Grundlagen

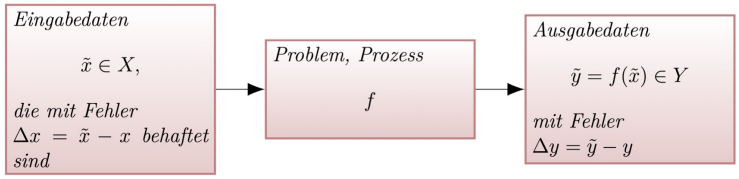
The numerical point of view goes back to the earliest mathematical writings. A tablet from the Yale Babylonian Collection (YBC 7289), gives a sexagesimal numerical approximation of the square root of 2, the length of the diagonal in a unit square.

Fehlerarten

- Modellfehler: Idealisierungen / Vereinfachungen
- Datenfehler: Messfehler
- Verfahrensfehler: numerische Verfahren berechnen Lösungen nur näherungsweise
- Rundungsfehler: reelle Zahlen können mit Floats nur näherungsweise dargestellt werden

Konditionierung

beschreibt wie stark Fehler verstärkt werden resp. welche Genauigkeit man bestenfalls bei gestörten Eingabedaten erwarten kann



absolute Fehler: $\|\vec{\Delta x}\|, \|\vec{\Delta y}\|$

relative Fehler: $\delta_x = \frac{\|\vec{\Delta x}\|}{\|\vec{x}\|}, \quad \delta_y = \frac{\|\vec{\Delta y}\|}{\|\vec{y}\|}$

absolute Kondition: $\frac{\|\vec{\Delta x}\|}{\|\vec{\Delta y}\|}$

relative Kondition: $\frac{\delta_y}{\delta_x}$

} je kleiner desto besser

falls $f: X \to Y$ eine reelle Funktion einer reellen Variable ist, also $X = Y = \mathbb{R}$ ist, gilt:

absolute Konditionszahl: $\kappa_{abs} = |f'(x)|$

relative Konditionszahl: $\kappa_{rel} = \left|f'(x)\frac{x}{f(x)}\right|$

} je kleiner desto besser

→ schlecht konditioniert: $\kappa_{rel} \gg 1$

Auslöschung

Verstärkung von Rundungsfehler bei Subtraktion zweier ungefähr gleich grosser Zahlen
→ schlecht konditionierte Operation

Stabilität

ein Algorithmus ist stabil, falls die während der Laufzeit erzeugten Fehler in der Grössenordnung des durch die Kondition bedingten unvermeidbaren Fehlers bleiben

digitale Zahlendarstellung

reelle Zahlen können i.A. nicht auf Computer dargestellt werden, da sie i.A. unendlich viele Nachkommastellen aufweisen
$\Rightarrow$ Rundungsfehler

Maschinenzahlen: (endliche) Menge der auf Computer darstellbaren Zahlen

normalisierte Gleitpunktdarstellung

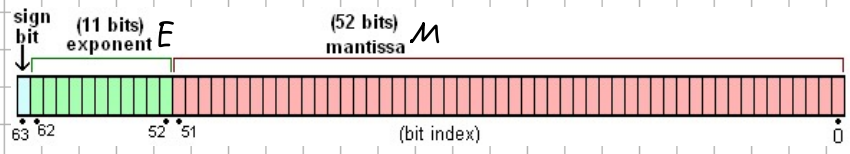
$x = f \cdot b^e$ wobei:

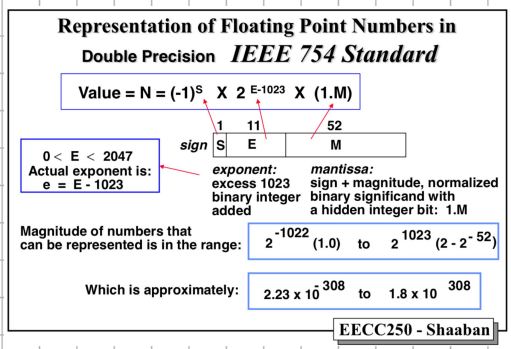
- b die Basis des Zahlensystems ist: $b \in \mathbb{N} \setminus \{1\}$
- der Exponent e eine ganze Zahl innerhalb gewisser fester Schranken ist: $r \leq e \leq R$
- die Mantisse/Fraktion f eine feste Anzahl m von Stellen hat:

$$f = \pm 0.d_1 \dots d_m = \pm\left(\sum_{j=1}^{m} d_j b^{-j}\right), \quad d_j \in \{0, 1, \dots, b-1\}$$

- für $x \neq 0$ stets $d_1 \neq 0$ ist (Eindeutigkeit der Darstellung)

Double/float64 nach IEEE



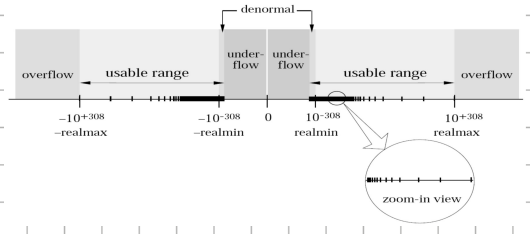


$$(-1)^{sign} \cdot 1.M \cdot 2^{E-(1023)_{10}}$$

$$= (-1)^{sign} \cdot (1.b_{51}b_{50}\dots b_0)_2 \cdot 2^{(b_{62}b_{61}\dots b_{52})_2 - (1023)_{10}}$$

$$= (-1)^{sign} \cdot (1 + b_{51}\cdot 2^{-1} + b_{50}\cdot 2^{-2} + \dots + b_1\cdot 2^{-51} + b_0\cdot 2^{-52})_{10} \cdot 2^{(b_{62}\cdot 2^{10} + b_{61}\cdot 2^9 + \dots + b_{53}\cdot 2 + b_{52}\cdot 1 - 1023)_{10}}$$

gewisse Bitmuster für $\pm\infty$, NaN, etc reserviert...



Maschinengenauigkeit

$eps = \frac{1}{2} \cdot 2^{1-m}$ mit m = Mantissenlänge (Anzahl Bits)

$\hat{=}$ Auflösungsvermögen des Rechners

$\hat{=}$ kleinst mögliche auf dem Rechner darstellbare Zahl, die zu 1 addiert von der Rundung nach wahrgenommen wird

für Double gilt mit $m = 52$: $eps \approx 2.22 \cdot 10^{-16}$

$\rightarrow$ falls $K \geq \frac{1}{eps} \approx 10^{15}$ $\Rightarrow$ schlecht konditioniert

```
import numpy as np
print(np.finfo(float).eps)
```

```
2.220446049250313e-16
```

Addition von mehreren Zahlen

man sollte Summanden in (betragsmässig) aufsteigender Reihenfolge addieren, um ein genaueres Ergebnis zu erhalten

Fehlerordnung

Fehler p-ter Ordnung -> wenn man Schrittweite halbiert, verringert sich der Fehler um $(1/2)^p$

symbolisch: $O(h^p)$ je grösser p, desto besser

Ableitung numerisch berechnen

Approximation der ersten Ableitung:

Rückwärtsdifferenz:

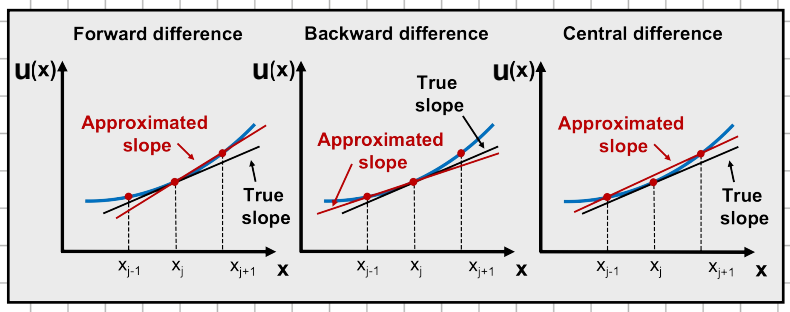
$$\frac{\partial}{\partial x} u(jh) \approx \frac{u_j - u_{j-1}}{h} \quad \text{mit } O(h)$$

Vorwärtsdifferenz:

$$\frac{\partial}{\partial x} u(jh) \approx \frac{u_{j+1} - u_j}{h} \quad \text{mit } O(h)$$

zentrale Differenz:

$$\frac{\partial}{\partial x} u(jh) \approx \frac{u_{j+1} - u_{j-1}}{2h} \quad \text{mit } O(h^2)$$



Approximation der zweiten Ableitung:

zentrale zweite Differenz:

$$\frac{\partial^2}{\partial x^2} u(jh) \approx \frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} \quad \text{mit } O(h^2)$$

Lineare Gleichungssysteme

Das Lineare Gleichungssystem

$$\begin{cases} a_{1,1}x_1 + \dots + a_{1,n}x_n = b_1 \\ \vdots \qquad\qquad \vdots \qquad\quad \vdots \\ a_{n,1}x_1 + \dots + a_{n,n}x_n = b_n \end{cases} \quad \text{bzw.} \quad \underline{A}\cdot\vec{x} = \vec{b} \quad \text{mit} \quad \vec{x}\in\mathbb{R}^n,\ \underline{A}\in\mathbb{R}^{n\times n},\ \vec{b}\in\mathbb{R}^n$$

hat eine eindeutige Lösung falls:

$$\operatorname{rang}(\underline{A}) = n \iff \det(\underline{A}) \neq 0 \iff \ker(\underline{A}) = \{\vec{0}\} \iff \underline{A} \text{ invertierbar}$$

$\underline{A}$ heisst in diesem Fall regulär

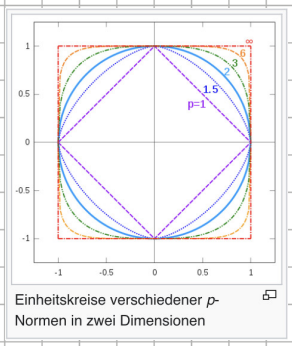
Cramersche Regel

$x_j = \dfrac{\det(\underline{A}_j)}{\det(\underline{A})}$, wobei $\underline{A}_j$ aus $\underline{A}$ entsteht indem man die j-te Spalte durch $\vec{b}$ ersetzt

numerisch sehr aufwändig

Vektornorm (p-Norm)

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

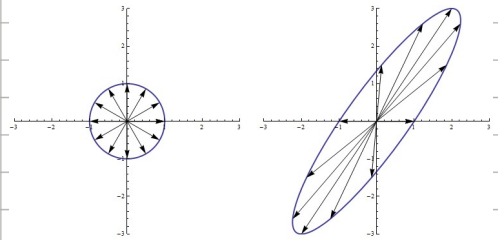


Einheitskreise verschiedener p-Normen in zwei Dimensionen

Matrixnorm (p-Norm)

$$\|\underline{B}\|_p := \sup_{\|\vec{x}\|_p = 1} \|\underline{B}\vec{x}\|_p$$

gibt an wie stark die Matrix $\underline{B}$ einen Vektor maximal verformt



für $\underline{B}\in\mathbb{R}^{m\times n}$ gilt:

$$\|\underline{B}\|_\infty = \max_{i=1,\dots,m} \sum_{k=1}^{n} |b_{i,k}|$$

$$\|\underline{B}\|_1 = \max_{k=1,\dots,n} \sum_{i=1}^{m} |b_{i,k}|$$

$$\underline{B} = \begin{bmatrix} b_{1,1} & \cdots & b_{1,n} \\ b_{2,1} & \cdots & b_{2,n} \\ \vdots & & \vdots \\ b_{m-1,1} & \cdots & b_{m-1,n} \\ b_{m,1} & \cdots & b_{m,n} \end{bmatrix}$$

für $\underline{A}\in\mathbb{R}^{n\times n}$ gilt:

$$\|\underline{A}\|_\infty = \max_{i=1,\dots,n} \sum_{k=1}^{n} |a_{i,k}|$$ maximale absolute Zeilensumme

$$\|\underline{A}\|_1 = \max_{k=1,\dots,n} \sum_{i=1}^{n} |a_{i,k}|$$ maximale absolute Spaltensumme

$$\|\underline{A}\|_2 = \sqrt{\lambda_{max}(\underline{A}^T\underline{A})}$$ Wurzel des grössten Eigenwerts von $\underline{A}^T\underline{A}$

