Mathematik 3 (Numerik)

Zusammenfassung

Fabian Damken

8. November 2023

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhaltsverzeichnis

1 Grundlagen

1.1 Vektornorm

Eine *Vektornorm* ist eine Abbildung $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}^+$ mit folgenden Eigenschaften:

Definitheit $\|c\| = 0 \iff x = 0$

Homogenität $\|\alpha x\| = |\alpha| \cdot \|x\|$ (für alle $\alpha \in \mathbb{R}$ und alle $x \in \mathbb{R}^n$)

Dreiecksungleichung $\|x + y\| \leq \|x\| + \|y\|$ (für alle $x, y \in \mathbb{R}^n$)

Wichtige Normen

p-Norm $\|x\|_p = \sqrt[p]{\sum_{i=1}^n |x_i|^p}$

Summennorm (1-Norm) $\|x\|_1 = \sum_{i=1}^n |x_i|$

Euklidische Norm (2-Norm) $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2} = \sqrt{x^T x}$

Maximumsnorm (∞-Norm) $\|x\|_\infty = \max_{i=1,\cdots,n} |x_i|$

1.2 Induzierte Matrixnormen

Sei $\|\cdot\|$ eine beliebige Norm auf $\mathbb{R}^n$. Dann ist auf $\mathbb{R}^{n\times n}$ eine dazugehörige Matrixnorm definiert durch:

$$\|A\| := \sup_{\|x\|=1} \|Ax\| = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}$$

für jede Matrix $A \in \mathbb{R}^{n\times n}$. Diese Norm wird *induzierte Matrixnorm* genannt.

Eine solche Norm hat folgende Eigenschaften:

Definitheit $\|A\| = 0 \iff A = 0$

Homogenität $\|\alpha A\| = |\alpha| \cdot \|A\|$ (für alle $\alpha \in \mathbb{R}$ und alle $A \in \mathbb{R}^{n\times n}$)

Dreiecksungleichung $\|A + B\| \leq \|A\| + \|B\|$ (für alle $A, B \in \mathbb{R}^{n\times n}$)

Verträglichkeit $\|Ax\| \leq \|A\| \cdot \|x\|$ (für alle $x \in \mathbb{R}^n$ und alle $A \in \mathbb{R}^{n\times n}$)

Submultiplikativität $\|AB\| \leq \|A\| \cdot \|B\|$ (für alle $A, B \in \mathbb{R}^{n\times n}$)

Wichtige Normen

Spaltensummennorm (1-Norm) $\|A\|_1 = \max_{j=1,\cdots,n} \sum_{i=1}^{n} |a_{ij}|$

Euklidische Norm (2-Norm) $\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$ (mit λ Eigenwert von A)

Zeilensummennorm (∞-Norm) $\|A\|_\infty = \max_{i=1,\cdots,n} \sum_{j=1}^{n} |a_{ij}|$

1.3 Konditionszahl

Sei $A \in \mathbb{R}^{n\times n}$ invertierbar und sei $\|\cdot\|$ eine induzierte Matrixnorm.
Dann ist

$$\mathrm{cond}(A) = \|A\| \cdot \left\|A^{-1}\right\|$$

die *Konditionszahl* von A bzgl. der Matrixnorm $\|\cdot\|$.

1.4 Spezielle Matrizen

- $A \in \mathbb{C}^{n\times n}$ ist *hermitesch* gdw. $A^H = A$, wobei $A^H := \overline{A}^T$.
- $A \in \mathbb{C}^{n\times n}$ ist *unität* gdw. $A^H = A^{-1}$.
- $A \in \mathbb{R}^{n\times n}$ ist *orthogonal* gdw. $A^T = A^{-1}$, bzw. $A^T A = AA^T = I$

1.5 Eigenwerte und Eigenvektoren

Eine Zahl $\lambda \in \mathbb{C}$ heißt *Eigenwert* einer Matrix $A \in \mathbb{C}^{n\times n}$ gdw. es einen Vektor $x \in \mathbb{C}^n, x \neq o$ gibt mit:

$$Ax = \lambda x$$

Ein solcher Vektor wird *Eigenvektor* genannt. Die Menge aller Eigenwerte $\sigma(A)$ heißt *Spektrum* von A.
Der Unterraum

$$\mathrm{Eig}_A(\lambda) := \{x \in C^N \mid (A - \lambda I)x = 0\}$$

wird *Eigenraum* von A zum Eigenwert λ genannt. Die Dimension

$$\gamma(\lambda) := \dim(\mathrm{Eig}_A(\lambda)) = n - \mathrm{rank}(A - \lambda I)$$

ist die *geometrische Vielfachheit* von λ und gibt die maximale Anzahl linear unabhängiger Eigenvektoren zu λ an.

1.5.1 Charakteristisches Polynom

λ ist ein Eigenwert von $A \in \mathbb{C}^{n\times n}$ gdw. gilt

$$\mathcal{X}(\lambda) \coloneqq \det(A - \lambda I) = 0$$

also wenn λ eine Nullstelle des *charakteristischen Polynoms* $\mathcal{X}$ von A ist.

Das charakteristische Polynom hat die Linearfaktorzerlegung

$$\mathcal{X}(\mu) = (-1)^n \cdot (\mu - \lambda_1)^{v_1} \cdot \dots \cdot (\mu - \lambda_k)^{v_k}$$

wobei $v(\lambda_i) = v_i \in \mathbb{N}$ die *algebraische Vielfachheit* von λ_i genannt wird.

1.5.2 Eigenschaften

Sei $A \in \mathbb{C}^{n\times n}$ eine beliebige Matrix, dann gilt:

- $\lambda \in \sigma(A) \implies \lambda \in \sigma(A^T) \wedge \overline{\lambda} \in \sigma(A^H)$
- Für jede reguläre Matrix A hat die zu A ähnliche Matrix $B = T^{-1}AT$ das selbe charakteristische Polynom und die selben Eigenwerte wie A. Ist x ein Eigenvektor von A, dann ist $y = T^{-1}x$ ein Eigenvektor von B.
- Ist A hermitesch, dann hat A nur reelle Eigenwerte.
  Ist A unitär, dann gilt $|\lambda| = 1$ für jeden Eigenwert λ.

1.5.3 Diagonalisierbarkeit

Eine Matrix $A \in \mathbb{C}^{n\times n}$ heißt *diagonalisierbar* gdw. sie n linear unabhängige Eigenvektoren besitzt.

1.6 $\mathcal{O}$-Notation

Für eine Funktion $g : \mathbb{N} \to \mathbb{R}$ bezeichnet $\mathcal{O}(g(n))$ die Menge aller Funktionen, die asymptotische nicht schneller wachsen als g, d.h.:

$$\mathcal{O}(g(n)) \coloneqq \{f : \mathbb{N} \to R \,|\, \exists n_0 \in \mathbb{N} : \exists c \in \mathbb{R} : \forall n \in N, n \geq n_0 : f(n) \leq c \cdot g(n)\}$$

2 Interpolation

Gegeben	Funktionaler Zusammenhang $y = f(x), \quad f : [a, b] \to \mathbb{R}, a < b \in \mathbb{R}$
Bekannt	Werte $y_i = f(x_i), \quad i = 0, \cdots, n$ (*Stützstellen*)
Ziel	Annäherung für $f(x)$ für beliebige $x \in [a, b]$
Interpolationsproblem	Suche einfache Ersatzfunktion $\Phi(x)$ mit $\Phi(x_i) = y_i, \quad i = 0, \cdots, n$
Wunsch	Der Fehler $\lvert f(x) - \Phi(x)\rvert$ sollte auf $[a, b]$ möglichst gering sein.

Interpolationsaufgabe Zu einer gegebenen Ansatzfunktion $\Phi(x; a_0, \cdots, a_n), \quad x \in \mathbb{R}$ mit Parametern $a_0, \cdots, a_n \in \mathbb{R}$ sollen zu gegebenen Paaren $(x_i, y_i) \quad i = 0, \cdots, n$ mit $x_i, y_i \in \mathbb{R}$ und $x_i \neq x_j$ für $i \neq j$ sollen die Parameter $a_0, \cdots, a_n$ so bestimmt werden, dass die Interpolationsbedingungen $\Phi(x_i; a_0, \cdots, a_n) = y_i, \quad i = 0, \cdots, n$ erfüllt sind. Die Paare (x_i, y_i) werden als *Stützpunkte* bezeichnet.

2.1 Polynominterpolation

Ansatzfunktion: Polynome vom Grad $\leq n$, also

$$p_n(x) = \Phi(x, a_0, \cdots, a_n) = a_0 + a_1 x + \cdots + a_n x^n$$

Interpolationsaufgabe: Finde ein Polynom $p_n(x)$ vom Grad $\leq n$, sodass die Interpolationsbedingungen $p_n(x_i) = y_i, \quad i = 0, \cdots, n$ erfüllt sind.

Anwendungen hierfür sind z.B.:

- Approximation einer Funktion auf einem Intervall
- Inverse Interpolation (Approximation von f^{-1} bei einer gegebenen Funktion f)
- Numerische Integration (siehe Kapitel 3)
- Numerische Differentiation

2.1.1 Eindeutigkeit

Es existiert genau ein Polynom vom Grad $\leq n$, welches die Interpolationsbedingungen erfüllt, und zwar $p_n(x)$. Die Lösung einer Interpolationsaufgabe ist also eindeutig.

