begin{pmatrix} 0 \\ 4 \\ 5 \end{pmatrix} - \underbrace{\frac{18}{9}}_{=2} \begin{pmatrix} -1 \\ 2 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix}$$

$$q_3 = \begin{pmatrix} 4 \\ -3 \\ 2 \end{pmatrix} - \frac{-6}{9} \begin{pmatrix} -1 \\ 2 \\ 2 \end{pmatrix} - \frac{10}{5} \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ -3 \\ 2 \end{pmatrix} + \frac{2}{3} \begin{pmatrix} -1 \\ 2 \\ 2 \end{pmatrix} - \begin{pmatrix} 4 \\ 0 \\ 2 \end{pmatrix}$$

$$= \frac{1}{3} \left(\begin{pmatrix} 12 \\ -9 \\ 6 \end{pmatrix} + \begin{pmatrix} -2 \\ 4 \\ 4 \end{pmatrix} - \begin{pmatrix} 12 \\ 0 \\ 6 \end{pmatrix} \right) = \frac{1}{3} \begin{pmatrix} -2 \\ -5 \\ 4 \end{pmatrix} \dots$$

$$Q = \begin{pmatrix} -\frac{1}{3} & \frac{2}{\sqrt{5}} & -\frac{2}{\sqrt{45}} \\ \frac{2}{3} & 0 & -\frac{5}{\sqrt{45}} \\ \frac{2}{3} & \frac{1}{\sqrt{5}} & \frac{4}{\sqrt{45}} \end{pmatrix}$$

Spaltenvektoren q_i: $|q_i| = 1$

NR: $|q_3| = \frac{1}{3}\sqrt{45} = \frac{1}{3}\sqrt{9 \cdot 5} = \sqrt{5}$

Länge der q_i

$$r_{ii} = |q_i|$$

$$r_{ij} = \frac{v_i^T q_i}{|q_i|}$$

$$R = \begin{pmatrix} 3 & 6 & -2 \\ 0 & \sqrt{5} & 2\sqrt{5} \\ 0 & 0 & \sqrt{5} \end{pmatrix}$$

$$\frac{10}{\sqrt{5}} = \frac{10\sqrt{5}}{\sqrt{5}\sqrt{5}} = \frac{10\sqrt{5}}{5} = 2\sqrt{5}$$

Cholesky-Zerlegung

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ & a_{22} & a_{23} \\ \text{sym.} & & a_{33} \end{pmatrix} = \begin{pmatrix} l_{11}^2 & l_{12} \cdot l_{11} & l_{13} \cdot l_{11} \\ & l_{12}^2 + l_{22}^2 & l_{12} \cdot l_{13} + l_{22} \cdot l_{23} \\ \text{sym.} & & l_{13}^2 + l_{23}^2 + l_{33}^2 \end{pmatrix}$$

$$A = L \cdot L^T = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \cdot \begin{pmatrix} l_{11} & l_{12} & l_{13} \\ 0 & l_{22} & l_{23} \\ 0 & 0 & l_{33} \end{pmatrix} = \dots$$

$$\Rightarrow \quad a_{11} = l_{11}^2 \rightarrow l_{11} = \sqrt{a_{11}}$$

$$a_{12} = l_{12} \cdot l_{11} \rightarrow l_{12} = \frac{a_{12}}{l_{11}}$$

$$a_{13} = l_{13} \cdot l_{11} \rightarrow l_{13} = \frac{a_{13}}{l_{11}}$$

$$\dots$$

Allg. Vorschrift:

$$l_{kk} = \left(a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2 \right)^{\frac{1}{2}}$$

$$l_{ik} = \frac{a_{ik} - \sum_{j=1}^{k-1} l_{ij} \cdot l_{kj}}{l_{kk}}, \quad i = k+1 : n$$

Bsp.:

$$A = \begin{pmatrix} 1 & 2 & 1 \\ & 13 & 2 \\ \text{sym.} & & 9 \end{pmatrix}$$

$$l_{11} = \sqrt{a_{11}} = \sqrt{1} = 1$$

$$l_{12} = \frac{a_{12}}{l_{11}} = \frac{2}{1} = 2$$

$$l_{13} = \frac{a_{13}}{l_{11}} = \frac{1}{1} = 1$$

$$l_{22} = \sqrt{a_{22} - l_{12}^2} = \sqrt{13 - 2^2} = \sqrt{9} = 3$$

$$l_{23} = \frac{a_{23} - l_{12} \cdot l_{13}}{l_{22}} = \frac{2 - 2 \cdot 1}{3} = 0$$

$$l_{33} = \sqrt{a_{33} - l_{13}^2 - l_{23}^2} = \sqrt{9 - 1^2 - 0^2} = \sqrt{8}$$

$$\Rightarrow \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 1 & 0 & \sqrt{8} \end{pmatrix}$$

Lineare Ausgleichsprobleme

Bestimmtheit von Gleichungssystemen

Sei durch $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ ein LGS $Ax = b$ bestimmt, dann ist dieses LGS...

... eindeutig lösbar, wenn die Lösungsmenge $\mathbb{L} = \{x \in \mathbb{R}^n \mid Ax = b\}$ aus nur einem Element besteht

... quadratisch, wenn das Gleichungssystem genauso viele Gleichungen wie Unbekannte hat ($n = m$)

... überbestimmt, wenn das Gleichungssystem mehr Gleichungen als Unbekannte hat ($n > m$) → im Allg. nicht lösbar

... unterbestimmt, wenn das Gleichungssystem weniger Gleichungen als Unbekannte hat ($n < m$)

(lin. Abhängige Zeilen zählen nicht einfach!)

Mehrere Messungen → überbestimmte LGS → lin. Ausgleichsprobleme

Lineares Ausgleichsproblem

$A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^m$

→ Ges.: $x \in \mathbb{R}^n$, sodass das Residuum $r(x) = b - Ax$ minimiert wird:

$$\hat{x} = \underset{x \in \mathbb{R}^n}{\operatorname{argmin}}\, r(x)$$

mit Methode der kleinsten Quadrate

Suche ein $x \in \mathbb{R}^n$, sodass

$$\|r(x)\|_2 = \|b - Ax\|_2 = \|Ax - b\|_2$$

minimiert wird.

x ist nur dann eindeutig bestimmt, wenn $\operatorname{rang}(A) = m$ (Spalten von A lin. unabh.)

mit bekannter QR-Zerlegung

$$\operatorname{argmin} \|Ax - b\|_2 = \operatorname{argmin} \|Rx - Q^T b\|_2$$

→ obere m Gleichungen erfüllen & „verkleinertes System" lösen, Rest: Residuum

$$\varphi(x) = \|Ax-b\|_2^2 = \|Q^T\cdot(Ax-b)\|_2^2 = \|Rx-Q^Tb\|_2^2 \to \min$$

obere Dreiecksmatrix

$$\varphi(x) = \left\| \begin{pmatrix} \tilde{R}x \\ 0 \end{pmatrix} - \begin{pmatrix} (Q^Tb)^{(1)} \\ (Q^Tb)^{(2)} \end{pmatrix} \right\|_2^2$$

$$= \underbrace{\|\tilde{R}x - (Q^Tb)^{(1)}\|_2^2}_{\geq 0} + \underbrace{\|(Q^Tb)^{(2)}\|_2^2}_{\text{konst.}}$$

Problem: wenn $\tilde{R}$ trapezoid ist

Normalengleichung

$$x \to \|b-Ax\|_2^2 = \sum_{i=1}^{n}\left(b_i - \sum_{j=1}^{m} a_{ij}\,x_j\right)^2$$

$$A^TAx = A^Tb$$

mit A^TA: symmetr. nxm-Matrix

Wenn rang(A) = m → A^TA invertierbar, pos. definit

rang(A) < m → A^TA semidefinit, nicht eindeutig lösbar

Lin. Ausgleichsproblem (Beispiel)

Ges.: $g: t \to a_0 + a_1\cdot t$

mit Normalengleichung für $P_1(1,1)$, $P_2(3,2)$, $P_3(5,6)$, $P_4(7,8)$.

$g: t \to \underline{a_0} + \underline{a_1}\cdot t$

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 5 \\ 1 & 7 \end{pmatrix} \quad x = \begin{pmatrix} a_0 \\ a_1 \end{pmatrix} \quad b = \begin{pmatrix} 1 \\ 2 \\ 6 \\ 8 \end{pmatrix}$$

Normalengleichung: $A^TAx = A^Tb$

$$A^TA = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 5 & 7 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & 3 \\ 1 & 5 \\ 1 & 7 \end{pmatrix} = \begin{pmatrix} 4 & 16 \\ 16 & 84 \end{pmatrix}$$

$$A^Tb = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 3 & 5 & 7 \end{pmatrix}\begin{pmatrix} 1 \\ 2 \\ 6 \\ 8 \end{pmatrix} = \begin{pmatrix} 17 \\ 93 \end{pmatrix}$$

$$\Rightarrow A^TA\cdot x = A^Tb \qquad \begin{pmatrix} 4 & 16 \\ 16 & 84 \end{pmatrix}\begin{pmatrix} a_0 \\ a_1 \end{pmatrix} = \begin{pmatrix} 17 \\ 93 \end{pmatrix}$$

$$\left(\begin{array}{cc|c} 4 & 16 & 17 \\ 16 & 84 & 93 \end{array}\right) \xrightarrow{II - 4\cdot I} \left(\begin{array}{cc|c} 4 & 16 & 17 \\ 0 & 20 & 25 \end{array}\right) \to a_1 = \frac{5}{4}$$

$$a_0 = \left(17 - 16\cdot\tfrac{5}{4}\right)\cdot\tfrac{1}{4} = -\tfrac{3}{4}$$

$$\Rightarrow x = \begin{pmatrix} -\frac{3}{4} \\ \frac{5}{4} \end{pmatrix} = \frac{1}{4}\begin{pmatrix} -3 \\ 5 \end{pmatrix}$$

Matrizen – Datenstrukturen & prakt. Verfahren

dünn besetzte Matrizen effizient speichern → keine Nullwerte speichern

Im Folgenden: Indizierung mit 1 beginnend!

Row and Column Storage (RaCS)

val	enthält alle Nichtnullwerte der Matrix (nn_A)
col_ind	enthält Spaltenindizes
row_ind	enthält Zeilenindizes

$\Rightarrow val_k = A_{ij} \Leftrightarrow col\_ind_k = j \wedge row\_ind_k = i$

Bsp.:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$val = [5, 8, 3, 6]$

$col\_ind = [1, 2, 3, 2]$

$row\_ind = [2, 2, 3, 4]$

→ Speicheraufwand: $3 \times nn_A \ll m \cdot n$ (Spalten, Zeilen)

Compressed Row Storage (CRS)

val	
col_ind	
row_ptr	enthält Zeiger auf die anderen Arrays: Ab welchem Eintrag beginnt die nächste Zeile? letzter Eintrag: $nn_A + 1$ (# Zeilen + 1 Eintrag) leere Zeile: Pointer doppelt setzen (direkt hintereinander)

$\Rightarrow val_k = A_{ij} \Leftrightarrow col\_ind_k = j \wedge row\_ptr_i \leq k < row\_ptr_{i+1}$

Bsp.:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$val = [5, 8, 3, 6]$

$col\_ind = [1, 2, 3, 2]$

$row\_ptr = [1, 1, 3, 4, 5]$

→ Speicheraufwand: $2 \cdot nn_A + m + 1$

Nachteil: Einfügen & Löschen schwierig.
kein wahlfreier Zugriff auf einzelne Elemente

Block Compressed Row Storage (BCRS)

val	enthält alle Nichtnullblöcke der Größe n
col_ind	enthält Spaltenindizes der Matrixblöcke
row_ptr	Zeiger, der dem Zeilenindex erhöht (enthält # Zeilen + 1 Einträge, letzter nn_A+1)

Bsp.:

$$\begin{pmatrix} 4 & 3 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 6 \end{pmatrix}$$

$$val = \left[\begin{pmatrix} 4 & 3 \\ 5 & 8 \end{pmatrix}, \begin{pmatrix} 3 & 0 \\ 0 & 6 \end{pmatrix}\right]$$

$$col\_ind = [1, 2]$$

$$row\_ptr = [1, 2, 3]$$

Falls Blöcke sehr dicht besetzt sind, ist die normale Speicherung effizienter.

Compressed Column Storage (CCS)

analog zu CRS, aber mit col_ptr statt row_ptr

$$\Rightarrow val_u = A_{ij} \Leftrightarrow row\_ind_u = i \wedge col\_ptr_j \leq u < row\_ptr_{j+1}$$

Bsp.:

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

$$val = [5, 8, 6, 3]$$

$$row\_ind = [2, 2, 4, 3]$$

$$col\_ptr = [1, 2, 4, 5, 5]$$

vor allem sinnvoll, wenn $m \gg n$

Bsp.:

$$A_b = \begin{pmatrix} 1 & 0 & 3 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 5 & 0 & 0 & 8 \\ 0 & 1 & 0 & 0 & 0 & 4 \\ 1 & 2 & 7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \begin{matrix} r_i \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{matrix}$$

Vorgehen:

1. Wie lang muss jedes Feld sein?
2. Übertragen der Werte + row_ind
3. col_ptr

$$val = [1, 2, 1, 1, 1, 2, 3, 5, 7, 1, 1, 8, 4] \quad \#\ 13$$

$$row\_ind = [1, 2, 5, 3, 4, 5, 1, 3, 5, 2, 1, 3, 4] \quad \#\ 13$$

$$col\_ptr = [1, 4, 7, 10, 10, 11, 14] \quad \#\ 7$$

Matrizen und Graphen

$n \times n$-Matrix als Graph (n Knoten)

$A \in \mathbb{R}^{n \times n}$ mit $V = \{P_i \mid 1 \leq i \leq n\}$, $E = \{(P_i, P_j) \mid a_{ij} \neq 0\}$

$\rightarrow$ gerichteter Graph $G = (V, E)$ (E: gerichtete Kanten)

umgekehrt: Adjazenzmatrix eines gerichteten Graphen: (n Knoten)

$$a_{ij} = \begin{cases} 1, & \text{falls Kante von } P_i \text{ nach } P_j \\ 0, & \text{sonst} \end{cases}$$

in $n \times n$-Matrix

$\rightarrow$ geschichtetes Umsortieren $\rightarrow$ unabhängige Teilprobleme

Blockmatrizen

$\rightarrow$ effizienteres Speichern und Bereitstellen von Daten

Diskretisierung und Quantisierung

DISKRETISIERUNG

> Faltung

-> Wie verhält sich ein Signal bei der Übertragung über ein System?

Für $f, g: D \subseteq \mathbb{R}^n \to \mathbb{C}$:

$$(f * g)(x) := \int_D f(\tau) \cdot g(x-\tau)\, d\tau$$

für fast alle x wohldefiniert

falls $f, g \in \mathcal{L}^1(\mathbb{R}^n)$ -> f, g integrierbare Fkt. mit endlichem uneigentl. Betragsintegral -> immer erfüllt

Faltung von periodischen Funktionen

f, g periodische Fkt. der Periode T einer Veränderlichen

$$(f * g)(x) := \frac{1}{T} \int_a^{a+T} f(\tau) g(x-\tau)\, d\tau$$

$(f * g)$ wieder T-periodische Fkt.