Konditionszahl einer Matrix

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

worst-case-Wert für Fehlerverstärkungsfaktor

→ i.A. aufwändig zu berechnen, da Inverse aufwändig ist

Condition Number

- If A is square and nonsingular, then $\text{cond}(A) = \|A\| \cdot \|A^{-1}\|$
- If A is singular, then $\text{cond}(A) = \infty$
- If A is nearly singular, then $\text{cond}(A)$ is large.
- The condition number measures the ratio of maximum stretch to maximum shrinkage:

$$\|A\| \cdot \|A^{-1}\| = \left(\max_{x \neq 0} \frac{\|Ax\|}{\|x\|}\right) \cdot \left(\min_{x \neq 0} \frac{\|Ax\|}{\|x\|}\right)^{-1}$$

relativer Fehler der Lösung

$$\delta_{\vec{x}} = \frac{\|\Delta\vec{x}\|}{\|\vec{x}\|} = \frac{\|\vec{x} - \tilde{\vec{x}}\|}{\|\vec{x}\|}$$

$$\frac{\|\Delta\vec{x}\|}{\|\vec{x}\|} \leq \kappa(\underline{A}) \frac{\|\Delta\vec{b}\|}{\|\vec{b}\|}$$

Fehlerverstärkung von $\vec{b}$ zu $\vec{x}$

$$\text{Fehlerverstärkungsfaktor} := \frac{\delta_{\vec{x}}}{\delta_{\vec{b}}} = \frac{\frac{\|\Delta\vec{x}\|}{\|\vec{x}\|}}{\frac{\|\Delta\vec{b}\|}{\|\vec{b}\|}}$$

$$\frac{\delta_{\vec{x}}}{\delta_{\vec{b}}} \leq \kappa(\underline{A})$$

Residuum

$$\vec{r} = \vec{b} - \underline{A}\tilde{\vec{x}}$$

→ Mass für Genauigkeit einer Annäherung $\tilde{x}$ der Lösung des LGS $\underline{A}\vec{x} = \vec{b}$
→ Aussagekraft hängt von Kondition ab:

$$\frac{\|\vec{r}\|}{\kappa(\underline{A})\|\vec{b}\|} \leq \frac{\|\vec{x} - \tilde{\vec{x}}\|}{\|\vec{x}\|} \leq \frac{\kappa(\underline{A})\|\vec{r}\|}{\|\vec{b}\|}$$

falls schlecht konditioniert ⇒ $\kappa(A)$ gross
⇒ Schranken für den relativen Fehler liegen weit auseinander

Gauss-Elimination ohne Pivotisierung

gegeben: LGS $\underline{A}\vec{x} = \vec{b}$

gesucht: $\vec{x}$

Vorgehen:

1. normal "Gaussen": $(\underline{A}|\vec{b}) \rightsquigarrow \cdots \rightsquigarrow (\underline{R}|\vec{c})$ mit rechter/oberer Dreiecksmatrix $\underline{R}$
2. Rückwärtseinsetzen: $\underline{R}\vec{x} = \vec{c}$ nach $\vec{x}$ auflösen

LU-Zerlegung mit Skalierung und Pivotisierung

gegeben: $\underline{A}$

gesucht: $\underline{P}, \underline{D}, \underline{L}, \underline{R}$ sodass $\underline{LR} = \underline{PDA}$

$$\underline{PDA} = \underline{LR}$$

$$\begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix} \begin{bmatrix} d_1 & & \\ & d_2 & \\ & & \ddots \\ & & & d_n \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} 1 & & \\ * & 1 & \\ \vdots & \ddots & \ddots \\ * & \cdots & * & 1 \end{bmatrix} \begin{bmatrix} * & * & \cdots & * & * \\ & * & \cdots & * & * \\ & & \ddots & \vdots & \vdots \\ & & & * & * \\ & & & & * \end{bmatrix}$$

Vorgehen:

1. bestimme Diagonalmatrix $\underline{D}$, welche auf der Diagonale die Zeilenskalierungsfaktoren enthält:

$$\underline{D} = \text{diag}(d_1, \ldots, d_n) \quad \text{mit} \quad d_i = \frac{1}{\sum_{k=1}^{n} |a_{i,k}|}$$

Skalierung verbessert die Kondition der Matrix

2. LU-Zerlegung mit Spaltenpivotisierung auf $\underline{DA}$ anwenden:

im j-ten Schritt schaut man ob unter dem Pivotelement irgendwo eine betragsmässig grössere Zahl vorhanden ist
→ falls dies der Fall ist wird diese Zeile mit der j-ten vertauscht
→ dieselben Zeilen vertauscht man auch bei $\underline{P}$ (diese speichert alle Vertauschungen)

dann die Elemente unterhalb Pivotelement eliminieren und die Faktoren in $\underline{L}$ speichern
→ dabei Folgendes beachten:
- nicht Zeilen mit Skalar multiplizieren!
- nur Vielfaches (Faktor a) der Pivotzeile zu unteren Zeile addieren
- Faktor a in Spalte von $\underline{L}$ notieren, deren Nummer mit der Nummer der Pivotzeile übereinstimmt

für $\underline{L}$: Zeile, Faktor, Spalte

$$I - a \cdot J$$

$\underline{P}$ (Permutationsmatrix):
Matrix mit Nullen und Einsen welche die Information über die Zeilenvertauschungen enthält
→ entsteht aus der Einheitsmatrix indem man alle Zeilenvertauschungen, welche man bei $\underline{A}$ vornimmt auch bei der Einheitsmatrix $\underline{E_n}$ macht

$$\underline{P} = \underline{P_{n-1,r_{n-1}}}\ \underline{P_{n-2,r_{n-2}}} \cdots \underline{P_{2,r_2}}\ \underline{P_{1,r_1}}\ \underline{E_n}$$

$$\underline{P^{-1}} = \underline{P^T}$$

Pivotisierung verbessert die Stabilität der LU-Zerlegung

LGS mit LU-Zerlegung lösen

Gegeben: $\underline{A}, \vec{b}$ sodass $\underline{A}\vec{x} = \vec{b}$ und $\underline{P}, \underline{D}, \underline{L}, \underline{R}$ sodass $\underline{P}\underline{D}\underline{A} = \underline{L}\underline{R}$

Gesucht: $\vec{x}$

Vorgehen:

1. $\underline{L}\vec{y} = \underline{P}\underline{D}\vec{b}$ nach $\vec{y}$ auflösen (Vorwärtseinsetzen)
2. $\underline{R}\vec{x} = \vec{y}$ nach $\vec{x}$ auflösen (Rückwärtseinsetzen)

Inverse mit LU-Zerlegung

Gegeben: $\underline{A}$

Gesucht: $\underline{A}^{-1} := \begin{bmatrix} | & & | \\ \vec{x}_1 & \cdots & \vec{x}_n \\ | & & | \end{bmatrix}$

Vorgehen:

$\underline{A}\vec{x}_i = \vec{e}_i$ für $i = 1, \ldots, n$ lösen, wobei $\vec{e}_i$ der i-te (kanonische) Einheitsvektor ist

Nachiteration

wegen Rundungsfehler hat man nicht exakt $\underline{L}$ und $\underline{R}$ sondern Annäherungen $\tilde{\underline{L}}$ und $\tilde{\underline{R}}$

beim Lösen von $\underline{A}\vec{x} = \vec{b}$ erhält man somit auch nur eine Näherung $\vec{x}_0$

es gibt also ein Residuum: $\vec{r}_0 = \vec{b} - \underline{A}\vec{x}_0$

mithilfe von Nachiteration kann man das Residuum iterativ verkleinern:

Pseudocode:

```
Für k = 0, 1, 2, ..., nr_of_iter berechne:
    L y_k = r_k;      // Vorwärtseinsetzen
    R δ_k = y_k;      // Rückwärtseinsetzen
    x_{k+1} = x_k + δ_k;   // Lösung updaten/verbessern
    r_{k+1} = b - A x_{k+1};   // Residuum berechnen
```

Cholesky-Zerlegung, LDLT-Zerlegung

effizientere Alternative zu LU für sym. pos. def. Matrizen (spd-Matrizen)
→ etwa doppelt so schnell um LGS zu lösen

falls $\underline{A} \in \mathbb{R}^{n \times n}$ symmetrisch ($\underline{A}^T = \underline{A}$) und positiv definit ($\vec{x}^T \underline{A} \vec{x} > 0$ für alle $\vec{x}$) ist:

- $\underline{A}$ hat vollen Rang
- Inverse $\underline{A}^{-1}$ ist auch sym. pos. def.
- alle Eigenwerte von $\underline{A} > 0$
- Determinante $\det(\underline{A}) > 0$
- alle Diagonaleinträge von $\underline{A} > 0$
- grösster Eintrag von $\underline{A}$ liegt auf der Diagonalen
- $\underline{A}$ besitzt (eindeutige) Cholesky-Zerlegung

gegeben: sym. pos. def. Matrix $\underline{A}$
gesucht: $\underline{L}, \underline{D}$ sodass $\underline{A} = \underline{L}\underline{D}\underline{L}^T$ mit { normierter linker/unterer Dreiecksmatrix $\underline{L}$; Diagonalmatrix $\underline{D}$ mit $d_{i,i} > 0 \; \forall i$

$$\underline{A} = \underline{L}\underline{D}\underline{L}^T \quad \rightarrow \text{entspricht LU-Zerlegung für } \underline{R} = \underline{D}\underline{L}^T$$

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{bmatrix} = \begin{bmatrix} 1 & & & \\ l_{2,1} & 1 & & \\ \vdots & l_{3,2} & \ddots & \\ l_{n,1} & l_{n,2} & \cdots & 1 \end{bmatrix} \begin{bmatrix} d_{1,1} & & & \\ & d_{2,2} & & \\ & & \ddots & \\ & & & d_{n,n} \end{bmatrix} \begin{bmatrix} 1 & l_{2,1} & \cdots & l_{n,1} \\ & 1 & l_{3,2} & \cdots \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix}$$

oder: alles in einer Matrix speichern → für in-place Implementierung

$$\underline{L}^{ip} = \begin{bmatrix} d_{1,1} & & & \\ l_{2,1} & d_{2,2} & & \\ \vdots & l_{3,2} & \ddots & \\ l_{n,1} & l_{n,2} & \cdots & d_{n,n} \end{bmatrix}$$

alternative Form → Cholesky-Zerlegung $\underline{A} = \underline{C}\underline{C}^T$

LDLT kann einfach in Cholesky umgeformt werden:

$\underline{A} = \underline{C}\underline{C}^T$ wobei $\underline{C} = \underline{L}\underline{D}^{1/2}$, $\underline{D}^{1/2} = \text{diag}(\sqrt{d_{1,1}}, \ldots, \sqrt{d_{n,n}})$ Diagonalelemente von $\underline{C}$

umgekehrt kann man mit den Diagonalelementen von $\underline{C}$ wieder $\underline{L}$ erhalten:

$\underline{L} = \underline{C}\underline{D}^{-1/2}$, $\underline{D}^{-1/2} = \text{diag}\left(\frac{1}{\sqrt{d_{1,1}}}, \ldots, \frac{1}{\sqrt{d_{n,n}}}\right)$ Kehrwerte der Diagonalelemente von $\underline{C}$

