
Mathematik 3 (Numerik)

Zusammenfassung
Fabian Damken



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhaltsverzeichnis

1	Grundlagen	4
1.1	Vektornorm	4
1.2	Induzierte Matrixnormen	4
1.3	Konditionszahl	5
1.4	Spezielle Matrizen	5
1.5	Eigenwerte und Eigenvektoren	5
1.5.1	Charakteristisches Polynom	5
1.5.2	Eigenschaften	6
1.5.3	Diagonalisierbarkeit	6
1.6	\mathcal{O} -Notation	6
2	Interpolation	7
2.1	Polynominterpolation	7
2.1.1	Eindeutigkeit	7
2.1.2	Naiver Lösungsansatz	7
2.1.3	Lagrange-Interpolation	8
2.1.4	Newtonsche Interpolationsformel	8
2.1.5	Fehlerabschätzungen	9
2.1.6	Runges Phänomen	9
2.1.7	Tschebyscheff-Abszissen	9
2.2	Spline-Interpolation	10
2.2.1	Lineare Splines	10
2.2.2	Kubische Splines	11
3	Integration	14
3.1	Geschlossene Newton-Cotes-Quadratur	14
3.1.1	Spezielle geschlossene Newton-Cotes-Formeln	14
3.2	Offene Newton-Cotes-Quadratur	15
3.2.1	Spezielle offene Newton-Cotes-Formeln	15
3.3	Vergleich Geschlossene vs. Offene Newton-Cotes-Quadratur	15
3.4	Summierte Newton-Cotes-Formeln	15
3.4.1	Summierte Trapezregel (geschlossen)	16
3.4.2	Summierte Simpson-Regel (geschlossen)	16
3.4.3	Summierte Rechteck-Regel (offen)	16
4	Gewöhnlichen Differentialgleichungen	17
4.1	Existenz- und Eindeutigkeit	17
4.2	Numerische Verfahren	17
4.2.1	Explizites Euler-Verfahren	17
4.2.2	Implizites Euler-Verfahren	18
4.2.3	Verfahren von Heun (1. RK-Verfahren 2. Ordnung)	18
4.2.4	Modifiziertes Euler-Verfahren (2. RK-Verfahren 2. Ordnung)	18
4.2.5	Klassisches Runge-Kutta-Verfahren 4. Ordnung (RK4)	18
4.2.6	Explizite Runge-Kutta-Verfahren und Butcher-Schema	19
4.2.7	Implizite Runge-Kutta-Verfahren und Butcher-Schema	19

4.3	Konvergenz und Konsistenz	20
4.3.1	Konsistenzordnungen	20
4.4	Steife Differentialgleichungen	21
4.4.1	Modellgleichung	22
4.4.2	Stabilität	22
5	Lineare Gleichungssysteme	24
5.1	Lösungstheorie	24
5.2	Gaußsches Eliminationsverfahren, Dreieckszerlegung	24
5.2.1	Lösung gestaffelter Gleichungssysteme	24
5.2.2	Gaußsches Eliminationsverfahren	24
5.2.3	LR-Zerlegung	24
5.2.4	Matrixdarstellung	25
5.2.5	Matrixklassen ohne Pivotsuche	25
5.3	Cholsky-Verfahren	25
5.3.1	Cholsky-Verfahren \leftrightarrow LR-Zerlegung	25
5.3.2	Verfahren	25
5.3.3	Eigenschaften	25
5.4	Fehlerabschätzungen und Rundungsfehlereinfluss	25
5.4.1	Fehlerabschätzungen für gestörte Gleichungssysteme	25
5.4.2	Rundungsfehleranalyse	25
6	Nichtlineare Gleichungssysteme	26
6.1	Newton-Verfahren	26
6.1.1	Lokales Newton-Verfahren	26
6.1.2	Globalisierung	26
7	Eigenwert- und Eigenvektorberechnung	28
7.1	Störungstheorie	28
7.2	Gershgorin-Kreise	28
7.3	Numerische Verfahren	28
7.3.1	Vektoriteration	28
7.3.2	QR-Verfahren	28

