Mathematik IV für Elektrotechnik
Mathematik III für B.Sc. Informatik

Vorlesungsskriptum

Stefan Ulbrich

Fachbereich Mathematik
Technische Universität Darmstadt

Sommersemester 2007

Inhaltsverzeichnis

Numerische Mathematik

Kapitel 1

Einführung

Viele Problemstellungen aus den Ingenieur- und Naturwissenschaften lassen sich durch mathematische Modelle beschreiben, in denen häufig lineare oder nichtlineare Gleichungssysteme, Integrale, Eigenwertprobleme, gewöhnliche oder partielle Differentialgleichungen auftreten. In nahezu allen praxisrelevanten Fällen läßt das mathematische Modell keine analytische Lösung zu. Vielmehr muss die Lösung durch geeignete Verfahren auf einem Rechner näherungsweise bestimmt werden. Hierbei ist es wichtig, dass das verwendete Verfahren robust, genau und möglichst schnell ist. Die Entwicklung derartiger Verfahren ist Gegenstand der Numerischen Mathematik, einem inzwischen sehr bedeutenden Gebiet der Angewandten Mathematik. Die Numerische Mathematik entwickelt effiziente rechnergestützte Verfahren zur Lösung mathematischer Problemstellungen, unter anderem der oben genannten. Die Vorlesung gibt eine Einführung in die numerische Behandlung der folgenden Problemstellungen

- Lineare Gleichungssysteme
- Nichtlineare Gleichungssysteme
- Eigenwertprobleme
- Interpolation
- Numerische Integration
- Anfangs- und Randwertprobleme für gewöhnliche Differentialgleichungen
- Partielle Differentialgleichungen

Kapitel 2

Lineare Gleichungssysteme: Direkte Methoden

2.1 Problemstellung und Einführung

In diesem Kapitel betrachten wir direkte Verfahren zur Lösung von linearen Gleichungssystemen.

Lineares Gleichungssystem: Gesucht ist eine Lösung $x \in \mathbb{R}^n$ von

$$Ax = b. \tag{2.1}$$

mit

(2.2)
$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \in \mathbb{R}^{n,n}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \in \mathbb{R}^n, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \in \mathbb{R}^n.$$

Die hier besprochenen direkten Methoden liefern–im Gegensatz zu den später behandelten iterativen Methoden–die Lösung von (2.1), rundungsfehlerfreie Rechnung vorausgesetzt, in endlich vielen Rechenschritten. Bekanntlich ist (2.1) die Matrixschreibweise für

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i, \quad i = 1, \ldots, n.$$

Lineare Gleichungssysteme treten in der Praxis als Hilfsproblem bei einer Vielzahl von Problemstellungen auf, z.B. bei der Lösung von Rand- und Randanfangswertaufgaben für gewöhnliche und partielle Differentialgleichungen (Schaltkreissimulation, elektromagnetische Felder, ...), in der Bildverarbeitung, usw. . Schätzungen besagen, dass etwa 75% der Rechenzeit im technisch-wissenschaftlichen Bereich auf die Lösung von linearen Gleichungssystemen entfällt.

Wir erinnern zunächst an folgenden Sachverhalt.

Proposition 2.1.1 *Das lineare Gleichungssystem* (2.1) *hat eine Lösung genau dann, wenn gilt*

$$rang(A) = rang(A, b).$$

Hierbei ist bekanntlich für eine Matrix $B \in \mathbb{R}^{n,m}$ der Rang definiert durch

$$\begin{aligned}\mathrm{Rang}(B) &= \textit{Maximalzahl } r \textit{ der linear unabhängigen Zeilenvektoren}\\ &= \textit{Maximalzahl } r \textit{ der linear unabhängigen Spaltenvektoren.}\end{aligned}$$

Das lineare Gleichungssystem (2.1) *hat eine eindeutige Lösung genau dann, wenn A invertierbar ist (oder gleichbedeutend:* $\det(A) \neq 0$ *). Die eindeutige Lösung lautet dann*

$$x = A^{-1}b.$$

2.2 Das Gaußsche Eliminationsverfahren, Dreieckszerlegung einer Matrix

Das grundsätzliche Vorgehen der Gauß-Elimination ist aus der Linearen Algebra bekannt. Wir werden das Verfahren wiederholen, algorithmisch aufbereiten (d.h. in programmierbarer Form aufschreiben) und dann matrizentheoretisch analysieren.

Die Grundidee des Gaußschen Eliminationsverfahrens besteht darin, das Gleichungssystem (2.1) durch die elementaren Operationen

- Addition eines Vielfachen einer Gleichung zu einer anderen,
- Zeilenvertauschungen, d.h. Vertauschen von Gleichungen
- Spaltenvertauschungen, die einer Umnummerierung der Unbekannten entsprechen,

in ein Gleichungssystem der Form

$$Ry = c, \quad y_{\sigma_i} = x_i, \quad i = 1, \dots, n,$$

mit der durchgeführten Spaltenpermutation $(\sigma_1, \dots, \sigma_n)$ und einer oberen Dreiecksmatrix

$$R = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \end{pmatrix}$$

zu überführen, das dieselben Lösungen wie (2.1) besitzt. (2.3) ist ein sogenanntes *gestaffeltes Gleichungssystem*, das man leicht durch Rückwärtssubstitution lösen kann, solange R invertierbar ist. Werden keine Spaltenvertauschungen durchgeführt, dann gilt $x = y$.

2.2.1 Lösung gestaffelter Gleichungssysteme

Wir gehen zunächst auf die Lösung *gestaffelter Gleichungssysteme*

(2.3)
$$Rx = c$$

mit einer oberen Dreiecksmatrix

(2.4)
$$R = \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \end{pmatrix},$$

sowie

(2.5)
$$Lx = d$$

mit einer unteren Dreiecksmatrix

$$L = \begin{pmatrix} l_{11} & & 0 \\ \vdots & \ddots & \\ l_{n1} & \cdots & l_{nn} \end{pmatrix},$$

ein. (2.3) und (2.5) sind besonders leicht zu lösen:

Satz 2.2.1 *Seien $R = (r_{ij}) \in \mathbb{R}^{n,n}$ und $L = (l_{ij}) \in \mathbb{R}^{n,n}$ invertierbare obere bzw. untere Dreiecksmatrizen und $c = (c_1, \ldots, c_n)^T$, $d = (d_1, \ldots, d_n)^T$ Spaltenvektoren. Dann lassen sich die Lösungen von* (2.3) *bzw.* (2.5) *folgendermaßen berechnen:*

a) **Rückwärtssubstitution für obere Dreieckssysteme** (2.3)**:**

$$\boxed{x_i = \frac{c_i - \sum_{j=i+1}^{n} r_{ij} x_j}{r_{ii}}, \quad i = n, n-1, \ldots, 1.}$$

b) **Vorwärtssubstitution für untere Dreieckssysteme** (2.5)**:**

$$\boxed{x_i = \frac{d_i - \sum_{j=1}^{i-1} l_{ij} x_j}{l_{ii}}, \quad i = 1, 2, \ldots, n.}$$

Beweis: zu a): Da R invertierbar ist, gilt

$$\det(R) = r_{11} r_{22} \cdots r_{nn} \neq 0,$$

also $r_{ii} \neq 0$, $i = 1, \ldots, n$. Somit ergibt sich

$$\begin{aligned} x_n &= \frac{c_n}{r_{nn}} \\ x_{n-1} &= \frac{c_{n-1} - r_{n-1,n} x_n}{r_{n-1,n-1}} \\ &\vdots \end{aligned}$$

und somit induktiv a).

zu b): Wegen $\det(L) = l_{11}l_{22}\cdots l_{nn} \neq 0$ gilt $l_{ii} \neq 0$, $i = 1, \ldots, n$. Somit ergibt sich

$$\begin{aligned} x_1 &= \frac{d_1}{l_{11}} \\ x_2 &= \frac{d_2 - l_{2,1}x_1}{l_{22}} \\ &\vdots \end{aligned}$$

und wir erhalten induktiv b). □

Bemerkung: Der Aufwand für die Rückwärtssubstitution ist $O(n^2)$ an elementaren Rechenoperationen, falls nicht zusätzlich eine spezielle Besetztheitsstruktur vorliegt (Dünnbesetztheit, Bandstruktur). □

2.2.2 Das Gaußsche Eliminationsverfahrens

Wir erklären nun die Vorgehensweise beim Gaußschen Eliminationsverfahren. Statt mit den Gleichungen (2.1) zu arbeiten, ist es bequemer, die Operationen an der um die rechte Seite erweiterten Koeffizientenmatrix

$$(A, b) = \left(\begin{array}{ccc|c} a_{11} & \cdots & a_{1n} & b_1 \\ \vdots & & \vdots & \vdots \\ a_{n1} & \cdots & a_{nn} & b_n \end{array}\right)$$

durchzuführen.

Beim Gaußschen Eliminationsverfahren geht man nun wie folgt vor:

Grundkonzept des Gaußschen Eliminationsverfahrens:

0. **Initialisierung:** $(A^{(1)}, b^{(1)}) = \left(\begin{array}{ccc|c} a_{11}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ \vdots & & \vdots & \vdots \\ a_{n1}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right) := (A, b).$

1. **Pivotsuche:** Suche eine Gleichung r, die von x_1 abhängt, also mit $a_{r1}^{(1)} \neq 0$ und

vertausche sie mit der ersten Gleichung:

$$(A^{(1)}, b^{(1)}) = \left(\begin{array}{ccc|c} a_{11}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ \vdots & & \vdots & \vdots \\ a_{r1}^{(1)} & \cdots & a_{rn}^{(1)} & b_r^{(1)} \\ \vdots & & \vdots & \vdots \\ a_{n1}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right) \rightsquigarrow \left(\begin{array}{ccc|c} a_{r1}^{(1)} & \cdots & a_{rn}^{(1)} & b_r^{(1)} \\ \vdots & & \vdots & \vdots \\ a_{11}^{(1)} & \cdots & a_{1n}^{(1)} & b_1^{(1)} \\ \vdots & & \vdots & \vdots \\ a_{n1}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right)$$

$$=: \left(\begin{array}{ccc|c} \tilde{a}_{11}^{(1)} & \cdots & \tilde{a}_{1n}^{(1)} & \tilde{b}_1^{(1)} \\ \vdots & & \vdots & \vdots \\ \tilde{a}_{n1}^{(1)} & \cdots & \tilde{a}_{nn}^{(1)} & \tilde{b}_n^{(1)} \end{array}\right) = (\tilde{A}^{(1)}, \tilde{b}^{(1)}).$$

Ist A invertierbar, dann existiert immer ein solches r, da wegen der Invertierbarkeit von A die erste Spalte nicht verschwinden kann.

2. **Elimination:** Subtrahiere geeignete Vielfache der ersten Gleichung von den übrigen Gleichungen derart, dass die Koeffizienten von x_1 in diesen Gleichungen verschwinden. Offensichtlich muss man hierzu jeweils das l_{i1}-fache mit

$$l_{i1} = \frac{\tilde{a}_{i1}^{(1)}}{\tilde{a}_{11}^{(1)}}$$

der ersten Gleichung von der i-ten Gleichung subtrahieren:

$$(\tilde{A}^{(1)}, \tilde{b}^{(1)}) \quad \rightsquigarrow \quad (A^{(2)}, b^{(2)}) = \left(\begin{array}{cccc|c} \tilde{a}_{11}^{(1)} & \tilde{a}_{12}^{(1)} & \cdots & \tilde{a}_{1n}^{(1)} & \tilde{b}_1^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} & b_2^{(2)} \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} & b_n^{(2)} \end{array}\right)$$

$$=: \left(\begin{array}{ccc|c} \tilde{a}_{11}^{(1)} & \cdots & \tilde{a}_{1n}^{(1)} & \tilde{b}_1^{(1)} \\ \hline 0 & & & \\ \vdots & \hat{A}^{(2)} & & \hat{b}^{(2)} \\ 0 & & & \end{array}\right).$$

3. **Iteration:** Wende für $k = 2, \dots, n-1$ Schritt 1. und 2. auf $(\hat{A}^{(k)}, \hat{b}^{(k)})$ an:

   1_k. Wähle ein Pivotelement $a_{rk}^{(k)} \neq 0$, $k \leq r \leq n$, vertausche Zeile k und r $\rightsquigarrow \quad (\tilde{A}^{(k)}, \tilde{b}^{(k)})$

   2_k. Subtrahiere das l_{ik}-fache mit

   $$l_{ik} = \frac{\tilde{a}_{ik}^{(k)}}{\tilde{a}_{kk}^{(k)}}$$

   der k-ten Gleichung von der i-ten Gleichung, $i = k+1, \dots, n$.
   $\rightsquigarrow \quad (A^{(k+1)}, b^{(k+1)})$

Nach k Eliminationsschritten

$$(A,b) =: (A^{(1)}, b^{(1)}) \to (A^{(2)}, b^{(2)}) \to \ldots \to (A^{(k+1)}, b^{(k+1)})$$

erhalten wir also eine Zwischenmatrix der Form

$$(A^{(k+1)}, b^{(k+1)}) = \left(\begin{array}{ccc|ccc|c} \tilde{a}_{11}^{(1)} & \cdots & \tilde{a}_{1k}^{(1)} & \cdots & & \tilde{a}_{1n}^{(1)} & \tilde{b}_1^{(1)} \\ 0 & \ddots & & & & \vdots & \vdots \\ & & \tilde{a}_{kk}^{(k)} & \cdots & & \tilde{a}_{kn}^{(k)} & \tilde{b}_k^{(k)} \\ \hline & & 0 & & & & \\ & & \vdots & \hat{A}^{(k+1)} & & & \hat{b}^{(k+1)} \\ & & 0 & & & & \end{array} \right).$$

Nach $n-1$ Eliminationsschritten liegt somit ein gestaffeltes Gleichungssystem (2.3)

$$Rx = c, \quad R = A^{(n)}, \quad c = b^{(n)}$$

vor.

Pivotstrategie

Das Element $a_{rk}^{(k)}$, das in Schritt 1_k bestimmt wird, heißt *Pivotelement*. Theoretisch kann man bei der Pivotsuche jedes $a_{rk}^{(k)} \neq 0$ als Pivotelement wählen. Die Wahl kleiner Pivotelemente kann aber zu einer dramatischen Verstärkung von Rundungsfehlern führen. Gewöhnlich trifft man daher die Wahl von $a_{rk}^{(k)}$ durch

Spaltenpivotsuche: Wähle $k \leq r \leq n$ mit $|a_{rk}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}|$.

