

Einführung in die Numerik WS2018/19

Dozent: Prof. Dr. ANDREAS FISCHER

18. November 2018

Inhaltsverzeichnis

I	Interpolation	2
1	Grundlagen	2
2	Interpolation durch Polynome	4
2.1	Existenz und Eindeutigkeit	4
2.2	NEWTON-Form des Interpolationspolynoms	5
2.3	Interpolationsfehler	6
3	Interpolation durch Polynomsplines	9
3.1	Polynomsplines	9
3.2	Interpolation durch kubische Polynomsplines	9
3.3	Interpolation mit kubischen C^2 -Splines	10
3.4	Eine Minimaleigenschaft kubischer C^2 -Interpolationssplines	13
3.5	Interpolationsfehler bei kubischer C^2 -Interpolation	14
II	numerische Integration (Quadratur)	15
1	Integration von Interpolationspolynomen	15
2	NEWTON-COTES-Formeln	16
3	spezielle NEWTON-COTES-Formeln	17
4	Zusammengesetzte NEWTON-COTES-Formeln	20
5	GAUSS'sche Quadraturformeln	21
III	direkte Verfahren für lineare Gleichungssysteme	23
1	GAUSS'scher Algorithmus für quadratische Systeme	23
1.1	Grundform des GAUSS'schen Algorithmus	23
1.2	Pivotisierung	26
1.3	LU-Faktorisierung	26
1.4	GAUSS'scher Algorithmus für trigonale Systeme	29
2	CHOLESKY-Faktorisierung für symmetrische positiv definite Matrizen	31
2.1	Existenz der CHOLESKY-Faktorisierung	31
2.2	Berechnung des CHOLESKY-Faktors	32
3	Lineare Quadratmittelprobleme	34
4	Kondition linearer Gleichungssysteme	35
IV	Kondition von Aufgaben und Stabilität von Algorithmen	36
1	Maschinenzahlen und Rundungsfehler	36
2	Fehleranalyse	37
V	Newton-Verfahren zur Lösung nichtlinearer Gleichungssysteme	38
1	Das NEWTON-Verfahren	38
2	Gedämpftes NEWTON-Verfahren	39

VI	lineare Optimierung	40
1	Ecken und ihre Charakterisierung	40
2	Simplex-Verfahren	41
3	Die Tableauform des Simplex-Verfahrens	42
4	Revidiertes Simplex-Verfahren	43
5	Bestimmung einer ersten zulässigen Basislösung	44
 Anhang		46
A	Listen	46
A.1	Liste der Theoreme	46
A.2	Liste der benannten Sätze, Lemmata und Folgerungen	47
 Index		48

Vorwort

Kapitel I

Interpolation

1. Grundlagen

Aufgabe:

Gegeben sind $n + 1$ Datenpaare $(x_0, f_0), \dots, (x_n, f_n)$, alles reelle Zahlen und paarweise verschieden.

Gesucht ist eine Funktion $F : \mathbb{R} \rightarrow \mathbb{R}$, die die Interpolationsbedingungen

$$F(x_0) = f_0, \dots, F(x_n) = f_n \quad (1)$$

genügt.

Definition (Stützstellen, Stützwerte)

Die x_0 bis x_n werden Stützstellen genannt.

Die f_0 bis f_n werden Stützwerte genannt.

Die oben gestellte Aufgabe wird zum Beispiel durch

$$F(x) = \begin{cases} 0 & x \notin \{x_0, \dots, x_n\} \\ f_i & x = x_i \end{cases}$$

gelöst. Weitere Möglichkeiten sind: Polygonzug, Treppenfunktion, Polynom, ...

- In welcher Menge von Funktionen soll F liegen?
- Gibt es im gewählten Funktionsraum für beliebige Datenpaare eine Funktion F , die den Interpolationsbedingungen genügt (eine solche Funktion heißt Interpolierende)?
- Ist die Interpolierende in diesem Raum eindeutig bestimmt?
- Welche weiteren Eigenschaften besitzt die Interpolierende, zum Beispiel hinsichtlich ihrer Krümmung oder der Approximation einer Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$ mit $f_k = f(x_k)$ für $k = 0, \dots, n$?
- Wie sollte man die Stützstellen wählen, falls nicht vorgegeben?
- Wie lässt sich die Interpolierende effizient bestimmen, gegebenenfalls auch unter der Berücksichtigung, dass neue Datenpaare hinzukommen oder dass sich nur die Stützwerte ändern?

■ Beispiel 1.1

k	0	1	2	3	4	5
x_k in s	0	1	2	3	4	5
f_k in °C	80	85,8	86,4	93,6	98,3	99,1

Interpolation im

- Raum der stetigen stückweise affinen Funktionen
- Raum der Polynome höchstens 5. Grades
- Raum der Polynome höchstens 4. Grades (Interpolation im Allgemeinen nicht lösbar, Regression nötig)

2. Interpolation durch Polynome

Π_n bezeichne den Vektorraum der Polynome von Höchstgrad n mit der üblichen Addition und Skalarmultiplikation. Für jedes $p \in \Pi_n$ gibt es $a_0, \dots, a_n \in \mathbb{R}$, sodass

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0 \quad (1)$$

und umgekehrt.

2.1. Existenz und Eindeutigkeit

Satz 2.1

Zu $n+1$ Datenpaaren $(x_0, f_0), \dots, (x_n, f_n)$ mit paarweise verschiedenen Stützstellen existiert genau ein Polynom $p \in \Pi_n$, dass die Interpolationsbedingung Gleichung (1) erfüllt.

Beweis. • Existenz: Sei $j \in \{0, \dots, n\}$ und $L_j : \mathbb{R} \rightarrow \mathbb{R}$ mit

$$L_j(x) := \prod_{\substack{i=0 \\ i \neq j}}^n \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n)}{(x_j - x_0) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_n)}$$

das LAGRANGE-Basispolynom vom Grad n . Offenbar gilt $L_j \in \Pi_n$ und

$$L_j(x_k) = \begin{cases} 1 & k = j \\ 0 & k \neq j \end{cases} = \delta_{jk} \quad (2)$$

Definiert man $p : \mathbb{R} \rightarrow \mathbb{R}$ durch

$$p(x) := \sum_{j=0}^n f_j \cdot L_j(x) \quad (3)$$

so ist $p \in \Pi_n$ und außerdem erfüllt p wegen Gleichung (2) die Interpolationsbedingung Gleichung (1)

- Eindeutigkeit: Angenommen es gibt Interpolierende $p, \tilde{p} \in \Pi_n$ mit $p \neq \tilde{p}$. Dann folgt $p - \tilde{p} \in \Pi_n$ und $(p - \tilde{p})(x_k) = p(x_k) - \tilde{p}(x_k) = 0$ für $k = 0, \dots, n$. Also hat $(p - \tilde{p})$ mindestens $n + 1$ Nullstellen, hat aber Grad n . Das heißt, dass $(p - \tilde{p})$ das Nullpolynom sein muss. \square

Definition (Interpolationspolynom)

Das Polynom, dass die Interpolationsbedingung erfüllt, heißt Interpolationspolynom zu $(x_0, f_0), \dots, (x_n, f_n)$.

► Bemerkung 2.2

- Die Darstellung Gleichung (3) heißt LAGRANGE-Form des Interpolationspolynoms.
- Um mittels Gleichung (3) einen Funktionswert $p(x)$ zu berechnen, werden $\mathcal{O}(n^2)$ Operationen benötigt; bei gleichabständigen Stützstellen kann man diesen Aufwand auf $\mathcal{O}(n)$ verringern. Ändern sich die Stützwerte, kann man durch Wiederverwendung von den $L_j(x)$ das $p(x)$ in $\mathcal{O}(n)$ Operationen berechnen.
- Man kann zeigen, dass L_0 bis L_n eine Basis von Π_n bilden.

2.2. Newton-Form des Interpolationspolynoms

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \cdots + c_n(x - x_0) \cdots (x - x_{n-1}) \quad (4)$$

mit Koeffizienten $c_0, \dots, c_n \in \mathbb{R}$. Die Berechnung des Koeffizienten c_j kann rekursiv durch Ausnutzen der Interpolationsbedingung Gleichung (1) erfolgen. Für c_0 erhält man

$$f_0 \stackrel{!}{=} p(x_0) = c_0$$

Seien c_0 bis c_{j-1} bereits ermittelt. Dann folgt:

$$f_j \stackrel{!}{=} p(x_j) = c_0 + \underbrace{\sum_{k=1}^{j-1} c_k(x_j - x_0) \cdots (x_j - x_{k-1})}_{\text{bekannt}} + c_j \underbrace{(x_j - x_0) \cdots (x_j - x_{j-1})}_{\text{bekannt}}$$

► Bemerkung 2.3

- Der Aufwand um die Koeffizienten c_0, \dots, c_n zu ermitteln ist $\mathcal{O}(n^2)$. Kommt ein Datenpaar hinzu, kann man Gleichung (4) um einen Summanden erweitern und mit $\mathcal{O}(n)$ Operationen c_{n+1} bestimmen.
- Sind die Koeffizienten c_0, \dots, c_n in Gleichung (4) bekannt, dann benötigt man zur Berechnung von $p(x)$ $\mathcal{O}(n)$ Operationen.
- Die Polynome $N_0, \dots, N_n : \mathbb{R} \rightarrow \mathbb{R}$ mit

$$N_0 = 1 \quad \text{und} \quad N_j = (x - x_0) \cdots (x - x_{j-1})$$

heißen NEWTON-Basispolynome und bilden eine Basis von Π_n .

Die Koeffizienten c_0, \dots, c_n ergeben sich wegen Gleichung (1) auch als Lösung des folgenden linearen Gleichungssystems:

$$\begin{pmatrix} 1 & & & & \\ 1 & (x_1 - x_0) & & & \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & & \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & (x_n - x_0) & (x_n - x_0)(x_n - x_1) & \cdots & \prod_{i=0}^{n-1} (x_n - x_i) \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}$$

Die Systemmatrix dieses linearen Gleichungssystems ist eine reguläre untere Dreiecksmatrix.

