Computerverifikation von Lösungen nichtlinearer Integralgleichungen

Zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften

an der Fakultät für Mathematik der
Universität Karlsruhe (TH)
genehmigte

Dissertation

von

Dipl.-Math. techn. Holger Obermaier

aus Haßmersheim

Tag der mündlichen Prüfung:	23. Juli 2003
Referent:	Prof. Dr. G. Alefeld
Korreferent:	Prof. Dr. M. Plum

An dieser Stelle möchte ich mich bei allen herzlich bedanken, die mich bei der Entstehung der vorliegenden Arbeit unterstützt haben. Mein Dank gilt besonders Herrn Prof. Dr. G. Alefeld, der das Thema dieser Dissertation anregte und der mir die Möglichkeit gab, am Institut für Angewandte Mathematik zu arbeiten, und der damit die Voraussetzungen für das Entstehen dieser Arbeit schuf. Ebenso gilt mein Dank Herrn Prof. Dr. M. Plum, dem ich ebenfalls wertvolle Anregungen verdanke.

Den Herren Dr. J. Mayer und Dr. M. Neher danke ich für das aufmerksame Korrekturlesen des Manuskripts.

Besonderen Dank schulde ich meinen Eltern für deren uneingeschränkte Unterstützung auf meinem Bildungsweg.

Inhaltsverzeichnis

Bezeichnungen

$\mathbb{N}$	Menge der natürlichen Zahlen
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}_+$	Menge der reellen Zahlen, die größer gleich Null sind
$\mathbb{R}^n$	Menge der reellen n-dimensionalen Vektoren
$\mathbb{R}^{n\times m}$	Menge der reellen $n \times m$-Matrizen
$\mathbb{IR}$	Menge der abgeschlossenen reellen Intervalle
$\mathbb{IR}^n$	Menge der abgeschlossenen reellen n-dimensionalen Intervallvektoren
$\mathbb{IR}^{n\times m}$	Menge der abgeschlossenen reellen $n \times m$-Intervallmatrizen
a	reelle Zahl
$\boldsymbol{a} = (a_i)$	reeller Vektor
$\boldsymbol{A} = (a_{i,j})$	reelle Matrix
$[a]$	abgeschlossenes reelles Intervall
$[\boldsymbol{a}] = ([a]_i)$	abgeschlossener reeller Intervallvektor
$[\boldsymbol{A}] = ([a]_{i,j})$	abgeschlossene reelle Intervallmatrix
$\underline{a}$, $\overline{a}$	Unter- bzw. Obergrenze des reellen Intervalls $[a]$
$C(D)$	Raum der auf D stetigen Funktionen.
$C^k(D)$	Raum der auf D k-mal stetig differenzierbaren Funktionen ($k \in \mathbb{N}$)
$C^\lambda(D)$	Raum der auf D Hölder-stetigen Funktionen ($0 < \lambda < 1$)
k_i	partielle Ableitung der Funktion k von n Variablen nach der i-ten Variable
$k_{i,j}$	partielle gemischte Ableitung $(k_i)_j$ der Funktion k von n Variablen
$\|\|\boldsymbol{a}\|\|$	Maximumnorm des Vektors $\boldsymbol{a}$
$\|\|\boldsymbol{A}\|\|$	Zeilensummennorm der Matrix $\boldsymbol{A}$
$\|\|f\|\|$	Supremumnorm der Funktion f
$L(X,Y)$	Raum der beschränkten linearen Operatoren, die den Banachraum X in den Banachraum Y abbilden
$L(X)$	Kurzschreibweise für $L(X,X)$
$K(X,Y)$	Raum der kompakten linearen Operatoren, die den Banachraum X in den Banachraum Y abbilden
$K(X)$	Kurzschreibweise für $K(X,X)$
$\mathcal{F}x$	linearer oder nichtlinearer Operator, angewandt auf x
$\mathcal{F}'[x_0]x$	Fréchetableitung des Operators $\mathcal{F}$ an der Stelle x_0, angewandt auf x
$B(x,r)$	offene Kugel um x mit Radius r
$\bar{B}(x,r)$	abgeschlossene Kugel um x mit Radius r

Die Arbeit ist in Kapitel, Abschnitte und Unterabschnitte gegliedert. Sätze, Theoreme, Definitionen und Bemerkungen werden innerhalb der Abschnitte gemeinsam fortlaufend nummeriert. Theorem 2.2.1 befindet sich somit in Abschnitt 2.2.

Einleitung

Die Fredholmsche Integralgleichung zweiter Art vom Urysohntyp

$$x(s) - \int_a^b k(s,t,x(t))\,dt = y(s), \qquad s \in [a,b] \tag{1}$$

ergibt sich beispielsweise bei der Umformulierung des Dirichletproblems für die Laplacegleichung auf einem Gebiet Ω mittels des Doppelschichtpotentials in eine Integralgleichung über den Rand des Gebietes $\partial\Omega$ (siehe Hackbusch [16, Abschnitt 7.4]). Dabei sind die Integrationsgrenzen $a, b \in \mathbb{R}$, der Kern $k : [a,b] \times [a,b] \times \mathbb{R} \to \mathbb{R}$ und die rechte Seite $y : [a,b] \to \mathbb{R}$ gegeben und genügen meist gewissen Glattheitsvoraussetzungen. Gesucht ist dann eine Funktion $x : [a,b] \to \mathbb{R}$ derart, dass Gleichung (1) erfüllt ist.

Da die Lösung dieser Integralgleichung im Allgemeinen nicht formelmäßig angegeben werden kann, sind wir auf Näherungslösungen angewiesen, welche die exakte Lösung approximieren. Der erste Teil dieser Arbeit besteht nun darin, solch eine Näherungslösung ω zu berechnen. Dazu verwenden wir das Newton-Verfahren im Banachraum $C([a,b])$, das in jedem Iterationsschritt eine lineare Integralgleichung liefert. Um diese lineare Gleichung näherungsweise zu lösen, verwenden wir das Nyström-Verfahren. Die Berechnung der Nyström-Lösung beschleunigen wir mit einem Zwei-Gitter-Verfahren. Die gute Startnäherung, die das Newton-Verfahren zur Konvergenz benötigt, erhalten wir mit einer einfachen Homotopiemethode.

Die Näherungslösung ω ist nur dann nützlich, wenn gewährleistet ist, dass sie eine gewisse Genauigkeit aufweist. Aber die Fehlerabschätzungen, die für die Teilschritte unseres Näherungsverfahrens zur Verfügung stehen, sind nur asymptotischer Art. Auch die Rundungsfehler, die bei der Berechnung auf einem Computer im Allgemeinen auftreten, können wir nur schwer erfassen. Selbst die noch viel grundlegendere Frage, ob überhaupt eine Lösung von (1) existiert, können wir aufgrund fehlender theoretischer Grundlagen meist nicht beantworten. Selbst falls der Defekt der Näherungslösung klein ist, lässt dies keine gesicherten Rückschlüsse auf die Existenz einer Lösung oder die Qualität der Näherungslösung zu.

Aus diesem Grund stellen wir als weiteren wesentlichen Teil unserer Arbeit ein vollständig auf dem Computer durchführbares Verfahren vor, das die Existenz einer Lösung von Gleichung (1) garantiert und eine Schranke α für den Fehler der zuvor berechneten Näherungslösung ω liefert („Verifikation“). Die einzelnen Schritte dieses Verfahrens lehnen sich an die Arbeiten [35, 36, 37] von Plum an, in denen Methoden zum Existenznachweis und zur Einschließung von Lösungen bei gewöhnlichen und partiellen Randwertproblemen vorgestellt werden.

Die Verifikation beginnt mit der Berechnung einer oberen Schranke δ für den Defekt der Näherungslösung ω. Danach bestimmen wir eine Konstante K, die die Norm der Inversen des um die Näherungslösung ω linearisierten Integraloperators nach oben beschränkt. Des Weiteren berechnen wir gewisse Funktionswerte einer Funktion $G : \mathbb{R}_+ \to \mathbb{R}_+$, welche die Linearitätsabweichung der Integralgleichung in einer Umgebung von ω abschätzt (vgl. Abschnitt 2.6). Mit anderen Worten, G beschränkt den Fehler, der dadurch entsteht, dass wir statt des Integrals $\int_a^b k(s,t,\omega(t)+x(t))\,dt$ das Integral $\int_a^b T_1(s,t,x(t))\,dt$ berechnen, wobei T_1 das Taylorpolynom erster Ordnung in der dritten Komponente von k an der Stelle ω ist. Die Abschätzung G soll so gut sein, dass $G(||x||) = o(||x||)$ für $||x|| \to 0$ gilt. Aus technischen Gründen wählen wir G stets monoton wachsend. Abschließend bestimmen wir eine Konstante α, so dass

$$\delta \leq \frac{\alpha}{K} - G(\alpha) \tag{2}$$

erfüllt ist. Diese Konstante α ist dann eine obere Schranke für den Fehler unserer Näherungslösung ω, d. h. es gilt

$$||x - \omega|| \leq \alpha.$$

Insbesondere wird damit auch die Existenz einer Lösung von (1) gezeigt. Falls (1) hingegen nicht lösbar ist, so bricht das Verifikationsverfahren ab. Aber selbst wenn eine Lösung von (1) existiert, ist das Verfahren aufgrund von Rundungsfehlern und Überschätzungen eventuell nicht durchführbar.

Einschließungsverfahren für Fredholmsche Integralgleichungen wurden auch schon in [7, 8, 11, 12, 23, 24] betrachtet. Dobner beschreibt in seiner Habilitationschrift [7] ein Verfahren, bei dem die Fredholmsche Integralgleichung in zwei Integralgleichungen aufspalten wird, von denen eine einen degenerierten, die andere einen kontrahierenden Kern besitzt. Eine Einschließung der ersten Integralgleichung erhält Dobner durch Einschließung der Lösungsmenge eines linearen Gleichungssystems mit Intervallkoeffizienten, eine Einschließung der Lösung der zweiten Integralgleichung durch Fixpunktiteration und Anwendung des Schauderschen Fixpunktsatzes. Klein beschreibt in [23, 24], wie diese Aufspaltung der Integralgleichung mittels der Taylorentwicklung erfolgen kann. Für nichtlineare Integralgleichungen schlägt er ein Newtonartiges Verfahren vor, das dann wieder die Einschließung von Lösungen linearer Integralgleichungen erfordert. In [8] beschreibt Dobner ein Verfahren, das eine Fehlerabschätzung für das Nyström-Verfahren nutzt, um eine Einschließung für die Lösung der Fredholmschen Integralgleichung zu erhalten. Gienger berechnet in [11, 12] zuerst mit Hilfe eines Zwei-Gitter-Verfahrens eine Näherungslösung. Im nachfolgenden Verifikationsschritt schließt er die Lösung der Fredholmschen Integralgleichung unter Verwendung von Abschätzungen über den Quadraturfehler und Einschließungen der Quadraturknoten und Quadraturgewichten ein.

In Kapitel 1 beschreiben wir die mathematischen Grundlagen, die wir für unser Einschließungsverfahren benötigen. Wir stellen die wichtigsten Sätze über lineare Fredholmsche Integralgleichungen vor, beschreiben das Newton-Verfahren und erläutern die Eigenschaften von Quadraturverfahren. Danach geben wir einen kurzen Überblick über die Intervallrechnung und ihre Implementierung auf einem Rechner. Zur Einschließung von Funktionswerten verwenden wir die zentrierte Form mit Steigung, welche meist bessere Einschließungen als die direkte intervallmäßige Auswertung liefert. Mit dem Intervall-Gauß-Algorithmus [1] und dem Verfahren von Rump [42] beschreiben wir zwei Verfahren zur Einschließung der Lösungsmenge von linearen Gleichungssystemen. Im letzten Abschnitt stellen wir die zum Verständnis der verwendeten Parallelisierungen notwendigen parallelen Programmiermodelle vor.

In Kapitel 2 befindet sich das Hauptergebnis dieser Arbeit, das Einschließungsverfahren. Dieses Verfahren formulieren wir zuerst als Algorithmengerüst, danach gehen wir ausführlich auf die einzelnen Schritte ein. Den Hauptsatz dieser Arbeit, der die Existenz und Einschließung der Lösung der Fredholmschen Integralgleichung zweiter Art vom Urysohntyp liefert, beweisen wir in diesem Kapitel. Einen weiteren Schwerpunkt bildet die Parallelisierung des Einschließungsverfahrens.

In Kapitel 3 stellen wir zuerst das Softwarepaket vor, das die einzelnen Schritte des Einschließungsverfahrens aus Kapitel 2 auf dem Computer implementiert. Die rechenintensiven Programmteile liegen sowohl in einer seriellen als auch in einer parallelen Variante vor. Um die Bedienung des Pakets komfortabler zu gestalten, haben wir eine graphische Oberfläche integriert, die zur zentralen Steuerung sowohl der seriellen als auch der parallelen Programmteile dient. Das Softwarepaket kann von der Internetseite des Autors heruntergeladen werden.

Des Weiteren zeigen wir in diesem Kapitel an vier Beispielen die Anwendbarkeit unseres Einschließungsverfahrens. Im ersten Beispiel behandeln wir eine lineare Integralgleichung, die sich aus der Umformulierung des Dirichletproblems der Laplacegleichung mittels des Doppelschichtpotentials in eine Integralgleichung ergibt. Im zweiten Beispiel untersuchen wir eine einfache nichtlineare Integralgleichung mit einem Parameter. Wir betrachten die Integralgleichung für drei unterschiedliche Parameterwerte, für die im ersten Fall genau zwei Lösungen existieren, im zweiten Fall genau eine Lösung und im dritten Fall keine Lösung existiert. Für diese Gleichung können wir optimale Schranken für die einzelnen Schritte des Einschließungsverfahrens berechnen und diese mit den vom Softwarepaket berechneten Werten vergleichen. Das dritte Beispiel untersucht eine nichtlineare Integralgleichung, von der keine Lösung in geschlossener Form bekannt ist. Im letzten Beispiel betrachten wir eine nichtlineare Integralgleichung, bei der die zweite Fréchetableitung nicht existiert.

Kapitel 1

Grundlagen

1.1 Integralgleichungen

Eine Integralgleichung ist eine Gleichung für eine gesuchte Funktion $x(s)$, bei der x auch unter einem Integral erscheint.

Je nach dem Integrationsbereich, über den sich die Integration erstreckt, unterscheidet wir zwei Typen von Integralgleichungen. Bei der *Fredholmschen Integralgleichung* ist der Integrationsbereich fest vorgegeben, wohingegen bei der *Volterraschen Integralgleichung* der Integrationsbereich von der Variablen s abhängt.

Des Weiteren unterscheiden wir zwischen Integralgleichungen erster und zweiter Art. Bei Integralgleichungen erster Art erscheint die unbekannte Funktion x nur unter dem Integral, bei Integralgleichungen zweiter Art auch außerhalb des Integrals.

Die lineare Fredholmsche Integralgleichung zweiter Art hat die Form

$$x(s) - \int_a^b k(s,t)x(t)\,dt = y(s), \qquad s \in [a,b].$$

Dabei heißt die gegebene Funktion $k : [a,b] \times [a,b] \to \mathbb{R}$ *Kern der Integralgleichung* und die gegebene Funktion $y : [a,b] \to \mathbb{R}$ *rechte Seite der Integralgleichung*.

Ein Untertyp der nichtlinearen Fredholmschen Integralgleichung zweiter Art ist die Urysohn-Gleichung

$$x(s) - \int_a^b k(s,t,x(t))\,dt = y(s), \qquad s \in [a,b]$$

und deren Spezialfall, die Hammerstein-Gleichung

$$x(s) - \int_a^b k(s,t)\,g(t,x(t))\,dt = y(s), \qquad s \in [a,b].$$

Die zuvor angegebenen Fredholmschen Integralgleichungen zweiter Art können wir mit

Hilfe der Definition des Fredholmschen Integraloperators $\mathcal{K}$ im linearen Fall durch

$$(\mathcal{K}x)(s) := \int_a^b k(s,t)x(t)\,dt$$

und im Fall der Urysohn-Gleichung durch

$$(\mathcal{K}x)(s) := \int_a^b k(s,t,x(t))\,dt$$

auch als Operatorgleichung

$$(\mathcal{I}-\mathcal{K})x = y$$

schreiben. Die für alle y eindeutige Lösbarkeit der Fredholmschen Integralgleichung zweiter Art ist somit äquivalent zur Existenz des Operators $(\mathcal{I}-\mathcal{K})^{-1}$.

Um lineare Integralgleichungen im abstrakten Rahmen untersuchen zu können, wird im Folgenden der Raum der beschränkten linearen Operatoren $L(X,Y)$ eingeführt. Da dieser Raum aber zu groß ist, um ähnliche Aussagen für die Lösbarkeit von linearen Operatorgleichungen zu erhalten, wie sie von linearen Gleichungssystemen bekannt sind, verwenden wir stattdessen für unsere Überlegungen den Raum der kompakten linearen Operatoren $K(X,Y)$. Mit diesen Hilfsmitteln ausgestattet stellen wir abschließend Eigenschaften der Lösung der linearen Fredholmschen Integralgleichung zweiter Art vor. Die in den folgenden Abschnitten angegebenen Sätze sind beispielsweise im Buch von Hackbusch [16] zu finden.

1.1.1 Der Banachraum $L(X,Y)$

Zunächst benötigen wir folgende

Definition 1.1.1 (Linearer Operator). *Seien X und Y Banachräume. Ein Operator $\mathcal{L}: X \to Y$ heißt* linear*, wenn*

$$\mathcal{L}(x+y) = \mathcal{L}x + \mathcal{L}y \qquad \textit{für alle } x,\ y \in X$$

und

$$\mathcal{L}(\alpha x) = \alpha\mathcal{L}x \qquad \textit{für alle } x \in X,\ \alpha \in \mathbb{R}$$

gelten.

Die Menge der linearen Operatoren, die den Banachraum X in den Banachraum Y abbilden, ist bezüglich der Addition

$$(\mathcal{L}+\mathcal{M})x := \mathcal{L}x + \mathcal{M}x \qquad \text{für alle } x \in X$$

und der Skalarmultiplikation

$$(\alpha\mathcal{L})x := \alpha(\mathcal{L}x) \qquad \text{für alle } x \in X,\ \alpha \in \mathbb{R}$$

ein Vektorraum. Durch

$$||\mathcal{L}||_{X\to Y} := \sup_{x\in X\setminus\{0\}} \frac{||\mathcal{L}x||_Y}{||x||_X}$$

ist auf dem Unterraum $\{\mathcal{L}|\ \mathcal{L}: X \to Y,\ \mathcal{L} \text{ linear},\ ||\mathcal{L}||_{X\to Y} < \infty\}$ eine kanonische Norm gegeben.

Der folgende Satz zeigt eine wichtige Eigenschaft linearer Operatoren, nämlich die Äquivalenz von Stetigkeit und Beschränktheit.

Satz 1.1.2. *Seien X und Y Banachräume. Ein linearer Operator $\mathcal{L}: X \to Y$ ist genau dann stetig, d. h.*

$$\lim_{\bar{x}\to x} \mathcal{L}\bar{x} = \mathcal{L}x \qquad \textit{für alle } x \in X,$$

wenn er beschränkt ist, d. h. wenn

$$||\mathcal{L}||_{X\to Y} < \infty$$

gilt.

Wir haben nun alle Begriffe zusammen, um den Raum $L(X,Y)$ zu definieren. Es zeigt sich, dass es sich dabei um einen Banachraum handelt.

Satz 1.1.3. *Seien X und Y Banachräume. Der Vektorraum*

$$L(X,Y) := \{\mathcal{L}|\ \mathcal{L}: X \to Y,\ \mathcal{L} \textit{ linear},\ ||\mathcal{L}||_{X\to Y} < \infty\}$$

versehen mit der kanonischen Operatornorm ist ein Banachraum.

Für $L(X,X)$ schreiben wir kurz $L(X)$.

Im Hinblick auf Integralgleichungen geben wir im folgenden Satz Bedingungen an, unter denen ein linearer Fredholmscher Integraloperator $\mathcal{K}$ auf dem Banachraum der stetigen Funktionen versehen mit der Supremumnorm $X = (C([a,b]), ||.||_\infty)$ in $L(C([a,b]))$ liegt.

Satz 1.1.4. *Für jedes $x \in C([a,b])$ gelte $\mathcal{K}x \in C([a,b])$. Weiter gelte*

$$\max_{s\in[a,b]} \int_a^b |k(s,t)|\, dt < \infty.$$

Dann gilt $\mathcal{K} \in L(C([a,b]))$ und

$$||\mathcal{K}|| = \max_{s\in[a,b]} \int_a^b |k(s,t)|\, dt.$$

1.1.2 Der abgeschlossene Unterraum $K(X,Y)$ von $L(X,Y)$

In diesem Abschnitt führen wir den Raum der kompakten linearen Operatoren $K(X,Y)$ ein (siehe Hackbusch [16] oder Heuser [17]). Dazu benötigen wir zunächst zwei Definitionen.

Definition 1.1.5 (Präkompaktheit einer Menge). *Sei X ein Banachraum. Eine Teilmenge $M \subseteq X$ heißt* präkompakt*, falls jede Folge (x_k) in M eine in X konvergente Teilfolge (x_{k_i}) besitzt.*

Definition 1.1.6 (Kompaktheit eines Operators). *Seien X und Y Banachräume. Ein Operator $\mathcal{K} \in L(X,Y)$ heißt kompakt, wenn das Bild der abgeschlossenen Einheitskugel $\mathcal{K}\bar{B}_1(0)$ präkompakt ist.*

Mit diesen beiden Definitionen kann sofort der folgende Satz bewiesen werden:

Satz 1.1.7. *Die Menge der* kompakten linearen Operatoren

$$K(X,Y) := \{\mathcal{K} \in L(X,Y)|\ \mathcal{K} \textit{ ist kompakt}\}$$

ist ein abgeschlossener linearer Unterraum von $L(X,Y)$.

Wie schon zuvor schreiben wir für $K(X,X)$ kurz $K(X)$.

Der folgende Satz enthält grundlegende Hilfsmittel, um die Kompaktheit eines Operators zu zeigen.

Satz 1.1.8. *Seien X, Y und Z Banachräume.*

1. *Ist mindestens einer der Operatoren $\mathcal{L} \in L(Y,Z)$ oder $\mathcal{M} \in L(X,Y)$ kompakt, so ist die Komposition $\mathcal{L} \circ \mathcal{M} \in L(X,Z)$ kompakt, d. h. $\mathcal{L} \circ \mathcal{M} \in K(X,Z)$.*

2. *Ist das Bild von $\mathcal{K} \in L(X,Y)$ endlichdimensional, so ist $\mathcal{K}$ kompakt, d. h. $\mathcal{K} \in K(X,Y)$.*

Die Identität auf dem Banachraum X ist nur dann kompakt, wenn X endlichdimensional ist. Der folgende Satz gibt Bedingungen an, unter denen die Einbettung $\mathcal{I} : C^\lambda([a,b]) \to C^\chi([a,b])$ für die unendlichdimensionalen Räume der hölderstetigen Funktionen kompakt ist.

Satz 1.1.9. *Ist $0 \leq \chi < \lambda$, so ist die Einbettung $\mathcal{I} : C^\lambda([a,b]) \to C^\chi([a,b])$, $x \mapsto \mathcal{I}x = x$ kompakt, d. h. $\mathcal{I} \in K(C^\lambda([a,b]), C^\chi([a,b]))$.*

Der folgende Satz liefert Bedingungen für die Kompaktheit des linearen Fredholmschen Integraloperators $\mathcal{K}$.

Satz 1.1.10. *Erfüllt die Kernfunktion k des linearen Fredholmschen Integraloperators die beiden Bedingungen*

$$\int_a^b |k(s,t)|\,dt < \infty \qquad \text{für alle } s \in [a,b],$$

$$\lim_{\bar{s}\to s} \int_a^b |k(\bar{s},t) - k(s,t)|\,dt = 0 \qquad \text{für alle } s \in [a,b],$$

so ist $\mathcal{K}$ kompakt auf $C([a,b])$, d. h. $\mathcal{K} \in K(C([a,b]))$.

Leichter in der Anwendung ist allerdings folgendes Korollar.

Korollar 1.1.11. *Ist der Kern k des linearen Fredholmschen Integraloperators $\mathcal{K}$ stetig, dann ist $\mathcal{K}$ kompakt auf $C([a,b])$, d. h. $\mathcal{K} \in K(C([a,b]))$.*

1.1.3 Eigenschaften der Lösung der linearen Fredholmschen Integralgleichung zweiter Art

Der nächste Satz gewährleistet, dass Gleichungen $(\mathcal{I}-\mathcal{K})x = y$ mit kompaktem Operator $\mathcal{K}$ ein Lösungsverhalten besitzen, das wir aus der Theorie der linearen Gleichungssysteme kennen (siehe Hackbusch [16] oder Heuser [17]).

Satz 1.1.12 (Fredholmsche Alternative). *Sei X ein Banachraum, $\mathcal{K} \in K(X)$ ein Operator und $\mathcal{I} \in L(X)$ die Identität. Dann gelten folgende Alternativen.*

Entweder der Operator $(\mathcal{I}-\mathcal{K})$ besitzt eine beschränkte Inverse $(\mathcal{I}-\mathcal{K})^{-1} : X \to X$, so dass die Gleichung

$$(\mathcal{I}-\mathcal{K})x = y$$

für jedes $y \in X$ eine eindeutige Lösung x besitzt.

Oder die Inverse $(\mathcal{I}-\mathcal{K})^{-1}$ existiert nicht, das bedeutet insbesondere, dass die Gleichung

$$(\mathcal{I}-\mathcal{K})x = y$$

nicht für jede rechte Seite $y \in X$ eine Lösung x besitzt. Im Falle, dass eine Lösung existiert, ist diese nicht mehr eindeutig. Insbesondere besitzt die homogene Gleichung

$$(\mathcal{I}-\mathcal{K})x = 0$$

dann nichttriviale Lösungen. Der Lösungsraum ist endlichdimensional.

Den vorangehenden Satz haben wir natürlich in Hinblick auf lineare Fredholmsche Integralgleichung zweiter Art mit kompaktem linearem Fredholmschen Integraloperator $\mathcal{K}$ formuliert. Wir setzen im Folgenden voraus, dass die Integralgleichung eindeutig lösbar ist, also die erste Alternative gilt, und beschäftigen uns nun mit den Glattheitseigenschaften, d. h. der Regularität der Lösung (siehe Hackbusch [16]). Dazu muss aber zuerst untersucht werden, welche Glattheit die Elemente im Bild von $\mathcal{K}$ aufweisen. Der folgende Satz macht Aussagen darüber.

Satz 1.1.13. *Für den Kern k des Fredholmschen Integraloperators $\mathcal{K}$ gelte $k \in C^\lambda([a,b] \times [a,b])$. Dann ist $\mathcal{K} \in L(C([a,b]), C^\lambda([a,b]))$.*

Definition 1.1.14. *Seien X, Y Banachräume mit $Y \subseteq X$. Dann heißt Y* stetig eingebettet *in X, falls die Einbettung $\mathcal{I} : Y \to X$ stetig ist.*

Wir haben nun alle notwendigen Hilfsmittel zusammen, um Aussagen über die Glattheit der Lösung der linearen Fredholmschen Integralgleichung zweiter Art machen zu können.

Satz 1.1.15 (Regularität). *Der Banachraum Y sei stetig in den Banachraum X eingebettet. Gilt $\mathcal{K} \in L(X, Y)$ und $(\mathcal{I} - \mathcal{K})^{-1} \in L(X)$, so gilt auch*

$$(\mathcal{I} - \mathcal{K})^{-1} \in L(Y).$$

1.2 Das Newton-Verfahren

Im Folgenden wollen wir Lösungen der Gleichung

$$\mathcal{F}x = 0 \tag{1.1}$$

mit einem Operator $\mathcal{F}$, der den Banachraum X in sich abbildet, finden. Dazu werden wir Gleichung (1.1) durch *Linearisierung durch Differentiation* in eine lineare Operatorgleichung überführen, die die ursprüngliche Gleichung in einer Umgebung eines Startwertes x_0 gut approximiert. Das Lösen dieser linearen Operatorgleichung liefert einen Punkt x_1, an dem wir erneut linearisieren können. Die Iteration dieses Vorgehens führt uns zur so genannten *Newton-Folge*. In diesem Abschnitt untersuchen wir die Konvergenz dieser Folge gegen eine Lösung x^*. Zunächst wollen wir aber den Begriff der Differenzierbarkeit von Funktionen, wie er aus der endlichdimensionalen Analysis bekannt ist, auf Operatoren ausdehnen. Die Beweise für die in diesem Abschnitt angegebenen Sätze sind beispielsweise im Buch von Rall [38] oder im Buch von Moore [28] zu finden.

Definition 1.2.1 (Fréchetableitung). *Seien X, Y Banachräume, $D \subseteq X$ eine offene Menge und $\mathcal{F} : D \to Y$ ein Operator. Existiert ein beschränkter linearer Operator $\mathcal{L} : X \to Y$ mit*

$$\lim_{h \to 0} \frac{||\mathcal{F}(x_0 + h) - \mathcal{F}x_0 - \mathcal{L}h||_Y}{||h||_X} = 0,$$

so heißt $\mathcal{F}$ (Fréchet-)differenzierbar *an der Stelle $x_0 \in D$. Der beschränkte Operator*

$$\mathcal{F}'[x_0] := \mathcal{L}$$

wird als erste (Fréchet-)Ableitung *von $\mathcal{F}$ an der Stelle x_0 bezeichnet.*

Ist $\mathcal{F}$ an jeder Stelle $x_0 \in X$ differenzierbar, so erhalten wir einen Operator $\mathcal{F}' : X \to L(X, Y)$, den wir als erste (Fréchet-)Ableitung *von $\mathcal{F}$ bezeichnen. Der Operator $\mathcal{F}$ heißt dann* (Fréchet-)differenzierbar.

Die zweite Fréchetableitung des differenzierbaren Operators $\mathcal{F}$ definieren wir als Fréchetableitung des Operators $\mathcal{F}'$. Das Bild $\mathcal{F}''x$ des Operator $\mathcal{F}'' : X \to L(X, L(X,Y))$ können wir somit für jedes $x \in X$ als bilinearen Operator auffassen.

Analog werden höhere Fréchetableitungen definiert.

Es gelten die folgenden einfachen Rechenregeln für die Fréchetableitung.

Satz 1.2.2. *Seien X, Y und Z Banachräume, $x_0 \in X$.*

a) Sei $\mathcal{L} \in L(X,Y)$. Dann gilt

$$\mathcal{L}'[x_0] = \mathcal{L}.$$

b) Seien $\mathcal{F}$, $\mathcal{G} : X \to Y$ zwei an der Stelle x_0 differenzierbare Operatoren. Dann ist die Summe $\mathcal{F} + \mathcal{G}$ in x_0 differenzierbar und es gilt

$$(\mathcal{F} + \mathcal{G})'[x_0] = \mathcal{F}'[x_0] + \mathcal{G}'[x_0].$$

c) Sei $\mathcal{G} : X \to Y$ ein an der Stelle x_0 und $\mathcal{F} : Y \to Z$ ein an der Stelle $\mathcal{G}x_0$ differenzierbarer Operator. Dann ist die Komposition $\mathcal{F} \circ \mathcal{G}$ differenzierbar in x_0 und es gilt

$$(\mathcal{F} \circ \mathcal{G})'[x_0] = \mathcal{F}'[\mathcal{G}x_0] \circ \mathcal{G}'[x_0].$$

Der nachfolgende Satz gibt uns ein weiteres Hilfsmittel, um die Fréchetableitung zu bestimmen.

Satz 1.2.3. *Seien X, Y Banachräume und $\mathcal{F} : X \to Y$ ein Operator. Der Operator $\mathcal{F}$ ist in $x_0 \in X$ genau dann differenzierbar, wenn es einen beschränkten linearen Operator $\mathcal{L}(x_0) : X \to Y$ und einen in x_0 stetigen Operator $\eta : X \to Y$ mit $\eta(x_0) = 0$ gibt, so dass*

$$\mathcal{F}(x_0 + h) - \mathcal{F}x_0 = (\mathcal{L}(x_0))h + \eta(x_0 + h)||h||_X$$

für alle $h \in X$ gilt. In diesem Fall gilt dann

$$\mathcal{L}(x_0) = \mathcal{F}'[x_0].$$

Nach den kurzen Betrachtungen zu den Eigenschaften der Fréchetableitung wenden wir uns wieder dem ursprünglichen Problem zu, eine Lösung x^* von (1.1) zu finden. Dazu suchen wir zunächst eine lineare Gleichung

$$\mathcal{L}x = y,$$

die die Gleichung (1.1) zumindest in einer Umgebung von x^* gut approximiert. Nach Satz 1.2.3 gilt für einen differenzierbaren Operator $\mathcal{F}$

$$\mathcal{F}x = \mathcal{F}x_0 + \mathcal{F}'[x_0](x - x_0) + \eta(x)||x - x_0||_X. \tag{1.2}$$

Ist x^* eine Lösung von (1.1) und setzen wir $x = x^*$, so geht die Gleichung (1.2) über in

$$\mathcal{F}x_0 + \mathcal{F}'[x_0](x^* - x_0) = -\eta(x^*)||x^* - x_0||_X.$$

Liegt x_0 nahe bei x^*, so ist $||\eta(x^*)||_Y||x^* - x_0||_X$ sehr klein und der Term $\eta(x^*)||x^* - x_0||_X$ kann vernachlässigt werden. Wir setzen daher die rechte Seite Null und erhalten so die lineare Gleichung

$$\mathcal{F}x_0 + \mathcal{F}'[x_0](x - x_0) = 0 \tag{1.3}$$

für die Unbekannte x. In obiger Notation lautet (1.3)

$$\mathcal{L}x = y \qquad \text{mit} \qquad \mathcal{L} := \mathcal{F}'[x_0], \quad y := \mathcal{F}'[x_0]x_0 - \mathcal{F}x_0.$$

Ist Gleichung (1.3) lösbar, so können wir ihre Lösung $x = x_1$ als Approximation von x^* ansehen. Wiederholen wir dieses Vorgehen, d. h. linearisieren wir den Operator $\mathcal{F}$ um x_k und lösen wir die entstehende lineare Operatorgleichung anschließend nach x_{k+1} auf, so erhalten wir die Newton-Iteration

$$x_{k+1} = x_k - (\mathcal{F}'[x_k])^{-1}\mathcal{F}x_k, \qquad k = 0, 1, 2, \ldots, \tag{1.4}$$

bzw. in einer Schreibweise mit Korrekturtermen $\triangle x_k$ die Iteration

$$\begin{aligned} \mathcal{F}'[x_k]\triangle x_k &= -\mathcal{F}x_k, \\ x_{k+1} &= x_k + \triangle x_k, \end{aligned} \qquad k = 0, 1, 2, \ldots .$$

Der iterative Prozess zur Gewinnung der *Newton-Folge* (x_k) wird als *Newton-Verfahren* bezeichnet.