Hierbei sollten die Zeilen von A ”equilibriert” sein, also ihre Normen dieselbe Größenordnung haben.

2.2.3 Praktische Implementierung des Gauß-Verfahrens

Bei der Realisierung auf einem Rechner speichert man in der Regel auch die verwendeten Multiplikatoren l_{ik}. Wir werden sehen, dass dann das Gaußsche Eliminationsverfahren ”gratis” eine Dreieckszerlegung oder LR-Zerlegung von A der Form

(2.6) $$LR = PA$$

liefert. Hierbei ist $R \in \mathbb{R}^{n,n}$ die obere Dreiecksmatrix (2.4) aus (2.3), $L \in \mathbb{R}^{n,n}$ eine untere Dreiecksmatrix der Form

$$L = \begin{pmatrix} 1 & & 0 & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & & \ddots & \ddots & \\ l_{n1} & & \cdots & l_{n,n-1} & 1 \end{pmatrix}, \tag{2.7}$$

und P eine *Permutationsmatrix*, die lediglich die Zeilen von A permutiert.

Wir erhalten die folgende Implementierung des Gauß-Verfahrens mit Spaltenpivotsuche:

Algorithmus 1 Gaußsches Eliminationsverfahren mit Spaltenpivotsuche

Setze $(A^{(1)}, b^{(1)}) = (A, b)$ *und* $L^{(1)} = 0 \in \mathbb{R}^{n,n}$.

Für $k = 1, 2, \ldots, n-1$*:*

1. **Spaltenpivotsuche:** *Bestimme* $k \le r \le n$ *mit*

$$|a_{rk}^{(k)}| = \max_{k \le i \le n} |a_{ik}^{(k)}|.$$

   Falls $a_{rk}^{(k)} = 0$*: STOP,* A *ist singulär.*

   Vertausche die Zeilen r *und* k *von* $(A^{(k)}, b^{(k)})$ *und von* $L^{(k)}$*. Das Ergebnis sei formal mit* $(\tilde{A}^{(k)}, \tilde{b}^{(k)})$, $\tilde{L}^{(k)}$ *bezeichnet.*

2. **Elimination:** *Subtrahiere für* $i = k+1, \ldots, n$ *das* l_{ik}*-fache,* $l_{ik} = \frac{\tilde{a}_{ik}^{(k)}}{\tilde{a}_{kk}^{(k)}}$*, der* k*-ten Zeile von* $(\tilde{A}^{(k)}, \tilde{b}^{(k)})$ *von der* i*-ten Zeile und füge die Multiplikatoren* l_{ik} *in* $\tilde{L}^{(k)}$ *ein. Das Ergebnis sei formal mit* $(A^{(k+1)}, b^{(k+1)})$ *und* $L^{(k+1)}$ *bezeichnet.*

   Im Detail: *Initialisiere* $(A^{(k+1)}, b^{(k+1)}) := (\tilde{A}^{(k)}, \tilde{b}^{(k)})$, $L^{(k+1)} := \tilde{L}^{(k)}$.

   Für $i = k+1, \ldots, n$*;*

$$\begin{aligned} l_{ik} &= \frac{\tilde{a}_{ik}^{(k)}}{\tilde{a}_{kk}^{(k)}}, \\ b_i^{(k+1)} &= \tilde{b}_i^{(k)} - l_{ik}\tilde{b}_k^{(k)}, \\ a_{ik}^{(k+1)} &= 0, \\ l_{ik}^{(k+1)} &= l_{ik} \quad \textit{(Multiplikator speichern)}. \end{aligned}$$

   Für $j = k+1, \ldots, n$*:*

$$a_{ij}^{(k+1)} = \tilde{a}_{ij}^{(k)} - l_{ik}\tilde{a}_{kj}^{(k)}$$

Ergebnis: $R := A^{(n)}$, $c := b^{(n)}$, $L := I + L^{(n)}$ *mit der Einheitsmatrix* $I \in \mathbb{R}^{n,n}$.

Bei einer Implementierung auf dem Rechner kann man für die Speicherung aller $A^{(k)}$, $b^{(k)}$, $\tilde{A}^{(k)}$, $\tilde{b}^{(k)}$ die Felder verwenden, in denen A und b gespeichert waren. $L^{(k)}$ kann man anstelle der enstehenden Nullen im strikten unteren Dreieck platzsparend speichern.

Bemerkung: Der Rechenaufwand ist $n^3/3 - n/3$ an elementaren Rechenoperationen, falls nicht zusätzlich eine spezielle Besetztheitsstruktur vorliegt. $\Box$

Matrixdarstellung der Eliminationsschritte

Formal läßt sich der Übergang $(A^{(k)}, b^{(k)}) \to (\tilde{A}^{(k)}, \tilde{b}^{(k)}) \to (A^{(k+1)}, b^{(k+1)})$ durch Multiplikation mit Matrizen darstellen. Tatsächlich gilt

$$(\tilde{A}^{(k)}, \tilde{b}^{(k)}) = P_k(A^{(k)}, b^{(k)}), \quad (A^{(k+1)}, b^{(k+1)}) = L_k(\tilde{A}^{(k)}, \tilde{b}^{(k)}) = L_k P_k(A^{(k)}, b^{(k)})$$

mit der *elementaren Permutationsmatrix* (vertausche Zeile k und r der Einheitsmatrix)

$$\text{(2.8)} \qquad \begin{array}{r} \\ \\ \\ k \rightarrow \\ \\ P_k = \\ \\ r \rightarrow \\ \\ \\ \end{array} \begin{pmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & 0 & & & & 1 & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & & 1 & & & & 0 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \end{pmatrix}$$

und der *elementaren Eliminationsmatrix*

$$\text{(2.9)} \qquad L_k = \begin{pmatrix} 1 & & & 0 & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -l_{k+1,k} & 1 & & \\ 0 & & \vdots & & \ddots & \\ & & -l_{nk} & & 0 & 1 \end{pmatrix}$$

Man prüft leicht, dass P_k, L_k invertierbar sind mit

$$\text{(2.10)} \qquad P_k^{-1} = P_k, \quad L_k^{-1} = \begin{pmatrix} 1 & & & 0 & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & l_{k+1,k} & 1 & & \\ 0 & & \vdots & & \ddots & \\ & & l_{nk} & & 0 & 1 \end{pmatrix}$$

Also ist $A^{(k+1)} = L_k P_k A^{(k)}$ wieder invertierbar und der Eliminationsschritt liefert, wie wir uns schon überlegt haben, die Struktur

$$A^{(k+1)} = \left(\begin{array}{ccc|cc} r_{11} & \cdots & r_{1k} & \cdots & r_{1n} \\ 0 & \ddots & & & \vdots \\ & & r_{kk} & \cdots & r_{kn} \\ \hline & & 0 & & \\ & & \vdots & \hat{A}^{(k+1)} & \\ & & 0 & & \end{array}\right). \tag{2.11}$$

Beispiel 2.2.1 Betrachte das Beispiel

$$\begin{pmatrix} 1 & 2 & -1 \\ 2 & -2 & 4 \\ 2 & 1 & -2 \end{pmatrix} x = \begin{pmatrix} 2 \\ 10 \\ -2 \end{pmatrix}$$

Dies liefert

$$\left(\begin{array}{ccc|c} 1 & 2 & -1 & 2 \\ 2 & -2 & 4 & 10 \\ 2 & 1 & -2 & -2 \end{array}\right) \to \left(\begin{array}{ccc|c} 2 & -2 & 4 & 10 \\ 1 & 2 & -1 & 2 \\ 2 & 1 & -2 & -2 \end{array}\right)$$

$$\to \begin{array}{c} -(\underbrace{\tfrac{1}{2}}_{=l_{21}}) \cdot \text{Zeile } 1 \\ -(\underbrace{1}_{=l_{31}}) \cdot \text{Zeile } 1 \end{array} \left(\begin{array}{ccc|c} 2 & -2 & 4 & 10 \\ 0 & 3 & -3 & -3 \\ 0 & 3 & -6 & -12 \end{array}\right)$$

$$\to \quad -\underbrace{1}_{=l_{32}} \cdot \text{Zeile } 2 \left(\begin{array}{ccc|c} 2 & -2 & 4 & 10 \\ 0 & 3 & -3 & -3 \\ 0 & 0 & -3 & -9 \end{array}\right)$$

Vollständige Pivotsuche

Anstelle der Spaltenpivotsuche kann man auch *vollständige Pivotsuche* durchführen, bei der man die Pivotsuche nicht auf die erste Spalte beschränkt. Schritt 1 in Algorithmus 1 ist dann wie folgt zu modifizieren:

Algorithmus 2 Gaußsches Eliminationsverfahren mit vollständiger Pivotsuche *Algorithmus 1 mit folgender Modifikation von Schritt 1:*

$1.'$ **Vollständige Pivotsuche:** *Bestimme* $k \le r \le n$, $k \le s \le n$ *mit*

$$|a_{rs}^{(k)}| = \max_{k \le i,j \le n} |a_{ij}^{(k)}|.$$

Falls $a_{rs}^{(k)} = 0$: STOP, A ist singulär.

Vertausche die Zeilen r und k sowie die Spalten s und k von $(A^{(k)}, b^{(k)})$ und von $L^{(k)}$. Das Ergebnis sei formal mit $(\tilde{A}^{(k)}, \tilde{b}^{(k)})$, $\tilde{L}^{(k)}$ bezeichnet.

Achtung: Bei jeder Spaltenvertauschung müssen die Komponenten von x entsprechend umnumeriert werden, d.h. nach Lösen von (2.3) müssen die Komponenten des Ergebnisvektors x zurückgetauscht werden. Jeder Eliminationsschritt lautet in Matrixschreibweise

$$(A^{(k+1)}, b^{(k+1)}) = L_k P_k (A^{(k)} Q_k, b^{(k)})$$

mit einer zusätzlichen elementaren Permutationsmatrix für die Spaltenvertauschung.

In der Regel wird vollständige Pivotsuche nur bei "fast singulären" Matrizen angewandt, um den Rundungsfehlereinfluß minimal zu halten. □

Wir wollen nun folgendes zeigen.

Satz 2.2.2 *Es sei $A \in \mathbb{R}^{n,n}$ nichtsingulär. Dann gilt:*

i) Das Gaußsche Eliminationsverfahren mit Spaltenpivotsuche aus Algorithmus 1 ist durchführbar und liefert eine obere Dreiecksmatrix R und eine rechte Seite c, so dass $Rx = c$ und $Ax = b$ dieselbe Lösung besitzen.

ii) Das Gaußsche Eliminationsverfahren mit vollständiger Pivotsuche aus Algorithmus 2 ist ebenfalls durchführbar. Bezeichnet $Q = Q_1 \cdots Q_{n-1}$ die durchgeführte Spaltenpermutation, dann haben $Ry = c$ und $Ax = b$ mit $x = Qy$ dieselbe Lösung.

Beweis: Wir betrachten nur Algorithmus 1, der Fall vollständiger Pivotsuche ist dann offensichtlich. Wir zeigen durch vollständige Induktion, dass für $k = 0, \ldots, n-1$ die Matrizen $A^{(k+1)}$ jeweils nichtsingulär von der Form sind

$$A^{(k+1)} = \left(\begin{array}{ccc|cc} r_{11} & \cdots & r_{1k} & \cdots & r_{1n} \\ 0 & \ddots & & & \vdots \\ & & r_{kk} & \cdots & r_{kn} \\ \hline & & 0 & & \\ & & \vdots & \hat{A}^{(k+1)} & \\ & & 0 & & \end{array}\right) \tag{2.11}$$

und dass $A^{(k+1)}x = b^{(k+1)}$ dieselbe Lösung hat wie $Ax = b$.

Für $k = 0$ ist das wegen $A^{(1)} = A$ und $b^{(1)} = b$ klar. Sei nun bereits bekannt, dass die Behauptung für $k-1$ richtig ist. Nun liefert (2.11)

$$0 \neq \det(A^{(k)}) = r_{11} \cdots r_{k-1,k-1} \det(\hat{A}^{(k)})$$

und somit ist auch $\hat{A}^{(k)}$ invertierbar. Daher ist die Spaltenpivotsuche erfolgreich. Der Eliminationsschritt läßt sich schreiben als $A^{(k+1)} = L_k P_k A^{(k)}$, $b^{(k+1)} = L_k P_k b^{(k)}$ und liefert nach unseren Vorüberlegungen die Form (2.11). Da $L_k P_k$ invertierbar ist, bleibt die Lösungsmenge unverändert. $\square$

2.2.4 Gewinnung einer Dreieckszerlegung

Wir zeigen nun, dass dann das Gaußsche Eliminationsverfahren eine Dreieckszerlegung oder LR-Zerlegung von A der Form

$$LR = PAQ \tag{2.6}$$

liefert. Hierbei ist $R \in \mathbb{R}^{n,n}$ die obere Dreiecksmatrix (2.4) aus (2.3), $L \in \mathbb{R}^{n,n}$ eine untere Dreiecksmatrix der Form

$$L = \begin{pmatrix} 1 & & 0 & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & & \ddots & \ddots & \\ l_{n1} & & \cdots & l_{n,n-1} & 1 \end{pmatrix}, \tag{2.7}$$

und P, Q sind *Permutationsmatrizen* für die durchgeführten Zeilen- bzw. Spaltenvertauschungen ($Q = I$ bei Spaltenpivotsuche).

Bemerkung: Eine Dreieckszerlegung (2.6) ist sehr nützlich, wenn man (2.1) für mehrere rechte Seiten lösen will. Tatsächlich gilt

$$Ax = b \quad \Longleftrightarrow \quad PAQy = Pb, \;\; x = Qy \quad \Longleftrightarrow \quad L\underbrace{Ry}_{=:z} = Pb, \;\; x = Qy.$$

Man erhält nun x durch folgende Schritte:

Vorwärts-Rückwärtssubstitution für Dreieckszerlegung:

Löse $Lz = Pb$ nach z durch Vorwärtssubstitution gemäß Satz 2.2.1.
Löse $Ry = z$ nach y durch Rückwärtssubstitution gemäß Satz 2.2.1.
Lösung: $x = Qy$.