Zu effizienten Berechnung eines Funktionswertes $p(x)$ nach Gleichung (4) mit gegebenen Koeffizienten

c_0, \dots, c_n kann man das HORNER-Schema anwenden. Überlegung für $n = 3$.

$$\begin{aligned} p(x) &= c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)(x - x_2) \\ &= c_0 + (x - x_0) \left[c_1 + (x - x_1) [c_2 + (x - x_2)c_3] \right] \end{aligned}$$

Für beliebiges n liefert das den folgenden Algorithmus:

■ **Algorithmus 2.4 (Horner-Schema für Newton-Form)**

Input: $n, x, c_0, \dots, c_n, x_0, \dots, x_n$

```

1  p = c_n
2  do j = n-1, 0, -1
3    p = c_j + (x - x_j) p
4  end do

```

2.3. Interpolationsfehler

Definition (Maximum-Norm)

Die Norm

$$\|g\|_\infty := \max_{x \in [a, b]} |g(x)| \quad \text{für } g \in C[a, b]$$

definiert die Maximum-Norm in $C[a, b]$.

Satz 2.5

Sei $f \in C[a, b]$. Dann existiert zu jedem $\varepsilon > 0$ ein Polynom p_ε mit $\|f - p_\varepsilon\| \leq \varepsilon$.

Also liegt die Menge aller Polynome (beliebig hohen Grades) direkt in $C[a, b]$.

Definition 2.6 (Stützstellensystem)

Stützstellensystem : $a \leq x_0^{(n)} < \dots < x_n^{(n)} \leq b$. Weiterhin bezeichne $p_n \in \Pi_n$ das zu den Datenpaaren $(x_k^{(n)}, f(x_k^{(n)}))$ gehörende eindeutig bestimmte Interpolationspolynom.

Satz 2.7 (Satz von Faber 1914)

Zu jedem Stützstellensystem gibt es $f \in C[a, b]$, sodass (p_n) nicht gleichmäßig gegen f konvergiert. $\|p_n - f\|_\infty \rightarrow 0$ bedeutet, dass (p_n) gleichmäßig gegen f konvergiert.

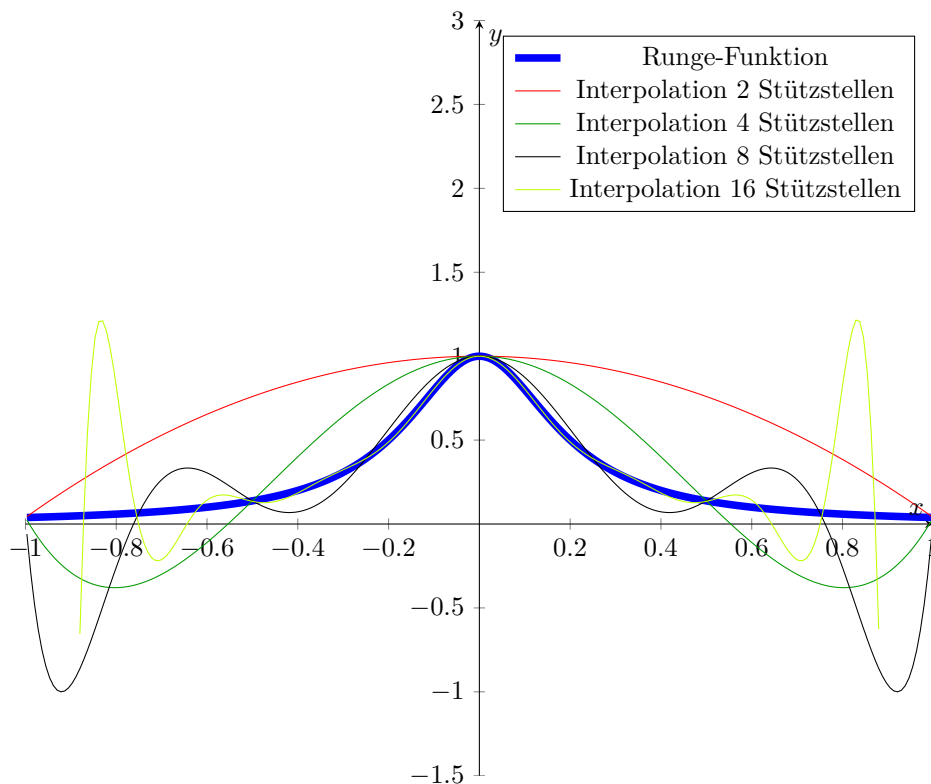
Nach einem Resultat von ERDÖS/VERTESI (1980) gilt sogar, dass $(p_n(x))$ fast überall divergiert.

■ **Beispiel 2.8 (Runge)**

$$f : \mathbb{R} \rightarrow \mathbb{R}, f(x) = \frac{1}{1+25x^2}$$

äquidistante Stützstellen $x_0, \dots, x_n, p \in \Pi_n$ als Interpolationspolynom

Stützstellen	interpoliertes Polynom
2	$1 - \frac{25x^2}{26}$
4	$3,31565x^4 - 4,27719x^2 + 1$
8	$53,6893x^8 - 102,815x^6 + 61,3672x^4 - 13,203x^2 + 1$
16	$15403,1x^{16} - 49713,5x^{14} + 63743,8x^{12} - 41870x^{10} + 15206x^8 - 3100,35x^6 + 351,984x^4 - 22,7759x^2 + 1$



Anmerkung

Wer mit Mathematica selber diese Polynome berechnen will, muss folgende Befehle benutzen:

- Funktion definieren: `f[x_]:=1/(1+25x^2)`
- Interpolationspolynome ausrechnen: `Expand[InterpolatingPolynomial[Table[{i,f[i]},{i,-1,-1,Schrittweite}],{x}]]`
- plotten: `Plot[f[x],InterpolatingPolynomial[Table[{i,f[i]},{i,-1,-1,Schrittweite}],{x}],{x,-1,1}]`

Satz 2.9

Sei $f \in C^{n+1}[a, b]$ und gelte $a \leq x_0 < \dots < x_n \leq b$. Mit $p_n \in \Pi_n$ werde das zu den Datenpaaren $(x_0, f(x_0)), \dots, (x_n, f(x_n))$ gehörende Interpolationspolynom bezeichnet. Dann existiert zu jedem $x \in [a, b]$ eine Zahl $\xi \in (a, b)$, so dass

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} w(x) \quad \text{für alle } x \in [a, b]$$

wobei $w(x) = (x - x_0) \cdot \dots \cdot (x - x_n)$

Beweis. Für $x = x_k$ mit $k = 0, \dots, n$ ist nicht zu zeigen, da p_n die Interpolationsbedingung erfüllt. Sei nun $x \in [a, b]$ fest gewählt mit $x \notin \{x_0, \dots, x_n\}$. Weiter seien

$$K = \frac{f(x) - p_n(x)}{w(x)} \quad \text{und} \quad F : \begin{cases} [a, b] \rightarrow \mathbb{R} \\ t \mapsto f(t) - p_n(t) - Kw(t) \end{cases}$$

Man stellt unter Beachtung der Interpolationsbedingung fest, dass $F(x_0) = F(x_1) = \dots = F(x_n) = 0$ und $F(x) = 0$. Also besitzt F mindestens $n + 2$ paarweise verschiedene Nullstellen in $[a, b]$. Da $F \in C^{n+1}[a, b]$ erhält man durch $n + 1$ -fache Anwendung des Satzes von Rolle, dass $F^{(n+1)}$ mindestens eine Nullstelle $\xi(x)$ in (a, b) besitzt. Also folgt

$$0 = F^{(n+1)}(\xi(x)) = f^{(n+1)}(\xi(x)) - \underbrace{p_n^{(n+1)}(\xi(x))}_{=0} - \underbrace{K w^{(n+1)}(\xi(x))}_{\text{Konstante}}$$

Da $w^{(n+1)} = (n + 1)!$, erhält man

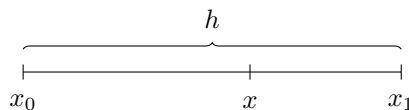
$$K = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}$$

Da $x \in [a, b]$ beliebig gewählt war, ist die Behauptung bewiesen. □

■ **Beispiel 2.10**

Sei $f \in C^2[a, b]$ mit $\|f\|_\infty \leq M$. Weiter sei $a = x_0 < x_1 = x_0 + h = b$. Mit Satz 2.9 folgt:

$$\begin{aligned} |f(x) - p_2(x)| &= \left| \frac{f''(\xi(x))}{2} (x - x_0)(x - x_1) \right| \\ &\leq \frac{1}{2} M \cdot \lambda(x) h \cdot (1 - \lambda(x)) h \\ &\leq \frac{1}{2} M \cdot h^2 \underbrace{\lambda(x)(1 - \lambda(x))}_{\leq 1/4} \\ &\leq \frac{1}{8} M \cdot h^2 \end{aligned}$$



$$\Rightarrow x = x_0 + \lambda \cdot (x_1 - x_0) = \lambda x_1 + (1 - \lambda)x_0$$

3. Interpolation durch Polynomsplines

3.1. Polynomsplines

Zur Abkürzung bezeichne Δ eine Zerlegung des Intervall $[a, b]$ durch die Stützstellen $a =: x_0 < \dots < x_n := b$.