2.1.2 Naiver Lösungsansatz

Die Interpolationsbedingungen liefern $n+1$ Gleichungen, womit sich mit Koeffizienten $a_0, \cdots, a_n$ ein lineares Gleichungssystem aufstellen lässt:

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}$$

Nachteile

- Hoher Rechenaufwand: Das Auflösen des Gleichungssystem benötigt $\mathcal{O}(n^3)$ Rechenoperationen.
- Die Koeffizientenmatrix (*Vandermonde-Matrix*) ist invertierbar, aber extrem schlecht konditioniert $\rightarrow$ Rundungsfehler werden dramatisch verstärkt.

2.1.3 Lagrange-Interpolation

$$p_n(x) = \sum_{k=0}^{n} y_k L_{k,n}(x) \qquad L_{k,n}(x) = \prod_{j=0,\, j\neq k}^{n} \frac{x - x_j}{x_k - x_j}$$

Die Lagrange-Polynome $L_{k,n}$ sind so gewählt, dass:

$$L_{k,n}(x_i) = \begin{cases} 1 & \text{falls } k = i \\ 0 & \text{sonst} \end{cases} =: \delta_{ki}$$

wobei δ_{ki} das Kronecker-Symbol ist.

Bewertung

- Vorteile
  - Rechenaufwand: $\mathcal{O}(n^2)$ zur Koeffizientenberechnung, $\mathcal{O}(n)$ zur Auswertung von $p_n(x)$
  - Intuitive Darstellung
- Nachteile
  - Hinzunahme von Stützstellen ist aufwendig.

2.1.4 Newtonsche Interpolationsformel

$$p_n(x) = y_0 + \sum_{i=1}^{n} \gamma_i (x - x_0) \cdot \cdots \cdot (x - x_{i-1}) \qquad \gamma_i = f_{[x_0, \cdots, x_i]}$$

Die Berechnung der Parameter erfolgt über die *dividierten Differenzen* $f_{[x_0,\cdots,x_i]} := \gamma_i$ zu den Stützstellen $x_0, \cdots, x_i$, wobei $f_{[x_0]} = \gamma_0 = y_0$. Allgemein werden die dividierten Differenzen über Rekursion berechnet (die Reihenfolge der x_i ist dabei irrelevant):

$$j = 0, \cdots, n: \quad f_{[x_j]} = y_j$$

$$k = 1, \cdots, n,\ j = 0, \cdots, n-k: \quad f_{[x_j,\cdots,x_{j+k}]} = \frac{f_{[x_{j+1},\cdots,x_{j+k}]} - f_{[x_j,\cdots,x_{j+k-1}]}}{x_{j+k} - x_j}$$

Die Berechnung der dividierten Differenzen kann z.B. mit folgendem Schema erfolgen:

$$\begin{array}{c|cccc} x_0 & f_{[x_0]} = y_0 & \searrow & & \\ & & f_{[x_0,x_1]} & \searrow & \\ x_1 & f_{[x_1]} = y_1 & \nearrow \; \searrow & & f_{[x_0,x_1,x_2]} \\ & & f_{[x_1,x_2]} & \nearrow \; \searrow & \\ x_2 & f_{[x_2]} = y_2 & \nearrow \; \searrow & & \vdots \\ \vdots & \vdots & & \vdots & \end{array}$$

Vorteile

- Rechenaufwand: $\mathcal{O}(n^2)$ zur Berechnung der dividierten Differenzen, $\mathcal{O}(n)$ zur Auswertung von $p_n(x)$
- Hinzunahme neuer Stützstellen erfordert nur die Berechnung von n zusätzlichen dividierten Differenzen.

2.1.5 Fehlerabschätzungen

Sei $f \in C^{n+1}([a,b])$ und $x_0, \cdots, x_n \in [a,b]$ verschiedene Punkte und sei $p_n(x)$ das eindeutige Interpolationspolynom vom Grad $\leq n$ zu den Stützwerten $(x_i, f(x_i)), \quad i = 0, \cdots, n$. Dann existiert zu jedem $x \in [a,b]$ ein $\xi_x \in [a,b]$ mit

$$f(x) - p_n(x) = \frac{f^{(n+1)(\xi_x)}}{(n+1)!}(x - x_0) \cdot \cdots \cdot (x - x_n)$$

Mit dem Knotenpolynom

$$\omega(x) = \prod_{i=0}^{n}(x - x_i)$$

gilt für den maximalen Fehler:

$$\max_{x\in[a,b]} |f(x) - p_n(x)| \leq \max_{x\in[a,b]} \frac{\left|f^{(n+1)}(x)\right|}{(n+1)!} \max_{x\in[a,b]} |\omega(x)| \leq \max_{x\in[a,b]} \frac{\left|f^{(n+1)}(x)\right|}{(n+1)!}(b-a)^{n+1}$$

Werden keine äquidistanten Stützstellen verwendet sondern Tschebyscheff-Abszissen, so verschärft sich die Fehlerabschätzung zu:

$$\max_{x\in[a,b]} |f(x) - p_n(x)| \leq \max_{x\in[a,b]} \frac{\left|f^{(n+1)}(x)\right|}{(n+1)!} \left(\frac{b-a}{2}\right)^{n+1} 2^{-n}$$

2.1.6 Runges Phänomen

Bei äquidistanter Wahl der Stützpunkte, d.h. $x_i = a + ih, \quad h = \frac{b-a}{n}$, ist i.A. nicht gewährleistet, dass gilt:

$$\lim_{n\to\infty} f(x) - p_n(x) = 0 \qquad \text{für alle } x \in [a,b]$$

Runges Phänomen beschreibt das Überschwingen des Interpolanten am Rand des Intervalls.

2.1.7 Tschebyscheff-Abszissen

Tschebyscheff-Abszissen dienen der Vermeidung von Runges Phänomen, in dem die Stützstellen nicht äquidistant gewählt werden:

$$x_i = \frac{b-a}{2} \cos\left(\frac{2i+1}{n+1} \cdot \frac{\pi}{2}\right) + \frac{b+a}{2} \qquad i = 0, \cdots, n$$

Dies liefert den minimalen Wert für $\max_{x\in[a,b]} |\omega(x)|$, und zwar:

$$\max_{x\in[a,b]} |\omega(x)| = \left(\frac{b-1}{2}\right)^{n+1} 2^{-n}$$

2.2 Spline-Interpolation

- Motivation: Höhere Anzahl an Stützstellen ergibt nicht immer eine bessere Approximation bei der Polynominterpolation.
- Lösung: Zerlegung von $[a, b]$ in Teilintervalle und Polynominterpolation auf den Teilintervallen mit Grad $\leq k$.
- Problem: Die Polynome passen an den Intervallgrenzen mglw. nicht zusammen.

  $\rightarrow$ Spline-Interpolation: Die Polynome gehen $k-1$-mal stetig ineinander über.

Splinefunktion Sei $\Delta = \{x_i \,|\, a = x_0 < x_1 < \cdots < x_n = b\}$ eine Zerlegung des Intervalls $[a, b]$, wobei die x_i *Knoten* genannt werden.

Dann ist eine *Splinefunktion* der Ordnung k zur Zerlegung Δ eine Funktion $s : [a, b] \rightarrow \mathbb{R}$ mit folgenden Eigenschaften:

- Es gilt $s \in C^{k-1}([a, b])$
- s stimmt auf jedem Intervall $[x_i, x_{i+1}]$ mit einem Polynom s_i vom Grad $\leq k$ überein.

Die Menge dieser Splinefunktionen wird mit $S_{\Delta,k}$ bezeichnet.

Splinefunktionen mit $k = 1$ werden *Lineare Splines* genannt, Splinefunktionen mit $k = 3$ werden *Kubische Splines* genannt.

Interpolationsaufgabe Bestimme zu einer Zerlegung $\Delta = \{x_i \,|\, a = x_0 < x_1 < \cdots < x_n = b\}$ und Werten $y_i \in \mathbb{R}, \quad i = 0, \cdots, n$ eine Funktion $s \in S_{\Delta,k}$ mit $s(x_i) = y_i, \quad i = 0, \cdots, n$.

2.2.1 Lineare Splines

- Ein linearer Spline $s \in S_{\Delta,1}$ ist stetig.
- s ist ein Polynom vom Grad ≤ 1 auf jedem Intervall $[x_i, x_{i+1}]$.
- Die Interpolationsbedingungen ergeben $s_i(x_i) = y_i$ und $s_i(x_{i+1}) = y_{i+1}$.
- Dies legt s_i fest (Lagrange-Interpolation):

$$s(x) = s_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} y_i + \frac{x - x_i}{x_{i+1} - x_i} y_{i+1} \qquad \text{für alle } x \in [x_i, x_{i+1}]$$

Eindeutigkeit

Zu einer Zerlegung Δ von $[a,b]$ und Werten $y_i, \quad i = 0, \cdots, n$ existiert genau ein interpolierender linearer Spline.

Fehlerabschätzung

Sei $f \in C^2([a,b])$. Dann gilt für jede Zerlegung $\Delta = \{x_i \,|\, a = x_0 < x_1 \cdots x_n = b\}$ von $[a,b]$ und den interpolierenden Spline $s \in S_{\Delta,1}$ von f:

$$\max_{x\in[a,b]} |f(x) - s(x)| \leq \frac{1}{8} \max_{x\in[a,b]} \left|f''(x)\right| h_{\text{max}}^2 \qquad \text{mit } h_{\text{max}} \coloneqq \max_{i=0,\cdots,n-1} x_{i+1} - x_i$$

2.2.2 Kubische Splines

- Ein kubischer Spline $s \in S_{\Delta,3}$ ist zweimal stetig differenzierbar.
- s'' ist stetig und stückweise linear, d.h. $s'' \in S_{\Delta,1}$.
- s wird durch Integration von s'' bestimmt.