Liegt also die Dreieckszerlegung vor, dann kann (2.1) für jede rechte Seite in $O(n^2)$ Operationen berechnet werden. $\square$

Wir haben bereits festgestellt, dass jeder Schritt des Gaußschen Eliminationsverfahrens mit Spaltenpivotsuche (Algorithmus 1) geschrieben werden kann als

$$(A^{(k+1)}, b^{(k+1)}) = L_k P_k (A^{(k)}, b^{(k)})$$

mit der Permutationsmatrix P_k aus (2.8) und der elementaren Eliminationsmatrix L_k aus (2.9). Ist nun $A \in \mathbb{R}^{n,n}$ nichtsingulär dann gilt nach Durchführung des Gaußschen Algorithmus

$$R = A^{(n)} = L_{n-1}P_{n-1}\cdots L_1P_1A.$$

Multiplikation mit

$$(L_{n-1}P_{n-1}\cdots L_1P_1)^{-1} = P_1^{-1}L_1^{-1}\cdots P_{n-1}^{-1}L_{n-1}^{-1}$$

ergibt wegen $P_k^{-1} = P_k$

$$A = P_1L_1^{-1}\cdots P_{n-1}L_{n-1}^{-1}R.$$

Sind im Eliminationsverfahren keine Zeilenvertauschungen nötig, dann erhalten wir

$$A = L_1^{-1}\cdots L_{n-1}^{-1}R =: LR.$$

und man rechnet mit (2.10) leicht nach, dass gilt

$$L = L_1^{-1}\cdots L_{n-1}^{-1} = \begin{pmatrix} 1 & & 0 & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & & \ddots & \ddots & \\ l_{n1} & & \cdots & l_{n,n-1} & 1 \end{pmatrix} = I + L^{(n)}.$$

Mit den vom Gauß-Verfahren gelieferten Matrizen L und R gilt also ohne Pivotsuche

$$A = LR.$$

Allgemein gilt der folgende Satz.

Satz 2.2.3 *Es sei $A \in \mathbb{R}^{n,n}$ nichtsingulär. Dann gilt:*

i) Das Gaußsche Eliminationsverfahren aus Algorithmus 1 liefert eine untere Dreiecksmatrix L der Form (2.7) *und eine obere Dreiecksmatrix R mit*

$$LR = PA.$$

Hierbei ist $P = P_{n-1}\cdots P_1$ eine Permutationsmatrix, wobei jeweils P_k die Permutationsmatrix für die Zeilenvertauschung im k-ten Schritt ist.

ii) Algorithmus 2 liefert eine Dreieckszerlegung

$$LR = PAQ$$

Hierbei ist P wie eben und $Q = Q_1\cdots Q_{n-1}$, wobei jeweils Q_k die Permutationsmatrix für die Spaltenvertauschung im k-ten Schritt ist.

Beweis: Finden keine Zeilenvertauschungen statt, dann haben wir die Behauptung bereits gezeigt.

Für Interessierte: Der allgemeine Fall ist etwas technisch. Setze $C_k = L_k^{-1} - I$. Man prüft leicht die Gültigkeit der Formel

$$L_k^{-1} P_i = P_i (P_i C_k + I), \quad i \geq k.$$

Dies ergibt

$$P_1 L_1^{-1} \cdots P_{n-1} L_{n-1}^{-1} = P_1 \cdots P_{n-1} (P_{n-1} \cdots P_2 C_1 + I) \cdots (P_{n-1} C_{n-2} + I) L_{n-1}^{-1} = P^{-1} L,$$

wobei die Faktoren

$$\tilde{L}_k = P_{n-1} \cdots P_{k+1} C_k + I$$

aus L_k^{-1} durch Permutation der Einträge l_{ik} entstehen, die sich aus den nachfolgenden Pivotschritten ergeben. Dieselben Einträge l_{ik} werden vom Gauß-Verfahren in $L^{(k+1}$ eingetragen und bis zum Ergebnis $L^{(n)}$ genauso vertauscht.

Algorithmus 2 ist nichts anderes als Algorithmus 1 angewendet auf AQ. $\square$

Es gibt einige wichtige Teilklassen von Matrizen, bei denen auf die Pivotsuche verzichtet werden kann:

Matrizenklassen, die keine Pivotsuche erfordern

- $A = A^T$ ist symmetrisch posititv definit, also

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

  Wir gehen hierauf noch ein.

- A ist strikt diagonaldominant, d.h.

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

  siehe Übung.

- A ist M-Matrix, d.h. es gilt

  $a_{ii} > 0,\ \ i = 1, \ldots, n,$
  $a_{ij} \leq 0,\ \ i \neq j$
  $D^{-1}(A - D),\ \ D = \operatorname{diag}(a_{11}, \ldots, a_{nn})$ hat lauter Eigenwerte vom Betrag < 1.

Nach Satz 2.2.3 gibt es für eine invertierbare Matrix A immer eine Permutationsmatrix P, so dass eine Dreieckszerlegung

$$LR = PA$$

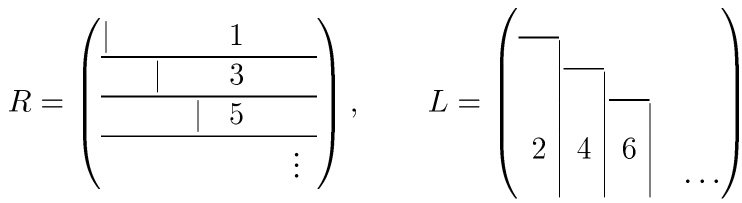
mit L, R der Form (2.7), (2.4) existiert. Tatsächlich sind L, R eindeutig bestimmt:

Satz 2.2.4 *Sei $A \in \mathbb{R}^{n,n}$ invertierbar und sei $P \in \mathbb{R}^{n,n}$ eine Permutationsmatrix, so dass eine Dreieckszerlegung* (2.6) *mit L, R der Form* (2.7)*,* (2.4) *existiert. Dann sind L und R eindeutig bestimmt.*

Beweis: **Für Interessierte:** Die Beziehung $LR = B$ liefert die folgenden n^2 Gleichungen für die n^2 unbekannten Einträge in L und R

$$b_{ij} = \sum_{k=1}^{\min(i,j)} l_{ik} r_{kj}, \quad (l_{ii} = 1).$$

Hieraus lassen sich l_{ik} und r_{kj} zum Beispiel in folgender Reihenfolge berechnen:



□

2.2.5 Das Cholesky-Verfahren

Für allgemeine invertierbare Matrizen kann das Gauß-Verfahren ohne Pivotsuche zusammenbrechen und wir werden sehen, dass auch aus Gründen der numerischen Stabilität eine Pivotsuche ratsam ist. Für die wichtige Klasse der positiv definiten Matrizen ist jedoch das Gauß-Verfahren immer ohne Pivotsuche numerisch stabil durchführbar.

Definition 2.2.5 *Eine reelle Matrix $A \in \mathbb{R}^{n,n}$ heißt positiv definit, falls gilt*

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

und positiv semi-definit, falls gilt

$$A = A^T, \quad x^T A x \geq 0 \quad \forall\, x \in \mathbb{R}^n.$$

Allgemeiner heißt eine komplexe Matrix $A \in \mathbb{C}^{n,n}$ positiv definit, falls gilt

$$A = A^H, \quad x^H A x > 0 \quad \forall\, x \in \mathbb{C}^n \setminus \{0\}.$$

und positiv semi-definit, falls gilt

$$A = A^H, \quad x^H Ax \geq 0 \quad \forall\, x \in \mathbb{C}^n.$$

Hierbei ist $A^H = (\bar{a}_{ji})_{1\leq i\leq n, 1\leq j\leq n}$, *wobei* $\bar{a}_{ji}$ *komplexe Konjugation bezeichnet.*

Positiv definite Matrizen treten sehr oft in Anwendungen auf, etwa bei der numerischen Lösung von elliptischen (z.B. Laplace-Gleichung) und parabolischen (z.B. Wärmeleitungsgleichung) partiellen Differentialgleichungen.

Satz 2.2.6 *Für eine positiv definite Matrix* A *existiert* A^{-1} *und ist wieder positiv definit. Zudem gilt: Alle Eigenwerte sind positiv, alle Hauptuntermatrizen* $A_{kl} = (a_{ij})_{k\leq i,j\leq l}$, $1 \leq k \leq l \leq n$ *sind wieder positiv definit und alle Hauptminoren* $\det(A_{kl})$ *sind positiv.*

Beweis: Elementare Übung aus der linearen Algebra. Siehe z.B. Stoer [St94]. □

Eine effiziente Variante des Gaußschen Verfahrens für Gleichungssysteme mit positiv definiter Matrix wurde von Cholesky angegeben. Das Cholesky-Verfahren beruht auf der folgenden Beobachtung

Satz 2.2.7 *Es sei* $A \in \mathbb{R}^{n,n}$ *positiv definit. Dann gibt es genau eine untere Dreieckmatrix* L *mit positiven Diagonaleinträgen* $l_{ii} > 0$, *so dass*

$$LL^T = A \quad \textit{(Cholesky-Zerlegung)}.$$

Ferner besitzt A *eine eindeutige Dreieckszerlegung*

$$\tilde{L}\tilde{R} = A,$$

wobei $\tilde{L} = LD^{-1}$, $\tilde{R} = DL^T$ *mit* $D = \operatorname{diag}(l_{11}, \ldots, l_{nn})$. *Sie wird vom Gauß-Verfahren ohne Pivotsuche geliefert wird.*

Der Beweis kann durch vollständige Induktion nach n erfolgen, wir wollen ihn aber nicht ausführen.

Die Cholesky-Zerlegung $LL^T = A$ berechnet man durch Auflösen der $\frac{n(n+1)}{2}$ Gleichungen (aus Symmetriegründen muss nur das untere Dreieck mit Diagonale betrachtet werden)

$$a_{ij} = \sum_{k=1}^{j} l_{ik} l_{jk}, \quad \text{für } j \leq i, \;\; i = 1, \ldots, n. \tag{2.12}$$

Man kann hieraus die Elemente von L spaltenweise in der Reihenfolge

$$l_{11}, \ldots, l_{n1}, l_{22}, \ldots, l_{n2}, \ldots, l_{nn}$$

berechnen. Für die erste Spalte von L (setze $j = 1$) ergibt sich

$$a_{11} = l_{11}^2, \text{ also } l_{11} = \sqrt{a_{11}}$$
$$a_{i1} = l_{i1} l_{11}, \text{ also } l_{i1} = a_{i1}/l_{11}.$$

Sukzessives Auflösen nach l_{ij}, $i = j, \dots, n$ liefert den folgenden Algorithmus.

Algorithmus 3 Cholesky-Verfahren zur Berechnung der Zerlegung $LL^T = A$

Für $j = 1, \dots, n$

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2}$$

Für $i = j+1, \dots, n$*:*

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}}{l_{jj}}$$

Bemerkung: Das Cholesky-Verfahren hat einige schöne Eigenschaften:

- Da das Cholesky-Verfahren die Symmetrie ausnutzt, benötigt es neben n Quadratwurzeln nur noch ca. $n^3/6$ Operationen. Dies ist etwa die Hälfte der beim Gauß-Verfahren benötigten Operationen.
- Aus (2.12) folgt
  $$|l_{ij}| \le \sqrt{a_{ii}}, \quad j \le i, \quad i = 1, \dots, n.$$
  Die Elemente der Matrix L können daher nicht zu groß werden. Dies ist ein wesentlicher Grund für die numerische Stabilität des Cholesky-Verfahrens.
- Das Cholesky-Verfahren ist die effizienteste allgemeine Testmethode auf positive Definitheit. Man muss hierbei Algorithmus 3 nur wie folgt erweitern:
  $$a = a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2. \qquad \textit{Falls } a \le 0\textit{: STOP, } A \textit{ nicht positiv definit.}$$
  Sonst setze $l_{jj} = \sqrt{a}$.

2.3 Fehlerabschätzungen und Rundungsfehlereinfluß

Bei der Beschreibung der direkten Verfahren zur Lösung von Gleichungssystemen sind wir bisher davon ausgegangen, dass alle Ausgangsdaten exakt vorliegen und die Rechnung ohne Rundungsfehler durchgeführt wird. Dies ist jedoch unrealistisch, denn insbesondere bei großen Systemen können Rundungsfehler die Rechnung erheblich beeinflussen.

2.3.1 Fehlerabschätzungen für gestörte Gleichungssysteme

Wir studieren zunächst, wie stark sich die Lösung eines linearen Gleichungssystems bei Störung von Matrix und rechter Seite ändert. Vorgelegt sei ein lineares Gleichungssystem

$$Ax = b$$

und ein gestörtes System

$$(A + \Delta A)\tilde{x} = b + \Delta b$$

mit ΔA und Δb "klein".

Frage: Wie klein ist $x - \tilde{x}$?

Diese Fragestellung ist von größter praktischer Bedeutung. Es stellt sich heraus, dass die sogenannte Kondition einer Matrix diesen Störeinfluss beschreibt.

Zur Messung von $x - \tilde{x}$, Δb und ΔA benötigen wir einen "Längenbegriff" für Vektoren und Matrizen.

Definition 2.3.1 *Eine* Vektornorm *auf* $\mathbb{R}^n$ *ist eine Abbildung* $x \in \mathbb{R}^n \mapsto \|x\| \in [0, \infty[$ *mit folgenden Eigenschaften:*

a) $\|x\| = 0$ *nur für* $x = 0$.

b) $\|\alpha x\| = |\alpha|\, \|x\|$ *für alle* $\alpha \in \mathbb{R}$ *und alle* $x \in \mathbb{R}^n$.

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

Nun sollen auch *Matrix-Normen* eingeführt werden. Sei hierzu $\|\cdot\|$ eine beliebige Norm auf $\mathbb{R}^n$. Dann können wir auf $\mathbb{R}^{n,n}$ eine zugehörige Matrix-Norm definieren durch

$$\|A\| := \sup_{\|x\|=1} \|Ax\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} \tag{2.13}$$

für $A \in \mathbb{R}^{n,n}$. Sie heißt die durch die *Vektornorm* $\|\cdot\|$ *indizierte Matrix-Norm.*

Sie hat wiederum die Eigenschaften

$\|A\| = 0$ nur für $A = 0$.

$\|\alpha A\| = |\alpha|\, \|A\|$ für alle $\alpha \in \mathbb{R}$ und alle $A \in \mathbb{R}^{n,n}$.