1 Grundlagen

1.1 Vektornorm

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

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

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

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

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

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

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

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

1.3 Konditionszahl

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

Dann ist

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

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. $A^T A = A A^T = I$

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 $x \in \mathbb{C}^n, x \neq 0$ gibt mit:

$$Ax = \lambda x$$

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

Der Unterraum

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

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

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

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

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

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:

- $\lambda \in \sigma(A) \implies \lambda \in \sigma(A^T) \wedge \bar{\lambda} \in \sigma(A^H)$
- 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 $g : \mathbb{N} \rightarrow \mathbb{R}$ bezeichnet $\mathcal{O}(g(n))$ die Menge aller Funktionen, die asymptotische nicht schneller wachsen als g , d.h.:

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

2 Interpolation

Gegeben Funktionaler Zusammenhang $y = f(x)$, $f : [a, b] \rightarrow \mathbb{R}, a < b \in \mathbb{R}$

Bekannt Werte $y_i = f(x_i)$, $i = 0, \dots, n$ (*Stützstellen*)

Ziel Annäherung für $f(x)$ für beliebige $x \in [a, b]$

Interpolationsproblem Suche einfache Ersatzfunktion $\Phi(x)$ mit $\Phi(x_i) = y_i$, $i = 0, \dots, n$

Wunsch Der Fehler $|f(x) - \Phi(x)|$ sollte auf $[a, b]$ möglichst gering sein.

Interpolationsaufgabe

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

2.1 Polynominterpolation

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

$$p_n(x) = \Phi(x, a_0, \dots, a_n) = a_0 + a_1x + \dots + a_nx^n$$

Interpolationsaufgabe: Finde ein Polynom $p_n(x)$ vom Grad $\leq n$, sodass die Interpolationsbedingungen $p_n(x_i) = y_i$, $i = 0, \dots, n$ 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, \dots, 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 → Rundungsfehler werden dramatisch verstärkt.

2.1.3 Lagrange-Interpolation

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

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

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

Die Berechnung der Parameter erfolgt über die *dividierten Differenzen* $f_{[x_0, \dots, x_i]} := \gamma_i$ zu den Stützstellen x_0, \dots, 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):

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

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

$$\begin{array}{l|l} x_0 & f_{[x_0]} = y_0 \quad \searrow \\ & f_{[x_0, x_1]} \quad \searrow \\ x_1 & f_{[x_1]} = y_1 \quad \swarrow & f_{[x_0, x_1, x_2]} \\ & f_{[x_1, x_2]} \quad \swarrow \\ x_2 & f_{[x_2]} = y_2 \quad \swarrow & \vdots \\ \vdots & \vdots & \vdots \end{array}$$

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

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

Mit dem Knotenpolynom

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

gilt für den maximalen Fehler:

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

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

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

2.1.6 Runges Phänomen

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

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

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:

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

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

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

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 $\text{Grad} \leq k$.
- Problem: Die Polynome passen an den Intervallgrenzen mglw. nicht zusammen.
→ Spline-Interpolation: Die Polynome gehen $k - 1$ -mal stetig ineinander über.

Splinefunktion

Sei $\Delta = \{x_i \mid a = x_0 < x_1 < \dots < x_n = b\}$ 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 $\Delta = \{x_i \mid a = x_0 < x_1 < \dots < x_n = b\}$ und Werten $y_i \in \mathbb{R}$, $i = 0, \dots, n$ eine Funktion $s \in S_{\Delta, k}$ mit $s(x_i) = y_i$, $i = 0, \dots, n$.