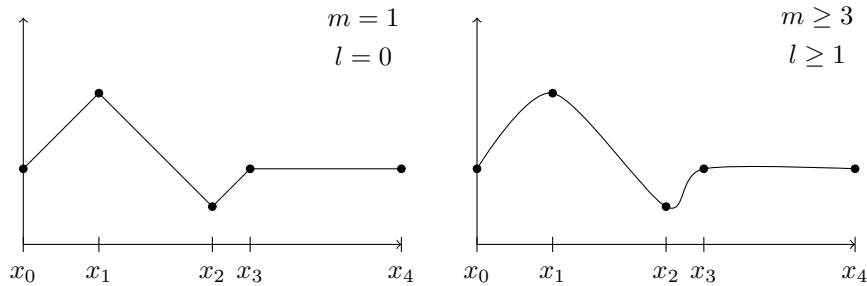
Definition 3.1 (Polynomspline)

Ein Polynomspline vom Grad $m \in \mathbb{N}$ und Glattheit $l \in \mathbb{N}$ zur Zerlegung Δ ist eine Funktion $s \in C^l[a, b]$ mit

$$s_k := s|_{[x_k, x_{k+1}]} \in \Pi_m \quad \text{für } k = 0, \dots, n-1$$

Dabei bezeichnet $s|_{[x_k, x_{k+1}]}$ die Einschränkung von s auf das Intervall $[x_k, x_{k+1}]$. Die Menge aller Splines wird mit $\mathcal{S}_m^l(\Delta)$ bezeichnet.

Folglich ist ein Polynomspline $s \in \mathcal{S}_m^l(\Delta)$ auf jedem der Teilintervall $[x_k, x_{k+1}]$ ein Polynom vom Höchstgrad m . Außerdem ist $s \in \mathcal{S}_m^l(\Delta)$ in allen Punkten $x \in [a, b]$ (also auch in den Stützstellen) l -mal stetig differenzierbar. $\mathcal{S}_m^l(\Delta)$ ist mit der üblichen Addition und Multiplikation ein Vektorraum. Speziell ist $\mathcal{S}_1^0(\Delta)$ die Menge aller stetigen stückweise affin linearen Funktionen.



3.2. Interpolation durch kubische Polynomsplines

Gegeben sei eine Zerlegung Δ und die Stützwerte f_0, \dots, f_n . Gesucht ist eine Funktion $s \in \mathcal{S}_3^l(\Delta)$ mit $l = 1, 2$ derart, dass

$$s(x_k) = f_k \quad \text{für } k = 0, \dots, n \tag{1}$$

Jede derartige Funktion heißt kubischer Interpolationspline.

Konstruktion eines solchen Splines:

$$\begin{aligned} h_k &:= x_{k+1} - x_k \quad \text{für } k = 0, \dots, n-1 \\ m_k &:= s'(x_k) \quad \text{für } k = 0, \dots, n-1 \end{aligned}$$

Wegen $l \in \{1, 2\}$ ist s zunächst stetig differenzierbar. Wegen $s_k = s|_{[x_k, x_{k+1}]}$ für $k = 0, \dots, n-1$ und $m = 3$ kann man folgenden Ansatz für s_k benutzen:

$$s_k(x) = a_k(x - x_k)^3 + b_k(x - x_k)^2 + c_k(x - x_k) + d_k \tag{2}$$

Aus den Interpolationsbedingungen Gleichung (1) und der stetigen Differenzierbarkeit aller Funktionen in $s \in \mathcal{S}_m^l(\Delta)$ für $l \geq 1$ ergeben sich folgende Forderungen an s_k , $k = 0, \dots, n-1$:

$$\begin{aligned} s_k(x_k) &= f_k & \text{und} & & s_k(x_{k+1}) &= f_{k+1} \\ s'_k(x_k) &= m_k & \text{und} & & s'_k(x_{k+1}) &= m_{k+1} \end{aligned} \quad (3)$$

Diese liefern:

$$\begin{aligned} d_k &= s_k(x_k) = f_k \\ c_k &= s'_k(x_k) = m_k \end{aligned} \quad (4)$$

und damit:

$$\begin{aligned} s_k(x_{k+1}) &= a_k h_k^3 + b_k h_k^2 + m_k h_k + f_k = f_{k+1} \\ s'_k(x_{k+1}) &= 3a_k h_k^2 + 2b_k h_k + m_k = m_{k+1} \end{aligned}$$

Damit ergeben sich a_k und b_k als eindeutige Lösung für das lineare Gleichungssystem

$$\begin{pmatrix} h_k^3 & h_k^2 \\ 3h_k^2 & 2h_k \end{pmatrix} \begin{pmatrix} a_k \\ b_k \end{pmatrix} = \begin{pmatrix} f_{k+1} - f_k - m_k h_k \\ m_{k+1} - m_k \end{pmatrix} \quad (5)$$

Die Determinante ist $-h_k^4 \neq 0$.

Satz 3.2

Sei eine Zerlegung Δ des Intervalls $[a, b]$ gegeben. Dann gibt es für beliebig gewählte reelle Zahlen f_0, \dots, f_n und m_0, \dots, m_n einen Interpolationsspline $s \in \mathcal{S}_3^1(\Delta)$, der den Interpolationsbedingungen

$$s'(x_0) = m_0, \dots, s'(x_n) = m_n$$

genügt. Außerdem gilt: $s|_{[x_k, x_{k+1}]} = s_k$ für $k=0, \dots, n-1$ mit s_k entsprechend Gleichung (2), wobei sich a_k, b_k, c_k, d_k aus Gleichung (4) und Gleichung (5) ergeben.

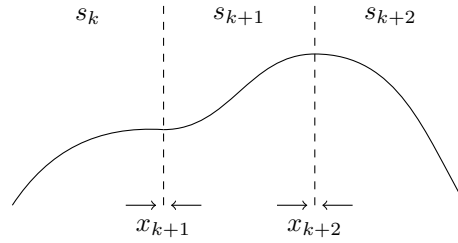
Für die Wahl der m_k gibt es verschiedene Möglichkeiten, zum Beispiel:

- Falls Ableitungswerte der zu interpolierenden Funktion f bekannt sind, kann man $m_k = f'(x_k)$ setzen.
- Man wählt m_0, \dots, m_n so, dass s zweimal stetig differenzierbar ist, das heißt $s \in \mathcal{S}_3^2(\Delta)$ statt $s \in \mathcal{S}_3^1(\Delta)$ gilt.

3.3. Interpolation mit kubischen C^2 -Splines

Damit ein kubischer Interpolationsspline s zu $\mathcal{S}_3^2(\Delta)$ gehört, muss neben den Forderungen in Gleichung (3) die Stetigkeit von s'' an den Stützstellen x_1, \dots, x_{n-1} gewährleistet sein. Also hat man zusätzliche Bedingungen

$$s''_k(x_{k+1}) = s''_{k+1}(x_{k+1}) \quad \text{für } k = 0, \dots, n-2$$



Mit Gleichung (2) ergibt sich $s''(x) = 6a_k(x - x_0) + 2b_k$ für $x \in [x_k, x_{k+1}]$ und damit $s''_k(x_{k+1}) = 6a_k h_k + 2b_k$ und $s''_{k+1}(x_{k+1}) = 2b_{k+1}$, also

$$3a_k h_k + b_k = b_{k+1} \quad \text{für } k = 0, \dots, n-2 \quad (6)$$

Aus Gleichung (5) folgt

$$a_k = \frac{-2}{h_k^3}(f_{k+1} - f_k) + \frac{1}{h_k^2}(m_k + m_{k+1})$$

$$b_k = \frac{3}{h_k^2}(f_{k+1} - f_k) - \frac{1}{h_k}(2m_k + m_{k+1})$$

für $k = 0, \dots, n-1$. Wegen Gleichung (6) erhält man für $k = 0, \dots, n-2$

$$\begin{aligned} & \frac{-6}{h_k^2}(f_{k+1} - f_k) + \frac{3}{h_k}(m_k + m_{k+1}) + \frac{3}{h_k^2}(f_{k+1} - f_k) - \frac{1}{h_k}(2m_k + m_{k+1}) \\ &= \frac{-6}{h_{k+1}^2}(f_{k+2} - f_{k+1}) - \frac{1}{h_{k+1}}(2m_{k+1} + m_{k+2}) \end{aligned}$$

Damit folgt

$$\frac{1}{h_k}(m_k + 2m_{k+1}) + \frac{1}{h_{k+1}}(2m_{k+1} + m_{k+2}) = \frac{3}{h_k^2}(f_{k+1} - f_k) + \frac{3}{h_{k+1}^2}(f_{k+2} - f_{k+1})$$

bzw. $h_{k+1}m_k + 2(h_{k+1} + h_k)m_{k+1} + h_k m_{k+2} = \frac{3h_{k+1}}{h_k}(f_{k+1} - f_k) + \frac{3h_k}{h_{k+1}}(f_{k+2} - f_{k+1})$

Also müssen die $n+1$ Zahlen m_0, \dots, m_n den $n-1$ Gleichungen des linearen Gleichungssystems

$$\begin{pmatrix} \lambda_0 & 2 & \mu_0 & & & \\ & \lambda_1 & 2 & \mu_1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \lambda_{n-2} & 2 & \mu_{n-2} \end{pmatrix} \begin{pmatrix} m_0 \\ m_1 \\ \vdots \\ m_n \end{pmatrix} = \begin{pmatrix} r_0 \\ r_1 \\ \vdots \\ r_{n-2} \end{pmatrix}$$

genügen, wobei λ_k, μ_k, r_k durch

$$\lambda_k = \frac{h_{k+1}}{h_k + h_{k+1}}$$

$$\mu_k = \frac{h_k}{h_k + h_{k+1}}$$

$$r_k = \frac{3h_{k+1}}{h_k(h_k + h_{k+1})}(f_{k+1} - f_k) + \frac{3h_k}{h_{k+1}(h_k + h_{k+1})}(f_{k+2} - f_{k+1})$$

für $k = 0, \dots, n - 2$ gegeben sind. Die Systemmatrix und die erweiterte Systemmatrix haben den Rang $n - 1$. Somit ist das Gleichungssystem lösbar, besitzt aber keine eindeutige Lösung. Um solche zu erhalten, kann man zusätzliche Bedingungen stellen, etwa

(a) **natürliche Randbedingungen:**

$$s''(x_0) = s''(x_n) = 0 \quad (7)$$

Diese sind gleichbedeutend mit

$$s_0''(x_0) = 6a_0(x - x_0) + 2b_0 = 0 \quad \text{und} \quad s_{n-1}''(x_n) = 6a_{n-1}(x_n - x_{n-1}) + 2b_{n-1} = 0$$

Also folgt

$$b_0 = 0 \quad \text{und} \quad 3a_{n-1}h_{n-1} + b_{n-1} = 0$$

Nutzt man noch die Darstellung für b_0 sowie für a_{n-1} und b_{n-1} , so folgt

$$2m_0 + m_1 = \frac{3}{h_0}(f_1 - f_0) \quad \text{und} \quad m_{n-1} + 2m_n = \frac{3}{h_{n-1}}(f_n - f_{n-1})$$

Fügt man beide Gleichungen geeignet zum obigen System hinzu, erhält man ein lineares Gleichungssystem mit einer regulären trigonalen Systemmatrix. Dieses kann in $\mathcal{O}(n)$ Operationen gelöst werden.