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

Zusätzlich sichert (2.13) die nützlichen Ungleichungen

$\|Ax\| \leq \|A\|\|x\|$ für alle $x \in \mathbb{R}^n$ und alle $A \in \mathbb{R}^{n,n}$ (Verträglichkeitsbedingung)

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

Beispiele:

$$\|x\|_2 = \sqrt{x^T x} \text{ induziert } \|A\|_2 = \sqrt{\lambda_{max}(A^T A)}$$

$$\|x\|_1 = \sum_{i=1}^{n} |x_i| \text{ induziert } \|A\|_1 = \max_{j=1,\dots,n} \sum_{i=1}^{n} |a_{ij}| \quad \text{(Spaltensummennorm)}$$

$$\|x\|_\infty = \max_{i=1,\dots,n} |x_i| \text{ induziert } \|A\|_\infty = \max_{i=1,\dots,n} \sum_{j=1}^{n} |a_{ij}| \quad \text{(Zeilensummennorm)}$$

Wir sind nun in der Lage, die bereits erwähnte Kondition einer Matrix einzuführen.

Definition 2.3.2 *Sei $A \in \mathbb{R}^{n,n}$ invertierbar und sei $\|\cdot\|$ eine induzierte Matrixnorm. Dann heißt die Zahl*

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

die Kondition *von A bezüglich der Matrixnorm.*

Man kann nun folgendes zeigen.

Satz 2.3.3 (Störeinfluss von Matrix und rechter Seite)
Sei $A \in \mathbb{R}^{n,n}$ invertierbar, $b, \Delta b \in \mathbb{R}^n$, $b \neq 0$ und $\Delta A \in \mathbb{R}^{n,n}$ mit $\|\Delta A\| < 1/\|A^{-1}\|$ mit einer beliebigen durch eine Norm $\|\cdot\|$ auf $\mathbb{R}^n$ induzierten Matrixnorm $\|\cdot\|$. Ist x die Lösung von

$$Ax = b$$

und $\tilde{x}$ die Lösung von

$$(A + \Delta A)\tilde{x} = b + \Delta b,$$

dann gilt

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

Beweis: Wir betrachten der Einfachheit halber nur den Fall $\Delta A = 0$. Dann liefert Subtraktion der gestörten und ungestörten Gleichung

$$A(\tilde{x} - x) = \Delta b,$$

also

$$\|\tilde{x} - x\| = \|A^{-1}\Delta b\| \leq \|A^{-1}\|\|\Delta b\|.$$

Wegen $\|b\| = \|Ax\| \leq \|A\|\|x\|$ folgt $1/\|x\| \leq \|A\|/\|b\|$ und somit

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leq \|A\|\|A^{-1}\|\frac{\|\Delta b\|}{\|b\|}.$$

□

Die Kondition bestimmt also die Sensitivität bezüglich Störungen von Matrix und rechter Seite.

2.3.2 Rundungsfehlereinfluß beim Gauß-Verfahren

Auf einem Rechner wird eine Zahl $\neq 0$ nach IEEE-Standard dargestellt in der Form

$$\pm 1, \alpha_1\, \alpha_2\, \ldots\, \alpha_{t-1} \cdot 2^b, \quad \alpha_i \in \{0,1\}, b \in \{b_-, \ldots, b_+\},$$

z.B. bei der heute üblichen doppelten Genauigkeit

$$t = 53 \text{ (ca. 15 Dezimalstellen)}, \; b_- = -1022, \; b_+ = 1024.$$

Alle elementaren Rechenoperationen sind nach IEEE-Standard so zu implementieren, dass das Ergebnis der Operation das gerundete exakte Ergebnis ist, ausser bei Exponenten-Über- oder Unterlauf. Bezeichnen wir mit $+_g$, $-_g$, usw. die Rechenoperationen auf einem Rechner, dann gilt also z.B.

$$x +_g y = \mathrm{rd}(x + y).$$

Hierbei rundet rd zur nächstgelegenen Gleitpunktzahl. Es gilt für den relativen Fehler

$$\frac{|x - \mathrm{rd}(x)|}{|x|} \leq 2^{-t} =: \text{eps}, \quad \text{eps: Maschinengenauigkeit.}$$

Somit gilt bei jeder Rechenoperation $\circ_g \in \{+_g, -_g, *_g, /_g\}$

$$x \circ_g y = (x \circ y)(1 + \varepsilon), \quad |\varepsilon| \leq \text{eps}.$$

Rundungsfehleranalyse für das Gauß-Verfahren

Durch eine elementare aber aufwendige Abschätzung der beim Gauß-Verfahren auftretenden Rundungsfehlerverstärkung erhält man folgendes Resultat.

Satz 2.3.4 *Sei $A \in \mathbb{R}^{n,n}$ invertierbar. Wendet man das Gauß-Verfahren auf einem Rechner mit Maschinengenauigkeit eps mit einer Pivot-Technik an, die $|l_{ij}| \leq 1$ sicherstellt (z.B. Spaltenpivotsuche oder totale Pivotsuche), dann errechnet man $\bar{L}$, $\bar{R}$ mit*

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

Hierbei sind P, Q die aus der Pivotsuche resultierenden Permutationen und

$$\bar{a} = \max_k \bar{a}_k, \quad \bar{a}_k = \max_{i,j} |a_{ij}^{(k)}|. \tag{2.14}$$

Berechnet man mit Hilfe von $\bar{L}$, $\bar{R}$ durch Vorwärts- und Rückwärtssubstitution eine Näherungslösung $\bar{x}$ von $Ax = b$, dann existiert eine Matrix E mit

$$(A + E)\bar{x} = b, \quad |e_{ij}| \le \frac{2(n+1)eps}{1 - neps}(|\bar{L}||\bar{R}|)_{ij} \le \frac{2(n+1)eps}{1 - neps} n\bar{a}.$$

Hierbei bezeichnet $|\bar{L}| = (|\bar{l}_{ij}|)$, $|\bar{R}| = (|\bar{r}_{ij}|)$.

Beweis: Siehe Stoer [St94]. □

Bemerkung: Mit Satz 2.3.3 kann man nun auch den relativen Fehler der Näherungslösung $\bar{x}$ abschätzen. □

Einfluß der Pivot-Strategie

Die Größe von $\bar{a}$ in (2.14) hängt von der Pivotstrategie ab. Man kann folgendes zeigen:

- **Spaltenpivotsuche:** $\bar{a}_k \le 2^k \max_{i,j} |a_{ij}|$.

  Diese Schranke kann erreicht werden, ist aber in der Regel viel zu pessimistisch. In der Praxis tritt fast immer $\bar{a}_k \le 10 \max_{i,j} |a_{ij}|$ auf.

- **Spaltenpivotsuche bei Tridiagonalmatrizen:** $\bar{a}_k \le 2 \max_{i,j} |a_{ij}|$.

- **Vollständige Pivotsuche:** $\bar{a}_k \le f(k) \max_{i,j} |a_{ij}|, \quad f(k) = k^{1/2}(2^1 3^{1/2} \cdots k^{1/(k-1)})^{1/2}$.

  $f(n)$ wächst recht langsam. Es ist bislang kein Beispiel mit $\bar{a}_k \ge (k+1) \max_{i,j} |a_{ij}|$ entdeckt worden.

Kapitel 3

Lineare Gleichungssysteme: Iterative Verfahren

Wir hatten in Kapitel 2 direkte Verfahren zur Lösung eines linearen Gleichungssystems betrachtet. In manchen Fällen sind jedoch iterative Verfahren vorzuziehen. Dies trifft inbesondere auf gewisse sehr große Gleichungssysteme mit dünn besetzter Koeffizientenmatrix zu (d.h. der Anteil der Nichtnullen in der Koeffizientenmatrix ist klein). Die Anwendung direkter Verfahren führt dann meist zu viel weniger dünn besetzten Faktoren in der Dreieckszerlegung. Durch geeignete Datenstrukturen kann hingegen Av in einem Rechenaufwand $O(\text{Zahl der Nichtnullen})$ berechnet werden. Dies macht iterative Verfahren attraktiv, die im Wesentlichen nur Produkte Av benötigen. Große dünn besetzte Matrizen treten z.B. bei der Diskretisierung von Randwert- und Anfangsrandwertproblemen partieller Differentialgleichungen auf. In diesem Fall ist die Koeffizientenmatrix oft symmetrisch und positiv definit oder eine M-Matrix. Wir werden sehen, dass sich solche Systeme sehr effizient durch iterative Verfahren lösen lassen, solange man keine zu hohe Anforderung an die Genauigkeit der Lösung stellt. Dies ist aber oft der Fall, da das Gleichungssystem selbst nur eine Approximation der tatsächlichen Lösung eines Randwertproblems etc. liefert und nur mit einer der Approximationsgüte vergleichbaren Genauigkeit gelöst werden muss.

3.1 Konstruktion von Iterationsverfahren

3.1.1 Einführung

Wir betrachten wieder das reelle lineare Gleichungssystem

$$Ax = b \tag{3.1}$$

mit $A \in \mathbb{R}^{n,n}$, $b \in \mathbb{R}^n$. Wir nehmen in diesem Abschnitt an, dass A invertierbar ist. Dann hat (3.1) die eindeutige Lösung

$$\bar{x} = A^{-1}b.$$

Zur Lösung von (3.1) betrachten wir Iterationsverfahren der Form

$$x^{(k+1)} = \Phi(x^{(k)}), \quad k = 0, 1, \dots$$

mit einem Startpunkt $x^{(0)} \in \mathbb{R}^n$. Für eine beliebige nichtsinguläre Matrix $B \in \mathbb{R}^{n,n}$ erhält man solche Iterationsvorschriften aus der Gleichung

$$Bx + (A - B)x = b, \ \text{ also } \quad Bx = (B - A)x + b$$

indem man setzt

$$Bx^{(k+1)} = (B - A)x^{(k)} + b. \tag{3.2}$$

Auflösen nach $x^{(k+1)}$ ergibt dann

$$x^{(k+1)} = x^{(k)} - B^{-1}(Ax^{(k)} - b) = (I - B^{-1}A)x^{(k)} + B^{-1}b =: \Phi(x^{(k)}). \tag{3.3}$$

Offensichtlich ist die Lösung $\bar{x}$ von (3.1) der einzige Fixpunkt von (3.3), es gilt also

$$x = (I - B^{-1}A)x + B^{-1}b \tag{3.4}$$

genau für $x = \bar{x}$. Die Differenz von (3.3) und (3.4) zeigt, dass der Fehler $r^{(k)} = x^{(k)} - \bar{x}$ die Iterationsvorschrift erfüllt

$$r^{(k+1)} = (I - B^{-1}A)r^{(k)}.$$

Insbesondere folgt für jede Vektornorm $\|\cdot\|$ mit induzierter Matrixnorm $\|\cdot\|$

$$\|r^{(k+1)}\| \le \|I - B^{-1}A\| \|r^{(k)}\| \tag{3.5}$$

Jede Wahl einer invertierbaren Matrix B führt auf ein mögliches Iterationsverfahren (3.3). Es wird umso brauchbarer sein, je besser es die folgenden Eigenschaften erfüllt:

a) Die Gleichung (3.2) ist leicht nach $x^{(k+1)}$ auflösbar.

b) Für eine induzierte Matrixnorm sollte $\|I - B^{-1}A\|$ kleiner 1 und möglichst klein sein.

Forderung a) läßt sich sicherstellen, wenn B z.B. eine Diagonal- oder Dreiecksmatrix ist. b) wird umso besser erfüllt sein, je genauer B und A übereinstimmen.

Bevor wir die Konvergenztheorie eines Iterationsverfahrens (3.3) untersuchen, geben wir noch einige wichtige Verfahren an.

3.1.2 Wichtige Iterationsverfahren

Zur einfachen Beschreibung von Iterationsverfahren verwenden wir folgende Standardzerlegung von A

$$A = D - L - U$$

mit

$$D = \begin{pmatrix} a_{11} & & & 0 \\ & \ddots & & \\ & & \ddots & \\ 0 & & & a_{nn} \end{pmatrix}, \quad L = -\begin{pmatrix} 0 & \cdots & \cdots & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix},$$

$$U = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{pmatrix}$$

Wir gehen im Folgenden davon aus, dass D invertierbar ist (also alle Diagonaleinträge von A nicht verschwinden). Dies ist z.B. bei positiv definiten oder strikt diagonaldominanten Matrizen sowie für M-Matrizen der Fall.

Jacobi-Verfahren

Beim *Gesamtschrittverfahren* oder *Jacobi-Verfahren* verwendet man

$$B := D.$$

Man erhält also mit (3.2) die Iterationsvorschrift

$$Dx^{(k+1)} = (L + U)x^{(k)} + b. \tag{3.6}$$

Komponentenweise ergibt sich somit

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n, \quad k = 0, 1, \ldots.$$

Bemerkung: Die Komponenten $x_i^{(k+1)}$ können unabhängig voneinander berechnet werden. Das Jacobi-Verfahren (3.6) läßt sich also effizent auf einem Parallelrechner implementieren. □

Gauß-Seidel-Verfahren

Beim *Gauß-Seidel-* oder *Einzelschrittverfahren* wählt man für B das untere Dreieck von A mit Diagonale, also

$$B = D - L.$$

Man erhält so für (3.2) die Iteration

$$(D - L)x^{(k+1)} = Ux^{(k)} + b.$$

Um $D - L$ nicht invertieren zu müssen, rechnet man mit der bequemeren Form

$$x^{(k+1)} = D^{-1}(Lx^{(k+1)} + Ux^{(k)} + b)$$

und komponentenweise

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n, \quad k = 0, 1, \ldots.$$

SOR-Verfahren

Beim *SOR-Verfahren (successive overrelaxation)* wählt man

$$B = \frac{1}{\omega}(D - \omega L), \;\; 0 < \omega < 2.$$

Nach Multiplikation mit ω erhält man die Vorschrift

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

mit $0 < \omega < 2$. Hieraus ergibt sich

$$x^{(k+1)} = (1-\omega)x^{(k)} + \omega D^{-1}(Lx^{(k+1)} + Ux^{(k)} + b),$$

also komponentenweise

$$\begin{aligned} x_i^{(k+1)} &= (1-\omega)x_i^{(k)} + \omega \frac{1}{a_{ii}} \left(b_i - \sum_{j>i} a_{ij} x_j^{(k)} - \sum_{j<i} a_{ij} x_j^{(k+1)} \right) \\ &= (1-\omega)x_i^{(k)} + \omega x_{i,\text{Einzelschritt}}^{(k+1)}, \quad i = 1, \ldots, n, \quad k = 0, 1, \ldots. \end{aligned}$$

3.2 Konvergenzresultate für Iterationsverfahren

Wir betrachten Iterationsverfahren der Form (3.3). Wir benötigen zunächst die folgende Definition.