Sind die Operatoren $\mathcal{F}$ und $\mathcal{F}'$ auf dem ganzen Banachraum X definiert, so kann der Prozess, aus x_k ein neues Folgenglied x_{k+1} zu gewinnen, nur dann abbrechen, wenn $(\mathcal{F}'[x_k])^{-1}$ nicht existiert. Für den nächsten Satz nehmen wir an, dass die Newton-Folge wohldefiniert ist und gegen

$$\bar{x} := \lim_{k \to \infty} x_k$$

konvergiert. Der nächste Satz gibt dann Bedingungen an, unter denen $\bar{x}$ eine Lösung von Gleichung (1.1) ist.

Satz 1.2.4. *Der Grenzwert $\bar{x}$ der Newton-Folge löst die Gleichung* (1.1), *wenn mindestens eine der folgenden Bedingungen erfüllt ist:*

- *Der Operator $\mathcal{F}'$ ist stetig bei $x = \bar{x}$,*
- *Es existiert eine Konstante $M > 0$, so dass in einer abgeschlossenen Kugel, die alle Folgenglieder x_k, $k \in \mathbb{N}$, enthält,*

  $$||\mathcal{F}'[x]||_{X \to Y} \leq M$$

  gilt.

- *Es existiert eine Konstante $K > 0$, so dass in einer abgeschlossenen Kugel um x_0, die alle Folgenglieder x_k, $k \in \mathbb{N}$, enthält,*

$$||\mathcal{F}''[x]||_{X \to L(X,Y)} \leq K$$

gilt.

Im folgenden Satz geben wir Voraussetzungen an, unter denen die Folge der Newton-Iterierten wohldefiniert ist und gegen eine Lösung von Gleichung (1.1) konvergiert.

Satz 1.2.5 (Kantorovič). *Der Operator $(\mathcal{F}'[x_0])^{-1}$ existiere, so dass das erste Glied*

$$x_1 = x_0 - (\mathcal{F}'[x_0])^{-1}\mathcal{F}x_0$$

der Newton-Folge definiert ist. Es gelten folgende Voraussetzungen:

1. *Es existiert eine Konstante $\beta_0 > 0$ mit $||(\mathcal{F}'[x_0])^{-1}||_{Y \to X} \leq \beta_0$.*
2. *Es existiere eine Konstante η_0 mit $||x_1 - x_0||_X \leq \eta_0$.*
3. *Es existieren Konstanten $K > 0$ und $r > 0$, so dass*

$$||\mathcal{F}''[x]||_{X \to L(X,Y)} \leq K$$

für alle $x \in \bar{B}(x_0, r)$, der abgeschlossenen Kugel um x_0 vom Radius r, gilt.

4. *Für $h_0 := \beta_0\eta_0 K$ gilt $h_0 \leq \frac{1}{2}$ und es gilt*

$$r \geq r_0 = \frac{1 - \sqrt{1 - 2h_0}}{h_0}\eta_0.$$

Dann ist die Newton-Folge (1.4) wohldefiniert und sie konvergiert ausgehend von x_0 gegen eine in $\bar{B}(x_0, r_0)$ liegende Lösung von (1.1). Insbesondere existieren alle Operatoren $(\mathcal{F}'[x_k])^{-1}$, $k = 1, 2, \ldots$.

Unter gewissen zusätzlichen Voraussetzungen kann nicht nur die Konvergenz des Newton-Verfahrens gegen eine Lösung von (1.1) gezeigt werden, sondern auch, dass diese Lösung in einer gewissen Umgebung eindeutig ist. Wir unterscheiden dabei die beiden Fälle $h_0 < \frac{1}{2}$ und $h_0 = \frac{1}{2}$.

Satz 1.2.6. *Es gelten die Voraussetzungen aus Satz 1.2.5. Zusätzlich gelte $h_0 < \frac{1}{2}$ und $||\mathcal{F}''[x]||_{X \to L(X,Y)} \leq K$ in der offenen Kugel $B(x_0, \tilde{r}_0)$ mit*

$$\tilde{r}_0 = \frac{1 + \sqrt{1 - 2h_0}}{h_0}\eta_0.$$

Dann hat (1.1) eine Lösung in $\bar{B}(x_0, r_0)$, die in der offenen Kugel $B(x_0, \tilde{r}_0)$ eindeutig ist.

Satz 1.2.7. *Gelten die Voraussetzungen von Satz 1.2.5 und ist zusätzlich* $h = \frac{1}{2}$, *so hat* (1.1) *eine Lösung, die in der abgeschlossenen Kugel* $\bar{B}(x_0, r_0) = \bar{B}(x_0, 2\eta_0)$ *eindeutig ist.*

Abschließend geben wir im folgenden Satz eine Abschätzung für die Konvergenzgeschwindigkeit des Newton-Verfahrens an.

Satz 1.2.8. *Sind die Voraussetzungen von Satz 1.2.5 erfüllt, so gilt die Fehlerabschätzung*

$$||x^* - x_k||_X \leq \frac{1}{2^{k-1}}(2h_0)^{2^k-1}\eta_0 \qquad \textit{für alle } k \in \mathbb{N}.$$

1.3 Quadraturverfahren

In diesem Abschnitt stellen wir das Newton-Cotes-Quadraturverfahren und das Gaußsche Quadraturverfahren vor, die für die numerische Integration stetiger Funktionen genutzt werden. Im Verfahren von Nyström dienen Quadraturverfahren zur Diskretisierung einer linearen Fredholmschen Integralgleichung zweiter Art. Ausführlicher werden diese beiden Verfahren und weitere Integrationsverfahren im Buch von Stoer [45] besprochen.

Eine Abbildung Q, die der stetigen Funktion f einen Näherungswert für $\int_a^b f(t)\,dt$ zuweist, d. h.

$$Q(f) \approx \int_a^b f(t)\,dt,$$

bezeichnen wir als *Quadraturformel*.

Ist für jedes $n \in \mathbb{N}$ eine solche Quadraturformel Q_n gegeben, so sprechen wir von einem *Quadraturverfahren*.

Eine natürliche Forderung für ein Quadraturverfahren ist, dass (Q_n) punktweise gegen den Wert des Integrals konvergieren soll:

$$\lim_{n\to\infty} Q_n(f) = \int_a^b f(t)\,dt \qquad \text{für alle } f \in C([a,b]).$$

Der folgende Satz gibt Bedingungen an, unter denen diese Forderung erfüllt ist.

Satz 1.3.1. *Ist das Quadraturverfahren* (Q_n) *für alle Polynome vom Grad kleiner gleich* γ_n *mit* $\gamma_n \to \infty$ *für* $n \to \infty$ *exakt, d. h. gilt*

$$E_n(p) := Q_n(p) - \int_a^b p(t)\,dt = 0 \qquad \textit{für alle Polynome } p \textit{ mit } \operatorname{grad} p \leq \gamma_n,$$

und ist $||Q_n|| = \sup\limits_{f\in C([a,b])\setminus\{0\}} \dfrac{|Q_n(f)|}{||f||}$ *gleichmäßig beschränkt auf* $C([a,b])$, *d. h. gilt*

$$\sup_{n\in\mathbb{N}} ||Q_n|| = \sup_{n\in\mathbb{N}} \sup_{f\in C([a,b])\setminus\{0\}} \frac{|Q_n(f)|}{||f||} < \infty,$$

dann konvergiert das Quadraturverfahren punktweise gegen den exakten Wert des Integrals

$$\lim_{n\to\infty} Q_n(f) = \int_a^b f(t)\,dt \qquad \text{für alle } f \in C([a,b]).$$

Bemerkung 1.3.2. *Alle Quadraturverfahren, die wir durch Summieren der Newton-Cotes-Formeln bis zur Ordnung 6 erhalten (siehe Abschnitt 1.3.1) und das Gaußsche Quadraturverfahren (siehe Abschnitt 1.3.2), erfüllen die Voraussetzungen des vorangehenden Satzes. Eine punktweise Konvergenz des Verfahrens gegen den exakten Wert des Integrals ist somit gesichert.*

1.3.1 Newton-Cotes-Quadraturverfahren

Ersetzen wir den Integranden $f(t)$ durch ein interpolierendes Polynom $p(t)$ und integrieren dieses, so erhalten wir gemäß

$$Q(f) := \int_a^b p(t)\,dt \approx \int_a^b f(t)\,dt$$

eine Quadraturformel von Newton-Cotes (siehe Stoer [45]).

Im Folgenden wollen wir eine explizite Darstellung für $Q(f)$ finden. Dazu wählen wir zuerst $n+1$ äquidistant im Intervall $[a,b]$ verteilte Punkte, so genannte Knoten, also $t_{n,i} = a + ih$, $i = 0,\dots,n$ mit Schrittweite $h = (b-a)/n$. Anschließend wählen wir das eindeutig bestimmte Interpolationspolynom p_n mit $\operatorname{grad} p_n \le n$, welches an den Knoten mit der Funktion f übereinstimmt:

$$p_n(t_{n,i}) = f(t_{n,i}), \qquad i = 0,\dots,n.$$

Wegen der Interpolationsformel von Lagrange

$$p_n(t) = \sum_{i=0}^n f(t_{n,i}) L_{n,i}(t) \qquad \text{mit} \qquad L_{n,i}(t) := \prod_{\substack{j=0\\ j\neq i}}^n \frac{t - t_{n,j}}{t_{n,i} - t_{n,j}}$$

erhalten wir

$$\begin{aligned} Q_n(f) &:= \int_a^b p_n(t)\,dt = \int_a^b \sum_{i=0}^n f(t_{n,i}) L_{n,i}(t)\,dt = \sum_{i=0}^n f(t_{n,i}) \int_a^b L_{n,i}(t)\,dt \\ &\overset{t=a+hs}{=} \sum_{i=0}^n f(t_{n,i}) h \int_0^n \prod_{\substack{j=0\\ j\neq i}}^n \frac{s-j}{i-j}\,ds = h \sum_{i=0}^n f(t_{n,i}) \alpha_{n,i}. \end{aligned}$$

Dabei sind $\alpha_{n,i}$, $i = 0,\dots,n$ Konstanten, die nur von n, nicht aber von der Funktion f oder dem Integrationsintervall $[a,b]$ abhängen.

Berechnen wir diese Konstanten $\alpha_{n,i}$ für $n = 1$, so erhalten wir

$$Q_1(f) := \int_a^b p_1(t)\,dt = \frac{h}{2}\Big[f(t_{1,0}) + f(t_{1,1})\Big],$$

die so genannte *Trapezregel*. Für $f \in C^2([a,b])$ ergibt sich deren Quadraturfehler $E_1(f)$ als

$$E_1(f) = Q_1(f) - \int_a^b f(t)\,dt = \frac{h^3}{12} f^{(2)}(\xi) \qquad \text{für ein } \xi \in (a,b).$$

Für $n = 2$ ergibt sich

$$Q_2(f) := \int_a^b p_2(t)\,dt = \frac{h}{3}\Big[f(t_{2,0}) + 4f(t_{2,1}) + f(t_{2,2})\Big],$$

die so genannte *Simpson-Regel*. Deren Quadraturfehler $E_2(f)$ ist für eine viermal differenzierbare Funktion f gegeben durch

$$E_2(f) = Q_2(f) - \int_a^b f(t)\,dt = \frac{h^5}{90} f^{(4)}(\xi) \qquad \text{für ein } \xi \in (a,b).$$

Üblicherweise wenden wir eine Quadraturformel nicht auf das gesamte Intervall $[a,b]$ an, sondern unterteilen dieses in mehrere Teilintervalle. Die Ergebnisse der Quadraturformel in den einzelnen Intervallen summieren wir dann zum Näherungswert für das Integral über $[a,b]$ auf, weshalb wir auch von *summierten Quadraturformeln* sprechen.

Unterteilen wir beispielsweise das Intervall $[a,b]$ in m gleich große Intervalle $[t]_{m,i} = [t_{m,i}, t_{m,i+1}]$, $i = 0,\dots,m-1$ und verwenden in jedem dieser Intervalle die Trapezregel um den Wert des Integrals zu approximieren, so gelangen wir zur *Trapezsumme*

$$\begin{aligned} Q_1^m(f) &:= \sum_{i=0}^{m-1} \frac{h}{2}\Big[f(t_{m,i}) + f(t_{m,i+1})\Big] \\ &= h\Big[\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \cdots + f(b-h) + \frac{1}{2}f(b)\Big]. \end{aligned}$$

Durch Aufsummieren der Quadraturfehler in den einzelnen Teilintervallen ergibt sich der Quadraturfehler der Trapezsumme zu

$$E_1^m(f) := Q_1^m(f) - \int_a^b f(t)\,dt = \frac{h^2}{12}(b-a) f^{(2)}(\xi) \qquad \text{für ein } \xi \in (a,b).$$

Unterteilen wir das Intervall $[a,b]$ dagegen für gerades m in $m/2$ gleich große Intervalle $[t]_{m/2,i} = [t_{m,2i}, t_{m,2(i+1)}]$, $i = 0,\dots,m/2-1$ und verwenden in jedem dieser Intervalle die Simpson-Regel, so erhalten wir die *Simpson-Summe*

$$\begin{aligned} Q_2^m &:= \sum_{i=0}^{m/2-1} \frac{h}{3}\Big[f(t_{m,2i}) + 4f(t_{m,2i+1}) + f(t_{m,2(i+1)})\Big] \\ &= \frac{h}{3}\Big[f(a) + 4f(a+h) + 2f(a+2h) + 4f(a+3h) + \cdots \\ &\qquad \cdots + 4f(b-3h) + 2f(b-2h) + 4f(b-h) + f(b)\Big] \end{aligned}$$

mit dem Quadraturfehler

$$E_2^m(f) := Q_2^m(f) - \int_a^b f(t)\,dt = \frac{h^4}{180}(b-a) f^{(4)}(\xi) \qquad \text{für ein } \xi \in (a,b).$$

1.3.2 Gaußsches Quadraturverfahren

In diesem Abschnitt wird durch geschickte Wahl der Knoten $t_{n,i}$, $i = 1, \ldots, n$, und Gewichte $w_{n,i}$, $i = 1, \ldots, n$, ein Quadraturverfahren

$$Q_n(f) := \sum_{i=1}^{n} w_{n,i} f(t_{n,i})$$

konstruiert, für das der Quadraturfehler

$$E_n(p) := \int_a^b p(t)\, dt - Q_n(p)$$

für Polynome möglichst hohen Grades verschwindet (siehe Stoer [45]).

Dazu betrachten wir die Folge der Polynome

$$\begin{aligned} p_0(t) &= 1, \\ p_{i+1}(t) &= (t - \delta_{i+1})p_i(t) - \gamma_{i+1}^2 p_{i-1}(t), \qquad i \geq 1 \end{aligned}$$

mit

$$\begin{aligned} p_{-1}(t) &\equiv 0, \\ \delta_{i+1} &:= \frac{(tp_i, p_i)}{(p_i, p_i)}, \qquad i \geq 0, && (1.5) \\ \gamma_{i+1}^2 &:= \begin{cases} 0, & i = 0, \\ \dfrac{(p_i, p_i)}{(p_{i-1}, p_{i-1})}, & i \geq 1, \end{cases} && (1.6) \end{aligned}$$

die bezüglich des Innenproduktes

$$(f, g) := \int_a^b f(t)g(t)\, dt$$

orthogonal sind. Der folgende Satz zeigt deren Bedeutung.

Satz 1.3.3.

1. *Seien $t_{n,i}$, $i = 1, \ldots, n$, die Nullstellen des Polynoms p_n und $w_{n,i}$, $i = 1, \ldots, n$, die Komponenten der Lösung des linearen Gleichungssystems*

$$\sum_{i=1}^{n} p_k(t_{n,i}) w_{n,i} = \begin{cases} (p_0, p_0), & k = 0, \\ 0, & k \geq 1, \end{cases} \qquad k = 1, \ldots, n. \tag{1.7}$$

Dann gilt $w_{n,i} > 0$, $i = 1, \ldots, n$ und

$$E_n(p) = 0 \qquad \text{für alle Polynome } p \text{ mit } \operatorname{grad} p \leq 2n - 1.$$

2. *Gilt umgekehrt für Zahlen* $t_{n,i}$, $i = 1, \ldots, n$ *und* $w_{n,i}$, $i = 1, \ldots, n$, *dass der Quadraturfehler* E_n *für alle Polynome vom Grad kleiner oder gleich* $2n - 1$ *verschwindet, so sind die Zahlen* $t_{n,i}$ *die Nullstellen des Polynoms* p_n *und die Zahlen* $w_{n,i}$ *erfüllen das Gleichungssystem* (1.7).

3. *Es gibt keine Knoten* $t_{n,i}$, $i = 1, \ldots, n$, *und keine Gewichte* $w_{n,i}$, $i = 1, \ldots, n$, *so dass* E_n *für alle Polynome vom Grad kleiner oder gleich* $2n$ *verschwindet.*

Bemerkung 1.3.4.

1. *Die Nullstellen des Polynoms* p_n, $n \in \mathbb{N}$ *sind reell und einfach und liegen alle im offenen Intervall* (a, b).

2. *Das Gleichungssystem* (1.7) *ist für jedes* $n \in \mathbb{N}$ *nichtsingulär und besitzt somit eine eindeutig bestimmte Lösung* $w_{n,i}$, $i = 1, \ldots, n$.

Der folgende Satz liefert eine Abschätzung für den Quadraturfehler.

Satz 1.3.5. *Für* $f \in C^{2n}([a, b])$ *gilt*

$$E_n(f) = \int_a^b f(t)\,dt - Q_n(f) = \frac{f^{(2n)}(\xi)}{(2n)!}(p_n, p_n) \qquad \text{für ein } \xi \in (a, b)$$

für den Fehler $E_n(f)$ *der Gauß-Quadratur.*

Der folgende Satz zeigt, dass sich die Aufgabe, die Knoten $t_{n,i}$ als Nullstellen des Polynoms p_n zu berechnen und die Gewichte $w_{n,i}$ als Lösung eines vollbesetzten linearen Gleichungssystems zu bestimmen, auf ein Eigenwertproblem für eine Tridiagonalmatrix reduzieren lässt.

Satz 1.3.6. *Sei*

$$\boldsymbol{J} := \begin{pmatrix} \delta_1 & \gamma_2 & & & \\ \gamma_2 & \delta_2 & \gamma_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-1} & \delta_{n-1} & \gamma_n \\ & & & \gamma_n & \delta_n \end{pmatrix}$$

mit δ_i, $i = 1, \ldots, n$ *wie in* (1.5) *und* γ_i, $i = 2, \ldots, n$ *wie in* (1.6). *Die Knoten der Gauß-Quadratur* $t_{n,i}$, $i = 1, \ldots, n$, *sind die Eigenwerte von* $\boldsymbol{J}$. *Sei* $\boldsymbol{v}^{(i)}$ *ein Eigenvektor zum Eigenwert* $t_{n,i}$ *mit der Normierung* $(\boldsymbol{v}^{(i)})^t\boldsymbol{v}^{(i)} = b - a$ *und* $v_1^{(i)}$ *seine erste Komponente, dann ergeben sich die Gewichte* $w_{n,i}$, $i = 1, \ldots, n$ *als*

$$w_{n,i} = (v_1^{(i)})^2.$$

Beweis. Siehe Golub und Welsch [13]. □

1.4 Intervallrechnung

Gleitkommaberechnungen auf einem Computer liefern konstruktionsbedingt im Allgemeinen keine exakten Ergebnisse. Wollen wir dennoch mit Hilfe eines Computers "exakt" rechnen, so bleibt uns nur die Möglichkeit das Ergebnis einer Berechnung in ein Intervall einzuschließen, dessen Intervallgrenzen Computerzahlen sind. Mit diesen computerrepräsentierbaren Intervallen wollen wir dann genauso rechnen können, wie wir es von reellen Zahlen gewöhnt sind. Insbesondere benötigen wir Verknüpfungen wie etwa Addition und Multiplikation. Um eine solche Maschinenintervallarithmetik einzuführen, gehen wir in zwei Schritten vor. Zuerst führen wir eine reelle Intervallrechnung ein und untersuchen deren Eigenschaften. Danach übertragen wir diese Intervallrechnung auf Maschinenintervalle.

Eine ausführliche Darstellung der Intervallrechnung sowie die hier nicht vorgeführten Beweise finden sich beispielsweise in Alefeld und Herzberger [1].

1.4.1 Reelle Intervallrechnung

Zunächst führen wir eine geeignete Notation ein.

Definition 1.4.1. *Die Menge aller abgeschlossenen reellen Intervalle bezeichnen wir mit* $\mathbb{IR}$. *Abgeschlossene Intervalle, also Elemente aus* $\mathbb{IR}$, *schreiben wir in eckigen Klammern, beispielsweise* $[a] \in \mathbb{IR}$. *Die Ober- bzw. Untergrenze von* $[a] \in \mathbb{IR}$ *bezeichnen wir mit* $\overline{a}$ *bzw.* $\underline{a}$. *Eine reelle Zahl* a *identifizieren wir mit dem Punktintervall* $[a, a]$.

Auf diese Menge wollen wir nun die von den reellen Zahlen bekannten arithmetischen Verknüpfungen, wie Addition oder Multiplikation, übertragen.

Definition 1.4.2 (Intervalloperationen). *Sei* $* \in \{+, -, \cdot, /\}$ *eine der bekannten arithmetischen Grundoperationen für reelle Zahlen. Dann gilt für* $[a],\ [b] \in \mathbb{IR}$:

$$[a] * [b] := \{a * b |\ a \in [a],\ b \in [b]\}.$$

Bei der Division wird zusätzlich $0 \notin [b]$ *vorausgesetzt, was im Folgenden nicht mehr besonders erwähnt wird. Verknüpfungen in* $\mathbb{IR}$ *bezeichnen wir mit den gleichen Symbolen, wie die reellen Operationen.*

Wollen wir diese Verknüpfungen explizit durchführen, ergeben sich mit $[a] = [\underline{a}, \overline{a}]$ und $[b] = [\underline{b}, \overline{b}]$ die folgenden Rechenregeln:

$$\begin{aligned}
[a] + [b] &= [\underline{a} + \underline{b}, \overline{a} + \overline{b}], \\
[a] - [b] &= [\underline{a} - \overline{b}, \overline{a} - \underline{b}] = [a] + [-1, -1] \cdot [b], \\
[a] \cdot [b] &= [\min\{\underline{a} \cdot \underline{b}, \underline{a} \cdot \overline{b}, \overline{a} \cdot \underline{b}, \overline{a} \cdot \overline{b}\}, \max\{\underline{a} \cdot \underline{b}, \underline{a} \cdot \overline{b}, \overline{a} \cdot \underline{b}, \overline{a} \cdot \overline{b}\}], \\
[a]/[b] &= [\underline{a}, \overline{a}] \cdot \left[\frac{1}{\overline{b}}, \frac{1}{\underline{b}}\right].
\end{aligned}$$

Die Menge $\mathbb{IR}$ ist also bezüglich der eingeführten Verknüpfungen abgeschlossen. Im folgenden Satz stellen wir einige elementare Eigenschaften der Intervalloperationen zusammen.

Satz 1.4.3 (Elementare Eigenschaften von Intervalloperationen). *Seien* $[a]$, $[b]$ *und* $[c]$ *Intervalle aus* $\mathbb{IR}$. *Dann gilt*

1. $[a] + [b] = [b] + [a], \qquad [a] \cdot [b] = [b] \cdot [a]$ *(Kommutativität).*
2. $([a] + [b]) + [c] = [a] + ([b] + [c]), \qquad ([a] \cdot [b]) \cdot [c] = [a] \cdot ([b] \cdot [c])$ *(Assoziativität).*
3. *Das Punktintervall* $[0, 0]$ *ist das eindeutig bestimmte neutrale Element der Addition, d. h. für alle* $[a] \in \mathbb{IR}$ *gilt*
$$[a] = [x] + [a] = [a] + [x] \qquad \Longleftrightarrow \qquad [x] = [0, 0].$$
Genauso gilt: Das Punktintervall $[1, 1]$ *ist das eindeutig bestimmte neutrale Element der Multiplikation, d. h. für alle* $[a] \in \mathbb{IR}$ *gilt*
$$[a] = [x] \cdot [a] = [a] \cdot [x] \qquad \Longleftrightarrow \qquad [x] = [1, 1].$$
4. *Ein beliebiges Element* $[a] = [\underline{a}, \overline{a}] \in \mathbb{IR}$ *mit* $\underline{a} \neq \overline{a}$ *besitzt keine inversen Elemente bezüglich der Addition und Multiplikation. Es gilt jedoch*
$$0 \in [a] - [a] \qquad \textit{und} \qquad 1 \in [a]/[a].$$
5. $[a] \cdot ([b] + [c]) \subseteq [a] \cdot [b] + [a] \cdot [c]$ *(Subdistributivität),*
$a \cdot ([b] + [c]) = a \cdot [b] + a \cdot [c]$ *für alle* $a \in \mathbb{R}$.

Im nächsten Satz stellen wir die Teilmengeneigenschaften von Intervalloperationen vor, die es uns ermöglichen die Ergebnisse von reellen Rechenoperationen einzuschließen. Für die Übertragung der Intervallrechnung auf die Maschinenintervallrechnung ist die erste Eigenschaft im nachfolgenden Satz von Bedeutung.

Satz 1.4.4 (Teilmengeneigenschaft). *Seien* $[a]_1$, $[a]_2$, $[b]_1$ *und* $[b]_2$ *Intervalle in* $\mathbb{IR}$ *mit* $[a]_1 \subseteq [b]_1$ *und* $[a]_2 \subseteq [b]_2$. *Dann gilt für die Verknüpfungen* $* \in \{+, -, \cdot, /\}$:
$$[a]_1 * [a]_2 \subseteq [b]_1 * [b]_2.$$
Insbesondere gilt also für reelle Zahlen $a_1 \in [b]_1$ *und* $a_2 \in [b]_2$
$$a_1 * a_2 \in [b]_1 * [b]_2.$$

Neben den Verknüpfungen $+$, $-$, $\cdot$ und $/$ wollen wir natürlich auch einstellige Operationen wie etwa $\sin(.)$, $\cos(.)$ oder $\mathrm{abs}(.)$ in Intervallarithmetik auswerten. Die nachfolgende Definition erklärt, was wir darunter verstehen wollen.

Definition 1.4.5 (Einstellige Operationen in $\mathbb{IR}$). *Bezeichnet* $r(x)$ *eine stetige einstellige Operation in* $\mathbb{R}$, *dann ist durch*
$$r([x]) := [\min_{x \in [x]} r(x), \max_{x \in [x]} r(x)]$$
eine (zugehörige) einstellige Operation in $\mathbb{IR}$ *erklärt.*

1.4.2 Abstand, Betrag und Durchmesser von reellen Intervallen und deren Eigenschaften

In diesem Abschnitt definieren wir den Begriff des Abstandes, der notwendig ist um die Begriffe der Konvergenz und Stetigkeit auf die Menge $\mathbb{IR}$ zu übertragen.

Definition 1.4.6 (Abstand). *Für zwei Intervalle* $[a] = [\underline{a}, \overline{a}]$ *und* $[b] = [\underline{b}, \overline{b}]$ *ist der* Abstand *definiert durch*

$$q([a],[b]) := \max\{|\underline{a} - \underline{b}|, |\overline{a} - \overline{b}|\}.$$

Durch den Abstand q wird auf $\mathbb{IR}$ eine Metrik eingeführt, da q die dafür notwendigen Eigenschaften besitzt:

1. $q([a],[b]) \geq 0$ für alle $[a],\ [b] \in \mathbb{IR}$,
   $q([a],[b]) = 0 \quad \Longleftrightarrow \quad [a] = [b]$, (Definitheit)
2. $q([a],[b]) = q([b],[a])$ für alle $[a],\ [b] \in \mathbb{IR}$, (Symmetrie)
3. $q([a],[b]) \leq q([a],[c]) + q([c],[b])$ für alle $[a],\ [b],\ [c] \in \mathbb{IR}$. (Dreiecksungleichung)

Bemerkung 1.4.7. *Für Punktintervalle geht der obige Abstandsbegriff in den für reelle Zahlen üblichen Abstandsbegriff* $q(a,b) = |a - b|$ *über.*

Da wir nun eine Metrik q auf $\mathbb{IR}$ erklärt haben, können wir die Begriffe Konvergenz und Stetigkeit genauso wie in anderen metrischen Räumen einführen.

Beispielsweise gilt für die Konvergenz

$$[a]_k \stackrel{k\to\infty}{\to} [a] \qquad \Longleftrightarrow \qquad q([a]_k,[a]) \stackrel{k\to\infty}{\to} 0,$$

woraus unmittelbar

$$[a]_k \stackrel{k\to\infty}{\to} [a] \qquad \Longleftrightarrow \qquad \underline{a}_k \stackrel{k\to\infty}{\to} \underline{a} \quad \text{und} \quad \overline{a}_k \stackrel{k\to\infty}{\to} \overline{a}$$

folgt.

Der nachfolgende Satz zeigt, dass $\mathbb{IR}$ mit dem oben gegebenen Abstandsbegriff ein vollständiger metrischer Raum ist.

Satz 1.4.8 (Vollständigkeit von $(\mathbb{IR}, q)$). *Der metrische Raum* $(\mathbb{IR}, q)$ *ist vollständig. (d. h. , dass jede Cauchyfolge von Intervallen ein Intervall als Grenzwert besitzt).*

In den beiden folgenden Sätzen untersuchen wir nun die Intervalloperationen und die einstelligen Operationen auf Stetigkeit.

Satz 1.4.9 (Stetigkeit der Intervalloperationen). *Die Intervalloperationen* $+, -, \cdot$ *und* $/$ *sind stetig.*

Satz 1.4.10 (Stetigkeit von einstelligen Operationen in $\mathbb{IR}$). *Ist $r(x)$ eine stetige einstellige Operation, so ist $r([x]) := [\min_{x\in[x]} r(x), \max_{x\in[x]} r(x)]$ eine stetige einstellige Operation in $\mathbb{IR}$.*

Nachfolgend wird der Betrag eines Intervalls definiert, der ein Maß für den Abstand des Intervalls zur Null darstellt und der für Punktintervalle in den bekannten Betragsbegriff übergeht.

Definition 1.4.11 (Betrag). *Für ein Intervall $[a] = [\underline{a}, \overline{a}] \in \mathbb{IR}$ heißt*

$$|[a]| := q([a],[0,0]) = \max\{|\underline{a}|, |\overline{a}|\} = \max_{a\in[a]} |a|$$

der Betrag *von $[a]$.*

Der Durchmesser eines Intervalls ist ein Maß für die Güte einer Einschließung.

Definition 1.4.12 (Durchmesser). *Für ein Intervall $[a] = [\underline{a}, \overline{a}] \in \mathbb{IR}$ heißt*

$$\operatorname{diam}([a]) := \overline{a} - \underline{a} \geq 0$$

der Durchmesser *von $[a]$.*

1.4.3 Intervallmäßige Auswertung und Wertebereiche reeller Funktionen

In diesem Abschnitt beschäftigen wir uns mit der Auswertung von Funktionsausdrücken und mit der Einschließung des Wertebereichs einer reellen Funktion. Als Generalvoraussetzung für den gesamten Abschnitt fordern wir, dass alle betrachteten Funktionen f reell und stetig sind.

Definition 1.4.13 (Funktionsausdruck). *Ein zu f gehörender* Funktionsausdruck *sei eine Rechenvorschrift, mit der zu jedem x nach endlich vielen ein- oder zweistelligen Operationen aus einer fest vorgegebenen Klasse, zu denen auch die entsprechenden Intervalloperationen definiert seien, der jeweilige Funktionswert bestimmt werden kann.*

Enthält der zu f gehörende Funktionsausdruck auch noch die Konstanten $a_1, \ldots, a_m$ so wird zur Verdeutlichung dafür $f(x; a_1, \ldots, a_m)$ geschrieben. O.B.d.A. setzen wir voraus, dass jede dieser Konstanten im Funktionsausdruck nur einmal vorkommt (käme eine Konstante mehrfach vor, so würden wir für sie mehrere Namen a_i vergeben und diese dann gleichsetzen).

Nachfolgend definieren wir für reelle Funktionen den Begriff des Wertebereichs.

Definition 1.4.14 (Wertebereich). *Die Menge*

$$W(f,[x];[a]_1,\ldots,[a]_m) := \{f(x;a_1,\ldots,a_m)|\ x\in[x],\ a_1\in[a]_1,\ \ldots,\ a_m\in[a]_m\}$$

heißt Wertebereich *der Funktion f.*

Wie wir anhand der Definition sehen, ist der Wertebereich der Funktion f unabhängig vom jeweiligen Funktionsausdruck für f.

Haben wir zu einer reellen Funktion einen Funktionsausdruck gegeben, so ist klar, wie wir diesen auszuwerten haben. Soll die Funktion hingegen intervallmäßig ausgewertet werden, so muss zuerst definiert werden, wie dies zu geschehen hat. Mit der folgende Definition erklären wir, was wir unter einer intervallmäßigen Auswertung verstehen wollen.

Definition 1.4.15 (Intervallmäßige Auswertung). *Ist eine Funktion f mit zugehörigem Funktionsausdruck $f(x; a_1, \ldots, a_m)$ gegeben, so bezeichnen wir $f([x]; [a]_1, \ldots, [a]_m)$ als* intervallmäßige oder intervallarithmetische Auswertung *von f, falls bei der Ersetzung aller Operanden durch Intervalle, die im Definitionsbereich enthalten sind, und aller Operationen durch Intervalloperationen stets ein definierter Ausdruck entsteht.*

Bemerkung 1.4.16.

- *Die intervallmäßige Auswertung der Funktion $g(x; a) = \frac{ax}{1-x}$ ist, wie wir am Beispiel*

$$g_1(x; a) = \frac{ax}{1-x} \qquad \Longrightarrow \qquad g_1([2,3]; [0,1]) = [-3, 0],$$
$$g_2(x; a) = \frac{a}{\frac{1}{x} - 1} \qquad \Longrightarrow \qquad g_2([2,3]; [0,1]) = [-2, 0]$$

  sehen können, von der Wahl des Funktionsausdruckes g_1 bzw. g_2 abhängig.

- *Nicht jeder reelle Funktionsausdruck führt zu einem definierten Intervallausdruck. Ist etwa die reelle Funktion definiert durch den Funktionsausdruck $f(x) = \frac{1}{x \cdot x + 0.5}$ und werten wir f intervallmäßig an der im natürlichen Definitionsbereich von f enthaltenen Stelle $[x] = [-1, 1]$ aus, so erhalten wir den undefinierten Ausdruck*

$$\begin{aligned} f([-1,1]) &= [1,1]/([-1,1] \cdot [-1,1] + [0.5, 0.5]) \\ &= [1,1]/([-1,1] + [0.5, 0.5]) \\ &= [1,1]/[-0.5, 1.5]. \end{aligned}$$

Aus der Definition der intervallmäßigen Auswertung von Funktionen ergibt sich sofort die Einschließungseigenschaft, die wir für die Einschließung des Wertebereichs einer reellen Funktion nützen können, sowie die Teilmengeneigenschaft, die von Bedeutung ist, wenn wir die Intervallrechnung auf den Computer übertragen.