Mit den *Momenten* $M_i = s_i''(x_i)$ ergibt sich der Ansatz:

$$s_i(x) = \frac{1}{6}\left(\frac{(x_{i+1}-x)^3}{x_{i+1}-x_i} M_i + \frac{(x-x_i)^3}{x_{i+1}-x_i} M_{i+1}\right) + c_i(x-x_i) + d_i$$

mit zwei Konstanten $c_i, d_i \in \mathbb{R}$:

$$\begin{aligned} c_i &= \frac{y_{i+1}-y_i}{h_i} - \frac{h_i}{6}(M_{i+1} - M_i) \\ d_i &= y_i - \frac{h_i^2}{6} M_i \end{aligned}$$

Zu Berechnung der Momente $M_i, \quad i = 0, \cdots, n$ müssen zusätzliche Randbedingungen verwendet werden, die alle eindeutige Lösungen für $M_0, \cdots, M_n$ liefern:

Natürliche Randbedingungen $s''(a) = s''(b) = 0$, d.h. $M_0 = M_n = 0$

Hermite-Randbedingungen $s'(a) = f'(a)$ und $s'(b) = f'(b)$

Diese Randbedingungen ergeben mit obigem Ansatz ein lineares, diagonaldominantes, tridiagonales Gleichungssystem:

$$\begin{bmatrix} \mu_0 & \lambda_0 & & & & & \\ \frac{h_0}{6} & \frac{h_0+h_1}{3} & \frac{h_1}{6} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \frac{h_{i-1}}{6} & \frac{h_{i-1}+h_i}{3} & \frac{h_i}{6} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} & \frac{h_{n-1}}{6} \\ & & & & & \lambda_n & \mu_n \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} b_0 \\ \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \vdots \\ \frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}} \\ \vdots \\ \frac{y_n-y_{n-1}}{h_{n-1}} - \frac{y_{n-1}-y_{n-2}}{h_{n-2}} \\ b_n \end{bmatrix}$$

Mit folgenden Werten für $\mu_0, \lambda_0, \lambda_n, \mu_n, b_0, b_n$:

- Natürliche Randbedingungen:

$$\begin{aligned}\lambda_0 &= \lambda_n = b_0 = b_n = 0\\ \mu_0 &= \mu_n = 1\end{aligned}$$

- Hermite Randbedingungen:

$$\begin{aligned}\mu_0 &= \frac{h_0}{3}\\ \mu_n &= \frac{h_{n-1}}{3}\\ \lambda_0 &= \frac{h_0}{6}\\ \lambda_n &= \frac{h_{n-1}}{6}\\ b_0 &= \frac{y_1 - y_0}{h_0} - f'(a)\\ b_n &= f'(b) - \frac{y_n - y_{n-1}}{h_{n-1}}\end{aligned}$$

Natürliche Randbedingungen

$$\begin{bmatrix} 1 & 0 & & & & & \\ \frac{h_0}{6} & \frac{h_0+h_1}{3} & \frac{h_1}{6} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \frac{h_{i-1}}{6} & \frac{h_{i-1}+h_i}{3} & \frac{h_i}{6} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} & \frac{h_{n-1}}{6} \\ & & & & & 0 & 1 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \vdots \\ \frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}} \\ \vdots \\ \frac{y_n-y_{n-1}}{h_{n-1}} - \frac{y_{n-1}-y_{n-2}}{h_{n-2}} \\ 0 \end{bmatrix}$$

Hermite Randbedingungen

$$\begin{bmatrix} \frac{h_0}{3} & \frac{h_0}{6} & & & & & \\ \frac{h_0}{6} & \frac{h_0+h_1}{3} & \frac{h_1}{6} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & \frac{h_{i-1}}{6} & \frac{h_{i-1}+h_i}{3} & \frac{h_i}{6} & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & \frac{h_{n-2}}{6} & \frac{h_{n-2}+h_{n-1}}{3} & \frac{h_{n-1}}{6} \\ & & & & & \frac{h_{n-1}}{6} & \frac{h_{n-1}}{3} \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_n \end{bmatrix} = \begin{bmatrix} \frac{y_1-y_0}{h_0} - f'(a) \\ \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \vdots \\ \frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}} \\ \vdots \\ \frac{y_n-y_{n-1}}{h_{n-1}} - \frac{y_{n-1}-y_{n-2}}{h_{n-2}} \\ f'(b) - \frac{y_n-y_{n-1}}{h_{n-1}} \end{bmatrix}$$

Fehlerabschätzung

Seien $h_{\text{min}} := \min_{i=0,\cdots,n-1} h_i$ und $h_{\text{max}} := \max_{i=0,\cdots,n-1} h_i$.

Dann gilt für $f \in C^4([a,b])$ mit $f''(a) = f''(b) = 0$ und jede Unterteilung Δ, $y_i = f(x_i)$ und dem kubischen Spline-Interpolanten $s \in S_{\Delta,3}$ zu den **natürlichen Randbedingungen** und $k = 1, 2$:

$$|f(x) - s(x)| \leq \frac{h_{\max}}{h_{\min}} \sup_{\xi \in [a,b]} \left| f^{(4)}(\xi) \right| h_{\max}^4$$
$$\left| f^{(k)}(x) - s^{(k)}(x) \right| \leq \frac{2h_{\max}}{h_{\min}} \sup_{\xi \in [a,b]} \left| f^{(4)}(\xi) \right| h_{\max}^{4-k}$$

Für die **hermite Randbedingungen** gelten schärfere Fehlerabschätzungen (sei alles definiert wie oben):

$$|f(x) - s(x)| \leq \frac{5}{384} \sup_{\xi \in [a,b]} \left| f^{(4)}(\xi) \right| h_{\max}^4$$
$$\left| f^{(k)}(x) - s^{(k)}(x) \right| \leq \frac{2h_{\max}}{h_{\min}} \sup_{\xi \in [a,b]} \left| f^{(4)}(\xi) \right| h_{\max}^{4-k}$$

3 Integration

Gegeben Ein funktionaler Zusammenhang $f : [a, b] \to \mathbb{R}$.

Ziel Näherungsweise Bestimmung des Integrals $\int_a^b f(x)\,dx$.

Integrationsaufgabe: Zu einem gegebenen, integrierbarem, $f : [a, b] \to \mathbb{R}$, berechne $I(f) = \int_a^b f(x)\,dx$.

Exakte Integrationsformel Eine Integrationsformel $J(f) = \sum_{i=0}^{n} \beta_i f(x_i)$ heißt *exakt vom Grad* n gdw. sie alle Polynome bis mindestens Grad n exakt integriert.

3.1 Geschlossene Newton-Cotes-Quadratur

$$I_n(f) = h\sum_{k=0}^{n} \alpha_{k,n} f(x_k) \qquad \alpha_{k,n} = \int_0^n \prod_{j=0,\, j\neq k}^{n} \frac{s-j}{k-j}\,ds \qquad h = \frac{b-a}{n} \qquad x_k = a + kh$$

- Die Werte $\alpha_{0,n}, \cdots, \alpha_{n,n}$ heißen *Gewichte*.
- Diese sind unabhängig von f und $[a, b]$ und somit tabellierbar.
- Es gilt immer:
$$h\sum_{k=0}^{n} \alpha_{k,n} = b - a, \quad \text{also } \sum_{k=0}^{n} \alpha_{k,n} = n$$
- Die geschlossene Newton-Cotes-Formel $I(f)$ ist exakt vom Grad n.

Fehlerabschätzung Es gilt für den Fehler $E_n(f) \coloneqq I(f) - I_n(f)$:

$$\left| \int_a^b f(x)\,dx - \int_a^b p_n(x)\,dx \right| \leq \int_a^b |f(x) - p_n(x)|\,dx \leq \max_{\xi\in[a,b]} \frac{\left|f^{(n+1)}(\xi)\right|}{(n+1)!}(b-a)^{n+2}$$

3.1.1 Spezielle geschlossene Newton-Cotes-Formeln

n	h	$\alpha_{k,n}$					**max. Fehler** $E_n(f)$	**Name**
1	$b-a$	$\frac{1}{2}$	$\frac{1}{2}$				$\max_{\xi\in[a,b]} \frac{\left\|f^{(2)}(\xi)\right\|}{12} h^3$	Trapezregel
2	$\frac{b-a}{2}$	$\frac{1}{3}$	$\frac{4}{3}$	$\frac{1}{3}$			$\max_{\xi\in[a,b]} \frac{\left\|f^{(4)}(\xi)\right\|}{90} h^5$	Simpson-Regel
3	$\frac{b-a}{3}$	$\frac{3}{8}$	$\frac{9}{8}$	$\frac{9}{8}$	$\frac{3}{8}$		$\max_{\xi\in[a,b]} \frac{3\left\|f^{(4)}(\xi)\right\|}{80} h^5$	3/8-Regel
4	$\frac{b-a}{4}$	$\frac{14}{45}$	$\frac{64}{45}$	$\frac{24}{45}$	$\frac{64}{45}$	$\frac{14}{45}$	$\max_{\xi\in[a,b]} \frac{8\left\|f^{(6)}(\xi)\right\|}{945} h^7$	Milne-Regel

Tabelle 3.1: Geschlossene Newton-Cotes-Formeln

Ab $n \geq 7$ treten negative Gewichte auf, weshalb das Verfahren zunehmend numerisch instabil wird.