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):

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

Eindeutigkeit

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

Fehlerabschätzung

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

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

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

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

3 Integration

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

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

Integrationsaufgabe: Zu einem gegebenen, integrierbarem, $f : [a, b] \rightarrow \mathbb{R}$, 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

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

- Die Werte $\alpha_{0,n}, \dots, \alpha_{n,n}$ heißen *Gewichte*.
- Diese sind unabhängig von f und $[a, b]$ und somit tabellierbar.
- Es gilt immer:

$$h \sum_{k=0}^n \alpha_{k,n} = b - a, \quad \text{also} \quad \sum_{k=0}^n \alpha_{k,n} = n$$

- Die geschlossene Newton-Cotes-Formel $I(f)$ ist exakt vom Grad n .

Fehlerabschätzung

Für ein $\xi \in [a, b]$ gilt für den Fehler $E_n(f) := I(f) - I_n(f)$:

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

3.1.1 Spezielle geschlossene Newton-Cotes-Formeln

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

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

3.2.1 Spezielle offene Newton-Cotes-Formeln

n	h	$\tilde{\alpha}_{k,n}$		max. Fehler $\tilde{E}_n(f)$	Name
0				$\frac{b-a}{2} \frac{f^{(2)}(\xi)}{3} h^3$	Rechteck-Regel
1	$b-a$	$-\frac{f^{(2)}(\xi)}{12} h^3$	Trapezregel	$\frac{b-a}{3} \frac{3f^{(2)}(\xi)}{4} h^3$	
2	$\frac{b-a}{2}$	$-\frac{f^{(4)}(\xi)}{90} h^5$	Simpson-Regel	$\frac{b-a}{4} \frac{28f^{(4)}(\xi)}{90} h^5$	
3	$\frac{b-a}{3}$	$-\frac{3f^{(4)}(\xi)}{80} h^5$	3/8-Regel		
4	$\frac{b-a}{4}$	$-\frac{8f^{(6)}(\xi)}{945} h^7$	Milne-Regel		

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

n	geschlossen			offen		
	h	$\tilde{E}_n(f)$	Name	h	$E_n(f)$	Name
0				$\frac{b-a}{2}$	$\frac{f^{(2)}(\xi)}{3} h^3$	Rechteck-Regel
1	$b-a$	$-\frac{f^{(2)}(\xi)}{12} h^3$	Trapezregel	$\frac{b-a}{3}$	$\frac{3f^{(2)}(\xi)}{4} h^3$	
2	$\frac{b-a}{2}$	$-\frac{f^{(4)}(\xi)}{90} h^5$	Simpson-Regel	$\frac{b-a}{4}$	$\frac{28f^{(4)}(\xi)}{90} h^5$	
3	$\frac{b-a}{3}$	$-\frac{3f^{(4)}(\xi)}{80} h^5$	3/8-Regel			
4	$\frac{b-a}{4}$	$-\frac{8f^{(6)}(\xi)}{945} h^7$	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 $[a, b]$ in m Teilintervalle der Länge $H = \frac{b-a}{m}$:

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

- 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.

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

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)

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

Fehler: $R_N^{(1)}(f) = -\frac{f''(\xi)}{12}(b-a)h^2$

3.4.2 Summierte Simpson-Regel (geschlossen)

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

Fehler: $R_N^{(2)}(f) = -\frac{f^{(4)}(\xi)}{180}(b-a)h^4$

3.4.3 Summierte Rechteck-Regel (offen)

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

Fehler: $\tilde{R}_N^{(0)}(f) = \frac{f''(\xi)}{6}(b-a)h^2$

4 Gewöhnlichen Differentialgleichungen

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

Gesucht Eine Funktion $y : [a, b] \rightarrow \mathbb{R}^n$, 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.

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

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

4.1 Existenz- und Eindeutigkeit

Sei $f : [a, b] \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ 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/Lindelöf).
- 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 AWP's hängt stetig vom Anfangswert ab.