Satz 1.4.17. *Sei f eine stetige Funktion der reellen Veränderlichen $x_1, \ldots, x_n$ mit zugehörigem Funktionsausdruck $f(x_1, \ldots, x_n; a_1, \ldots, a_m)$. Ferner sei die intervallmäßige Auswertung $f(\tilde{[x]}_1, \ldots, \tilde{[x]}_n; \tilde{[a]}_1, \ldots, \tilde{[a]}_m)$ für die Intervalle $\tilde{[x]}_1, \ldots, \tilde{[x]}_n$ und $\tilde{[a]}_1, \ldots, \tilde{[a]}_m$ definiert. Dann gilt:*

1. *Für alle $[x]_k \subseteq \tilde{[x]}_k$, $k = 1, \ldots, n$, und alle $[a]_k \subseteq \tilde{[a]}_k$, $k = 1, \ldots, m$, ist*

$$W(f, [x]_1, \ldots, [x]_n; [a]_1, \ldots, [a]_m) \subseteq f([x]_1, \ldots [x]_n; [a]_1 \ldots, [a]_m)$$

   (Einschließungseigenschaft).

2. *Für alle* $[x]_k \subseteq [\hat{x}]_k \subseteq [\tilde{x}]_k$, $k = 1, \ldots, n$, *und alle* $[a]_k \subseteq [\hat{a}]_k \subseteq [\tilde{a}]_k$, $k = 1, \ldots, m$, *ist*

$$f([x]_1, \ldots, [x]_n; [a]_1, \ldots, [a]_m) \subseteq f([\hat{x}]_1, \ldots, [\hat{x}]_n; [\hat{a}]_1, \ldots, [\hat{a}]_m)$$

   (Teilmengeneigenschaft).

In den beiden folgenden Sätze beschäftigen wir uns mit der Güte, mit der der Wertebereich einer Funktion f durch die intervallmäßige Auswertung der Funktion approximiert werden kann.

Satz 1.4.18. *Sei f eine reelle Funktion einer Veränderlichen x mit zugehörigem Funktionsausdruck $f(x; a_1, \ldots, a_m)$. Die intervallmäßige Auswertung des Ausdruckes für f existiere für $[\tilde{x}]; [a]_1, \ldots, [a]_m \in \mathbb{IR}$. Der Funktionsausdruck, den man erhält, wenn man im Funktionsausdruck für f bei jedem Auftreten von x eine neue Variable x_i einführt, genüge in jeder dieser Variablen im Intervall $[\tilde{x}]$ einer Lipschitzbedingung. Dann gibt es ein $\gamma \geq 0$, so dass für $[x] \subseteq [\tilde{x}]$*

$$q(W(f, [x]; [a]_1 \ldots, [a]_m), f([x]; [a]_1 \ldots, [a]_m)) \leq \gamma \operatorname{diam}([x])$$

gilt. Für $\operatorname{diam}([x]) \to 0$ *gilt also* $f([x]; [a]_1, \ldots, [a]_m) \to W(f, [x]; [a]_1, \ldots [a]_m)$.

Bemerkung 1.4.19.

- *Die geforderte Lipschitzbedingung in jeder der neuen Variablen lässt sich bei Funktionen, die aus intervallmäßig auswertbaren Operationen zusammengesetzt sind, oft erfüllen.*
- *Die Approximation des Wertebereichs ist abhängig von der Wahl des Funktionsausdruckes.*

Satz 1.4.20. *Sei f eine reelle Funktion mit zugehörigem Funktionsausdruck $f(x)$. Seien weiterhin die Voraussetzungen des vorherigen Satzes erfüllt, dann gilt für $[x] \subseteq [\tilde{x}]$*

$$\operatorname{diam}(f([x])) \leq c \operatorname{diam}([x]), \qquad \textit{mit einem } c \geq 0.$$

1.4.4 Maschinenintervallarithmetik

In den vorangegangenen Abschnitten haben wir beschrieben, wie mit Intervallen gerechnet werden kann. Hierbei haben wir immer die reellen Zahlen als zugrundeliegenden Körper für die Intervalle angenommen. Wollen wir die Intervallrechnung auf Computern verwirklichen, so haben wir dort nicht alle reellen Zahlen, sondern nur eine endliche Anzahl von *Maschinenzahlen* zur Verfügung (siehe Alefeld und Herzberger [1], Kulisch und Miranker [25]).

Definition 1.4.21 (Maschinenzahlen). *Reelle Zahlen, die sich in der Form*

$$x = m \cdot b^e$$

darstellen lassen, wobei die Basis b fest vorgegeben ist und für die Mantisse m und den Exponenten e nur eine feste Anzahl von endlich vielen Stellen zur Verfügung stehen, werden als Maschinenzahlen *bezeichnet.*

Die Menge aller Maschinenzahlen eines Rechners bezeichnen wir mit $\mathbb{R}_M$, wobei wir zusätzlich noch $\mathbb{R}_M = -\mathbb{R}_M$ fordern.

Bemerkung 1.4.22. *Auf dem Computer wird häufig die Basis $b = 2$ verwendet und die Mantisse durch $\frac{1}{2} \leq |m| < 1$ normalisiert. Der ganzzahlige Exponent liegt dann zwischen $\underline{e}$ und $\overline{e}$.*

Da uns auf einem Rechner nur endlich viele Maschinenzahlen zur Verfügung stehen, ergibt sich das Problem, wie wir eine reelle Zahl $z \in [\min\limits_{x\in\mathbb{R}_\mathrm{M}} x, \max\limits_{x\in\mathbb{R}_\mathrm{M}} x]$ durch eine Maschinenzahl $\tilde{z} \in \mathbb{R}_\mathrm{M}$ repräsentieren wollen. Die in der nächsten Definition beschriebene *Rundung* zeigt, wie wir dazu vorgehen wollen.

Definition 1.4.23 (Rundung). *Eine Abbildung $\mathrm{rnd_M} : \mathbb{R} \to \mathbb{R}_M$ wird als* Rundung *bezeichnet, wenn aus $x \leq y$ auch $\mathrm{rnd_M}\, x \leq \mathrm{rnd_M}\, y$ folgt (Monotonie). Eine Abbildung $\mathrm{rnd_M} : \mathbb{R} \to \mathbb{R}_M$, für die außerdem $x = \mathrm{rnd_M}\, x$ für alle $x \in \mathbb{R}_M$ gilt, heißt* optimale Rundung*.*

Um später reelle Zahlen in Intervalle mit Maschinenzahlen als Intervallgrenzen einzuschließen, benötigen wir eine gerichtete Rundung.

Definition 1.4.24 (Gerichtete Rundung). *Eine Rundung $\underline{\mathrm{rnd}}_\mathrm{M} : \mathbb{R} \to \mathbb{R}_M$, für die*

$$\underline{\mathrm{rnd}}_\mathrm{M}\, x \leq x \text{ für alle } x \in \mathbb{R}$$

gilt, heißt nach unten gerichtete Rundung*.*

Bemerkung 1.4.25. *Aus jeder nach unten gerichteten Rundung kann man durch die einfache Definition $\overline{\mathrm{rnd}}_\mathrm{M}\, x := -\,\underline{\mathrm{rnd}}_\mathrm{M}(-x)$ eine nach oben gerichtete Rundung erhalten.*

Wir haben bisher gesehen, wie wir mit Hilfe einer Rundung für reelle Zahlen $x \in \mathbb{R}$ einen Repräsentanten auf dem Computer $\mathrm{rnd_M}\, x \in \mathbb{R}_\mathrm{M}$ finden können. Die zuvor eingeführten gerichteten Rundungen wollen wir nun nutzen, um auch für reelle Intervalle $[x] \in \mathbb{IR}$ eine Repräsentation auf dem Rechner zu finden.

Zuerst definieren wir aber:

Definition 1.4.26 (Maschinenintervall). *Ein Maschinenintervall $[x]$ ist gegeben durch $[x] = [\underline{x}, \overline{x}]$ mit $\underline{x},\ \overline{x} \in \mathbb{R}_M$ und $\underline{x} \leq \overline{x}$. Die Menge aller Maschinenintervalle $\mathbb{IR}_M$ ist somit gegeben durch*

$$\mathbb{IR}_M := \{[\underline{x}, \overline{x}]|\ \underline{x},\ \overline{x} \in \mathbb{R}_M \text{ und } \underline{x} \leq \overline{x}\} \subseteq \mathbb{IR}.$$

Auch für die Menge der Maschinenintervalle müssen wir wieder eine geeignete Rundung definieren.

Definition 1.4.27 (Rundung). *Die Abbildung* $\mathrm{rnd_M} : \mathbb{IR} \to \mathbb{IR}_M$ *heißt* Rundung*, falls für* $[x], [y] \in \mathbb{IR}$

$$[x] \subseteq \mathrm{rnd_M}[x]$$

sowie

$$[x] \subseteq [y] \qquad \Longrightarrow \qquad \mathrm{rnd_M}[x] \subseteq \mathrm{rnd_M}[y]$$

gelten.

Bemerkung 1.4.28.

- *Die zweite Bedingung ist ähnlich der Forderung der Monotonie bei der Rundung reeller Zahlen.*
- *Die erste Bedingung wird notwendig sein, um für Operationen zwischen Maschinenintervallen ähnliche Resultate zu erhalten, wie wir sie von Operationen zwischen Intervallen kennen.*
- *Jede Intervallrundung lässt sich durch eine nach unten und eine nach oben gerichtete Rundung*

  $$\mathrm{rnd_M}[x] = \mathrm{rnd_M}[\underline{x}, \overline{x}] = [\underline{\mathrm{rnd}}_\mathrm{M}\, \underline{x}, \overline{\mathrm{rnd}}_\mathrm{M}\, \overline{x}]$$

  darstellen.

Über Verknüpfungen von Maschinenzahlen haben wir zwar zuvor nicht gesprochen, aber es ist klar, dass auch das Resultat einer Verknüpfung, abgesehen von Über- oder Unterläufen, selbst wieder eine Maschinenzahl sein muss. Dies erreichen wir, indem wir zuerst die Verknüpfung exakt durchführen und danach runden. Auf ähnliche Weise können wir diese Verknüpfungen auch für Maschinenintervalle definieren.

Definition 1.4.29 (Maschinenintervalloperationen). *Für* $[a], [b] \in \mathbb{IR}_M$ *wird die Operation* $*_M \in \{+, -, \cdot, /\}$ *durch*

$$[a] *_M [b] := \mathrm{rnd_M}([a] * [b]) \in \mathbb{IR}_M,$$

erklärt, wobei $\mathrm{rnd_M}$ *eine vorgegebene Intervallrundung ist.*

Wenn aus dem Kontext klar ist, dass es sich um eine Maschinenintervalloperation handelt, schreiben wir wieder $*$ anstelle von $*_M$. Definieren wir die Verknüpfung zwischen Maschinenintervallen auf diese Weise, so zeigen die folgenden Sätze, dass sich die grundlegenden Eigenschaften der Intervallrechnung auf die Maschinenintervallrechnung vererben.

Satz 1.4.30 (Teilmengeneigenschaft). *Sei* $* \in \{+, -, \cdot, /\}$ *und* $[a]$, $[b]$, $[\tilde{a}]$, $[\tilde{b}] \in \mathbb{IR}_M$ *mit* $[a] \subseteq [\tilde{a}]$ *und* $[b] \subseteq [\tilde{b}]$. *Dann gilt*

$$[a] * [b] \subseteq [\tilde{a}] * [\tilde{b}].$$

Mit anderen Worten: Die von Intervallen bekannte Teilmengeneigenschaft gilt auch für Maschinenintervalle.

Für die Einschließung von Rundungsfehlern oder die Einschließung des Wertebereichs einer Funktion liefert der folgende Satz die entscheidenden Aussagen.

Satz 1.4.31. *Für die Maschinenintervallrundung* $\mathrm{rnd_M}$, *die durch gerichtete Rundungen* ($\underline{\mathrm{rnd}}_\mathrm{M}$, $\overline{\mathrm{rnd}}_\mathrm{M}$) *der Grenzen gewonnen wird, und für* $* \in \{+, -, \cdot, /\}$ *gilt für* $[a]$, $[b] \in \mathbb{IR}_M$

$$[a] * [b] \subseteq [a] *_M [b] \in \mathbb{IR}_M$$

sowie

$$a \in [a],\ b \in [b] \qquad \Longrightarrow \qquad a * b \in [a] *_M [b] \in \mathbb{IR}_M.$$

Insbesondere folgt damit auch für $a, b \in \mathbb{R}$:

$$a * b \in \mathrm{rnd_M}[a, a] *_M \mathrm{rnd_M}[b, b] \in \mathbb{IR}_M.$$

1.5 Steigungsarithmetik

In Abschnitt 1.4.3 haben wir die Einschließung des Wertebereichs einer Funktion f durch die intervallmäßige Auswertung des Funktionsausdrucks für f untersucht. In diesem Abschnitt wollen wir nun Einschließungen des Wertebereichs einer Funktion f mittels der zentrierten Form mit Steigung gewinnen, da diese meist enger sind (siehe Ratz [40]).

Anschließend beschreiben wir, wie die Berechnung einer Einschließung der Steigung auf einem Rechner automatisch erfolgen kann (siehe Ratz [40], Bräuer et al. [5]). Das hierzu verwendete Vorgehen ist ähnlich dem automatischen Differenzieren (siehe Griewank [15], Rall [39]).

Wir definieren zuerst, was wir unter der Steigung einer Funktion verstehen wollen.

Definition 1.5.1 (Steigung). *Sei* $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$. *Eine Funktion* $s_f : D \times D \to \mathbb{R}^n$ *mit*

$$f(\boldsymbol{x}) = f(\boldsymbol{c}) + (s_f(\boldsymbol{x}, \boldsymbol{c}))^T (\boldsymbol{x} - \boldsymbol{c})$$

heißt Steigung *von* f *bezüglich* $(\boldsymbol{x}, \boldsymbol{c}) \in D \times D$.

Bemerkung 1.5.2.

- *Im eindimensionalen Fall, d. h. für $D \subseteq \mathbb{R}$, hat die Steigung die Form*

$$s_f(x,c) = \begin{cases} \dfrac{f(x)-f(c)}{x-c}, & x \neq c, \\ \tilde{s}, & x = c \end{cases}$$

  mit beliebigem $\tilde{s} \in \mathbb{R}$. Ist f stetig differenzierbar und wählen wir $\tilde{s} := f'(c)$, so ist die Steigung s_f stetig bezüglich x und c.

- *Nach der Definition der Steigung der Funktion $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ gilt für jedes $\boldsymbol{x} \in [\boldsymbol{x}] \subseteq D$ und jedes $\boldsymbol{c} \in [\boldsymbol{c}] \subseteq D$*

$$f(\boldsymbol{x}) = f(\boldsymbol{c}) + (s_f(\boldsymbol{x},\boldsymbol{c}))^T(\boldsymbol{x}-\boldsymbol{c}).$$

  Ist somit $[\boldsymbol{f}]^s \in \mathbb{IR}^n$ eine Einschließung der Menge

$$\{s_f(\boldsymbol{x},\boldsymbol{c}) |\ \boldsymbol{x} \in [\boldsymbol{x}],\ \boldsymbol{c} \in [\boldsymbol{c}],\ \boldsymbol{x} \neq \boldsymbol{c}\},$$

  so können wir die Funktion f durch die zentrierte Form mit Steigung

$$\begin{aligned} f(\boldsymbol{x}) &= f(\boldsymbol{c}) + (s_f(\boldsymbol{x},\boldsymbol{c}))^T(\boldsymbol{x}-\boldsymbol{c}) \in f(\boldsymbol{c}) + ([\boldsymbol{f}]^s)^T(\boldsymbol{x}-\boldsymbol{c}) \\ &\subseteq f([\boldsymbol{c}]) + ([\boldsymbol{f}]^s)^T(\boldsymbol{x}-[\boldsymbol{c}]) \subseteq f([\boldsymbol{c}]) + ([\boldsymbol{f}]^s)^T([\boldsymbol{x}]-[\boldsymbol{c}]) \end{aligned}$$

  für alle $\boldsymbol{x} \in [\boldsymbol{x}]$ einschließen. Die unnötig erscheinende Einschließung des Vektors $\boldsymbol{c}$ in den Intervallvektor $[\boldsymbol{c}]$ ist dadurch motiviert, das wir den Vektor $\boldsymbol{c}$ bei konkreter Rechnung auf einem Computer eventuell gar nicht darstellen können. Oft wählen wir $[\boldsymbol{c}]$ als Einschließung von $\mathrm{mid}([\boldsymbol{x}])$.

- *Steigungen von Funktionen existieren nicht nur für differenzierbare Funktionen, sondern beispielsweise auch für die Funktionen* abs, max, min *und die Verzweigungsfunktion*

$$\chi(x,y_1,y_2) := \begin{cases} y_1, & x < 0, \\ y_2, & x \geq 0. \end{cases}$$

Im Folgenden wollen wir nun zeigen, wie die Steigung einer Funktion durch ein Vorgehen ähnlich dem automatischen Differenzieren berechnet werden kann. Dazu definieren wir eine Steigungsarithmetik, die Grundoperationen und einstellige Operationen für so genannte Steigungstripel implementiert. Die Beweise der hier vorgestellten Sätze findet man beispielsweise in Ratz [40].

Definition 1.5.3 (Steigungstripel). *Ein Tripel $\langle f \rangle = ([f]^x, [f]^c, [\boldsymbol{f}]^s)^T$ mit $[f]^x, [f]^c \in \mathbb{IR}$, $[\boldsymbol{f}]^s \in \mathbb{IR}^n$ heißt* Steigungstripel *bezüglich der Funktion $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ und der Intervallvektoren $[\boldsymbol{x}], [\boldsymbol{c}] \subseteq D$, falls*

$$\begin{aligned} \forall \boldsymbol{x} \in [\boldsymbol{x}] : &\quad f(\boldsymbol{x}) \in [f]^x, \\ \forall \boldsymbol{c} \in [\boldsymbol{c}] : &\quad f(\boldsymbol{c}) \in [f]^c, \\ \forall \boldsymbol{x} \in [\boldsymbol{x}],\ \forall \boldsymbol{c} \in [\boldsymbol{c}],\ \exists \boldsymbol{s} \in [\boldsymbol{f}]^s : &\quad f(\boldsymbol{x}) = f(\boldsymbol{c}) + \boldsymbol{s}^T(\boldsymbol{x}-\boldsymbol{c}) \end{aligned}$$

gilt.

Bemerkung 1.5.4.

- *Im Folgenden setzen wir stets* $[\boldsymbol{c}] \subseteq [\boldsymbol{x}]$ *voraus.*
- *Die Tripel*

$$\langle\lambda\rangle = \begin{pmatrix} \lambda \\ \lambda \\ \begin{pmatrix} 0 \\ 0 \end{pmatrix} \end{pmatrix}, \qquad \langle s\rangle = \begin{pmatrix} [s] \\ \mathrm{mid}([s]) \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix} \qquad \textit{und} \qquad \langle t\rangle = \begin{pmatrix} [t] \\ \mathrm{mid}([t]) \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix}$$

sind jeweils ein Steigungstripel bezüglich der konstanten Funktion $f_1(s,t) = \lambda$, *der Funktion* $f_2(s,t) = s$ *und der Funktion* $f_3(s,t) = t$ *und bezüglich der Intervallvektoren* $([s],[t])^T$ *und* $(\mathrm{mid}([s]), \mathrm{mid}([t]))^T$.

Im nächsten Satz beschreiben wir Rechenregeln für Steigungstripel, die der Summen-, Produkt- und Quotientenregel der Differentialrechnung entsprechen.

Satz 1.5.5 (Grundoperationen der Steigungsarithmetik). *Seien* $\langle f\rangle$, $\langle g\rangle$ *Steigungstripel bezüglich der Funktionen* $f, g : D \subseteq \mathbb{R}^n \to \mathbb{R}$ *und der Intervalle* $[\boldsymbol{x}], [\boldsymbol{c}] \subseteq D$, *so sind*

$$\begin{aligned} \langle f\rangle \pm \langle g\rangle &:= \begin{pmatrix} [f]^x \pm [g]^x \\ [f]^c \pm [g]^c \\ [\boldsymbol{f}]^s \pm [\boldsymbol{g}]^s \end{pmatrix}, \\ \langle f\rangle \cdot \langle g\rangle &:= \begin{pmatrix} [f]^x \cdot [g]^x \\ [f]^c \cdot [g]^c \\ [f]^x \cdot [\boldsymbol{g}]^s + [\boldsymbol{f}]^s \cdot [g]^c \end{pmatrix}, \\ \langle f\rangle / \langle g\rangle &:= \begin{pmatrix} [f]^x / [g]^x \\ [f]^c / [g]^c \\ ([\boldsymbol{f}]^s - ([f]^c/[g]^c) \cdot [\boldsymbol{g}]^s)/[g]^x \end{pmatrix} \end{aligned}$$

Steigungstripel bezüglich der Funktionen $f \pm g$, $f \cdot g$ *bzw.* f/g *und der Intervalle* $[\boldsymbol{x}], [\boldsymbol{c}] \subseteq D$.

Im nachfolgenden Satz beschreiben wir eine Kettenregel, mit der Steigungstripel für Kompositionen von Funktionen berechnet werden.

Satz 1.5.6 (Einstellige Operationen der Steigungsarithmetik). *Sei* $\langle f\rangle$ *ein Steigungstripel bezüglich der Funktion* $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ *und der Intervalle* $[\boldsymbol{x}], [\boldsymbol{c}] \subseteq D$. *Es existiere die intervallmäßige Auswertung der Funktion* $g : \mathbb{R} \to \mathbb{R}$ *für die Intervalle* $[f]^x$ *und* $[f]^c$. *Das Intervall* $[g]^s \in \mathbb{IR}$ *sei eine Einschließung der Menge* $\{s_g(x,c) |\ x \in [f]^x,\ c \in [f]^c,\ x \neq c\}$. *Dann ist*

$$g(\langle f\rangle) := \begin{pmatrix} g([f]^x) \\ g([f]^c) \\ [g]^s \cdot [\boldsymbol{f}]^s \end{pmatrix}$$

ein Steigungstripel bezüglich der Funktion $g \circ f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ *und der Intervalle* $[\boldsymbol{x}], [\boldsymbol{c}] \subseteq D$.

Die letzten beiden Sätze zeigen, wie Steigungstripel für zusammengesetzte Funktionen berechnet werden. Es verbleibt die Aufgabe, eine möglichst gute Einschließung der Menge

$$\{s_g(x,c) |\ x \in [f]^x,\ c \in [f]^c,\ x \neq c\}$$

zu bestimmen. Nach dem Mittelwertsatz der Differentialrechnung gilt für eine stetig differenzierbare Funktion g:

$$\{s_g(x,c) |\ x \in [f]^x,\ c \in [f]^c,\ x \neq c\} \subseteq g'([f]^x).$$

Diese Einschließung ist aber meist nur grob. Der nachfolgende Satz zeigt, wie für konvexe bzw. konkave Funktionen bessere Einschließungen gewonnen werden können.

Satz 1.5.7 (Einschließung der Steigung für konvexe bzw. konkave Funktionen). *Sei die Funktion $f : D \subseteq \mathbb{R} \to \mathbb{R}$ stetig differenzierbar, $[x], [c] \in \mathbb{IR}$, $[c] \subseteq [x] \subseteq D$. Die spezielle Steigung (siehe Bemerkung 1.5.2)*

$$s_f(x,c) := \begin{cases} \dfrac{f(x) - f(c)}{x - c}, & x \neq c, \\ f'(x), & x = c \end{cases}$$

können wir durch

$$s_f(\underline{x},\underline{c}) \leq s_f(x,c) \leq s_f(\overline{x},\overline{c}) \qquad \text{für alle } x \in [x],\ c \in [c],\ x \neq c$$

einschließen, falls f auf $[x]$ konvex ist, bzw. durch

$$s_f(\overline{x},\overline{c}) \leq s_f(x,c) \leq s_f(\underline{x},\underline{c}) \qquad \text{für alle } x \in [x],\ c \in [c],\ x \neq c$$

einschließen, falls f auf $[x]$ konkav ist.

Bemerkung 1.5.8.

- *Die Berechnung einer Einschließung der Menge*

  $$\{s_g(x,c) |\ x \in [f]^x,\ c \in [f]^c,\ x \neq c\}$$

  wird für viele elementare (auch einige nicht differenzierbare) Funktionen ausführlich bei Ratz in [40] besprochen.

- *In Bräuer et al. [5] wird eine Computerimplementierung der oben beschriebenen Steigungsarithmetik beschrieben.*

1.6 Einschließung der Lösungsmenge eines linearen Gleichungssystems

In diesem Abschnitt stellen wir zwei Verfahren, den Intervall-Gauß-Algorithmus und das Verfahren von Rump, vor. Beide liefern eine garantierte Einschließung der Lösung eines linearen Gleichungssystems. Wir gehen von einem linearen Gleichungssystem

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{1.8}$$

mit $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ und $\boldsymbol{b} \in \mathbb{R}^n$ aus. Dabei seien die Matrix $\boldsymbol{A}$ und der Vektor $\boldsymbol{b}$ allerdings nicht exakt bekannt, sondern nur, dass $\boldsymbol{A} \in [\boldsymbol{A}]$ und $\boldsymbol{b} \in [\boldsymbol{b}]$ mit $[\boldsymbol{A}] \in \mathbb{IR}^{n\times n}$ und $[\boldsymbol{b}] \in \mathbb{IR}^n$ gelten. Somit liegt die Lösung von (1.8) sicherlich in

$$L := \{\boldsymbol{x} \in \mathbb{R}^n |\ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \text{ mit } \boldsymbol{A} \in [\boldsymbol{A}],\ \boldsymbol{b} \in [\boldsymbol{b}]\}. \tag{1.9}$$

Die beiden oben erwähnten Verfahren liefern nun einen Intervallvektor $[\boldsymbol{x}] \in \mathbb{IR}^n$, der L enthält und somit auch die Lösung von (1.8) einschließt.

1.6.1 Der Intervall-Gauß-Algorithmus

Der Intervall-Gauß-Algorithmus (IGA) (siehe Alefeld [1, S. 218]) ergibt sich als direkte Übertragung des bekannten Gauß-Algorithmus zum Lösen von Gleichungssystemen mit reellen Einträgen auf Gleichungssysteme mit Intervalleinträgen. In der Rekursion des klassischen Algorithmus werden nur alle reellen Operationen durch Intervalloperationen ersetzt. Somit ist $[\boldsymbol{x}] = \mathrm{IGA}([\boldsymbol{A}], [\boldsymbol{b}])$ das Ergebnis der folgenden Rekursion

$$[a]_{i,j}^{(k+1)} := \begin{cases} [a]_{i,j}^{(k)}, & 1 \le i \le k,\ 1 \le j \le n, \\ [a]_{i,j}^{(k)} - \dfrac{[a]_{i,k}^{(k)} \cdot [a]_{k,j}^{(k)}}{[a]_{k,k}^{(k)}}, & k < i \le n,\ k < j \le n, \\ 0, & \text{sonst}, \end{cases} \qquad i,j = 1,\dots,n.$$

$$[b]_{i}^{(k+1)} := \begin{cases} [b]_{i}^{(k)}, & 1 \le i \le k, \\ [b]_{i}^{(k)} - \dfrac{[a]_{i,k}^{(k)} \cdot [b]_{k}^{(k)}}{[a]_{k,k}^{(k)}}, & k < i \le n, \end{cases} \qquad i = 1,\dots,n,$$

$$[x]_i := \frac{1}{[a]_{i,i}^{(n)}} \left([b]_i^{(n)} - \sum_{j=i+1}^{n} [a]_{i,j}^{(n)} \cdot [x]_j \right), \qquad i = n,\dots,1$$

mit $[\boldsymbol{A}]^{(1)} := [\boldsymbol{A}]$ und $[\boldsymbol{b}]^{(1)} := [\boldsymbol{b}]$.

Aus dem nachfolgendem Satz folgt nun für L aus (1.9) die entscheidende Inklusion $L \subseteq [\boldsymbol{x}] = \mathrm{IGA}([\boldsymbol{A}], [\boldsymbol{b}])$:

Satz 1.6.1. *Sind die einzelnen Schritte des Intervall-Gauß-Algorithmus durchführbar (d. h. gilt $0 \notin [a]_{k,k}^{(k)}$ für $k = 1,\dots,n$), so gilt für $k = 1,\dots,n-1$*

$$\begin{aligned} &\{\boldsymbol{x} \in \mathbb{R}^n |\ \boldsymbol{A}^{(k)}\boldsymbol{x} = \boldsymbol{b}^{(k)} \text{ mit } \boldsymbol{A}^{(k)} \in [\boldsymbol{A}]^{(k)},\ \boldsymbol{b}^{(k)} \in [\boldsymbol{b}]^{(k)}\} \\ &\qquad \subseteq \{\boldsymbol{x} \in \mathbb{R}^n |\ \boldsymbol{A}^{(k+1)}\boldsymbol{x} = \boldsymbol{b}^{(k+1)} \text{ mit } \boldsymbol{A}^{(k+1)} \in [\boldsymbol{A}]^{(k+1)},\ \boldsymbol{b}^{(k+1)} \in [\boldsymbol{b}]^{(k+1)}\}. \end{aligned}$$

Bemerkung 1.6.2.

- *Die obige Rekursion des Intervall-Gauß-Algorithmus können wir nur dann durchführen, wenn in jedem Schritt jeweils $0 \notin [a]_{k,k}^{(k)}$ gilt. Gibt es einen Rekursionsschritt k, für den $0 \in [a]_{k,k}^{(k)}$ gilt, so können wir den Algorithmus eventuell dadurch fortsetzen, dass wir die k-te Zeile mit einer Zeile mit einem Index größer k vertauschen. Die Inklusionseigenschaft für L ändert sich dadurch nicht.*

- *Durch Pivotstrategien, d. h. durch gezieltes Vertauschen von Zeilen in jedem Iterationsschritt, können wir die Stabilität des Algorithmus erhöhen. Eine Strategie besteht beispielsweise darin, im k-ten Rekursionsschritt diejenige Zeile als k-te Zeile zu wählen, bei der das Element $[a]_{i,k}^{(k)}$ mit $i \geq k$, nach Äquilibrierung der Zeilen, betragsmäßig am größten ist.*

- *Der Intervall-Gauß-Algorithmus ist nur durchführbar, wenn jede Matrix $\boldsymbol{A} \in [\boldsymbol{A}]$ invertierbar ist. Diese Bedingung ist aber nur notwendig und nicht immer hinreichend.*

1.6.2 Das Verfahren von Rump

Das Verfahren von Rump (siehe Rump [42]) ist ein iteratives Verfahren zur gesicherten Lösung von linearen Gleichungssystemen. Üblicherweise liefert es bessere Einschließungen als der Intervall-Gauß-Algorithmus.

Das Verfahren beruht auf den Aussagen des folgenden Satzes, einer Anpassung des Satzes von Brouwer (siehe Satz 1.7.3) auf das gegebene Problem.

Satz 1.6.3. *Gegeben seien die Intervallmatrix $[\boldsymbol{A}] \in \mathbb{IR}^{n\times n}$ und der Intervallvektor $[\boldsymbol{b}] \in \mathbb{IR}^n$. Die Funktion $F : \mathbb{IR}^n \to \mathbb{IR}^n$ sei definiert durch*

$$F([\boldsymbol{y}]) := \tilde{\boldsymbol{x}} + \boldsymbol{C}([\boldsymbol{b}] - \boldsymbol{A}[\boldsymbol{y}]) + (\boldsymbol{I} - \boldsymbol{C}[\boldsymbol{A}])([\boldsymbol{y}] - \tilde{\boldsymbol{x}}),$$

wobei $\tilde{\boldsymbol{x}} \in \mathbb{R}^n$ und $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ beliebig gewählt seien und $\boldsymbol{I}$ die Einheitsmatrix bezeichne. Gilt dann für ein $\Omega \in \mathbb{IR}^n$

$$F(\Omega) \subsetneqq \Omega, \tag{1.10}$$

so gilt:

1. *Die Matrix $\boldsymbol{C}$ und jede Matrix $\boldsymbol{A} \in [\boldsymbol{A}]$ sind regulär.*

2. *Jedes Gleichungssystem $\boldsymbol{Ax} = \boldsymbol{b}$ mit $\boldsymbol{A} \in [\boldsymbol{A}]$ und $\boldsymbol{b} \in [\boldsymbol{b}]$ ist eindeutig lösbar und*

$$\forall \boldsymbol{A} \in [\boldsymbol{A}],\ \forall \boldsymbol{b} \in [\boldsymbol{b}] : \quad \boldsymbol{Ax} = \boldsymbol{b} \qquad \Longrightarrow \qquad \boldsymbol{x} \in \bigcap_{i\geq 0} F^i(\Omega).$$

Bemerkung 1.6.4.

- *Üblicherweise wählen wir die Matrix $\boldsymbol{C}$ aus Satz 1.6.3 als Näherung für die Matrix $(\mathrm{mid}([\boldsymbol{A}]))^{-1}$, um so möglichst $\rho(|\boldsymbol{I} - \boldsymbol{C}[\boldsymbol{A}]|) < 1$ zu erreichen. Dies sichert dann nach Alefeld und Herzberger [1, S. 178] die Konvergenz des Iterationsverfahrens $[\boldsymbol{y}]^{k+1} = F([\boldsymbol{y}]^k)$.*

- *Den Vektor $\tilde{\boldsymbol{x}}$ aus Satz 1.6.3 wählen wir gewöhnlich als Näherung für den Vektor $(\mathrm{mid}([\boldsymbol{A}]))^{-1}\,\mathrm{mid}([\boldsymbol{b}])$, damit $[\boldsymbol{y}] - \tilde{\boldsymbol{x}}$ nahe am Nullvektor liegt.*

Die Einschließung der Menge L aus (1.9) wird beim Verfahren von Rump, ausgehend von einer Gleitkommanäherung, durch Vergröberung des Einschließungsintervalls durch $[\boldsymbol{y}] = [1-\varepsilon, 1+\varepsilon][\boldsymbol{x}] + [\boldsymbol{\delta}]$ gewonnen. Dabei wählen wir beispielsweise $\varepsilon = 0.1$ und setzen für den Intervallvektor $[\boldsymbol{\delta}] := ([-\delta, \delta])$, wobei δ die kleinste darstellbare Maschinenzahl bezeichnet. Die Vergröberung findet hierbei iterativ solange statt, bis das Intervall unter der Abbildung F in sich selbst abgebildet wird und somit nach Satz 1.6.3 die Menge L eingeschlossen wird oder wegen zu vieler Iterationsschritte abgebrochen wird.