Definition 3.2.1 *Das Verfahren* (3.3) *heißt* konvergent, *falls es für jeden Startpunkt* $x^{(0)}$ *eine Folge* $(x^{(k)})$ *erzeugt mit* $\lim_{k\to\infty} x^{(k)} = \bar{x}$ *mit der eindeutigen Lösung* $\bar{x}$ *von* (3.1).

Um die Konvergenzeigenschaften des Verfahrens (3.3) zu untersuchen, ist der Spektralradius der Iterationsmatrix $I - B^{-1}A$ entscheidend.

Definition 3.2.2 *Sei* $A \in \mathbb{R}^{n,n}$ *beliebig mit Eigenwerten* $\lambda_i(A) \in \mathbb{C}$, $i = 1, \dots, n$ *(mit ihren Vielfachheiten gezählt). Dann ist der* Spektralradius $\rho(A)$ *der Matrix A definiert durch*

$$\rho(A) = \max_{1\le i\le n} |\lambda_i(A)|.$$

Es gilt folgender Satz.

Satz 3.2.3 *i) Das Verfahren* (3.3) *ist genau dann konvergent, wenn gilt*

$$\rho(I - B^{-1}A) < 1.$$

ii) Das Verfahren (3.3) *ist insbesondere konvergent, wenn mit einer induzierten Matrixnorm* $\|\cdot\|$ *gilt*

$$\|I - B^{-1}A\| < 1.$$

Für einen Teil des Beweises benötigen wir das folgende wichtige Resultat.

Satz 3.2.4 *Es sei* $A \in \mathbb{R}^{n,n}$ *beliebig. Dann gilt:*

i) Für jedes $\varepsilon > 0$ *existiert eine Vektornorm, so dass mit der induzierten Matrix-Norm* $\|\cdot\|$ *gilt*

$$\|A\| \le \rho(A) + \varepsilon.$$

ii) Für jede von einer Vektornorm induzierte Matrix-Norm $\|\cdot\|$ *gilt*

$$\rho(A) \le \|A\|.$$

Beweis: zu i): Siehe z.B. Stoer und Bulirsch [SB90] oder Törnig und Spellucci [TS88].

zu ii): A hat einen Eigenwert λ mit $|\lambda| = \rho(A)$. Ist v ein zugehöriger Eigenvektor, dann gilt

$$\|A\| = \max_{\|x\|=1} \|Ax\| \ge \left\| A\frac{v}{\|v\|}\right\| = \left\|\lambda \frac{v}{\|v\|}\right\| = |\lambda|.$$

□

Beweis: (von Satz 3.2.3) Wir setzen $G = I - B^{-1}A$.

zu i): "Konvergenz $\Longrightarrow \rho(G) < 1$": Es existiert ein Eigenwert λ von G mit $|\lambda| = \rho(G)$. Sei v ein zugehöriger Eigenvektor, also $v \neq 0$ mit

$$Gv = \lambda v.$$

Mit dem Startpunkt $x^{(0)} = \bar{x} + v$ gilt dann für den Fehler $r^{(k)} = x^{(k)} - \bar{x}$

$$r^{(0)} = v, \quad r^{(k+1)} = Gr^{(k)} = G^{k+1}v = \lambda^{k+1}v.$$

Da das Verfahren konvergent ist, gilt $\lim_{k\to\infty} r^{(k)} = 0$. Dies erfordert $\lim_{k\to\infty} \lambda^k = 0$, also $|\lambda| < 1$.

"$\rho(G) < 1 \Longrightarrow$ Konvergenz": Wir finden $\varepsilon > 0$ mit $\mu := \rho(G) + \varepsilon < 1$. Nach Satz 3.2.4, i) existiert eine induzierte Matrixnorm $\|\cdot\|$ mit $\|G\| \leq \rho(G) + \varepsilon = \mu < 1$. Für den Fehler gilt also

$$\|r^{(k+1)}\| = \|Gr^{(k)}\| \leq \|G\|\|r^{(k)}\| \leq \mu\|r^{(k)}\|,$$

und das ergibt $\|r^{(k)}\| \leq \mu^k\|r^{(0)}\| \to 0$ für $k \to \infty$.

zu ii): Nach Satz 3.2.4, ii) gilt dann

$$\rho(G) \leq \|G\| < 1$$

und die Konvergenz folgt aus i). □

Wir wenden dieses Resultat zunächst auf das Jacobi-Verfahren an. Dann ist $B = D$, also

$$I - B^{-1}A = I - D^{-1}A = D^{-1}(L + U).$$

Dann gilt für die Zeilensummennorm $\|\cdot\|_\infty$

$$\|I - B^{-1}A\|_\infty = \max_i \sum_{j\neq i} \frac{|a_{ij}|}{|a_{ii}|} = \max_i \frac{\sum_{j\neq i}|a_{ij}|}{|a_{ii}|}. \tag{3.7}$$

Wir erhalten unmittelbar den ersten Teil des folgenden Satzes. Der zweite Teil ergibt sich für die Spaltensummennorm $\|I - B^{-1}A\|_1$ anstelle $\|I - B^{-1}A\|_\infty$.

Satz 3.2.5 *i) (Starkes Zeilensummenkriterium). Das Jacobi-Verfahren ist konvergent, wenn A strikt diagonaldominant ist.*

ii) (Starkes Spaltensummenkriterium). Das Jacobi-Verfahren ist konvergent, wenn A^T strikt diagonaldominant ist.

iii) Ist A strikt diagonaldominant, dann ist auch das Gauß-Seidel-Verfahren und das SOR-Verfahren für $0 < \omega \leq 1$ konvergent.

Beweis: zu i): Ist A strikt diagonaldominant, dann folgt aus (3.7) sofort $\|I - B^{-1}A\|_\infty < 1$, und somit ist das Jacobi-Verfahren konvergent nach Satz 3.2.3, ii).

zu ii): Ist A^T strikt diagonaldominant, dann folgt analog $\|I - B^{-1}A\|_1 < 1$.

zu ii): Siehe z.B. Törnig und Spellucci [TS88]. □

Man kann dies noch verschärfen für irreduzible Matrizen:

Definition 3.2.6 *Eine Matrix $A \in \mathbb{R}^{n,n}$ heißt* reduzibel*, falls es eine Permutationsmatrix P gibt mit*

$$P^T AP = \begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix}$$

mit quadratischen Matrizen $B_{11} \in \mathbb{R}^{p,p}$, $B_{22} \in \mathbb{R}^{q,q}$, $p + q = n$, $p > 0$, $q > 0$.

Ist dies nicht der Fall, dann heißt A irreduzibel*. A heißt* irreduzibel-diagonaldominant*, wenn A irreduzibel und diagonaldominant ist und es ein i gibt mit*

$$|a_{ii}| > \sum_{j \neq i} |a_{ij}|.$$

Satz 3.2.7 *Satz 3.2.5 gilt auch, falls A irreduzibel-diagonaldominant ist.*

Bisher haben wir für das SOR-Verfahren nur Konvergenzresultate im Fall $0 < \omega \leq 1$, wobei $\omega = 1$ in der Praxis die schnellste Konvergenz liefert. Interessant ist also eigentlich der Fall $1 < \omega < 2$ (overrelaxation).

Wir erinnern an die

Definition 3.2.8 *A ist eine M-Matrix, wenn gilt*

$$a_{ii} > 0, \quad i = 1, \ldots, n, \qquad a_{ij} \leq 0, \quad i \neq j.$$

und zudem $\rho(D^{-1}(L + U)) < 1$ (d.h. das Jacobi-Verfahren ist konvergent).

M-Matrizen treten sehr häufig auf, z.B. bei vielen Diskretisierungen von Randwertproblemen und Anfangsrandwertproblemen für partielle Differentialgleichungen.

Es gelten folgende wichtige Konvergenzsätze für das SOR-Verfahren.

Satz 3.2.9 (Varga) *Ist A eine irreduzible M-Matrix, dann ist der Spektralradius der Iterationsmatrix des SOR-Verfahrens*

$$\rho((D - \omega L)^{-1}((1 - \omega)D + \omega U)) < 1$$

und monoton fallend in $0 < \omega \leq \omega_0$ mit einem $\omega_0 > 1$. Insbesondere konvergiert das SOR-Verfahren für $0 < \omega \leq \omega_0$.

Hierbei ist ω_0 leider unbekannt. Weiter gilt

Satz 3.2.10 *Es sei A symmetrisch und positiv definit. Dann konvergiert das SOR-Verfahren für $0 < \omega < 2$.*

Beweis: Siehe z.B. Törnig und Spellucci [TS88]. $\square$

Zusatz: Für *konsistent geordnete* Matrizen läßt sich der optimale Relaxationsparameter ω_0 explizit angeben. Eine Matrix $A \in \mathbb{R}^{n,n}$ heißt *konsistent geordnet* wenn die Eigenwerte der Matrix

$$J(\alpha) = D^{-1}(\alpha L + \alpha^{-1} U)$$

für $\alpha \neq 0$ nicht von α abhängen.

Ist $G_{SOR}(\omega)$ die Iterationsmatrix des SOR-Verfahrens und $G_J = (I - D^{-1}A)$ die des Jacobi-Verfahrens, dann gilt zudem der Satz

Satz 3.2.11 (Young,Varga)
Es sei $A \in \mathbb{R}^{n,n}$ symmetrisch, positiv definit und konsistent geordnet. Dann gilt

$$\rho(G_{SOR}(\omega_0)) \leq \rho(G_{SOR}(\omega)) < 1, \quad 0 < \omega < 2$$

mit

$$\omega_0 = \frac{2}{1 + \sqrt{1 - \rho(G_J)^2}}.$$

Für Details verweisen wir auf Törnig und Spellucci [TS88], Stoer und Bulirsch [SB90].

Kapitel 4

Interpolation

Gegeben sei eine Ansatzfunktion $\Phi(x; a_0, \ldots, a_n)$, $x \in \mathbb{R}$, die von Parametern $a_0, \ldots, a_n \in \mathbb{R}$ abhängt. In diesem Kapitel beschäftigen wir uns mit der folgenden

Interpolationsaufgabe: Zu gegebenen Paaren

$$(x_i, y_i), \quad i = 0, \ldots, n \text{ mit } x_i, y_i \in \mathbb{R}, \quad x_i \neq x_j \text{ für } i \neq j$$

sollen die Parameter $a_0, \ldots, a_n$ so bestimmt werden, dass die Interpolationsbedingungen gelten

$$\Phi(x_i; a_0, \ldots, a_n) = y_i, \quad i = 0, \ldots, n.$$

Die Paare (x_i, y_i) werden als *Stützpunkte* bezeichnet.

4.1 Polynominterpolation

Hier verwendet man als Ansatzfunktion Polynome vom Grad $\leq n$, also

$$p_n(x) = \Phi(x; a_0, \ldots, a_n) = a_0 + a_1 x + \ldots + a_n x^n.$$

Die Interpolationsaufgabe lautet dann: Finde ein Polynom $p_n(x)$ vom Grad $\leq n$, das die Interpolationsbedingungen erfüllt

$$p_n(x_i) = y_i, \quad i = 0, \ldots, n. \tag{4.1}$$

4.1.1 Interpolationsformel von Lagrange

Als einfachste Lösung bietet sich folgendes Vorgehen an: Wir betrachten das

Lagrangesche Interpolationspolynom

$$p_n(x) = \sum_{i=0}^{n} y_i L_{i,n}(x) \quad \text{mit} \quad L_{i,n}(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \tag{4.2}$$

Die Lagrange-Polynome sind gerade so gewählt, dass gilt

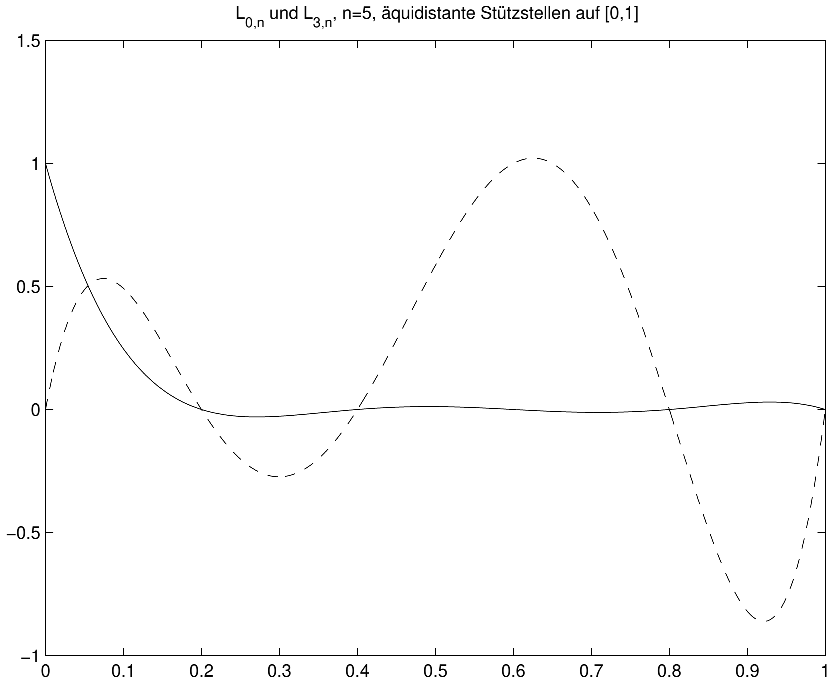
$$L_{i,n}(x_k) = \begin{cases} 1 & \text{falls } k = i, \\ 0 & \text{sonst.} \end{cases}$$

Daher erfüllt p_n in (4.2) die Interpolationsbedingungen (4.1). Tatsächlich ist dies die einzige Lösung der Interpolationsaufgabe:

Satz 4.1.1 *Es gibt genau ein Polynom $p_n(x)$ vom Grad $\leq n$, das die Interpolationsbedingungen* (4.1) *erfüllt, nämlich* (4.2).