(b) **Vollständige Randbedingungen:** Sind $f'(a)$ und $f'(b)$ bekannt, dann können die zusätzlichen Bedingungen

$$s'(x_0) = f'(a) \quad \text{und} \quad s'(x_n) = f'(b) \quad (8)$$

mittels $m_0 = f'(a)$ und $m_n = f'(b)$ geeignet in das Gleichungssystem eingefügt werden, so dass man analog zu Fall (a) eine trigonale reguläre Systemmatrix erhält.

(c) **Periodische Spline-Interpolation:** Falls

$$f'(a) = f'(b) \quad (9)$$

und $f''(a) = f''(b)$ gilt, dann sind

$$s'(x_0) = s'(x_n) \quad \text{und} \quad s''(x_0) = s''(x_n) \quad (10)$$

sinnvolle Randbedingungen, woraus sich zwei zusätzliche lineare Gleichungen zur Ergänzung des Gleichungssystems ableiten lassen.

(d) (nicht in der Vorlesung) **Not-in-knot Bedingung:** Es soll zusätzlich

$$s_0'''(x_1) = s_1'''(x_1) \quad \text{und} \quad s_{n-2}'''(x_{n-1}) = s_{n-1}'''(x_{n-1})$$

gelten, das heißt s ist auf $[x_0, x_2]$ und auf $[x_{n-2}, x_n]$ ein Polynom dritten Grades. Man erhält daraus die Forderungen $a_0 = a_1$ und $a_{n-2} = a_{n-1}$, woraus sich zusätzliche Gleichungen in den

Variablen m_0, m_1, m_2 und m_{n-2}, m_{n-1}, m_n ergeben.

3.4. Eine Minimaleigenschaft kubischer C^2 -Interpolationssplines

Durch

$$\langle f, g \rangle := \int_a^b f(x)g(x)dx \quad \text{bzw.} \quad \|g\|_2 := \sqrt{\int_a^b g(x)^2 dx} \quad \text{für } f, g \in L^2[a, b]$$

ist ein Skalarprodukt bzw. eine Norm in $L^2[a, b]$ definiert.

Satz 3.3

Seien $f \in C^2[a, b]$, Δ eine Zerlegung von $[a, b]$ und $f_k := f(x_k)$ für $k = 0, \dots, n$. Für einen Interpolationsspline $s \in \mathcal{S}_3^2(\Delta)$, der die natürlichen, vollständigen oder periodischen Randbedingungen (bei letzteren gelte Gleichung (9)) erfüllt, gilt:

$$\|s''\|_2^2 = \|f''\|_2^2 - \|f'' - s''\|_2^2 \leq \|f''\|_2^2$$

Beweis. Durch Nachrechnen sieht man

$$\int_a^b (f''(x))^2 dx - \int_a^b (f''(x) - s''(x))^2 dx = \int_a^b (s''(x))^2 dx + 2 \int_a^b [f''(x) - s''(x)] s''(x) dx$$

Es wird nun $J := \int_a^b [f''(x) - s''(x)] s''(x) dx = 0$ gezeigt. Mit Hilfe partieller Integration folgt

$$J = [f'(x) - s'(x)] s''(x) \Big|_a^b - \int_a^b [f'(x) - s'(x)] s'''(x) dx$$

wobei s''' auf jedem Teilintervall $[x_k, x_{k+1}]$ konstant ist. Dies ergibt wegen Gleichung (1)

$$\begin{aligned} \int_a^b [f'(x) - s'(x)] s'''(x) dx &= \sum_{k=0}^{n-1} s''' \left(x_k + \frac{h_k}{2} \right) \int_{x_k}^{x_{k+1}} f'(x) - s'(x) dx \\ &= \sum_{k=0}^{n-1} s''' \left(x_k + \frac{h_k}{2} \right) ([f(x_{k+1}) - s(x_{k+1})] - [f(x_k) - s(x_k)]) \\ &= 0 \end{aligned}$$

und damit

$$J = [f'(x) - s'(x)] s''(x) \Big|_a^b = [f'(b) - s'(b)] s''(b) - [f'(a) - s'(a)] s''(a)$$

Nutzt man nun noch Gleichung (7), Gleichung (8) bzw. Gleichung (9) mit Gleichung (10), so folgt $J = 0$. \square

Anmerkung

- Gleichung (7): natürliche Randbedingungen: $s''(a) = s''(b) = 0$
- Gleichung (8): vollständige Randbedingungen: $s(a') = f'(a)$, $s'(b) = f'(b)$
- Gleichung (9) und Gleichung (10): periodische Randbedingungen: $s'(a) = s'(b)$, $s''(a) = s''(b)$, $f'(a) = f'(b)$

3.5. Interpolationsfehler bei kubischer C^2 -Interpolation

Anmerkung

Die CAUCHY-SCHWARZ-Ungleichung hat folgende Form:

$$|\langle f, g \rangle| \leq \|f\|_2 \cdot \|g\|_2$$

Satz 3.4

Seien $f \in C^2[a, b]$, Δ eine Zerlegung von $[a, b]$ und $f_k := f(x_k)$ für $k = 0, \dots, n$. Für einen Interpolationsspline $s \in \mathcal{S}_3^2(\Delta)$, der die natürlichen, vollständigen oder periodischen Randbedingungen (bei letzteren gelte Gleichung (9)) erfüllt, gilt:

$$\|f - s\|_\infty \leq \frac{1}{2} h^{3/2} \|f''\|_2$$

wobei $h := \max\{h_0, \dots, h_{n-1}\}$.

Beweis. Die Funktion $r := f - s$ hat wegen Gleichung (1) die $n + 1$ Nullstellen x_0, \dots, x_n . Der maximale Abstand benachbarter Nullstellen ist h . Nach dem Satz von Rolle besitzt r' mindestens n Nullstellen. Der Abstand zweier Nullstellen von r' ist durch $2h$ nach oben beschränkt. Sei $z \in [a, b]$ so gewählt, dass $|r'(z)| = \|r'\|_\infty$. Dann gilt $|z - z^0| \leq h$ für die z am nächsten liegende Nullstelle z^0 von r' . O.B.d.A. sei $z^0 \leq z$. Mit der CAUCHY-SCHWARZ-Ungleichung folgt:

$$\begin{aligned} \|r'\|_\infty^2 &= |r'(z) - \underbrace{r'(z^0)}_{z^0 \text{ NST}}|^2 \\ &\stackrel{*}{=} \left| \int_{z^0}^z r''(x) \cdot 1 dx \right|^2 \\ &\stackrel{\text{CS}}{\leq} \int_{z^0}^z r''(x)^2 dx \cdot \underbrace{\int_{z^0}^z 1^2 dx}_{=z-z^0 \leq h} \\ &\stackrel{\text{UG}}{\leq} h \|r''\|_2^2 \end{aligned} \tag{11}$$

*: Anwendung des Hauptsatzes der Integralrechnung

Sei nun $y \in [a, b]$ so gewählt, dass $|r(y)| = \|r\|_\infty$. Dann gilt $|y - y_0| \leq h/2$ für die y am nächsten liegende Nullstelle y_0 von r . O.B.d.A. sei $y_0 \leq y$. Mit Gleichung (11) ergibt sich

$$\|r\|_\infty = |r(y) - r(y_0)| = \left| \int_{y_0}^y r'(x) dx \right| \leq \max |r'(x)| \cdot \int_{y_0}^y dx \leq \frac{1}{2} \|r'\|_\infty \leq \frac{1}{2} h^{3/2} \|r''\|_2$$

Mit Satz 3.3 hat man $\|r''\|_2 \leq \|f\|_2$ und damit die Behauptung. \square

► Bemerkung 3.5

Besitzt f eine höhere Glattheit, so kann die obige Fehlerschranke bezüglich der h -Potenz verbessert werden. Es lassen sich ferner Abschätzungen für $\|f' - s'\|_\infty$ und $\|f'' - s''\|_\infty$ herleiten.

Kapitel II

numerische Integration (Quadratur)

1. Integration von Interpolationspolynomen

Für eine Funktion $f \in C[a, b]$ ist eine Näherung für den Wert des bestimmten Integrals

$$J(f) := \int_a^b f(x) dx$$

gesucht. Seien $a \leq x_0 < \dots < x_n \leq b$ Stützstellen und $f_k = f(x_k)$ für $k = 0, \dots, n$. Weiter bezeichne $p_n \in \Pi_n$ das zugehörige Interpolationspolynom. Dann kann man

$$Q_n(f) := J(p_n) = \int_a^b p_n(x) dx$$

als Näherung für $J(f)$ verwenden. Mit der LAGRANGE-Form des Interpolationspolynoms sieht man, dass

$$Q_n(f) = \int_a^b \sum_{k=0}^n f_k \cdot L_k(x) dx = \sum_{k=0}^n f_k \cdot \int_a^b L_k(x) dx$$

das heißt die Quadraturformel $Q_n(f)$ ist die gewichtete Summe von Funktionswerten der Funktion f mit den Gewichten $\int_a^b L_k(x) dx$.