In Algorithmus 1.6.5 beschreiben wir das Verfahren von Rump in Pseudocode. Als kleine Modifikation zum Satz 1.6.3 iterieren wir hierbei mit dem Differenzvektor $[\boldsymbol{y}] - \tilde{\boldsymbol{x}}$, der nahe am Nullvektor liegen sollte.

Algorithmus 1.6.5 (Verfahren von Rump).

$[\boldsymbol{\delta}] = ([-\delta, \delta])_{i=1,\dots,n}$
$\boldsymbol{C} = (\mathrm{mid}([\boldsymbol{A}]))^{-1}$
$[\tilde{\boldsymbol{C}}] = \boldsymbol{I} - \boldsymbol{C}[\boldsymbol{A}]$
$\tilde{\boldsymbol{x}} = \boldsymbol{C}\boldsymbol{b}$
$[\boldsymbol{z}] = \boldsymbol{C}(\boldsymbol{b} - [\boldsymbol{A}]\tilde{\boldsymbol{x}})$
$[\boldsymbol{x}] = [\boldsymbol{z}]$
$step = 0$

`Do` *(Iteration)*
    $step = step + 1$
    $[\boldsymbol{y}] = [1-\varepsilon, 1+\varepsilon][\boldsymbol{x}] + [\boldsymbol{\delta}]$ *(Vergröberung)*
    $[\boldsymbol{x}] = [\boldsymbol{z}] + [\tilde{\boldsymbol{C}}][\boldsymbol{y}]$
`Until` $([\boldsymbol{x}] \subseteq \mathrm{int}([\boldsymbol{y}])$ `Or` $step > max\_step)$ *(auf Inklusion prüfen)*

`If` $([\boldsymbol{x}] \subseteq \mathrm{int}([\boldsymbol{y}]))$ `Then`
    Lösungsmenge liegt in $\tilde{\boldsymbol{x}} + [\boldsymbol{x}]$
`Else`
    Lösung konnte nicht eingeschlossen werden.

1.7 Fixpunktsätze

Nachfolgend stellen wir die Fixpunktsätze von Banach, Brouwer und Schauder vor (siehe Heuser [19]), die Bedingungen angeben, unter denen eine Lösung x^* der Fixpunktgleichung

$$x = f(x)$$

existiert. Viele Probleme der Mathematik können auf die Untersuchung von Fixpunktgleichungen zurückgeführt werden, weswegen Fixpunktsätze ein zentrales Hilfsmittel darstellen.

Für den Banachschen Fixpunktsatz, dem wohl bekanntesten Fixpunktsatz, stellt sich der Begriffe der Kontraktion als zentral heraus.

Definition 1.7.1 (Kontraktion). *Eine Abbildung f des Banachraums X in sich selbst heißt* kontrahierend *auf $D \subseteq X$, wenn es eine Konstante $\lambda < 1$ gibt, so dass*

$$||f(x) - f(y)||_X \leq \lambda ||x - y||_X$$

für alle $x, y \in D$ gilt.

Mit diesen Begriffen formulieren wir nun den Banachschen Fixpunktsatz, der neben Aussagen über die Existenz eines Fixpunktes auch eine Berechnungsvorschrift angibt, um diesen zu bestimmen.

Satz 1.7.2 (Banachscher Fixpunktsatz). *Sei D eine nichtleere abgeschlossene Teilmenge eines Banachraums und sei $f : D \to D$ kontrahierend auf D mit Kontraktionskonstante $\lambda < 1$. Dann besitzt f in D genau einen Fixpunkt x^* und für einen beliebigen Startwert $x_0 \in D$ konvergiert die Folge der sukzessiven Approximationen*

$$x_{k+1} = f(x_k), \qquad k = 1, 2, \ldots$$

gegen den Fixpunkt x^*. Des weiteren gilt die Abschätzung

$$||x_k - x^*||_X \leq \frac{1}{1-\lambda} ||x_k - x_{k+1}||_X \leq \frac{\lambda^k}{1-\lambda} ||x_1 - x_0||_X, \qquad k = 1, 2, \ldots.$$

Verzichten wir auf die recht restriktive Forderung, dass f eine kontrahierende Abbildung ist, so können wir nicht mehr auf die Eindeutigkeit des Fixpunktes hoffen. Auch die im Banachschen Fixpunktsatz angegebene Berechnungsvorschrift, um den Fixpunkt durch sukzessive Approximation $x_{k+1} = f(x_k)$, $k = 1, 2, \ldots$, zu bestimmen, geht verloren. Beschränken wir uns zuerst auf endlichdimensionale Räume, so erhalten wir den Brouwerschen Fixpunktsatz:

Satz 1.7.3 (Brouwerscher Fixpunktsatz). *Sei K eine nichtleere, konvexe und kompakte Teilmenge des Raumes $\mathbb{R}^n$ und $f : K \to K$ eine stetige Abbildung. Dann hat f mindestens einen Fixpunkt in K.*

Eine Verallgemeinerung des Brouwerschen Fixpunktsatzes auf unendlichdimensionale Räume führt uns zum Schauderschen Fixpunktsatz:

Satz 1.7.4 (Schauderscher Fixpunktsatz). *Sei K eine nichtleere konvexe abgeschlossene Teilmenge eines normierten Raumes, $f : K \to K$ eine stetige Abbildung und $f(K)$ relativ kompakt, dann besitzt f mindestens einen Fixpunkt in K.*

1.8 Parallele Programmiermodelle

In diesem Abschnitt beschäftigen wir uns mit Programmiermodellen, d. h. mit Abstraktionen von real existierenden Rechnersystemen. Mit dem Begriff des Rechnersystems bezeichnen wir dabei die Gesamtheit der Hardware und der darauf verfügbaren Systemsoftware wie Compilern und Laufzeitbibliotheken. Die Modelle sollen einerseits einfach sein, um die Rechnersysteme in möglichst große Klassen unterteilen zu können, andererseits sollen sie die realen Rechnersysteme aber auch gut beschreiben. Die nachfolgende Darstellung lehnt sich eng an das Vorgehen in Rauber und Rünger [41] an.

Gründe für die Verwendung von Programmiermodellen sind:

- Sie bieten eine Abstraktion von der speziellen Hardware und ermöglichen somit, parallele Algorithmen herstellerunabhängig zu formulieren.
- Sie erlauben eine Abschätzung der Laufzeit des parallelen Programms und somit einen Vergleich der Effizienz verschiedener paralleler Algorithmen.
- Sie gestatten asymptotische Aussagen über Algorithmen, die auf Rechenanlagen nicht zu gewinnen sind.

Ein Programmiermodell beschreibt unter anderem:

- Die vom Programmierer nutzbaren Ebenen der Parallelität (z. B. Parallelität auf Instruktionsebene, Parallelität in Schleifen, Parallelität auf Prozessebene).
- Ob die Parallelität implizit oder explizit programmiert werden kann.
- Wie die parallel ablaufenden Prozesse erzeugt werden.
- Wie der Datenaustausch zwischen Prozessen erfolgt.

1.8.1 Ebenen der Parallelität

Bei Programmen unterscheiden wir verschiedene Ebenen, auf denen Befehle parallel ausgeführt werden können. Die Unterteilung in diese Ebenen erfolgt anhand der so genannten *Granularität*, die beschreibt, wie viele Instruktionen parallel und unabhängig voneinander ausgeführt werden können. Von grobkörniger Granularität sprechen wir, wenn sehr viele Instruktionen parallel ausgeführt werden können, bevor Datenaustausch nötig ist, also beispielsweise bei Parallelität auf Prozessebene. Bei feinkörniger Granularität können nur wenige Instruktionen parallel ausgeführt werden, bevor wieder Kommunikation nötig ist, wie etwa bei Parallelität auf Instruktionsebene. Datenparallelität und parallele Schleifen weisen üblicherweise eine mittlere Granularität auf.

Parallelität auf Instruktionsebene

Bei der Abarbeitung eines Programms können mehrere voneinander unabhängige Instruktionen parallel in einem Prozessor ausgeführt werden.

Das folgende Beispiel verdeutlicht diese Möglichkeit bei Prozessoren mit mehreren Ausführungseinheiten: Es soll der Ausdruck $c = a + b$ berechnet werden. Ein Prozessor mit nur einer Ausführungseinheit führt dazu die Instruktionen

(1) `Lade` a `in Register1`
(2) `Lade` b `in Register2`
(3) `Addiere Register1 und Register2 und lege Ergebnis in Register1 ab`
(4) `Schreibe Register1 nach` c

aus. Ein Prozessor mit zwei Ausführungseinheiten berechnet hingegen in kürzerer Zeit:

(1) `Lade` a `in Register1`   `Lade` b `in Register2`
(2) `Addiere Register1 und Register2 und lege Ergebnis in Register1 ab`
(3) `Schreibe Register1 nach` c

Die Verteilung der einzelnen Instruktionen auf die Berechnungseinheiten erfolgt zumeist durch einen in Hardware auf der CPU realisierten Instruktionsscheduler.

Bei VLIW-Prozessoren (Very-Long-Instruction-Word), wie etwa dem Itanium-Prozessor, werden in einem Instruktionswort mehrere einzelne Instruktionen übertragen, die parallel abgearbeitet werden. Die Zusammenstellung der einzelnen Instruktionen zu einem Instruktionswort erfolgt durch den Compiler. Er ist dafür verantwortlich, dass die einzelnen Instruktionen voneinander unabhängig ausgeführt werden können und nach Möglichkeit alle Ausführungseinheiten beschäftigt sind.

Parallele Schleifen

Schleifen sind ein wichtiger Bestandteil jeder Programmiersprache. Sie erlauben die wiederholte Ausführung von Anweisungen. Sind diese Anweisungen untereinander unabhängig, so können sie auch parallel von mehreren Prozessoren bearbeitet werden.

Beispielsweise kann die serielle Schleife

```
do i=1,n
  a(i) = b(i) + c(i)
enddo
```

bei der die einzelnen Anweisungen voneinander unabhängig zu bearbeiten sind, in der Programmiersprache High Performance Fortran (HPF) als forall-Schleife

```
!HPF$ forall (i = 1:n) a(i) = b(i) + c(i)
```

realisiert werden, die dann vom Compiler auf mehrere Prozessoren verteilt werden kann.

Datenparallelität

Sind auf unterschiedlichen Daten die gleichen Operationen durchzuführen, so kann man eine Parallelisierung durch gleichmäßiges Verteilen der Datenelemente auf die zur Verfügung stehenden Prozessoren erreichen.

Ein Beispiel aus der Programmiersprache Fortran90, bei der Datenparallelität genutzt wird, ist die Vektoranweisung

```
a(1:n) = b(1:n) + c(1:n)
```

bei der der Compiler die einzelnen Additionen und Zuweisungen von verschiedenen Prozessoren durchführen lassen kann. Ebenso können die Matrixaddition

```
A(1:n,1:n) = B(1:n,1:n) + C(1:n,1:n)
```

und die Matrixmultiplikation

```
A = matmul(B, C)
```

datenparallel durchgeführt werden.

Parallelität auf Prozessebene

Bei entsprechender Unterstützung durch das Betriebssystem in Form von so genanntem *Multitasking* können auf einem Rechner mehrere unterschiedliche oder gleiche Prozesse parallel ablaufen. Die Unterteilung der Berechnung in einzelne Teilprozesse ist durch den Benutzer festzulegen.

1.8.2 Implizite und explizite Parallelität

Programmiermodelle unterscheiden sich auch in der Art, wie die Parallelität dargestellt wird.

Bei der impliziten Darstellung von Parallelität werden nur die notwendigen Berechnungen und ihre Abhängigkeiten untereinander, nicht aber die Berechnungsreihenfolge oder die Aufteilung der Berechnungen und Daten festgelegt. Die automatisch parallelisierenden Compiler folgen diesem Ansatz. Sie versuchen selbstständig ein vorgegebenes sequentielles Programm in ein paralleles Programm zu überführen. Dabei steht der Compiler vor der schwierigen Aufgabe, die Berechnungen und Daten so auf die Prozessoren zu verteilen, dass daraus eine möglichst gute Lastverteilung resultiert und gleichzeitig der notwendige Datenaustausch zwischen den Prozessoren niedrig bleibt.

Bei der expliziten Darstellung von Parallelität übernimmt der Programmierer die Aufteilung der Daten und Berechnungen auf die einzelnen Prozessoren. Ist Informationsaustausch zwischen den Prozessoren notwendig, so muss dieser explizit programmiert werden. Vertreter dieses Programmieransatzes sind beispielsweise die Message-Passing-Bibliotheken MPI (siehe [29], [30]) und PVM (siehe Geist et al. [10]).

1.8.3 Erzeugung von Prozessen

Die einzelnen Teilaufgaben von parallelen Programmen werden in parallel zueinander laufenden Prozessen realisiert. Programmiermodelle unterscheiden sich durch die Art, wie diese Prozesse erzeugt werden.

Fork-Join-Konstrukt

Beim Fork-Join-Konstrukt handelt es sich um eine Möglichkeit, Prozesse dynamisch, d. h. während der Laufzeit, zu erzeugen und auch wieder zu beenden. Ein bereits existierender Prozess erzeugt durch den Aufruf der Funktion `fork` einen so genannten Kindprozess, der eine Kopie des Erzeugerprozesses ist. Durch den gemeinsamen Aufruf der Funktion `join` werden die beiden Prozesse wieder zu einem zusammengeführt, der Kindprozess wird beendet, der Erzeugerprozess arbeitet weiter. Ruft zuerst der Erzeugerprozess die Funktion `join` auf, so wartet er solange, bis auch der Kindprozess diese Funktion aufruft. Ruft zuerst der Kindprozess die Funktion `join` auf, so wird er sofort beendet. Da beide Prozesse, Kind- und Erzeugerprozess, die gleichen Daten und den gleichen Programmcode besitzen, müssen sie, wenn sie unterschiedliche Programmteile ausführen sollen, die Programmausführung von der Prozessnummer abhängig machen.

SPMD

Beim SPMD-Modell (Single Program Multiple Data) wird bei Programmstart eine feste Anzahl von Prozessen erzeugt. Diese Anzahl der Prozesse ist statisch, d. h. sie ändert sich über die gesamte Laufzeit des Programms nicht. Alle Prozesse arbeiten das gleiche Programm ab. Sollen von den Prozessen unterschiedliche Programmteile ausgeführt werden, so muss die Ausführung dieser Programmteile von der Prozessnummer abhängig gemacht werden.

Eine Realisierung des SPMD-Modells finden wir beispielsweise in der Programmbibliothek MPI. Zum Starten von zwei Prozessen, die beide das Programm `program` ausführen, verwenden wir dort das Kommando

```
mpirun -np 2 program
```

Mit dem Befehl `MPI_Comm_rank` kann ein mit der Bibliothek MPI parallelisiertes Programm die eigene Prozessnummer erfragen. Der Befehl `MPI_Comm_size` liefert die Anzahl der mit `mpirun` gestarteten Prozesse.

1.8.4 Datenaustausch

Ein wichtiger Aspekt bei der Parallelisierung ist der Datenaustausch zwischen den Prozessen. Je nach zu Grunde liegender Hardware ergeben sich die beiden nachfolgenden Programmierkonzepte.

Rechner mit gemeinsamem Speicher

Bei Rechnern mit gemeinsamem Speicher können Daten zwischen den Prozessen über den gemeinsamen Speicher ausgetauscht werden. Jeder Prozess besitzt einen Speicherbereich für private und einen Speicherbereich für gemeinsame Variablen. Der Datenaustausch zwischen Prozessen erfolgt ausschließlich über die gemeinsamen Variablen. Private Variablen können von anderen Prozessen nicht zugegriffen werden. Will ein Prozess P_i einem Prozess P_j Daten übermitteln, so stellt er diese durch Umkopieren in gemeinsame Variablen zur Verfügung. Prozess P_j kann sie dann zur weiteren Verarbeitung in seine privaten Variablen umkopieren.

Mit Hilfe von Synchronisationsoperationen muss dabei verhindert werden, dass Prozess P_j aus den gemeinsamen Variablen liest, bevor Prozess P_i die gewünschten Daten zur Verfügung gestellt hat. Ebenso muss das gleichzeitige Schreiben mehrerer Prozesse auf den gleichen gemeinsamen Variablen unterbunden werden, da sonst Inkonsistenzen auftreten können.

Eine solche notwendige Synchronisation zwischen den Prozessen P_i und P_j kann mit Hilfe eines so genannten binären *Semaphors* realisiert werden. Ein binäres Semaphor besteht aus einer Variable s, die nur die beiden Werte 1 ($\hat{=}$ ein Prozess schreibt auf die gemeinsamen Variablen) oder 0 ($\hat{=}$ kein Prozess schreibt auf die gemeinsamen Variablen) annehmen kann, und den beiden unteilbaren Funktionen `signal(s)` und `wait(s)`. Die Funktion `signal(s)` setzt die Variable s auf den Wert 0. Die Funktion `wait(s)` wartet bis s den Wert 0 hat, setzt dann den Wert von s auf 1 und fährt mit der Befehlsausführung fort. Die Grundbelegung von s ist 0, d. h. kein anderer Prozess schreibt auf die gemeinsamen Variablen. Um das gleichzeitige Schreiben von mehreren Prozessen auf gemeinsame Variablen zu verhindern, muss nun jeder Prozess vor dem eigentlichen Schreiben die Funktion `wait(s)` und nach dem Schreiben die Funktion `signal(s)` ausführen.

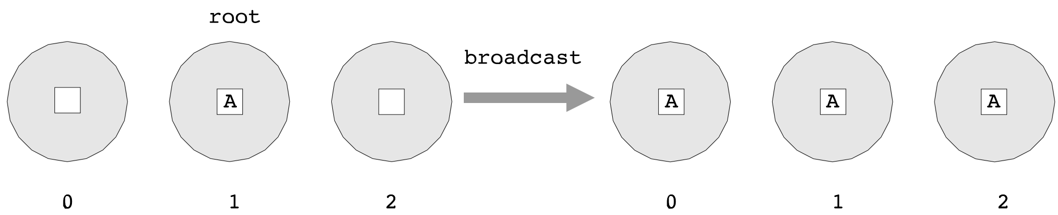
Rechner mit verteiltem Speicher

Bei Rechnern mit verteiltem Speicher kann der Datenaustausch nicht wie bei Rechnern mit gemeinsamem Speicher über einen allen Prozessen zugänglichen Speicherbereich erfolgen, sondern es sind explizite Kommunikationsanweisungen notwendig, um die Daten vom lokalen Speicherbereich eines Prozesses in den lokalen Speicherbereich des anderen Prozesses zu transferieren. Man spricht in diesem Zusammenhang auch von Nachrichtenaustausch (englisch: *message passing*). Man unterscheidet etwa die nachfolgenden Kommunikationsarten.

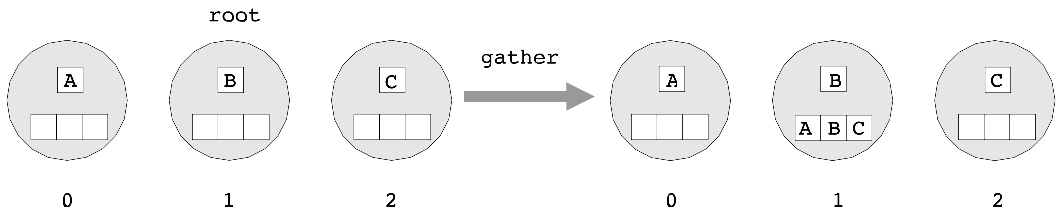
Punkt-zu-Punkt-Kommunikation Bei dieser Kommunikationsart sind immer genau zwei Prozesse beteiligt. Der eine Prozess versendet (Operation `send`) eine Nachricht aus seinem lokalen Speicherbereich, der andere Prozess empfängt diese Nachricht (Operation `receive`) und legt sie in seinem lokalen Speicherbereich ab.

Globale Kommunikation Bei der globalen Kommunikation sind immer alle Prozesse beteiligt. Einer dieser Prozesse übernimmt eine Sonderaufgabe, weil von ihm die Datenverteilung ausgeht oder auf ihm die Datensammlung stattfindet. Wir bezeichnen diesen Prozess im Folgenden als root-Prozess.

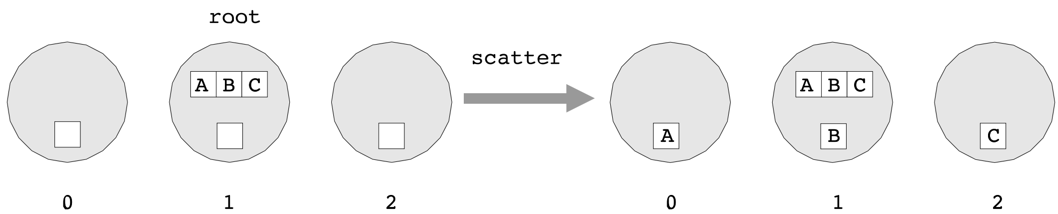
- **Rundruf (`broadcast`)** Bei dieser Kommunikationsart versendet der root-Prozess eine Nachricht an alle Prozesse, so dass nach der Kommunikation alle Prozesse diese Nachricht in ihrem lokalen Speicher vorliegen haben.



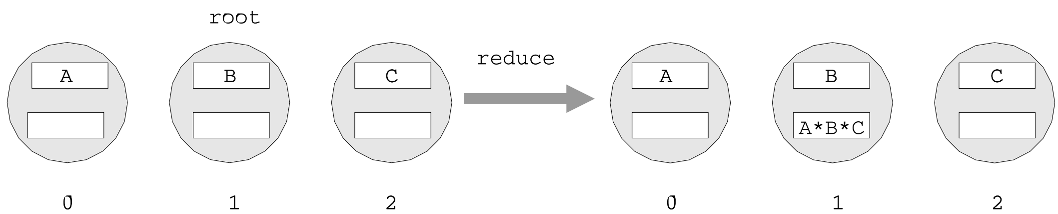
- **Sammeloperation** (`gather`) Bei dieser Kommunikationsart empfängt der root-Prozess von allen Prozessen eine Nachricht. Nach der Kommunikation liegt also im lokalen Speicher des root-Prozesses die Vereinigung aller von den Prozessen gesendeten Nachrichten vor.



- **Verteilungsoperation** (`scatter`) Bei dieser Kommunikationsart werden Nachrichten vom root-Prozess gleichmäßig an alle Prozesse verteilt. Nach der Kommunikation hat jeder Prozess einen Teil der Nachrichten in seinem lokalen Speicher vorliegen.



- **Reduktionsoperation** (`reduce`) Bei dieser Kommunikationsart wird von allen Prozessen gemeinsam eine Rechenoperation, etwa eine Addition, durchgeführt und das Ergebnis dieser Berechnung im lokalen Speicher des root-Prozesse abgelegt.



Kapitel 2

Das Einschließungsverfahren

2.1 Motivation

Integralgleichungen, wie sie beispielsweise in der Modellierung physikalischer Vorgänge vorkommen, seien sie linear oder auch nichtlinear, sind meist zu komplex, um sie in geschlossener Form lösen zu können. Selbst um das Verhalten von Lösungen solcher Integralgleichungen qualitativ untersuchen zu können, sind wir auf Näherungslösungen angewiesen. Die Existenz einer solchen Näherungslösung gibt allerdings noch keinen Hinweis auf die Existenz einer exakten Lösung. Auch bei sehr kleinem Defekt der Näherungslösung ist die Existenz einer exakten Lösung keinesfalls gesichert. Selbst wenn die Existenz einer exakten Lösung der Integralgleichung bekannt ist, wissen wir meist nicht, wie weit unsere Näherungslösung von dieser entfernt ist, da Konvergenzsätze im Allgemeinen nur qualitative, asymptotische Aussagen machen. Im Extremfall liefert ein numerisches Verfahren zwei dicht beieinander liegende Näherungslösungen, für die wir dennoch nicht entscheiden können ob zwei, eine oder keine exakte Lösung(en) in ihrer Nähe liegen.

Mit dem im nächsten Abschnitt beschriebenen Algorithmus wollen wir diese Probleme numerischer Lösungen beseitigen. Wir geben einzelne Schritte an, die, falls sie durchführbar sind, die Existenz einer exakten Lösung in einer Umgebung einer zuvor berechneten Näherungslösung garantieren und den Fehler der Näherungslösung garantiert abschätzen. Die einzelnen Schritte des Algorithmus sind dabei so gestaltet, dass sie auf einem Computer vollautomatisch ablaufen können.

Der Algorithmus folgt den von Plum in [35, 36, 37] vorgestellten Ideen für gewöhnliche und partielle Randwertprobleme. Die einzelnen Schritte werden auf das hier untersuchte Problem der nichtlinearen Integralgleichung übertragen.

Im nächsten Abschnitt beschreiben wir zunächst das grundsätzliche Vorgehen und formulieren den entscheidenden Satz zur Existenz und Einschließung der Lösung einer nichtlinearen Fredholmschen Integralgleichung zweiter Art vom Urysohntyp. Anschließend präzisieren wir die einzelnen Schritte und beschreiben ihre Implementierung, insbesondere die Parallelisierung, die die Rechenzeit deutlich verkürzt.

2.2 Das Algorithmengerüst

In diesem Abschnitt beweisen wir einen Satz (Theorem 2.2.1), der die Existenz einer Lösung der nichtlinearen Fredholmschen Integralgleichung zweiter Art vom Urysohntyp

$$x(s) - \int_a^b k(s,t,x(t))\,dt = y(s), \qquad s \in [a,b] \tag{2.1}$$

garantiert und eine Fehlerabschätzung liefert.

Wir verwenden im Folgenden häufig die Operatorschreibweise

$$(\mathcal{I} - \mathcal{K})x = y \tag{2.2}$$

mit der Identität $\mathcal{I}$ auf $(C([a,b]), ||.||_\infty)$ und dem Fredholm-Operator

$$(\mathcal{K}x)(s) := \int_a^b k(s,t,x(t))\,dt.$$

Da wir im Folgenden nur noch den Banachraum $X = (C([a,b]), ||.||_\infty)$ betrachten, schreiben wir sowohl für die Norm $||.||_X$ als auch die Operatornorm $||.||_{X\to X}$ kurz $||.||$.

Um Theorem 2.2.1 anwenden zu können, benötigen wir für Gleichung (2.1) eine numerische Lösung, ohne zuvor überhaupt die Existenz einer Lösung sichergestellt zu haben. Erfüllt diese Näherung ω gewisse Voraussetzungen, so garantiert der Satz die Lösbarkeit der Gleichung (2.1) und gibt eine Fehlerschranke α für die Näherung ω an.

Eine Formulierung von Existenz- und Einschließungsaussagen wird natürlich nicht für beliebige Kerne k und rechte Seiten y möglich sein, sondern erfordert gewisse Glattheitseigenschaften dieser beiden Funktionen. Konkret fordern wir für den Kern k und die partiellen Ableitungen k_3, $k_{1,3}$, dass sie aus $C([a,b] \times [a,b] \times \mathbb{R})$ sind, und für die rechte Seite y, dass sie aus $C^1([a,b])$ ist.

Da dieser Satz eher algorithmischer Natur ist, müssen wir zunächst die Teilschritte vorstellen, bevor wir den Satz formulieren können.

Schritt 1: Berechne eine Näherungslösung ω der nichtlinearen Fredholmschen Integralgleichung zweiter Art vom Urysohntyp (2.1). Diese Näherungslösung ω liege in der Form

$$\omega(s) := y(s) + \sum_{j=1}^{n} w_{n,j}\, k(s, t_{n,j}, \omega_{n,j})$$

vor, die sich in natürlicher Weise aus der Nyström-Diskretisierung ergibt. Dabei seien $w_{n,j}$, $j = 1, \ldots, n$, gegebene Quadraturgewichte, $t_{n,j}$, $j = 1, \ldots, n$, gegebene Quadraturknoten und $\omega_{n,j} = \omega(t_{n,j})$, $j = 1, \ldots, n$. Es gilt somit $\omega \in C^1([a,b])$. (Siehe Abschnitt 2.3.)

Schritt 2: Berechne eine Konstante $\delta \geq 0$, die den Defekt der Näherungslösung ω

$$(d(\omega))(s) := \omega(s) - \int_a^b k(s,t,\omega(t))\,dt - y(s)$$

in der Unendlichnorm nach oben beschränkt:

$$||d(\omega)|| \leq \delta \tag{2.3}$$

(siehe Abschnitt 2.4).

Schritt 3: Zeige mit Hilfe der Neumannschen Reihe, dass der am Punkt ω mit Hilfe der Fréchetableitung linearisierte Operator $\mathcal{I} - \mathcal{K}$

$$\mathcal{L} : \begin{cases} C([a,b]) \to C([a,b]), \\ x \mapsto (\mathcal{I} - \mathcal{K})'[\omega]x = x - \int_a^b k_3(.,t,\omega(t))x(t)\,dt = (\mathcal{I} - \tilde{\mathcal{K}})x \end{cases}$$

bijektiv und damit invertierbar ist (dass die Fréchetableitung die angegebene Form hat, zeigen wir in Hilfssatz 2.3.1). Berechne dann eine Konstante K, welche die Operatornorm der Inversen des Operators $\mathcal{L}$ nach oben beschränkt:

$$||\mathcal{L}^{-1}|| \leq K \tag{2.4}$$

(siehe Abschnitt 2.5).

Schritt 4: Berechne eine monoton wachsende Funktion $G : \mathbb{R}_+ \to \mathbb{R}_+$ mit $G(\alpha) = o(\alpha)$ für $\alpha \to 0$, die den Operator

$$(\mathcal{G}x)(s) := \int_a^b \Big(k_3(s,t,\omega(t))x(t) - \big[k(s,t,\omega(t)+x(t)) - k(s,t,\omega(t))\big]\Big)\,dt$$

durch

$$||\mathcal{G}x|| \leq G(||x||) \qquad \text{für alle } x \in C([a,b]) \tag{2.5}$$

abschätzt (siehe Abschnitt 2.6).

Schritt 5: Bestimme eine Konstante $\alpha \geq 0$, die der Bedingung

$$\delta \leq \frac{\alpha}{K} - G(\alpha) \tag{2.6}$$

genügt (siehe Abschnitt 2.7).

Nun sind wir in der Lage, einen Satz zu formulieren, der die Existenz einer Lösung von (2.1) gewährleistet und der eine Fehlerschranke für die Näherungslösung ω liefert.

Theorem 2.2.1. *Seien $k, k_3, k_{1,3} \in C([a,b] \times [a,b] \times \mathbb{R})$, $y \in C^1([a,b])$. Es sei ω die Näherungslösung aus Schritt 1. Sind die Voraussetzungen in Schritt 2 bis Schritt 5 erfüllt, so besitzt die nichtlineare Fredholmsche Integralgleichung zweiter Art vom Urysohntyp*

$$x(s) - \int_a^b k(s,t,x(t))\,dt = y(s), \qquad s \in [a,b] \tag{2.7}$$

eine Lösung $x \in C([a,b])$. In diesem Fall gilt die Einschließung

$$\boxed{||\omega - x|| \leq \alpha}.$$

Beweis. Wir formen (2.7) in eine äquivalente Fixpunktgleichung für den Fehler $\tilde{x} := x - \omega$ um:

$$\begin{aligned}
& (\mathcal{I} - \mathcal{K})x = y \qquad \Longleftrightarrow \qquad (\mathcal{I} - \mathcal{K})(\tilde{x} + \omega) = y \\
\Longleftrightarrow \qquad & \tilde{x} + \omega - \mathcal{K}(\tilde{x} + \omega) = y \qquad \Longleftrightarrow \qquad \tilde{x} - \mathcal{K}(\tilde{x} + \omega) + \mathcal{K}\omega = -\omega + \mathcal{K}\omega + y \\
\Longleftrightarrow \qquad & \tilde{x} - \tilde{\mathcal{K}}\tilde{x} + \tilde{\mathcal{K}}\tilde{x} - \mathcal{K}(\tilde{x} + \omega) + \mathcal{K}\omega = -d(\omega) \qquad \Longleftrightarrow \qquad \mathcal{L}\tilde{x} + \mathcal{G}\tilde{x} = -d(\omega) \\
\Longleftrightarrow \qquad & \tilde{x} = \mathcal{L}^{-1}(-d(\omega) - \mathcal{G}\tilde{x}) =: \mathcal{T}\tilde{x}.
\end{aligned}$$

Wie wir nachfolgend zeigen werden, gilt:

1. $\mathcal{T} : C([a,b]) \to C([a,b])$ ist kompakt.

2. $\mathcal{T}D \subseteq D$ mit $D := \{u \in C([a,b]) : ||u|| \leq \alpha\}$.

Somit erhalten wir nach dem Schauderschen Fixpunktsatz (siehe Satz 1.7.4), dass der Operator $\mathcal{T}$ in D mindestens einen Fixpunkt $\tilde{x}^*$ besitzt. Aus obigen Umformungen folgt nun, dass $x^* = \omega + \tilde{x}^*$ die Integralgleichung (2.7) löst.

Es verbleiben also zwei Behauptungen zu zeigen:

1. $\mathcal{T} \in K(C([a,b]))$:

   - Nach dem dritten Schritt ist der Operator $\mathcal{L}$ bijektiv und die Inverse beschränkt. Es gilt somit $\mathcal{L}^{-1} \in L(C([a,b]))$.
   - Die Funktion $-d(\omega) - \mathcal{G}x$ liegt nach den Glattheitsvoraussetzungen an den Kern k und an die rechte Seite y für alle $x \in C([a,b])$ in $C^1([a,b])$.
   - Nach den Glattheitsvoraussetzungen an den Kern k_3 liegt der Operator $\tilde{\mathcal{K}}$ in $L(C([a,b]), C^1([a,b]))$. Wie im letzten Punkt schon gezeigt, gilt $\mathcal{L}^{-1} \in L(C([a,b]))$. Nach Satz 1.1.15 gilt dann sogar $\mathcal{L}^{-1} \in L(C^1([a,b]))$.
   - Die Einbettung $\mathcal{I} : C^1([a,b]) \to C([a,b])$ ist nach Satz 1.1.9 kompakt.
   - Die Komposition $\mathcal{T}x = \mathcal{I} \circ \mathcal{L}^{-1}(-d(\omega) - \mathcal{G}x)$ ist nach Satz 1.1.8 kompakt.