Beweis: Das Polynom (4.2) hat Grad $\leq n$ und erfüllt (4.1). Gäbe es eine weitere Lösung $\tilde{p}_n(x)$, dann ist $p_n(x) - \tilde{p}_n(x)$ ein Polynom vom Grad $\leq n$ mit $n + 1$ verschiedenen Nullstellen $x_0, \ldots, x_n$, muss also identisch 0 sein. $\square$

Bemerkung: (4.2) zeigt, dass p_n linear von y_i abhängt. $\square$



Für theoretische Zwecke ist die Darstellung (4.2) von Lagrange sehr nützlich. Es hat aber den Nachteil, dass es bei Hinzunahme einer Stützstelle völlig neu berechnet werden muss. In der Praxis ist die folgende *Newtonsche Interpolationsformel* angenehmer.

4.1.2 Newtonsche Interpolationsformel

Wir wählen als Ansatz die *Newtonsche Darstellung*

$$p_n(x) = \gamma_0 + \gamma_1(x - x_0) + \gamma_2(x - x_0)(x - x_1) + \ldots + \gamma_n(x - x_0)(x - x_1)\cdots(x - x_{n-1}).$$

Einsetzen in (4.1) liefert nun

$$\begin{aligned}
\gamma_0 &= y_0\\
\gamma_1 &= \frac{y_1 - y_0}{x_1 - x_0}\\
\gamma_2 &= \frac{\frac{y_2-y_1}{x_2-x_1} - \frac{y_1-y_0}{x_1-x_0}}{x_2 - x_0}\\
&\vdots
\end{aligned}$$

Man bezeichnet $f_{[x_0,\ldots,x_i]} := \gamma_i$ als die i-te *dividierte Differenz* zu den Stützstellen $x_0, \ldots, x_i$, wobei $f_{[x_0]} = \gamma_0 = y_0$.

Allgemein berechnen sich die dividierten Differenzen zu den Stützstellen $x_0, \ldots, x_{i+1}$ über die Rekursion

$$f_{[x_0,\ldots,x_{i+1}]} = \frac{f_{[x_1,\ldots,x_{i+1}]} - f_{[x_0,\ldots,x_i]}}{x_{i+1} - x_0} \quad \text{mit} \quad f_{[x_i]} = y_i. \tag{4.3}$$

Man erhält das

Newtonsche Interpolationspolynom

$$p_n(x) = \sum_{i=0}^{n} \gamma_i(x - x_0)\cdots(x - x_{i-1}), \quad \gamma_i = f_{[x_0,\ldots,x_i]} \tag{4.4}$$

mit den dividierten Differenzen $f_{[x_0,\ldots,x_i]}$ aus (4.3).

Begründung: Für $n = 0$ ist die Darstellung klar. Sind $p_{1,\ldots,i+1}$ und $p_{0,\ldots,i}$ die Interpolanten in $x_1, \ldots, x_{i+1}$ bzw. $x_0, \ldots, x_i$ vom Grad $\leq i$, dann gilt

$$\begin{aligned}
p_{i+1}(x) &= \frac{(x - x_0)p_{1,\ldots,i+1}(x) + (x_{i+1} - x)p_{0,\ldots,i}(x)}{x_{i+1} - x_0}\\
&= \frac{f_{[x_1,\ldots,x_{i+1}]} - f_{[x_0,\ldots,x_i]}}{x_{i+1} - x_0}(x - x_0)\cdots(x - x_i) + \underbrace{\text{Polynom vom Grad } i}_{:=q_i(x)}.
\end{aligned}$$

Da der erste Summand in $x_0, \ldots, x_i$ verschwindet, gilt $q_i(x) = p_i(x)$ wegen (4.1). Vergleich mit (4.4) liefert (4.3). $\square$

Wir erhalten also folgende Vorschrift zur Berechnung der Koeffizienten $\gamma_i = f_{[x_0,\ldots,x_i]}$:

Berechnung der dividierten Differenzen:

Setze $f_{[x_j]} = y_j$, $j = 0, \ldots, n$.

Berechne für $k = 1, \ldots, n$ *und* $j = 0, \ldots, n-k$:

$$f_{[x_j,\ldots,x_{j+k}]} = \frac{f_{[x_{j+1},\ldots,x_{j+k}]} - f_{[x_j,\ldots,x_{j+k-1}]}}{x_{j+k} - x_j}.$$

Wir erhalten also das Schema

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

4.1.3 Fehlerabschätzungen

Nimmt man an, dass die Stützwerte von einer Funktion $f : [a, b] \to \mathbb{R}$ herrühren, also

$$y_i = f(x_i), \quad i = 0, \ldots, n,$$

dann erhebt sich die Frage, wie gut das Interpolationspolynom p_n auf $[a, b]$ mit f übereinstimmt. Es gilt der folgende Satz:

Satz 4.1.2 *Sei* f $(n+1)$*-mal stetig differenzierbar, kurz* $f \in C^{n+1}([a, b])$. *Seien* $x_0, \ldots, x_n \in [a, b]$ *verschiedene Punkte und sei* p_n *das eindeutige Interpolationspolynom vom Grad* $\leq n$ *zu den Stützwerten* $(x_i, f(x_i))$, $i = 0, \ldots, n$. *Dann existiert zu jedem* $x \in [a, b]$ *ein* $\xi_x \in [a, b]$ *mit*

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

Das Restglied der Interpolation hat also zwei Faktoren: Das sogenannte *Knotenpolynom*

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

und den Faktor $\frac{f^{(n+1)}(\xi_x)}{(n+1)!}$. Durch Abschätzung beider Terme ergibt sich zum Beispiel folgende Fehlerabschätzung.

Korollar 4.1.3 *Unter den Voraussetzungen von Satz 4.1.2 gilt*

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

Achtung: Bei äquidistanter Wahl der Stützpunkte, also $x_i = a + ih$, $h = (b-a)/n$, ist nicht immer gewährleistet, dass gilt

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

Zum Beispiel ist dies für $f(x) = \frac{1}{1+x^2}$ auf $[a,b] = [-5,5]$ der Fall. □

Als Ausweg kann man x_i als die sog. *Tschebyscheff-Abszissen* wählen, für die $\max_{x\in[a,b]} |\omega(x)|$ minimal wird: Wahl der

Tschebyscheffabszissen

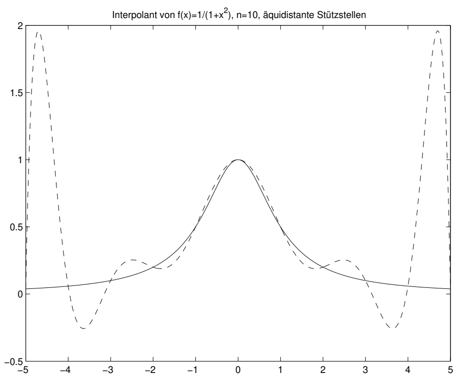
$$x_i = \frac{b-a}{2}\cos\left(\frac{2i+1}{n+1}\frac{\pi}{2}\right) + \frac{b+a}{2}, \quad i = 0,\dots,n. \tag{4.5}$$

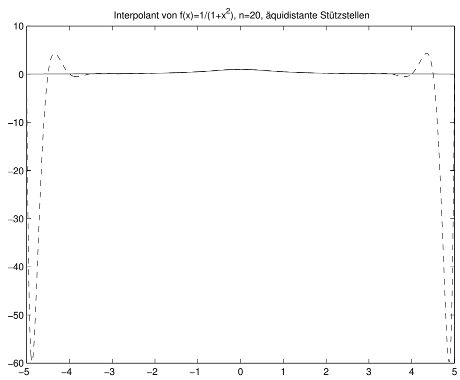
liefert den minimalen Wert für $\max_{x\in[a,b]} |\omega(x)|$, nämlich

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

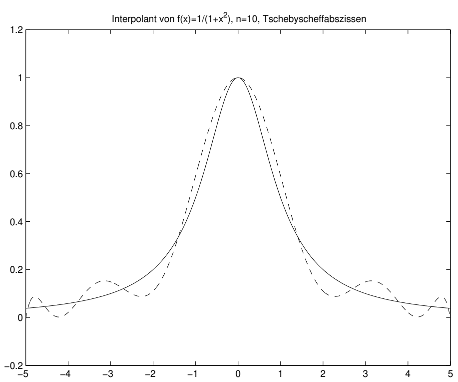
Beipiel: $f(x) = \frac{1}{1+x^2}$ auf $[a,b] = [-5,5]$. Wie bereits erwähnt, geht bei äquidistanten Stützstellen der Fehler $f(x) - p_n(x)$ für $n \to \infty$ nicht an allen Stellen $x \in [a,b]$ gegen 0.

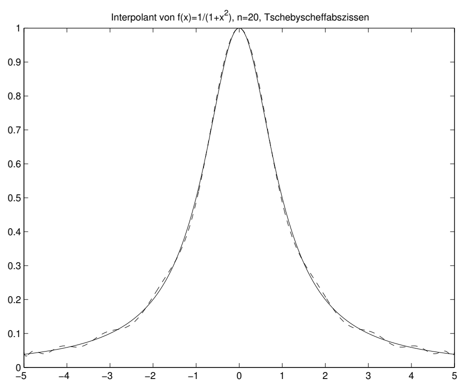
Interpolant bei äquidistanten Stützstellen:





Interpolant bei Tschebyscheffstützstellen:





Allgemein sollte man in der Praxis nicht n sehr groß wählen, sondern besser stückweise in kleinen Intervallen vorgehen, siehe 4.2.

4.1.4 Anwendungen der Polynominterpolation

Wir geben eine Auswahl von Anwendungen für die Polynominterpolation an:

1. **Approximation einer Funktion auf einem Intervall:** Wir haben gesehen, dass hierzu nicht äquidistante Stützstellen sondern die Tschbyscheffabszissen gewählt werden sollten.

2. **Inverse Interpolation:** Sei $f : [a, b] \to \mathbb{R}$ bijektiv, also $f'(x) \neq 0$ auf $[a, b]$. Sind dann (x_i, y_i), $y_i = f(x_i)$, Stützpunkte von f, dann sind (y_i, x_i) wegen $x_i = f^{-1}(y_i)$ Stützpunkte für f^{-1} und eine Approximation von f^{-1} kann durch Interpolation der Stützpunkte (y_i, x_i) gewonnen werden.

3. **Numerische Integration:** (Kapitel 5)
   Zur näherungsweisen Berechnung des Integrals einer Funktion kann man zunächst ein Interpolationspolynom bestimmen, das anschließend einfach integriert werden kann: $\displaystyle\int_a^b f(x)\,dx \approx \int_a^b p_n(x)\,dx.$

4. **Numerische Differentiation:** Mit einem Interpolationspolynom p_n von f ist p_n' eine Approximation von f'.

4.2 Spline-Interpolation

Bei der Polynominterpolation wird die Funktion f auf dem Intervall $[a, b]$ durch *ein* Polynom vom Grad n interpoliert. Wir hatten festgestellt, dass große Genauigkeit nicht immer durch die Wahl vieler Stützstellen sichergestellt werden kann.

Als Ausweg kann man stückweise Interpolation verwenden. Hierbei zerlegt man das Ausgangsintervall $[a, b]$ in kleine Teilintervalle und verwendet auf jedem Teilintervall ein interpolierendes Polynom fester Ordnung. An den Intervallgrenzen sorgt man dafür, dass die Polynome k-mal stetig differenzierbar ineinander übergehen, wobei k fest ist, und die Welligkeit des Interpolanten möglichst klein ist. Dieses Konzept führt auf die Spline-Interpolation.

4.2.1 Grundlagen

Sei

$$\Delta = \{x_i \,:\, a = x_0 < x_1 < \ldots < x_n = b\}$$

eine Zerlegung des Intervalls $[a, b]$. Aus historischen Gründen nennt man die x_i *Knoten*.

Definition 4.2.1 *Eine* Splinefunktion der Ordnung l zur Zerlegung Δ *ist eine Funktion* $s : [a, b] \to \mathbb{R}$ *mit folgenden Eigenschaften*

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

Die Menge dieser Splinefunktionen bezeichnen wir mit $S_{\Delta,l}$.

Im Folgenden betrachten wir nur den Fall $l = 1$ (*lineare Splines*) und $l = 3$ (*kubische Splines*).

Wir wollen nun Splines zur Interpolation verwenden und betrachten folgende Aufgabenstellung:

Spline-Interpolation:

Zu einer Zerlegung $\Delta = \{x_i : a = x_0 < x_1 < \ldots < x_n = b\}$ und Werten $y_i \in \mathbb{R}$, $i = 0, \ldots, n$ bestimme $s \in S_{\Delta,l}$ mit

$$s(x_i) = y_i, \quad i = 0, \ldots, n. \tag{4.6}$$

4.2.2 Interpolation mit linearen Splines

Ein linearer Spline $s \in S_{\Delta,1}$ ist stetig und auf jedem Intervall $[x_i, x_{i+1}]$ ein Polynom s_i vom Grad ≤ 1. Die Interpolationsbedingungen (4.6) erfordern daher $s_i(x_i) = y_i$, $s_i(x_{i+1}) = y_{i+1}$ und legen s_i eindeutig fest zu

$$s(x) = s_i(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} y_i + \frac{x - x_i}{x_{i+1} - x_i} y_{i+1} \quad \forall\, x \in [x_i, x_{i+1}]. \tag{4.7}$$

Definieren wir die "Dachfunktionen"

$$\varphi_i(x) = \begin{cases} 0 & \text{falls } x < x_{i-1}, \\ \frac{x - x_{i-1}}{x_i - x_{i-1}} & \text{falls } x \in [x_{i-1}, x_i], \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & \text{falls } x \in [x_i, x_{i+1}], \\ 0 & \text{falls } x > x_{i+1}. \end{cases}$$

mit beliebigen Hilfsknoten $x_{-1} < a$ und $x_{n+1} > b$, dann erhalten wir für $s(x)$ auf $[a, b]$ die bequeme Darstellung

$$s(x) = \sum_{i=0}^{n} y_i \varphi_i(x), \quad x \in [a, b].$$

Satz 4.2.2 *Zu einer Zerlegung* $\Delta = \{x_i : a = x_0 < x_1 < \ldots < x_n = b\}$ *von* $[a, b]$ *und Werten* y_i*,* $i = 0, \ldots, n$*, existiert genau ein interpolierender linearer Spline.*

Ferner gilt folgende Fehlerabschätzung.