4.2 Numerische Verfahren

Grundidee: Zerlege das Intervall $[a, b]$ in Teilintervalle $t_j = a + jh$, $j = 0, 1, \dots, N$, $h = \frac{b-a}{N}$ und approximiere das Integral des AWP's 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

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

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:

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

4.2.2 Implizites Euler-Verfahren

Verfahrensfunktion:

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

Verfahren:

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

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

Verfahren:

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

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

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

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

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

4.2.6 Explizite Runge-Kutta-Verfahren und Butcher-Schema

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

$$k_i(t, u, h) = k_i := f\left(t + \gamma_i h, u + h \sum_{j=1}^{i-1} \alpha_{i,j} k_j\right), \quad i = 1, \dots, r$$

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

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	$\alpha_{r,1}$	\dots	\dots	$\alpha_{r,r-1}$	0	
	β_1	β_2	\dots	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	<table style="border-collapse: collapse; margin: 0 auto;"> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">1</td> <td style="padding: 5px;">1</td> </tr> </table>	0	0	1	1																					
0	0																									
1	1																									
Modifiziertes Euler-Verfahren	<table style="border-collapse: collapse; margin: 0 auto;"> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td></td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">$\frac{1}{2}$</td> <td style="padding: 5px;">$\frac{1}{2}$</td> <td style="padding: 5px;">0</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">1</td> </tr> </table>	0	0		$\frac{1}{2}$	$\frac{1}{2}$	0	0	0	1																
0	0																									
$\frac{1}{2}$	$\frac{1}{2}$	0																								
0	0	1																								
Verfahren von Heun	<table style="border-collapse: collapse; margin: 0 auto;"> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td></td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">1</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">0</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;"></td> <td style="padding: 5px;">$\frac{1}{2}$</td> <td style="padding: 5px;">$\frac{1}{2}$</td> </tr> </table>	0	0		1	1	0		$\frac{1}{2}$	$\frac{1}{2}$																
0	0																									
1	1	0																								
	$\frac{1}{2}$	$\frac{1}{2}$																								
Klassisches RK-Verfahren 4. Ordnung (RK4)	<table style="border-collapse: collapse; margin: 0 auto;"> <tr> <td style="border-right: 1px solid black; padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td></td> <td></td> <td></td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">$\frac{1}{2}$</td> <td style="padding: 5px;">$\frac{1}{2}$</td> <td style="padding: 5px;">0</td> <td></td> <td></td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">$\frac{1}{2}$</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">$\frac{1}{2}$</td> <td style="padding: 5px;">0</td> <td></td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;">1</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">0</td> <td style="padding: 5px;">1</td> <td style="padding: 5px;">0</td> </tr> <tr> <td style="border-right: 1px solid black; padding: 5px;"></td> <td style="padding: 5px;">$\frac{1}{6}$</td> <td style="padding: 5px;">$\frac{1}{3}$</td> <td style="padding: 5px;">$\frac{1}{3}$</td> <td style="padding: 5px;">$\frac{1}{6}$</td> </tr> </table>	0	0				$\frac{1}{2}$	$\frac{1}{2}$	0			$\frac{1}{2}$	0	$\frac{1}{2}$	0		1	0	0	1	0		$\frac{1}{6}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{6}$
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:

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

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

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

$$\begin{array}{c|cccccc}
 \gamma_1 & \alpha_{1,1} & \cdots & \cdots & \alpha_{1,r-1} & \alpha_{1,r} \\
 \gamma_2 & \alpha_{2,1} & \cdots & \cdots & \alpha_{2,r-1} & \alpha_{2,r} \\
 \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
 \gamma_r & \alpha_{r,1} & \cdots & \cdots & \alpha_{r,r-1} & \alpha_{r,r} \\
 \hline
 & \beta_1 & \beta_2 & \cdots & \beta_{r-1} & \beta_r
 \end{array}$$

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

4.3 Konvergenz und Konsistenz

Sei Φ eine Verfahrensfunktion. Dann heißt die Größe

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

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**

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

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:

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

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):

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

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

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 $c \in \mathbb{R}^n$):

$$\begin{aligned} y'(t) &= Ay(t) + c, \quad t \in [a, b] \\ y(a) &= y_0 \end{aligned}$$