Pseudocode in-place LDLT-Zerlegung

Für $k = 1, 2, \ldots, n$:
  $d_{k,k} \leftarrow a_{k,k} - \sum_{j=1}^{k-1} l_{k,j}^2 d_{j,j}$;
  Für $i = k+1, \ldots, n$:
  $l_{i,k} \leftarrow (a_{i,k} - \sum_{j=1}^{k-1} l_{i,j} d_{j,j} l_{k,j}) / d_{k,k}$;

Pseudocode Cholesky-Zerlegung

Für $k = 1, 2, \ldots, n$:
  $c_{k,k} \leftarrow \text{sqrt}(a_{k,k} - \sum_{j=1}^{k-1} c_{k,j}^2)$;
  Für $i = k+1, \ldots, n$:
  $c_{i,k} \leftarrow (a_{i,k} - \sum_{j=1}^{k-1} c_{i,j} c_{k,j}) / c_{k,k}$;

LGS $\underline{A}\vec{x} = \vec{b}$ mit LDLT:

1. $\underline{L}\vec{y} = \vec{b}$ nach $\vec{y}$ auflösen
2. $\underline{L}^T\vec{x} = \underline{D}^{-1}\vec{y}$ nach $\vec{x}$ auflösen

LGS $\underline{A}\vec{x} = \vec{b}$ mit Cholesky:

1. $\underline{C}\vec{y} = \vec{b}$ nach $\vec{y}$ auflösen
2. $\underline{C}^T\vec{x} = \vec{y}$ nach $\vec{x}$ auflösen

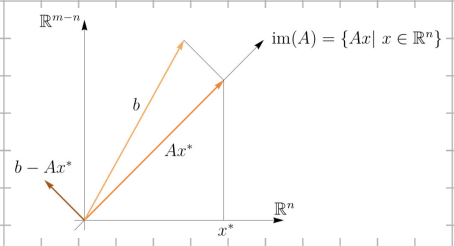
Lineare Ausgleichsrechnung

Lineares Ausgleichproblem

Gegeben: LGS $\underline{A}\vec{x} = \vec{b}$ mit $\underline{A} \in \mathbb{R}^{m \times n}$, $m > n$ (Zeilen > Spalten)

=> überbestimmtes LGS ($\hat{=}$ mehr Messungen als Modellparameter)
=> i.A. keine exakte Lösung

beste Lösung (im Sinne von kleinste Fehlerquadrate):



Normalgleichung: $\boxed{\underline{A}^T\underline{A}\vec{x}^* = \underline{A}^T\vec{b}}$

=> $\vec{x}^*$ minimiert (quadratischen) Fehler: $\|\vec{b} - \underline{A}\vec{x}^*\|^2 = \min_{\vec{x} \in \mathbb{R}^n} \|\vec{b} - \underline{A}\vec{x}\|$

Normalgleichung mit LDLT lösen

1. berechne LDLT-Zerlegung von $\underline{A}^T\underline{A}$: $\underline{L}\,\underline{D}\,\underline{L}^T = \underline{A}^T\underline{A}$
2. $\underline{L}\vec{y} = \underline{A}^T\vec{b}$ nach $\vec{y}$ auflösen (Vorwärtseinsetzen)
3. $\underline{L}^T\vec{x} = \underline{D}^{-1}\vec{y}$ nach $\vec{x}$ auflösen (Rückwärtseinsetzen)

Nachteile:
- Berechnung von $\underline{A}^T\underline{A}$ für grosse m aufwendig und teilweise ungenau
- Rundungsfehler in $\underline{A}^T\underline{A}$ und $\underline{A}^T\vec{b}$ werden über Cholesky-Verfahren teilweise verstärkt
- kann im Extremfall instabil werden

-> mit QR-Zerlegung versucht man diese Probleme zu umgehen
↳ stabiler und nur leicht höherer Rechenaufwand als LDLT

spektrale Konditionszahl einer Matrix $\underline{A} \in \mathbb{R}^{m \times n}$

$$\kappa_2(\underline{A}) = \frac{\sqrt{\lambda_{max}(\underline{A}^T\underline{A})}}{\sqrt{\lambda_{min}(\underline{A}^T\underline{A})}}$$

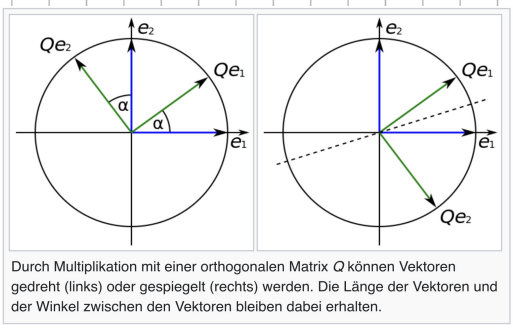
Wurzel des grössten Eigenwerts von $\underline{A}^T\underline{A}$ geteilt durch Wurzel des kleinsten Eigenwerts von $\underline{A}^T\underline{A}$

falls Spalten von $\underline{A}$ linear abhängig ($\underline{A}$ singulär) => $\kappa_2(\underline{A}) = \infty$

orthogonale Matrizen

Definition: eine Matrix $\underline{Q} \in \mathbb{R}^{m \times m}$ ist orthogonal falls $\underline{Q}^T \underline{Q} = \underline{E}_m$

Inverse: $\boxed{\underline{Q}^{-1} = \underline{Q}^T}$



Durch Multiplikation mit einer orthogonalen Matrix Q können Vektoren gedreht (links) oder gespiegelt (rechts) werden. Die Länge der Vektoren und der Winkel zwischen den Vektoren bleiben dabei erhalten.

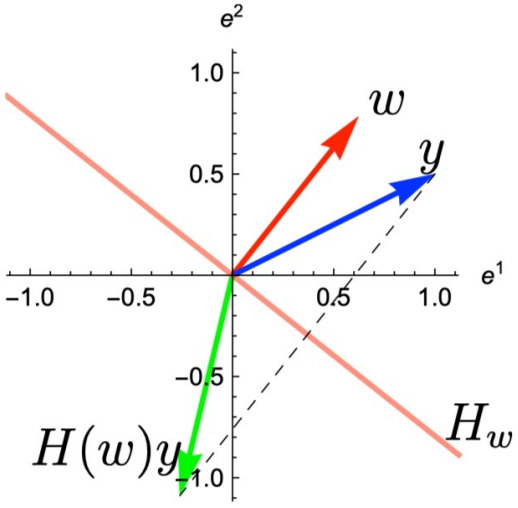
Eigenschaften:

- $\underline{Q}$ orthogonal $\Rightarrow$ $\underline{Q}^T$ orthogonal
- $\underline{Q}, \tilde{\underline{Q}}$ orthogonal $\Rightarrow$ $\underline{Q}\tilde{\underline{Q}}$ orthogonal
- Konditionszahl: $\kappa_2(\underline{Q}) = 1$
- $\kappa(\underline{A}) = \kappa(\underline{A}\underline{Q}) = \kappa(\underline{Q}\underline{A})$
- $\|\underline{Q}\vec{x}\| = \|\vec{x}\|$ für beliebigen Vektor $\vec{x}$
- $\|\underline{A}\| = \|\underline{Q}\underline{A}\| = \|\underline{A}\underline{Q}\|$ für beliebige Matrix $\underline{A}$

Householder-Matrix

$$\underline{H}(\vec{w}) = \underline{E}_n - 2\frac{\vec{w}\vec{w}^T}{\langle \vec{w}, \vec{w} \rangle}$$

$$= \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix} - \frac{2}{\langle \vec{w}, \vec{w} \rangle} \begin{bmatrix} w_1 w_1 & \cdots & w_1 w_n \\ \vdots & & \vdots \\ w_n w_1 & \cdots & w_n w_n \end{bmatrix}$$



$\underline{H}(\vec{w})\,\vec{y}$ entspricht der Spiegelung von $\vec{y}$ an der durch $\vec{w}$ definierten Hyperebene H_w ($\vec{w}$ steht senkrecht auf dieser Ebene)

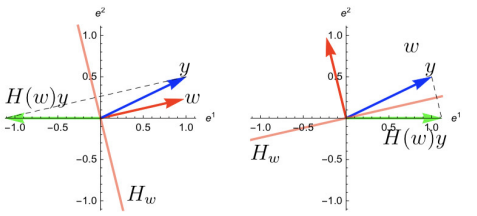
$\underline{H}(\vec{w})$ ist symmetrisch ($\underline{H}^T = \underline{H}$) und orthogonal ($\underline{H}^{-1} = \underline{H}^T$)

Bei der QR-Zerlegung wählt man $\vec{w}$ in jedem Schritt so, dass $\vec{y}$ durch die Spiegelung auf der $\vec{e}_1$-Achse landet:

$$\underline{H}(w)\,\vec{y} \overset{!}{=} \pm \|\vec{y}\|_2 \vec{e}_1$$

$$\Rightarrow \boxed{\vec{w} = \vec{y} \pm \text{sign}(y_1)\,\|\vec{y}\|\vec{e}_1}$$

wobei $\text{sign}(s) = \begin{cases} 1 & \text{für } s \geq 0 \\ -1 & \text{sonst} \end{cases}$



Um Auslöschung zu vermeiden wählt man: $\vec{w} = \vec{y} + \text{sign}(y_1)\,\|\vec{y}\|\vec{e}_1$

QR-Zerlegung

1. $\underline{Q_1 A} = \left(\begin{bmatrix} \underline{H(\vec{w}^{(1)})} \end{bmatrix}\right) \begin{pmatrix} | & * \cdots * \\ \vec{y}^{(1)} & \vdots \quad \vdots \\ | & * \cdots * \end{pmatrix} = \begin{pmatrix} * & * & * \cdots & * \\ 0 & | & * \cdots & * \\ \vdots & \vec{y}^{(2)} & \vdots & \vdots \\ 0 & | & * \cdots & * \end{pmatrix} = \underline{A^{(2)}}$

2. $\underline{Q_2 Q_1 A} = \begin{pmatrix} 1 & 0 \cdots 0 \\ 0 & \\ \vdots & \underline{H(\vec{w}^{(2)})} \\ 0 & \end{pmatrix} \begin{pmatrix} * & * & * \cdots & * \\ 0 & | & * \cdots & * \\ \vdots & \vec{y}^{(2)} & \vdots & \vdots \\ 0 & | & * \cdots & * \end{pmatrix} = \begin{pmatrix} * & * & \cdots & * \\ 0 & * & * & \cdots * \\ \vdots & 0 & | & * \cdots * \\ 0 & 0 & \vec{y}^{(3)} & \vdots \\ & & | & * \cdots * \end{pmatrix} = \underline{A^{(3)}}$

3. $\underline{Q_3 Q_2 Q_1 A} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 \cdots & 0 \\ \vdots & 0 & & \\ 0 & 0 & \underline{H(\vec{w}^{(3)})} & \end{pmatrix} \begin{pmatrix} * & * & \cdots & * \\ 0 & * & * & \cdots * \\ \vdots & 0 & | & * \cdots * \\ & \vdots & \vec{y}^{(3)} & \vdots \\ 0 & 0 & | & * \cdots * \end{pmatrix} = \begin{pmatrix} * & * & \cdots & * \\ 0 & * & * & \cdots * \\ & 0 & * & * \cdots * \\ \vdots & \vdots & 0 & | * \cdots * \\ & & \vec{y}^{(4)} & \vdots \\ 0 & 0 & 0 & | * \cdots * \end{pmatrix} = \underline{A^{(4)}}$

usw...

im i-ten Schritt wählt man jeweils: $\vec{w}^{(i)} = \vec{y}^{(i)} + sign(y_1^{(i)}) \|\vec{y}^{(i)}\| \vec{e}_1$

wobei $sign(s) = \begin{cases} 1 & \text{für } s \geq 0 \\ -1 & \text{sonst} \end{cases}$