Satz 4.2.3 *Sei* $f \in C^2([a, b])$*. Dann gilt für jede Zerlegung* $\Delta = \{x_i\ ;\ a = x_0 < x_1 < \ldots < x_n = b\}$ *von* $[a, b]$ *und den zugehörigen interpolierenden linearen Spline* $s \in S_{\Delta,1}$ *von* f

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

Beweis: Auf jedem Intervall $[x_i, x_{i+1}]$ ist s ein interpolierendes Polynom vom Grad ≤ 1. Daher gilt nach Satz 4.1.2

$$|f(x) - s(x)| = \frac{|f''(\xi)|}{2!}(x_{i+1} - x)(x - x_i) \le \frac{|f''(\xi)|}{2!} \frac{h_{max}^2}{4} \quad \forall\, x \in [x_i, x_{i+1}]$$

mit einem $\xi \in [x_i, x_{i+1}]$. Daraus folgt unmittelbar die Behauptung. $\Box$

4.2.3 Interpolation mit kubischen Splines

Kubische Splines sind zweimal stetig differenzierbar aus kubischen Polynomen zusammengesetzt. Wir werden sehen, dass die Interpolation mit kubischen Splines es gestattet, gegebene Punkte durch eine Funktion minimaler Krümmung zu interpolieren.

Berechnung kubischer Spline-Interpolanten

Ist $s \in S_{\Delta,3}$ ein kubischer Spline, dann ist s'' offensichtlich stetig und stückweise linear, also $s'' \in S_{\Delta,1}$. Es bietet sich daher an, s_i durch Integration von s_i'' zu bestimmen.

Seien $M_i = s_i''(x_i)$. Man nennt M_i *Momente*. Dann gilt nach (4.7)

$$s_i''(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} M_i + \frac{x - x_i}{x_{i+1} - x_i} M_{i+1}.$$

Zweifache Integration ergibt dann den 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 Konstanten $c_i, d_i \in \mathbb{R}$. Wir berechnen c_i und d_i aus den Bedingungen

$$s_i(x_i) = y_i, \quad s_i(x_{i+1}) = y_{i+1}.$$

Mit

$$h_i = x_{i+1} - x_i$$

liefert dies

$$d_i = y_i - \frac{h_i^2}{6} M_i, \quad c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{h_i}{6}(M_{i+1} - M_i).$$

Einsetzen in die Gleichungen $s_i'(x_i) = s_{i-1}'(x_i)$ ergibt schließlich folgende Gleichungen für die Momente M_i:

$$\text{(4.8)} \quad \frac{h_{i-1}}{6} M_{i-1} + \frac{h_{i-1} + h_i}{3} M_i + \frac{h_i}{6} M_{i+1} = \frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}, \quad i = 1, \dots, n-1.$$

Dies sind $n-1$ Gleichungen für $n+1$ Unbekannte. Der Spline-Interpolant wird eindeutig durch zwei zusätzlich Randbedingungen:

Wichtige Randbedingungen für kubische Splines:

a) Natürliche Randbedingungen: $s''(a) = s''(b) = 0$, also $M_0 = M_n = 0$

b) Hermite-Randbedingungen: $s'(a) = f'(a)$, $s'(b) = f'(b)$, also

$$\frac{h_0}{3} M_0 + \frac{h_0}{6} M_1 = \frac{y_1 - y_0}{h_0} - f'(a), \quad \frac{h_n}{3} M_n + \frac{h_n}{6} M_{n-1} = f'(b) - \frac{y_n - y_{n-1}}{h_{n-1}}.$$

Für jeden der Fälle a)-b) ergibt sich zusammen mit (4.8) eine eindeutige Lösung für $M_0, \dots, M_n$.

Für a) und b) erhält man ein strikt diagonaldominantes tridiagonales Gleichungssystem der Form

$$\text{(4.9)} \quad \begin{pmatrix} \mu_0 & \lambda_0 & & & & \\ \frac{h_0}{6} & \frac{h_0+h_1}{3} & \frac{h_1}{6} & & & \\ & \ddots & \ddots & \ddots & & \\ & & \frac{h_{i-1}}{6} & \frac{h_{i-1}+h_i}{3} & \frac{h_i}{6} & \\ & & & \ddots & \ddots & \ddots \\ & & & & \lambda_n & \mu_n \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ \vdots \\ M_n \end{pmatrix} = \begin{pmatrix} b_0 \\ \frac{y_2-y_1}{h_1} - \frac{y_1-y_0}{h_0} \\ \vdots \\ \frac{y_{i+1}-y_i}{h_i} - \frac{y_i-y_{i-1}}{h_{i-1}} \\ \vdots \\ b_n \end{pmatrix}.$$

Für a) kann man zum Beispiel $b_0 = b_n = \lambda_0 = \lambda_n = 0$ und $\mu_0 = \mu_n = 1$ wählen. Wegen der strikten Diagonaldominanz ist nach dem Satz von Gershgorin 0 kein Eigenwert und daher ist die Koeffizientenmatrix invertierbar.

Minimaleigenschaften kubischer Splines

Es zeigt sich, dass der kubische Spline-Interpolant mit Randbedingung a) oder b) unter allen zweimal stetig differenzierbaren minimale Krümmung im folgenden Sinne hat:

Satz 4.2.4 *Gegeben sei eine beliebige Funktion $f \in C^2([a,b])$ und eine Unterteilung Δ von $[a,b]$. Dann gilt für den kubischen Spline-Interpolanten $s \in S_{\Delta,3}$ mit Randbedingungen a) oder b)*

$$\int_a^b f''(x)^2\,dx = \int_a^b s''(x)^2\,dx + \int_a^b (f''(x) - s''(x))^2\,dx \geq \int_a^b s''(x)^2\,dx.$$

Beweis: Siehe zum Beispiel [St94], [Pl00]. □

Fehlerabschätzung für kubische Spline-Interpoaltion

Unter Verwendung der Tatsache, dass die Momente $\hat{M}_i = f''(x_i)$ das Gleichungssystem (4.9) auf $O(h_{max}^3)$ mit $h_{max} = \max_{0 \leq i < n} h_i$ erfüllen und die Norm der Inversen der Koeffizientenmatrix in (4.9) von der Ordnung $O(1/h_{min})$ ist mit $h_{min} = \min_{0 \leq i < n} h_i$, kann man folgendes Resultat zeigen.

Satz 4.2.5 *Sei $f \in C^4([a,b])$ mit $f''(a) = f''(b) = 0$. Dann gibt es eine Konstante $C > 0$, so dass für jede Unterteilung Δ mit dem kubischen Spline-Interpolanten $s \in S_{\Delta,3}$ zu Randbedingungen a) gilt*

$$|f^{(k)}(x) - s^{(k)}(x)| \leq \frac{C h_{max}}{h_{min}} \sup_{\xi \in [a,b]} |f^{(4)}(\xi)| h_{max}^{4-k}, \quad k = 0, 1, 2.$$

Beweis: Siehe zum Beispiel [Pl00]. □

Kapitel 5

Numerische Integration

In diesem Kapitel stellen wir einige wichtige Verfahren zur näherungsweisen Berechnung bestimmter Integrale $\int_a^b f(x)\,dx$ vor.

Integrationsaufgabe:

Zu gegebenem integrierbarem $f : [a,b] \to \mathbb{R}$ berechne

$$I(f) = \int_a^b f(x)\,dx.$$

Schon für relativ einfache Funktionen läßt sich die Stammfunktion nicht analytisch angeben, etwa $\frac{\sin x}{x}$ und e^{-x^2}. Man ist dann auf numerische Integrationsverfahren angewiesen.

Wichtige numerische Integrationsverfahren beruhen darauf, f mit Hilfe einiger Stützpunkte $(x_i, f(x_i))$, $x_i \in [a,b]$ durch ein Polynom p_n zu interpolieren und dann dieses zu integrieren. Diese Vorgehensweise liefert die Integralapproximation

$$I_n(x) = \int_a^b p_n(x)\,dx$$

und wird als *interpolatorische Quadratur* bezeichnet.

5.1 Newton-Cotes-Quadratur

5.1.1 Geschlossene Newton-Cotes-Quadratur

Wir wählen für $n \in \mathbb{N}$ die äquidistanten Stützstellen

$$x_i = a + ih, \quad i = 0, \ldots, n, \qquad \text{mit } h = \frac{b-a}{n}.$$

Dann lautet das Interpolationspolynom p_n in Lagrange-Darstellung

$$p_n(x) = \sum_{i=0}^{n} f(x_i) L_{i,n}(x), \quad L_{i,n}(x) = \prod_{\substack{j=0\\j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

Wir erhalten nun die numerische Quadraturformel

$$I_n(f) = \int_a^b p_n(x)\,dx = \sum_{i=0}^{n} f(x_i) \int_a^b L_{i,n}(x)\,dx.$$

Mit der Substitution $x = a + sh$ und $s \in [0, n]$ ergibt sich die

Geschlossene Newton-Cotes Formel:

$$I_n(f) = h \sum_{i=0}^{n} \alpha_{i,n} f(x_i),$$

(5.1)
$$\text{mit} \quad \alpha_{i,n} = \int_0^n \prod_{\substack{j=0\\j\neq i}}^{n} \frac{s - j}{i - j}\,ds.$$

Die Zahlen $\alpha_{0,n}, \ldots, \alpha_{n,n}$ heißen *Gewichte*. Sie sind *unabhängig* von f und $[a, b]$ und somit tabellierbar. Es gilt stets

$$\sum_{i=0}^{n} h\alpha_{i,n} = b - a.$$

Definition 5.1.1 *Eine Integrationsformel $J(f) = \sum_{i=0}^{n} \beta_i f(x_i)$ heißt* exakt vom Grad n, *falls sie alle Polynome bis mindestens vom Grad n exakt integriert.*

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

Es ist wichtig, eine Abschätzung für den Fehler

$$E_n(f) := I(f) - I_n(f)$$

zur Verfügung zu haben. Nach Korollar 4.1.3 gilt

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

mit einem $\xi \in [a, b]$. Dies ergibt

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

Eine verfeinerte Restgliedabschätzung ergibt sich durch Taylorentwicklung. Dies liefert die folgende Tabelle.

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

Für $n \geq 7$ treten leider negative Gewichte auf und die Formeln werden numerisch unbrauchbar.

5.1.2 Offene Newton-Cotes-Quadratur

Hier wählen wir für $n \in \mathbb{N} \cup \{0\}$ die in $]a, b[$ liegenden äquidistanten Stützstellen

$$x_i = a + ih, \quad i = 1, \dots, n+1, \quad \text{mit } h = \frac{b-a}{n+2}.$$

Geht man völlig analog vor, dann erhält man wiederum interpolatorische Interpolationsformeln, die

Offene Newton-Cotes Formel:

$$\tilde{I}_n(f) == h \sum_{i=1}^{n+1} \tilde{\alpha}_{i,n} f(x_i), \quad \tilde{\alpha}_{i,n} = \int_0^{n+2} \prod_{\substack{j=1 \\ j \neq i}}^{n+1} \frac{s-j}{i-j}\, ds.$$

Für den Quadraturfehler ergeben sich ähnliche Formeln wie im geschlossenen Fall:

n	$\alpha_{i,n}$			Fehler $\tilde{E}_n(f)$	Name
0	2			$\frac{f^{(2)}(\xi)}{3}h^3$	Rechteck-Regel
1	$\frac{3}{2}$	$\frac{3}{2}$		$\frac{3f^{(2)}(\xi)}{4}h^3$	
2	$\frac{8}{3}$	$-\frac{4}{3}$	$\frac{8}{3}$	$\frac{28f^{(4)}(\xi)}{90}h^5$	

5.2 Die summierten Newton-Cotes-Formeln

Die Newton-Cotes-Formeln liefern nur genaue Ergebnisse, solange das Integrationsintervall klein und die Zahl der Knoten nicht zu groß ist. Es bietet sich wieder an, das Intervall $[a, b]$ in kleinere Intervalle zu zerlegen und auf diesen jeweils mit einer Newton-Cotes-Formel zu arbeiten.

Wir zerlegen dazu das Intervall $[a, b]$ in m Teilintervalle der Länge $H = \frac{b-a}{m}$, wenden die Newton-Cotes-Formeln vom Grad n einzeln auf diese Teilintervalle an und summieren die

Teilintegrale auf: Mit

$$N = m \cdot n, \quad H = \frac{b-a}{m}, \quad h = \frac{H}{n}$$
$$x_i = a + ih, i = 0, \dots, N,$$
$$y_j = a + jH, j = 0, \dots, m$$

ergibt sich wegen

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

die

Summierte geschlossene Newton-Cotes-Formel

$$S_N^{(n)}(f) = h \sum_{j=0}^{m-1} \sum_{i=0}^{n} \alpha_{i,n} f(x_{jn+i}).$$

Die Gewichte $\alpha_{i,n}$ ergeben sich wieder aus (5.1). Der Quadraturfehler

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

ergibt sich durch Summation der Fehler auf den Teilintervallen.

Satz 5.2.1 *Sei $f \in C^{n+2}([a,b])$. Dann existiert eine Zwischenstelle $\xi \in]a,b[$ mit*

$$R_N^{(n)}(f) = \begin{cases} C(n) f^{(n+2)}(\xi)(b-a)h^{n+2} & \textit{für gerades } n, \\ C(n) f^{(n+1)}(\xi)(b-a)h^{n+1} & \textit{für ungerades } n. \end{cases}$$

Hierbei ist $C(n)$ eine nur von n abhängige Konstante.

Wir geben noch die gebräuchlichsten summierten Formeln mit Quadraturfehler an:

Summierte Trapezregel: (geschlossen, $n = 1$)

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

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

Summierte Simpson-Regel: (geschlossen, $n = 2$)

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

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

Summierte Rechteck-Regel: (offen, $n = 0$, $2m = N$, $h = \frac{b-a}{N}$)

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

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

Kapitel 6

Nichtlineare Gleichungssysteme

6.1 Einführung

Wir betrachten in diesem Kapitel Verfahren zur Lösung von nichtlinearen Gleichungssystemen.

Nichtlineares Gleichungssystem: Gesucht ist eine Lösung $x \in D$ von

$$F(x) = 0$$

mit einer gegebenen Abbildung

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

$D \subset \mathbb{R}^n$ nichtleer und abgeschlossen.