3.2 Offene Newton-Cotes-Quadratur

$$\tilde{I}_n(f) = h \sum_{k=1}^{n+1} \tilde{\alpha}_{k,n} f(x_k) \qquad \tilde{\alpha}_{k,n} = \int_0^{n+2} \prod_{j=1,\, j\neq k}^{n+1} \frac{s-j}{k-j}\, ds \qquad h = \frac{b-a}{n+2} \qquad x_k = a + kh$$

3.2.1 Spezielle offene Newton-Cotes-Formeln

n	h	$\tilde{\alpha}_{k,n}$			**max. Fehler** $\tilde{E}_n(f)$	**Name**
0	$\frac{b-a}{2}$	2			$\max_{\xi\in[a,b]} \frac{\lvert f^{(2)}(\xi)\rvert}{3} h^3$	Rechteck-Regel
1	$\frac{b-a}{3}$	$\frac{3}{2}$	$\frac{3}{2}$		$\max_{\xi\in[a,b]} \frac{3\lvert f^{(2)}\rvert(\xi)}{4} h^3$	
2	$\frac{b-a}{4}$	$\frac{8}{3}$	$-\frac{4}{3}$	$\frac{8}{3}$	$\max_{\xi\in[a,b]} \frac{28\lvert f^{(4)}(\xi)\rvert}{90} h^5$	

Tabelle 3.2: Offene Newton-Cotes-Formeln

- Vorteil offener Formeln: kleineres h bei gleichem n.
- Die Fehlerordnung von $n = 1$ ist wie bei $n = 0$, also kein zusätzlicher Nutzen.
- Ab $n \geq 2$ können negative Gewichte auftreten $\implies$ Numerisch instabil.
- Somit ist nur die Rechteck-Regel empfehlenswert.

3.3 Vergleich Geschlossene vs. Offene Newton-Cotes-Quadratur

	geschlossen			**offen**		
n	h	$\tilde{E}_n(f)$	**Name**	h	$E_n(f)$	**Name**
0				$\frac{b-a}{2}$	$\max_{\xi\in[a,b]} \frac{\lvert f^{(2)}(\xi)\rvert}{3} h^3$	Rechteck-Regel
1	$b-a$	$\max_{\xi\in[a,b]} \frac{\lvert f^{(2)}(\xi)\rvert}{12} h^3$	Trapezregel	$\frac{b-a}{3}$	$\max_{\xi\in[a,b]} \frac{3\lvert f^{(2)}(\xi)\rvert}{4} h^3$	
2	$\frac{b-a}{2}$	$\max_{\xi\in[a,b]} \frac{\lvert f^{(4)}(\xi)\rvert}{90} h^5$	Simpson-Regel	$\frac{b-a}{4}$	$\max_{\xi\in[a,b]} \frac{28\lvert f^{(4)}(\xi)\rvert}{90} h^5$	
3	$\frac{b-a}{3}$	$\max_{\xi\in[a,b]} \frac{3\lvert f^{(4)}(\xi)\rvert}{80} h^5$	3/8-Regel			
4	$\frac{b-a}{4}$	$\max_{\xi\in[a,b]} \frac{8\lvert f^{(6)}(\xi)\rvert}{945} h^7$	Milne-Regel			

Tabelle 3.3: Vergleich: Offene/Geschlossene Newton-Cotes-Quadratur

3.4 Summierte Newton-Cotes-Formeln

- Problematik: Die Newton-Cotes-Formeln liefern nur genaue Ergebnisse, wenn das Integrationsintervall klein und die Anzahl der Knoten nicht zu groß ist.

- Idee: Zerlege $[a, b]$ im m Teilintervalle der Länge $H = \frac{b-a}{m}$:

$$y_j = a + jH, \quad j = 0, \cdots, m$$

- Es gilt:

$$I(f) = \sum_{j=0}^{m-1} \int_{y_j}^{y_{j+1}} f(x)\, dx$$

- Wende nun die Newton-Cotes-Formel vom Grad n einzeln auf die Teilintervalle an und summiere das Ergebnis.

$$S_N^{(n)}(f) = h \sum_{j=0}^{m-1} \sum_{i=0}^{n} \alpha_{i,n} f(x_{jn+i}) \qquad x_k = a + kh \qquad h = \frac{b-a}{N} \qquad N = nm$$

Die Gewichte $\alpha_{i,n}$ sind die Gewichte der geschlossenen Newton-Cotes-Formel.
Der Quadraturfehler

$$R_N^{(n)}(f) = I(f) - S_N^{(n)}(f)$$

ergibt sich durch Summierung der Fehler auf den Teilintervallen.

3.4.1 Summierte Trapezregel (geschlossen)

$$S_N^{(1)}(f) = \frac{h}{2} \sum_{j=0}^{m-1} \big(f(x_j) + f(x_{j+1})\big) \qquad x_j = a + jh \qquad h = \frac{b-a}{m}$$

Fehler: $R_N^{(1)}(f) \leq \max_{\xi \in [a,b]} \frac{|f''(\xi)|}{12}(b-a)h^2$

3.4.2 Summierte Simpson-Regel (geschlossen)

$$S_N^{(2)}(f) = \frac{h}{3} \sum_{j=0}^{m-1} \big(f(x_{2j}) + 4f(x_{2j+1}) + f(x_{2j+2})\big) \qquad x_j = a + jh \qquad h = \frac{b-a}{2m}$$

Fehler: $R_N^{(2)}(f) \leq \max_{\xi \in [a,b]} \frac{|f^{(4)}(\xi)|}{180}(b-a)h^4$

3.4.3 Summierte Rechteck-Regel (offen)

$$\tilde{S}_N^{(0)}(f) = 2h \sum_{j=1}^{m} f(x_{2j-1}) \qquad x_j = a + jh \qquad h = \frac{b-a}{N}$$

Fehler: $\tilde{R}_N^{(0)}(f) \leq \max_{\xi \in [a,b]} \frac{|f''(\xi)|}{6}(b-a)h^2$

4 Gewöhnlichen Differentialgleichungen

Gegeben Eine Funktion $f : [a, b] \times \mathbb{R}^n \to \mathbb{R}^n$ und ein Anfangswert $y_0 \in \mathbb{R}^n$

Gesucht Eine Funktion $y : [a, b] \to \mathbb{R}^n$, deren Ableitung y' eine gewöhnliche Differentialgleichung der Form

$$y'(t) = f(t, y(t)), \quad t \in [a, b]$$

erfüllt und zudem der Anfangsbedingung $y(a) = y_0$ genügt.

$$\begin{aligned} y'(t) &= f(t, y(t)), \quad t \in [a, b] \\ y(a) &= y_0 \end{aligned} \tag{AWP}$$

Oftmals bezeichnet t dabei die Zeit, woher der Name *Anfangswertproblem* stammt.

4.1 Existenz- und Eindeutigkeit

Sei $f : [a, b] \times \mathbb{R}^n \to \mathbb{R}^n$ Lipschitz-stetig. Dann gilt:

- Zu jedem Anfangswert besitzt das AWP exakt eine Lösung $y \in C^1([a, b]; \mathbb{R}^n)$ (Satz von Picard/Lindelöff).
- Sind y, z Lösungen zu den Anfangswerten $y(a) = y_0$ bzw. $z(a) = z_0$, dann gilt:

  $$\forall t \in [a, b] \|y(t) - z(t)\| \leq e^{L(t-a)} \|y_0 - z_0\|$$

  Wobei L die Lipschitz-Konstante darstellt. Anders ausgedrückt: Die Lösung eines AWPs hängt stetig vom Anfangswert ab.

4.2 Numerische Verfahren

Grundidee: Zerlege das Intervall $[a, b]$ in Teilintervalle $t_j = a + jh$, $j = 0, 1, \cdots, N$, $h = \frac{b-a}{N}$ und approximiere das Integral des AWPs durch interpolatorische Quadratur und erhalte eine annähernde Lösung $u_j \approx y(t_j)$. Der Fehler $e_j = y(t_j) - u_j$ wird dabei *Diskretisierungsfehler* genannt.

Dabei können alle folgenden Verfahren als

$$u_0 = y_0 u_{j+1} = u_j + h\Phi(t_j, h; u_j, u_{j+1}), \quad j = 0, \cdots, N - 1$$

geschrieben werden. Die Funktion $\Phi(t, h; u, v)$ heißt dann *Verfahrensfunktion*. Hängt diese nicht von v ab, heißt das Verfahren *explizit*, sonst *implizit*.