2. Newton-Cotes-Formeln

Falls die Stützstellen gleichabständig sind mit $x_0 = a$ und $x_n = b$, das heißt

$$x_{k+1} = x_k + h \quad \text{für } k = 0, \dots, n-1 \quad (1)$$

mit der Schrittweite $h = \frac{b-a}{n}$ gilt, so nennt man $Q_n(f)$ geschlossene NEWTON-COTES-Formel. Ist $a < x_0$ und $x_n < b$ und gilt Gleichung (1) mit $h = \frac{b-a}{n+2}$, so bezeichnet man $Q_n(f)$ als offene NEWTON-COTES-Formel. Im Folgenden wollen wir uns auf den Fall geschlossener NEWTON-COTES-Formeln beschränken.

3. spezielle Newton-Cotes-Formeln

Für $n = 1$ erhält man die Trapezformel mit $x_0 = a$, $x_1 = b$ und $h = b - a$ wie folgt:

$$Q_1(f) = f_0 \int_a^b L_0(x) dx + f_1 \int_a^b L_1(x) dx$$

Mit

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} = \frac{b - x}{h} \quad \text{und} \quad L_1(x) = \frac{x - x_0}{x_1 - x_0} = \frac{x - a}{h}$$

$$\int_a^b L_0(x) dx = \frac{1}{h} \int_a^b (b - x) dx = \frac{1}{h} \left(bx - \frac{1}{2} x^2 \right) \Big|_a^b = \frac{1}{h} \left(\frac{b^2}{2} - ab + \frac{a^2}{2} \right) = \frac{h}{2}$$

$$\int_a^b L_1(x) dx = \frac{h}{2}$$

folgt

$$Q_1(f) = \frac{h}{2}(f_0 + f_1)$$

Für Polynomgrad $n = 2$ erhält man auf ähnliche Weise die SIMPSON-Formel (auch KEPLER'sche Fassregel genannt):

$$Q_2(f) = \frac{h}{3}(f_0 + 4f_1 + f_2)$$

Für Polynomgrade bis $n = 6$ findet man weitere Formeln in der Literatur. Formeln nur $n > 6$ werden aus numerischen Gründen nicht verwendet. Es können dann negative Gewichte auftreten.

Satz 3.1

(a) Sei $f \in C^2[a, b]$. Dann gilt:

$$|Q_1(f) - J(f)| \leq \frac{1}{12} h^3 \|f''\|_\infty$$

(b) Sei $f \in C^4[a, b]$. Dann gilt:

$$|Q_2(f) - J(f)| \leq \frac{1}{12} h^5 \|f^{(4)}\|_\infty$$

Beweis. (a) Für die Trapezformel erhält man mit Satz 2.9

$$|Q_1(f) - J(f)| = \left| \int_a^b f(x) - p_1(x) dx \right| \leq \frac{\|f''\|_\infty}{2!} \int_a^b |(x-a)(x-b)| dx$$

Mit $y := x - \frac{a+b}{2}$ ergibt sich:

$$x - a = y + \frac{b-a}{2} = y + \frac{h}{2} \quad \text{und} \quad x - b = y + \frac{a-b}{2} = y - \frac{h}{2}$$

$$\int_a^b |(x-a)(x-b)| dx = - \int_{-h/2}^{h/2} \left(y + \frac{h}{2}\right) \left(y - \frac{h}{2}\right) dy = - \left(\frac{1}{3}y^3 - \frac{1}{4}h^2y\right) \Big|_{-h/2}^{h/2} = \frac{1}{6}h^3$$

(b) Zur Analyse des Quadraturfehlers der SIMPSON-Formel untersuchen wir zunächst

$$W := \int_a^b (x-a)(x-x_1)(x-b) dx$$

$$\bar{W} := \int_a^b |(x-a)(x-x_1)(x-b)| dx$$

Mit der Substitution $y := x - x_1$ folgen $x - a = y + h$, $x - x_1 = y$ und $x - b = y - h$, also

$$W = \int_a^b (y+h)y(y-h) dy = 0 \tag{1}$$

da die zu integrierende Funktion $y \mapsto (y+h)y(y-h)$ ungerade ist. Weiter erhält man

$$\bar{W} = \int_{-h}^h |(y+h)y(y-h)| dy = -2 \int_0^h (y^2 - h^2)y dy = \frac{1}{2}h^4 \tag{2}$$

Wegen Satz 2.9 haben wir

$$f(x) - p_2(x) = \frac{f'''(\xi(x))}{3!} w(x) = \frac{1}{6} f'''(x_1) w(x) + \frac{1}{6} (f'''(\xi(x)) - f'''(x_1)) w(x)$$

für alle $x \in [a, b]$. Insbesondere ist daher $x \mapsto f'''(\xi(x))$ eine Funktion aus $C^1[a, b]$. Durch Integration folgt mit Gleichung (1):

$$\begin{aligned} \left| \int_a^b f(x) - p_2(x) dx \right| &= \frac{1}{6} \left| f'''(x_1) \int_a^b w(x) dx + \int_a^b (f'''(\xi(x)) - f'''(x_1)) w(x) dx \right| \\ &= \frac{1}{6} \left| \int_a^b (f'''(\xi(x)) - f'''(x_1)) w(x) dx \right| \\ &\leq \max_{x \in [a, b]} \left\{ |f'''(x) - f'''(x_1)| \cdot \int_a^b |w(x)| dx \right\} \end{aligned} \tag{3}$$

Da $f \in C^4[a, b]$ erhält man mit dem Mittelwertsatz

$$|f'''(x) - f'''(x_1)| = |f^{(4)}(\zeta(x))| \cdot |x - x_1| \leq h \|f^{(4)}\|_\infty$$

mit $\zeta \in (a, b)$. Deshalb und wegen Gleichung (2) folgt aus Gleichung (3) weiter

$$|Q_2(f) - J(f)| = \left| \int_a^b f(x) - p_2(x) dx \right| \leq \frac{1}{12} h^5 \|f^{(4)}\|_\infty \quad \square$$

► Bemerkung 3.2

Durch verfeinerte Abschätzungen kann man in Satz 3.1 b)

$$|Q_2(f) - J(f)| \leq \frac{1}{90} h^5 \|f^{(4)}\|_\infty \quad (4)$$

erreichen. Auch für Quadraturformeln $Q_n(f)$ mit $n > 2$ lassen sich entsprechende Ergebnisse für den Quadraturfehler herleiten, insbesondere hat der Quadraturfehler für $n = 3$ mit $f \in C^4[a, b]$ die Ordnung h^5 und für $n = 4$ mit $f \in C^6[a, b]$ die Ordnung h^7 .

4. Zusammengesetzte Newton-Cotes-Formeln

Um den Quadraturfehler weiter zu reduzieren, bietet es sich unter Berücksichtigung der Abschätzung des Quadraturfehlers in Satz 3.1 an, das Intervall $[a, b]$ in r Teilintervalle zu zerlegen und auf jedem der Teilintervalle dieselbe Quadraturformel (niedriger Ordnung) anzuwenden. Dazu wird das Intervall $[a, b]$ in $l = rn$ Elementarintervalle gleicher Länge zerlegt, wobei n die Ordnung des auf jedem Teilintervall zu verwendenden Interpolationspolynoms ist. Mit f_0, \dots, f_l werden die Funktionswerte an den Stellen $x_k = a + kh$ für $k = 0, \dots, l$ bezeichnet, wobei $h = \frac{b-a}{l}$ die Länge des Elementarintervalls ist. Die zusammengesetzte Trapezformel ist dann gegeben durch:

$$T_h(f) = \frac{h}{2}(f_0 + 2f_1 + 2f_2 + \dots + 2f_{l-1} + f_l)$$

die zusammengesetzte SIMPSON-Formel durch

$$S_h(f) = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \dots + 2f_{l-2} + 4f_{l-1} + f_l)$$

Satz 4.1

(a) Für $f \in C^2[a, b]$ gilt

$$|T_h(f) - J(f)| \leq \frac{b-a}{12} h^2 \|f''\|_\infty$$

(b) Für $f \in C^4[a, b]$ gilt

$$|S_h(f) - J(f)| \leq \frac{b-a}{180} h^4 \|f^{(4)}\|_\infty$$

Beweis. Wendet man Satz 3.1 auf die SIMPSON-Formel (unter Beachtung von Gleichung (4)) für $[x_k, x_{k+2}]$ anstelle von $[a, b]$ an, so folgt

$$|S_h(f) - J(f)| = \left| S_h(f) - \sum_{k=0}^{r-1} \int_{x_{2k}}^{x_{2k+2}} f(x) dx \right| \leq \frac{1}{90} r h^5 \|f^{(4)}\|_\infty \leq \frac{b-a}{2 \cdot 90} h^4 \|f^{(4)}\|_\infty$$

und damit Behauptung b). Behauptung a) zeigt man auf ähnliche Weise. \square

5. Gauss'sche Quadraturformeln

Wir gehen zunächst vom Interpolationsfehler (vgl. Satz 2.9)