Viele praxisrelevante Probleme sind nichtlinear und erfordern die Lösung nichtlinearer Gleichungssysteme. So führt zum Beispiel die Diskretisierung nichtlinearer partieller Differentialgleichungen auf große nichtlineare Gleichungssysteme.

Im Gegensatz zu linearen Gleichungssystemen, bei denen nur genau eine, keine oder ein ganzer affiner Unterraum als Lösung auftreten kann, sind bei nichtlinearen Gleichungen auch mehrere oder unendlich viele isolierte Lösungen möglich.

Beispiel 6.1.1

1. $n = 1$, $D = \mathbb{R}$, $F(x) = x^2 - a$, $a > 0$.
   Es gibt zwei reelle Lösungen $x = \pm\sqrt{a}$.

2. $n = 1$, $D = \mathbb{R}$, $F(x) = x^2 + a$, $a > 0$.

   Es existiert keine reelle Lösung.

3. $n = 1$, $D = \mathbb{R}$, $F(x) = x\sin(x)$.

   Es gibt unendlich viele Lösungen $x = k\pi$, $k \in \mathbb{Z}$.

4. Schnittpunkte des Einheitskreises mit der Geraden $G : x_2 = ax_1 + b$, $a, b \in \mathbb{R}$: $n = 2$, $D = \mathbb{R}^2$, $F(x) = \binom{x_1^2+x_2^2-1}{x_2-ax_1-b}$.

   Je nach Wahl von a, b gibt es zwei, eine oder keine relle Lösung.

Sehr oft ist die Funktion F stetig differenzierbar, d.h. die partiellen Ableitungen $\frac{\partial F_i}{\partial x_j}$, $1 \le i, j \le n$ existieren und sind stetig. In diesem Fall gilt (Taylorentwicklung erster Ordnung)

$$F(x+s) = F(x) + F'(x)s + R(x;s)$$

mit der Jacobi-Matrix

$$F'(x) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1}(x) & \cdots & \frac{\partial F_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial F_n}{\partial x_1}(x) & \cdots & \frac{\partial F_n}{\partial x_n}(x) \end{pmatrix}.$$

und einem Restglied $R(x;s)$, wobei

$$\lim_{s \to 0} \frac{\|R(x;s)\|}{\|s\|} = 0, \quad \text{kurz:} \quad R(x;s) = o(\|s\|).$$

Dies ist wesentlich für die Entwicklung schneller Lösungsverfahren.

6.2 Das Newton-Verfahren

Das Newton-Verfahren ist eines der wichtigsten Verfahren zur Lösung nichtlinearer Gleichungssysteme, da es nahe der Lösung sehr schnell konvergiert. Der Einfachheit halber nehmen wir im folgenden den Fall $D = \mathbb{R}^n$ an.

Wir betrachten das Newton-Verfahren zur Lösung eines nichtlinearen Gleichungssystems

$$F(x) = 0 \tag{6.1}$$

mit $F : \mathbb{R}^n \to \mathbb{R}^n$ stetig differenzierbar.

6.2.1 Herleitung des Verfahrens

Anschauliche Herleitung im eindimensionalen Fall

Sei zunächst $n = 1$. Dann ist $F(x)$ eine reelle Funktion. Sei $x^{(k)}$ eine Näherung einer Lösung $\bar{x}$ von (6.1). Die Idee des Newton-Verfahrens besteht darin, F in $x^{(k)}$ durch die Tangente an $(x, F(x))$ im Punkt $x^{(k)}$ zu approximieren und als nächste Iterierte $x^{(k+1)}$ die Nullstelle der Tangente zu wählen.

Die Tangentengleichung lautet

$$y = F(x^{(k)}) + F'(x^{(k)})(x - x^{(k)})$$

und $x^{(k+1)}$ ist die Lösung von

$$F(x^{(k)}) + F'(x^{(k)})(x - x^{(k)}) = 0.$$

Im Falle $F'(x^{(k)}) \neq 0$ ergibt sich

$$x^{(k+1)} = x^{(k)} - F'(x^{(k)})^{-1}F(x^{(k)}).$$

Es gilt also

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

wobei $s^{(k)}$ die Lösung der Gleichung ist

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

Beispiel: Für $F(x) = x^2 - a$, $a > 0$ ergibt sich

$$x^{(k+1)} = x^{(k)} - \frac{1}{2x^{(k)}}((x^{(k)})^2 - a) = \frac{1}{2}\left(x^{(k)} + \frac{a}{x^{(k)}}\right).$$

Der allgemeine Fall

Zur allgemeinen Motivation des Newton-Verfahrens für (6.1) sei $x^{(k)} \in \mathbb{R}^n$ ein gegebener Punkt. Dann ist $\bar{x}$ eine Lösung von (6.1) genau dann, wenn $\bar{x} = x^{(k)} + s$ gilt mit einer Lösung s von

$$F(x^{(k)} + s) = 0. \tag{6.2}$$

Die Idee des Newton-Verfahrens besteht darin, $F(x^{(k)} + s)$ durch die Taylorentwicklung erster Ordnung zu ersetzen: Es gilt

$$F(x^{(k)} + s) = F(x^{(k)}) + F'(x^{(k)})s + o(\|s\|)$$

mit der Jacobi-Matrix $F'(x^{(k)})$ von F in $x^{(k)}$ und das Restglied wird für kurze s klein.

Bei der k-ten Iteration des Newton-Verfahrens ersetzt man daher (6.2) durch die linearisierte Gleichung

$$F(x^{(k)}) + F'(x^{(k)})s = 0.$$

Dies ergibt

Algorithmus 4 Lokales Newton-Verfahren für Gleichungssysteme

Wähle einen Startpunkt $x^{(0)} \in \mathbb{R}^n$.

Für $k = 0, 1, \dots$:

1. *Falls* $F(x^{(k)}) = 0$*: STOP mit Ergebnis* $x^{(k)}$.
2. *Berechne den Newton-Schritt* $s^{(k)} \in \mathbb{R}^n$ *durch Lösen der* Newton-Gleichung
$$F'(x^{(k)})s^{(k)} = -F(x^{(k)}).$$
3. *Setze* $x^{(k+1)} = x^{(k)} + s^{(k)}$.

6.2.2 Superlineare und quadratische lokale Konvergenz des Newton-Verfahrens

Wir werden sehen, dass unter geeigneten Voraussetzungen die schnelle lokale Konvergenz des Newton-Verfahrens gezeigt werden kann.

Wir verwenden im folgenden der Einfachheit halber immer die euklidische Norm $\|\cdot\|_2$ mit induzierter Matrix-Norm $\|\cdot\|_2$, obwohl wir genausogut jede andere Norm verwenden könnten.

Der folgende Satz zeigt die superlineare bzw. quadratische lokale Konvergenz des Newton-Verfahrens.

Satz 6.2.1 (Schnelle lokale Konvergenz des Newton-Verfahrens)
Sei $F : \mathbb{R}^n \to \mathbb{R}^n$ *stetig differenzierbar und sei* $\bar{x} \in \mathbb{R}^n$ *ein Punkt mit* $F(\bar{x}) = 0$ *und* $F'(\bar{x})$ *nichtsingulär. Dann gibt es* $\delta > 0$*, so dass gilt:*

i) $\bar{x}$ *ist die einzige Nullstelle von* F *auf* $B_\delta(\bar{x})$

ii) Für alle $x^{(0)} \in B_\delta(\bar{x})$ *terminiert Algorithmus 4 entweder mit* $x^{(k)} = \bar{x}$ *oder erzeugt eine Folge* $(x^{(k)}) \subset B_\delta(\bar{x})$*, die superlinear gegen* $\bar{x}$ *konvergiert, d.h.*

$$\lim_{k\to\infty} x_k = \bar{x}, \quad \textit{wobei } \|x_{k+1} - \bar{x}\|_2 \le \nu_k \|x_k - \bar{x}\|_2$$

mit einer Nullfolge $\nu_k \searrow 0$.

iii) Ist F' Lipschitz-stetig auf $B_\delta(\bar{x})$ mit Konstante L, gilt also

$$\|F'(x) - F'(y)\|_2 \le L\|x - y\|_2 \quad \forall\, x, y \in B_\delta(\bar{x}),$$

dann konvergiert $(x^{(k)})$ sogar quadratisch gegen $\bar{x}$, d.h.

$$\lim_{k\to\infty} x_k = \bar{x}, \quad \textit{wobei } \|x_{k+1} - \bar{x}\|_2 \le C\|x_k - \bar{x}\|_2^2,$$

wobei $C = \|F'(\bar{x})^{-1}\|_2 L$ gewählt werden kann.

Hinweis: *F' ist automatisch Lipschitz-stetig, falls F zweimal stetig differenzierbar ist.*

Leider konvergiert das Newton-Verfahren aus Algorithmus 4 in der Regel nur für Startpunkte, die nahe genug an einer Lösung $\bar{x}$ liegen.

Beispiel 6.2.1 Betrachte $F(x) = \frac{x}{\sqrt{1+x^2}}$. F hat die eindeutige Nullstelle $\bar{x}$ und ist stetig differenzierbar mit $F'(x) > 0$. Trotzdem konvergiert das Newton-Verfahren für jeden Startpunkt mit $|x^{(0)}| > 1$ nicht. Siehe Übung.

Um Konvergenz für beliebige Startpunkte erzielen zu können, muss man das Newton-Verfahren geeignet globalisieren.

6.2.3 Globalisierung des Newton-Verfahrens

In diesem Abschnitt beschreiben wir eine Modifikation des Newton-Verfahrens, die für eine große Klasse von Funktionen F globale Konvergenz, d.h. Konvergenz von einem beliebigen Startpunkt aus, sicherstellt.

Den Ausgangspunkt bildet die Beobachtung, dass jede Lösung $\bar{x}$ von (6.1) ein globales Minimum des Minimierungsproblems

$$\min_{x\in\mathbb{R}^n} \|F(x)\|_2^2$$

ist.

Wir wenden nun folgende Strategie an:

- Wir verwenden den Newton-Schritt $s^{(k)}$ mit einer Schrittweite $\sigma_k \in]0, 1]$, wählen also als Ansatz für die neue Iterierte

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

- Wir bestimmen die Schrittweite σ_k so, dass gilt

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

  und die Abnahme "ausreichend groß" ist.

Durch Taylorentwicklung der Funktion

$$\phi(\sigma) := \|F(x^{(k)} + \sigma s^{(k)})\|_2^2$$

in $\sigma = 0$ erhält man

$$\phi(\sigma) = \phi(0) + \phi'(0)\sigma + o(\sigma) = \|F(x^{(k)})\|_2^2 + 2\sigma F(x^{(k)})^T F'(x^{(k)})s^k + o(\sigma)$$

und Einsetzen der Newton-Gleichung $F'(x^{(k)})s^k = -F(x^{(k)})$ liefert

$$\|F(x^{(k)} + \sigma s^{(k)})\|_2^2 = \|F(x^{(k)})\|_2^2 - 2\sigma\|F(x^{(k)})\|_2^2 + o(\sigma).$$

Ist $\delta \in]0,1[$ fest, dann gilt im Fall $F(x^{(k)}) \neq 0$ also für σ klein genug

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

Dies zeigt, dass die folgende Schrittweitenwahl nach Armijo Sinn macht:

Schrittweitenwahl nach Armijo:

Sei $\delta \in]0, 1/2[$ (gute Wahl z.B. $\delta = 10^{-3}$) fest gegeben. Wähle das größte $\sigma_k \in \{1, \frac{1}{2}, \frac{1}{4}, \dots\}$ mit

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

Wir erhalten insgesamt folgendes Verfahren:

Algorithmus 5 Globalisiertes Newton-Verfahren für Gleichungssysteme

Wähle einen Startpunkt $x^{(0)} \in \mathbb{R}^n$.

Für $k = 0, 1, \dots$:

1. *Falls $F(x^{(k)}) = 0$: STOP mit Ergebnis $x^{(k)}$.*
2. *Berechne den Newton-Schritt $s^{(k)} \in \mathbb{R}^n$ durch Lösen der* Newton-Gleichung
$$F'(x^{(k)})s^{(k)} = -F(x^{(k)}).$$
3. *Bestimme σ_k nach der Armijo-Regel* (6.4).

4. *Setze* $x^{(k+1)} = x^{(k)} + \sigma_k s^{(k)}$.

Es gilt folgender Konvergenzsatz.

Satz 6.2.2 *Sei* $F : \mathbb{R}^n \to \mathbb{R}^n$ *stetig differenzierbar und* $x^{(0)} \in \mathbb{R}^n$ *beliebig. Ist* $F'(x)$ *invertierbar für alle* x *in der Niveaumenge*

$$N_f(x^{(0)}) := \left\{ y \, : \, f(y) \leq f(x^{(0)}) \right\}, \quad f(x) = \|F(x)\|_2^2$$

und ist $N_f(x^{(0)})$ *kompakt (also beschränkt und abgeschlossen), dann terminiert Algorithmus 5 mit Startpunkt* $x^{(0)}$ *entweder endlich oder erzeugt eine Folge* $(x^{(k)}) \subset N_f(x^{(0)})$*, für die gilt:*

i) $(x^{(k)})$ *konvergiert gegen eine Lösung* $\bar{x}$ *von* (6.1).

ii) Es gibt $l \geq 0$ *mit* $\sigma_k = 1$ *für alle* $k \geq l$*. Das Verfahren geht also in das lokale Newton-Verfahren über und konvergiert superlinear bzw. quadratisch gegen* $\bar{x}$*.*

Literaturverzeichnis

[DB02] P. Deuflhard, F. Bornemann. *Numerische Mathematik II.* de Gruyter, Berlin, 2002.

[Pl00] R. Plato. *Numerische Mathematik kompakt.* Vieweg Verlag, Braunschweig, 2000.

[St94] J. Stoer. *Numerische Mathematik 1.* Springer Verlag, Berlin, 1994.

[SB90] J. Stoer, R. Bulirsch. *Numerische Mathematik 2.* Springer Verlag, Berlin, 1990.

[TS88] W. Törnig, P. Spellucci. *Numerische Mathematik für Ingenieure und Physiker 1.* Springer Verlag, Berlin, 1988.

[TS90] W. Törnig, P. Spellucci. *Numerische Mathematik für Ingenieure und Physiker 2.* Springer Verlag, Berlin, 1990.

[We92] J. Werner. *Numerische Mathematik 2.* Vieweg Verlag, Braunschweig, 1992.