Diskrete Faltung

$f, g: D \to \mathbb{C}$ Funktionen mit diskretem Definitionsbereich $D \subseteq \mathbb{Z}$

$$(f * g)(n) = \sum_{k \in D} f(k) g(n-k)$$

Rechengesetze

Die Menge $\mathcal{L}^1(\mathbb{R}^n)$ bildet zsm. mit der Addition & Faltung einen kommutativen Ring, d.h.:

- Kommutativität $f * g = g * f$
- Assoziativität $f * (g * h) = (f * g) * h$
- Distributivität $f * (g+h) = (f * g) + (f * h)$
- Assoziativität mit skalarer Multiplikation $\lambda (f * g) = (\lambda f) * g = f * (\lambda g)$

Glättungsfilter Tiefpass-, Bartlettfilter oder diskrete Gausskt.

-> positive Gewichte mit Summe 1

Kantendetektion Sobeloperator

Separierbare Filter → Matrixdarstellung Rang 1

$$[w(i,j)] = [u(i) \cdot v(j)]$$

($u(i)$: jede Zeile, $v(j)$: jeder Spalte)

→ effizient: $2hN^2$ Multiplikationen (statt h^2N^2)

2D Array $N \times N$

Filter $h \times h$

> Fourier-Transformation

harmonische Schwingungen: $e^{ihx} = \cos(hx) + i\sin(hx)$

mit Wellenzahl h, Wellenlänge $\lambda = \frac{2\pi}{h}$, Frequenz $\nu = \frac{h}{2\pi}$

Superpositionsdarstellung

Jede *Fkt. $f \in \mathrm{Abb}(\mathbb{R}, \mathbb{C})$ ist durch Superposition von harmon. Schwingungen darstellbar.

$$f(x) = \frac{1}{(2\pi)^n} \int_{\mathbb{R}^n} (\mathcal{F}f)(y)\, e^{iyx}\, dy$$

$\mathcal{F}f$ heißt Spektralfunktion oder Fourier-Transformierte von f.

Es gilt:

$$(\mathcal{F}f)(y) = \frac{1}{(2\pi)^{n/2}} \int_{\mathbb{R}^n} f(x)\, e^{-iyx}\, dx$$

* quadrat. integrierbar

Faltungssatz

Für faltbare Fkt. f, g gilt:

$$\mathcal{F}(f * g) = \sqrt{2\pi} \cdot \mathcal{F}f \cdot \mathcal{F}g \quad \text{bzw.} \quad \mathcal{F}f * \mathcal{F}g = \sqrt{2\pi}\, \mathcal{F}(f \cdot g)$$

> Abtasttheorem (Nyquist-Shannon)

Ist das Spektrum $\tilde{g}(w)$ einer kontinuierlichen Fkt. $g(x)$ bandbegrenzt, sprich gilt

$$\tilde{g}(w) = 0 \quad \text{für } |w| \geq w_{max},$$

und wählt man eine Schrittweite Δx mit $\Delta x \cdot w_{max} \leq \pi$, so kann $g(x)$ aus den Abtastwerten $(g(j \cdot \Delta x))_{j \in \mathbb{Z}}$ exakt rekonstruiert werden.

→ Abtastrate $\frac{1}{\Delta x}$ min. doppelt so groß wie max. Frequenz $\frac{w_{max}}{2\pi}$

> Aliasing und Anti-Aliasing

Aliasing → Artefakte durch „falsches" Abtasten

→ tritt auf, wenn das zu messende Signal hochfrequente Anteile enthält, die gemäß Abtasttheorem nicht reproduzierbar sind → Unterabtastung: niedrigere Frequenz wird vorgetäuscht

Antialiasing → Entfernen der höheren Frequenzen vor dem Abtasten

→ Multiplikation mit Tiefpass im Frequenzraum (X-Fkt.)

→ Falten mit sinc-Fkt. im Ortsraum

QUANTISIERUNG

→ approximiere $r \in \mathbb{R}$, $r_{min} \leq r \leq r_{max}$ im Wertebereich $W := [r_{min}, r_{max}]$

→ nahezu gleichverteilte Punkte: Teile Intervall in äquidistante Teilintervalle und nimm jeweils Mittelpunkt als Repräsentanten
→ uniforme Quantisierung

→ ungleich verteilte Punkte: verschiedene Strategien
- Median Cut
- Least Square Minimizer
- Maschinenzahlen
→ nicht uniforme Quantisierung

Gleitpunktzahlen und Maschinenzahlen

Menge der normalisierten t-stelligen Gleitpunktzahlen $FP_{B,t}$ zur Basis B

$$FP_{B,t} = \{ \pm M \cdot B^E \mid M, E \in \mathbb{Z}, E \text{ bel.}, B^{t-1} \leq M < B^t \text{ oder } M = 0 \}$$

M Mantisse Normalisierung durch $B^{t-1} \leq M$
E Exponent

Menge der Maschinenzahlen

$$MFP_{B,t,\alpha,\beta} = \{ q \in FP_{B,t} \mid \alpha \leq E \leq \beta \}$$

B, t, α, β durch Implementierung festgelegt, nie explizit gespeichert

→ Speichern von VZ, Mantisse, Exponent (in IEEE-754)

Rechnen mit Gleitpunktzahlen → Rundungsfehler
→ relativer Fehler $\rho = \frac{1}{B^{t-1}} = \frac{1}{2^{23}} \approx 1{,}19 \cdot 10^{-7}$
(auf Auflösung beschränkt)
→ absoluter Fehler kann durch log. Verteilung sehr groß werden

Diskretisierungs-, Iterationsfehler

Ergebnis einer Gleitpunktiteration $\stackrel{!}{=}$ exakte Resultatsberechnung mit Rundung

⇒ Es existiert ein $\tilde{\varepsilon} \in O(\rho)$, so dass

$$a *_{FP} b = (a * b) \cdot (1 + \varepsilon)$$

wobei $|\varepsilon| \leq |\varepsilon(*, a, b)| \leq \tilde{\varepsilon}|$

Bei Fließkommaoperationen gilt Kommutativität; Assoziativität und Distributivität aber nicht.

Bzgl. des rel. Fehlers: Mult. und Div. gutartig
Add. und Sub. gefährlich (Auslöschungseffekt)!

Kondition und Stabilität von Verfahren

Kondition eines Problems vom Verfahren unabhängig → Kondition Eigenschaft des Problems
Stabilität abhängig vom Verfahren
→ Anfälligkeit eines Verfahrens gegenüber Störungen

gut konditioniertes Problem
→ Fehler der Eingabedaten wird nicht verstärkt

$$\frac{|F(\tilde{x}) - F(x)|}{|F(x)|} \approx \frac{|\tilde{x} - x|}{|x|}$$

(vorwärts)stabiles Verfahren
→ Rechenfehler des Verfahrens akkumulieren nicht

$$\frac{|\tilde{F}(y) - F(y)|}{|F(y)|} \ll 1$$

→ arithmet. Grundoperationen immer stabil,
ABER: Hintereinanderausführung von stabilen Operationen nicht automatisch stabil

Akzeptable Ergebnisse

Ergebnis y' akzeptabel, wenn es als exakt betrachtetes Ergebnis zu nur leicht gestörten Eingabewerten verstanden werden kann

$\|x' - x\| \leq \varepsilon$ ⇒ rückwärtsstabiles Verfahren

→ x normale Eingabe, $F(x)$ exaktes Ergebnis
x' leicht gestörte Eingabe, $F(x')$ gestörtes Ergebnis
ε sehr klein

→ akzeptable Ergebnisse müssen nicht genau sein (abhängig von Konditionierung)

⊕ Gute Kondition

- Kleine Eingabestörungen sind unkritisch zu betrachten, da sie nur zu kleinen Störungen der Ergebnisse führen
- Es lohnt sich einen guten Algorithmus zu entwickeln, der die gute Kondition ausnutzt.

⊖ Schlechte Kondition

- Lösungen reagieren empfindlich auf kleinste Störungen der Eingabe
- Verbesserung des genutzten Verfahrens sinnlos (Verfahren ändert nichts an Kondition des Problems)
- Aufgabenstellung ist kritisch zu hinterfragen

Kondition

→ relative Kondition χ eines Problems $y = F(x)$ mit $F \in \text{Abb}(\mathbb{R}^m, \mathbb{R}^n)$ als die kleinste Zahl $K \geq 0$, für die gilt, dass $\delta > 0$ existiert, sodass für alle $\tilde{x}$ mit $0 < \|\tilde{x} - x\| < \delta$ gilt, dass

$$\frac{\|F(\tilde{x}) - F(x)\|}{\|F(x)\|} \leq \chi \cdot \frac{\|\tilde{x} - x\|}{\|x\|}$$

Konditionszahl einer Matrix

Kondition einer (invertierbaren) Matrix $A \in \mathbb{R}^{n \times n}$

$$K(A) = \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|} \Rightarrow \chi(AB) \leq \chi(A) \cdot K(B)$$

→ LR mit Pivot & QR Zerlegung stabil → bei QR:
$\chi(A) = \chi(Q \cdot R) = \chi(R)$

> Quantisierung von Farbwerten mit dem median-cut-algorithm

Eingabe: Punktewolke P in Gebiet G, Anzahl an Schritten N
Globale Variablen: s aktuelle Anzahl an Schritten, vorinitialisiert mit 0

① Finde eine am besten passende umgebende Box der Wolke P
→ Rechteck/Quader
② Quader an der längsten Kante teilen, sodass beide Teile (in etwa) gleich viele Punkte enthalten → Median bilden
③ Falls $s \geq N$ Stopp, sonst ①

Wahl der Repräsentanten → Mittelwert der enthaltenen Punkte, oder Mittelpunkt der Quader

> Vektorquantisierung

Minimierer der geringsten Quadrate (least square minimizer) in 1D

→ nicht gleichverteilte Werte mit nicht äquidistanter Schrittweite und einem Repräsentanten aus einem Schrittweitenintervall

$$P(r \leq R) = \int_{-\infty}^{R} p(\rho)\, d\rho$$

⇝ Zufallsvariable r $\Rightarrow$ $p(\rho) = 0$, wenn $r_{max} < 0 < r_{min}$

$\Rightarrow$ mittl. quadrat. Fehler $\varepsilon = \sum_{j=0}^{L-1} \int_{w_j}^{w_{j+1}} (\rho - r_j)^2 p(\rho)\, d\rho$

→ ε wird minimiert, falls die Bedingung der nächsten Nachbarn

$$w_j = \frac{r_{j-1} + r_j}{2}$$

und die Schwerpunktbedingung

$$r_j = \frac{\int_{w_j}^{w_{j+1}} \rho \cdot p(\rho)\, d\rho}{\int_{w_j}^{w_{j+1}} p(\rho)\, d\rho}$$

gelten.

Mehrdimensionale Approximation

$r \in \mathbb{R}^n$ mit $v_{i,min} \leq r_i \leq v_{i,max}$ und $1 \leq i \leq n$ approximieren

$\Rightarrow$ Wertebereich $W := \bigtimes_{i=1}^{n} [v_{i,min}, v_{i,max}]$

L Vektoren r_i = Repräsentanten der disjunkten Teilgebiete W_i

$\Rightarrow$ Orientierungsfehler $\varepsilon = \sum_{i=1}^{L} \int_{W_i} \|r - r_i\|^2 \cdot p(r)\, dr$

mit r-dim. Dichte $p(r)$

Fehlerfunktional ε wird für gegebene Repräsentanten r_i mit $1 \leq i \leq L$ minimiert, wenn $r \in W_i$ gilt, dass

$$\forall j \neq i \quad \|r - r_i\| \leq \|r - r_j\|$$

Zerlegung von W in disjunkte W_i gegeben:

$$r_i = \frac{\int_{W_i} r \cdot p(r)\, dr}{\int_{W_i} p(r)\, dr}$$

Iteratives Verfahren zur Bestimmung der optimalen Vektorquantisierung

je M n-dim Trainingsvektoren x_i und L n-dim. Kodevektoren r_i:
Partition P des $\mathbb{R}^n$
$Q \in Abb(\mathbb{R}^n, \mathbb{R}^n)$ Trainingsvektor → Kodevektor
x_i in P_j → x_i wird durch $Q(x_i) = r_j$ approximiert

→ Fehler $\varepsilon = \frac{1}{M \cdot n} \cdot \sum_{i=1}^{M} \|x_i - Q(x_i)\|^2$ soll minimiert werden

Optimalitätskriterien

↳ nearest neighbor condition

$$P_i = \{x \mid \forall 1 \le h \le L : \|x - r_i\|^2 \le \|x - r_h\|^2\}$$

↳ centroid condition

$$r_i = \frac{1}{|M_i|} \cdot \sum_{m \in M_i} x_m$$

mit $M_i = \{m \mid x_m \in P_i\}$

Lloyds-Algorithmus (k-means clustering)

① Berechne mit

$$r_1^* = \frac{1}{M} \cdot \sum_{i=1}^{M} x_i \qquad \varepsilon^* = \frac{1}{M \cdot n} \cdot \sum_{i=1}^{M} \|x_i - r_i^*\|$$

unter der Annahme, dass $P_1 = \mathbb{R}^n$ und $M_1 = \{1, \dots, M\}$, eine Startlösung.

② Kodevektoren aufspalten in

$$r_i^{(0)} = (1+\beta) \cdot r_i^* \quad , \quad r_{i+L}^{(0)} = (1-\beta) \cdot r_i^*$$

③ Samples den neuen Kodevektoren zuordnen
→ bestimme P_i, M_i

④ Berechne die neuen Kodevektoren r_i

⑤ Solange der Abstand der alten zu den neuen Kodevektoren größer ist als eine vorher festgelegte Schranke → ③

⑥ Falls die gewünschte Anzahl Kodevektoren noch nicht erreicht → ②

Rekonstruktion kontinuierlicher Daten

Interpolation und Approximation

Interpolation

Bekannt seien n ($n \in \mathbb{N}$) Punkte im Raum. Soll die rekonstruierte Funktion nun **exakt** durch die Punkte verlaufen, so spricht man von **Interpolation**. Interpolation ist nicht in jedem Fall das bessere Verfahren.

Approximation

Bekannt seien n ($n \in \mathbb{N}$) Punkte im Raum. Soll die rekonstruierte Funktion nun nur **näherungsweise** durch die Punkte verlaufen, so spricht man von **Approximation**. Vorteil: Besserer Umgang mit Messfehlern.