→ damit eliminiert man alle Einträge von $\vec{y}^{(i)}$ ausser dem obersten

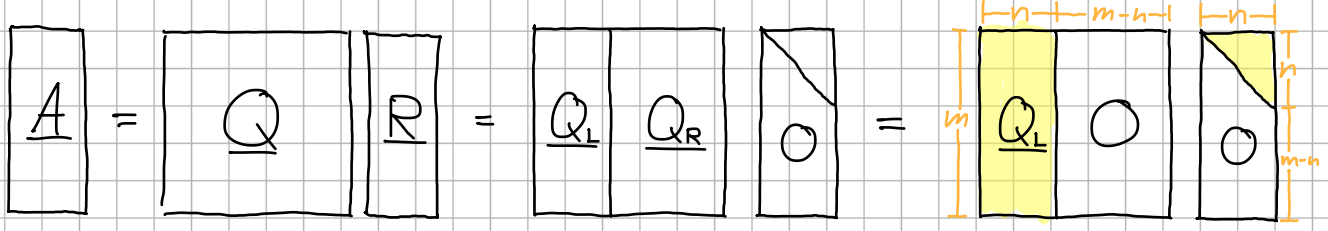
man erhält somit eine rechte obere Dreiecksmatrix $\underline{R} = \underline{Q_n} \cdots \underline{Q_1 A}$

mit $\underline{Q} := (\underline{Q_n} \cdots \underline{Q_1})^{-1} \overset{*}{=} (\underline{Q_n} \cdots \underline{Q_1})^T = \underline{Q_1^T} \cdots \underline{Q_n^T}$ folgt:

$\underline{A} = \underline{Q}\,\underline{R}$ wobei $\underline{Q} = \underline{Q_1^T} \cdots \underline{Q_n^T}$

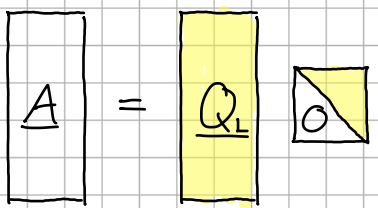
*: weil orthogonal

falls $m > n$:



=> man benötigt nur die ersten n Zeilen von $\underline{R}$ und erste n Spalten von $\underline{Q}$

=> reduzierte QR-Zerlegung:



Normalengleichung mit QR von $\underline{A}$ (!) lösen

$$\underline{A}^T\underline{A}\vec{x} = \underline{A}^T\vec{b} \Leftrightarrow (\underline{Q}\underline{R})^T\underline{Q}\underline{R}\vec{x} = (\underline{Q}\underline{R})^T\vec{b} \Leftrightarrow \underline{R}^T\underbrace{\underline{Q}^T\underline{Q}}_{=\underline{E}}\underline{R}\vec{x} = \underline{R}^T\underline{Q}^T\vec{b} \Leftrightarrow \underline{R}^T\underline{R}\vec{x} = \underline{R}^T\underline{Q}^T\vec{b}$$

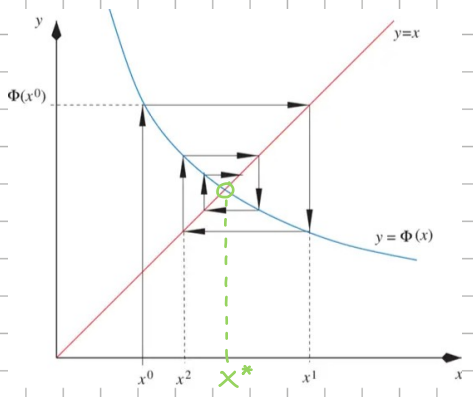
$$\Leftrightarrow \boxed{\underline{R}\vec{x} = \underline{Q}^T\vec{b} = \vec{c}}$$

$$\begin{bmatrix} \diagdown & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} | \\ \vec{x} \\ | \end{bmatrix} = \boxed{\underline{Q}^T}\begin{bmatrix} | \\ \vec{b} \\ | \end{bmatrix} = \begin{bmatrix} | \\ \vec{c} \\ | \end{bmatrix} \rightarrow \text{nach } \vec{x} \text{ auflösen (Rückwärtseinsetzen)}$$

Vorteil: i.A. schlecht konditionierte Matrix $\underline{A}^T\underline{A}$ kommt nicht mehr vor, da die QR-Zerlegung von $\underline{A}$ direkt verwendet werden kann

Nichtlineare Gleichungssysteme ($m=n$ resp. # Gleichungen = # Unbekannte)

Fixpunktiteration



Fixpunktgleichung: $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} \Phi_1(x_1,\dots,x_n) \\ \vdots \\ \Phi_n(x_1,\dots,x_n) \end{pmatrix}$ bzw $\vec{\Phi}(\vec{x}) = \vec{x}$

gesucht: Fixpunkt $\vec{x}^*$, sodass Fixpunktgleichung $\vec{\Phi}(\vec{x}^*) = \vec{x}^*$ erfüllt ist

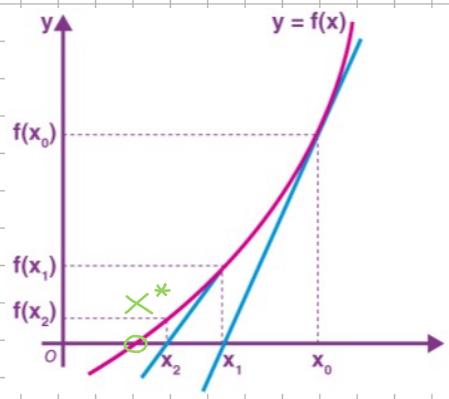
Iterationsvorschrift: $\boxed{\vec{x}_{k+1} = \vec{\Phi}(\vec{x}_k)}$

Voraussetzungen:

- es muss Fixpunktgleichung $\vec{\Phi}(\vec{x}) = \vec{x}$ vorliegen → falls man $\vec{f}(\vec{x}) = \vec{0}_n$ hat, muss man $\vec{x}$ auf beiden Seiten addieren: $\vec{\Phi}(\vec{x}) := \vec{x} + \vec{f}(\vec{x}) = \vec{x}$
- $\vec{\Phi}: E \to E$ mit $E \subseteq \mathbb{R}^n$ «Selbstabbildung»
- $\max \| \underline{D\Phi}(\vec{x}) \| < 1$ «Kontraktion»

⇒ Dann existiert genau ein Fixpunkt $\vec{x}^* \in E$ und Fixpunktiteration konvergiert linear gegen $\vec{x}^*$

Newtonverfahren



Nullstellengleichung: $\begin{pmatrix} f_1(x_1,\dots,x_n) \\ \vdots \\ f_n(x_1,\dots,x_n) \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$ bzw $\vec{f}(\vec{x}) = \vec{0}_n$

gesucht: Lösung $\vec{x}^*$, sodass Nullstellengleichung $\vec{f}(\vec{x}^*) = \vec{0}_n$ erfüllt ist

Iterationsvorschrift:

$\boxed{\begin{aligned} &\underline{Df}(\vec{x}_k)\,\vec{\delta}_k = \vec{f}(\vec{x}_k) \quad \text{nach } \vec{\delta}_k \text{ auflösen} \\ &\vec{x}_{k+1} = \vec{x}_k - \vec{\delta}_k \end{aligned}}$

falls eine Lösung $\vec{x}^*$ existiert und man gute Startwerte (idealerweise eine Näherungslösung für $\vec{x}^*$) hat, konvergiert das Newtonverfahren unter gewissen Voraussetzungen quadratisch gegen $\vec{x}^*$

mögliches Problem:
falls Jakobi in einer Nullstelle $\vec{x}^*$ singulär (nicht invertierbar), weil bspw. die Graphen der Einzelnen Komponenten f_i die $x_1 \dots x_n$-Ebene bei $\vec{x}^*$ nur berühren, aber keinen Nulldurchgang haben, kann $\|\vec{\delta}_k\|$ in der Nähe von $\vec{x}^*$ sehr gross werden und das Verfahren divergiert

vereinfachtes Newtonverfahren

um Rechenaufwand zu reduzieren benutzt man oft auch das Vereinfachte Newtonverfahren, bei dem man die Jakobi, bzw. dessen LU-Zerlegung, nur einmal, am Anfang, für den Startwert berechnet:

Iterationsvorschrift:

$$\underline{Df}(\vec{x}_0)\,\vec{\delta}_k = \vec{f}(\vec{x}_k) \text{ nach } \vec{\delta}_k \text{ auflösen} \quad (\text{am besten mit LU von } \underline{Df}(\vec{x}_0))$$

$$\vec{x}_{k+1} = \vec{x}_k - \vec{\delta}_k$$

einzelner Schritt weniger aufwändig, braucht jedoch normalerweise mehr Schritte

Nichtlineare Ausgleichsrechnung ($m > n$ resp # Gleichungen > # Unbekannte)

Motivation:

m Messungen b_i zu Zeitpunkten t_i, $i \in \{1, \dots, m\}$, für math. Modell mit n Parametern x_j, $j \in \{1, \dots, n\}$:

$$\begin{cases} y(t_1, \vec{x}) = b_1 \\ \vdots \\ y(t_m, \vec{x}) = b_m \end{cases} \Leftrightarrow \begin{cases} y(t_1, \vec{x}) - b_1 = 0 \\ \vdots \\ y(t_m, \vec{x}) - b_m = 0 \end{cases} \quad \text{resp} \quad \vec{F}(\vec{x}) = \vec{0}_m$$

mit $F_i(\vec{x}) := y(t_i, \vec{x}) - b_i$

Problemstellung:

gegeben: $\vec{F}: D \subseteq \mathbb{R}^n \to \mathbb{R}^m$ mit $m > n$ $\Rightarrow$ überbestimmtes nichtlineares GS

gesucht: $\vec{x}^*$, sodass $\vec{F}(\vec{x}^*) = \vec{0}_m$

$\Rightarrow$ i.A. keine Lösung, da Bild $\text{im}(\vec{F})$ ein nur n-dimensionales Objekt im m-dimensionalen $\mathbb{R}^m$ ist

$\Rightarrow$ man minimiert stattdessen die Residuenquadratsumme $\|\vec{F}(\vec{x})\|^2 = F_1(\vec{x})^2 + \dots + F_m(\vec{x})^2$ resp die Hälfte davon $\frac{1}{2}\|\vec{F}(\vec{x})\|^2 = \frac{1}{2}\vec{F}(\vec{x})^T \vec{F}(\vec{x}) =: \phi(\vec{x})$ → haben die gleichen Minima

ϕ hat in $\vec{x}^*$ ein lokales Minimum genau dann wenn
- $\vec{\nabla}\phi(\vec{x}^*) = \vec{0}_n$
- $\underline{H\phi}(\vec{x}^*)$ sym. pos. def.