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!} w_n(x) \quad \text{für } x \in [a, b]$$

mit $w_n(x) = (x - x_0) \dots (x - x_n)$ aus. Bezogen auf das ganze Intervall $[a, b]$, kann man etwa $\|f - p_n\|_\infty$ oder $\|f - p_n\|_2$ als Maß für diesen Fehler verwenden. Da man über $f^{(n+1)}$ nicht verfügt, wird anstelle dessen $\|w_n\|_\infty$ oder $\|w_n\|_2$ untersucht. Dieses Fehlermaß ist offenbar nur von der Lage der Stützstellen $x_0, \dots, x_n \in [a, b]$ abhängig. Zur Vereinfachung beschränkt man sich zunächst auf das Intervall $[-1, 1]$. Die Aufgabe, die Funktion

$$F_\infty : \begin{cases} \mathbb{R}^{n+1} & \rightarrow \mathbb{R} \\ F_\infty(x_0, \dots, x_n) & \mapsto \max_{x \in [-1, 1]} |(x - x_0) \dots (x - x_n)| \end{cases}$$

unter der Bedingung $(x_0, \dots, x_n) \in [-1, 1]^{n+1}$ zu minimieren, hat als Lösung die Nullstellen des sogenannten TSCHEBYSCHOW-Polynoms T_{n+1} der Ordnung $n + 1$. Die Funktion

$$F_2 : \begin{cases} \mathbb{R}^{n+1} & \rightarrow \mathbb{R} \\ F_2(x_0, \dots, x_n) & \mapsto \int_{-1}^1 (x - x_0)^2 \dots (x - x_n)^2 dx \end{cases}$$

wird unter der Bedingung $(x_0, \dots, x_n) \in [-1, 1]^{n+1}$ durch die Nullstellen des sogenannten LEGENDRE-Polynoms P_{n+1} minimiert. Die LEGENDRE-Polynome können rekursiv wie folgt definiert werden:

$$\begin{aligned} P_0(t) &= 1 \\ P_1(t) &= t \\ &\vdots \\ (k+1)P_{k+1}(t) &= (2k+1)tP_k(t) - kP_{k-1}(t) \end{aligned}$$

für alle $t \in \mathbb{R}$ und $k = 1, 2, \dots$. Für $n = 1$ erhält man zum Beispiel $P_2(t) = t^2 - \frac{1}{3}$ mit den Nullstellen $x_{1/2} = \pm \frac{\sqrt{3}}{3}$. Das Interpolationspolynom zu diesen Stützstellen und den Stützwerten $f_0 = f(x_0)$ sowie $f_1 = f(x_1)$ lautet dann (in der LAGRANGE-Form)

$$q_1(x) = f_0 \frac{x - x_1}{x_0 - x_1} + f_1 \frac{x - x_0}{x_1 - x_0}$$

Wegen

$$\begin{aligned} \int_{-1}^1 \frac{x - x_1}{x_0 - x_1} dx &= -\frac{\sqrt{3}}{2} \int_{-1}^1 \left(x - \frac{\sqrt{3}}{3} \right) dx = 1 \\ \int_{-1}^1 \frac{x - x_0}{x_1 - x_0} dx &= 1 \end{aligned}$$

hat man die GAUSS-LEGENDRE-Quadraturformel für das Integral $\int_{-1}^1 f(x)dx$ für $n = 1$:

$$f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$$

Man kann zeigen, dass die GAUSS-LEGENDRE-Quadraturformel mit $n+1$ Stützstellen (also den Nullstellen von P_{n+1}) Polynome bis zum Grad $2n+1$ exakt integriert. Bei den geschlossenen NEWTON-COTES-Formeln ist dies nur bis Grad n (falls n ungerade) bzw. bis zum Grad $n+1$ (falls n gerade) möglich (vgl. Übungsaufgabe). Spezielle Modifikationen der GAUSS-LEGENDRE-Quadratur beziehen einen Randpunkt (GAUSS-RADAU) oder beide Randpunkte (GAUSS-LOBATTO) des Integrals mit ein.

Falls über $[a, b]$ zu integrieren ist, führt eine Variablentransformation auf ein Integral über $[-1, 1]$ zum Ziel. Mit

$$y = \frac{2}{b-a} \left(x - \frac{a+b}{2} \right)$$

ergibt sich $x = \frac{b-a}{2}y + \frac{a+b}{2}$, $dx = \frac{b-a}{2}dy$ und

$$\int_a^b f(x)dx = \frac{b-a}{2} \int_{-1}^1 f\left(\frac{b-a}{2}y + \frac{a+b}{2}\right) dy$$

Kapitel III

direkte Verfahren für lineare Gleichungssysteme

1. Gauss'scher Algorithmus für quadratische Systeme

1.1. Grundform des Gauss'schen Algorithmus

■ Beispiel 1.1

$$\begin{array}{rcl} 2x_1 - 2x_2 + 4x_3 = 10 & E_1 \\ x_1 + 3x_2 + 6x_3 = 25 & E_2 \\ -x_1 + 2x_2 + x_3 = 6 & E_3 \end{array}$$

E_1 behalten $\rightarrow E'_1$, $E_2 - \frac{1}{2}E_1 \rightarrow E'_2$, $E_3 + \frac{1}{2}E_1 \rightarrow E'_3$

$$\begin{array}{rcl} 2x_1 - 2x_2 + 4x_3 = 10 & E'_1 \\ 4x_2 + 4x_3 = 20 & E'_2 \\ x_2 + 3x_3 = 11 & E'_3 \end{array}$$

E'_1 behalten $\rightarrow E''_1$, E'_2 behalten $\rightarrow E''_2$, $E'_3 - \frac{1}{4}E'_2 \rightarrow E''_3$

$$\begin{array}{rcl} 2x_1 - 2x_2 + 4x_3 = 10 & E''_1 \\ 4x_2 + 4x_3 = 20 & E''_2 \\ 2x_3 = 6 & E''_3 \end{array}$$

$\Rightarrow x_3 = 3, x_2 = 2, x_1 = 1$

Alle drei Systeme sind äquivalent, das heißt ihre Lösungsmengen sind gleich. Das letzte System wird Dreieckssystem oder System in Zeilenstufenform oder gestaffeltes System genannt.

Gegeben seien $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ und $b = (b_i) \in \mathbb{R}^n$. Gesucht ist, falls vorhanden, eine Lösung des linearen Gleichungssystems

$$\begin{array}{rclcl} a_{11}x_1 + \dots + a_{1n}x_n & = & b_1 \\ \vdots & & \vdots & \vdots & \vdots \\ a_{n1}x_1 + \dots + a_{nn}x_n & = & b_n \end{array}$$

bzw. in Matrix-Schreibweise: $Ax = b$.

Prinzipielles Vorgehen

1. Vorwärtselimination (unter Voraussetzung der Durchführbarkeit): Schrittweise Transformation der erweiterten Koeffizientenmatrix

$$(A, b) = (A^{(1)}, b^{(1)}) \rightarrow (A^{(2)}, b^{(2)}) \rightarrow \dots \rightarrow (A^{(n)}, b^{(n)}) = (U, z)$$

wobei U eine obere Dreiecksmatrix ist. Der Eliminationsschritt $(A^{(k)}, b^{(k)}) \rightarrow (A^{(k+1)}, b^{(k+1)})$ für $k = 1, \dots, n - 1$ verwendet die Eliminationsfaktoren

$$l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

um die i -te Zeile der neuen Matrix aus der alten Matrix zu bestimmen

$$\begin{aligned} \text{neue Zeile } i &= \text{alte Zeile } i && \text{für } i = 1, \dots, k \\ \text{neue Zeile } i &= \text{alte Zeile } i - l_{ik} \cdot \text{neue Zeile } k && \text{für } i = k + 1, \dots, n \end{aligned}$$

2. Rücksubstitution (unter Voraussetzung der Durchführbarkeit): Lösung des Gleichungssystems $Ux = z$ nach x für gegebenes U, z

■ Algorithmus 1.2 (Vorwärtselimination)

Input: n, A, b

```

1  do k = 1, n-1
2  do i = k+1, n
3    lik = aik / akk
4    bi = bi - likbk
5  do j = k+1, n
6    aij = aij - likakj
7  end do
8  end do
9  end do
    
```

Output: (U, z) und l_{ik} für $i > k$. U steht in der oberen Hälfte von A mit Hauptdiagonale, b enthält z , die Zahlen l_{ik} lassen sich in der unteren Hälfte von A abspeichern.

■ Algorithmus 1.3 (Rücksubstitution)

Input: n, U, z

```

1  do i = n, 1, -1
2    s = 0
3  do j = i+1, n
4    s = s + uijxj
    
```

```

5   end do
6   end do

```

Output: x

Der Aufwand bei uneingeschränkter Durchführbarkeit von Algorithmus 1.2 ist $\sim \frac{2}{3}n^3$ und Algorithmus 1.3 ist $\sim n^2$

Durchführbarkeit

Der Algorithmus 1.2 ist genau dann durchführbar, wenn $a_{kk}^{(k)} \neq 0$ für alle $k = 1, \dots, n - 1$ gilt. Gilt auch $a_{nn}^{(n)} \neq 0$, so folgt $u_{ii} \neq 0$ für $i = 1, \dots, n$ und damit die Durchführbarkeit von Algorithmus 1.3.

Definition 1.4 (streng diagonaldominant)

Eine Matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ heißt streng diagonaldominant, wenn

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}| \quad \text{für alle } i = 0, \dots, n$$

Lemma 1.5

Ist die Matrix $A \in \mathbb{R}^{n \times n}$ streng diagonaldominant, so sind Algorithmus 1.2 und Algorithmus 1.3 durchführbar

Beweis (nicht in der Vorlesung). Die Matrix $A^{(1)}$ sei streng diagonaldominant. Weiter seien die Matrizen $A^{(k)}$ für ein $k \in \{1, \dots, n - 1\}$ durch Vorwärtselimination erzeugt und streng diagonaldominant. Dies zieht $|a_{kk}^{(k)}| > 0$ nach sich, so dass die Erzeugung von $A^{(k+1)}$ durch Vorwärtselimination wohldefiniert ist. Es wird nun gezeigt, dass $A^{(k+1)}$ wieder streng diagonaldominant ist. Da $A^{(1)} = A$ als streng diagonaldominant vorausgesetzt wurde, folgt dann die Durchführbarkeit der gesamten Vorwärtselimination durch vollständige Induktion. Sei $i > k$ eine Zeile der Matrix $A^{(k+1)}$. Dann hat man

$$\begin{aligned}
 \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}^{(k+1)}| &= \sum_{\substack{j=k+1 \\ j \neq i}}^n |a_{ij}^{(k+1)}| = \sum_{\substack{j=k+1 \\ j \neq i}}^n \left| a_{ij}^{(k)} - \frac{a_{kj}^{(k)} a_{ik}^{(k)}}{a_{kk}^{(k)}} \right| \\
 &\leq \sum_{\substack{j=k+1 \\ j \neq i}}^n |a_{ij}^{(k)}| + \left| \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right| \sum_{\substack{j=k+1 \\ j \neq i}}^n |a_{kj}^{(k)}| \\
 &< |a_{ii}^{(k)}| - |a_{ik}^{(k)}| + \left| \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \right| \left(|a_{kk}^{(k)}| - |a_{ki}^{(k)}| \right) \\
 &= |a_{ii}^{(k)}| - \left| \frac{a_{ik}^{(k)} a_{ki}^{(k)}}{a_{kk}^{(k)}} \right| \\
 &\leq \left| a_{ii}^{(k)} - \frac{a_{ik}^{(k)} a_{ki}^{(k)}}{a_{kk}^{(k)}} \right| \\
 &= |a_{ii}^{(k+1)}|
 \end{aligned}$$