Interpolation

unbekannte Funktion $f: \mathbb{R} \to \mathbb{R}$

- → bekannt: Stützstellen x_i mit Stützwerten y_i $(1 \leq i \leq n)$
- → f aus bekannten Punkten näherungsweise bestimmen
- → Wähle eine Klasse K von Funktionen zum Starten der Rekonstruktion
  - ↳ Voraussetzungen for K:
    - f in K gut approximierbar
    - K auf Computern gut darstellbar und manipulierbar
    - Polynome werden oft stückweise verwendet

Lokale Verfahren

- interpolierter Wert $p(x)$ hängt nur von „benachbarten" Werten ab
  - → z.B.: $x_i < x < x_{i+1}$
    - → nur y_i und y_{i+1} gehen ein
- nearest neighbor (stückweise konstant)
- linear (stückweise linear)
- Catmull-Rom (stückweise kubisches Polynom)

Globale Verfahren

- alle Werte y_i gehen ein
- Polynom-Interpolation
- B-Spline-Interpolation

> Stückweise konstante Interpolation nearest neighbor

→ suche nächsten Nachbarn = Stützstelle x_j

→ interpolierter Wert $= y_j$

$$p(x) = \begin{cases} y_1 & x_1 \le x \le \frac{1}{2}(x_1+x_2) \\ y_i & \frac{1}{2}(x_{i-1}+x_i) < x \le \frac{1}{2}(x_i+x_{i+1}) \\ y_n & \frac{1}{2}(x_{n-1}+x_n) < x \le x_n \end{cases} \quad \text{for alle } 2 \le i \le n-1$$

Rechenaufwand: Nur Suche nach nächsten Nachbar

→ äquidistant: $x_i = a + ih$

$$i = \text{int}((x-a)/h) \rightarrow O(1)$$

→ geordnete Stützstellen: $O(\log(n))$

→ sonst: $O(n \log(n))$

in vielen Fällen zu ungenau

Fehler bei glatten f:

$$|p(x) - f(x)| \le \frac{h}{2} \cdot \max_{x_1 \le \xi \le x_n} \{|f'(\xi)|\} \quad \text{mit } h = \max_{1 \le i \le n} \{x_{i+1} - x_i\}$$

> Stückweise lineare Interpolation

→ suche nächst gelegene linke und rechte Stützstelle $x_i \le x \le x_{i+1}$

→ interpoliere im Intervall $[x_i, x_{i+1}]$ linear

$$p_i(x) = m_i(x - x_i) + y_i \quad \text{mit} \quad m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

Alternativ gewichtetes Mittel von y_i und y_{i+1}:

$$p(x) = w_1 y_i + w_2 y_{i+1} \quad \text{mit} \quad w_1 = \frac{x_{i+1} - x}{x_{i+1} - x_i}, \quad w_2 = \frac{x - x_i}{x_{i+1} - x_i} = 1 - w_1$$

> Catmull-Rom: stückweise kubische Interpolation

1. Schätze an jeder Stelle die erste Ableitung (Steigung)
   → $\{y_1', y_2', \dots, y_n'\}$
2. Finde auf jedem Teilintervall $[x_i, x_{i+1}]$ das (eindeutige) kubische Polynom p_i, welches in den beiden Endpunkten die Stützwerte und die geschätzten Ableitungen interpoliert:

$$p(x) = a_0(x_{i+1} - x)^3 + a_1(x_{i+1} - x)^2(x - x_i) + a_2(x_{i+1} - x)(x - x_i)^2 + a_3(x - x_i)^3$$

mit

$$a_0 = \frac{y_i}{(x_{i+1} - x_i)^3}, \quad a_1 = 3a_0 + \frac{y_i'}{(x_{i+1} - x_i)^2},$$

$$a_2 = 3a_3 - \frac{y_{i+1}'}{(x_{i+1} - x_i)^2}, \quad a_3 = \frac{y_{i+1}}{(x_{i+1} - x_i)^3}$$

Geschätzte Ableitungen:

- Vorwärts-Differenz $f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$

- Rückwärts-Differenz $f'(x) = \frac{f(x) - f(x-h)}{h} + O(h)$

- Zentrale Differenz $f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$

↳ exakt (gewichtetes Mittel von Vorwärts- und Rückwärts-D)

$$y_i' = \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}} \cdot y_i'_{fw} + \frac{x_{i+1} - x_i}{x_{i+1} - x_{i-1}} \cdot y_i'_{bw}$$

> Fehlerabschätzung (lokale Interpolation)

Annahme: äquidistante Stützstellen mit Schrittweite h, f genügend diffbar

Nearest Neighbor	$O(h)$	$\leq \lvert f'(\xi)\rvert / 2h$
Lineare Interpolation	$O(h^2)$	$\leq \lvert f''(\xi)\rvert / 8h^2$
Catmull Rom:	$O(h^3)$	$\leq \lvert f'''(\xi)\rvert / 24h^3$

(auch gültig, wenn h = max. Abstand aufeinanderfolgender Stützstellen)

> Polynominterpolation

Polynome vom Grad $n-1$ bilden einen n-dim. Vektorraum

$$\mathbb{P}_n = \text{span}\{1, x, x^2, \dots, x^{n-1}\} = \left\{\sum_{i=0}^{n-1} c_i x^i\right\}$$ Taylorbasis, Monombasis

→ Vandermonde

$$\underbrace{\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-1} \end{bmatrix}}_{c} = \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}}_{y}$$

Vandermonde-Matrix

quadrat., voll besetzt, nicht singulär (alle $n+1$ Stützstellen paarweise verschieden)

$$\det(A) = \prod_{1 \leq j < k \leq n} (x_j - x_k) \neq 0$$

schon bei moderatem n (und äquidistanten Stützstellen) schlecht konditioniert

⇒ Zu n Stützstellen gibt es ein eindeutig bestimmtes Interpolationspolynom vom Grad $n-1$.

→ Lösen mit LR-Zerlegung
- → $O(n^3)$
- → schlecht konditioniert

Rekursionsformel von Aitken-Neville

$$p_{i,n}(x) = \begin{cases} y_i & , n = 0 \\ p_{i,n-1}(x) \cdot \frac{x_{i+n} - x}{x_{i+n} - x_i} + p_{i+1,n-1}(x) \cdot \frac{x - x_i}{x_{i+n} - x_i} & , \text{sonst} \end{cases}$$

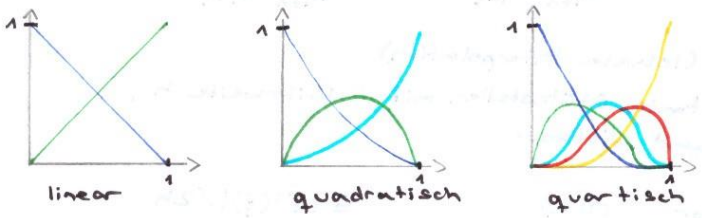
Alternative Polynombasen

> Bernsteinpolynombasen

$$B_i^{n-1}(x) = \binom{n-1}{i} (1-x)^{n-1-i} x^i$$

Basis für Polynome vom Grad $n-1$

$$\mathbb{P}_{n-1} = \text{span}\{B_0^{n-1}(x), B_1^{n-1}(x), \dots, B_{n-1}^{n-1}(x)\}$$



Eigenschaften:

$B_0^n(x) \geq 0$ falls $0 < x < 1$

$\sum_{i=0}^{n} B_i^n(x) = 1$

Formeln s. Folien

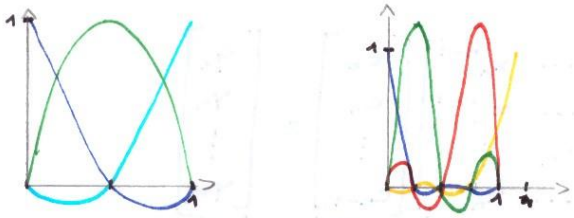
→ $(1-x)$, x, Pascal'sches Dreieck

⇝ Für Polynominterpolation eher nicht geeignet

> Lagrangepolynombasen

$$L_i(x) = \frac{\prod_{\substack{j=1 \\ j \neq i}} (x - x_j)}{\prod_{\substack{j=1 \\ j \neq i}}^{n} (x_i - x_j)} \qquad (i = 1, \dots, n)$$

$$L_i(x_j) = \begin{cases} 1 & , j = i \\ 0 & , \text{sonst} \end{cases}$$



L_i sind lin. unabh. → Basis von $\mathbb{P}_n$

$L_i(x_j) = \delta_{ij}$

$$p(x) = \sum_{i=1}^{n} y_i L_i(x)$$

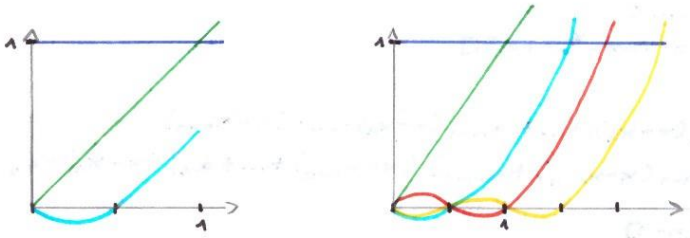
Hinzufügen eines neuen Punktes → alles muss neu berechnet werden (jedes L_i hängt von allen Stützpunkten ab)

> Newtonpolynombasen

$$N_i(x) = \prod_{j=1}^{i-1} (x - x_j) \qquad 1 \leq i \leq n$$

$$p(x) = \sum_{i=1}^{n} a_{i-1} \cdot N_i(x) \quad \rightarrow a_j: \text{ Algorithmus von Aitken-Neville}$$

→ für alle x effizient auswertbar



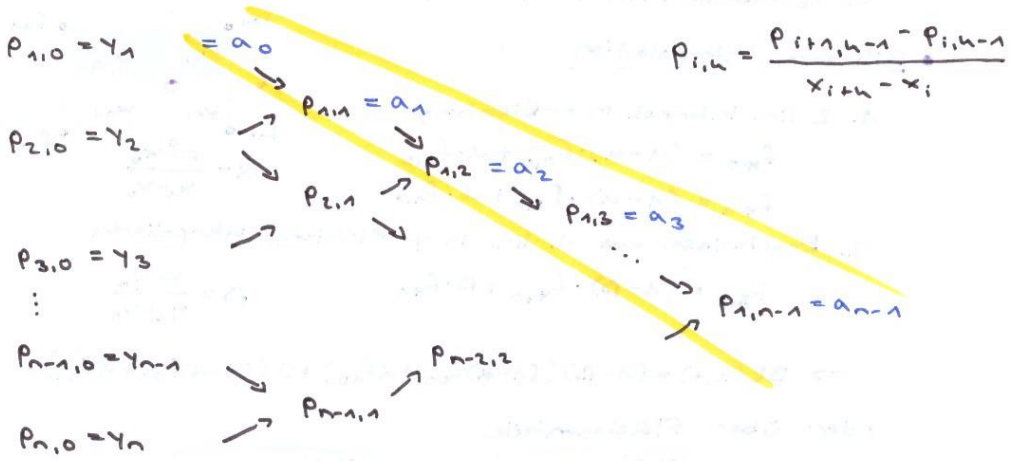
```
def evaluateNewton(x, a, xi, n):
    eva = a[n-1]
    for i in range(n-2, -1, -1):
        eva = eva * (x-xi[i]) + a[i]
    return eva
```

Algorithmus von Aitken-Neville

Eingabe: Stützstellen x_i, Stützwerte y_i mit $1 \leq i \leq n$
Ausgabe: Koeffizienten a_i mit $1 \leq i \leq n$
Komplexität: $O(n^2)$

$$P = (p_{i,k})_{\substack{1 \leq i \leq n-k \\ 0 \leq k \leq n-1}}$$

$p_{1:n,0} = y_{1:n}$
for $k = 1$ to n do
  for $i = 1$ to $n-k$ do
    $p_{i,k} = \frac{p_{i+1,k-1} - p_{i,k-1}}{x_{i+k} - x_i}$
  end for
end for
$a = p_1$



Effiziente Auswertung des Newton-Polynoms mit Horner-artigem Schema:
→ klassisch für $p(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1} = (\dots(a_{n-1} x + a_{n-2})x + \dots + a_1)$
$\cdot x + a_0$

```
sum = a[n-1]
for i=n-2 ... 0
    sum = sum * x + a[i]
```

→ modifiziert

$$p(x) = a_0 + a_1(x-x_1) + \dots + a_{n-1}(x-x_1)\cdot\dots\cdot(x-x_{n-1})$$
$$= (\dots(a_{n-1}(x-x_{n-1}) + a_{n-2})(x-x_{n-2}) + \dots + a_1)(x-x_1) + a_0$$

```
sum = a[n-1]
for i = n-2 ... 0
    sum = sum * (x - x[i+1]) + a[i]
```

Multivariate Interpolation

- Funktionen mehrerer Veränderlicher $f: \mathbb{R}^n \to \mathbb{R}$
  - bivariat $f(x,y)$
  - trivariat $f(x,y,z)$
- Stützstellen: (x_i, y_i) → Stützwerte $f_i = f(x_i, y_i)$

→ Konstruiere eine interpolierende Funktion g: $g(x_i, y_i) = f_i$

- 2D-Gitter: Ecken $V = \{V_i\}$ — geometr. Eigenschaften → Lage
  Kanten $E = \{E_{ij}\}$, Zellen $F = \{F_i\}$ — topologische Eigenschaften → Nachbarschaftsbeziehungen

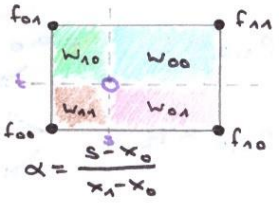
→ planare Gitter ohne Löcher: $|F| - |E| + |V| = 1$
(Betrag = Anzahl)

⇒ Datenstrukturen: 2 Listen
Liste der Punkte (Koordinaten der Ecken)
Liste der Zellen → Verweise auf vertex list

> Lokale Interpolationsverfahren

→ nur Daten in der Umgebung des jeweiligen Punktes berücksichtigen
↳ zellenweise: nur Werte in den Eckpunkten der Zellen

• Bilineare Interpolation



$$\alpha = \frac{s - x_0}{x_1 - x_0}$$

1. 2 lin. Interpol. in x-Richtung

$$f_{*0} = (1-\alpha)\cdot f_{0,0} + \alpha \cdot f_{1,0}$$
$$f_{*1} = (1-\alpha)\cdot f_{0,1} + \alpha \cdot f_{1,1}$$

2. Ergebnisse aus 1. lin. in y-Richtung interpolieren

$$f_{s,t} = (1-\beta)\cdot f_{*,0} + \beta \cdot f_{*,1} \qquad \beta = \frac{t - y_0}{y_1 - y_0}$$

$$\Rightarrow BL(s,t) = (1-\beta)((1-\alpha)f_{00} + \alpha f_{10}) + \beta((1-\alpha)f_{01} + \alpha f_{11})$$

oder über Flächenanteile

$$BL(s,t) = \frac{y_1 - t}{y_1 - y_0}\cdot\frac{x_1 - s}{x_1 - x_0}\cdot f_{0,0} + \frac{y_1 - t}{y_1 - y_0}\cdot\frac{s - x_0}{x_1 - x_0}\cdot f_{1,0}$$
$$+ \frac{t - y_0}{y_1 - y_0}\cdot\frac{s - x_0}{x_1 - x_0}\cdot f_{1,1} + \frac{t - y_0}{y_1 - y_0}\cdot\frac{x_1 - s}{x_1 - x_0}\cdot f_{0,1}$$

$$= w_{00} \cdot f_{0,0} + w_{1,0} \cdot f_{1,0} + w_{1,1} \cdot f_{1,1} + w_{0,1} \cdot f_{0,1}$$

Reihenfolge kann auch vertauscht werden. (Erst y-, dann x-Richtung)

Bivariate Polynome auf Tensorprodukten

Liegen die Stützstellen auf einem Tensorproduktgitter, so kann die Interpolation zuerst in x-Richtung dann in y-Richtung – oder umgekehrt – jeweils als eine 1D Interpolation durchgeführt werden.