Gradient von ϕ: $\vec{\nabla}\phi(\vec{x}) = \underline{DF}(\vec{x})^T \vec{F}(\vec{x})$

Hessesche Matrix von ϕ: $\underline{H\phi}(\vec{x}) = \underline{DF}(\vec{x})^T \underline{DF}(\vec{x}) + \sum_{i=1}^{m} F_i(\vec{x})\, \underline{HF_i}(\vec{x})$

numerisch aufwändig → wird oft weggelassen

Newton-Verfahren

Idee: Nullstelle von $\vec{\nabla}\phi$ bestimmen → normales Newtonverfahren auf $\vec{\nabla}\phi(\vec{x})$ anwenden

Iterationsvorschrift:

$\underline{H\phi}(\vec{x}_k)\,\vec{\delta}_k = \vec{\nabla}\phi(\vec{x}_k)$ nach $\vec{\delta}_k$ auflösen

$\vec{x}_{k+1} = \vec{x}_k - \vec{\delta}_k$

Gauss-Newton-Verfahren

Idee: $\vec{F}(\vec{x})$ in jedem Schritt mit 1ten Taylorpolynom linearisieren und ein Minimum davon mit linearer Ausgleichsrechnung finden

$$\vec{F}(\vec{x}) \approx \vec{F}(\vec{x}_k) + \underline{DF}(\vec{x}_k)\underbrace{(\vec{x} - \vec{x}_k)}_{=\vec{\delta}} \stackrel{!}{=} \vec{0}_m$$

$$\Rightarrow \underset{m\times n}{\underline{DF}(\vec{x}_k)}\ \underset{n}{\vec{\delta}} = -\underset{m}{\vec{F}(\vec{x}_k)} \quad \rightarrow \text{überbestimmtes LGS, keine Lösung}$$

$$\Rightarrow \underbrace{\underline{DF}(\vec{x}_k)^T\,\underline{DF}(\vec{x}_k)}_{n\times n}\ \underset{n}{\vec{\delta}} = -\underbrace{\underline{DF}(\vec{x}_k)^T\,\vec{F}(\vec{x}_k)}_{n} \quad \rightarrow \text{Normalgleichung}$$

Pseudocode

Für $k = 0, 1, 2, \ldots$:

$\underline{J}_k = DF(\vec{x}_k)$, $\vec{y}_k = \vec{F}(\vec{x}_k)$;

$\underline{J}_k^T \underline{J}_k \vec{\delta}_k = -\underline{J}_k^T \vec{y}_k$ nach $\vec{\delta}_k$ auflösen;

$\vec{x}_{k+1} = \vec{x}_k + \vec{\delta}_k$;

Falls $\|\underline{DF}(\vec{x}_k)^T \vec{F}(\vec{x}_k)\| < tol$: // Norm des Gradienten von $\Phi = \frac{1}{2}\|\vec{F}(\vec{x})\|^2$

Abbruch; // falls keine Steigung mehr vorhanden => Minimum erreicht

um Stabilität zu verbessern wird Newtonschritt oft gedämpft
-> mit Dämpfung verhindert man, dass man zu weit springt und damit ansteigenden Bergauf wandert

Pseudocode für gedämpfte Variante

Für $k = 0, 1, 2, \ldots$:

$\underline{J}_k = DF(\vec{x}_k)$, $\vec{y}_k = \vec{F}(\vec{x}_k)$;

$\underline{J}_k^T \underline{J}_k \vec{\delta}_k = -\underline{J}_k^T \vec{y}_k$ nach $\vec{\delta}_k$ auflösen;

$\mu = 1.0$;

Solange $\|\vec{F}(\vec{x}_k + \mu\vec{\delta}_k)\|^2 > \|\vec{F}(\vec{x}_k)\|^2$ und $\mu > \mu_{min}$: // gehts bergauf?

$\mu = \mu/2$; // dann lieber weniger weit springen

$\vec{x}_{k+1} = \vec{x}_k + \mu\vec{\delta}_k$;

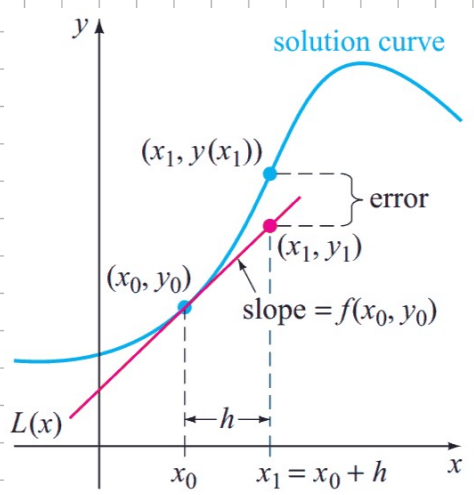
Falls $\|\underline{DF}(\vec{x}_k)^T \vec{F}(\vec{x}_k)\| < tol$: // Norm des Gradienten von $\Phi = \frac{1}{2}\|\vec{F}(\vec{x})\|^2$

Abbruch; // falls keine Steigung mehr vorhanden => Minimum erreicht

andere Variante um Stabilität zu verbessern: Levenberg-Marquardt-Verfahren

-> Kombination aus Newton-Verfahren und Gradient-Descent:
https://www.youtube.com/watch?v=dPZo74SbkeQ

gewöhnliche DGL (ODE)



Anfangswertproblem:

$$y'(x) = f(x, y), \quad x \in [a, b] \qquad \text{Steigung von } y \text{ bei } x$$
$$y(x_0) = y_0$$

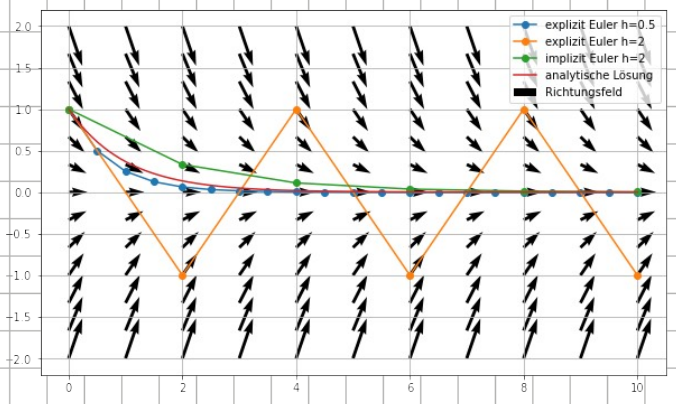
Idee:

$[a, b]$ äquidistant unterteilen:

$$a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$$
$$\text{mit } x_k = x_0 + k \cdot h, \quad h = \frac{b-a}{n}$$

ausgehend von $y_0 = y(x_0)$ mithilfe von $y'(x) = f(x, y)$ (→ Richtungsfeld) Schritt für Schritt Näherungen y_k für $y(x_k)$ bestimmen

einstufige Verfahren



Euler explizit:

mit Steigung am aktuellen Punkt (x_k, y_k) einen Schritt machen:

$$f(x_k, y_k) \approx \frac{y_{k+1} - y_k}{h}$$
$$\Rightarrow \boxed{y_{k+1} = y_k + h \cdot f(x_k, y_k)} \qquad \begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}$$

- unterschätzt exakte Lösung
- oszilliert bei zu grosser Schrittweite h

Euler implizit:

mit Steigung am nächsten Punkt (x_{k+1}, y_{k+1}) einen Schritt machen
nicht direkt möglich, da y_{k+1} noch nicht bekannt => implizite Gleichung für y_{k+1}

$$f(x_{k+1}, y_{k+1}) \approx \frac{y_{k+1} - y_k}{h}$$
$$\Rightarrow \boxed{y_{k+1} = y_k + h \cdot f(x_{k+1}, y_{k+1})} \qquad \begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$
→ mit Newtonverfahren nach y_{k+1} auflösen

- überschätzt exakte Lösung

allgemeines einstufiges RK-Verfahren

$$k = f(x_{i-1} + c \cdot h, y_{i-1} + h \cdot a \cdot k) \qquad \begin{array}{c|c} c & a \\ \hline & b \end{array}$$
$$y_i = y_{i-1} + h \cdot b \cdot k$$

→ explizit falls $a = 0$

mehrstufige Verfahren

Verfahren nach Runge (RK2):

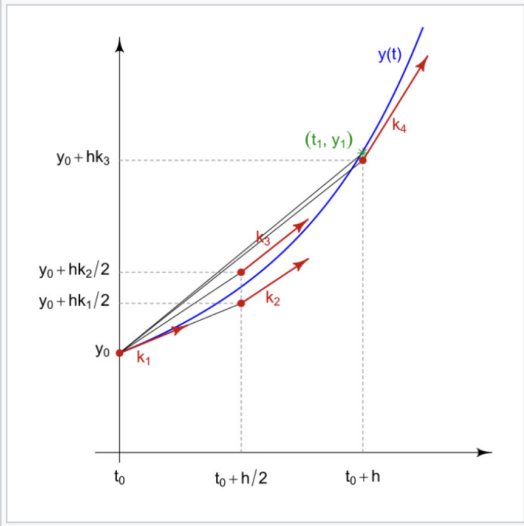
$$k_1 = f(x_{i-1}, y_{i-1})$$
$$k_2 = f(x_{i-1} + 1/2 \cdot h,\ y_{i-1} + 1/2 \cdot h \cdot k_1)$$
$$y_i = y_{i-1} + h \cdot 1 \cdot k_2$$

0	0	0
0.5	0.5	0
	0	1

Verfahren nach Heun:

$$k_1 = f(x_{i-1}, y_{i-1})$$
$$k_2 = f(x_{i-1} + 1 \cdot h,\ y_{i-1} + 1 \cdot h \cdot k_1)$$
$$y_i = y_{i-1} + h \cdot (1/2 \cdot k_1 + 1/2 \cdot k_2)$$

0	0	0
1	1	0
	0.5	0.5



Das klassische Runge-Kutta-Verfahren mittelt in jedem Schritt vier Hilfssteigungen (rot)

klassisches Runge-Kutta-Verfahren (RK4):

$$k_1 = f(x_{i-1}, y_{i-1})$$
$$k_2 = f(x_{i-1} + 1/2 \cdot h,\ y_{i-1} + 1/2 \cdot h \cdot k_1)$$
$$k_3 = f(x_{i-1} + 1/2 \cdot h,\ y_{i-1} + 1/2 \cdot h \cdot k_2)$$
$$k_4 = f(x_{i-1} + 1 \cdot h,\ y_{i-1} + 1 \cdot h \cdot k_3)$$
$$y_i = y_{i-1} + h \cdot \left(\tfrac{1}{6}k_1 + \tfrac{1}{3}k_2 + \tfrac{1}{3}k_3 + \tfrac{1}{6}k_4\right)$$

0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{2}$	0	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	0	0
1	0	0	1	0
	$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$

allgemeine s-stufige RK-Verfahren

$$k_1 = f(x_{i-1} + c_1 h,\ y_{i-1} + h(a_{11}k_1 + \cdots + a_{1s}k_s))$$
$$\vdots$$
$$k_s = f(x_{i-1} + c_s h,\ y_{i-1} + h(a_{s1}k_1 + \cdots + a_{ss}k_s))$$
$$y_i = y_{i-1} + h \cdot (b_1 k_1 + \cdots + b_s k_s)$$

c_1	a_{11}	$\cdots$	a_{1s}
$\vdots$	$\vdots$		$\vdots$
c_s	a_{s1}	$\cdots$	a_{ss}
	b_1	$\cdots$	b_s

resp

$\vec{c}$	A
	$\vec{b}^T$

(F)IRK-Verfahren (fully implicit):

mind. ein Eintrag von A oberhalb Hauptdiagonale $\neq 0$

man muss in jedem Schritt Stufengleichung $\vec{F}(\vec{k}) \stackrel{!}{=} \vec{0}_s$ lösen (z.B mit Newton-Verfahren) um die Steigungen $k_1, \ldots, k_s$ zu berechnen:

$$\vec{F}(\vec{k}) = \begin{pmatrix} F_1(k_1, \ldots, k_s) \\ \vdots \\ F_s(k_1, \ldots, k_s) \end{pmatrix} := \begin{pmatrix} k_1 - f(x_{i-1} + c_1 h,\ y_{i-1} + h(a_{11}k_1 + \cdots + a_{1s}k_s)) \\ \vdots \\ k_s - f(x_{i-1} + c_s h,\ y_{i-1} + h(a_{s1}k_1 + \cdots + a_{ss}k_s)) \end{pmatrix} \stackrel{!}{=} \vec{0}_s$$

DIRK-Verfahren (diagonally implicit):

A ist linke (untere) Dreiecksmatrix (nur Nullen oberhalb Hauptdiagonale)