4.2.1 Explizites Euler-Verfahren

Verfahrensfunktion:

$$\Phi(t; u) = f(t, u)$$

Verfahren:

$$\begin{aligned} u_0 &\coloneqq y_0 \\ u_{j+1} &\coloneqq u_j + hf(t_j,\, u_j), \quad j = 0, \cdots, N-1 \end{aligned}$$

4.2.2 Implizites Euler-Verfahren

Verfahrensfunktion:

$$\Phi(t, h; u, v) = f(t + h,\, v)$$

Verfahren:

$$\begin{aligned} u_0 &\coloneqq y_0 \\ u_{j+1} &\coloneqq u_j + hf(t_{j+1},\, u_{j+1}), \quad j = 0, \cdots, N-1 \end{aligned}$$

4.2.3 Verfahren von Heun (1. RK-Verfahren 2. Ordnung)

Verfahren:

$$\begin{aligned} u_0 &\coloneqq y_0 \\ u_{j+1} &\coloneqq u_j + \frac{h}{2}(k_1 + k_2), \quad j = 0, \cdots, N-1 \\ k_1 &\coloneqq f(t_j,\, u_j) \\ k_2 &\coloneqq f(t_{j+1},\, u_j + hk_1) \end{aligned}$$

4.2.4 Modifiziertes Euler-Verfahren (2. RK-Verfahren 2. Ordnung)

$$\begin{aligned} u_0 &\coloneqq y_0 \\ u_{j+1} &\coloneqq u_j + hk_2, \quad j = 0, \cdots, N-1 \\ k_1 &\coloneqq f(t_j,\, u_j) \\ k_2 &\coloneqq f\Big(t_j + \frac{h}{2},\, u_j + \frac{h}{2}k_1\Big) \end{aligned}$$

4.2.5 Klassisches Runge-Kutta-Verfahren 4. Ordnung (RK4)

$$
\begin{aligned}
u_0 &\coloneqq y_0 \\
u_{j+1} &\coloneqq u_j + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad j = 0, \cdots, N-1 \\
k_1 &\coloneqq f(t_j,\, u_j) \\
k_2 &\coloneqq f\Big(t_j + \frac{h}{2},\, u_j + \frac{h}{2}k_1\Big) \\
k_3 &\coloneqq f\Big(t_j + \frac{h}{2},\, u_j + \frac{h}{2}k_2\Big) \\
k_4 &\coloneqq f\Big(t_{j+1},\, u_j + hk_3\Big)
\end{aligned}
$$

4.2.6 Explizite Runge-Kutta-Verfahren und Butcher-Schema

Ein r-stufiges Runge-Kutta-Verfahren hat den allgemeinen Aufbau:

$$
\begin{aligned}
k_i(t, u, h) = k_i &\coloneqq f\Big(t + \gamma_i h,\, u + h\sum_{j=1}^{i-1} \alpha_{i,j} k_j\Big), \quad i = 1, \cdots, r \\
\Phi(t, h; u) &= \sum_{i=1}^{r} \beta_i k_i
\end{aligned}
$$

Die Werte für γ_i, β_j und α_{kl} lassen sich durch das *Butcher-Schema* kompakt beschreiben:

γ_1	0				
γ_2	$\alpha_{2,1}$	0			
γ_3	$\alpha_{3,1}$	$\alpha_{3,2}$	0		
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$	
γ_r	$\alpha_{r,1}$	$\cdots$	$\cdots$	$\alpha_{r,r-1}$	0
	β_1	β_2	$\cdots$	β_{r-1}	β_r

Tabelle 4.1: Butcher-Schema für explizite r-stufige RK-Verfahren

Für die oben vorgestellten Verfahren ergeben sich folgende Butcher-Schemata:

Explizites Euler-Verfahren

0	0
	1

Modifiziertes Euler-Verfahren

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

Verfahren von Heun

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

Klassisches RK-Verfahren 4. Ordnung (RK4)

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

Tabelle 4.2: Klassische Butcher-Schemata

4.2.7 Implizite Runge-Kutta-Verfahren und Butcher-Schema

Ähnlich wie für explizite RK-Verfahren (siehe 4.2.6) lässt sich das Butcher-Schema auch für r-stufige implizite Runge-Kutta-Verfahren verallgemeinern:

$$k_i(t,u,h) = k_i := f\Big(t+\gamma_i h,\, u + h\sum_{j=1}^{r}\alpha_{ij}k_j\Big), \quad i = 1,\cdots,r$$

$$\Phi(t,h;u) = \sum_{i=1}^{r}\beta_i k_i$$

Das Butcher-Schema bildet dann keine strikte untere Diagonalmatrix mehr und sieht wie folgt aus:

γ_1	$\alpha_{1,1}$	$\cdots$	$\cdots$	$\alpha_{1,r-1}$	$\alpha_{1,r}$
γ_2	$\alpha_{2,1}$	$\cdots$	$\cdots$	$\alpha_{2,r-1}$	$\alpha_{2,r}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
γ_r	$\alpha_{r,1}$	$\cdots$	$\cdots$	$\alpha_{r,r-1}$	$\alpha_{r,r}$
	β_1	β_2	$\cdots$	β_{r-1}	β_r

Tabelle 4.3: Butcher-Schema für implizite r-Stufige RK-Verfahren

4.3 Konvergenz und Konsistenz

Sei Φ eine Verfahrensfunktion. Dann heißt die Größe

$$\tau(t,h) = \frac{1}{h}\Big(y(t+h) - y(t) - h\Phi\big(t,h;y(t),y(t+h)\big)\Big), \quad h>0, t\in[a,b-h]$$
$$= \frac{1}{h} \times \text{Defekt beim Ensetzen der Lösung in das Verfahren}$$

lokaler Abbruchfehler oder *Konsistenzfehler* des Verfahrens an der Stelle t.

- **Konsistenz von der Ordnung p**

$$\exists C > 0,\, \bar{h} > 0 : \forall 0 < h \leq \bar{h},\, t \in [a, b-h] : \|\tau(t,h)\| \leq Ch^p$$

- **Stabilität**

$$\exists K > 0 : \forall t \in [a,b], u, v, \tilde{u}, \tilde{v} \in \mathbb{R}^n : \|\Phi(t,h;u,v) - \Phi(t,h;\tilde{u},\tilde{v})\| \leq K(\|u-\tilde{u}\| + \|v-\tilde{v}\|)$$

- **Konvergenz von der Ordnung p**

$$\exists M > 0,\, H > 0 : \forall j = 0, \cdots, N,\, h = \frac{b-a}{N} \leq H : \|e_j\| = \|y(t_j) - u_j\| \leq Mh^p$$

Ist ein Verfahren (APX) konsistent von der Ordnung p und stabil, dann ist es auch konvergent von der Ordnung p.

4.3.1 Konsistenzordnungen

Für die oben vorgestellten Verfahren ergeben sich folgende Konsistenzordnungen:

Verfahren	**Konsistenzordnung**
Explizites Euler-Verfahren	1
Implizites Euler-Verfahren	1
Verfahren von Heun	2
Modifiziertes Euler-Verfahren	2
RK4	4

Tabelle 4.4: Konsistenzordnungen

Konsistenzordnungen von Runge-Kutta-Verfahren

Durch das Butcher-Schema können Verfahren von beliebiger Konsistenzordnung p erzeugt werden (hierzu muss die Stufenanzahl r groß genug gewählt werden).

Bis zu Konsistenzordnung $p = 4$ lässt sich die Konsistenzordnung einfach nachrechnen (seien γ_i, β_j und α_{kl} wie im Butcher-Schema unter 4.2.7). Dann gilt: Das Verfahren ist von Konsistenzordnung…

$p = 1$ wenn gilt:

$$\sum_{i=1}^{r} \beta_i = 1$$

$p = 2$ wenn die Anforderungen für $p = 1$ gelten und gilt:

$$\sum_{i=1}^{r} \beta_i \gamma_i = \frac{1}{2}$$

$p = 3$ wenn die Anforderungen für $p = 2$ gelten und gilt:

$$\sum_{i=1}^{r} \beta_i \gamma_i^2 = \frac{1}{3} \qquad \sum_{i,j=1}^{r} \beta_i \alpha_{i,j} \gamma_j = \frac{1}{6}$$

$p = 4$ wenn die Anforderungen für $p = 3$ gelten und gilt:

$$\sum_{i=1}^{r} \beta_i \gamma_i^3 = \frac{1}{4} \qquad \sum_{i,j=1}^{r} \beta_i \gamma_i \alpha_{i,j} \gamma_j = \frac{1}{8}$$
$$\sum_{i,j=1}^{r} \beta_i \alpha_{i,j} \gamma_j^2 = \frac{1}{12} \qquad \sum_{i,j,k=1}^{r} \beta_i \alpha_{i,j} \alpha_{j,k} \gamma_k = \frac{1}{24}$$

Somit lassen sich die Konsistenzordnungen in Tabelle 4.4 leicht nachrechnen.

4.4 Steife Differentialgleichungen

Ausgangspunkt: Ein Anfangswertproblem über ein System von n gewöhnlichen Differentialgleichungen (AWPn):

$$y'(t) = f(t, y(t)), \quad t \in [a, b]$$
$$y(a) = y_0$$

mit $f : [a, b] \times \mathbb{R}^n \to \mathbb{R}^n, \quad y_0 \in \mathbb{R}^n$.

Bei einer *steifen Differentialgleichung* ist die Lösung zusammengesetzt auf einem langsam veränderlichen Teil (meist abklingend) und einem Anteil, der i.A. sehr schnell gedämpft wird.

Ist das Systems linear (LAWPn), d.h. das AWPn ist durch folgende Gleichungen gegeben (mit einer Matrix $A \in \mathbb{R}^n$ und einem Vektor $c \in \mathbb{R}^n$):

$$y'(t) = Ay(t) + c, \quad t \in [a, b]$$
$$y(a) = y_0$$

Sei fernen A diagonalisierbar mit den Eigenwerten λ_i und den Eigenvektoren v_i. Dann hat die allgemeine Lösung folgende Form (mit einer partikulären Lösung y_P):

$$y(t) = y_H(t) + y_P(t), \quad y_H(t) = \sum_{i=1}^{n} C_i e^{\lambda_i t} v_i$$

Gilt nun $\mathrm{Re}(\lambda_i) < 0$ für $i = 1, \cdots, n$, so gilt aufgrund von $\left|e^{\lambda_i t}\right| = e^{\mathrm{Re}(\lambda_i) t}$:

$$\lim_{t \to \infty} y_H(t) \to 0$$

Die Lösungen nähern sich also insgesamt der partikulären Lösung y_P an.