Falls $i \leq k$, so ändert sich $A^{(k+1)}$ gegenüber $A^{(k)}$ bezüglich der Zeile i nicht. Also ist $A^{(k+1)}$ streng diagonaldominant und man schließt auf die Durchführbarkeit der Vorwärtselimination für $k = 1, \dots, n - 1$ und insbesondere auf $|a_{ii}^{(n)}| > 0$ für $i = 1, \dots, n$. Die Matrix $A^{(n)}$ enthält die Matrix U im oberen Dreieck, deren Diagonalelemente

sind gerade $a_{11}^{(n)}, \dots, a_{nn}^{(n)}$, also ist auch die Rücksubstitution wohldefiniert. \square

1.2. Pivotisierung

Die Regularität der Matrix $A \in \mathbb{R}^{n \times n}$ ist zwar äquivalent zur Lösbarkeit des linearen Gleichungssystems $Ax = b$, für jeden beliebigen Vektor $b \in \mathbb{R}^n$, jedoch sichert die Regularität nicht die Durchführbarkeit der Grundform des GAUSS'schen Algorithmus. Um die Durchführbarkeit bei regulärem A zu erzwingen, kann man eine Spaltenpivotisierung der Matrix durchführen. Dabei werden in jedem Durchlauf der Vorwärtselemination auf bestimmte Weise Zeilen der Matrix (A, b) vertauscht:

- Bestimme $p = p(k) \in \{k, \dots, n\}$, sodass $|a_{pk}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}|$.
 k -te Spalte von $A^{(k)}$ heißt Pivotspalte, $a_{pk}^{(k)}$ heißt Pivotelement, die Regularität von A sichert dann $a_{pk}^{(k)} \neq 0$
- Vertausche die Zeilen p und k in der Matrix $(A^{(k)}, b^{(k)})$.
praktisch: Zeilentausch nicht ausführen, sondern einen Permutationsvektor mitführen.
formal: Beschreibung der Zeilen- und Spaltenvertauschungen durch Permutationsmatrizen. Dazu sei $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$ eine Permutation und e_i bezeichne den i -ten kanonischen Einheitsvektor. Dann heißt $P_\pi = (e_{\pi(1)}, \dots, e_{\pi(n)})$ Permutationsmatrix.

Satz 1.6

Ist die Matrix A regulär, so ist der GAUSS'sche Algorithmus mit Spaltenpivotisierung (bei exakter Arithmetik) durchführbar.

Weitere Pivotisierungstechniken sind insbesondere die Zeilenpivotisierung (in Analogie zur Spaltenpivotisierung) und die vollständige Pivotisierung.

1.3. LU-Faktorisierung

Der k -te Schritt von Algorithmus 1.2 (ohne Pivotisierung) lässt sich schreiben als

$$\begin{aligned} A^{(k+1)} &= L_k A^{(k)} \\ b^{k+1} &= L_k b^{(k)} \end{aligned} \tag{1}$$

(b) Es wird durch vollständige Induktion gezeigt, dass

$$L_1^{-1} \cdot \dots \cdot L_{n-1}^{-1} = \mathbb{1} + \sum_{i=1}^k l_i e_i^T \tag{2}$$

für $k = 1, \dots, n - 1$ gilt. Daraus ergibt sich unmittelbar die zweite Aussage des Satzes. Für $k = 1$ folgt Gleichung (2) direkt aus Teil (a). Sei nun Gleichung (2) für ein $k < n - 1$ erfüllt. Dann folgt mit $e_i^T l_{k+1} = 0$ für $i < k$, dass

$$\begin{aligned} L_1^{-1} \cdot \dots \cdot L_{k+1}^{-1} &= \left(\mathbb{1} + \sum_{i=1}^k l_i e_i^T \right) L_{k+1}^{-1} \\ &= \left(\mathbb{1} + \sum_{i=1}^k l_i e_i^T \right) (\mathbb{1} + l_{k+1} e_{k+1}^T) \\ &= \mathbb{1} + \sum_{i=1}^k l_i e_i^T + l_{k+1} e_{k+1}^T + \sum_{i=1}^k l_i e_i^T l_{k+1} e_{k+1}^T \\ &= \mathbb{1} + \sum_{i=1}^{k+1} l_i e_i^T \end{aligned} \quad \square$$

Aus $A^{(k+1)} = L_k A^{(k)}$ nach Gleichung (1) folgt

$$A = A^{(1)} = L_1^{-1} A^{(2)} = \dots = L_1^{-1} \cdot \dots \cdot L_{n-1}^{-1} A^{(n)} = L \cdot U$$

und analog $b = Lz$.

Satz 1.8

Seien $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ gegeben. Falls Algorithmus 1.2 ohne Pivotisierung durchführbar ist, dann gilt $A = LU$ mit der oberen Dreiecksmatrix $U = A^{(n)}$ und der unteren Dreiecksmatrix $L = (l_{ik})$ mit

$$l_{ik} = \begin{cases} 0 & i < k \\ 1 & i = k \\ l_{ik} & i > k \end{cases}$$

■ **Algorithmus 1.9 (LU-Version der Grundform des Gauss'schen Algorithmus)**

Input: A, b

- 1 compute L, U
- 2 solve $Lz=b$
- 3 solve $Ux=z$

Output: x, L, U

Der Aufwand an Rechenoperationen bei uneingeschränkter Durchführbarkeit: $\frac{2}{3}n^3 + 2n^2$.

Ein Vorteil der LU-Version besteht darin, dass man mit einer ermittelten LU-Faktorisierung von A das Gleichungssystem $Ax = b$ für mehrere rechte Seiten b mit dem Aufwand von je $2n^2$ lösen kann.

Satz 1.10

Sei $A \in \mathbb{R}^{n \times n}$ regulär. Dann gibt es eine durch Zeilenvertauschungen aus A hervorgegangene Matrix \tilde{A} , für die Algorithmus 1.2 ohne Pivotisierung durchführbar ist und $\tilde{A} = \tilde{L}\tilde{U}$.

Beweis. Nach Satz 1.6 ist der GAUSS'sche Algorithmus mit Spaltenpivotisierung durchführbar. Wendet man die dabei vorkommenden Zeilenvertauschungen vor Beginn von Algorithmus 1.2 auf A an, entsteht die Matrix \tilde{A} . Für $A = \tilde{A}$ liefert Satz 1.8 die Darstellung $\tilde{A} = \tilde{L}\tilde{U}$. Die Regularität von U folgt aus der Regularität von A mit dem Determinantenmultiplikationssatz. \square

1.4. Gauss'scher Algorithmus für trigonale Systeme

Sei $A \in \mathbb{R}^{n \times n}$ trigonal, das heißt

$$A = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ \gamma_2 & \alpha_2 & \beta_2 & & & \\ & \gamma_3 & \alpha_3 & \beta_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \gamma_{n-1} & \alpha_{n-1} & \beta_{n-1} \\ & & & & \gamma_n & \alpha_n \end{pmatrix} \tag{3}$$

Die Speicherung kann mittels geeigneter Vektoren, etwa

$$\alpha = (\alpha_1, \dots, \alpha_n)^T \quad \beta = (\beta_1, \dots, \beta_{n-1}, 0)^T \quad \gamma = (0, \gamma_2, \dots, \gamma_n)^T \tag{4}$$

erfolgen. Angenommen Algorithmus 1.2 ist ohne Pivotisierung durchführbar. Dann gibt es eine LU-Faktorisierung von A . Aus der Trigonalität von A und aus $A = LU$ ergibt sich die folgende Gestalt für L und U

$$L = \begin{pmatrix} 1 & & & & & \\ l_2 & 1 & & & & \\ & l_3 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & l_n & 1 \end{pmatrix} \quad \text{und} \quad U = \begin{pmatrix} d_1 & \beta_1 & & & & \\ & d_2 & \beta_2 & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & \ddots & \beta_{n-1} \\ & & & & & d_n \end{pmatrix} \tag{5}$$

mit $d_1 = \alpha_1$ und $l_k = \frac{\gamma_k}{d_{k-1}}$, $d_k = \alpha_k - l_k \beta_k$ für $k = 2, \dots, n$.

■ **Algorithmus 1.11 (LU-Faktorisierung einer trigonalen Matrix ohne Pivotisierung)**

Input: α, β, γ entsprechend Gleichung (3) und Gleichung (4)

```

1  d1 = α1
2  do k = 2, n

```



```
3    $l_k = \gamma_k / d_{k-1}$ 
4    $d_k = \alpha_k - l_k \beta_k$ 
5   end do
```

Output: $l = (0, l_2, \dots, l_n)^T$, $d = (d_1, \dots, d_n)^T$ für Gleichung (5)

► **Bemerkung 1.12**

Der Aufwand für Algorithmus 1.11 beträgt etwa $3n$ Operationen. Auch das Lösen der Dreieckssysteme $Lz = b$ und $Ux = z$ ist billig (je etwa $2n^2$). Bei Spaltenpivotisierung kommt in U im Allgemeinen eine zweite Nebendiagonale hinzu. Die dargestellte Verfahrensweise lässt sich auf Systeme mit Bandmatrizen erweitern, die mehr als je eine untere bzw. obere Nebendiagonale besitzen.