Vorteil gegenüber FIRK: Stufengleichungen $\vec{F}(\vec{k}) \stackrel{!}{=} \vec{0}_s$ können sequenziell berechnet werden, da in j^{ter} Stufe $k_1, \ldots, k_{j-1}$ bereits bekannt sind

SDIRK-Verfahren (singly diagonally implicit):

A ist linke (untere) Dreiecksmatrix und alle Diagonalelemente sind gleich

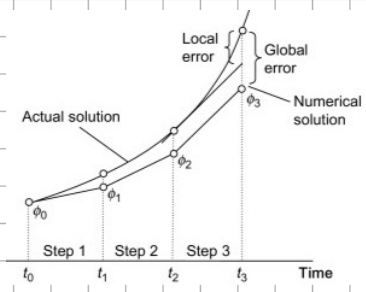
haben vor allem für grosse Systeme von gDgln einen Vorteil

ERK-Verfahren (explicit)

alle Einträge von A auf und oberhalb der Hauptdiagonale = 0

Vorteil: müssen keine Gleichungen gelöst werden, in der j^{ten} Stufe steht explizite Formel für k_j → können direkt eins nach dem andern berechnet werden

globaler Diskretisierungsfehler



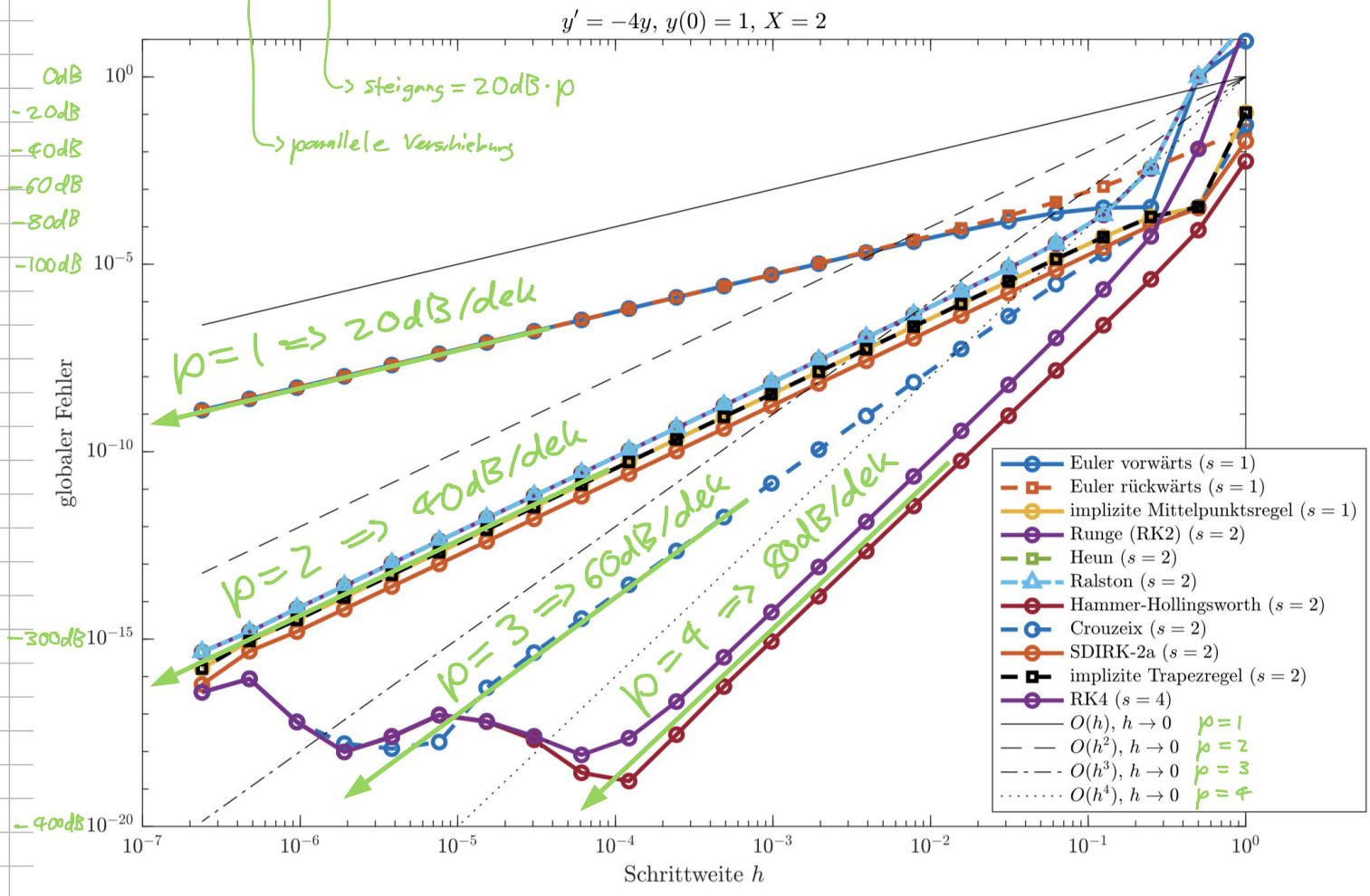
globaler Fehler an Stelle $x_k = a + kh$ ist definiert als:

$e(x_k, h) := y(x_k) - y_k$ (exakter Wert - numerische Approximation)

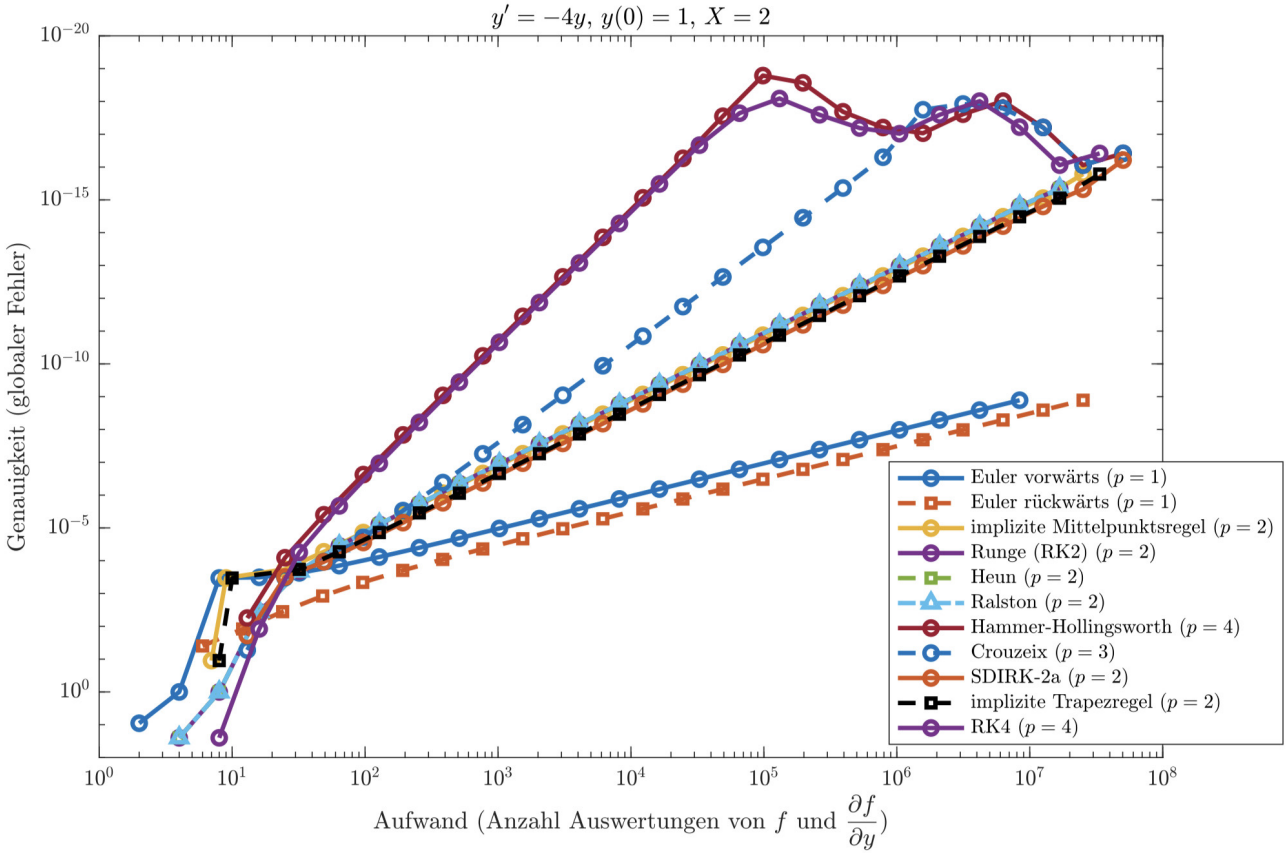
Konvergenzordnung

Verfahren hat Konvergenzordnung p (→ Steigung in loglog(h,e)) falls für irgendeine Konstante $C > 0$ (→ parallele Verschiebung in loglog(h,e))

$e(x_k, h) \leq C \cdot h^p$, $h \to 0$ für alle x_k gilt

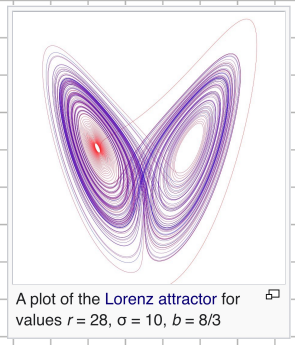


Genauigkeits-Aufwand-Diagramm



⇒ bei gleichem Aufwand höhere Genauigkeit mit mehrstufigen RK-Verfahren

Systeme von gewöhnlichen DGLn



A plot of the Lorenz attractor for values $r = 28$, $\sigma = 10$, $b = 8/3$

Anfangswertproblem:

$$\vec{y}'(x) = \vec{f}(x, \vec{y}) \quad \text{resp} \quad \begin{pmatrix} \frac{d}{dx} y_1(x) \\ \vdots \\ \frac{d}{dx} y_n(x) \end{pmatrix} = \begin{pmatrix} f_1(x, y_1, \dots y_n) \\ \vdots \\ f_n(x, y_1, \dots y_n) \end{pmatrix}, \quad x \in [a, b]$$
$$\vec{y}(x_0) = \vec{y}_0$$

alle Verfahren für gDGL kann man auf Systeme von gDGLn anwenden, indem man y durch $\vec{y}$, f durch $\vec{f}$, k_i durch $\vec{k}_i$ etc. ersetzt

allgemeine s-stufige RK-Verfahren

$$\begin{aligned} \vec{k}_1 &= \vec{f}(x_{i-1} + c_1 h, \vec{y}_{i-1} + h(a_{11}\vec{k}_1 + \dots + a_{1s}\vec{k}_s)) \\ &\vdots \\ \vec{k}_s &= \vec{f}(x_{i-1} + c_s h, \vec{y}_{i-1} + h(a_{s1}\vec{k}_1 + \dots + a_{ss}\vec{k}_s)) \\ \vec{y}_i &= \vec{y}_{i-1} + h \cdot (b_1\vec{k}_1 + \dots + b_s\vec{k}_s) \end{aligned}$$

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array} \quad \text{resp} \quad \begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b}^T \end{array}$$

(F)IRK-Verfahren (fully implicit):

man muss in jedem Schritt System von s·n nichtlineare Gleichungen für s·n Unbekannte lösen:

$$\vec{F}(\vec{k}) = \begin{pmatrix} \vec{F}_1(\vec{k}_1, \dots, \vec{k}_s) \\ \vdots \\ \vec{F}_s(\vec{k}_1, \dots, \vec{k}_s) \end{pmatrix} := \begin{pmatrix} \vec{k}_1 - \vec{f}(x_{i-1} + c_1 h, \vec{y}_{i-1} + h(a_{11}\vec{k}_1 + \dots + a_{1s}\vec{k}_s)) \\ \vdots \\ \vec{k}_s - \vec{f}(x_{i-1} + c_s h, \vec{y}_{i-1} + h(a_{s1}\vec{k}_1 + \dots + a_{ss}\vec{k}_s)) \end{pmatrix} \overset{!}{=} \begin{pmatrix} \vec{0}_n \\ \vdots \\ \vec{0}_n \end{pmatrix} = \vec{0}_{sn}$$