2. $\mathcal{T} : C([a, b]) \to C([a, b])$ ist eine Selbstabbildung auf D:

   Sei $u \in D$ beliebig, fest. Dann gilt

$$\begin{aligned} ||\mathcal{T}u|| &= ||\mathcal{L}^{-1}(-d(\omega) - \mathcal{G}u)|| \leq ||\mathcal{L}^{-1}|| \, || - d(\omega) - \mathcal{G}u|| \overset{(2.4)}{\leq} K|| - d(\omega) - \mathcal{G}u|| \\ &\overset{\triangle\text{-Ungl.}}{\leq} K(||d(\omega)|| + ||\mathcal{G}u||) \overset{(2.3),(2.5)}{\leq} K(\delta + G(\alpha)) \overset{(2.6)}{\leq} \alpha, \end{aligned}$$

   also $\mathcal{T}u \in D$.

$\square$

2.3 Berechnung einer Näherungslösung ω

In diesem Abschnitt beschreiben wir, wie eine Näherungslösung ω der nichtlinearen Fredholmschen Integralgleichung zweiter Art vom Urysohntyp

$$x(s) - \int_a^b k(s, t, x(t)) \, dt = y(s), \qquad s \in [a, b] \tag{2.8}$$

mittels des Newton-Verfahrens im Banachraum $C([a, b])$ berechnet werden kann. Zuerst wird hierzu die Fréchet-Ableitung des Integraloperators gebildet und danach die Anwendung des Newton-Verfahrens in $C([a, b])$ formuliert. Um einen guten Startwert für die Newton-Iteration zu erhalten, werden wir eine einfache Homotopiemethode vorstellen. Anschließend diskutieren wir das Nyström-Verfahren, um die in jedem Newton-Schritt auftretende lineare Fredholmsche Integralgleichung zweiter Art näherungsweise zu lösen. Durch ein Zwei-Gitter-Verfahren können wir die Berechnung der Nyström-Lösung wesentlich beschleunigen.

Alternativ zu dem oben beschriebenen Vorgehen können wir die Integralgleichung auch zuerst mit Hilfe der Nyström-Diskretisierung in ein n-dimensionales nichtlineares Gleichungssystem überführen und dieses dann mit dem Newton-Verfahren im $\mathbb{R}^n$ näherungsweise lösen. Nach Ortega und Rheinboldt [32] ist das lineare Gleichungssystem, das wir durch das Newton-Verfahren erhalten, identisch mit dem linearen Gleichungssystem, welches das zuerst beschriebene Vorgehen liefert. Wenn wir auch für das nichtlineare Gleichungssystem eine entsprechende Homotopiemethode formulieren, sind die linearen Gleichungssysteme im Prädiktorschritt identisch mit denen aus dem oben beschriebenen Vorgehen. Deshalb sind beide Vorgehensweisen gleichwertig. Wir haben uns für das zuerst beschriebene Vorgehen entschieden, da es für die Berechnung einer Näherungslösung einer linearen Integralgleichung ein effizientes Zwei-Gitter-Verfahren gibt, das ansonsten erst sinngemäß auf das lineare Gleichungssystem zu übertragen wäre.

Die Literatur, die sich mit der Bestimmung einer Näherungslösung eines nichtlinearen Gleichungssystems beschäftigt, ist umfangreich. Einen guten Überblick vermitteln die Bücher von Schwetlick [44] und Ortega und Rheinboldt [33]. Sie behandeln neben dem hier beschriebenen Newton-Verfahren auch modifizierte Newton-Verfahren, Abstiegsverfahren und vieles mehr.

Um das Newton-Verfahren wie im $\mathbb{R}^n$ formulieren zu können, müssen wir zuerst den Ableitungsbegriff von reellwertigen Funktionen auf Operatoren verallgemeinern. Definition 1.2.1 beschreibt die übliche Verallgemeinerung, die so genannte *Fréchetableitung*. Der folgende Hilfssatz gibt die Fréchetableitung bei dem von uns untersuchten nichtlinearen Operator konkret an.

Hilfssatz 2.3.1 (Fréchetableitung). *Seien $k, k_3 \in C([a,b] \times [a,b] \times \mathbb{R})$, $y \in C([a,b])$. Dann ist die Fréchetableitung des nichtlinearen Operators*

$$\mathcal{F} : \begin{cases} C([a,b]) \to C([a,b]), \\ x \mapsto x - \displaystyle\int_a^b k(.,t,x(t))\,dt - y \end{cases}$$

an der Stelle $x_0 \in C([a,b])$ gegeben durch den linearen Operator

$$\mathcal{F}'[x_0] : \begin{cases} C([a,b]) \to C([a,b]), \\ x \mapsto x - \displaystyle\int_a^b k_3(.,t,x_0(t))\,x(t)\,dt. \end{cases}$$

Beweis. Es ist zu zeigen, dass

$$\lim_{h\to 0} \frac{||\mathcal{F}(x_0+h) - \mathcal{F}x_0 - \mathcal{F}'[x_0]h||}{||h||} = 0$$

gilt. Wir betrachten dazu zuerst den Zähler dieses Ausdrucks:

$$\begin{aligned}
&||\mathcal{F}(x_0+h) - \mathcal{F}x_0 - \mathcal{F}'[x_0]h|| \\
&\quad = \max_{s\in[a,b]} \Big| x_0(s) + h(s) - \int_a^b k(s,t,x_0(t)+h(t))\,dt - y(s) \\
&\qquad\quad - x_0(s) + \int_a^b k(s,t,x_0(t))\,dt + y(s) - h(s) + \int_a^b k_3(s,t,x_0(t))\,h(t)\,dt \Big| \\
&\quad = \max_{s\in[a,b]} \Big| \int_a^b \big[-k(s,t,x_0(t)+h(t)) + k(s,t,x_0(t)) + k_3(s,t,x_0(t))\,h(t) \big]\,dt \Big| \\
&\quad = \max_{s\in[a,b]} \Big| \int_a^b \big[-\int_0^1 k_3(s,t,x_0(t)+\theta h(t))\,d\theta\,h(t) + k_3(s,t,x_0(t))\,h(t) \big]\,dt \Big| \\
&\quad = \max_{s\in[a,b]} \Big| \int_a^b \int_0^1 \big[k_3(s,t,x_0(t)) - k_3(s,t,x_0(t)+\theta h(t)) \big]\,d\theta\,h(t)\,dt \Big| \\
&\quad \le \max_{s\in[a,b]} \int_a^b \int_0^1 |k_3(s,t,x_0(t)) - k_3(s,t,x_0(t)+\theta h(t))|\,d\theta\,|h(t)|\,dt \\
&\quad \le ||h|| \max_{s\in[a,b]} \int_a^b \int_0^1 |k_3(s,t,x_0(t)) - k_3(s,t,x_0(t)+\theta h(t))|\,d\theta\,dt.
\end{aligned}$$

Auf Grund der gleichmäßigen Stetigkeit von k_3 in $[a,b] \times [a,b] \times [-||x_0|| - 1, ||x_0|| + 1]$ gibt es für jedes $\varepsilon > 0$ ein $\delta \in (0,1)$, so dass für alle $h \in C([a,b])$ mit $||h|| \le \delta$

$$|k_3(s,t,x_0(t)) - k_3(s,t,x_0(t)+\theta h(t))| \le \frac{\varepsilon}{b-a}$$

für alle $s, t \in [a, b]$ und alle $\theta \in [0, 1]$ gilt. Wir erhalten somit für beliebiges $\varepsilon > 0$

$$\begin{aligned}
&\frac{||\mathcal{F}(x_0 + h) - \mathcal{F}x_0 - \mathcal{F}'[x_0]h||}{||h||} \\
&\quad \le \frac{||h||}{||h||} \max_{s\in[a,b]} \int_a^b \int_0^1 |k_3(s,t,x_0(t)) - k_3(s,t,x_0(t) + \theta h(t))|\, d\theta\, dt \\
&\quad \le \varepsilon \qquad \text{falls } ||h|| \text{ hinreichend klein ist.}
\end{aligned}$$

□

2.3.1 Das Newton-Verfahren

In diesem Abschnitt wenden wir das in Abschnitt 1.2 abstrakt vorgestellte Newton-Verfahren konkret auf die näherungsweise Lösung einer nichtlinearen Integralgleichung an.

Eine Lösung der Fredholmschen Integralgleichung zweiter Art vom Urysohntyp (2.8) zu finden, ist gleichbedeutend mit der Bestimmung einer Nullstelle x^* der Operatorgleichung

$$\mathcal{F}x = x - \int_a^b k(.,t,x(t))\, dt - y = 0.$$

Die Newton-Iteration für diese Operatorgleichung ist ausgehend von einer Startnäherung $x_0 \in C([a, b])$ gegeben durch

$$\left\{ \begin{aligned} \mathcal{F}'[x_k]\triangle x_k &= -\mathcal{F}x_k, \\ x_{k+1} &= x_k + \triangle x_k, \end{aligned} \right. \qquad k = 0, 1, 2, \ldots. \tag{2.9}$$

Im Fall der nichtlinearen Fredholmschen Integralgleichung zweiter Art (2.8) lautet (2.9)

$$\left\{ \begin{aligned} \triangle x_k(s) - \int_a^b \underbrace{k_3(s,t,x_k(t))}_{=:\tilde{k}(s,t)} \triangle x_k(t)\, dt &= \underbrace{-x_k(s) + \int_a^b k(s,t,x_k(t))\, dt + y(s)}_{=:\tilde{r}(s)}, \\ x_{k+1}(s) &= x_k(s) + \triangle x_k(s) \end{aligned} \right.$$

für $k = 0, 1, 2, \ldots$. In jedem Newton-Schritt ist somit eine lineare Fredholmsche Integralgleichung zweiter Art zu lösen. Das Vorgehen hierfür beschreiben wir in Abschnitt 2.3.3. Zunächst gehen wir aber im Abschnitt 2.3.2 darauf ein, wie eine geeignete Startnäherung x_0 für das Newton-Verfahren gewonnen werden kann.

2.3.2 Eine Homotopiemethode für das Newton-Verfahren

Um Konvergenz des Newton-Verfahrens zu erreichen, benötigen wir eine gute Startnäherung x_0, die wir mit einem einfachen Homotopieverfahren gewinnen.

Wir betrachten beim Homotopieverfahren anstelle des Operators $\mathcal{F}$ und der Gleichung $\mathcal{F}x = 0$ eine ganze Schar von Operatoren $\mathcal{F}_\lambda$ und Gleichungen $\mathcal{F}_\lambda x = 0$ mit $\lambda \in [0, 1]$. Diese Schar von Operatoren ist mit dem ursprünglich zu untersuchenden Operator durch die Forderung $\mathcal{F}_1 = \mathcal{F}$ verbunden. Des Weiteren sei die Schar so gewählt, dass die Lösung von $\mathcal{F}_0 x = 0$ sofort ersichtlich ist. Wir suchen nun für jedes $\lambda \in [0, 1]$ nach der Lösung der Operatorgleichung $\mathcal{F}_\lambda x = 0$. Daher ist die Lösung $x^*(s)$ somit zusätzlich abhängig vom Homotopieparameter λ, also $x^* = x^*(\lambda, s)$. Die Gleichung $\mathcal{F}_\lambda x = 0$ lösen wir nun für verschiedene Werte von λ. Wir beginnen mit der exakten Lösung für $\lambda = 0$ und lösen dann $\mathcal{F}_\lambda x = 0$ für $\lambda > 0$ näherungsweise mit dem Newton-Verfahren. Aus der berechneten Näherungslösung der Gleichung $\mathcal{F}_\lambda x = 0$ erhalten wir durch Taylorentwicklung jeweils eine Startnäherung für das nächstgrößere λ.

Konkret wählen wir die Operatorenschar $\mathcal{F}_\lambda$ als Linearkombination

$$\mathcal{F}_\lambda := (1 - \lambda)\hat{\mathcal{F}} + \lambda \mathcal{F}, \qquad \lambda \in [0, 1]$$

eines einfachen Operators $\hat{\mathcal{F}}$ und des Operators $\mathcal{F}$. Später in diesem Abschnitt werden wir die spezielle Wahl $\hat{\mathcal{F}}x := x - y$, also

$$\mathcal{F}_\lambda x := x - \lambda \int_a^b k(., t, x(t))\, dt - y,$$

genauer betrachten.

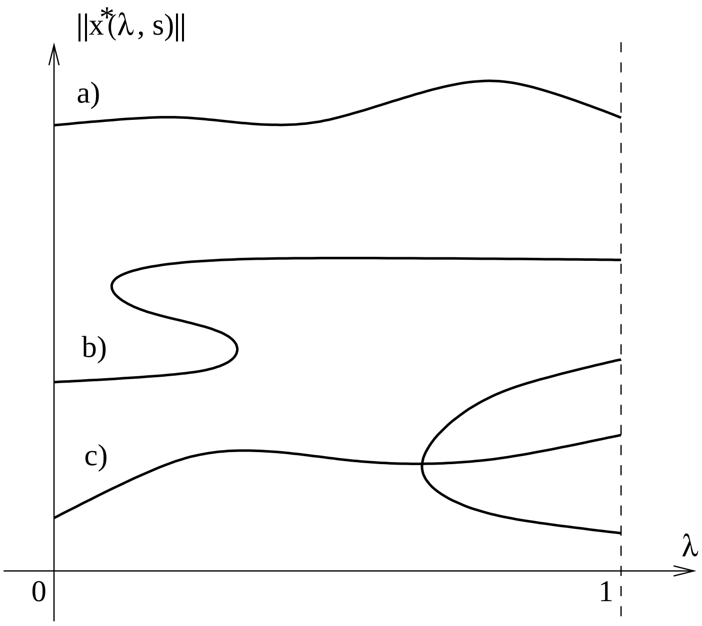


Abb. 2.1: Drei typische Verläufe des Lösungspfades: a) einfacher Pfad, b) Pfad mit Umkehrpunkten, c) Pfad mit Verzweigungspunkt.

In Abbildung 2.1 zeigen wir drei typische Verläufe des „Lösungspfades“ $x^*(\lambda, s)$. Das Homotopieverfahren, das wir verwenden werden, kann nur dem einfachen Pfad a) folgen.

Für die numerischen Beispiele in Kapitel 3 genügt dies. Auf die Behandlung von Umkehrpunkten (Pfad b)) und Verzweigungspunkten (Pfad c)) verzichten wir hier. Das Buch von Allgower und Georg [3] enthält ausführliche Anleitungen zur Behandlung dieser beiden Sonderfälle.

Wollen wir das Verhalten der Lösung $x^*(\lambda, s)$ entlang des Pfades in λ-Richtung für ein $s \in [a, b]$ näherungsweise beschreiben, so können wir dazu den Anfang der Taylorreihe

$$x^*(\lambda + \triangle\lambda, s) = x^*(\lambda, s) + \triangle\lambda \frac{\partial x^*}{\partial \lambda}(\lambda, s) + \frac{(\triangle\lambda)^2}{2} \frac{\partial^2 x^*}{\partial \lambda^2}(\lambda, s) + \cdots \tag{2.10}$$

verwenden.

Das Homotopieverfahren berechnet eine Näherungslösung der Gleichung $\mathcal{F}_1 x = \mathcal{F} x = 0$ unter Verwendung schon bekannter Näherungslösungen der Gleichungen $\mathcal{F}_{\lambda_k} x = 0$ mit $0 = \lambda_0 < \lambda_1 < \lambda_2 < \ldots < \lambda_{\nu-1} < 1$. In einem Homotopieschritt wird dabei ausgehend von einer zuvor berechneten Näherungslösung $\tilde{x}^*(\lambda_k, s)$ für $\mathcal{F}_{\lambda_k} x = 0$ eine Startnäherung für die Lösung $x^*(\lambda_k + \triangle\lambda_k, s)$ der Operatorgleichung $\mathcal{F}_{\lambda_k + \triangle\lambda_k} x = 0$ mit Hilfe des Anfangs der Taylorreihe berechnet und diese Startnäherung dann mit dem Newton-Verfahren verbessert.

Den Homotopieschritt werten wir als erfolgreich, wenn das Newton-Verfahren mit der zuvor berechneten Startnäherung an der Stelle $\lambda_k + \triangle\lambda_k$ konvergiert (vgl. hierzu auch Bemerkung 2.3.4). In diesem Fall setzen wir $\lambda_{k+1} := \lambda_k + \triangle\lambda_k$ und führen von λ_{k+1} ausgehend erneut einen Homotopieschritt durch. Dabei können wir gegebenenfalls die Homotopieschrittweite auch noch erhöhen, etwa $\triangle\lambda_{k+1} := 1.3 \times \triangle\lambda_k$.

Konvergiert das Newton-Verfahren an der Stelle $\lambda_k + \triangle\lambda_k$ hingegen nicht, so verringern wir die Schrittweite des Homotopieschrittes, etwa $\triangle\lambda_k := 0.5 \times \triangle\lambda_k$. Mit dieser neuen Schrittweite versuchen wir dann wieder mit Hilfe des Newton-Verfahrens eine Näherungslösung für $\mathcal{F}_{\lambda_k + \triangle\lambda_k} x = 0$ zu berechnen. Dieses Verringern der Schrittweite $\triangle\lambda_k$ wiederholen wir solange bis das Newton-Verfahren konvergiert oder die Homotopieschrittweite so klein wird, dass der Rechenaufwand zu groß wird und wir deswegen das Verfahren abbrechen müssen.

Wir starten das Homotopieverfahren mit den Parametern $\lambda_0 = 0$ und $\triangle\lambda_0 = 1$, so dass im günstigsten Fall nur ein einziger Homotopieschritt benötigt wird.

Das Homotopieverfahren endet erfolgreich, wenn für ein $\nu \in \mathbb{N}$ der Homotopieschritt von $\lambda_{\nu-1} < 1$ nach $\lambda_\nu = 1$ erfolgreich durchgeführt werden kann.

Bemerkung 2.3.2. *Das zuvor beschriebene Vorgehen kann auch als zweistufiger Prozess gesehen werden (vergleiche Abbildung 2.2). Die Taylorentwicklung* (2.10) *dient als Prädiktorschritt, der eine Näherung voraussagt, die dann mit dem Newton-Verfahren als Korrektorschritt verbessert wird. Neben der hier beschriebenen Vorhersage der Näherungslösung mit Hilfe der Taylorentwicklung werden in der Literatur beispielsweise auch Verfahren beschrieben, die auf der numerischen Lösung von Differentialgleichungen basieren. Um die Näherungslösung zu verbessern sind neben dem Newton-Verfahren auch modifizierte Newton-Verfahren oder Abstiegsverfahren möglich.*

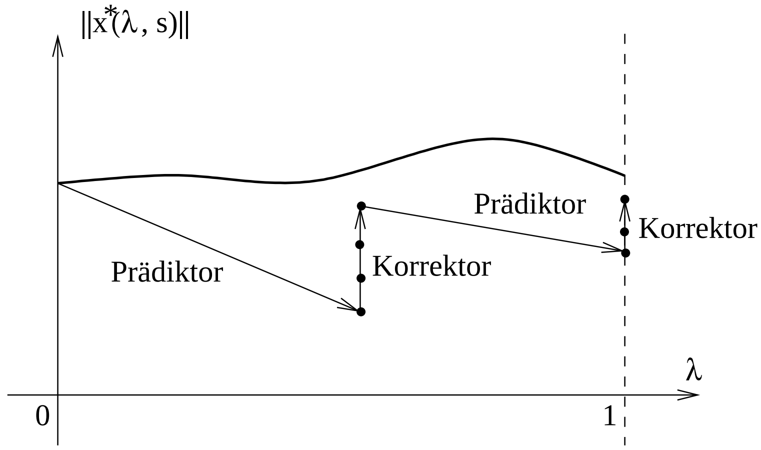


Abb. 2.2: Prädiktor- und Korrektorschritte des Homotopieverfahrens.

Im Folgenden wollen wir nur noch die spezielle Wahl des Homotopieoperators

$$\mathcal{F}_\lambda x := x - \lambda \int_a^b k(.,t,x(t))\,dt - y$$

betrachten und beschränken uns im Prädiktorschritt auf die ersten drei Glieder der Taylorreihe.

Bisher haben wir uns noch keine Gedanken darüber gemacht, ob die Ableitungen $\frac{\partial x^*}{\partial \lambda}(\lambda, s)$ und $\frac{\partial^2 x^*}{\partial \lambda^2}(\lambda, s)$, die wir zur Bestimmung einer Startnäherung mittels des Anfangs der Taylorreihe benötigen, existieren und wie wir diese bestimmen können. Ableiten der Gleichung $\mathcal{F}_\lambda x = 0$ nach dem Homotopieparameter λ liefert uns Gleichungen, die, wenn sie lösbar sind, die unbekannten Ableitungen festlegen:

$$\frac{\partial x^*}{\partial \lambda}(\lambda, s) - \int_a^b \Big(k(s,t,x^*(\lambda,t)) + \lambda k_3(s,t,x^*(\lambda,t)) \frac{\partial x^*}{\partial \lambda}(\lambda,t) \Big)\,dt = 0, \quad \text{(2.11a)}$$

$$\begin{aligned} \frac{\partial^2 x^*}{\partial \lambda^2}(\lambda, s) - \int_a^b \Big(& 2k_3(s,t,x^*(\lambda,t)) \frac{\partial x^*}{\partial \lambda}(\lambda,t) \\ & + \lambda \Big[k_{3,3}(s,t,x^*(\lambda,t)) \left(\frac{\partial x^*}{\partial \lambda}(\lambda,t) \right)^2 + k_3(s,t,x^*(\lambda,t)) \frac{\partial^2 x^*}{\partial \lambda^2}(\lambda,t) \Big] \Big)\,dt = 0. \quad \text{(2.11b)} \end{aligned}$$

Betrachten wir diese Gleichungen nun an einer festen Stelle $\lambda = \lambda_k$ und wählen wir zur Abkürzung die Bezeichnungen

$$\begin{aligned} x_I(s) &:= \frac{\partial x^*}{\partial \lambda}(\lambda_k, s), \\ k_I(s,t) &:= \lambda_k k_3(s,t,x^*(\lambda_k,t)), \\ y_I(s) &:= \int_a^b k(s,t,x^*(\lambda_k,t))\,dt, \end{aligned}$$

$$\begin{aligned}
x_{II}(s) &:= \frac{\partial^2 x^*}{\partial \lambda^2}(\lambda_k, s),\\
k_{II}(s,t) &:= \lambda_k k_3(s,t,x^*(\lambda_k,t)),\\
y_{II}(s) &:= \int_a^b \big(2k_3(s,t,x^*(\lambda_k,t))x'(t) + \lambda_k k_{3,3}(s,t,x^*(\lambda_k,t))x'^2(t)\big)\,dt,
\end{aligned}$$

so sehen wir, dass es sich bei den Gleichungen (2.11) um zwei lineare Fredholmsche Integralgleichungen zweiter Art

$$\begin{aligned}
x_I(s) - \int_a^b k_I(s,t)x_I(t)\,dt &= y_I(s),\\
x_{II}(s) - \int_a^b k_{II}(s,t)x_{II}(t)\,dt &= y_{II}(s),
\end{aligned}$$

handelt. Da $k_I = k_{II}$ gilt, liegt tatsächlich sogar nur eine Integralgleichung mit den beiden unterschiedliche rechte Seiten y_I und y_{II} vor. Diese können wir wie die im Newton-Schritt auftretenden linearen Integralgleichungen mit den in Abschnitt 2.3.3 beschriebenen Verfahren numerisch lösen.

Die Gleichungen (2.11) vereinfachen sich für den ersten Homotopieschritt mit $\lambda_0 = 0$ zu

$$\begin{aligned}
x^*(0,s) &= y(s),\\
\frac{\partial x^*}{\partial \lambda}(0,s) &= \int_a^b k(s,t,y(t))\,dt,\\
\frac{\partial^2 x^*}{\partial \lambda^2}(0,s) &= \int_a^b 2k_3(s,t,y(t))\frac{\partial x^*}{\partial \lambda}(0,t)\,dt.
\end{aligned}$$

In Algorithmus 2.3.3 beschreiben wir zur Zusammenfassung das einfache Homotopieverfahren noch einmal in Codeform.

Algorithmus 2.3.3 (Homotopiemethode).

$k = 0, \quad step = 0, \quad \lambda_0 = 0, \quad \triangle\lambda_0 = 1$

`Repeat` *(Prädiktorschleife)*

  $step = step + 1$

  $\lambda_{k+1} = \lambda_k + \triangle\lambda_k$

  $x_0(\lambda_{k+1}, s) = x(\lambda_k, s) + \triangle\lambda_k \frac{\partial x}{\partial \lambda}(\lambda_k, s) + \frac{(\triangle\lambda_k)^2}{2}\frac{\partial^2 x}{\partial \lambda^2}(\lambda_k, s)$

  `For` $(m = 0, \ldots, m_{\max})$ *(Korrektorschleife)*

    $x_{m+1}(\lambda_{k+1}, s) = x_m(\lambda_{k+1}, s) - (\mathcal{F}'[x_m(\lambda_{k+1}, s)])^{-1}\mathcal{F}(x_m(\lambda_{k+1}, s))$

    `If` (Konvergenz) `Then` $x(\lambda_{k+1}, s) = x_{m+1}(\lambda_{k+1}, s)$, `Break`

  (Schrittweitenanpassung)

  `If` (Konvergenz) `Then` $k = k + 1, \quad \triangle\lambda_k = \min(1.3 \times \triangle\lambda_{k-1}, 1 - \lambda_k)$

  `Else` $\triangle\lambda_k = 0.5 \times \triangle\lambda_k$

`Until` ((Konvergenz `And` $\lambda_k = 1$) `Or` $step \geq max\_step$)

Bemerkung 2.3.4.

- *Die Schrittweitenanpassung bei Konvergenz gemäß* $\triangle\lambda_k = \min(1.3 \times \triangle\lambda_{k-1}, 1 - \lambda_k)$ *sorgt für eine Schrittweitenvergrößerung um den Faktor 1.3, wenn* $\lambda_{k+1} = \lambda_k + \triangle\lambda_k < 1$ *ist. Ansonsten wird die Schrittweite auf den maximal möglichen Betrag* $1 - \lambda_k$ *gesetzt.*

- *Da wir im Algorithmus 2.3.3 nicht beliebig viele Newton-Schritte ausführen können und nach endlich vielen Schritten abbrechen, bezeichnen wir das Newton-Verfahren als konvergent, wenn innerhalb dieser vorgegebenen maximalen Schrittzahl der Defekt der Integralgleichung kleiner einer vorgegebenen kleinen Schranke wird.*

In der Literatur werden Homotopiemethoden oft auch als Fortsetzungsmethoden oder Einbettungsmethoden bezeichnet. Das Buch von Allgower und Georg [3] stellt ausgefeilte Methoden zur Schrittweitensteuerung und Pfadverfolgung bei Homotopiemethoden vor, die weit über das hier Dargestellte hinausgehen. Im Buch von Nocedal und Wright [31] wird speziell auf die richtige Wahl des Richtungsvektors entlang des Homotopiepfades eingegangen. Auch die schon zuvor angegebenen Bücher von Schwetlick [44] und Ortega und Rheinboldt [33] beschäftigen sich mit diesem Thema.

2.3.3 Approximation der Lösung einer linearen Fredholmschen Integralgleichung zweiter Art

Wie wir in Abschnitt 2.3.1 beschrieben haben, liefert jeder Newton-Schritt eine lineare Fredholmsche Integralgleichung zweiter Art

$$\triangle x(s) - \int_a^b \tilde{k}(s,t)\triangle x(t)\,dt = \tilde{r}(s), \qquad s \in [a,b], \tag{2.12}$$

bzw. in Operatorschreibweise

$$(\mathcal{I} - \tilde{\mathcal{K}})\triangle x = \tilde{r} \qquad \text{mit} \qquad \tilde{\mathcal{K}}\triangle x := \int_a^b \tilde{k}(.,t)\triangle x(t)\,dt. \tag{2.13}$$

Auch im Prädiktorschritt des einfachen Homotopieverfahrens (siehe Abschnitt 2.3.2) erhalten wir lineare Fredholmsche Integralgleichungen zweiter Art.

Da wir eine lineare Fredholmsche Integralgleichung zweiter Art im Allgemeinen nicht exakt lösen können, stellen wir im folgenden Abschnitt zwei numerische Methoden vor, welche eine solche Integralgleichung näherungsweise lösen. Die erste Methode, das Nyström-Verfahren, diskretisiert die Integralgleichung durch eine Quadraturformel und löst das dabei entstehende vollbesetzte lineare Gleichungssystem. Die zweite Methode, das Zwei-Gitter-Verfahren, ist ein iteratives Verfahren, das die diskrete Gleichung nur näherungsweise löst, dafür aber auch einen geringeren Rechenaufwand benötigt.

Das Nyström-Verfahren

Zur Approximation des Integrals in Gleichung (2.12) verwenden wir eine Quadraturformel $Q_n(f)$,

$$\int_a^b f(t)\,dt \approx Q_n(f) := \sum_{j=1}^{n} w_{n,j} f(t_{n,j}),$$

eines punktweise konvergenten Quadraturverfahrens (Q_n) mit gegebenen Quadraturgewichten $w_{n,j}$, $j = 1, \ldots, n$, und Quadraturknoten $t_{n,j}$, $j = 1, \ldots, n$ (siehe Abschnitt 1.3). Anstelle der gesuchten Lösung $\triangle x$ von (2.12) berechnen wir nun eine Näherungsfunktion $\triangle x_n(s)$, indem wir die durch Anwendung der Quadraturformel für $s \in [a, b]$ resultierende semidiskrete Gleichung

$$\triangle x_n(s) - \sum_{j=1}^{n} w_{n,j} \tilde{k}(s, t_{n,j}) \triangle x_n(t_{n,j}) = \tilde{r}(s), \qquad s \in [a, b] \tag{2.14}$$

nach $\triangle x_n(s)$ auflösen.

Mit dem *Nyström-Operator*

$$\tilde{\mathcal{K}}_n x := \sum_{j=1}^{n} w_{n,j} \tilde{k}(., t_{n,j}) x(t_{n,j})$$

ergibt sich für (2.14) in Operatorschreibweise die so genannte *Nyström-Gleichung*

$$(\mathcal{I} - \tilde{\mathcal{K}}_n) \triangle x_n = \tilde{r}. \tag{2.15}$$

Das *Nyström-Verfahren* (siehe Hackbusch [16, Abschnitt 4.7]) besteht nun darin, diese Nyström-Gleichung zu lösen.

Kennen wir die Unbekannten $\triangle x_n(t_{n,j})$ aus Gleichung (2.14), so ist

$$\triangle x_n(s) := \tilde{r}(s) + \sum_{j=1}^{n} w_{n,j} \tilde{k}(s, t_{n,j}) \triangle x_n(t_{n,j}) \tag{2.16}$$

Lösung der Nyström-Gleichung (2.15).

Die Unbekannten $\triangle x_n(t_{n,j})$ können wir bestimmen, indem wir Gleichung (2.14) zuerst nur für $s := t_{n,i}$, $i = 1, \ldots, n$, betrachten. Wir erhalten

$$\triangle x_n(t_{n,i}) - \sum_{j=1}^{n} w_{n,j} \tilde{k}(t_{n,i}, t_{n,j}) \triangle x_n(t_{n,j}) = \tilde{r}(t_{n,i}), \qquad i = 1, \ldots, n.$$

Diese n Gleichungen ergeben ein lineares Gleichungssystem

$$(\boldsymbol{I}_n - \tilde{\boldsymbol{K}}_n) \triangle \boldsymbol{x}_n = \tilde{\boldsymbol{r}}_n \tag{2.17}$$

mit der (n,n)-Einheitsmatrix $\boldsymbol{I}_n$, der Matrix

$$\tilde{\boldsymbol{K}}_n = (k^n_{i,j}), \qquad k^n_{i,j} := w_{n,j}\tilde{k}(t_{n,i}, t_{n,j}),$$

der rechten Seite

$$\tilde{\boldsymbol{r}}_n = (\tilde{r}_{n,i}), \qquad \tilde{r}_{n,i} := \tilde{r}(t_{n,i})$$

und dem unbekannten Vektor

$$\triangle\boldsymbol{x}_n = (\triangle x_{n,i}), \qquad \triangle x_{n,i} := \triangle x_n(t_{n,i}).$$

Dieses lineare Gleichungssystem ist nach Satz 2.3.5 für hinreichend großes n eindeutig lösbar. Die Komponenten von $\triangle\boldsymbol{x}_n$ definieren nach Gleichung (2.16) eine Lösung der Nyström-Gleichung (2.15).

Auf die Konvergenztheorie des Verfahrens gehen wir hier nicht näher ein, siehe z. B. Hackbusch [16, Abschnitt 4.7.2, 4.7.3]. Wir halten nur das wesentliche Ergebnis fest.

Satz 2.3.5 (Konvergenz des Nyström-Verfahrens). *Es seien $\tilde{k} \in C([a,b] \times [a,b])$ und $\tilde{r} \in C([a,b])$. Das Quadraturverfahren (Q_n) sei punktweise konvergent. Ist die lineare Fredholmsche Integralgleichung zweiter Art* (2.12) *eindeutig nach $\triangle x$ auflösbar, so gibt es ein $n_0 \in \mathbb{N}$, so dass für alle $n \geq n_0$ das lineare Gleichungssystem* (2.17) *eindeutig auflösbar ist. Für die dann eindeutig bestimmte Funktion $\triangle x_n$ aus* (2.16) *gilt*

$$||\triangle x - \triangle x_n|| \overset{n\to\infty}{\to} 0.$$

Regularisieren einer linearen Fredholmschen Integralgleichung zweiter Art

In diesem Abschnitt beschreiben wir, wie die Konvergenz der Nyström-Lösung (2.16) gegen die Lösung der linearen Fredholmschen Integralgleichung zweiter Art (2.12) durch Regularisieren der Integralgleichung beschleunigt werden kann. Dazu untersuchen wir zuerst, von welchen Einflüssen die Konvergenzgeschwindigkeit des Nyström-Verfahrens abhängt.

Das verwendete Quadraturverfahren (Q_n) sei *konsistent* mit der Konsistenzordnung $\chi > 0$, d. h. es gebe eine Konstante C_Q, so dass

$$|Q_n(f) - \int_a^b f(t)\,dt| \leq C_Q \left(\frac{b-a}{n}\right)^{\chi} ||f||_{C^{\chi}([a,b])}$$

für alle $f \in C^{\chi}([a,b])$ gilt, und es sei *stabil*, d. h. es gelte

$$\sup_{n\in\mathbb{N}} \sum_{j=1}^{n} |w_{n,j}| < \infty.$$

Satz 2.3.6 (Konvergenzgeschwindigkeit des Nyström-Verfahrens). *Sei $\tilde{k} \in C^{\chi}([a,b] \times [a,b])$ und $\tilde{r} \in C^{\chi}([a,b])$. Weiterhin existiere $(\mathcal{I} - \tilde{\mathcal{K}})^{-1}$ und sei beschränkt. Dann gibt es eine Konstante C, so dass die Nyström-Lösung $\triangle x_n$ für hinreichend großes n der Fehlerabschätzung*

$$||\triangle x - \triangle x_n|| \leq C \left(\frac{b-a}{n}\right)^{\chi} ||\tilde{r}||_{C^{\chi}([a,b])}$$

genügt.