- Dabei klingen Summanden in y_H mit $\mathrm{Re}(\lambda_i) \ll -1$ sehr schnell und Summanden mit $\mathrm{Re}(\lambda_i) \not\ll -1$ deutlich langsamer ab.
- Existieren Eigenwerte mit $\mathrm{Re}(\lambda_i) \ll -1$ *und* Eigenwerte mit schwach negativem Realteil, wird das System *steif* genannt.

Problematik Numerische Verfahren (insbesondere explizite Verfahren) haben oftmals Probleme bei der Approximation von steifen Differentialgleichungen. Sie benötigen häufig sehr kleine Schrittweiten, um annähernd eine Lösung zu approximieren.

4.4.1 Modellgleichung

Beobachtung: Arbeitet ein numerisches Verfahren für alle DGL $z' = \text{diag}(\lambda_1, \cdots, \lambda_n)z$ zuverlässig, dann liefert es auch für das steife System $y' = Ay, \quad y(0) = y_0$ gute Ergebnisse.

Zum testen von numerischen Verfahren wird eine *Modellgleichung* definiert:

$$y' = \lambda y, \quad y(0) = 1, \quad \text{mit } \lambda \in \mathbb{C}, \text{Re}(\lambda) < 0$$

Diese hat die Lösung $y = e^{\lambda t}$ und aufgrund von $\text{Re}(\lambda) < 0$ gilt $\lim_{t\to\infty} y(t) = 0$. Die Lösung fällt also abhängig von der Größe von $|\text{Re}(\lambda)|$ sehr unterschiedlich stark ab.

Damit ein numerisches Verfahren gut geeignet ist, muss die numerisch berechnete Näherungslösung von

$$y' = \lambda y, \quad y(0) = 1, \text{mit } \lambda \in \mathbb{C}, \text{Re}(\lambda) < 0$$

soll die Eigenschaften der analytischen Lösung $y = e^{\lambda t}$, insbesondere $\lim_{t\to\infty} y(t) = 0$ möglichst gut widerspiegeln.

4.4.2 Stabilität

Stabilitätsfunktion und -gebiet Bei vielen Einschrittverfahren produziert die Anwendung auf die Modellgleichung eine Verfahrensvorschrift der Form:

$$u_{j+1} = R(q)u_j \quad \text{mit } q = \lambda h$$

Mit einer Funktion $R : D \to \mathbb{C}, \quad 0 \in D \subseteq \mathbb{C}$. Diese Funktion wird *Stabilitätsfunktion* genannt.

Die Menge

$$S = \{q \in \mathbb{C} \,|\, |R(q)| \leq 1\}$$

heißt dann *Stabilitätsgebiet*.

Für ein r-stufiges Runge-Kutta-Verfahren nach Butcher-Schema kann die Modellgleichung wie folgt berechnet werden (wobei $\beta = (\beta_1, \cdots, \beta_r)^T \in \mathbb{R}^r$, $A = (\alpha_{i,j})$ die Matrix der α-Koeffizienten und $\mathbb{1} \in \mathbb{R}^r$ der Einsvektor ist):

$$u_{j+1} = \big(1 + \lambda h \beta^T (I - \lambda h A)^{-1} \mathbb{1}\big) u_j = \big(1 + q\beta^T (I - qA)^{-1} \mathbb{1}\big) u_j$$

A-Stabilität Ein Verfahren heißt *absolut stabil* (*A-stabil*) gdw. seine Anwendung auf die Modellgleichung für jede Schrittweite $h > 0$ eine Folge $(u_j)_{j\in\mathbb{N}_0}$ produziert mit:

$$\forall j = 0, 1, \cdots : |u_{j+1}| \leq |u_j|$$

Anders ausgedrückt: Die produzierte folge muss monoton fallend sein.

$$\text{A-stabil} \iff \forall q \in C, \text{Re}(q) < 0 : |R(q)| \leq 1 \iff S \supset \{q \in \mathbb{C} \,|\, \text{Re}(q) < 0\}$$

L-Stabilität Ein Verfahren heißt *L-stabil* gdw. es A-stabil ist und die Stabilitätsfunktion zudem gegen 0 konvergiert, d.h.:

$$\lim_{q\to-\infty} R(q) = 0$$

5 Lineare Gleichungssysteme

Gegeben Ein lineares Gleichungssystem (LGS) $Ax = b$ mit:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \in \mathbb{R}^{n \times n}, \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ v_n \end{bmatrix} \in \mathbb{R}^n$$

Gesucht Eine Lösung $x \in \mathbb{R}^n$ des Gleichungssystems.

5.1 Lösungstheorie

Ein LGS hat eine Lösung gdw. gilt $\operatorname{rank}(A) = \operatorname{rank}(A, b)$. Das LGS hat eine eindeutige Lösung gdw. A regulär ist (d.h. $\det(A) \neq 0$). Die Lösung lautet dann $x = A^{-1}b$.

5.2 Gaußsches Eliminationsverfahren, Dreieckszerlegung

Bei dem gaußschen Eliminationsverfahren wird versucht, das LGS durch die elementaren Operationen

- Addition eines Vielfachen einer Gleichung zu einer anderen.
- Zeilenvertauschung, d.h. Vertauschung von Gleichungen.
- Spaltenvertauschung, d.h. Umnummerierung der Unbekannten.

in ein gestaffeltes Gleichungssystem $Ry = c, \quad y_{\sigma_i} = x_i, \quad i = 1, \cdots, n$ umzuformen, welches die selben Lösungen besitzt wie das LGS mit Spaltenpermutationen $\sigma_1, \cdots, \sigma_n$ und einer oberen Dreiecksmatrix R.

5.2.1 Lösung gestaffelter Gleichungssysteme

Rückwärtssubstitution Sei $Ry = c$ ein gestaffeltes Gleichungssystem mit einer oberen Dreiecksmatrix

$$R = \begin{bmatrix} r_{1,1} & \cdots & r_{1,n} \\ & \ddots & \vdots \\ 0 & & r_{n,n} \end{bmatrix}$$

Dieses Gleichungssystem lässt sich leicht durch Rückwärtssubstitution lösen (mit R invertierbar und $c = (c_1, \cdots, c_n)^T$):

$$y_i = \frac{c_i - \sum_{j=i+1}^{n} r_{i,j} y_j}{r_{i,i}}, \quad i = n, n-1, \cdots, 1$$

Der Berechnungsaufwand liegt dabei in $\mathcal{O}(n^2)$, sofern keine spezielle Besetztheit vorliegt.

Vorwärtssubstitution Sei $Lz = d$ ein gestaffeltes Gleichungssystem mit einer unteren Dreiecksmatrix:

$$L = \begin{bmatrix} l_{1,1} & & 0 \\ \vdots & \ddots & \\ l_{n,1} & \cdots & l_{n,n} \end{bmatrix}$$

Dieses Gleichungssystem lässt sich leicht durch Vorwärtssubstitution lösen (mit L invertierbar und $d = (d_1, \cdots, d_n)^T$):

$$z_i = \frac{d_i - \sum_{j=1}^{i-1} l_{i,j} z_j}{l_{i,i}}, \quad i = 1, 2, \cdots, n$$

Der Berechnungsaufwand liegt dabei in $\mathcal{O}(n^2)$, sofern keine spezielle Besetztheit vorliegt.

5.2.2 Gaußsches Eliminationsverfahren

Sämtliche Operationen bei dem gaußschen Eliminationsverfahren werden an der erweiterten Koeffizientenmatrix vorgenommen:

$$(A, b) = \left[\begin{array}{ccc|c} a_{1,1} & \cdots & a_{1,n} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ a_{n,1} & \cdots & a_{n,n} & b_n \end{array}\right]$$

Grundversion

Initialisiere $(A^{(1)}, b^{(1)}) \leftarrow (A, b)$
for $k = 1, \cdots, n-1$ **do**
  Finde $r \in \{k, \cdots, n\}$ mit $a_{r,k}^{(k)} \neq 0$ (Pivotsuche)
  if $\forall r : a_{r,k}^{(k)} = 0$ **then**
    return A nicht invertierbar
  Vertausche Zeile r und k, erhalte $(\tilde{A}^{(k)}, \tilde{b}^{(k)})$
  for $i = k+1, \cdots, n$ **do**
    Subtrahiere $l_{i,k} = \frac{\tilde{a}_{i,k}^{(k)}}{\tilde{a}_{k,k}^{(k)}}$ der k-ten Gleichung von der i-ten Gleichung
  Erhalte $(A^{(k+1)}, b^{(k+1)})$
return $(A^{(n)}, b^{(n)})$

Pivotstrategie

- Das Element $a_{r,k}^{(k)}$ heißt *Pivotelement*.
- Theoretisch ist es möglich, dass jedes Pivotelement $\neq 0$ ist.
- Die Wahl kleiner Pivotelemente kann jedoch zu einer dramatischen Verstärkung von Rundungsfehlern führen.
- Um dies zu vermeiden, muss eine geeignete Pivotstrategie verwendet werden.