ERK-Verfahren (explicit)

man muss keine Gleichungen lösen, $\vec{k}_i$ können eins nach dem anderen direkt mit $\vec{f}$ berechnet werden

Stabilitätsanalyse

Stabilitätsfunktion $R: \mathbb{C} \to \mathbb{C}$ eines allg. s-stufigen RK-Verfahrens mit Verfahrensparameter $\underline{A} \in \mathbb{R}^{s \times s}$ und $\vec{b} \in \mathbb{R}^s$ ist gegeben durch:

$$R(z) = \frac{\det(\underline{E}_s - z\underline{A} + z\vec{1}_s\vec{b}^T)}{\det(\underline{E}_s - z\underline{A})} = \frac{\det\left(\begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix} - z\begin{bmatrix} a_{11} \cdots a_{1s} \\ \vdots \quad \vdots \\ a_{s1} \cdots a_{ss} \end{bmatrix} + z\begin{bmatrix} b_1 \cdots b_s \\ \vdots \quad \vdots \\ b_1 \cdots b_s \end{bmatrix}\right)}{\det\left(\begin{bmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{bmatrix} - z\begin{bmatrix} a_{11} \cdots a_{1s} \\ \vdots \quad \vdots \\ a_{s1} \cdots a_{ss} \end{bmatrix}\right)} \in \mathbb{C}, \quad z \in \mathbb{C}$$

Stabilitätsgebiet S des Verfahrens:

$$S = \{z \in \mathbb{C} \text{ für die } |R(z)| < 1 \text{ ist}\} \subseteq \mathbb{C}$$

Verfahren ist im Punkt $\vec{e}(x)$ des Richtungsfeldes stabil, falls für alle Eigenwerte λ_j der Jakobi $\underline{J} = \partial_{\vec{y}} \vec{f}(x, \vec{e}(x))$ das Produkt aus Schrittweite h und Eigenwert λ_j im Stabilitätsgebiet liegt:

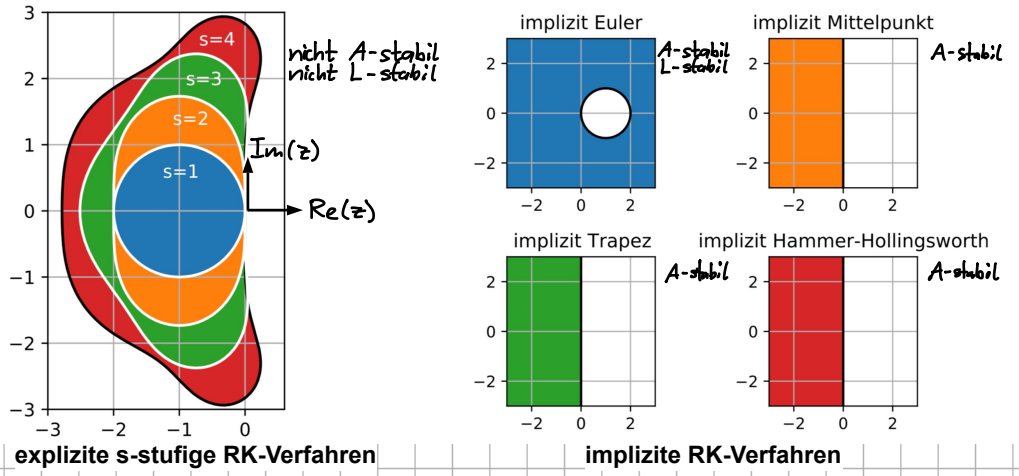
$$|R(h\lambda_j)| < 1 \quad \text{für alle Eigenwerte } \lambda_j \text{ von } \underline{J} \Rightarrow \text{Verfahren in } \vec{e}(x) \text{ stabil}$$

Verfahren ist

- A-stabil, falls „$\mathrm{Re}(z) < 0 \Rightarrow z \in S$" erfüllt ist
  → komplette linke Halbebene im Stabilitätsgebiet
- L-stabil, falls A-stabil und zusätzlich $\lim_{\mathrm{Re}(z)\to\infty} |R(z)| = 0$
  → wenn man weit rechts geht, kommt man irgendwann ins Stabilitätsgebiet

für explizite Verfahren ist S immer beschränkt $\Rightarrow$ nicht A-stabil

Stabilitätsbereiche S diverser RK-Verfahren:



Energieerhaltung

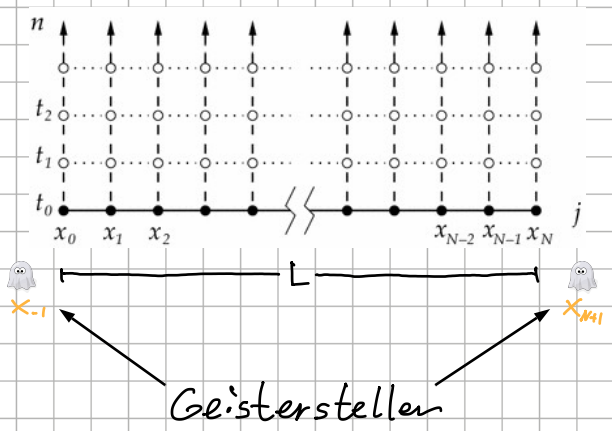
- konservativ $\Rightarrow$ Energie bleibt erhalten, falls dies beim physikalischen System auch so ist
- dissipativ $\Rightarrow$ Energie geht verloren → «numerische Reibung»
- anti-dissipativ $\Rightarrow$ Energie nimmt zu → unerwünscht!

Übersicht

Runge-Kutta-Verfahren	s	p		Stabilität	Energie, $\lambda = -\imath$, $h > 0$
Euler vorwärts	1	1	explizit	–	anti-dissipativ
Euler rückwärts	1	1	implizit	L-stabil	dissipativ
implizite Mittelpunktsregel	1	2	implizit	A-stabil	konservativ
Runge (RK2)	2	2	explizit	–	anti-dissipativ
Heun	2	2	explizit	–	anti-dissipativ
Ralston	2	2	explizit	–	anti-dissipativ
Hammer-Hollingsworth	2	4	implizit	A-stabil	konservativ
Crouzeix	2	3	implizit	L-stabil	dissipativ
SDIRK-2a	2	2	implizit	L-stabil	dissipativ
implizite Trapezregel	2	2	implizit	A-stabil	konservativ
RK4	4	4	explizit	–	anti-dissipativ für $h > 2\sqrt{2}$ konservativ für $h = 2\sqrt{2}$ dissipativ für $h < 2\sqrt{2}$

→ überschätzt exakte Lösung $\Rightarrow E\uparrow$
→ unterschätzt exakte Lösung $\Rightarrow E\downarrow$
→ Mischung aus beiden Euler $\Rightarrow E$ const.

partielle DGL (PDE)



Problemstellung:
Wie entwickelt sich Temperaturverteilung $u(t,x)$ in Stab der Länge L, wenn am Anfang ($t=0$) die Temperatur gemäss $\phi(x)$ verteilt ist

→ (vereinfachte) Wärmeleitungsgleichung:

$$\frac{\partial}{\partial t} u(t,x) = \frac{\partial^2}{\partial x^2} u(t,x), \quad \text{kurz: } \dot{u} = u''$$

Diskretisierung:
Schrittweite in Zeit: $t = n\Delta t$
Schrittweite im Ort: $x = jh$, $j = 0, \dots, N$ mit $h = L/N$
→ manchmal zusätzlich noch Einführung von Geisterstellen x_{-1}, x_{N+1} notwendig

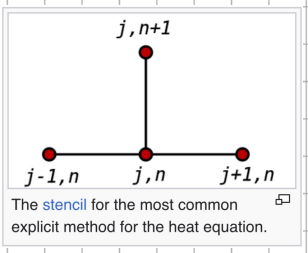
$u(t,x) = u(n\Delta t, jh) \approx u_{n,j}$ Temperatur bei Ort x_j zur Zeit t_n

Anfangsbedingungen: $\phi(x) \triangleq$ Temperaturverteilung des Stabes zur Zeit $t = 0$

Randbedingungen:
Dirichlet → Werte (durch Funktion) vorgegeben: $u(t,0) = l(t)$, $u(t,L) = r(t)$
Neumann → (örtliche) Ableitung vorgegeben $u'(t,0) = g_l(t)$, $u'(t,L) = g_r(t)$

isotherme Randbedingungen: Dirichlet mit konstanten Werten: $l(t)$, $r(t)$ const.
adiabatische Randbedingungen: Neumann mit $g_0(t) = g_l(t) = g_r(t) = 0$

FTCS (forward time, central space) explizites Verfahren



The stencil for the most common explicit method for the heat equation.

$$\dot{u} \approx \frac{u_{n+1,j} - u_{n,j}}{\Delta t}$$

$$u'' \approx \frac{u_{n,j+1} - 2u_{n,j} + u_{n,j-1}}{h^2}$$

$$\dot{u} = u'' \rightsquigarrow \boxed{\frac{u_{n+1,j} - u_{n,j}}{\Delta t} = \frac{u_{n,j+1} - 2u_{n,j} + u_{n,j-1}}{h^2}}$$ Differenzengleichung

nach $u_{n+1,j}$ aufgelöst:

$$\boxed{u_{n+1,j} = u_{n,j} + \frac{u_{n,j+1} - 2u_{n,j} + u_{n,j-1}}{h^2}\Delta t}$$ Temperatur bei Ort x_j zur Zeit t_{n+1}

→ für innere Punkte hat man also eine explizite Berechnungsvorschrift

äussere beide Punkte → durch Randbedingungen gegeben

Voraussetzung für Stabilität: $\boxed{\frac{\Delta t}{h^2} \leq \frac{1}{2}}$

→ bei hoher Ortsauflösung (h klein, h^2 sehr klein) ⇒ sehr viele Zeitschritte nötig
→ oft nicht brauchbar (ausser mit guter Grafikkarte)

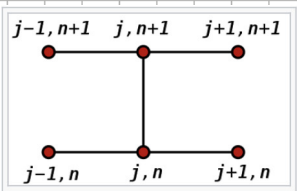
Alternative: Crank-Nicolson-Verfahren → keine Stabilitätsbedingung

Crank-Nicolson-Verfahren — implizites Verfahren

Mischung aus FTCS und BTCS (backward time, centered space)

analog zu Trapezmethode ⇒ unbedingt stabil

$$(\delta u)_{n,j} := \frac{u_{n,j+1} - 2u_{n,j} + u_{n,j-1}}{h^2}$$



Differenzenstern (engl. *stencil*) für das Crank-Nicolson-Verfahren beim 1D-Problem. *j* ist der Ort und *n* ist die Zeit.