Beweis. Siehe Hackbusch [16, Satz 4.7.16]. □

Bemerkung 2.3.7.

- *Die Formeln für den Quadraturfehler der verschiedenen Quadraturverfahren in Abschnitt 1.3 zeigen, dass für alle dort vorgestellten Verfahren die Konsistenzordnung mindestens so groß ist wie die Ordnung, mit der der Quadraturfehler gegen Null konvergiert.*
- *Alle Quadraturverfahren, die wir durch Summieren der Newton-Cotes-Formeln bis zur Ordnung 6 erhalten (siehe Abschnitt 1.3.1) und das Gaußsche Quadraturverfahren (siehe Abschnitt 1.3.2) haben positive Gewichte und sind deswegen stabil.*

Wir wollen nun die lineare Fredholmsche Integralgleichung zweiter Art (2.12) in eine dazu äquivalente lineare Fredholmsche Integralgleichung zweiter Art transformieren, für die die rechte Seite der transformierten Gleichung möglichst glatt ist und somit das Nyström-Verfahren nach Satz 2.3.6 mit hoher Ordnung konvergiert. Der Vorgang, die Integralgleichung in eine Integralgleichung mit glatter rechter Seite zu überführen, wird als *Regularisieren* bezeichnet.

Wir machen den Ansatz

$$\triangle x = \tilde{r} + \varphi$$

mit einer noch zu bestimmenden Funktion $\varphi : [a,b] \to \mathbb{R}$. Setzen wir diese Darstellung in die Gleichung (2.13)

$$(\mathcal{I} - \tilde{\mathcal{K}})\triangle x = \tilde{r}$$

ein, so erhalten wir

$$\tilde{r} + \varphi - \tilde{\mathcal{K}}(\tilde{r} + \varphi) = \tilde{r}, \qquad \text{also} \qquad \varphi - \tilde{\mathcal{K}}\varphi = \underbrace{\tilde{\mathcal{K}}\tilde{r}}_{=:\psi}.$$

Ist nun $\tilde{r} \in C([a,b])$ und fordern wir zusätzlich $\tilde{\mathcal{K}} \in L(C([a,b]), C^{\chi}([a,b]))$, dann ist $\psi = \tilde{\mathcal{K}}\tilde{r} \in C^{\chi}([a,b])$. Lösen wir nun anstelle der linearen Fredholmschen Integralgleichung zweiter Art

$$(\mathcal{I} - \tilde{\mathcal{K}})\triangle x = \tilde{r}$$

die lineare Fredholmsche Integralgleichung zweiter Art

$$(\mathcal{I} - \tilde{\mathcal{K}})\varphi = \psi$$

mit glatterer rechter Seite ψ, so konvergiert nach Satz 2.3.6 die Nyström-Näherung φ_n mit höherer Konvergenzordnung gegen φ. Danach setzen wir

$$\triangle x_n := \tilde{r} + \varphi_n.$$

Da

$$||\triangle x - \triangle x_n|| = ||\tilde{r} + \varphi - \tilde{r} - \varphi_n|| = ||\varphi - \varphi_n||$$

gilt, ergibt sich diese hohe Konvergenzordnung auch für die gesuchte Funktion $\triangle x_n$.

Der Aufwand des Nyström-Verfahrens erhöht sich durch die Regularisierung nur dadurch, dass wir zusätzlich die Komponenten des Vektors der rechten Seite

$$\psi_{n,i} = \psi(t_{n,i}) = \int_a^b \tilde{k}(t_{n,i}, t)\tilde{r}(t)\, dt$$

berechnen müssen. Die numerische Berechnung des Integrals ist aber schwierig, da wir ja gerade $\tilde{r}$ als nicht besonders glatt vorausgesetzt haben. Deshalb ist eine geschickte numerische Integration oder aber eine exakt Auswertung des Integrals notwendig.

Das Zwei-Gitter-Verfahren

Da in jedem Homotopieschritt ein Newton-Verfahren durchzuführen ist und in jedem Newton-Schritt eine lineare Integralgleichung zu lösen ist, stellt sich der Aufwand zur Lösung einer linearen Integralgleichung als zentral für den Aufwand zur Bestimmung einer Näherungslösung heraus.

Wie wir im vorletzten Abschnitt gesehen haben, besteht der Hauptaufwand bei der Lösung einer linearen Fredholmschen Integralgleichung zweiter Art mit Hilfe des Nyström-Verfahrens darin, ein lineares Gleichungssystem mit einer vollbesetzten Matrix $\boldsymbol{I}_n - \tilde{\boldsymbol{K}}_n$ zu lösen. Verwenden wir zum Lösen dieses Gleichungssystems ein direktes Verfahren, beispielsweise den Gauß-Algorithmus, so beträgt der Rechenaufwand $O(n^3)$.

In diesem Abschnitt stellen wir ein von Atkinson in [4, Abschnitt 6.2] beschriebenes iteratives Verfahren vor, das die Nyström-Gleichung (2.15)

$$(\mathcal{I} - \tilde{\mathcal{K}}_n)\triangle x_n = \tilde{r}$$

näherungsweise löst und dafür nur einen Aufwand von $O(n^2)$ benötigt.

Setzen wir anstelle der Lösung $\triangle x_n$ eine Näherungslösung $\triangle x_n^{(0)}$ in die Nyström-Gleichung (2.15) ein, so wird diese nicht mehr exakt gelöst, sondern weist einen Defekt von

$$d^{(0)} = \tilde{r} - (\mathcal{I} - \tilde{\mathcal{K}}_n)\triangle x_n^{(0)} \tag{2.18}$$

auf. Setzen wir $\tilde{r}$ aus Gleichung (2.15) in die Defektgleichung (2.18) ein, so erhalten wir

$$d^{(0)} = (\mathcal{I} - \tilde{\mathcal{K}}_n)(\triangle x_n - \triangle x_n^{(0)}), \tag{2.19}$$

also

$$\triangle x_n = \triangle x_n^{(0)} + (\mathcal{I} - \tilde{\mathcal{K}}_n)^{-1} d^{(0)}.$$

Wenn wir in der letzten Gleichung $(\mathcal{I} - \tilde{\mathcal{K}}_n)^{-1}$ nur näherungsweise berechnen, erhalten wir wieder nicht die exakte Lösung $\triangle x_n$ der Nyström-Gleichung, sondern eine weitere Näherung $x_n^{(1)}$. Für diese Näherungslösung können wir wieder, wie für $x_n^{(0)}$, einen Defekt und damit eine neue Näherungslösung berechnen. Durch Wiederholen dieses Vorgehens erhalten wir ein Iterationsverfahren.

Im Folgenden stellen wir zwei unterschiedliche Methoden vor, die anstelle der Nyström-Gleichung $(\mathcal{I} - \tilde{\mathcal{K}}_n)\delta_n^{(k)} = d^{(k)}$ eine approximierende Gleichung lösen.

Methode 1 für das Zwei-Gitter-Verfahren

Existiert für ein $m < n$ die Inverse $(\mathcal{I} - \tilde{\mathcal{K}}_m)^{-1}$, so können wir diese als Approximation für $(\mathcal{I} - \tilde{\mathcal{K}}_n)^{-1}$ verwenden und erhalten so das Iterationsverfahren

$$\left.\begin{aligned} d^{(k)} &= \tilde{r} - (\mathcal{I} - \tilde{\mathcal{K}}_n)\triangle x_n^{(k)}, \\ \triangle x_n^{(k+1)} &= \triangle x_n^{(k)} + (\mathcal{I} - \tilde{\mathcal{K}}_m)^{-1} d^{(k)}, \end{aligned}\right\} \qquad k = 0, 1, 2, \ldots \tag{2.20}$$

mit einer beliebigen Startnäherung $\triangle x_n^{(0)}$, beispielsweise $\triangle x_n^{(0)} = 0$.

Bemerkung 2.3.8. *Den im Iterationsverfahren auftretenden Ausdruck $(\mathcal{I} - \tilde{\mathcal{K}}_m)^{-1} d^{(k)}$ fassen wir als Synonym für die Lösung $\delta^{(k)}$ der linearen Gleichung*

$$(\mathcal{I} - \tilde{\mathcal{K}}_m)\delta^{(k)} = d^{(k)}$$

auf. Die Inverse wird nicht explizit benötigt.

Der folgende Satz macht Aussagen zur Konvergenz des Iterationsverfahrens (2.20).

Satz 2.3.9 (Konvergenz des Zwei-Gitter-Verfahren nach Methode 1). *Die lineare Fredholmsche Integralgleichung zweiter Art $(\mathcal{I} - \tilde{\mathcal{K}})\triangle x = \tilde{r}$ sei für alle $\tilde{r} \in C([a, b])$ eindeutig lösbar. Sei $\tilde{k} \in C([a, b] \times [a, b])$ und sei das zur Diskretisierung von $\tilde{\mathcal{K}}_n$ verwendete Quadraturverfahren punktweise konvergent (vgl. Abschnitt 1.3). Dann gibt es ein $m \in \mathbb{N}$, so dass das Iterationsverfahren* (2.20) *für beliebige Startnäherungen $\triangle x_n^{(0)}$ für alle $n > m$ konvergiert und*

$$||\triangle x_n - \triangle x_n^{(k)}|| \overset{k \to \infty}{\rightarrow} 0$$

gilt.

Gemäß Beweis dieses Satzes in Atkinson [4, Theorem 6.1] ist nur in jedem zweiten Iterationsschritt mit einer Verminderung des Fehlers $||\triangle x_n - \triangle x_n^{(k)}||$ zu rechnen. Tatsächlich gibt es lineare Fredholmsche Integralgleichungen zweiter Art, bei denen das Zwei-Gitter-Verfahren nach Methode 1 in einem Schritt eine Fehlerreduktion, im nächsten aber eine Fehlervergrößerung und im übernächsten wieder eine Fehlerverminderung zeigt. Dieses ungleichmäßige Konvergenzverhalten ist nicht wünschenswert, weswegen wir mit Methode 2 ein Zwei-Gitter-Verfahren vorstellen, das in jedem Schritt eine Fehlerverminderung liefert.

Methode 2 für das Zwei-Gitter-Verfahren

Wie in (2.19) gesehen erfüllt der Fehler $\triangle x_n - \triangle x_n^{(0)}$ die Gleichung

$$(\mathcal{I} - \tilde{\mathcal{K}}_n)(\triangle x_n - \triangle x_n^{(0)}) = d^{(0)}.$$

Wenden wir auf beide Seiten dieser Gleichung den „glättenden“ Operator $\tilde{\mathcal{K}}_n$ an (vergleiche dazu den Abschnitt über die Regularisierung einer linearen Fredholmschen Integralgleichung), so ergibt sich

$$\begin{aligned}
&\tilde{\mathcal{K}}_n(\mathcal{I} - \tilde{\mathcal{K}}_n)(\triangle x_n - \triangle x_n^{(0)}) = \tilde{\mathcal{K}}_n d^{(0)} \\
\iff \quad & (\mathcal{I} - \tilde{\mathcal{K}}_n)\underbrace{\tilde{\mathcal{K}}_n(\triangle x_n - \triangle x_n^{(0)})}_{=:\delta^{(0)}} = \tilde{\mathcal{K}}_n d^{(0)} \\
\iff \quad & \delta^{(0)} = (\mathcal{I} - \tilde{\mathcal{K}}_n)^{-1}\tilde{\mathcal{K}}_n d^{(0)}.
\end{aligned} \tag{2.21}$$

Die gesuchte Lösung $\triangle x_n$ der Nyström-Gleichung können wir dann mit Hilfe der Funktionen $\triangle x_n^{(0)}$, $d^{(0)}$ und $\delta^{(0)}$ schreiben als

$$\begin{aligned}
\triangle x_n &\overset{(2.15)}{=} \tilde{r} + \tilde{\mathcal{K}}_n\triangle x_n = \tilde{r} + \tilde{\mathcal{K}}_n\triangle x_n + \tilde{\mathcal{K}}_n\triangle x_n^{(0)} - \tilde{\mathcal{K}}_n\triangle x_n^{(0)} \\
&= \tilde{r} + \tilde{\mathcal{K}}_n(\triangle x_n - \triangle x_n^{(0)}) + \tilde{\mathcal{K}}_n\triangle x_n^{(0)} = \tilde{r} + \delta^{(0)} + \tilde{\mathcal{K}}_n\triangle x_n^{(0)} \\
&\overset{(2.18)}{=} \tilde{r} + \delta^{(0)} + d^{(0)} - \tilde{r} + \triangle x_n^{(0)} = \triangle x_n^{(0)} + \delta^{(0)} + d^{(0)}.
\end{aligned}$$

Auch diesmal bestimmen wir, wie schon in Methode 1, den Operator $(\mathcal{I} - \tilde{\mathcal{K}}_n)^{-1}$ in Gleichung (2.21) nicht exakt, sondern approximieren ihn durch einen Operator $(\mathcal{I} - \tilde{\mathcal{K}}_m)^{-1}$ mit $m < n$. Die Approximation wird uns diesmal besser gelingen, da wir sie nur auf der Menge der geglätteten Funktionen $\tilde{\mathcal{K}}_n d^{(0)}$ benötigen. Da wir nur eine Näherung von $(\mathcal{I} - \tilde{\mathcal{K}}_n)^{-1}$ verwenden, erhalten wir nicht die exakte Lösung $\triangle x_n$ der Nyström-Gleichung, sondern nur eine weitere Näherungslösung $\triangle x_n^{(1)}$, für die wir analog eine weitere Näherungslösung bestimmen können. Durch Wiederholen dieses Vorgehens erhalten wir das Iterationsverfahren

$$\left.\begin{aligned}
d^{(k)} &= \tilde{r} - (\mathcal{I} - \tilde{\mathcal{K}}_n)\triangle x_n^{(k)}, \\
\delta^{(k)} &= (\mathcal{I} - \tilde{\mathcal{K}}_m)^{-1}\tilde{\mathcal{K}}_n d^{(k)}, \\
\triangle x_n^{(k+1)} &= \triangle x_n^{(k)} + \delta^{(k)} + d^{(k)},
\end{aligned}\right\} \qquad k = 0, 1, 2, \ldots \tag{2.22}$$

mit einer beliebigen Startnäherung $\triangle x_n^{(0)}$, beispielsweise $\triangle x_n^{(0)} = 0$.

Im nächsten Satz untersuchen wir das Konvergenzverhalten dieser Iteration.

Satz 2.3.10 (Konvergenz des Zwei-Gitter-Verfahrens nach Methode 2). *Die lineare Fredholmsche Integralgleichung zweiter Art $(\mathcal{I}-\tilde{\mathcal{K}})\triangle x = \tilde{r}$ sei für alle $\tilde{r} \in C([a,b])$ eindeutig lösbar. Sei $\tilde{k} \in C([a,b]\times[a,b])$ und sei das zur Diskretisierung von $\tilde{\mathcal{K}}$ verwendete Quadraturverfahren punktweise konvergent (vgl. Abschnitt 1.3). Dann gibt es ein $m \in \mathbb{N}$, so dass das Iterationsverfahren (2.22) für beliebige Startnäherungen $\triangle x_n^{(0)}$ für alle $n > m$ konvergiert und*

$$||\triangle x_n - \triangle x_n^{(k)}|| \overset{k\to\infty}{\to} 0$$

gilt.

Der Beweis dieses Satzes in Atkinson [4, Theorem 6.2] zeigt, dass für genügend großes m die Beziehung

$$||\triangle x_n - \triangle x_n^{(k+1)}|| \leq \tau_m ||\triangle x_n - \triangle x_n^{(k)}|| \qquad \text{mit} \qquad \tau_m < 1$$

für die Differenz zwischen gesuchter Nyström-Lösung $\triangle x_n$ und iterierter Lösung $\triangle x_n^{(k+1)}$ gilt. Im Gegensatz zum Zwei-Gitter-Verfahren nach Methode 1 findet also in jedem Schritt eine Fehlerreduktion statt. Diese Beziehung können wir auch nutzen, um ein Abbruchkriterium für das Iterationsverfahren zu bestimmen. Es gilt

$$\begin{aligned} ||\triangle x_n - \triangle x_n^{(k+1)}|| &\leq \tau_m ||\triangle x_n - \triangle x_n^{(k)}|| = \tau_m ||\triangle x_n - \triangle x_n^{(k+1)} + \triangle x_n^{(k+1)} - \triangle x_n^{(k)}|| \\ &\leq \tau_m (||\triangle x_n - \triangle x_n^{(k+1)}|| + ||\triangle x_n^{(k+1)} - \triangle x_n^{(k)}||) \end{aligned}$$

$$\Longrightarrow \qquad ||\triangle x_n - \triangle x_n^{(k+1)}|| \leq \frac{\tau_m}{1-\tau_m} ||x_n^{(k+1)} - x_n^{(k)}||.$$

Die Differenz zwischen Nyström-Lösung $\triangle x_n$ und iterierter Lösung $\triangle x_n^{(k+1)}$ lässt sich also durch die Differenz von zwei aufeinander folgenden Iterierten abschätzen. Die hierbei auftretende Konstante τ_m können wir durch

$$\tau_m \approx \frac{||\triangle x_n - \triangle x_n^{(k+1)}||}{||\triangle x_n - \triangle x_n^{(k)}||} \approx \frac{\frac{\tau_m}{1-\tau_m} ||x_n^{(k+1)} - x_n^{(k)}||}{\frac{\tau_m}{1-\tau_m} ||x_n^{(k)} - x_n^{(k-1)}||} = \frac{||\triangle x_n^{(k+1)} - \triangle x_n^{(k)}||}{||\triangle x_n^{(k)} - \triangle x_n^{(k-1)}||} =: \nu_k$$

approximieren. Wir beenden somit das Iterationsverfahren, wenn

$$\frac{\nu_k}{1-\nu_k} ||\triangle x_n^{(k+1)} - \triangle x_n^{(k)}|| < \varepsilon$$

mit einem vorgegebenen kleinen ε gilt.

Definieren wir in Analogie zum Nyström-Verfahren Vektoren, die die Funktionswerte von $\triangle x_n^{(k)}$, $d^{(k)}$, $\delta^{(k)}$ und $\tilde{r}$ an den Quadraturknoten $t_{n,i}$, $i = 1, \ldots, n$, enthalten, gemäß

$$\begin{aligned} \triangle \boldsymbol{x}_n^{(k)} &:= (\triangle x_n^{(k)}(t_{n,i}))_{i=1,\ldots,n}, & \boldsymbol{d}^{(k)} &:= (d^{(k)}(t_{n,i}))_{i=1,\ldots,n}, \\ \boldsymbol{\delta}^{(k)} &:= (\delta^{(k)}(t_{n,i}))_{i=1,\ldots,n}, & \tilde{\boldsymbol{r}} &:= (\tilde{r}(t_{n,i}))_{i=1,\ldots n} \end{aligned}$$

sowie Matrizen

$$
\begin{aligned}
\boldsymbol{A}_m &= (a^m_{i,j})_{\substack{i=1,\dots,n\\ j=1,\dots,m}}, & a^m_{i,j} &:= w_{m,j}\tilde{k}(t_{n,i}, t_{m,j}),\\
\boldsymbol{A}'_m &= (a'^m_{i,j})_{\substack{i=1,\dots,m\\ j=1,\dots,m}}, & a'^m_{i,j} &:= w_{m,j}\tilde{k}(t_{m,i}, t_{m,j}),\\
\boldsymbol{A}_n &= (a^n_{i,j})_{\substack{i=1,\dots,n\\ j=1,\dots,n}}, & a^n_{i,j} &:= w_{n,j}\tilde{k}(t_{n,i}, t_{n,j}),\\
\boldsymbol{A}'_n &= (a'^n_{i,j})_{\substack{i=1,\dots,m\\ j=1,\dots,n}}, & a'^n_{i,j} &:= w_{n,j}\tilde{k}(t_{m,i}, t_{n,j}),
\end{aligned}
$$

so können wir das Iterationsverfahren (2.22) schreiben als

$$
\left.
\begin{aligned}
\boldsymbol{d}^{(k)} &= \tilde{\boldsymbol{r}} - (\boldsymbol{I}_n - \boldsymbol{A}_n)\triangle\boldsymbol{x}_n^{(k)},\\
\boldsymbol{v} &= \boldsymbol{A}_n\boldsymbol{d}^{(k)},\\
\boldsymbol{v}' &= \boldsymbol{A}'_n\boldsymbol{d}^{(k)},\\
\boldsymbol{\delta}'^{(k)} &= (\boldsymbol{I}_m - \boldsymbol{A}'_m)^{-1}\boldsymbol{v}',\\
\boldsymbol{\delta}^{(k)} &= \boldsymbol{v} + \boldsymbol{A}_m\boldsymbol{\delta}'^{(k)},\\
\triangle\boldsymbol{x}_n^{(k+1)} &= \triangle\boldsymbol{x}_n^{(k)} + \boldsymbol{\delta}^{(k)} + \boldsymbol{d}^{(k)},
\end{aligned}
\right\}
\qquad k = 0, 1, 2, \dots\ . \tag{2.23}
$$

2.3.4 Darstellung der Näherungslösung

Das Newton-Verfahren für den letzten Homotopieschritt, d. h. für $\mathcal{F}x = \mathcal{F}_1 x = 0$, liefert bei Beendigung nach μ Schritten eine Näherungslösung x_μ. Diese Näherungslösung hat eine Darstellung

$$
x_\mu(s) = x_0(s) + \sum_{k=0}^{\mu-1} \triangle x_k(s), \qquad s \in [a, b]
$$

als Summe von Korrekturtermen, welche jeweils Näherungslösung einer linearen Integralgleichung

$$
\triangle x_k(s) - \int_a^b \tilde{k}_k(s,t)\triangle x_k(t)\,dt = \tilde{r}_k(s), \qquad s \in [a, b]
$$

sind. Für diese linearen Integralgleichungen bestimmen wir mit dem Nyström-Verfahren Näherungslösungen in der Form

$$
\triangle x_k(s) \approx \triangle\tilde{x}_k(s) = \tilde{r}_k(s) + \sum_{j=1}^{n} w_{n,j}\tilde{k}_k(s, t_{n,j})\triangle\tilde{x}_k(t_{n,j}), \qquad s \in [a, b].
$$

Die Näherungslösung $\tilde{x}_\mu$ für $\mathcal{F}x = 0$ hat somit die für weitere Berechnungen ungünstige Form

$$
x_\mu(s) \approx \tilde{x}_\mu(s) = x_0(s) + \sum_{k=0}^{\mu-1} \triangle\tilde{x}_k(s) = x_0(s) + \sum_{k=0}^{\mu-1}\Big[\tilde{r}_k(s) + \sum_{j=1}^{n} w_{n,j}\tilde{k}_k(s, t_{n,j})\triangle\tilde{x}_k(t_{n,j})\Big].
$$

Setzen wir für $i = 1, \ldots, n$

$$\begin{aligned}\omega_{n,i} &:= \tilde{x}_\mu(t_{n,i}) \\ &= x_0(t_{n,i}) + \sum_{k=0}^{\mu-1} \Big[\tilde{r}_k(t_{n,i}) + \sum_{j=1}^{n} w_{n,j} \tilde{k}_k(t_{n,i}, t_{n,j}) \triangle \tilde{x}_k(t_{n,j})\Big] \\ &= x_0(t_{n,i}) + \sum_{k=0}^{\mu-1} \triangle \tilde{x}_k(t_{n,i}),\end{aligned}$$

so können wir durch Anwenden der Nyström-Diskretisierung auf die nichtlineare Integralgleichung (2.8) eine kontinuierliche Näherungslösung in der Form

$$\omega(s) := y(s) + \sum_{i=1}^{n} w_{n,i} k(s, t_{n,i}, \omega_{n,i}), \qquad s \in [a, b]$$

gewinnen.

2.3.5 Zusammenfassung

Um eine Näherungslösung ω der nichtlinearen Fredholmschen Integralgleichung (2.8) zu erhalten, linearisieren wir diese Gleichung mittels des Newton-Verfahrens. Die dabei in jedem Newton-Schritt auftretende lineare Integralgleichung lösen wir näherungsweise mit Hilfe des Nyström-Verfahrens. Um die Konvergenzgeschwindigkeit des Nyström-Verfahrens zu beschleunigen, regularisieren wir gegebenenfalls zuerst die lineare Integralgleichung. Die semidiskrete Nyström-Gleichung lösen wir möglichst effizient mit Hilfe des Zwei-Gitter-Verfahrens nach Methode 2. Um eine Startnäherung für das Newton-Verfahren zu gewinnen, verwenden wir ein Homotopieverfahren, welches in jedem Schritt die Berechnung einer Näherungslösung einer nichtlinearen Fredholmschen Integralgleichung zweiter Art erfordert. Zur Vorhersage einer Startnäherung für den nächsten Homotopieschritt ist das Lösen linearer Fredholmscher Integralgleichungen nötig, wozu wir wieder das Zwei-Gitter-Verfahren nach Methode 2 verwenden.

2.3.6 Parallelisierung

Der Hauptrechenaufwand bei der Bestimmung einer Näherungslösung ω besteht in der numerischen Lösung von linearen Fredholmschen Integralgleichungen zweiter Art mit dem Zwei-Gitter-Verfahren nach Methode 2. Dazu müssen wir in jedem Schritt dieses Verfahrens einige Matrix-Vektor-Multiplikationen durchführen und ein lineares Gleichungssystem auf dem groben Gitter lösen. Da die Matrix des linearen Gleichungssystems i.A. vollbesetzt und unsymmetrisch ist, verwenden wir die LU-Faktorisierung um das Gleichungssystem zu lösen.

Unsere Parallelisierung setzt somit ganz unten bei den Komponenten des Zwei-Gitter-Verfahren an. Im nächsten Abschnitt beschreiben wir zwei Routinen zur parallelen Matrix-

Vektor-Multiplikation mit einer rechteckigen Matrix. Anschließend stellen wir eine Routine zur parallelen LU-Faktorisierung vor.

Parallele Matrix-Vektor-Multiplikation

In diesem Abschnitt beschreiben wir, wie für die Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ und den Vektor $\boldsymbol{x} \in \mathbb{R}^m$ das Matrix-Vektor-Produkt $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ mit p Prozessoren berechnet werden kann. Die beiden Algorithmen sind dem Buch von Golub und van Loan [14] entnommen, in dem sich auch weitergehende Anregungen zur Parallelisierung von Matrixberechnungen befinden.

Als ersten Schritt unterteilen wir die Matrix $\boldsymbol{A}$ und die Vektoren $\boldsymbol{x}$ und $\boldsymbol{y}$ in Blockkomponenten möglichst gleicher Größe:

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{1,1} & \boldsymbol{A}_{1,2} & \cdots & \boldsymbol{A}_{1,p} \\ \boldsymbol{A}_{2,1} & \boldsymbol{A}_{2,2} & \cdots & \boldsymbol{A}_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{p,1} & \boldsymbol{A}_{p,2} & \cdots & \boldsymbol{A}_{p,p} \end{pmatrix}, \qquad \boldsymbol{x} = \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_p \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_p \end{pmatrix}$$

mit

$$\begin{aligned} \boldsymbol{A}_{i,j} &\in \mathbb{R}^{n_i\times m_j}, & i,j &= 1,\dots,p, \\ \boldsymbol{x}_i &\in \mathbb{R}^{m_i}, & i &= 1,\dots,p, \\ \boldsymbol{y}_i &\in \mathbb{R}^{n_i}, & i &= 1,\dots,p. \end{aligned}$$

Die Dimensionen der Blockkomponenten der Matrizen und Vektoren wählen wir dabei gerade so, dass $\sum_{i=1}^{p} n_i = n$ und $\sum_{i=1}^{p} m_i = m$ gilt. Zur Realisierung der Matrix-Vektor-Multiplikation betrachten wir zwei Möglichkeiten zur Aufteilung der Blockkomponenten von $\boldsymbol{A}$ auf die einzelnen Prozessoren (siehe Abbildung 2.3). In Abbildung 2.3 a) zeigen wir die Aufteilung in Blockspalten, in Abbildung 2.3 b) die Aufteilung in Blockzeilen. Jede dieser Aufteilungen kann im konkreten Anwendungsfall Vorteile gegenüber der anderen haben.

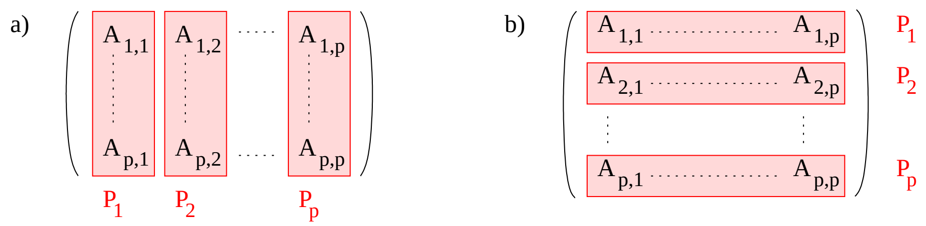


Abb. 2.3: Aufteilung der Matrix $\boldsymbol{A}$ für p Prozessoren.
a) Blockspalten-Aufteilung.
b) Blockzeilen-Aufteilung.

In beiden Fällen teilen wir die Vektoren $\boldsymbol{x}$ bzw. $\boldsymbol{y}$ in Blockkomponenten der Dimension m_i, $i = 1,\dots,p$, bzw. n_i, $i = 1,\dots,p$, auf (siehe Abbildung 2.4), wobei die i-te Blockkomponente dem Prozessor P_i zugeordnet wird.

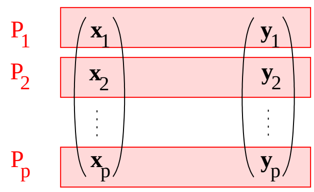


Abb. 2.4: Aufteilung zweier Vektoren $\boldsymbol{x}$ und $\boldsymbol{y}$ für p Prozessoren

Berechnung von $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ bei Aufteilung in Blockspalten

Bei der Aufteilung der Matrix $\boldsymbol{A}$ in Blockspalten berechnet der Prozessor P_j, $j = 1, \ldots, p$, die Matrix-Vektor-Produkte $\boldsymbol{y}_{i,j} = \boldsymbol{A}_{i,j}\boldsymbol{x}_j$, $i = 1, \ldots, p$. Die dafür notwendigen Matrix- und Vektor-Komponenten liegen dabei lokal auf dem Prozessor vor. Zur Berechnung der Vektor-Komponente $\boldsymbol{y}_i = \sum_{j=1}^{p} \boldsymbol{y}_{i,j}$ muss Kommunikation zwischen allen Prozessoren stattfinden. Die Berechnung der globalen Summe kann durch einen Fan-In-Algorithmus in $\log p$ Schritten berechnet werden (siehe Frommer [9, Kapitel 2]).

Algorithmus 2.3.11 berechnet das Matrix-Vektor-Produkt $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ bei einer Aufteilung in Blockspalten.

Algorithmus 2.3.11 (Matrix-Vektor-Multiplikation, Blockspalten).

$rank =$ `GetRank()`, $size =$ `GetSize()`
`For` $i = 1$ `To` $size$
  $\tilde{\boldsymbol{y}} = \boldsymbol{A}_{i,rank}\boldsymbol{x}_{rank}$
  `Reduce`$(+,\ \tilde{\boldsymbol{y}},\ \boldsymbol{y}_{rank})$ `To` P_i

Berechnung von $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ bei Aufteilung in Blockzeilen

Bei der Aufteilung der Matrix $\boldsymbol{A}$ in Blockzeilen berechnet jeder Prozessor P_i, $i = 1, \ldots, p$, alle seine Anteile $\boldsymbol{y}_{i,j} = \boldsymbol{A}_{i,j}\boldsymbol{x}_j$, $j = 1, \ldots, p$, zum Ergebnis $\boldsymbol{y}_i = \sum_{j=1}^{p} \boldsymbol{y}_{i,j}$. Dazu werden die entsprechenden Blockkomponenten von $\boldsymbol{x}$ benötigt. Zunächst wird in jedem Prozessor P_i, $i = 1, \ldots, p$, die Multiplikation mit dem Diagonalblock $\boldsymbol{A}_{i,i}$ durchgeführt, da die entsprechenden Komponenten $\boldsymbol{x}_i$ bei Programmstart vorliegt. Die Multiplikation in den darauf folgenden Schritten $i = 2, \ldots, p$ mit der i-ten oberen und der $(p-i)$-ten unteren Nebendiagonalen kann parallel erfolgen (Abbildung 2.5). Dazu ist im Prozessor P_i in jedem Schritt die aktuell vorliegende Komponente von $\boldsymbol{x}$ zum linken Prozessornachbarn in einer Ringtopologie zu versenden. Das Versenden dieser Vektor-Komponente kann im Hintergrund erfolgen und so hinter den Berechnungen verborgen werden (*latency hiding*).

Algorithmus 2.3.12 berechnet das Matrix-Vektor-Produkt $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ bei einer Aufteilung in Blockzeilen.

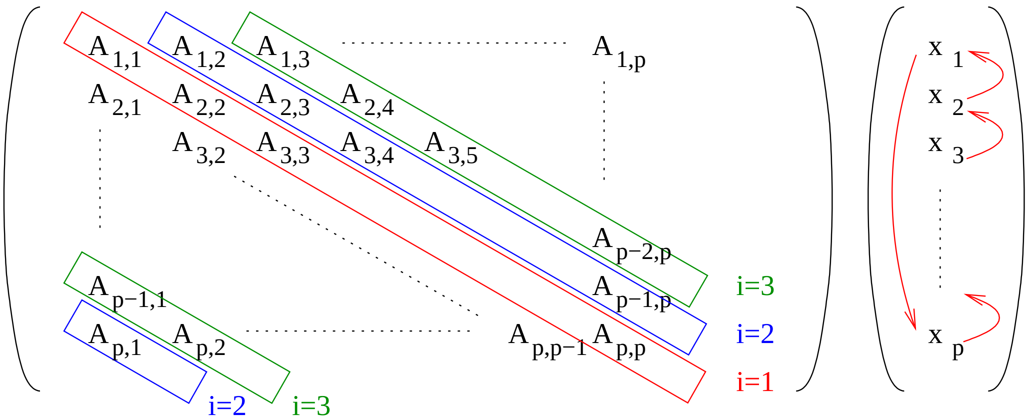


Abb. 2.5: Diagonalweises Durchlaufen der Matrix bei Matrix-Vektor-Multiplikation. Im Schritt k wird der Anteil von der k-ten oberen und der $(p-k)$-ten unteren Nebendiagonalen zur Matrix-Vektor-Multiplikation berechnet.

Algorithmus 2.3.12 (Matrix-Vektor-Multiplikation, Blockzeilen).