Multilineare Interpolation im n-D Raum

Es gibt $2^{n-1} + 2^{n-2} + \dots + 2 + 1$ lin. Interpolationen in alle n Raumrichtungen.

Baryzentrische Koordinaten

3 Punkte R, S, T in der Ebene, nicht auf einer Geraden
→ jeder Punkt der Ebene:

$$P = \rho R + \sigma S + \tau T$$

wobei gelten muss $1 = \rho + \sigma + \tau$

(ρ, σ, τ) sind die baryzentrischen Koordinaten von P bzgl. $\Delta(R,S,T)$

Lösen des Gleichungssystems:

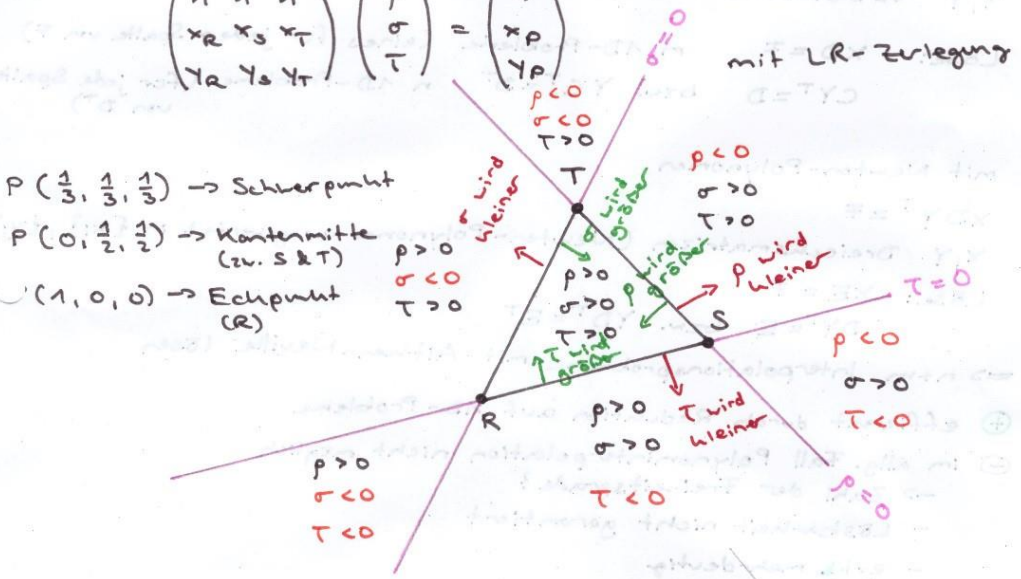
$$\begin{pmatrix} 1 & 1 & 1 \\ x_R & x_S & x_T \\ y_R & y_S & y_T \end{pmatrix} \begin{pmatrix} \rho \\ \sigma \\ \tau \end{pmatrix} = \begin{pmatrix} 1 \\ x_P \\ y_P \end{pmatrix}$$

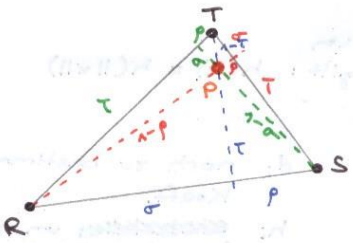
mit LR-Zerlegung

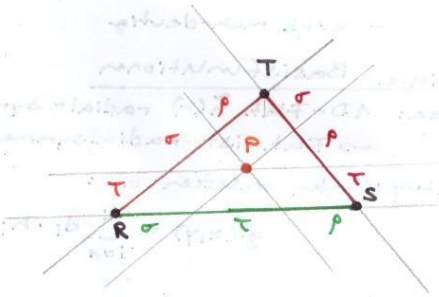
$P\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$ → Schwerpunkt

$P\left(0, \frac{1}{2}, \frac{1}{2}\right)$ → Kantenmitte (zw. S & T)

$(1, 0, 0)$ → Eckpunkt (R)







Geometr. Interpolation:

$$\rho = \frac{\text{area}(\Delta(P,S,T))}{\text{area}(\Delta(P,S,T))} \qquad \sigma = \frac{\text{area}(\Delta(R,P,T))}{\text{area}(\Delta(P,S,T))}$$

$$\tau = \frac{\text{area}(\Delta(R,S,P))}{\text{area}(\Delta(P,S,T))}$$

Lineare Interpolation

Dreieck $\Delta(R,S,T)$ durch f_R, f_S, f_T

Wert des lin. Interpolanten an einer Stelle $P \in \Delta(R,S,T)$

$$f_P = \rho \cdot f_R + \sigma \cdot f_S + \tau \cdot f_T$$

in 3D: Tetraeder

$$P = \rho R + \sigma S + \tau T + \nu U$$

$$\rho + \sigma + \tau + \nu = 1$$

4× in x-, 2× in y-, 1× in z-Richtung

Tensor-Produktansatz

$$XCY^T = F$$

X, Y Vandermonde-Matrizen zu Stützstellen $\{x_i\}, \{y_j\}$

Löse: $XD = F$ m 1D-Probleme (eines für jede Spalte von F)

$CY^T = D$ bzw. $YC^T = D^T$ n 1D-Probleme (für jede Spalte von D^T)

mit Newton-Polynomen

$$XDY^T = F$$

X, Y Dreiecksmatrizen (Newton-Polynome ausgewertet in $\{x_i\}, \{y_j\}$)

Löse: $XE = F$

$DY^T = E$ bzw. $YD^T = E^T$

⇒ n+m Interpolationsprobleme mit Aithen-Neville lösen

⊕ effizient durch Reduktion auf 1D-Probleme

⊖ im allg. Fall Polynominterpolation nicht möglich

→ Zahl der Freiheitsgrade?

- Lösbarkeit nicht garantiert
- evtl. mehrdeutig

Radiale Basisfunktionen

Idee: 1D-Fkt. $\tilde{h}(r)$ radial-sym. fortsetzen

→ Fkt. ist radialsymmetr., wenn gilt: $h(x) = h(\|x\|)$

Interpolanten ansetzen als:

$$g(x,y) = \sum_{i=1}^{n} d_i \cdot h_i(x,y)$$

d_i: noch zu bestimmende Koeff.

h_i: um Stützstelle (x_i, y_i) zentrierte Basisfkt.

$\tilde{h}(r)$ 1D → mit $h_0(x,y) = \tilde{h}(\sqrt{x^2+y^2})$ radialsym. fortgesetzt

für um die Stützstelle zentrierte radiale Basisfunktion

$$h_i(x,y) = h_0(x-x_i, y-y_i)$$

Koeff. bestimmen durch n Interpolationsbedingungen

$$g(x_j,y_j) = \sum_{i=1}^{n} d_i \cdot h_i(x_j,y_j) = f_j$$

Radiale Basisfunktion mit linearer Präzision

→ lin. Fkt. werden exakt rekonstruiert

Ansatz: $f(x,y) = a + bx + cy + \sum_{i=1}^{n} d_i h_i(x,y)$

→ alle Momente der Ordnungen ≤ 1 verschwinden

Koeff. durch Lösen von: $(n+3) \times (n+3)$ GS

$$\begin{pmatrix} 0 & 0 & 0 & \Sigma_j h_1(x_j,y_j) & \cdots & \Sigma_j h_n(x_j,y_j) \\ 0 & 0 & 0 & \Sigma_j x_j h_1(x_j,y_j) & \cdots & \Sigma_j x_j h_n(x_j,y_j) \\ 0 & 0 & 0 & \Sigma_j y_j h_1(x_j,y_j) & \cdots & \Sigma_j y_j h_n(x_j,y_j) \\ 1 & x_1 & y_1 & h_1(x_1,y_1) & \cdots & h_n(x_1,y_1) \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & y_n & h_1(x_n,y_n) & \cdots & h_n(x_n,y_n) \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d_1 \\ \vdots \\ d_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}$$

Freiformmodellierung und Bézierkurven

Lokalität

Eine lokale Veränderung der Eingabedaten (z.B. eines Stützwertes) darf möglichst nur lokale Auswirkungen haben

→ Kurvengestalt ändert sich idealerweise nur in einem kleinen Bereich um den Stützpunkt

→ Catmull-Rom

→ Bézier-Kurven

Bernstein-Polynome

$$B_i^n(x) = \binom{n}{i}(1-x)^{n-i} \cdot x^i \qquad (i = 0,1,2,\ldots,n)$$

$$P_{n+1} = \text{span}\{B_i^n(x) \mid 0 \leq i \leq n\}$$

für $t \in [0,1]$: $0 \leq B_i^n(t) \leq 1$ und $B_i^n(t)$ i-fache NS in $t=0$

$(n-i)$-fache NS in $t=1$

für alle t: $\sum_{i=0}^{n} B_i^n(t) = 1$

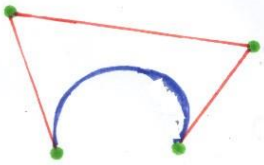
Bezierkurven

Geg.: Kontrollpunkte $B = \{b_0,\ldots,b_n\} \subset \mathbb{R}^d$

→ Kurve C im $\mathbb{R}^d$ vom Grad n = Bezierkurve

$$C(t) = \sum_{i=0}^{n} b_i B_i^n(t) \quad \text{mit } t \in [0,1]$$

Verbindet man die Bézier-Punkte durch einen Polygonzug, erhält man das Kontrollpolygon der Bézier-Kurve.



Geometrie des Kontrollpolygons
→ grob Geometrie der Kurve

Formeigenschaften ⇒ Modifikation der Kontrollpunkte führt zu vorhersehbarer (intuitiver) Änderung der Bezier-Kurve

1. Interpolation der Endpunkte
   → $C(0) = b_0$, $C(1) = b_n$
2. Tangentiale Tangentenbedingung
   → In den Endpunkten nähert sich die Kurve tangential an das Kontrollpolygon an
   $C'(0) = n(b_1 - b_0)$, $C'(1) = n(b_n - b_{n-1})$
3. Konvexe Hülle
   → Bezierkurve liegt in konvexer Hülle der Kontrollpunkte
4. Affine Invarianz
   → Affine Abbildung: $\Phi(x) = Ax + b$
   Soll eine Bézierkurve affin transformiert werden, dann müssen nur die Kontrollpunkte affin transformiert werden
   $$C(t) = \sum_{i=0}^{n} b_i \cdot B_i^n(t)$$
   $$\rightarrow \Phi(C(t)) = \sum_{i=0}^{n} \Phi(b_i) \cdot B_i^n(t)$$
   oder: transformierte Kontrollpunkte erzeugen die transformierte Bézier kurve
5. Variationsreduzierend
   → Kurve schwankt höchstens so stark wie das Kontrollpolygon
   Für jede Gerade g gilt:
   #(Schnittpunkte g mit Kurve) ≤ #(Schnittpunkte von g mit Kontrollpolygon)

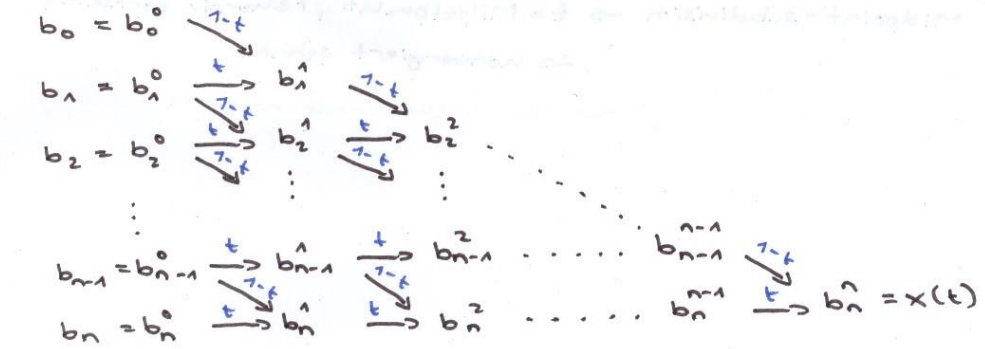
→ Auswertung

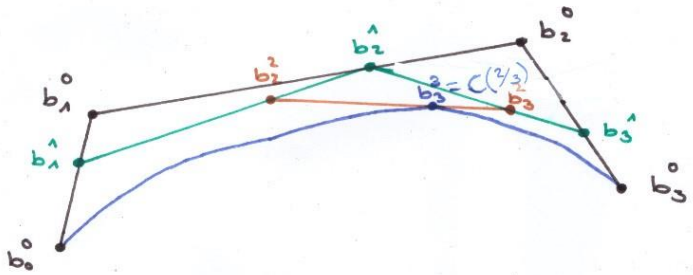
naive Auswertung der Bernsteinpolynome teuer, ineffizient
Horner-Bézier: etwas effizienter
fortgesetzte lin. Interpolation mit de Casteljau: stabil

```
def de_casteljau(P, t):
    n = P.shape[0]
    b = [redacted] np.zeros((n,n))
```

```
b[0] = P
for h in range (1,n):
    for i in range (h,n):
        b[h,i] = (1-t) * b[h-1,i-1]
        b[h,i] += t * b[h-1, i]
return b[n-1, n-1]
```

de Casteljau-Algorithmus





geometrisch:
für $t = \frac{2}{3}$
$n = 3$

Beispiel:

$b_0 = \begin{pmatrix} 0 \\ 9 \end{pmatrix}$ $b_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ $b_2 = \begin{pmatrix} 9 \\ 0 \end{pmatrix}$ $b_3 = \begin{pmatrix} 12 \\ 6 \end{pmatrix}$ $t = \frac{1}{3}$

$\begin{pmatrix} 0 \\ 9 \end{pmatrix} \xrightarrow{2/3} \begin{pmatrix} 0 \\ 6 \end{pmatrix}$

$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \xrightarrow{1/3} \begin{pmatrix} 3 \\ 0 \end{pmatrix}$

$\begin{pmatrix} 9 \\ 0 \end{pmatrix} \xrightarrow{1/3} \begin{pmatrix} 10 \\ 2 \end{pmatrix}$