- Hierzu gibt es die folgenden Strategien:

  Spaltenpivotsuche Wähle $r \in \{k, \cdots, n\}$ mit $\left|a_{r,k}^{(k)}\right| = \max_{i=k,\cdots,n} \left|a_{i,k}^{(k)}\right|$.

  Vollständige Pivotsuche Wähle $r, s \in \{k, \cdots, n\}$ mit $\left|a_{r,s}^{(k)}\right| = \max_{i,j=k,\cdots,n} \left|a_{i,j}^{(k)}\right|$.

- Für beide Verfahren sollten die Zeilen von A „äquilibriert“ sein, d.h. die Normen der Zeilen sollten in der gleichen Größenordnung liegen.

Verfahren mit Spaltenpivotsuche

```
1  Initialisiere (A^(1), b^(1)) ← (A, b)
2  for k = 1, ··· , n − 1 do
3      Finde r ∈ {k, ··· , n} mit |a_{r,k}^(k)| = max_{i=k,···,n} |a_{i,k}^(k)| (Spaltenpivotsuche)
4      if a_{r,k}^(k) = 0 then
5          return A nicht invertierbar
6      Vertausche Zeile r und k, erhalte (Ã^(k), b̃^(k))
7      for i = k + 1, ··· , n do
8          Subtrahiere l_{i,k} = ã_{i,k}^(k) / ã_{k,k}^(k) der k-ten Gleichung von der i-ten Gleichung
9      Erhalte (A^(k+1), b^(k+1))
10 return (A^(n), b^(n))
```

Verfahren mit vollständiger Pivotsuche

```
1  Initialisiere (A^(1), b^(1)) ← (A, b)
2  for k = 1, ··· , n − 1 do
3      Finde r, s ∈ {k, ··· , n} mit |a_{r,s}^(k)| = max_{i,j=k,···,n} |a_{i,j}^(k)| (vollständige P
4      if a_{r,s}^(k) = 0 then
5          return A nicht invertierbar
6      Vertausche Zeile r und k sowie Spalten s und k, erhalte (Ã^(k), b̃^(k))
7      for i = k + 1, ··· , n do
8          Subtrahiere l_{i,k} = ã_{i,k}^(k) / ã_{k,k}^(k) der k-ten Gleichung von der i-ten Gleichun
9      Erhalte (A^(k+1), b^(k+1))
10 return (A^(n), b^(n))
```

Nach der Lösung des Dreieckssystems müssen die Spalten in x zurück getauscht werden!

5.2.3 LR-Zerlegung

Speicherung der Multiplikatoren $l_{i,k}$ in einer separaten unteren Diagonalmatrix L:

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

Dies liefert $LR = PAQ$, wobei $P = P_{n-1} \cdots P_2 \cdot P_1$ die Permutationsmatrix der Zeilen ist (d.h. die Permutationen der Zeilen von A). In einem Einzelschritt der Permutation werden die Zeilen k und r der Einheitsmatrix getauscht. $Q = Q_1 \cdot Q_2 \cdots Q_{n-1}$ ist die Permutationsmatrix der Spalten, die analog zur Zeilenpermutationsmatrix erstellt wird. Im Falle der normalen Spaltenpivotsuche gilt $Q = I$.

Durch eine solche LR-Zerlegung kann, nach Lösung eines LGS $Ax = b$ ein anderes Gleichungssystem $Ay = c$ gelöst werden (gleiche Matrix, anderes erwartetes Ergebnis):

1. Löse $Lz = Pc$ nach z durch Vorwärtssubstitution.
2. Löse $Ry = z$ nach y durch Rückwärtssubstitution.

5.2.4 Matrixklassen ohne Pivotsuche

Es wird keine Pivotsuche benötigt, wenn z.B.:

- $A = A^T$ symmetrisch positiv definit ist, also $\forall x \in \mathbb{R}^n \setminus \{0\} : x^T Ax > 0$ gilt.
- A strikt diagonaldominant ist, also $|a_{i,i}| > \sum_{j=1,\, j\neq i}^{n} |a_{i,j}|, \quad i = 1, \cdots, n$ gilt.
- A eine M-Matrix ist, d.h. es gilt:
  - $a_{i,i} > 0, \quad i = 1, \cdots n$
  - $a_{i,j} \leq 0, \quad i \neq j$
  - $D^{-1}(A - D), \quad D = \text{diag}(a_{1,1}, \cdots, a_{n,n})$ hat nur Eigenwerte λ mit $|\lambda| < 1$

5.3 Cholesky-Verfahren

Sei $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit. Dann existiert exakt eine untere Dreieckmatrix L mit positiven Diagonaleinträgen $l_{i,i} > 0$, sodass:

$$LL^T = A$$

gilt. Diese Zerlegung wird *Cholesky-Zerlegung* genannt. Außerdem besitzt A eine eindeutige Dreieckszerlegung $\tilde{L}\tilde{R} = A$, wobei $\tilde{L} = LD^{-1}$, $\tilde{R} = DL^T$ mit $D = \text{diag}(l_{1,1}, \cdots, l_{n,n})$. Diese Zerlegung wird vom Gauß-Algorithmus ohne Pivotsuche produziert.

5.3.1 Verfahren

for $j = 1, \cdots, n-1$ **do**

$$i_{j,j} = \sqrt{a_{j,j} - \sum_{k=1}^{j-1} l_{j,k}^2}$$

  for $i = j+1, \cdots, n$ **do**

$$l_{i,j} = \frac{a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} l_{j,k}}{l_{j,j}}$$

5.3.2 Eigenschaften

- Durch das Ausnutzen der Symmetrie benötigt das Cholesky-Verfahren nur etwas die Hälfte an Rechenschritten im Gegensatz zu dem Gauß-Algorithmus.
- Das Cholesky-Verfahren ist zusätzlich die effizienteste Methode auf positive Definitheit, indem folgendes geprüft wird:
  1. $a = a_{j,j} - \sum_{k=1}^{j-1} l_{j,k}^2$
  2. Falls $a \leq 0$: Stopp, A ist nicht positiv definit.
  3. Setze ansonsten $l_{j,j} = \sqrt{a}$.

5.4 Fehlerabschätzungen und Rundungsfehlereinfluss

Gerade bei großen Matrizen können Rundungsfehler die Rechnung erheblich beeinflussen. Somit muss betrachtet werden, wie sich die einzelnen Verfahren bei Störung der Matrix beeinflussen lassen.

5.4.1 Fehlerabschätzungen für gestörte Gleichungssysteme

Sei $Ax = b$ das Gleichungssystem und $(A + \Delta A)\tilde{x} = b + \Delta b$ das gestörte Gleichungssystem mit ΔA, Δb klein. Die Frage ist nun, wie klein der Fehler $x - \tilde{x}$ ist?

Sei $A \in \mathbb{R}^{n \times n}$ invertierbar, b, $\Delta b \in \mathbb{R}^n$, $b \neq 0$ und $\Delta A \in \mathbb{R}^{n \times n}$ mit $\|\Delta A\| < \frac{1}{\|A^{-1}\|}$ mit einer beliebigen durch eine Vektornorm $\|\cdot\|$ induzierte Matrixnorm $\|\cdot\|$. Seien ferner x, $\tilde{x}$ die Lösungen des Gleichungssystems. Dann gilt für den relativen Fehler:

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq \frac{\operatorname{cond}(A)}{1 - \operatorname{cond}(A)\frac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right)$$

5.4.2 Rundungsfehleranalyse

Sei $A \in \mathbb{R}^{n \times n}$ invertierbar und eps die Maschinengenauigkeit. Außerdem wird das Gauß-Verfahren mit einer Pivotstrategie ausgeführt, die $|l_{i,j}| \leq 1$ sicherstellt (z.B. Spaltenpivotsuche oder vollständige Pivotsuche).

Dann wird $\bar{L}$, $\bar{R}$ wie folgt errechnet:

$$\bar{L}\bar{R} = PAQ + F, \quad |f_{i,j}| \leq 2j\bar{a}\frac{\text{eps}}{1-\text{eps}}$$

Dabei sind P, Q die resultierenden Permutationen und

$$\bar{a} = \max_k \bar{a}_k \qquad \bar{a}_k = \max_{i,j} \left| a_{i,j}^{(k)} \right|$$

Wird eine Näherungslösung $\bar{x}$ durch Vorwärts- und Rückwärtssubstitution berechnet, dann existiert eine Matrix E mit:

$$(A+E)\bar{x} = b \qquad |e_{i,j}| \leq \frac{2(n+1)\text{eps}}{1-n\cdot\text{eps}} \left|\bar{L}\right|_{i,j} \left|\bar{R}\right|_{i,j} \leq \frac{2(n+1)\text{eps}}{1-n\cdot\text{eps}} n\bar{a}$$

Für $\bar{a}_k$ gelten folgende Abschätzungen:

Spaltenpivotsuche $\bar{a}_k \leq 2^k \max_{i,j} |a_{i,j}|$ (In der Regel ist diese Schranke viel zu pessimistisch, in der Praxis tritt fast immer $\bar{a}_k \leq 10 \max_{i,j} |a_{i,j}|$ auf.)