$rank =$ `GetRank()`, $\quad size =$ `GetSize()`
$pred = (rank - 1 - 1) \mod size + 1$
$succ = (rank - 1 + 1) \mod size + 1$
$\boldsymbol{y}_{rank} = 0, \quad \boldsymbol{x} = \boldsymbol{x}_{rank}$
$j = rank$
`For` $i = 1$ `To` $size$
  `ISend(`$\boldsymbol{x}$`)` `To` $pred$
  `IReceive(`$\tilde{\boldsymbol{x}}$`)` `From` $succ$
  $\boldsymbol{y}_{rank} = \boldsymbol{y}_{rank} + \boldsymbol{A}_{rank,j}\boldsymbol{x}$
  `Wait(`$\tilde{\boldsymbol{x}}$`)`
  $\boldsymbol{x} = \tilde{\boldsymbol{x}}$
  $j = (j - 1 + 1) \mod size + 1$

Die parallele LU-Faktorisierung mit Pivotsuche

Um ein lineares Gleichungssystem

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

mit nichtsingulärer Matrix $\boldsymbol{A} = (a_{i,j}) \in \mathbb{R}^{n\times n}$ und rechter Seite $\boldsymbol{b} = (b_i) \in \mathbb{R}^n$ zu lösen, setzen wir zunächst voraus, dass für die Matrix $\boldsymbol{A}$ eine Zerlegung der Form

$$\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$$

existiert, so dass die Matrix $\boldsymbol{L} = (l_{i,j}) \in \mathbb{R}^{n\times n}$ untere Dreiecksgestalt und die Matrix $\boldsymbol{U} = (u_{i,j}) \in \mathbb{R}^{n\times n}$ obere Dreiecksgestalt besitzen (*LU-Zerlegung von* $\boldsymbol{A}$). Die Lösung

$\boldsymbol{x} = (x_i) \in \mathbb{R}^n$ ergibt sich dann leicht über die beiden gestaffelten Gleichungssysteme

$$\boldsymbol{L}\tilde{\boldsymbol{x}} = \boldsymbol{b}, \qquad \boldsymbol{U}\boldsymbol{x} = \tilde{\boldsymbol{x}}.$$

Die Berechnung der Faktoren $\boldsymbol{L}$ und $\boldsymbol{U}$ dieser Zerlegung erfordert dabei den höchsten Rechenaufwand. Deshalb beschreiben wir zuerst, wie diese Zerlegung seriell, dann parallel erzeugt wird. Bei der Parallelisierung folgen wir dem Vorgehen in Frommer [9, Kapitel 4].

In Algorithmus 2.3.13 geben wir die einfachste Form der LU-Zerlegung in Codeform an.

Algorithmus 2.3.13 (serielle LU-Zerlegung ohne Pivotsuche).

$\boldsymbol{L} = \boldsymbol{I}, \quad \boldsymbol{U} = \boldsymbol{A}$

For $k = 1$ To $n-1$

  For $i = k+1$ To n

    $l_{i,k} = \dfrac{u_{i,k}}{u_{k,k}}, \quad u_{i,k} = 0$

    For $j = k+1$ To n

      $u_{i,j} = u_{i,j} - l_{i,k}u_{k,j}$

Die beiden nachfolgenden Matrizen zeigen schematisch die Veränderungen, die der Algorithmus im k-ten Schritt (hier $k = 3$) an den Matrizen $\boldsymbol{L}$ und $\boldsymbol{U}$ durchführt. Mit dem Symbol $\star$ gekennzeichnete Einträge sind in diesem Schritt neu erzeugte, von Null verschiedene Elemente. Mit $\circ$ gekennzeichnete Elemente werden im weiteren Verlauf der LU-Zerlegung nicht weiter benötigt und auch nicht weiter verändert.

$$\boldsymbol{L} = \begin{pmatrix} 1 & & & & & \\ \circ & 1 & & & & \\ \circ & \circ & 1 & & & \\ \circ & \circ & \star & 1 & & \\ \circ & \circ & \star & & 1 & \\ \vdots & \vdots & \vdots & & & \ddots & \\ \circ & \circ & \star & & & & 1 \end{pmatrix}, \qquad \boldsymbol{U} = \begin{pmatrix} \circ & \circ & \circ & \circ & \circ & \dots & \circ \\ & \circ & \circ & \circ & \circ & \dots & \circ \\ & & \circ & \circ & \circ & \dots & \circ \\ & & & \star & \star & \dots & \star \\ & & & \star & \star & \dots & \star \\ & & & \vdots & \vdots & \ddots & \vdots \\ & & & \star & \star & \dots & \star \end{pmatrix}.$$

Die Faktoren $\boldsymbol{L}$ und $\boldsymbol{U}$ werden auf Grund dieser Speicherbenutzung bei der praktischen Implementierung der LU-Zerlegung auf einem Rechner im Speicherbereich der Matrix $\boldsymbol{A}$ abgelegt und benötigen damit keinen zusätzlichen Speicher.

Der einfache Algorithmus 2.3.13 birgt aber Probleme, denn auch für eine nichtsinguläre Matrix $\boldsymbol{A}$ kann das Element $u_{k,k}$, durch das im k-ten Schritt dividiert wird, verschwinden.

Ein weiteres Problem ergibt sich, wenn im k-ten Schritt $u_{k,k}$ sehr klein ist. In diesem Fall sind die Zahlen $l_{i,k}$, $i = k+1, \dots n$, sehr groß, was zu großen Rundungsfehlern bei der Berechnung von $\boldsymbol{L}$ und $\boldsymbol{U}$ führen kann.

Aus diesen beiden Gründen berechnen wir keine Faktorisierung von $\boldsymbol{A}$, sondern verwenden bestimmte *Pivotstrategien*, die Zeilen oder Spalten von $\boldsymbol{A}$ vertauschen, und bestimmen eine Zerlegung der umsortierten Matrix.

Bei der *Spaltenpivotsuche* bestimmen wir im k-ten Schritt das betragsgrößte Element von $\boldsymbol{U}$ in der k-ten Spalte unterhalb der Hauptdiagonale einschließlich des Diagonalelementes. Danach vertauschen wir die zugehörige Zeile s und die Zeile k. Auf die daraus resultierende, modifizierte Matrix $\boldsymbol{U}$ wenden wir wie schon bei der Faktorisierung ohne Pivotsuche einen Gauß-Eliminationsschritt an. Durch dieses Vorgehen berechnen wir eine LU-Zerlegung der Matrix $\boldsymbol{PA} = \boldsymbol{LU}$, mit einer durch die Vertauschungen in den einzelnen Schritten bestimmten Permutationsmatrix $\boldsymbol{P}$. Aus den beiden gestaffelten Gleichungssystemen

$$\boldsymbol{L}\tilde{\boldsymbol{x}} = \boldsymbol{Pb}, \qquad \boldsymbol{U}\boldsymbol{x} = \tilde{\boldsymbol{x}}$$

bestimmen wir nun die gesuchte Lösung von $\boldsymbol{Ax} = \boldsymbol{b}$.

Bei der *Zeilenpivotsuche* bestimmen wir im k-ten Schritt das betragsgrößte Element von $\boldsymbol{U}$ in der k-ten Zeile rechts der Hauptdiagonalen einschließlich des Diagonalelementes und vertauschen danach die zugehörige Spalte s und die Spalte k. Auf die so modifizierte Matrix $\boldsymbol{U}$ wenden wir wieder einen Gauß-Eliminationsschritt an. Auf diese Weise berechnen wir eine LU-Zerlegung der Matrix $\boldsymbol{AP} = \boldsymbol{LU}$, die über die beiden gestaffelten Gleichungssysteme

$$\boldsymbol{L}\tilde{\tilde{\boldsymbol{x}}} = \boldsymbol{b}, \qquad \boldsymbol{U}\tilde{\boldsymbol{x}} = \tilde{\tilde{\boldsymbol{x}}} \tag{2.24}$$

und $\boldsymbol{x} = \boldsymbol{P}\tilde{\boldsymbol{x}}$ zu einer Lösung von $\boldsymbol{Ax} = \boldsymbol{b}$ führt.

Eine serielle Version der LU-Zerlegung mit Zeilenpivotsuche hat die in Algorithmus 2.3.14 beschriebene Form:

Algorithmus 2.3.14 (serielle LU-Zerlegung mit Zeilenpivotsuche).

$\boldsymbol{P} = \boldsymbol{I}, \quad \boldsymbol{L} = \boldsymbol{I}, \quad \boldsymbol{U} = \boldsymbol{A}$

`For` $k = 1$ `To` $n-1$

  Bestimme einen Index s mit $|u_{k,s}| = \max\limits_{j=k,\dots,n} |u_{k,j}|$

  Vertausche die Spalten mit Index s und k von $\boldsymbol{U}$ und $\boldsymbol{P}$

  `For` $i = k+1$ `To` n

    $l_{i,k} = \dfrac{u_{i,k}}{u_{k,k}}, \quad u_{i,k} = 0$

    `For` $j = k+1$ `To` n

      $u_{i,j} = u_{i,j} - l_{i,k}u_{k,j}$

Auch hier können wir wieder, um zusätzliche Belegung von Speicherplatz zu vermeiden, die Matrizen $\boldsymbol{L}$ und $\boldsymbol{U}$ im Speicherbereich der Matrix $\boldsymbol{A}$ ablegen. Zur Speicherung der Matrix $\boldsymbol{P}$ genügt es, einen Vektor anzulegen, der die Spaltenvertauschungen während des Algorithmus mitprotokolliert.

Für die parallele Durchführung der LU-Faktorisierung wollen wir die Matrix $\boldsymbol{A}$ nicht auf allen Prozessoren vollständig vorliegen haben, sondern nur immer die Zeilen, die im jeweiligen Prozessor benötigt werden. In Definition 2.3.15 erklären wir, wie die Datenverteilung erfolgen soll, wenn die Prozessoren $P_1, \dots, P_p$ zur Verfügung stehen.

Definition 2.3.15 (Zyklisch nach Zeilen (Spalten) abgespeichert). *Die Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ heißt im Parallelrechner mit den Prozessoren $P_1, \dots, P_p$* zyklisch nach Zeilen

(Spalten) abgespeichert*, falls Prozessor P_i genau die Zeilen (Spalten) j von $\boldsymbol{A}$ enthält mit* $(j \mod p) + 1 = i$.

In Algorithmus 2.3.16 beschreiben wir die parallele LU-Zerlegung mit Zeilenpivotsuche für Matrizen, die zyklisch nach Zeilen abspeichert sind. Hier können wir wieder die Matrizen $\boldsymbol{L}$ und $\boldsymbol{U}$ im Speicherbereich der Matrix $\boldsymbol{A}$ ablegen und die Matrix $\boldsymbol{P}$ durch einen Vektor repräsentieren.

Algorithmus 2.3.16 (parallele LU-Zerlegung mit Zeilenpivotsuche).

$rank =$ `GetRank()`, $\quad size =$ `GetSize()`
$myrows = \{i \in \mathbb{N} | \, (i \mod size) + 1 = rank, \; i \leq n\}$
$\boldsymbol{P} = \boldsymbol{I}, \quad \boldsymbol{L} = \boldsymbol{I}, \quad \boldsymbol{U} = \boldsymbol{A}$ (nur für Zeilen aus $myrows$)
`For` $k = 1$ `To` $n - 1$
  `If` $(k \in myrows)$ `Then`
    Bestimme einen Index s mit $|u_{k,s}| = \max_{j=k,\dots,n} |u_{k,j}|$
    `Broadcast(`$\{u_{k,k}, u_{k,k+1}, \dots, u_{k,n}, s\}$`)`
    $(t_k, t_{k+1}, \dots, t_n) = (u_{k,k}, u_{k,k+1}, \dots, u_{k,n})$
  `Else`
    `Receive(`$\{t_k, t_{k+1}, \dots, t_{kn}, s\}$`) From` $P_{(k \mod size)+1}$
  Vertausche Spalte mit Index s und k von $\boldsymbol{U}$ und $\boldsymbol{P}$ (nur für Zeilen aus $myrows$)
  Vertausche Elemente mit Index s und k von $\boldsymbol{t}$
  `For` $i = k + 1$ `To` n
    `If` $(i \in myrows)$ `Then`
      $l_{i,k} = \dfrac{u_{i,k}}{t_k}, \quad u_{i,k} = 0$
      `For` $j = k + 1$ `To` n
        $u_{i,j} = u_{i,j} - l_{i,k} t_j$

Auf die Parallelisierung der Auflösung nach dem gesuchten Vektor $\boldsymbol{x}$ aus der LU-Zerlegung (siehe (2.24)) gehen wir hier nicht näher ein. Da diese einen bedeutend geringeren Rechenaufwand benötigt als die LU-Zerlegung ($O(n^2)$ im Vergleich zu $O(n^3)$) und viel Kommunikation für eine verteilte Berechnung erfordert, kann es sinnvoll sein, die Lösung $\boldsymbol{x}$ auf nur einem Prozessor zu berechnen und danach an alle anderen zu versenden.

2.4 Berechnung einer Konstanten δ

In diesem Abschnitt stellen wir eine Möglichkeit für die Abschätzung des Defektes

$$(d(\omega))(s) = \omega(s) - \int_a^b k(s, t, \omega(t)) \, dt - y(s)$$

in der Unendlichnorm vor. Wir berechnen also eine Konstante δ, so dass

$$||d(\omega)|| \leq \delta \tag{2.25}$$

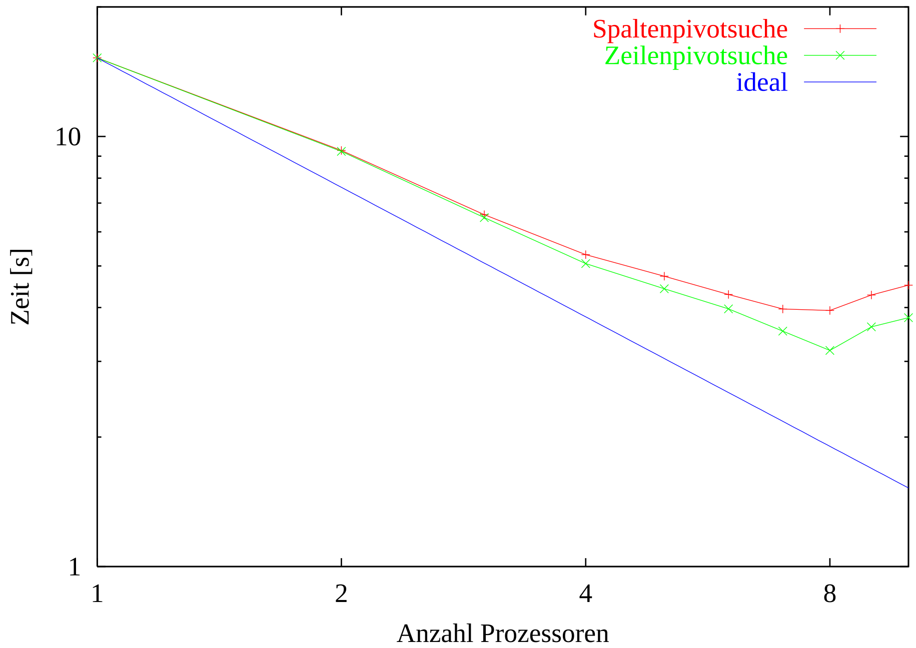


Abb. 2.6: Benötigte Rechenzeit für die LU-Faktorisierung einer 1000×1000 Matrix auf einem Cluster mit 800 MHz Pentium III Prozessoren und Gigabit-Ethernet. Die Faktorisierung wurde einmal mit Spaltenpivotsuche und einmal mit Zeilenpivotsuche durchgeführt.

gilt. Das hier beschriebene Vorgehen beruht auf der Verwendung der zentrierten Form mit Steigung (siehe Abschnitt 1.5). Stellen wir die Näherungslösung ω in der Form

$$\omega(s) = y(s) + \sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j})$$

mit gegebenen Quadraturknoten $t_{n,j}$, $j = 1, \dots, n$, und Quadraturgewichten $w_{n,j}$, $j = 1, \dots, n$, dar (vergleiche Abschnitt 2.3.4), so nimmt der Defekt $d(\omega)$ die einfache Gestalt

$$\begin{aligned} (d(\omega))(s) &= \omega(s) - \int_a^b k(s, t, \omega(t))\, dt - y(s) \\ &= y(s) + \sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j}) - \int_a^b k(s, t, \omega(t))\, dt - y(s) \\ &= \sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j}) - \int_a^b k(s, t, \omega(t))\, dt \end{aligned} \tag{2.26}$$

an. Wir unterteilen das Intervall $[a, b]$ in m gleich große Intervalle $[s]_{m,i}$, $i = 1, \dots, m$, mit Durchmesser $h = (b - a)/m$:

$$[s]_{m,i} := [a + (i - 1)h, a + ih], \qquad i = 1, \dots, m.$$

Außerdem setzen wir:

$$s^c_{m,i} := \mathrm{mid}([s]_{m,i}), \qquad [t]_{m,i} := [s]_{m,i}, \qquad t^c_{m,i} := s^c_{m,i}, \qquad i = 1, \ldots, m.$$

Damit können wir für die beiden Funktionen $f_1(s,t) = s$ und $f_2(s,t) = t$ auf den Intervallen $[s]_{m,i}$ bzw. $[t]_{m,i}$, $i = 1, \ldots, m$, Steigungstupel definieren (siehe Bemerkung 1.5.4):

$$\langle s\rangle_{m,i} := \begin{pmatrix} [s]_{m,i} \\ s^c_{m,i} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix}, \qquad \langle t\rangle_{m,i} := \begin{pmatrix} [t]_{m,i} \\ t^c_{m,i} \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix}, \qquad i = 1, \ldots, m.$$

Die Näherungslösung ω, die in Ausdruck (2.26) auftretende Summe und den Integranden werten wir in Steigungsarithmetik aus:

$$\langle\omega\rangle_{m,i} = \begin{pmatrix} [\omega]^x_{m,i} \\ [\omega]^c_{m,i} \\ \begin{pmatrix} 0 \\ [\omega]^{s_2}_{m,i} \end{pmatrix} \end{pmatrix} := y(\langle t\rangle_{m,i}) + \sum_{j=1}^{n} w_{n,j} k(\langle t\rangle_{m,i}, t_{n,j}, \omega_{n,j}), \qquad i = 1, \ldots, m,$$

$$\langle\Sigma\rangle_{m,i} = \begin{pmatrix} [\Sigma]^x_{m,i} \\ [\Sigma]^c_{m,i} \\ \begin{pmatrix} [\Sigma]^{s_1}_{m,i} \\ 0 \end{pmatrix} \end{pmatrix} := \sum_{j=1}^{n} w_{n,j} k(\langle s\rangle_{m,i}, t_{n,j}, \omega_{n,j}), \qquad i = 1, \ldots, m,$$

$$\langle k\rangle_{i,j} = \begin{pmatrix} [k]^x_{i,j} \\ [k]^c_{i,j} \\ \begin{pmatrix} [k]^{s_1}_{i,j} \\ [k]^{s_2}_{i,j} \end{pmatrix} \end{pmatrix} := k(\langle s\rangle_{m,i}, \langle t\rangle_{m,j}, \langle\omega\rangle_{m,j}), \qquad i, j = 1, \ldots, m.$$

Für $s \in [s]_{m,i}$, $1 \le i \le m$, und $t \in [t]_{m,j}$, $1 \le j \le m$, gilt daher

$$k(s,t,\omega(t)) \in [k]^c_{i,j} + [k]^{s_1}_{i,j}(s - s^c_{m,i}) + [k]^{s_2}_{i,j}(t - t^c_{m,j}).$$

Als Majorante für $k(s,t,\omega(t))$ für $s \in [s]_{m,i}$, $1 \le i \le m$, $t \in [t]_{m,j}$, $1 \le j \le m$, erhalten wir somit

$$\begin{aligned} k(s,t,\omega(t)) &\le \overline{k}^c_{i,j} + \max(\underline{k}^{s_1}_{i,j}(s - s^c_{m,i}), \overline{k}^{s_1}_{i,j}(s - s^c_{m,i})) + \max(\underline{k}^{s_2}_{i,j}(t - t^c_{m,j}), \overline{k}^{s_2}_{i,j}(t - t^c_{m,j})) \\ &= \overline{k}^c_{i,j} + (s - s^c_{m,i}) \begin{Bmatrix} \underline{k}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \overline{k}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} + (t - t^c_{m,j}) \begin{Bmatrix} \underline{k}^{s_2}_{i,j}, & t \le t^c_{m,j}, \\ \overline{k}^{s_2}_{i,j}, & t > t^c_{m,j} \end{Bmatrix}. \end{aligned}$$

Ebenso erhalten wir als Minorante

$$k(s,t,\omega(t)) \ge \underline{k}^c_{i,j} + (s - s^c_{m,i}) \begin{Bmatrix} \overline{k}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \underline{k}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} + (t - t^c_{m,j}) \begin{Bmatrix} \overline{k}^{s_2}_{i,j}, & t \le t^c_{m,j}, \\ \underline{k}^{s_2}_{i,j}, & t > t^c_{m,j} \end{Bmatrix}.$$

Diese Majorante verwenden wir nun, um eine Majorante für das Integral über k zu bestimmen. Dazu zerlegen wir das Intervall $[a,b]$ zuerst in Teilintervalle und integrieren dann:

$$\begin{aligned}
\int_a^b k(s,t,\omega(t))\,dt &= \sum_{j=1}^{m} \int_{[t]_{m,j}} k(s,t,\omega(t))\,dt \\
&\le \sum_{j=1}^{m} \int_{[t]_{m,j}} \Bigg[\overline{k}^{c}_{i,j} + (s - s^c_{m,i}) \begin{Bmatrix} \underline{k}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \overline{k}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} \\
&\quad + (t - t^c_{m,j}) \begin{Bmatrix} \underline{k}^{s_2}_{i,j}, & t \le t^c_{m,j}, \\ \overline{k}^{s_2}_{i,j}, & t > t^c_{m,j} \end{Bmatrix} \Bigg]\,dt \\
&= h \sum_{j=1}^{m} \overline{k}^{c}_{i,j} + h(s - s^c_{m,i}) \begin{Bmatrix} \sum_{j=1}^m \underline{k}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \sum_{j=1}^m \overline{k}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} \\
&\quad + \sum_{j=1}^{m} \int_{\underline{t}_{m,j}}^{t^c_{m,j}} (t - t^c_{m,j}) \underline{k}^{s_2}_{i,j}\,dt + \sum_{j=1}^{m} \int_{t^c_{m,j}}^{\overline{t}_{m,j}} (t - t^c_{m,j}) \overline{k}^{s_2}_{i,j}\,dt \\
&= h \sum_{j=1}^{m} \overline{k}^{c}_{i,j} + h(s - s^c_{m,i}) \begin{Bmatrix} \sum_{j=1}^m \underline{k}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \sum_{j=1}^m \overline{k}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} \\
&\quad + \sum_{j=1}^{m} \frac{1}{2}(t - t^c_{m,j})^2 \underline{k}^{s_2}_{i,j} \Big|_{\underline{t}_{m,j}}^{t^c_{m,j}} + \sum_{j=1}^{m} \frac{1}{2}(t - t^c_{m,j})^2 \overline{k}^{s_2}_{i,j} \Big|_{t^c_{m,j}}^{\overline{t}_{m,j}} \\
&= h \sum_{j=1}^{m} \overline{k}^{c}_{i,j} + h(s - s^c_{m,i}) \begin{Bmatrix} \sum_{j=1}^m \underline{k}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \sum_{j=1}^m \overline{k}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} \\
&\quad - \sum_{j=1}^{m} \frac{1}{2}(\underline{t}_{m,j} - t^c_{m,j})^2 \underline{k}^{s_2}_{i,j} + \sum_{j=1}^{m} \frac{1}{2}(t^c_{m,j} - t^c_{m,j})^2 \overline{k}^{s_2}_{i,j} \\
&= h \sum_{j=1}^{m} \overline{k}^{c}_{i,j} + h(s - s^c_{m,i}) \begin{Bmatrix} \sum_{j=1}^m \underline{k}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \sum_{j=1}^m \overline{k}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} + \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}).
\end{aligned}$$

Analog erhalten wir als Minorante

$$\int_a^b k(s,t,\omega(t))\,dt \ge h \sum_{j=1}^{m} \underline{k}^{c}_{i,j} + h(s - s^c_{m,i}) \begin{Bmatrix} \sum_{j=1}^m \overline{k}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \sum_{j=1}^m \underline{k}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} - \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}).$$

Für $s \in [s]_{m,i}$, $1 \le i \le m$, können wir die Summe in Ausdruck (2.26) durch

$$\sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j}) \in [\Sigma]^c_{m,i} + [\Sigma]^{s_1}_{m,i}(s - s^c_{m,i})$$

einschließen. Damit erhalten wir wieder, diesmal für die Summe in Ausdruck (2.26), eine Majorante bzw. Minorante:

$$\sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j}) \leq \overline{\Sigma}^{c}_{m,i} + (s - s^{c}_{m,i}) \begin{Bmatrix} \underline{\Sigma}^{s_1}_{m,i}, & s \leq s^{c}_{m,i}, \\ \overline{\Sigma}^{s_1}_{m,i}, & s > s^{c}_{m,i} \end{Bmatrix},$$

$$\sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j}) \geq \underline{\Sigma}^{c}_{m,i} + (s - s^{c}_{m,i}) \begin{Bmatrix} \overline{\Sigma}^{s_1}_{m,i}, & s \leq s^{c}_{m,i}, \\ \underline{\Sigma}^{s_1}_{m,i}, & s > s^{c}_{m,i} \end{Bmatrix}.$$

Diese Majorante und Minorante liefern für $s \in [s]_{m,i}$, $1 \leq i \leq m$, eine Majorante

$$\begin{aligned} &\sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j}) - \int_a^b k(s, t, \omega(t))\, dt \\ &\leq \overline{\Sigma}^{c}_{m,i} - h \sum_{j=1}^{m} \underline{k}^{c}_{i,j} + (s - s^{c}_{m,i}) \begin{Bmatrix} \underline{\Sigma}^{s_1}_{m,i} - h \sum_{j=1}^{m} \overline{k}^{s_1}_{i,j}, & s \leq s^{c}_{m,i}, \\ \overline{\Sigma}^{s_1}_{m,i} - h \sum_{j=1}^{m} \underline{k}^{s_1}_{i,j}, & s > s^{c}_{m,i} \end{Bmatrix} + \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}) \end{aligned}$$

bzw. Minorante

$$\begin{aligned} &\sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j}) - \int_a^b k(s, t, \omega(t))\, dt \\ &\geq \underline{\Sigma}^{c}_{m,i} - h \sum_{j=1}^{m} \overline{k}^{c}_{i,j} + (s - s^{c}_{m,i}) \begin{Bmatrix} \overline{\Sigma}^{s_1}_{m,i} - h \sum_{j=1}^{m} \underline{k}^{s_1}_{i,j}, & s \leq s^{c}_{m,i}, \\ \underline{\Sigma}^{s_1}_{m,i} - h \sum_{j=1}^{m} \overline{k}^{s_1}_{i,j}, & s > s^{c}_{m,i} \end{Bmatrix} - \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}) \end{aligned}$$

für Ausdruck (2.26). Die Majorante bzw. Minorante ist in den Intervallen $[\underline{s}_{m,i}, s^{c}_{m,i}]$ und $[s^{c}_{m,i}, \overline{s}_{m,i}]$ linear und nimmt ihr Maximum bzw. Minimum an den Randpunkten dieser Intervalle an. Als mögliche Extremalwerte der Majorante bzw. Minorante in $[s]_{m,i}$, $1 \leq i \leq m$, erhalten wir somit die folgenden sechs Werte:

$$\begin{aligned} p^1_{m,i} &:= \overline{\Sigma}^{c}_{m,i} - h \sum_{j=1}^{m} \underline{k}^{c}_{i,j} - \frac{h}{2}(\underline{\Sigma}^{s_1}_{m,i} - h \sum_{j=1}^{m} \overline{k}^{s_1}_{i,j}) + \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}), \\ p^2_{m,i} &:= \overline{\Sigma}^{c}_{m,i} - h \sum_{j=1}^{m} \underline{k}^{c}_{i,j} + \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}), \\ p^3_{m,i} &:= \overline{\Sigma}^{c}_{m,i} - h \sum_{j=1}^{m} \underline{k}^{c}_{i,j} + \frac{h}{2}(\overline{\Sigma}^{s_1}_{m,i} - h \sum_{j=1}^{m} \underline{k}^{s_1}_{i,j}) + \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}), \\ p^4_{m,i} &:= \underline{\Sigma}^{c}_{m,i} - h \sum_{j=1}^{m} \overline{k}^{c}_{i,j} - \frac{h}{2}(\overline{\Sigma}^{s_1}_{m,i} - h \sum_{j=1}^{m} \underline{k}^{s_1}_{i,j}) - \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}), \\ p^5_{m,i} &:= \underline{\Sigma}^{c}_{m,i} - h \sum_{j=1}^{m} \overline{k}^{c}_{i,j} - \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}), \\ p^6_{m,i} &:= \underline{\Sigma}^{c}_{m,i} - h \sum_{j=1}^{m} \overline{k}^{c}_{i,j} + \frac{h}{2}(\underline{\Sigma}^{s_1}_{m,i} - h \sum_{j=1}^{m} \overline{k}^{s_1}_{i,j}) - \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([k]^{s_2}_{i,j}). \end{aligned}$$

Damit können wir nun den Defekt in der Unendlichnorm beschränken:

$$\begin{aligned}
||d(\omega)|| &= \max_{s\in[a,b]} |d(\omega)(s)| \\
&= \max_{i=1,\dots,m} \max_{s\in[s]_{m,i}} \left| \sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j}) - \int_a^b k(s,t,\omega(t))\, dt \right| \\
&\leq \max_{i=1,\dots,m} \max \left\{ |p^1_{m,i}|, |p^2_{m,i}|, |p^3_{m,i}|, |p^4_{m,i}|, |p^5_{m,i}|, |p^6_{m,i}| \right\}.
\end{aligned}$$

Setzen wir also

$$\delta := \max_{i=1,\dots,m} \max \left\{ |p^1_{m,i}|, |p^2_{m,i}|, |p^3_{m,i}|, |p^4_{m,i}|, |p^5_{m,i}|, |p^6_{m,i}| \right\},$$

so ist Gleichung (2.25) erfüllt.

2.4.1 Parallelisierung

Die in den beiden folgenden Abschnitten vorgestellten Möglichkeiten zur Parallelisierung der Berechnung einer oberen Schranke des Defektes beruhen auf der Idee, das Gesamtproblem in sehr viele kleine, voneinander unabhängig berechenbare Teilprobleme zu zerlegen und diese dann möglichst geschickt auf die verfügbaren Prozessoren zu verteilen.

Bei der Defektberechnung sind solche unabhängig voneinander behandelbare Teilprobleme etwa die Berechnung des Steigungstripels für die Näherungslösung $\langle\omega\rangle_{m,i}$, $i = 1, \dots, m$, und die Berechnung von $\max \left\{ |p^1_{m,i}|, |p^2_{m,i}|, |p^3_{m,i}|, |p^4_{m,i}|, |p^5_{m,i}|, |p^6_{m,i}| \right\}$, $i = 1, \dots, m$.

Die wesentliche Schwierigkeit bei der Verteilung der Teilprobleme auf die zur Verfügung stehenden Prozessoren besteht darin, dass der Rechenzeitbedarf zum Lösen der einzelnen Teilprobleme nicht unbedingt konstant ist. Ein weiteres Problem entsteht, falls die verwendeten Prozessoren nicht alle gleich schnell sind. Eine optimale Verteilung der Teilaufgaben zu Beginn der Berechnung (*statische Lastverteilung*) ist deswegen im Allgemeinen nicht möglich. Stattdessen wird eine *dynamische Lastverteilung* notwendig, die die Teilprobleme während des Berechnungsvorgangs auf die Prozesse verteilt.

Im Folgenden werden zwei dynamische Lastverteilungsverfahren vorgestellt. Das erste ist das Master-Slave-Verfahren, das mit einer zentralen Aufgabenverteilung arbeitet. Das im darauf folgenden Abschnitt dargestellte Verfahren verwendet einen Ansatz mit dezentraler Aufgabenverteilung.

Für einen umfassenderen Einblick in das Problem der Lastverteilung verweisen wir auf die Bücher von Sanders und Worsch [43] oder Rauber und Rünger [41].

Lastverteilung durch Master-Slave-Verfahren

Die Grundidee des Master-Slave-Verfahrens besteht darin, dass ein Prozess, der so genannte *Master*, die Verwaltung der Teilprobleme zentral erledigt. Er weiß somit genau, welche Teilprobleme bereits abgearbeitet wurden, welche in Bearbeitung sind und welche noch auf ihre Erledigung warten. Alle anderen Prozesse, die so genannten *Slaves*, kümmern sich nicht um Verwaltungsaufgaben, sondern fordern beim Master-Prozess eine noch nicht bearbeitete Teilaufgabe an, führen diese dann durch und liefern das Ergebnis beim Master-Prozess ab. Sind alle Teilaufgaben abgearbeitet, so bekommen die Slave-Prozesse bei der Anfrage beim Master anstelle eines neuen Teilproblems mitgeteilt, dass sie sich nun beenden können. Algorithmus 2.4.1 gibt das hier beschriebene Verfahren in Pseudocode wieder.

Algorithmus 2.4.1 (Lastverteilung durch Master-Slave-Verfahren).

```
If (GetRank()= 1) Then        (Master-Prozess)
    Unterteile Arbeit in einzelne Arbeitsaufträge.
    Versorge jeden Slave-Prozess mit einem Arbeitsauftrag.
    Repeat
        Empfange ein Ergebnis von einem Slave-Prozess.
        Schicke diesem Slave-Prozess einen neuen Arbeitsauftrag.
    Until (alle Arbeitsaufträge sind versendet)
    Empfange restliche Ergebnisse von Slaves und sende Endesignal an Slaves.
Else      (Slave-Prozess)
    Repeat
        Empfange einen Auftrag vom Master-Prozess
        If (Auftrag = Arbeitsauftrag) Then
            Berechne Ergebnis des Arbeitsauftrags.
            Sende Ergebnis an Master-Prozess zurück.
        Else
            Gesamte Arbeit ist erledigt, Prozess kann beendet werden.
    Until ()
```

In Abbildung 2.7 wird der Rechenzeitbedarf zur Berechnung einer oberen Schranke δ des Defektes bei Parallelisierung durch das Master-Slave-Verfahren für eine unterschiedliche Anzahl von Teilintervallen m und Prozessoren dargestellt. Wie wir sehen, skaliert die Lastverteilung für die hier gezeigte Prozessorenanzahl sehr gut. In dem doppeltlogarithmischen Diagramm ist der Zusammenhang zwischen verwendeten Prozessoren und Rechenzeitbedarf beinahe linear.