$$\boxed{\frac{u_{n+1,j} - u_{n,j}}{\Delta t} = \frac{1}{2}(\delta u)_{n,j} + \frac{1}{2}(\delta u)_{n+1,j}}$$

halb FTCS, halb BTCS

alle Unbekannten (Index $n+1$ ≙ nächster Zeitpunkt) auf linke Seite bringen führt auf LGS für innere Punkte:

$$-\frac{\Delta t}{2h^2} u_{n+1,j-1} + \left(1 + \frac{\Delta t}{h^2}\right) u_{n+1,j} - \frac{\Delta t}{2h^2} u_{n+1,j+1} = u_{n,j} + \frac{\Delta t}{2}(\delta u)_{n,j}$$

$$\Leftrightarrow \begin{bmatrix} 1+\frac{\Delta t}{h^2} & -\frac{\Delta t}{2h^2} & & \\ -\frac{\Delta t}{2h^2} & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{\Delta t}{2h^2} \\ & & -\frac{\Delta t}{2h^2} & 1+\frac{\Delta t}{h^2} \end{bmatrix} \begin{bmatrix} u_{n+1,0} \\ u_{n+1,1} \\ \vdots \\ u_{n+1,N-1} \\ u_{n+1,N} \end{bmatrix} = \begin{bmatrix} u_{n,1} + \frac{\Delta t}{2}(\delta u)_{n,1} \\ \vdots \\ u_{n,N-1} + \frac{\Delta t}{2}(\delta u)_{n,N-1} \end{bmatrix}$$

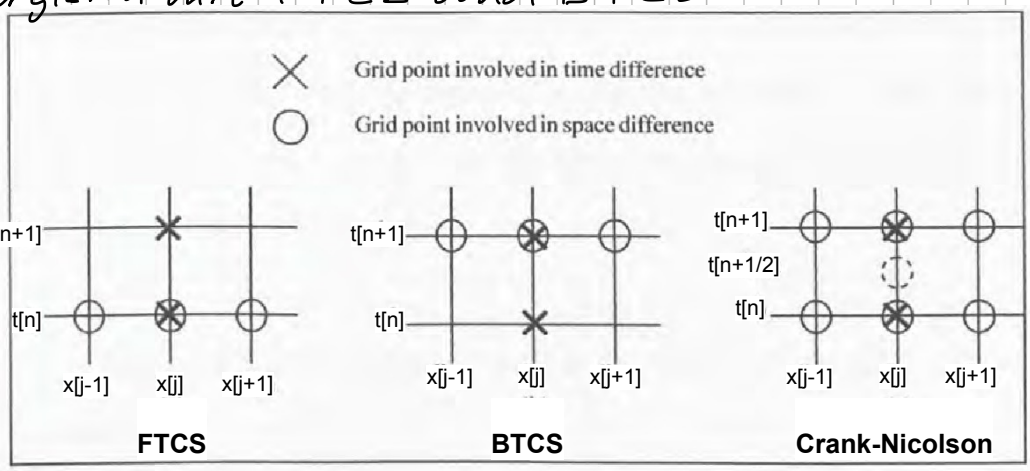
$N-1 \times N+1$ (Matrix) · $N+1$ (Vektor) · $N-1$ (rechte Seite)

$$\Leftrightarrow \begin{bmatrix} 1+\frac{\Delta t}{h^2} & -\frac{\Delta t}{2h^2} & & \\ -\frac{\Delta t}{2h^2} & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{\Delta t}{2h^2} \\ & & -\frac{\Delta t}{2h^2} & 1+\frac{\Delta t}{h^2} \end{bmatrix} \begin{bmatrix} u_{n+1,1} \\ \vdots \\ u_{n+1,N-1} \end{bmatrix} = \begin{bmatrix} u_{n,1} + \frac{\Delta t}{2}(\delta u)_{n,1} \\ \vdots \\ u_{n,N-1} + \frac{\Delta t}{2}(\delta u)_{n,N-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta t}{2h^2} u_{n+1,0} \\ \\ \frac{\Delta t}{2h^2} u_{n+1,N} \end{bmatrix}$$

$N-1 \times N-1$ · $N-1$ · $N-1$ · $N-1$

- $N-1 \times N-1$: konstante, symmetrisch positiv definite Matrix
- $N-1$: innere Punkte zu nächstem Zeitpunkt
- $N-1$: aus letztem Zeitpunkt
- $N-1$: durch Randwerte gegeben

Vergleich mit FTCS und BTCS:



Splines

gegeben: $n+1$ Stützstellen (x_i, y_i), $i = 0, \dots, n$

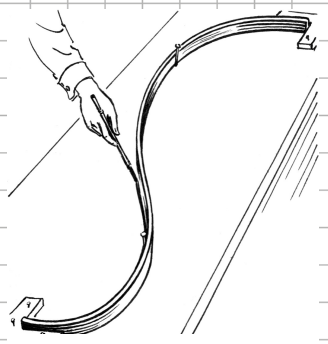
Spline k^{ten} Grades:

Funktion welche stückweise aus Polynomen k^{ten} Grades zusammengesetzt ist und durch vorgegebenen Stützstellen («Knoten») geht

$$S(x) = \begin{cases} S_1(x), & x \in [x_0, x_1] \\ S_2(x), & x \in [x_1, x_2] \\ \vdots \\ S_n(x), & x \in [x_{n-1}, x_n] \end{cases}$$

an Knoten müssen $k-1$ Ableitungen übereinstimmen;

- bei $k=1$ müssen die Knoten einfach (mit Massstab) verbunden werden
- bei $k=3$ (kubische Splines) müssen an den Knoten auch die Steigung und Krümmung übereinstimmen



Wortherkunft: Der Begriff stammt aus dem Schiffbau: eine lange dünne Latte (Straklatte, englisch spline), die an einzelnen Punkten durch Molche fixiert wird, biegt sich genau wie ein kubischer Spline. Dabei wird die Spannungsenergie minimal.

stückweise lineare Interpolation ($k=1$)

$$S_i(x) = \frac{x_i - x}{x_i - x_{i-1}} y_{i-1} + \frac{x - x_{i-1}}{x_i - x_{i-1}} y_i \quad \text{wobei } i = 1, 2, \dots, n$$

kubische Splines ($k=3$)

$$S_i(x) = \frac{1}{6}\frac{(x_i - x)^3}{h_i} M_{i-1} + \frac{1}{6}\frac{(x - x_{i-1})^3}{h_i} M_i + \underbrace{\left(\frac{y_i - y_{i-1}}{h_i} - \frac{h_i}{6}(M_i - M_{i-1})\right)}_{c_i}(x - x_{i-1}) + \underbrace{y_{i-1} - \frac{h_i^2}{6} M_{i-1}}_{d_i}$$

$$= \frac{1}{6}\frac{(x_i - x)^3}{h_i} M_{i-1} + \frac{1}{6}\frac{(x - x_{i-1})^3}{h_i} M_i + c_i (x - x_{i-1}) + d_i \quad \text{wobei } i = 1, 2, \dots, n$$

$$\text{mit } c_i = \frac{y_i - y_{i-1}}{h_i} - \frac{h_i}{6}(M_i - M_{i-1}), \quad d_i = y_{i-1} - \frac{h_i^2}{6} M_{i-1}, \quad h_i = x_i - x_{i-1}$$

für innere Momente M_i, $i \in \{1, 2, \dots, n-1\}$, müssen folgende $n-1$ Gleichungen erfüllt sein:

$$\frac{h_i}{6} M_{i-1} + \frac{h_i + h_{i+1}}{3} M_i + \frac{h_{i+1}}{6} M_{i+1} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}$$

aus Randbedingungen erhält man 2 weitere Gleichungen mit welchen man die äussersten beiden Momente M_0 und M_n eliminieren kann

man erhält dann ein LGS für die inneren Momente:

$$\underline{A}\vec{M} = \vec{b} \quad \text{mit } \vec{M} := (M_1, \dots, M_{n-1})^T \in \mathbb{R}^{n-1}, \quad \underline{A} \in \mathbb{R}^{(n-1)\times(n-1)}, \quad \vec{b} \in \mathbb{R}^{n-1}$$

Randbedingungen

Annahme:

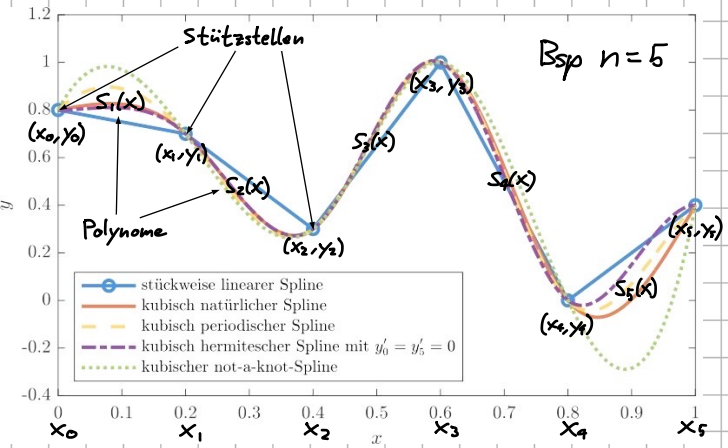
$a = x_0 < x_1 < \dots < x_{n-1} < x_n = b$

mit $x_i = a + ih$

→ alle x_i denselben Abstand h

$\underline{A}\vec{M} = \vec{b}$ für verschiedene Randbedingungen

↳ nächste Seite



natürlich («freier Rand»): $s''(a) = s''(b) = 0$

$$\begin{bmatrix} 4 & 1 & & & \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & 1 & \\ & & 1 & 4 & \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} y_0 - 2y_1 + y_2 \\ y_1 - 2y_2 + y_3 \\ \vdots \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix}$$

$$M_0 = M_n = 0$$

periodisch: $s'(a) = s'(b)$, $s''(a) = s''(b)$

$$\begin{bmatrix} 15 & 4 & & & -1 \\ 4 & \ddots & \ddots & & \\ & \ddots & \ddots & 4 & \\ & & 4 & 15 & \\ -1 & & & & \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} 4y_0 - 8y_1 + 4y_2 \\ 4y_1 - 8y_2 + 4y_3 \\ \vdots \\ 4y_{n-3} - 8y_{n-2} + 4y_{n-1} \\ 4y_{n-2} - 8y_{n-1} + 4y_n \end{bmatrix} + \frac{6}{h^2} \begin{bmatrix} y_n - y_{n-1} - y_1 + y_0 \\ 0 \\ \vdots \\ 0 \\ y_n - y_{n-1} - y_1 + y_0 \end{bmatrix}$$

$$M_0 = M_n = \frac{3}{2h^2}\left(y_1 - y_0 - y_n + y_{n-1}\right) - \frac{1}{4}\left(M_1 + M_{n-1}\right)$$

hermitesch («eingespannter Rand»): $s'(a) = y_0'$, $s'(b) = y_n'$

$$\begin{bmatrix} 7 & 2 & & & \\ 2 & \ddots & \ddots & & \\ & \ddots & \ddots & 2 & \\ & & 2 & 7 & \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} 3y_0 - 5y_1 + 2y_2 \\ 2y_1 - 4y_2 + 2y_3 \\ \vdots \\ 2y_{n-3} - 4y_{n-2} + 2y_{n-1} \\ 2y_{n-2} - 5y_{n-1} + 3y_n \end{bmatrix} + \frac{6}{h} \begin{bmatrix} y_0' \\ 0 \\ \vdots \\ 0 \\ -y_n' \end{bmatrix}$$

$$M_0 = -\frac{1}{2}M_1 + \frac{3}{h}\left(\frac{y_1 - y_0}{h} - y_0'\right), \quad M_n = -\frac{1}{2}M_{n-1} + \frac{3}{h}\left(y_n' - \frac{y_n - y_{n-1}}{h}\right)$$

not-a-knot: $s_1'''(x_1) = s_2'''(x_1)$, $s_{n-1}'''(x_{n-1}) = s_n'''(x_{n-1})$

$$\begin{bmatrix} 6 & 0 & & & \\ 1 & \ddots & \ddots & & \\ & \ddots & \ddots & 1 & \\ & & 0 & 6 & \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \frac{6}{h^2} \begin{bmatrix} y_0 - 2y_1 + y_2 \\ y_1 - 2y_2 + y_3 \\ \vdots \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{bmatrix}$$

$$M_0 = 2M_1 - M_2, \quad M_n = 2M_{n-1} - M_{n-2}$$