$\begin{pmatrix} 12 \\ 6 \end{pmatrix}$

$\begin{pmatrix} 3 \\ 0 \end{pmatrix} \to \begin{pmatrix} 7 \\ 4 \end{pmatrix}$

$\begin{pmatrix} 10 \\ 2 \end{pmatrix} \to \begin{pmatrix} 5\frac{1}{3} \\ 2\frac{2}{3} \end{pmatrix}$

$\begin{pmatrix} 7 \\ 4 \end{pmatrix} \to \begin{pmatrix} 22/3 \\ 3 \end{pmatrix}$ $\Rightarrow C(\frac{1}{3}) = \begin{pmatrix} 22/3 \\ 3 \end{pmatrix}$

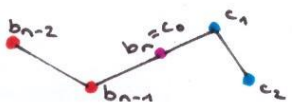
Glatter Übergang zwischen benachbarten Bézier-Kurven

letzter Kontrollpunkt = erster Kontrollpunkt

$C(t) = \sum_i b_i B_i^n(t)$ $D(t) = \sum_i c_i B_i^n(t)$ $\overset{\text{stetig}}{\Rightarrow} b_n = c_0$

$C'(1) = n(b_n - b_{n-1})$ & $D'(0) = n(c_1 - c_0)$

$\Rightarrow C^1$-Stetigkeit



-> noch stärkere Glattheit: C^2-Stetigkeit

2 Polynome n-ten Grades -> max. C^{n-1}-Stetigkeit

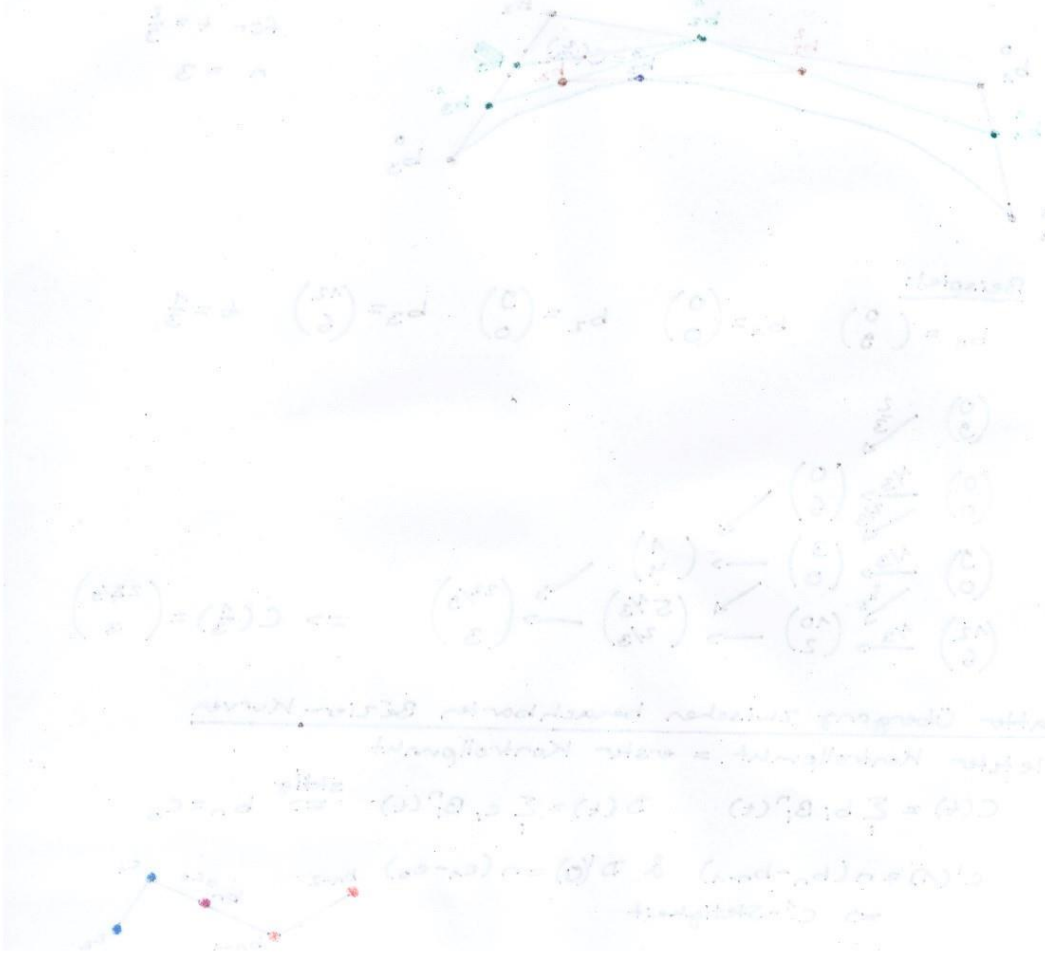
=> B-Spline-Kurven

Für Bézierflächen gelten dieselben Eigenschaften wie for Bézier-Kurven außer die Variationsreduktion.

Midpoint-subdivision -> t = Mittelpunkt, rekursiv fortsetzen

-> konvergiert schnell

~> Teilstrecken immer halbieren



Beispiel: Lagrange-Interpolation

$$P_n(x) = \sum_{k=0}^{n} \underbrace{f_k}_{\text{einfach y-Werte}} \cdot L_k(x) \quad \text{mit} \quad L_k(x) = \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j}$$

$P_0 = (-1, 1) \rightarrow L_0 = \frac{x-0}{-1-0} \cdot \frac{x-1}{-1-1} \cdot \frac{x-3}{-1-3} = -\frac{1}{8} \cdot x(x-1)(x-3)$

$P_1 = (0, 3) \rightarrow L_1 = \frac{x+1}{0+1} \cdot \frac{x-1}{0-1} \cdot \frac{x-3}{0-3} = \frac{1}{3} \cdot (x+1)(x-1)(x-3)$

$P_2 = (1, -7) \rightarrow L_2 = \frac{x+1}{1+1} \cdot \frac{x-0}{1-0} \cdot \frac{x-3}{1-3} = -\frac{1}{4}(x+1) \cdot x(x-3)$

$P_3 = (3, 3) \rightarrow L_3 = \frac{x+1}{3+1} \cdot \frac{x-0}{3-0} \cdot \frac{x-1}{3-1} = \frac{1}{24}(x+1) \cdot x(x-1)$

$$P_3(x) = \frac{11}{4}x^3 - 6x^2 - \frac{27}{4}x + 3$$

(aus: $P_3(x) = -\frac{1}{8}x(x-1)(x-3) + (x+1)(x-1)(x-3) + \frac{3}{4}(x+1)x(x-3) + \frac{1}{8}(x+1)x(x-1)$)

Beispiel: Newton-Interpolation

$$P_n(x) = \sum_{k=0}^{n} c_k \cdot N_k(x) \quad \text{mit} \quad N_k(x) = \prod_{j=0}^{k-1} (x - x_j)$$

x_k	y_k		
-1	1	$\frac{3-1}{0-(-1)} = 2$	$\frac{-10-2}{1-(-1)} = -6$
0	3	$\frac{-7-3}{1-0} = -10$	$\frac{5-(-10)}{3-0} = 5$
1	-7		$\frac{5-(-6)}{3-(-1)} = \frac{11}{4}$
3	3		

benachbarte y-Werte subtrahiert durch benachbarte x-Werte subtrahiert

benachbarte neue Werte subtrahiert durch äußere x-Werte

$$P_3(x) = \underbrace{1}_{N_1=1} + \underbrace{2(x+1)}_{N_2} + \underbrace{(-6)(x+1)(x-0)}_{N_3} + \underbrace{\frac{11}{4}(x+1)(x-0)(x-1)}_{N_4}$$

$$= \frac{11}{4}x^3 - 6x^2 - \frac{27}{4}x + 3$$

→ neue Punkte können einfach hinzugefügt werden

Beispiel: Bilineare Interpolation:

$$f(x,y_1) = \frac{x_2 - x}{x_2 - x_1} \cdot v_{11} + \frac{x - x_1}{x_2 - x_1} \cdot v_{21}$$

entlang wird interpoliert

$$f(x,y_2) = \frac{x_2 - x}{x_2 - x_1} \cdot v_{12} + \frac{x - x_1}{x_2 - x_1} \cdot v_{21}$$

$$\Rightarrow f(x,y) = \frac{y_2 - y}{y_2 - y_1} \cdot f(x,y_1) + \frac{y - y_1}{y_2 - y_1} \cdot f(x,y_2)$$

$x_1 = 2 \quad y_1 = 3 \qquad x = 3 \quad y = 4$

$x_2 = 4 \quad y_2 = 5$

$v_{11} = 10 \quad v_{12} = 20 \quad v_{21} = 30 \quad v_{22} = 40$

$$\rightarrow f(x,3) = \frac{4 - x}{2} \cdot 10 + \frac{x - 2}{2} \cdot 30 \overset{x=3}{=} 5 + 15 = 20$$

$$f(x,5) = \frac{4 - x}{2} \cdot 20 + \frac{x - 2}{2} \cdot 40 \overset{x=3}{=} 10 + 20 = 30$$

$$\Rightarrow f(3,4) = \frac{5 - y}{2} \cdot 20 + \frac{y - 3}{2} \cdot 30 \overset{y=4}{=} 10 + 15 = 25$$

Singulärwertzerlegung und Hauptkomponentenanalyse

Lineare Algebra

▶ Eigenvektoren und -werte

$$Av = \lambda \cdot v \iff (A - \lambda \cdot Id)v = 0$$

Eigenvektor $v \in \mathbb{C}^m, v \neq 0$

Eigenwert $\lambda \in \mathbb{C}$ → NS des char. Polynoms $p_A(\lambda) = \det(A - \lambda Id)$

↝ Polynom m-ten Grades → m Lösungen (mit Vielfachheit)

▶ Normen auf Vektorräumen

Sei V ein $\mathbb{R}$-Vektorraum, so ist die Abbildung $\|\cdot\| : V \to \mathbb{R}$ eine Norm genau dann, wenn die folgenden Axiome erfüllt sind:

(1) Positive Definitheit:

$\|v\| \geq 0$ für alle $v \in V$, $\|v\| = 0 \iff v = 0$

(2) Homogenität:

$\|\lambda \cdot v\| = |\lambda| \cdot \|v\|$ für alle $v \in V$, $\lambda \in \mathbb{R}$

(3) Dreiecksungleichung:

$\|v + w\| \leq \|v\| + \|w\|$ für alle $v, w \in V$

Ist $\|\cdot\|$ eine Norm, so ist $(V, \|\cdot\|)$ ein normierter Raum.

p-Normen $1 \leq p < \infty$ auf $\mathbb{R}^n$

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

→ $\|x\|_1 := \sum_i |x_i|$ Summennorm

→ $\|x\|_2 := \sqrt{\sum_i |x_i|^2}$ Euklid. Norm

→ $\|x\|_\infty := \sup_{1 \leq i \leq n} |x_i| = \max_{1 \leq i \leq n} |x_i|$ Maximumnorm

Äquivalente Normen

$$c_1 \|x\|_p \leq \|x\|_q \leq c_2 \|x\|_p \quad \text{für alle } x \in V$$

c_1, c_2 Konstant

→ im endlich dim. alle Normen äquivalent

Matrixnorm (Operatornorm)

V, W normierte Vektorräume

F lineare Abbildung, $F \in Abb(V, W)$

$$\Rightarrow \|F\| = \sup_{v \in V \setminus \{0\}} \frac{\|Fv\|_W}{\|v\|_V} = \sup_{\|v\|_V = 1} \|Fv\|_W$$

- jede Vektornorm definiert eine Matrix-Norm
- assoz. Matrixnorm ist Norm (definit, homogen, sub-additiv),

aber mehr als das:

→ $|||Id||| = 1$

→ sub-multiplikativ: $|||AB||| \leq |||A||| \cdot |||B|||$

→ mit Vektorraum kompatibel $\|Ax\| \leq |||A||| \cdot \|x\|$ für alle x

→ $|||A||| \geq |\lambda|$ für jeden Eigenwert λ von A

Spaltensummennorm → induziert durch Summennorm

$$\|A\|_1 = \max_{\|x\|_1} \|Ax\|_1 = \max_{j=1,\dots,m} \sum_{i=1}^{m} |a_{ij}|$$

Spektralnorm → induziert durch euklid. Norm

$$\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2 = \max_{\|x\|_2=1} \|Ax\|_2 = \sqrt{\lambda_{max}(A^T A)} = \sigma_1$$

→ für invertierbare A: $\|A\|_2 = \rho(A)$

Zeilensummennorm → induziert durch Supremumsnorm

$$\|A\|_\infty = \max_{i=1,\dots,m} \sum_{j=1}^{m} |a_{ij}|$$

Frobeniusnorm

$$\|A\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} |a_{ij}|^2} = \sqrt{\text{spur}(A^T A)} = \left(\sum \sigma_i^2 \right)^{\frac{1}{2}}$$

Spektralradius

$$\rho(A) := \max_{i=1,\dots,m} |\lambda_i(A)| \quad , \quad \lambda_i(A) \text{ i-ter Eigenwert von } A$$

▷ Konditionszahl einer Matrix

Sei A eine nicht singuläre Matrix, so ist die Kondition von A bzgl. einer Norm definiert als

$$\chi(A) = \frac{\max_{\|x\|=1} \|Ax\|}{\min_{\|x\|=1} \|Ax\|} \qquad \chi(A) = \|A\| \cdot \|A^{-1}\|$$

mit A^{-1} Pseudoinverse

Ist A außerdem quadratisch:

$$\chi(A) = \|A\| \cdot \|A^{-1}\|$$

$$\Rightarrow \quad \chi(A) \geq \left| \frac{\lambda_{max}}{\lambda_{min}} \right| \geq 1$$

A nicht singulär oder nicht quadratisch: $\chi(A) = \infty$

▷ Eigenwerte und Diagonalisierbarkeit

Basis aus Eigenvektoren $Av_i = \lambda_i v_i$ für $i = 1, 2, \dots, m$

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_m \end{bmatrix} \quad \Rightarrow \quad A = V \cdot D \cdot V^{-1}$$

▷ Spektralsatz für symmetrische Matrizen

Sei $A \in \mathbb{R}^{n \times n}$ eine reelle symmetrische Matrix, so gibt es eine Orthonormalbasis aus Eigenvektoren. Gleichwertig dazu ist die

Existenz einer Faktorisierung $A = VDV^!$ mit $D = \text{diag}(\lambda_1, \dots, \lambda_n)$, den Spalten der orthogonalen Matrix V als normierte Eigenvektoren von A.

▷ Diagonaldominanz

Sei $A \in \mathbb{R}^{n \times n}$, so heiße A ...

... (zeilenweise) diagonaldominant genau dann, wenn für alle $i \in \{1, \dots, n\}$

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

gilt.

... (zeilenweise) schwach diagonaldominant genau dann, wenn für alle $i \in \{1, \dots, n\}$

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

gilt.