Spaltenpivotsuche (bei Tridiagonalmatrizen) $\bar{a}_k \leq 2 \max_{i,j} |a_{i,j}|$

Vollständige Pivotsuche $\bar{a}_k \leq f(k) \max_{i,j} |a_{i,j}|$ mit $f(k) = \sqrt{k} \cdot \prod_{i=1}^{k-1} \sqrt[i]{i+1}$ ($f(k)$ wächst sehr langsam. Bislang ist noch kein Beispiel mit $\bar{a}_k \geq (k+1) \max_{i,j} |a_{i,j}|$ entdeckt worden.)

6 Nichtlineare Gleichungssysteme

Gesucht ist eine Lösung $x \in D$ von

$$F(x) = 0$$

mit einer gegebenen Abbildung

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_n \end{bmatrix} : D \to \mathbb{R}^n$$

wobei $D \subseteq \mathbb{R}$ nichtleer und abgeschlossen und F mindestens einmal stetig differenzierbar mit einer Jacobi-Matrix $F'(x)$.

Im Gegensatz zu linearen Gleichungssystemen können nichtlineare Gleichungssysteme mehrere oder unendliche viele (isolierte) Lösungen besitzen.

6.1 Newton-Verfahren

Angenommen es gilt $D = \mathbb{R}^N$, also $F : \mathbb{R}^n \to \mathbb{R}^n$.

6.1.1 Lokales Newton-Verfahren

1 Wähle einen Startpunkt $x^{(0)} \in \mathbb{R}^n$
2 **for** $k = 0, 1, \cdots$ **do**
3 **if** $F(x^{(k)}) = 0$ **then**
4 **return** $x^{(k)}$
5 **else**
6 Berechne Newton-Schritt $s^{(k)} \in \mathbb{R}^n$ durch Lösen der Newton-Gleichung

$$F'(x^{(k)})\, s^{(k)} = -F(x^{(k)})$$

7 Setze $x^{(k+1)} \leftarrow x^{(k)} + s^{(k)}$

Konvergenz

Das Newton-Verfahren konvergiert i.d.R. nur für Startpunkte, die nahe genug an der Lösung liegen (lokale Konvergenz).

6.1.2 Globalisierung

Modifikation des Newton-Verfahrens, sodass für jeden Newton-Schritt eine Schrittweite von $\sigma_k \in (0, 1]$ verwendet wird:

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

Die Schrittweite wird so bestimmt, dass

$$\left\| F(x^{(k+1)}) \right\|_2 < \left\| F(x^{(k)}) \right\|_2$$

gilt und die Abnahme „ausreichend groß“ ist.

Schrittweitenwahl nach Armijo

Sei $\delta \in (0, \frac{1}{2})$ fest gegeben (z.B. $\delta = 10^{-3}$). Dann wird als Schrittweite das größte $\sigma_k \in \{1, \frac{1}{2}, \frac{1}{2}, \cdots\} = \{\frac{1}{2^k} \mid k \in \mathbb{N}\}$ gewählt, sodass gilt:

$$\left\| F(x^{(k)} + \sigma_k s^{(k)}) \right\|_2^2 \leq \left\| F(x^{(k)}) \right\|_2^2 - 2\delta\sigma_k \left\| F(x^{(k)}) \right\|_2^2$$

Globalisiertes Newton-Verfahren

1 Wähle einen Startpunkt $x^{(0)} \in \mathbb{R}^n$
2 **for** $k = 0, 1, \cdots$ **do**
3 **if** $F(x^{(k)}) = 0$ **then**
4 **return** $x^{(k)}$
5 **else**
6 Berechne Newton-Schritt $s^{(k)} \in \mathbb{R}^n$ durch Lösen der Newton-Gleichung

$$F'(x^{(k)})\, s^{(k)} = -F(x^{(k)})$$

7 Bestimme σ_k nach Armijo-Regel
8 Setze $x^{(k+1)} \leftarrow x^{(k)} + \sigma_k s^{(k)}$
9 **end**
10 **end**

7 Eigenwert- und Eigenvektorberechnung

7.1 Störungstheorie

- Bei oberen/unteren Diagonalmatrizen sind die Eigenwerte die Diagonalelemente.
- Verfahren wie das QR-Verfahren reduzieren das strikte untere Dreieck.
- Die Störungsresultate für Eigenwerte liefern u.a. Schranken, wie gut die Diagonalelemente mit den Eigenwerten übereinstimmen.

Bezeichnet $\lambda_i(A), \quad i = 1, \cdots, n$ die angeordneten Eigenwerte einer Matrix $A \in \mathbb{C}^{n\times n}$, dann sind die Abbildungen $A \in \mathbb{C}^{n\times n} \mapsto \lambda_i(A), \quad i = 1, \cdots, n$ stetig. D.h. die Eigenwerte hängen stetig von der Matrix ab.

7.1.1 Gershgorin-Kreise

Gershgorin-Kreise werden zur Abschätzung der Lage der Eigenwerte verwendet.
Sei $A = (a_{i,j}) \in \mathbb{C}^{n\times n}$ beliebig. Dann gilt $\sigma(A) \subset \bigcup_{i=1}^{n} K_i$ mit den *Gershgorin-Kreisen*

$$K_i := \left\{ \mu \in \mathbb{C} \,|\, |\mu - a_{i,i}| \leq \sum_{j=1,\, j\neq i}^{n} |a_{i,j}| \right\}, \quad i = 1, \cdots, n$$

Ist die Vereinigung G_1 von k Gershgorin-Kreisen disjunkt von der Vereinigung G_2 der restlichen $n - k$ Gershgorin-Kreisen, dann enthält G_1 genau k und G_2 genau $n - k$ Eigenwerte von A.
Bei einer Störung einer Matrix verschieben sich die Gershgorin-Kreise leicht.

7.2 Numerische Verfahren

Die numerischen Verfahren zur Berechnung der Eigenwerte lassen sich in zwei Klassen aufteilen:

- **Vektoriteration**
  Beginnend mit einem Startvektor wird dieser so lange verfeinert, bis die Eigenvektoren angenähert sind.
- **Ähnlichkeitstransformationen**
  Beginnend von der Matrix aus wird diese so lange transformiert, bis das untere Dreieck gegen Null konvergiert und die Eigenwerte auf der Hauptdiagonalen stehen.

Sei im folgenden $A \in \mathbb{C}^{n\times n}$ die Matrix, von der die Eigenwerte bestimmt werden soll.

7.2.1 Vektoriteration

Für eine Matrix $B \in \mathbb{C}^{n\times n}$ ist die Vektoriteration gegeben durch:

$$z^{(k+1)} = \frac{1}{\left\|Bz^{(k)}\right\|} Bz^{(k)}, \quad k = 0, 1, \cdots$$

Mit einem Startvektor $k^{(0)} \in \mathbb{C}^n \setminus \{0\}$.

Bei einer geeigneten Wahl von B ergibt $z^{(k)}$ eine Näherung für den betragsmäßig größten Eigenwert λ. Die Näherung für den Eigenwert λ ergibt sich dann durch den *Rayleighquotienten*:

$$R(z^{(k)}, B) = \frac{(z^{(k)})^H Bz^{(k)}}{(z^{(k)})^H z^{(k)}}$$

Konvergenz

Vektoriteration von Mises

Mit der *einfachen Vektoriteration von Mises* wird $B = A$ gewählt.

Die Konvergenz geht dann direkt aus 7.2.1 hervor.

Nachteile

- Langsame Konvergenz bei schlechter Trennung der Eigenwerte.
- Einschränkung auf die Bestimmung des betragsmäßig größten Eigenwert.

Lösung: Inverse Vektoriteration von Wielandt.

Inverse Vektoriteration von Wielandt

Sei μ eine gute Näherung eines Eigenwertes λ_j, sodass $|\lambda_j - \mu| \ll |\lambda_i - \mu|$ gilt für alle $\mu \neq \lambda_j$. Die *inverse Vektoriteration von Wielandt* ist dann:

$$z^{(k+1)} = \frac{\hat{z}^{(k+1)}}{\left\|\hat{z}^{(k+1)}\right\|} \qquad \text{mit } \hat{z}^{(k+1)} = (A - \mu I)^{-1} z^{(k)}$$

In der Praxis wird jedoch nicht $(A - \mu I)^{-1}$ bestimmt, sondern $(A - \mu I)\hat{z}^{(k+1)} = z^{(k)}$ gelöst.

7.2.2 QR-Verfahren (von Francis)

Bei dem QR-Verfahren werden unitäre Ähnlichkeitstransformationen auf die Matrix $A^{(1)} = A \in \mathbb{C}^{n\times n}$ angewandt.

```
Initialisiere A^(1) ← A
for l = 1, 2, ··· do
    Berechne QR-Zerlegung Q_l R_l = A^(l),
            wobei Q_l ∈ C^{n×n} unitär, R ∈ C^{n×n} obere Dreiecksmatrix
    A^(l+1) ← R_l Q_l
```

Die Berechnung dieser QR-Zerlegung kann z.B. mit Hilfe des Householder-Verfahrens geschehen (siehe 5).

Konvergenz

Shift-Techniken

Verbreitete Shift-Strategie

Berechnung der Eigenvektoren Die Eigenvektoren können bspw. mit der inversen Vektoriteration berechnet werden, wobei dort als Shifts die berechneten μ verwendet werden.

Householder-Verfahren zur Berechnung

Gegeben Eine Matrix $B \in \mathbb{C}^{n \times n}$.

Ziel Eine unitäre Matrix $Q \in \mathbb{C}^{n \times n}$ und eine obere Dreiecksmatrix $R \in \mathbb{C}^{n \times n}$ mit $B = QR$.