2. Cholesky-Faktorisierung für symmetrische positiv definite Matrizen

Definition 2.1 (positiv definit)

Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt positiv definit, wenn

$$x^T A x > 0 \quad \forall x \in \mathbb{R}^n \setminus \{0\}$$

Bekannt sind folgende Zusammenhänge:

- Falls A symmetrisch ist, besitzt A nur reelle Eigenwerte.
- Sei A symmetrisch. Dann ist A genau dann positiv definit, wenn alle Eigenwerte von A positiv sind.
- Eine Matrix A ist genau dann positiv definit, wenn ihr symmetrischer Anteil $\frac{1}{2}(A + A^T)$ positiv definit ist.

2.1. Existenz der Cholesky-Faktorisierung

Satz 2.2

Sei $A \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit. Dann existiert genau eine untere Dreiecksmatrix $L = (l_{ik})$ mit $l_{kk} > 0$, sodass

$$A = LL^T$$

Beweis. Mittels vollständiger Induktion. Für $n = 1$ gilt $A = (a_{11})$ mit $a_{11} > 0$ (wegen positiver Definitheit). Also gilt die Behauptung genau für $L = (l_{11})$ mit $l_{11} = \sqrt{a_{11}}$. Sei nun die Behauptung für $n - 1$ erfüllt. Die Matrix A kann man wegen ihrer Symmetrie mit geeigneten $A_{n-1} \in \mathbb{R}^{(n-1) \times (n-1)}$, $a \in \mathbb{R}^{n-1}$ und $a_{nn} \in \mathbb{R}$ schreiben als

$$A = \begin{pmatrix} A_{n-1} & a \\ a^T & a_{nn} \end{pmatrix} \quad (1)$$

Aus der positiven Definitheit von A folgt

$$0 < \begin{pmatrix} x \\ 0 \end{pmatrix}^T A \begin{pmatrix} x \\ 0 \end{pmatrix} = x^T A_{n-1} x \quad (2)$$

das heißt A_{n-1} ist positiv definit. Nach Induktionsvoraussetzung gibt es genau eine untere Dreiecksmatrix $L_{n-1} = (l_{ij}^{(n-1)})$ mit $l_{kk}^{(n-1)} > 0$ für $k = 1, \dots, n - 1$. Um A als Produkt der Form $L_n L_n^T$ mit einer unteren Dreiecksmatrix $L_n \in \mathbb{R}^{n \times n}$ mit ausschließlich positiven Hauptdiagonalelementen darzustellen, ist der Ansatz

$$L_n = \begin{pmatrix} L_{n-1} & 0 \\ c^T & l_{nn} \end{pmatrix}$$

mit freien Parametern $c \in \mathbb{R}^{n-1}$ und $l_{nn} > 0$ die einzige Möglichkeit. Der Ansatz liefert

$$L_n L_n^T = \begin{pmatrix} L_{n-1} & 0 \\ c^T & l_{nn} \end{pmatrix} \begin{pmatrix} L_{n-1}^T & c \\ 0 & l_{nn} \end{pmatrix} = \begin{pmatrix} L_{n-1} L_{n-1}^T & L_{n-1} c \\ c^T L_{n-1} & c^T c + l_{nn}^2 \end{pmatrix} = \begin{pmatrix} A_{n-1} & a \\ a^T & a_{nn} \end{pmatrix}$$

Also müssen c und l_{nn} den Bedingungen $L_{n-1} c = a$ und $c^T c + l_{nn}^2 = a_{nn}$ genügen. Da L_{n-1} regulär ist, folgt

$$c = L_{n-1}^{-1} a \tag{3}$$

so dass c eindeutig bestimmt ist. Es wird nun gezeigt, dass $a_{nn} - c^T c > 0$ ist und damit $l_{nn}^2 > 0$ folgt. Somit ist dann l_{nn} in L_n durch die Bedingung $l_{nn} > 0$ eindeutig festgelegt auf $l_{nn} = \sqrt{a_{nn} - c^T c}$. Da A positiv definit ist und die Darstellung Gleichung (1) mit $A_{n-1} = L_{n-1} L_{n-1}^T$ nach Induktionsvoraussetzung angenommen werden kann, folgt

$$0 < (x^T, y) A \begin{pmatrix} x \\ y \end{pmatrix} = x^T L_{n-1} L_{n-1}^T x + 2y a^T x + y^2 a_{nn}$$

für alle $(x, y) \in \mathbb{R}^{n-1} \times \mathbb{R}$ mit $(x^T, y) \neq 0$. Setzt man speziell $(x, y) = (\hat{x}, \hat{y})$ mit

$$\hat{x} = (L_{n-1}^T)^{-1} L_{n-1}^{-1} a \quad \text{und} \quad \hat{y} = -1$$

so folgt mit Gleichung (3) das Gewünschte.

$$0 < a^T (L_{n-1}^T)^{-1} L_{n-1}^{-1} a - 2a^T (L_{n-1}^T)^{-1} L_{n-1}^{-1} a + a_{nn} = -c^T c + a_{nn} \quad \square$$

2.2. Berechnung des Cholesky-Faktors

Sei A wieder symmetrisch und positiv definit. Aus $A = LL^T$ folgt durch elementweises Vergleichen beider Matrizen

$$a_{ik} = (LL^T)_{ik} = \sum_{j=1}^l l_{ij} (L^T)_{kj} = \sum_{j=1}^k l_{ij} l_{kj}$$

Dies sind $\frac{n(n+1)}{2}$ Gleichungen für $\frac{n(n+1)}{2}$ Unbekannte l_{ik} , da A symmetrisch ist. Diese Gleichungen lassen sich durch spaltenweise Bestimmung der l_{ik} in L wie folgt lösen.

■ Algorithmus 2.3 (Cholesky-Faktorisierung)

Input: n , A symmetrisch und positiv definit

```

1  do k = 1, n
2    lkk = √(akk - ∑j=1k-1 lkj2)
3    do i = k+1, n
4      lik = (aik - ∑j=1k-1 lij lkj) / lkk
5    end do
6  end do
```

Output: l_{ik} für $1 \leq k \leq i \leq n$

► **Bemerkung 2.4**

Der Aufwand von Algorithmus 2.3 beträgt etwa $\frac{n^3}{3}$ Operationen. Der Algorithmus ist genau dann durchführbar, wenn A symmetrisch und positiv definit ist. Andernfalls entsteht im Verlauf des Verfahrens ein nicht-positiver Wert unter der Wurzel. Um das Ziehen der Quadratwurzel zu vermeiden, kann man anstelle von $A = LL^T$ die Umformung

$$A = LL^T = LD^{-1}D^2D^{-1}L^T$$

mit $D = \text{diag}(l_{11}, \dots, l_{nn})$ verwenden. Dann ist $\tilde{L} = LD^{-1}$ genau eine untere Dreiecksmatrix mit $l_{kk} = 1$ für $k = 1, \dots, n$ und $\tilde{D} = D^2$ eine Diagonalmatrix mit $\tilde{D} = \text{diag}(l_{11}^2, \dots, l_{nn}^2)$, sodass die sogenannte LDL-Faktorisierung $A = LL^T = \tilde{L}\tilde{D}\tilde{L}^T$ folgt.

■ **Algorithmus 2.5 (Wurzelfreie Cholesky-Faktorisierung)**

Input: n , A symmetrisch und positiv definit

```

1  do k = 1, n
2   $\widetilde{d}_{kk} = a_{kk} - \sum_{j=1}^{k-1} \widetilde{l}_{kj}^2 \widetilde{d}_{jj}$ 
3  do i = k+1, n
4   $\widetilde{l}_{ik} = \left( a_{ik} - \sum_{j=1}^{k-1} \widetilde{d}_{jj} \widetilde{l}_{ij} \widetilde{l}_{kj} \right) / \widetilde{d}_{kk}$ 
5  end do
6  end do

```

Output: $\widetilde{l}_{ik}, \widetilde{d}_{kk}$ für $1 \leq k \leq i \leq n$

3. Lineare Quadratmittelprobleme

4. Kondition linearer Gleichungssysteme

Kapitel IV

Kondition von Aufgaben und Stabilität von Algorithmen

1. Maschinenzahlen und Rundungsfehler

2. Fehleranalyse

Kapitel V

Newton-*Verfahren zur Lösung nichtlinearer Gleichungssysteme*

1. Das Newton-Verfahren

2. Gedämpftes Newton-Verfahren

Kapitel VI

lineare Optimierung

1. Ecken und ihre Charakterisierung

2. Simplex-Verfahren

3. Die Tableauform des Simplex-Verfahrens

4. Revidiertes Simplex-Verfahren

5. Bestimmung einer ersten zulässigen Basislösung

Anhang

Anhang A: Listen

A.1. Liste der Theoreme

A.2. Liste der benannten Sätze, Lemmata und Folgerungen

Satz I.2.7: Satz von FABER 1914 6

Index

- HORNER-Schema, 6
- KEPLER'sche Fassregel, 17
- LAGRANGE-Form, 4
- LEGENDRE-Polynoms, 21
- SIMPSON-Formel, 17
- TSCHEBYSCHOW-Polynoms, 21

- Basispolynom
 - LAGRANGE-Basispolynom, 4
 - NEWTON-Basispolynome, 5

- Dreieckssystem, 23

- Funktionsraum, 2

- gestaffeltes System, 23

- Interpolationsbedingungen, 2
- Interpolationspolynom, 4
- Interpolierende, 2

- kubischer Interpolationspline, 9

- Maximum-Norm, 6

- Newton-Cotes-Formel
 - geschlossene NEWTON-COTES-Formel, 16
 - offene NEWTON-COTES-Formel, 16

- Permutationsmatrix, 26
- Pivotelement, 26
- Pivotisierung
 - Spaltenpivotisierung, 26
 - vollständige Pivotisierung, 26
 - Zeilenpivotisierung, 26
- Pivotspalte, 26
- Polynomspline, 9
- positiv definit, 31

- Stützstellen, 2
- Stützstellensystem, 6
- Stützwerte, 2
- streng diagonaldominant, 25

- Trapezformel, 17

- Zeilenstufenform, 23
- zusammengesetzte SIMPSON-Formel, 20
- zusammengesetzte Trapezformel, 20