< unzerlegbar

Singulärwertezerlegung

Sei $A \in \mathbb{R}^{m \times n}$ eine bel. Matrix, so gibt es eine Faktorisierung $A = U \Sigma V^T$.

Dabei gilt:

- $U \in \mathbb{R}^{m \times m}$ und $V \in \mathbb{R}^{n \times n}$ sind orthogonal
- Σ ist eine Diagonalmatrix mit $\Sigma_{ij} = 0$ für $i \neq j$ und $\Sigma_{11} \geq \Sigma_{22} \geq \dots \geq 0$ (Singulärwerte absteigend sortiert)
- Die Spalten von U bzw. V sind die Eigenvektoren von AA^T bzw. A^TA.
- $\sigma_i = \Sigma_{ii} \neq 0$ sind die Singulärwerte von A, genauer $\sigma_i = \sqrt{\lambda_i}$, wobei λ_i die Eigenwerte von AA^T oder A^TA sind.

Mit der Singulärwertezerlegung lässt sich einfach bestimmen:

- Der Rang entspricht genau der Anzahl der zu 0 verschiedenen Singulärwerte.
  $\text{rang}(A) = r$ aus $\sigma_1 \geq \dots \geq \sigma_r > 0$
- Der Kern entspricht dem durch die Zeilen V_i mit $i > r$ aufgespannten Raum. ~> „überzählige Zeilen"
  $\ker(A) = \text{span}\{V_{r+1}^T, \dots, V_n^T\}$
- Das Bild entspricht dem aufgespannten Raum aus den Spalten U_i mit $i \leq r$.
  $\text{im}(A) = \text{span}\{U_1, \dots, U_r\}$
- Die Euklidische Matrixnorm entspricht dem größten Singulärwert.
  $\|A\|_2 = \sigma_1 = \sigma_{max}$
- Die Konditionszahl entspricht dem Quotienten aus größtem und

kleinstem Singulärwert:

$$\kappa_2 = \frac{\sigma_1}{\sigma_r} = \frac{\sigma_{max}}{\sigma_{min}}$$

Pseudo-Inverse

→ Annäherung an tatsächliche Inverse, falls nicht bestimmbar:

$$A^{\wedge} = V \Sigma^{\wedge} U^T \quad \text{wobei} \quad \Sigma^{\wedge} = 1/\Sigma_{ii}$$

→ $A^{\wedge}$ und $\Sigma^{\wedge}$ sind n×m-Matrizen wie Σ^T und A^T

$$x = A^{\wedge} b = \sum_{i=1}^{r} \frac{1}{\sigma_i} (u_i \circ b) v_i$$

$$A = \left(u_1 \mid u_2 \mid u_3\right) \begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix} \begin{pmatrix} v_1 \\ \hline v_2 \\ \hline v_3 \end{pmatrix}$$

Beispiel:

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix}$$

1. Berechnung der Singulärwerte = √Eigenwerte von AA^T

$$AA^T = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix} \cdot \begin{pmatrix} 3 & 2 \\ 2 & 3 \\ 2 & -2 \end{pmatrix} = \begin{pmatrix} 17 & 8 \\ 8 & 17 \end{pmatrix}$$

$$p(\lambda) = \det(AA^T - \lambda \cdot E_2)$$
$$= \lambda^2 - 34\lambda + 225$$
$$= (\lambda - 25) \cdot (\lambda - 9) \quad \Rightarrow \quad \lambda_1 = 25, \ \lambda_2 = 9$$

$$\Rightarrow \quad \sigma_1 = \sqrt{25} = 5 \qquad \sigma_2 = \sqrt{9} = 3$$

2. Berechnung der Rechtssingulärvektoren = normierte Eigenvektoren der Eigenwerte (3. a) → als Zeilen

Eigenvektor zu $\lambda_1 = 25$

$$A^TA - 25 \cdot E_3 = \begin{pmatrix} -12 & 12 & 2 \\ 12 & -12 & -2 \\ 2 & -2 & -17 \end{pmatrix} \xrightarrow{\text{Gauss}} \begin{pmatrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\rightarrow v_3^{\wedge} = 0, \quad v_1^{\wedge} = v_2^{\wedge}$$

$$\Rightarrow v^1 = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0\right)^T$$

Eigenvektor zu $\lambda_2 = 9$

$$A^TA - 9 \cdot E_3 = \begin{pmatrix} 4 & 12 & 2 \\ 12 & 4 & -2 \\ 2 & -2 & -1 \end{pmatrix} \xrightarrow{\text{Gauss}} \begin{pmatrix} 1 & 0 & -\frac{1}{5} \\ 0 & 1 & \frac{1}{5} \\ 0 & 0 & 0 \end{pmatrix}$$

$$\rightarrow v^2 = \left(1/\sqrt{18}, -1/\sqrt{18}, 4/\sqrt{18}\right)^T$$

Eigenvektor zu $\lambda_3 = 0$

$$A^TA = \begin{pmatrix} 13 & 12 & 2 \\ 12 & 13 & -2 \\ 2 & -2 & 8 \end{pmatrix} \xrightarrow{\text{Gauss}} \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & -2 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\rightarrow v^3 = \left(\tfrac{2}{3}, -\tfrac{2}{3}, -\tfrac{1}{3}\right)^T$$

3. Berechnung der Linkssingulärvektoren

$$A = U \Sigma V^T = U \cdot \begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}$$

$\sigma_i \cdot u_i = A \cdot v_i$

Berechnung von u_1

$$u_1 = \frac{1}{\sigma_1} \cdot A \cdot v_1 = \frac{1}{5} \cdot \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix} \cdot \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{pmatrix} = \frac{1}{5} \begin{pmatrix} 5/\sqrt{2} \\ 5/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$

Berechnung von u_2

$$u_2 = \frac{1}{\sigma_2} \cdot A \cdot v_2 = \frac{1}{3} \cdot \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix} \cdot \begin{pmatrix} 1/\sqrt{18} \\ -1/\sqrt{18} \\ 4/\sqrt{18} \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$

$$\Rightarrow A = \underbrace{\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}}_{:= U} \cdot \underbrace{\begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{pmatrix}}_{=: \Sigma} \cdot \underbrace{\begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{pmatrix}}_{=: V^T}$$

Low-rank-approximation

Mit SVD Matrix durch Matrix mit niedrigerem Rang approximieren

Ges.: A_h vom Rang h

$$A_h = \min_{X : \operatorname{rang}(X) = h} \|A - X\|_F$$

$$\Rightarrow A_h = U \operatorname{diag}(\sigma_1, \dots, \sigma_h, \underbrace{0, \dots, 0}_{\text{setze } \sigma_{h+1} \text{ bis } \sigma_r \text{ Null}}) V^T$$

Warum?

- → Speichereffizient: Nur $h \cdot (m+n)$ Werte statt $m \cdot n$
- → Schnelle Berechnung von Matrix-Vektor-Multiplikationen:

$$A_h x = \sum_{i=1}^{h} \sigma_i (v_i \circ x) u_i$$

→ $h \cdot (2m+n)$ statt $2n \cdot m$ arithmet. Operationen

Approximationsfehler

beste Approximation bzgl. der Frobenius-Norm $\|\cdot\|_F$

$$\min_{X : \operatorname{rang}(X) = h} \|A - X\|_F^2 = \|A - A_h\|_F^2 = \sum_{j > h} \sigma_j^2$$

$$\min_{X : \operatorname{rang}(X) = h} \|A - X\|_2 = \|A - A_h\|_2 = \sigma_{h+1}$$

PCA (Hauptachsentransformation)

unter Beibehaltung der wesentlichen Merkmale
hochdim. Probleme → niedrigdim.

Geg.: Daten $\{X_i\}_{1 \le i \le N} \in \mathbb{R}^n$

Schritte zur Bestimmung der PCA

- Balanciere die Daten um den Mittelwert

$$\bar{X} = \frac{1}{N} \sum_i X_i \quad , \quad \bar{X}_i = X_i - \bar{X}$$

- Bilde Kovarianz-Matrix (sym., pos. def.)

$$C = \mathrm{cov}(X,X) = \frac{1}{N-1} \sum_i \bar{X}_i \cdot \bar{X}_i^T$$

- Diagonalisierung der Kovarianzmatrix

$$C = Q \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & & \ddots & \\ 0 & & \dots & \lambda_n \end{pmatrix} Q^T$$

→ Q orthogonal, $\lambda_1 \ge \lambda_2 \ge \dots \ge \lambda_n \ge 0$

→ Spalten q_j von Q sind die Eigenvektoren
= „Hauptachsen"

Numerische Bestimmung der SVD

→ Exakte Bestimmung der SVD nicht möglich

1. Ähnlichkeitstransformation

$B_1 = S^{-1} B S$ und B haben dieselben EVs, S transformiert EVs

→ S als Produkt von $n-2$ Householder-Spiegelungen

$$Q_i = Id - 2 n_i \cdot n_i^T = Q_i^T$$

2. EVs und EVs einer tridiag., pos. sem. def. Matrix D

→ QR-Zerlegung: $B_0 = Q_0 R_0 \rightarrow B_1 = R_0 Q_0$

$\rightarrow Q_0 B_1 Q_0^T = B_0$

$B_i = Q_i R_i \rightarrow B_{i+1} := R_i Q_i$

→ Aufwand nur $O(n)$

Iterative Verfahren

Viele numerische Probleme lassen sich nicht mit endlich vielen Schritten lösen. (→ Nullstellen, Eigenwerte, Min-Max-Suche)

→ Iterativer Lösungsansatz:
- > geschätzter Startwert x_0
- > Startwert sukzessive verbessern $x_{i+1} = \Phi_i(x_i)$
  oder $x_{i+1} = \Phi_i(x_i, x_{i-1}, \dots)$

Φ_i nicht von i abhängig = stationäres Verfahren

Fixpunktiteration

Sei $M \neq \emptyset$ und $\Phi \in Abb(M,M)$ eine Abbildung. Ein $x \in M$, das den Zusammenhang $\Phi(x) = x$ erfüllt, heißt Fixpunkt von Φ.

Wenn die Iterationsfolge $x_{n+1} = \Phi(x_n)$ konvergiert mit $\lim_n x_n := x^*$ und Φ stetig ist, dann ist der Grenzwert x^* immer ein Fixpunkt von Φ

$$\Phi(x^*) = x^*$$

→ Wann konvergieren Fixpunktiterationen?

Vollständigkeit, Banachraum
Ein normierter $\mathbb{K}$-VR $(V, \|\cdot\|)$ heiße genau dann vollständig, wenn jede Cauchyfolge in V gegen ein Element in V konvergiert.
Ein Vektorraum heiße genau dann Banachraum, wenn er normiert und vollständig ist.
Ein Banachraum, dessen Norm durch ein Skalarprodukt erzeugt wird, heiße Hilbertraum.

Kontraktion
Sei $(V, \|\cdot\|)$ ein $\mathbb{R}$-Vektorraum, $M \subseteq V$ und $\Phi \in Abb(M,V)$. Wir sagen Φ heiße Kontraktion genau dann, wenn eine Konstante $0 \leq k < 1$ existiert, so dass für alle $x, y \in M$ gelte, dass

$$\|\Phi(x) - \Phi(y)\| \leq k \|x - y\| \quad (*)$$

Eine Konstante k, die diesen Zusammenhang erfüllt, heißt auch Kontraktionskonstante von Φ.

→ bedeutet anschaulich, dass die Bilder von Φ näher beieinander liegen als die Urbilder von Φ.

Stetigkeit
Jede Kontraktion ist stetig.

(*) ist schwer zu überprüfen. Vereinfachung, die aber Auswahlmenge an Funktionen einschränkt:

Kontraktionskriterium

Im Fall $V = \mathbb{R}$ ist für die Kontraktionseigenschaft hinreichend, dass f differenzierbar ist mit $k_* := \sup_{x \in M} |f'| < 1$.

k_* ist dann die Kontraktionskonstante von Φ.

Fixpunktsatz von Banach

Sei $(V, \|\cdot\|)$ ein Banachraum und sei $\emptyset = M \subseteq V$ eine abgeschlossene Teilmenge von V, sei $\Phi \in \mathrm{Abb}(M, V)$ selbstabbildend, sprich $\Phi(M) \subseteq M$, sowie eine Kontraktion. Dann hat Φ genau einen Fixpunkt $x^* \in M$ und x^* ist Grenzwert der Folge $*$, wobei $x_0 \in M$ beliebig ist.

Sei k zudem noch eine Kontraktionskonstante von Φ, so fällt der Approximationsfehler in jedem Iterationsschritt um mindestens den Faktor k, also $\|x_{n+1} - x_*\| \leq k \|x_n - x_*\|$ damit auch $\|x_n - x_*\| \leq k^n \|x_0 - x_*\|$

Falls $M = V$ ist die Eigenschaft der Abgeschlossenheit und Selbstabbildungseigenschaft erfüllt. (trivial)

Newtonverfahren

→ Iteratives Verfahren zur Bestimmung der Nullstelle von $f(x)$:

Startwert x_0

Nullstelle der Linearisierung $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$

→ Fixpunkt-Iteration für $\Phi(x) = x - \frac{f(x)}{f'(x)}$

konvergiert, wenn x_0 „nahe" bei der Nullstelle liegt.

Fehlerreduktion: $\|x_{n+1} - x^*\| \approx c \cdot \|x_n - x^*\|^2$

$(f \in C^2, f'(x^*) \neq 0)$

asymptot. Kontraktionskonst. $= 0$

Lineare und quadrat. Konvergenz

Sei $(x_n) \subset (V, \|\cdot\|)$ eine Folge mit Grenzwert $x^* \in V$ und V sei normiert. Das Verfahren heiße ...

... linear konvergent genau dann, wenn für ein $k < 1$ der Zusammenhang $|x_n - x^*| \leq k \|x_{n-1} - x^*|$ erfüllt ist.

... quadratisch konvergent genau dann, wenn für ein $c > 0$ der Zusammenhang $\|x_n - x^*\| \leq c \cdot \|x_{n-1} - x^*\|^2$ erfüllt ist.

Fixpunktiteration min. quadratisch konvergent, genau dann wenn die asymtot. Kontraktionskonstante $k^* = 0$ ist.

→ Fixpunktiterationen im Allgemeinen linear konvergent

im $\mathbb{R}^n$: $x_{n+1} := x_n - [Jf(x_n)]^{-1} \cdot f(x_n)$

↑ Jacobi-Matrix

lokal quadratisch konvergent

Lokale quadrat. Konvergenz

Es existiere eine Umgebung $U_\varepsilon(x_*)$ um x_* derart, dass wenn der Startwert $x_0 \in U_\varepsilon(x_*)$ aus ebendieser Umgebung gewählt sei, das Verfahren dann quadratisch konvergent heiße.