Sei ferner 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 $\Re(\lambda_i) < 0$ für $i = 1, \dots, n$, so gilt aufgrund von $|e^{\lambda_i t}| = e^{\Re(\lambda_i)t}$:

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

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

- Dabei klingen Summanden in y_H mit $\Re(\lambda_i) \ll -1$ sehr schnell und Summanden mit $\Re(\lambda_i) \not\ll -1$ deutlich langsamer ab.
- Existieren Eigenwerte mit $\Re(\lambda_i) \ll -1$ 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 $z' = \text{diag}(\lambda_1, \dots, \lambda_n)z$ zuverlässig, dann liefert es auch für das steife System $y' = Ay$, $y(0) = y_0$ gute Ergebnisse.

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

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

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

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

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

soll die Eigenschaften der analytischen Lösung $y = e^{\lambda t}$, insbesondere $\lim_{t \rightarrow \infty} y(t) = 0$ 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 $R : D \rightarrow \mathbb{C}$, $0 \in D \subseteq \mathbb{C}$. Diese Funktion wird *Stabilitätsfunktion* genannt.

Die Menge

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

heißt dann *Stabilitätsgebiet*.

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

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

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:

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

Anders ausgedrückt: Die produzierte folge muss monoton fallend sein.

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

L-Stabilität

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

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

5 Lineare Gleichungssysteme

5.1 Lösungstheorie

5.2 Gaußsches Eliminationsverfahren, Dreieckszerlegung

5.2.1 Lösung gestaffelter Gleichungssysteme

5.2.2 Gaußsches Eliminationsverfahren

Grundkonzept

Pivotstrategie

Spaltenpivotsuche

Vollständige Pivotsuche

Algorithmen

5.2.3 LR-Zerlegung

Vollständige Pivotsuche

Content

5.2.4 Matrixdarstellung

Content

5.2.5 Matrixklassen ohne Pivotsuche

Content

5.3 Cholsky-Verfahren

Content

5.3.1 Cholsky-Verfahren \leftrightarrow LR-Zerlegung

Content

5.3.2 Verfahren

Content

5.3.3 Eigenschaften

Content

5.4 Fehlerabschätzungen und Rundungsfehlereinfluss

Content

5.4.1 Fehlerabschätzungen für gestörte Gleichungssysteme

Content

5.4.2 Rundungsfehleranalyse

Content

Spaltenpivotsuche

Content

Vollständige Pivotsuche

Content

6 Nichtlineare Gleichungssysteme

Gesucht ist eine Lösung $x \in D$ von

$$F(x) = 0$$

mit einer gegebenen Abbildung

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

wobei $D \subseteq \mathbb{R}^n$ 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 $F : \mathbb{R}^n \rightarrow \mathbb{R}^n$.

6.1.1 Lokales Newton-Verfahren

```
1 Wähle einen Startpunkt  $x^{(0)} \in \mathbb{R}^n$ 
2 for  $k = 0, 1, \dots$  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  $x^{(k+1)} \leftarrow x^{(k)} + s^{(k)}$ 
8   end
9 end
```

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

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

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 $\sigma_k \in \{1, \frac{1}{2}, \frac{1}{4}, \dots\} = \{\frac{1}{2^k} \mid k \in \mathbb{N}\}$ gewählt, sodass gilt:

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

Globalisiertes Newton-Verfahren

```
1 Wähle einen Startpunkt  $x^{(0)} \in \mathbb{R}^n$ 
2 for  $k = 0, 1, \dots$  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  $\sigma_k$  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

7.2 Gershgorin-Kreise

7.3 Numerische Verfahren

7.3.1 Vektoriteration

Konvergenz

Vektoriteration von Mises

Vektoriteration von Wiedlandt

Inverse Vektoriteration von Wiedlandt

7.3.2 QR-Verfahren

Grundlegende Eigenschaften

Content



Konvergenz

 Content

Shift-Techniken

 Content

Verbreitete Shift-Strategie

 Content

Berechnung der Eigenvektoren

 Content

Householder-Verfahren zur Berechnung

 Content