„Effizienteres" Newtonverfahren im $\mathbb{R}^n$

① Löse das lineare Gleichungssystem $(Jf)(x_n) \cdot \Delta x = -f(x_n)$

② $x_{n+1} := x_n + \Delta x$

Sekanten-Verfahren

zweistufiges iteratives Verfahren ohne Kenntnis der Ableitung

2 Startwerte: x_0, x_1

→ Nullstelle der Sekante: $x_{i+1} = \frac{x_{i-1} \cdot f(x_i) - x_i \, f(x_{i-1})}{f(x_i) - f(x_{i-1})}$

Konvergenzordnung: $p = \frac{\sqrt{5}+1}{2}$

Bisektionsverfahren

zweistufiges Verfahren

Führt sicher zum Ziel, aber langsam → nach i Schritten $|x_{i+1} - x_i| \leq 2^{-i} |x_1 - x_0|$

2 Startwerte $x_0 < x_1$, sodass $f(x_0) \cdot f(x_1) < 0$

→ VZ-Wechsel im Intervall $[x_0, x_1]$ → min. 1 Nullstelle ($f(x)$ stetig)

Iterationsschritt:

Mittelpunkt bestimmen $x_2 = \frac{x_0 + x_1}{2}$

Falls $f(x_0) \cdot f(x_2) < 0$ → $[x_0, x_2]$

Falls $f(x_1) \cdot f(x_2) < 0$ → $[x_2, x_1]$

Falls $f(x_1) \cdot f(x_2) = 0$ → x_2 ist eine Nullstelle

Konvergenz in etwa linear

Regula falsi

→ Mischung aus Sekanten- und Bisektionsverfahren

Startpunkte x_0, x_1, sodass $f(x_0) \cdot f(x_1) < 0$ 🟢

Schnittpunkt der Sekante mit der x-Achse 🟠

neues Intervall: $f(x_2) f(x_1) < 0$ → $[x_1, x_2]$ sonst → x_0, x_2

konvergiert garantiert, aber langsam

Konvergenz in etwa linear

Vergleich der Verfahren

Newton	+	konvergiert sehr schnell ($p=2$)
	−	benötigt Werte der Ableitungen
	−	konvergiert nur für Startwerte nahe bei der Nullstelle
Sekanten	+	konvergiert ziemlich schnell ($p = 1{,}62$)

+ benötigt keine Werte der Ableitungen
- konvergiert nur für Startwerte nahe bei der Nullstelle

Bisektion und regula falsi
- konvergiert langsam ($p=1$)
+ benötigt keine Werte der Ableitungen
+ sicher konvergent

Abbruchkriterien

1. Fehlerabschätzung nutzen, falls vorhanden
2. max. Iterationszahl erreicht
3. $\|x_{i+n} - x_i\| < \varepsilon$
4. $\|F(x_i)\| < \varepsilon$ (Nullstellensuche) bzw. $\|\Phi(x_i) - x_i\| < \varepsilon$ (Fixpunktiteration)
5. $\|x_i\| > M$ für eine große Schranke M → nicht konvergent

→ Kombination mehrerer Kriterien

Gauss-Seidel-Verfahren

```
x = np.zeros((n,1))
while np.linalg.norm(b-A@x) > eps:
    for k in range(n):
        left = sum(a[k,0:k] * x[0:k,0])
        right = sum(a[k,k+1] * x[k+1:,0])
        x[k] = (b[k]-left-right)/a[k,k]
return x
```

LGS $Ax = b$ mit $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ mit $n \in \mathbb{N}$
alle Diagonaleinträge $a_{ii} \neq 0$

→ Iterationsvorschrift:

$$x_{m+1,i} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_{m+1,j} - \sum_{j=i+1}^{n} a_{ij} x_{m,j} \right)$$

$A = L \cdot D \cdot R$

$\Rightarrow V_{GS} = -(L+D)^{-1} R$

$(L+D)x^{i+1} + Rx^i = b$
$x^{i+1} = (L+D)^{-1}(b - Rx^i)$
→ $(L+D)$ wird nicht wirklich invertiert → Vorwärtssubst.

→ konvergiert im Allg. (bei tridiagonalen Matrizen) um den Faktor 2 schneller als das Jacobiverfahren.

Jacobi-Verfahren

```
x = np.zeros((n,1))
while np.linalg.norm(b-A@x) > eps:
    x_new = np.zeros((n,1))
    for k in range(n):
        left = sum(a[k,0:k] * x[0:k,0])
        right = sum(a[k,k+1:] * x[k+1:,0])
        x_new[k] = (b[k]-left-right)/a[k,k]
return x_new
```

lineares Gleichungssystem $Ax = b$ mit $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ mit $n \in \mathbb{N}$.
alle Diagonaleinträge $a_{ii} \neq 0$

$\Phi(x) := -D^{-1}Rx + D^{-1}b$

$x_{m+1} := \Phi(x_m)$

→ Iterationsschritt: $(x_{m+1})_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j \in \{1,\dots,n\} \setminus \{i\}} a_{ij} (x_m)_j \right)$

$V_J = -D^{-1}(L+R)$ $\quad b = Dx^{i+1} + (L+R)x^i$

→ neue, bessere Werte werden ignoriert, das Verfahren ist dafür aber besser parallelisierbar

Hinreichendes Konvergenzkriterium

Sei $\Phi(x) = x$, dann ist Φ eine Kontraktion bezüglich der $\|\cdot\|_\infty$-Norm auf ganz $\mathbb{R}^n$, falls die Systemmatrix A diagonaldominant ist.

→ Jacobi-Verfahren konvergiert

Sei A unzerlegbar und schwach diagonaldominant, so konvergieren ebenfalls alle Verfahren. Eine Matrix A heißt unzerlegbar, wenn ihr Graph stark zusammenhängend ist.

SOR-Verfahren

Relaxation

→ Konvergenzbeschleunigung durch Relaxation

Ersetze x^{i+1} durch $wx^{i+1} + (1-w)x^i$

$w > 1$: Überrelaxation

$0 < w < 1$: Unterrelaxation

→ SOR $1 < w < 2$ (Successive Over Relaxation)

→ Gauss-Seidel $w = 1$

→ Schnellere Konvergenz durch Relaxation (Über) Parameter w

```
x = np.zeros((n,1))
while np.linalg.norm(b-A@x) > eps:
    for k in range(n):
        left = sum(a[k,0:k] * x[0:k,0])
        right = sum(a[k,k+1:] * x[k+1:,0])
        x_tmp = (b[k] - left - right)/a[k,k]
        x[k] = (1-w) * x[k] + x_tmp * w
return x
```

→ Überrelaxation: Zielwert bewusst überschießen (abschwächen)

$w \in (0,2)$, sonst Divergenz

$x_n = H_w x_{n-1} + w(D+wL)^{-1}b$

$H_w = (D+wL)^{-1}[(1-w)D - wR] = E_n - w(D+wL)^{-1}A$

$$x_i^{m+1} = (1-w) \cdot x_i^m + w \left(\frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{m+1} - \sum_{j=i+1}^{n} a_{ij} x_j^m \right) \right)$$

Bedingungen für A wie bei den anderen Verfahren

$$V_{SOR} = -\left(\frac{1}{\omega}D + L\right)^{-1}\left(\frac{\omega-1}{\omega}D + R\right)$$

Jacobi $O(n)$
Gauß-Seidel $O(n)$
SOR $O(\sqrt{n})$

A unzerlegbar & schwach-diagonaldominant → SOR, Jacobi, Gauss-Seidel konvergieren
A strikt diagonaldominant → Gauss-Seidel, Jacobi & SOR konvergieren

A pos. definit → SOR $(0 < \omega < 2)$ und Gauss-Seidel konvergieren

A und $D - 2A$ positiv definit → SOR, Gauss-Seidel & Jacobi konvergieren

---

$$\rho_{GS/J} \approx 1 - \frac{\alpha}{n}$$
$$\rho_{SOR} \approx 1 - \frac{\alpha}{\sqrt{n}}$$

Vektoriteration

$Ax = b \Rightarrow A = \hat{A} + E \Rightarrow x^{i+1} = \hat{A}^{-1}(b - Ex^i)$

allg. Form: $\Phi(x) = Vx + d$

Entscheidung für lin. Konvergenz $\rho(V) < 1$

Iterative Verfahren vs. Direkte Verfahren

↓

Iterations- & Rundungsfehler werden abgeschwächt

Schnell eine (grobe) Näherung, kann ggf. abbrechen

↓

Rundungsfehler pflanzen sich fort, u.U. mit Verstärkung

Lösung erst am Ende

Beispiel: SOR-Verfahren

$$A = \begin{pmatrix} 4 & -2 & 1 \\ -1 & 5 & -2 \\ 1 & 1 & 3 \end{pmatrix} \quad b = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \quad w = \frac{1}{2} \quad x^{(0)} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

$$s_1 = (4 \quad -2 \quad 1) \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = 4 - 4 + 1 = 1$$

$$x_1^{(1)} = x_1^{(0)} - \frac{w}{A(1,1)} \cdot (s_1 - b_1) = 1 - \frac{1/2}{4} \cdot (1 - 2)$$

$$= 1{,}125$$

$$\tilde{x}^{(1)} = \begin{pmatrix} 1{,}125 \\ 2 \\ 1 \end{pmatrix} \text{ aus } x^0$$

$$s_2 = (-1 \quad 5 \quad -2) \begin{pmatrix} 1{,}125 \\ 2 \\ 1 \end{pmatrix} = 6{,}875$$

$$x_2^{(1)} = x_2^{(0)} - \frac{w}{A(2,2)} \cdot (s_2 - b_2) = 2 - \frac{1/2}{5} \cdot (6{,}875 - 4) = 1{,}7125$$

$$\tilde{x}^{(1)} = \begin{pmatrix} 1{,}125 \\ 1{,}7125 \\ 1 \end{pmatrix}$$

$$s_3 = (1 \quad 1 \quad 3) \begin{pmatrix} 1{,}125 \\ 1{,}7125 \\ 1 \end{pmatrix} = 5{,}8375$$

$$x_3^{(1)} = x_3^{(0)} - \frac{w}{A(3,3)} \cdot (s_3 - b_3) = 1 - \frac{1/2}{3} \cdot (5{,}8375 - 6) \; 1{,}027$$

$$\Rightarrow x^{(1)} = \begin{pmatrix} 1{,}125 \\ 1{,}7125 \\ 1{,}027 \end{pmatrix}$$

Beispiel: Gauss-Seidel

Lösungsvorschrift Matrix-Vektor Notation

$$x^{(k+1)} = \left(I - (D-L)^{-1}\right) \cdot x^{(k)} + (D-L)^{-1} \cdot b$$

$$= (D-L)^{-1} \cdot U \cdot x^{(k)} + (D-L)^{-1} \cdot b$$

Erste Iteration ($k=0$):

$$\rightarrow x^{(1)} = (D-L)^{-1} \cdot U \cdot x^{(0)} + (D-L)^{-1} \cdot b$$

$$A = \begin{pmatrix} 4 & -2 & 1 \\ -1 & 5 & -2 \\ 1 & 1 & 3 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \qquad x^{(0)} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

$$(D-L)^{-1} = \begin{pmatrix} 4 & 0 & 0 \\ -1 & 5 & 0 \\ 1 & 1 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 1/4 & 0 & 0 \\ 1/20 & 1/5 & 0 \\ -1/10 & -2/3 & 1/3 \end{pmatrix}$$

$$(D-L)^{-1} \cdot U = \begin{pmatrix} 1/4 & 0 & 0 \\ 1/20 & 1/5 & 0 \\ -1/10 & -2/3 & 1/3 \end{pmatrix} \cdot \begin{pmatrix} 0 & 2 & -1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 2 & -1/4 \\ 0 & 1/10 & 7/20 \\ 0 & -1/5 & -1/30 \end{pmatrix}$$

$$(D-L)^{-1} \cdot U \cdot x^{(0)} = \begin{pmatrix} 0 & 2 & -1/4 \\ 0 & 1/10 & 7/20 \\ 0 & -1/5 & -1/30 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 3/4 \\ 11/20 \\ -0,433 \end{pmatrix}$$

$$(D-L)^{-1} \cdot b = \begin{pmatrix} 1/4 & 0 & 0 \\ 1/20 & 1/5 & 0 \\ -1/10 & -2/3 & 1/3 \end{pmatrix} \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} = \begin{pmatrix} 1/2 \\ 9/10 \\ 1,533 \end{pmatrix}$$

$$x^{(1)} = (D-L)^{-1} \cdot U \cdot x^{(0)} + (D-L)^{-1} \cdot b = \begin{pmatrix} 3/4 \\ 11/20 \\ -0,433 \end{pmatrix} + \begin{pmatrix} 1/2 \\ 9/10 \\ 1,533 \end{pmatrix} = \begin{pmatrix} 1,25 \\ 1,45 \\ 1,1 \end{pmatrix}$$

Komponentenweise Darstellung (wie SOR, aber $\omega = 1$)

$$s_1 = (4 \quad -2 \quad 1) \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = 4 - 4 + 1 = 1$$

$$x_1^{(1)} = x_1^{(0)} - \frac{1}{A(1,1)} \cdot (s_1 - b_1) = 1 - \frac{1}{4} \cdot (1 - 2) = 1,25$$

$$\tilde{x}^{(1)} = \begin{pmatrix} 1{,}25 \\ 2 \\ 1 \end{pmatrix}$$

$$s_2 = (-1 \quad 5 \quad -2) \begin{pmatrix} 1{,}25 \\ 2 \\ 1 \end{pmatrix} = 6{,}75$$

$$x_2^{(1)} = x_2^{(0)} - \frac{1}{A(2,2)} \cdot (s_2 - b_2) = 2 - \frac{1}{5} \cdot (6{,}75 - 4) = 1{,}45$$

$$\tilde{x}^{(1)} = \begin{pmatrix} 1{,}25 \\ 1{,}45 \\ 1 \end{pmatrix}$$

$$s_3 = (-1 \quad 1 \quad 3) \begin{pmatrix} 1{,}25 \\ 1{,}45 \\ 1 \end{pmatrix} = 5{,}7$$

$$x_3^{(1)} = x_3^{(0)} - \frac{1}{A(3,3)} (s_3 - b_3) = 1 - \frac{1}{3} \cdot (5{,}7 - 6) = 1{,}1$$

$$\Rightarrow x^{(1)} = \begin{pmatrix} 1{,}25 \\ 1{,}45 \\ 1{,}1 \end{pmatrix}$$

Beispiel: Jacobi-Iteration

$$A = \begin{pmatrix} 4 & -2 & 1 \\ -1 & 5 & -2 \\ 1 & 1 & 3 \end{pmatrix} \qquad b = \begin{pmatrix} 2 \\ 4 \\ 6 \end{pmatrix} \qquad x^{(0)} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

Erste Iteration: $x^{(1)} = D^{-1} \cdot (L+U) \cdot x^{(0)} + D^{-1} b$

Komponentenweise Darstellung:

Wie Gauss-Seidel, aber am Anfang alle s_i bestimmen

1. Hilfsgrößen bestimmen:

$$s_1 = (4 \quad -2 \quad 1) \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = 1$$

$$s_2 = (-1 \quad 5 \quad -2) \cdot \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = 7$$

$$s_3 = (1 \quad 1 \quad 3) \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix} = 6$$

2. Lösungskomponenten bestimmen:

$$x_1^{(1)} = x_1^{(0)} - \frac{1}{A_{11}} \cdot (s_1 - b_1) = 1 - \frac{1}{4} \cdot (1 - 2) = 1{,}25$$

$$x_2^{(1)} = x_2^{(0)} - \frac{1}{A_{22}} \cdot (s_2 - b_2) = 2 - \frac{1}{5} \cdot (7 - 4) = 1{,}4$$

$$x_3^{(1)} = x_3^{(0)} - \frac{1}{A_{33}} \cdot (s_3 - b_3) = 1 - \frac{1}{3} \cdot (6 - 6) = 1$$

$$\Rightarrow x^{(1)} = \begin{pmatrix} 1{,}25 \\ 1{,}4 \\ 1 \end{pmatrix}$$

Zelluläre Automaten und die Lattice-Boltzmann-Methode

Zelluläre Automaten

-> Game of Life

2D-Gitter bel. Größe
Jede Zelle: 2 Zustände -> tot oder lebend
Regeln: 1. 3 lebende Nachbarn -> Zelle wird lebendig
2. < 2 lebende Nachbarn -> Zelle stirbt
3. > 3 lebende Nachbarn -> Zelle stirbt

-> Lattice-Gas Zelluläre Automaten (LGCA)

jede Zelle mehrere gerichtete Einträge
jeder Eintrag: 1 oder 0
2-stufige Update-Regel:
-> Stream: Jeder Eintrag wandert in entsprechende Nachbarzelle
-> Collide: innerhalb der Zelle tauschen Einträge
⊕ einfach zu implementieren, inhärent parallel, trivial Massen- und Energieerhaltend
⊖ „spurious invariants", auf modernen Computern sehr ineffizient, physikalisch nicht immer korrekt

-> Molekulardynamik

-> Navier-Stokes-Gleichung

-> Lattice-Boltzmann-Methode

Update wie bei LGCA
Randbehandlung (leicht)
└> jede Zelle speichert zusätzlich ihren Typ
alle Randpunkte vor Stream aktualisieren

Nichtlineare Optimierung

Lagrange Multiplikatoren

Das Minimum/Maximum x^* einer diff.baren Funktion $F: \mathbb{R}^n \to \mathbb{R}$ unter den Nebenbedingungen $g_1(x) = c_1,\ g_2(x) = c_2, \dots g_m(x) = c_m$ erfüllt folgende Bedingungen:

Es gibt Skalare $(\lambda_1, \lambda_2, \dots, \lambda_m)$ so dass

$$\text{grad}(F)(x^*) = \lambda_1 \text{grad}(g_1)(x^*) + \dots + \lambda_m \text{grad}(g_m)(x^*)$$

→ n Gleichungen + m Nebenbedingungen

→ n+m Gleichungen für n+m Unbekannte $(x^*_1, \dots, x^*_n, \lambda_1, \dots, \lambda_n)$

Optimierung mit Newton-Verfahren

$$x_{n+1} = x_n - [H_F(x_n)]^{-1} \text{grad}(F)(x_n)$$

mit $H_F(x_i) = J_{\text{grad}(F)} = \begin{bmatrix} \frac{\partial^2 F}{\partial x_1 \partial x_1} & \dots & \frac{\partial^2 F}{\partial x_n \partial x_1} \\ \frac{\partial^2 F}{\partial x_1 \partial x_n} & \dots & \frac{\partial^2 F}{\partial x_n \partial x_n} \end{bmatrix}$

→ aufwändig für große n

oft schwierig/zu teuer die 2. Ableitung zu bestimmen

Nullstellenbestimmung

$$x_{i+1} = x_i - \frac{g(x_i)}{g'(x_i)} \quad (1D)$$

$$x_{i+1} = x_i - J_G(x_i)^{-1} G(x_i)$$

Abstiegsverfahren

Sei ein Startwert $x_0 \in \mathbb{R}^n$ sowie eine Startsuchrichtung $s_0 \in \mathbb{R}^n$ gegeben. Dann heißt die Iterationsvorschrift

$$x_{n+1} = x_n + \alpha_n \cdot s_n$$

mit $\alpha_n > 0$ als Suchweite und s_0 als Suchrichtung Abstiegsverfahren, falls $s_n \circ \nabla f(x_n) < 0$ gilt.

Algorithmus:

Eingabe: zu minimierendes Funktional $F: \mathbb{R}^n \to \mathbb{R}$

Ausgabe: Stelle x^*, die ungefähr einem Minimum von F entspricht

```
Wähle x_0 ∈ R^n
k = 0
while k genügt Abbruchkriterium nicht do
    Bestimme Abstiegsrichtung d^k mit < d^k, ∇F(x_k)> < 0
    Bestimme Schrittweite T_k > 0 mit F(x_k + T_k · d^k) < F(x_k)
    x_{k+1} = x_k + T_k · d^k
end while
```

Optimale Schrittweite

Die optimale Schrittweite kann man durch das näherungsweise Lösen des 1D Minimierungsproblems

$$t_n = \underset{t>0}{\text{argmin}} \{f(x_n + t \cdot s_n)\}$$

→ Newtonverfahren: $t_n = 1$

→ besser dämpfen: $t < 1$

Abstiegsverfahren: Goldener Schnitt

Starte mit 3 Werten $a < b < c$ für die $f(a) > f(b) < f(c)$
Wähle einen neuen Punkt d: $a < d < c$
$f(d) < f(b)$ → d als mittleren Punkt des neuen Tripels
sonst → b als mittleren Punkt des neuen Tripels

→ Wahl von d:
Größeres Intervall gem. Goldenen Schnitt teilen, sodass das größere Teil außen liegt.

Auffinden der Suchrichtung

Newton-Verfahren: $s_i = -[H_F(x_i)]^{-1} \operatorname{grad}(F)(x_i)$, $t_i = 1$ (pos. definit)
Gradientenverfahren: $s_i = -\operatorname{grad}(F)(x_i)$, $t_i = ?$ (steepest descent)

→ Gradientenverfahren

$d^k = -\nabla F(x_k)$ → laufe in Richtung des neg. Gradienten

⊕ einfacher, schneller Algorithmus
(für gut konditionierte und strikt konvexe Funktionen)
⊖ viele Probleme nicht strikt konvex,
Fkt. müssen stetig diffbar sein

$x_{n+1} = x_n - \alpha_n \nabla f(x_n)$ mit z.B. $\alpha_n = \frac{\langle \nabla f(x_n), \nabla f(x_n) \rangle}{\langle A \nabla f(x_n), \nabla f(x_n) \rangle}$

→ Verfahren der konjugierten Gradienten (cg-Verfahren)

quadrat. Funktional: $F(x) = x^T A x + 2 b^T x + c$
(mit A pos. definit, quadrat.)
→ 2 Vektoren u und v sind A konjugiert, wenn $u^T A v = 0$
$\langle u, Av \rangle = 0$

Wählt man als Suchrichtungen (s_i) eine Folge an paarweise konjugierten Richtungen und dazu optimalen Schrittweiten t_i, so ist man nach (spätestens) n Schritten im Minimum.

Eingabe: $A \in \mathbb{R}^{n \times n}$ s.p.d., $b \in \mathbb{R}^n$
Ausgabe: Stelle x^*, die ungefähr $Ax + b = 0$ erfüllt

Wähle $x_0 \in \mathbb{R}^n$
$g^0 = b + Ax^0$
$d^0 = -g^0$
$\alpha^0 = \langle d^0, Ad^0 \rangle^{-1} \cdot \langle g^0, g^0 \rangle$
for $k = 0$; $\|g^k\| \geq \varepsilon$; $k = k+1$ do // Abbruchkriterium
$x_{k+1} = x_k + \alpha_k \cdot d_k$
$g_{k+1} = g^k + \alpha_k \cdot Ad^k$
$\beta_k = \langle g^k, g^k \rangle^{-1} \cdot \langle g^{k+1}, g^{k+1} \rangle$
$d^{k+1} = -g^{k+1} + \beta_k \cdot d^k$
$\alpha_{k+1} = \langle d^{k+1}, Ad^{k+1} \rangle^{-1} \cdot \langle g^{k+1}, g^{k+1} \rangle$
end for

Lösen lin. GS mit pos. definiter Koeffizientenmatrix

① Wähle x^1 beliebig und berechne

$$g^1 = b + Ax^1 \quad , \quad s^1 = -g^1 \quad , \quad \alpha^1 = \frac{g^1 \circ g^1}{s^1 \circ As^1}$$

② Für $k \in \{1, 2, \dots, n\}$:

(2.1) Falls $\|g^{k+1}\| < \varepsilon$ halte an

(2.2) Bestimme von x^k in Richtung s^k das Minimum $x^{k+1} = x^k + \alpha^k s^k$

(2.3) Aktualisiere den Gradienten: $g^{k+1} = g^k + \alpha^k A s^k$

(2.4) Aktualisiere die Suchrichtung, sodass sie A-konjugiert zu allen bisherigen ist: $s^{k+1} = -g^{k+1} + \beta^k \cdot s^k$ mit $\beta^k = \frac{g^{k+1} \circ g^{k+1}}{g^k \circ g^k}$

(2.5) Neue optimale Schrittweite: $\alpha^{k+1} = \frac{g^{k+1} \circ g^{k+1}}{s^{k+1} \circ A s^{k+1}}$

Lösen allg. LGS

① $g^1 = \nabla f(x^0) \quad , \quad s^1 = -g^1 \quad , \quad \alpha^1 = \underset{t}{\operatorname{argmin}} \{ f(x^0 + t s^0) \}$

② Für $k \in \{1, 2, \dots, n\}$

(2.1) Falls $\|g^{k+1}\| < \varepsilon$ halte an

(2.2) $x^{k+1} = x^k + \alpha^k s^k$

(2.3) $g^{k+1} = \nabla f(x^{k+1})$

(2.4) $s^{k+1} = -g^{k+1} + \beta^k \cdot s^k$

mit $\beta^k = \frac{g^{k+1} \circ g^{k+1}}{g^k \circ g^k}$ / $\beta^k = \frac{g^{k+1} \circ (g^{k+1} - g^k)}{g^k \circ g^k}$

(2.5) $\alpha^{k+1} = \underset{t}{\operatorname{argmin}} \{ f(x^{k+1} + t s^{k+1}) \}$

⊕ theoret. nach n Schritten exakte Lösung (aber Rundungsfehler)

Rundungsfehler meist durch spätere Iterationen korrigiert

mit jedem Schritt Näherung an die Lösung

Für voll besetzte Matrix: $O(n^3)$

dünn besetzte Matrix: $O(n^2)$

Quasi-Newton-Verfahren

→ ersetze $[H_F(x_i)]^{-1}$ durch eine (Folge) positiv definiter Matrizen H_k für die gilt $x^k - x^{k-1} = H_k [\text{grad}(F)(x^k) - \text{grad}(F)(x^{k-1})]$

→ Konvergenz bleibt superlinear

Quadratur – Volumen- und Flächenberechnung – Numerische Integration

Monte Carlo Verfahren

$$V = \frac{\#\text{ hits}}{\#\text{ samples}}$$

konvergiert sehr langsam

Newton-Cotes-Formeln

Geg.: $f(x)$

→ bestimme $p(x)$ (interpoliert Funktionswerte an äquidist. Stellen)

→ als Näherungswert: $\int_a^b p(x)\,dx$ (statt $\int_a^b f(x)\,dx$)

Zu $n+1$ Sample-Punkten der Funktion $f(x)$

$\{(x_0, f(x_0)), (x_1, f(x_1)), \dots (x_n, f(x_n))\}$

gibt es genau ein interpolierendes Polynom vom Grad n.

z.B.: $p(x) = \sum_{i=0}^{n} f(x_i) \cdot L_i(x)$

⇒ n = 0 Mittelpunktsregel

Sample-Punkt $(x_0, f(x_0))$: $x_0 = \frac{a+b}{2}$

$p_0(x) = f(x_0) = \text{const.}$

Näherungswert: $\int_a^b f(x)\,dx \approx \int_a^b p_0(x)\,dx = (b-a) f(x_0)$

↳ Approximation durch ein Rechteck

$x_0 = a$ linke Rechtecksregel

$x_0 = \frac{a+b}{2}$ Mittelpunktsregel

$x_0 = b$ rechte Rechtecksregel

Fehlerabschätzung: $\text{Error} \le \frac{(b-a)^3}{24} \cdot \max_x \{|f''(x)|\}$

⇒ n = 1 Trapezregel

2 Sample-Punkte $(x_0, f(x_0)), (x_1, f(x_1))$: $x_0 = a, x_1 = b$

$p_1(x) = \frac{b-x}{b-a} f(x_0) + \frac{x-a}{b-a} f(x_1)$ (lin. Interpolant)

Näherungswert: $\int_a^b f(x)\,dx \approx \int_a^b p_1(x)\,dx = (b-a)\frac{f(a)+f(b)}{2}$

Fehlerabschätzung: $\text{Error} \le \frac{(b-a)^3}{12} \cdot \max_x \{|f''(x)|\}$

⇒ n = 2 Simpsonregel oder Keplersche Faßregel

3 Sample-Punkte $(x_i, f(x_i))$, $i = 0,1,2$: $x_0 = a, x_1 = \frac{a+b}{2}, x_2 = b$

$$p_2(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} f(x_0) + \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} f(x_1) + \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} f(x_2)$$

Näherungswert: $\int_a^b f(x)\,dx \approx \int_a^b p_2(x)\,dx = (b-a)\left(\frac{1}{6} f(x_0) + \frac{4}{6} f(x_1) + \frac{1}{6} f(x_2)\right)$

Fehlerabschätzung: $\text{Error} \le \frac{(b-a)^5}{5760} \cdot \max_x \{|f^{(4)}(x)|\}$

⇒ n = bel.

→ Intervallenden geschlossen:

$x_i = a + i \frac{b-a}{n}$

speziell: $a=0, b=1 : x_i = \frac{i}{n}$

-> Intervallenden offen

$x_i = a + (2i+1) \cdot \frac{b-a}{2n+2}$

speziell: $a=0, b=1: x_i = \frac{2i+1}{2n+2}$

Newtons 3/8 - Regel

$$\int_a^b f(x)dx \approx \frac{b-a}{8} (f(a) + 3f(x_1) + 3f(x_2) + f(b))$$

-> Fehlerabschätzung: $\text{Error} \leq \frac{(b-a)^5}{6480} \cdot \max_x \{|f^{(4)}(x)|\}$

-> Verwendung in iterierter Form, da sehr ungenau

-> Iterierte Trapezregel $O(h^2)$

-> Iterierte Simpsonregel $O(h^4)$

Richardson Extrapolation / Romberg - Quadratur

↳ Verbesserung der Genauigkeit