Bei der Verwendung von mehr als 16 Prozessoren kann sich die zentrale Verwaltung der Aufgaben durch den Master-Prozess aber als Nadelöhr herausstellen, weil sämtliche Kommunikation über den Master-Prozess abläuft. Ein weiteres Problem dieses Algorithmus ist, dass die Prozessoren nicht exklusiv genutzt werden. Stehen beispielsweise 16 Prozessoren zur Verfügung, so müssen 17 Prozesse gestartet werden, ein Master-Prozess und 16 Prozesse, welche die Teilprobleme bearbeiten. Auf einem Prozessor laufen somit zwei Prozesse, was Prozessumschaltung erfordert. Im nächsten Abschnitt stellen wir daher einen Algorithmus vor, der die beschriebenen Probleme vermeidet, indem die einzelnen Teilprobleme dezentral und gleichberechtigt über alle Prozesse hinweg verwaltet werden.

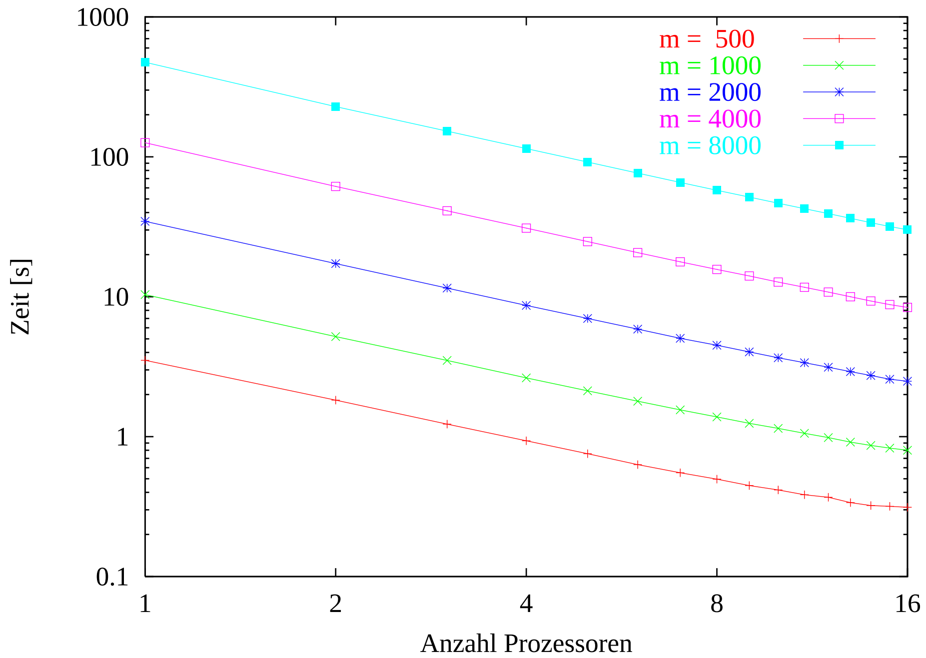


Abb. 2.7: Rechenzeit des Programms zur Berechnung einer oberen Schranke δ für den Defekt bei unterschiedlicher Anzahl von Teilintervallen m und Prozessoren. Die Lastverteilung erfolgte mittels Algorithmus 2.4.1. Die Berechnungen wurden auf einem Cluster mit 800 MHz Pentium III Prozessoren und Gigabit-Ethernet durchgeführt.

Lastverteilung durch zufälliges Anfragen

Im Algorithmus 2.4.2 wird die zentrale Aufgabenverwaltung des Master-Slave-Verfahrens durch einen dezentralen Ansatz ersetzt, der Engpässe vermeidet.

Die Teilprobleme werden gleichmäßig über alle Prozesse verteilt, so dass jedem Prozess von Beginn an eine bestimmte Anzahl von Teilproblemen, für die er die Verwaltung übernimmt, zur Verfügung steht. Hat ein Prozess die ihm zugeordneten Teilprobleme abgearbeitet, so sucht er sich neue Teilaufgaben, indem er zufällig bei einem anderen Prozess anfragt, ob dieser noch unerledigte Teilaufgaben hat. Bei einer positiven Antwort werden die unerledigten Teilaufgaben zwischen den beiden Prozessen aufgeteilt.

Schwierig wird es in dieser Situation festzustellen, ob es über alle Prozesse hinweg noch unerledigte Teilaufgaben gibt, da kein Prozess eine Liste über alle bereits erledigten Teilaufgaben führt. Dieses Problem wird durch unterschiedliche Prioritäten für die einzelnen Prozesse gelöst. Wird ein höher priorisierter Prozess während der eigenen Suche nach einer Teilaufgabe von einem niedriger priorisierten Prozess angefragt, so schickt er diesen in einen Schlafend-Zustand. In diesem Schlafend-Zustand verhält sich der Prozess passiv und stellt keine neuen Anfragen. Der Prozess mit der höchsten Priorität kann dann feststellen, ob alle Prozesse nach Arbeit suchen, und dann beschließen, die Programmabarbeitung zu beenden. Man kann dieses Vorgehen auch als eine Art Master-Slave-Verfahren betrachten, bei dem der höher priorisierte Prozess die Aufsicht über alle niedriger priorisierten Prozesse führt. Findet ein Prozess noch unerledigte Teilaufgaben, so teilt er diese mit allen von ihm in den Schlafend-Zustand versetzten Prozesse.

Algorithmus 2.4.2 beschreibt das Verfahren in Pseudocode.

Algorithmus 2.4.2 (Lastverteilung durch zufälliges Anfragen).

have work

Ist lokal keine Arbeit mehr vorhanden gehe nach "searching".

Erledige eine Teilaufgabe.

Ist eine Arbeitsanfrage vorhanden (Signal Arbeitsanfrage), so teile die lokal vorhandene Arbeit zwischen eigenem Prozess und dem anfragenden Prozess auf. Sende den einen Teil der Arbeit an den anfragenden Prozess (Signal Arbeitsantwort). Ist jetzt lokal keine Arbeit mehr vorhanden gehe nach "searching". Sind weitere Arbeitsanfragen vorhanden so wiederhole das Aufteilen der Arbeit.

Gehe nach "have work".

searching

Stelle an einen zufällig ausgewählten Prozess eine Arbeitsanfrage (Signal Arbeitsanfrage). Dem eigenen Prozess bekannte schlafende Prozesse sollen dabei nicht erneut angefragt werden.

Empfange Antwort:

- Signal Arbeitsanfrage: Hat der eigene Prozess eine höhere Priorität als der anfragende Prozess, dann schicke den anfragenden Prozess schlafen (Signal schlafen). Danach empfange von ihm alle von ihm schlafen gelegten Prozesse. Schlafen alle Prozesse außer dem eigenen Prozess, so gehe zu "end". Hat der eigene Prozess eine niedrigere Priorität als der anfragende Prozess schicke eine leere Arbeitsantwort (Signal Arbeitsantwort). Warte weiterhin auf Antwort.
- Signal Arbeitsantwort: Ist die Arbeitsantwort eine leere Arbeitsantwort, so stelle erneut eine Arbeitsanfrage an einen zufällig ausgewählten Prozess. Springe dazu an den Beginn von "searching". Ist die Arbeitsantwort nicht leer, so wecke (Signal Aufwachen) alle dem eigenen Prozess bekannten schlafenden Prozesse auf und verteile dabei gleichzeitig die erhaltene Arbeit gleichmäßig auf den eigenen Prozess und die bekannten schlafenden Prozesse. Gehe danach zu "have work".
- Signal schlafen: Sende die vom eigenen Prozess schlafen gelegten Prozesse an den Prozess, der den eigenen Prozess schlafen schickt. Die Verwaltung der Prozesse geht dabei auf diesen Prozess über. Gehe zu "sleeping".

sleeping

Empfange Nachricht:

- Signal Arbeitsanfrage: Antworte mit leerer Arbeitsantwort (Signal Arbeitsantwort). Empfange erneut eine Nachricht indem zum Beginn von “sleeping” gesprungen wird.
- Signal Aufwachen: Empfange den gleichzeitig mit der Übertragung des Aufwachsignals erhaltenen Arbeitsauftrag. Ist es ein leerer Arbeitsauftrag, dann gehe nach “searching”, ansonsten gehe nach “have work”.
- Signal Ende: Gehe zu “end”.

end

Schicke den beiden Blätter-Prozessen in einer Baumanordnung der Prozesse das Signal Ende. Sammle die getane Arbeit zur weiteren Verarbeitung ein. Beende die eigene Arbeit danach.

Bemerkung 2.4.3. *Im Zustand “searching” wird bei Eingang des Signals Arbeitsanfrage je nach eigener Priorität unterschiedlich verfahren. Dies ist notwendig, um Deadlocks in solchen Situationen zu verhindern, in denen Prozess A eine Arbeitsanfrage an Prozess B und Prozess B gleichzeitig eine Arbeitsanfrage an Prozess A stellt.*

In Abbildung 2.8 wird der Rechenzeitbedarf des mit Lastverteilung durch zufälliges Anfragen parallelisierten Programms zur Berechnung einer oberen Schranke δ des Defektes für unterschiedliche Anzahlen von Teilintervallen m über der Anzahl der verwendeten Prozessoren aufgetragen. Wie wir sehen können, skaliert der Algorithmus sehr gut. Selbst bei der kleinsten Problemgröße sehen wir im doppeltlogarithmischen Diagramm nahezu eine lineare Abhängigkeit zwischen Rechenzeitbedarf und Anzahl der Prozessoren. Eine Verdoppelung der Anzahl der Prozessoren führt also zu einer Halbierung der Rechenzeit. Wir können ebenfalls ablesen, dass eine Verdoppelung der Problemgröße zu einer Vervierfachung des Rechenzeitbedarfs führt, der Algorithmus zur Bestimmung einer oberen Schranke δ_m ist also ein $O(m^2)$-Algorithmus.

2.5 Berechnung einer Konstanten K

In diesem Abschnitt beschäftigen wir uns mit dem Problem, eine Konstante K zu finden, so dass die Inverse $\mathcal{L}^{-1}$ des um die Näherungslösung ω linearisierten Operators $\mathcal{I} - \mathcal{K}$

$$\mathcal{L}x := (\mathcal{I} - \mathcal{K})'[\omega]x = (\mathcal{I} - \tilde{\mathcal{K}})x = x - \int_a^b \tilde{k}(.,t)x(t)\,dt = x - \int_a^b k_3(.,t,\omega(t))x(t)\,dt$$

durch K nach oben beschränkt ist, d.h. so dass

$$||\mathcal{L}^{-1}|| \leq K$$

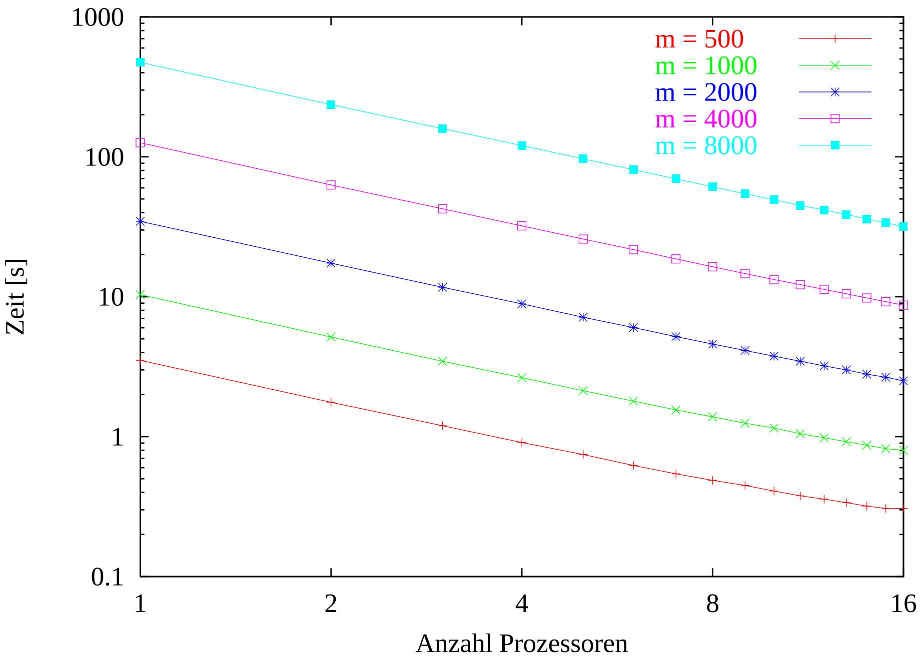


Abb. 2.8: Rechenzeit des Programms zur Berechnung einer oberen Schranke δ des Defektes bei unterschiedlicher Anzahl von Teilintervallen m und Prozessoren. Die Lastverteilung erfolgte mittels Algorithmus 2.4.2. Die Berechnungen wurden auf einem Cluster mit 800 MHz Pentium III Prozessoren und Gigabit-Ethernet durchgeführt.

gilt. Die Existenz einer solchen Konstanten setzt implizit voraus, dass $\mathcal{L}$ bijektiv und damit invertierbar ist. Ist $\mathcal{L}$ nicht invertierbar, so können wir das in Theorem 2.2.1 beschriebene Verfahren zur Einschließung einer Lösung der Fredholmschen Integralgleichung zweiter Art nicht anwenden. Sowohl die Invertierbarkeit von $\mathcal{L}$ als auch die gesuchte Konstante K erhalten wir aus folgendem Satz (vgl. Rall [38, Theorem 10.1]), einer Modifikation des Satzes über die Neumannsche Reihe, der es erlaubt, statt $\mathcal{L}$ einen anderen Operator $\tilde{\mathcal{L}}$, der nahe bei $\mathcal{L}$ liegt, zu untersuchen.

Satz 2.5.1 (Abschätzung von $||\mathcal{L}^{-1}||_{X\to X}$). *Sei X ein Banachraum mit der Norm $||.||_X$ und $\mathcal{L} \in L(X)$ ein beschränkter linearer Operator von X nach X. Der Operator $\mathcal{L}^{-1}$ existiert genau dann, wenn es einen beschränkten linearen Operator $\tilde{\mathcal{L}} \in L(X)$ gibt, für den $\tilde{\mathcal{L}}^{-1}$ existiert und*

$$||\tilde{\mathcal{L}} - \mathcal{L}||_{X\to X} < \frac{1}{||\tilde{\mathcal{L}}^{-1}||_{X\to X}}$$

gilt. In diesem Fall ist $||\mathcal{L}^{-1}||_{X\to X}$ durch

$$||\mathcal{L}^{-1}||_{X\to X} \leq \frac{||\tilde{\mathcal{L}}^{-1}||_{X\to X}}{1 - ||\mathcal{I} - \tilde{\mathcal{L}}^{-1}\mathcal{L}||_{X\to X}} \leq \frac{||\tilde{\mathcal{L}}^{-1}||_{X\to X}}{1 - ||\tilde{\mathcal{L}}^{-1}||_{X\to X}||\tilde{\mathcal{L}} - \mathcal{L}||_{X\to X}}. \tag{2.27}$$

beschränkt.

Wählen wir den approximierenden Operator $\tilde{\mathcal{L}}$ einfach genug, so können wir leicht überprüfen, ob $\tilde{\mathcal{L}}^{-1}$ existiert, und die benötigten Normen gegebenenfalls berechnen. Im Folgenden konstruieren wir deshalb zunächst einen leicht zu handhabenden Operator $\tilde{\mathcal{L}} : C([a,b]) \to C([a,b])$ mit entartetem Kern. Anschließend geben wir Abschätzungen für die in Satz 2.5.1 vorkommenden Normen an und zeigen, dass $\tilde{\mathcal{L}}$ die für die Anwendung des Satzes notwendigen Abbildungseigenschaften besitzt.

2.5.1 Konstruktion des approximierenden Operators $\tilde{\mathcal{L}}$

In diesem Abschnitt beschreiben wir eine Möglichkeit, den linearen Operator $\mathcal{L}$ durch einen einfacheren Operator $\tilde{\mathcal{L}}$ zu approximieren, bei der die Kernfunktion $\tilde{k}$ durch eine bilineare Approximation ersetzt wird (siehe Hackbusch [16, Abschnitt 4.2]).

Definieren wir als Schrittweite $h := (b-a)/m$ und setzen wir $s_{m,i} := a+ih$, $i = 0,\dots,m$, so zerfällt das Intervall $[a,b]$ in m gleichgroße Teilintervalle $[s]_{m,i} = [s_{m,i-1}, s_{m,i}]$, $i = 1,\dots,m$. Wir betrachten nun Funktionen

$$\Phi_{m,0}(s) := \begin{cases} 1 - \frac{1}{h}(s - s_{m,0}), & s \in [s]_{m,1}, \\ 0, & \text{sonst}, \end{cases}$$

$$\Phi_{m,i}(s) := \begin{cases} 1 + \frac{1}{h}(s - s_{m,i}), & s \in [s]_{m,i}, \\ 1 - \frac{1}{h}(s - s_{m,i}), & s \in [s]_{m,i+1}, \\ 0, & \text{sonst}, \end{cases} \qquad i = 1,\dots,m-1,$$

$$\Phi_{m,m}(s) := \begin{cases} 1 + \frac{1}{h}(s - s_{m,m}), & s \in [s]_{m,m}, \\ 0, & \text{sonst}, \end{cases}$$

die stückweise linear sind und die $\Phi_{m,i}(s_{m,j}) = \delta_{i,j}$ erfüllen.

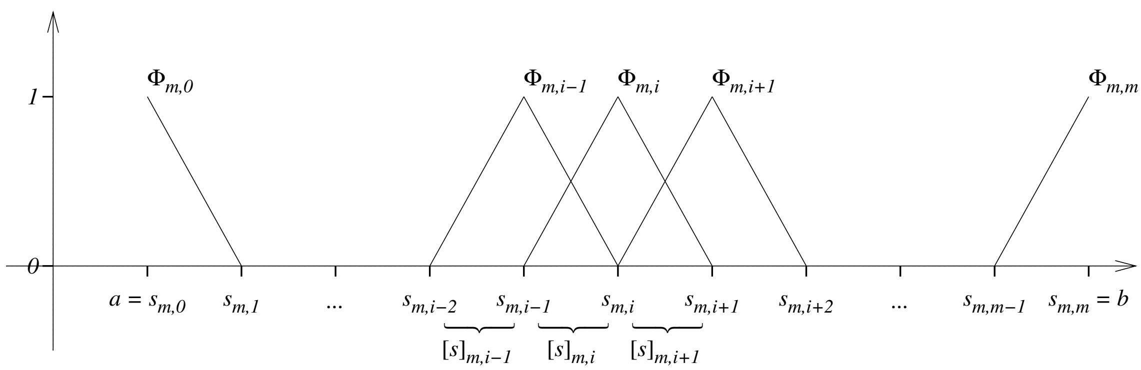


Abb. 2.9: Die Hütchenfunktionen $\Phi_{m,i}$, $i = 0,\dots,m$.

Mit Hilfe dieser so genannten *Hütchenfunktionen* (siehe Abbildung 2.9) wird die Kernfunktion von $\mathcal{L}$ bilinear approximiert durch

$$\tilde{k}(s,t) \approx \tilde{k}_m(s,t) := \sum_{i=0}^{m}\sum_{j=0}^{m} \tilde{k}(s_{m,i}, s_{m,j})\Phi_{m,i}(s)\Phi_{m,j}(t) = \sum_{i=0}^{m}\sum_{j=0}^{m} \tilde{k}_{i,j}\Phi_{m,i}(s)\Phi_{m,j}(t)$$

mit $\tilde{k}_{i,j} := \tilde{k}(s_{m,i}, s_{m,j}) = k_3(s_{m,i}, s_{m,j}, \omega(s_{m,j}))$. Mit dem Operator

$$\tilde{\mathcal{L}}x = \mathcal{L}_m x = (\mathcal{I} - \tilde{\mathcal{K}}_m)x := x - \int_a^b \tilde{k}_m(.,t)x(t)\,dt \tag{2.28}$$

approximieren wir nun

$$\mathcal{L}x = (\mathcal{I} - \tilde{\mathcal{K}})x = x - \int_a^b \tilde{k}(.,t)x(t)\,dt. \tag{2.29}$$

2.5.2 Abschätzen der Norm des Operators $\mathcal{L}^{-1}$

Wir wollen nun die Konstante K bestimmen (siehe Satz 2.5.1). Dabei ergeben sich die zwei Teilprobleme für die Operatoren $\tilde{\mathcal{L}}$ aus (2.28) und $\mathcal{L}$ aus (2.29):

1. Bestimme eine Konstante K_1 mit $||\tilde{\mathcal{L}} - \mathcal{L}|| \leq K_1$ (siehe 2.5.3).
2. Zeige zunächst, dass der Operator $\tilde{\mathcal{L}}$ bijektiv und damit invertierbar ist. Bestimme dann eine Konstante K_2 mit $||\tilde{\mathcal{L}}^{-1}|| \leq K_2$ (siehe 2.5.4).

Gilt dann $K_1 < 1/K_2$, so sind die Voraussetzungen von Satz 2.5.1 erfüllt. Der Operator $\mathcal{L}^{-1}$ existiert somit und seine Operatornorm ist nach oben beschränkt durch

$$||\mathcal{L}^{-1}|| \leq \frac{||\tilde{\mathcal{L}}^{-1}||}{1 - ||\tilde{\mathcal{L}}^{-1}||\,||\tilde{\mathcal{L}} - \mathcal{L}||} \leq \frac{K_2}{1 - K_1K_2} =: K.$$

Bemerkung 2.5.2. *Die für die Anwendung des Satzes 2.5.1 notwendige Abbildungseigenschaft $\tilde{\mathcal{L}} \in L(C([a,b]))$ folgt aus Gleichung* (2.28), *$\tilde{\mathcal{L}}^{-1} \in L(C([a,b]))$ folgt aus der Darstellung von $x = \tilde{\mathcal{L}}^{-1}y$ in Gleichung* (2.32).

2.5.3 Bestimmung einer Konstanten K_1

In diesem Abschnitt beschreiben wir eine Möglichkeit, den Fehler $||\tilde{\mathcal{L}} - \mathcal{L}||$ mit Hilfe der zentrierten Form mit Steigung abzuschätzen:

$$||\tilde{\mathcal{L}} - \mathcal{L}|| \leq K_1. \tag{2.30}$$

Die Definition von $\tilde{\mathcal{L}}$ und $\mathcal{L}$ liefert:

$$||\tilde{\mathcal{L}} - \mathcal{L}|| = ||\mathcal{I} - \tilde{\mathcal{K}}_m - \mathcal{I} + \tilde{\mathcal{K}}|| = ||\tilde{\mathcal{K}} - \tilde{\mathcal{K}}_m||.$$

Da der Operator $\tilde{\mathcal{K}} - \tilde{\mathcal{K}}_m$ ein Fredholmscher Integraloperator ist, gilt nach Satz 1.1.4:

$$||\tilde{\mathcal{L}} - \mathcal{L}|| = \max_{s \in [a,b]} \int_a^b |\tilde{k}(s,t) - \tilde{k}_m(s,t)|\, dt. \tag{2.31}$$

Wir setzen

$$[t]_{m,i} := [s]_{m,i}, \qquad s^c_{m,i} = t^c_{m,i} := \mathrm{mid}([s]_{m,i}), \qquad i = 1, \ldots, m.$$

Wir bestimmen zuerst ein Steigungstupel für die Funktion $\tilde{k}_m$ auf $[s]_{m,i} \times [t]_{m,j}$, $1 \leq i, j \leq m$:

$$\begin{aligned}
\tilde{k}_m(s,t) &= \sum_{i=0}^{m} \sum_{j=0}^{m} \tilde{k}_{i,j} \Phi_{m,i}(s) \Phi_{m,j}(t) \\
&= \tilde{k}_{i,j} \Phi_{m,i}(s) \Phi_{m,j}(t) + \tilde{k}_{i,j-1} \Phi_{m,i}(s) \Phi_{m,j-1}(t) \\
&\quad + \tilde{k}_{i-1,j} \Phi_{m,i-1}(s) \Phi_{m,j}(t) + \tilde{k}_{i-1,j-1} \Phi_{m,i-1}(s) \Phi_{m,j-1}(t) \\
&= \Phi_{m,i}(s) \big[\tilde{k}_{i,j} \Phi_{m,j}(t) + \tilde{k}_{i,j-1} \Phi_{m,j-1}(t)\big] \\
&\quad + \Phi_{m,i-1}(s) \big[\tilde{k}_{i-1,j} \Phi_{m,j}(t) + \tilde{k}_{i-1,j-1} \Phi_{m,j-1}(t)\big] \\
&= \Phi_{m,i}(s) \Big[\frac{1}{2}(\tilde{k}_{i,j} + \tilde{k}_{i,j-1}) + \frac{1}{h}(\tilde{k}_{i,j} - \tilde{k}_{i,j-1})(t - t^c_{m,j})\Big] \\
&\quad + \Phi_{m,i-1}(s) \Big[\frac{1}{2}(\tilde{k}_{i-1,j} + \tilde{k}_{i-1,j-1}) + \frac{1}{h}(\tilde{k}_{i-1,j} - \tilde{k}_{i-1,j-1})(t - t^c_{m,j})\Big] \\
&= \frac{1}{2}\Big[\frac{1}{2}(\tilde{k}_{i,j} + \tilde{k}_{i,j-1}) + \frac{1}{h}(\tilde{k}_{i,j} - \tilde{k}_{i,j-1})(t - t^c_{m,j}) \\
&\qquad\qquad + \frac{1}{2}(\tilde{k}_{i-1,j} + \tilde{k}_{i-1,j-1}) + \frac{1}{h}(\tilde{k}_{i-1,j} - \tilde{k}_{i-1,j-1})(t - t^c_{m,j})\Big] \\
&\quad + (s - s^c_{m,i}) \frac{1}{h}\Big[\frac{1}{2}(\tilde{k}_{i,j} + \tilde{k}_{i,j-1}) + \frac{1}{h}(\tilde{k}_{i,j} - \tilde{k}_{i,j-1})(t - t^c_{m,j}) \\
&\qquad\qquad - \frac{1}{2}(\tilde{k}_{i-1,j} + \tilde{k}_{i-1,j-1}) - \frac{1}{h}(\tilde{k}_{i-1,j} - \tilde{k}_{i-1,j-1})(t - t^c_{m,j})\Big] \\
&= \frac{1}{4}(\tilde{k}_{i,j} + \tilde{k}_{i,j-1} + \tilde{k}_{i-1,j} + \tilde{k}_{i-1,j-1}) \\
&\quad + (s - s^c_{m,i}) \frac{1}{2h}(\tilde{k}_{i,j} + \tilde{k}_{i,j-1} - \tilde{k}_{i-1,j} - \tilde{k}_{i-1,j-1}) \\
&\quad + (t - t^c_{m,j}) \frac{1}{h}\Big[\frac{1}{2}(\tilde{k}_{i,j} + \tilde{k}_{i-1,j} - \tilde{k}_{i,j-1} - \tilde{k}_{i-1,j-1}) \\
&\qquad\qquad + \underbrace{(s - s^c_{m,i})}_{\in [-\frac{h}{2}, \frac{h}{2}]} \frac{1}{h}(\tilde{k}_{i,j} + \tilde{k}_{i-1,j-1} - \tilde{k}_{i,j-1} - \tilde{k}_{i-1,j})\Big] \\
&\in [\tilde{k}_m]^c_{i,j} + [\tilde{k}_m]^{s_1}_{i,j}(s - s^c_{m,i}) + [\tilde{k}_m]^{s_2}_{i,j}(t - t^c_{m,j})
\end{aligned}$$

mit

$$
\begin{aligned}
[\tilde{k}_m]^c_{i,j} &:= \frac{1}{4}(\tilde{k}_{i,j} + \tilde{k}_{i,j-1} + \tilde{k}_{i-1,j} + \tilde{k}_{i-1,j-1}),\\
[\tilde{k}_m]^{s_1}_{i,j} &:= \frac{1}{2h}(\tilde{k}_{i,j} + \tilde{k}_{i,j-1} - \tilde{k}_{i-1,j} - \tilde{k}_{i-1,j-1}),\\
[\tilde{k}_m]^{s_2}_{i,j} &:= \frac{1}{h}\big[\min\{\tilde{k}_{i,j} - \tilde{k}_{i,j-1}, \tilde{k}_{i-1,j} - \tilde{k}_{i-1,j-1}\}, \max\{\tilde{k}_{i,j} - \tilde{k}_{i,j-1}, \tilde{k}_{i-1,j} - \tilde{k}_{i-1,j-1}\}\big].
\end{aligned}
$$

Da $\tilde{k}_m$ auf $[s]_{m,i} \times [t]_{m,j}$, $1 \le i, j \le m$, bilinear ist, enthält das Intervall

$$[\tilde{k}_m]^x_{i,j} := \big[\min\{\tilde{k}_{i,j}, \tilde{k}_{i,j-1}, \tilde{k}_{i-1,j}, \tilde{k}_{i-1,j-1}\}, \max\{\tilde{k}_{i,j}, \tilde{k}_{i,j-1}, \tilde{k}_{i-1,j}, \tilde{k}_{i-1,j-1}\}\big]$$

alle Funktionswerte von $\tilde{k}_m(s,t)$ für $s \in [s]_{m,i}$, $t \in [t]_{m,j}$. Damit können wir nun durch

$$\langle \tilde{k}_m \rangle_{i,j} := \begin{pmatrix} [\tilde{k}_m]^x_{i,j} \\ [\tilde{k}_m]^c_{i,j} \\ \begin{pmatrix} [\tilde{k}_m]^{s_1}_{i,j} \\ [\tilde{k}_m]^{s_2}_{i,j} \end{pmatrix} \end{pmatrix}$$

ein Steigungstupel für $\tilde{k}_m$ auf $[s]_{m,i} \times [t]_{m,j}$, $1 \le i, j \le m$, definieren. Da wir im folgenden die Kernfunktion $\tilde{k}$ durch Steigungsarithmetik auswerten wollen, definieren wir für die beiden Funktionen $f_1(s,t) = s$ mit $s \in [s]_{m,i}$, $i = 1, \dots, m$, und $f_2(s,t) = t$ mit $t \in [t]_{m,j}$, $j = 1, \dots, m$, die Steigungstupel

$$\langle s \rangle_{m,i} := \begin{pmatrix} [s]_{m,i} \\ s^c_{m,i} \\ \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{pmatrix}, \qquad \langle t \rangle_{m,i} := \begin{pmatrix} [t]_{m,i} \\ t^c_{m,i} \\ \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{pmatrix}, \qquad i = 1, \dots, m.$$

Setzen wir

$$\langle q \rangle_{i,j} = \begin{pmatrix} [q]^x_{i,j} \\ [q]^c_{i,j} \\ [q]^{s_1}_{i,j} \\ [q]^{s_2}_{i,j} \end{pmatrix} := \mathrm{abs}(\tilde{k}(\langle s \rangle_{m,i}, \langle t \rangle_{m,j}) - \langle \tilde{k}_m \rangle_{i,j}),$$

so erhalten wir für alle $s \in [s]_{m,i}$, $1 \le i \le m$, sowie $t \in [t]_{m,j}$, $1 \le j \le m$, eine Majorante für den Integranden aus Gleichung (2.31):

$$|\tilde{k}(s,t) - \tilde{k}_m(s,t)| \le \overline{q}^c_{i,j} + (s - s^c_{m,i}) \begin{Bmatrix} \underline{q}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \overline{q}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} + (t - t^c_{m,j}) \begin{Bmatrix} \underline{q}^{s_2}_{i,j}, & t \le t^c_{m,j}, \\ \overline{q}^{s_2}_{i,j}, & t > t^c_{m,j} \end{Bmatrix}.$$

Damit können wir nun eine Majorante für das Integral aus (2.31) bestimmen:

$$
\begin{aligned}
&\int_a^b |\tilde{k}(s,t) - \tilde{k}_m(s,t)|\,dt \\
&\quad = \sum_{j=1}^{m} \int_{[t]_{m,j}} |\tilde{k}(s,t) - \tilde{k}_m(s,t)|\,dt \\
&\quad \le \sum_{j=1}^{m} \int_{[t]_{m,j}} \left[\overline{q}^c_{i,j} + (s - s^c_{m,i}) \begin{Bmatrix} \underline{q}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \overline{q}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} + (t - t^c_{m,j}) \begin{Bmatrix} \underline{q}^{s_2}_{i,j}, & t \le t^c_{m,j}, \\ \overline{q}^{s_2}_{i,j}, & t > t^c_{m,j} \end{Bmatrix} \right] dt \\
&\quad = \sum_{j=1}^{m} \left[h\overline{q}^c_{i,j} + h(s - s^c_{m,i}) \begin{Bmatrix} \underline{q}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \overline{q}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} \right. \\
&\qquad\qquad \left. + \int_{\underline{t}_{m,j}}^{t^c_{m,j}} (t - t^c_{m,j}) \underline{q}^{s_2}_{i,j}\,dt + \int_{t^c_{m,j}}^{\overline{t}_{m,j}} (t - t^c_{m,j}) \overline{q}^{s_2}_{i,j}\,dt \right] \\
&\quad = \sum_{j=1}^{m} \left[h\overline{q}^c_{i,j} + h(s - s^c_{m,i}) \begin{Bmatrix} \underline{q}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \overline{q}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} \right. \\
&\qquad\qquad \left. + \frac{\underline{q}^{s_2}_{i,j}}{2} (t - t^c_{m,j})^2 \Big|_{\underline{t}_{m,j}}^{t^c_{m,j}} + \frac{\overline{q}^{s_2}_{i,j}}{2} (t - t^c_{m,j})^2 \Big|_{t^c_{m,j}}^{\overline{t}_{m,j}} \right] \\
&\quad = \sum_{j=1}^{m} \left[h\overline{q}^c_{i,j} + h(s - s^c_{m,i}) \begin{Bmatrix} \underline{q}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \overline{q}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} - \frac{\underline{q}^{s_2}_{i,j}}{2} \frac{h^2}{4} + \frac{\overline{q}^{s_2}_{i,j}}{2} \frac{h^2}{4} \right] \\
&\quad = h \sum_{j=1}^{m} \overline{q}^c_{i,j} + h(s - s^c_{m,i}) \sum_{j=1}^{m} \begin{Bmatrix} \underline{q}^{s_1}_{i,j}, & s \le s^c_{m,i}, \\ \overline{q}^{s_1}_{i,j}, & s > s^c_{m,i} \end{Bmatrix} + \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([q]^{s_2}_{i,j}).
\end{aligned}
$$

Die Majorante ist auf den Intervallen $[\underline{s}_{m,i}, s^c_{m,i}]$ und $[s^c_{m,i}, \overline{s}_{m,i}]$, $1 \le i \le m$, jeweils linear und nimmt somit ihre Extrema am Rand an. Da als Extremwert nur

$$
\begin{aligned}
p^1_{m,i} &= h \sum_{j=1}^{m} \overline{q}^c_{i,j} - \frac{h^2}{2} \sum_{j=1}^{m} \underline{q}^{s_1}_{i,j} + \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([q]^{s_2}_{i,j}), \\
p^2_{m,i} &= h \sum_{j=1}^{m} \overline{q}^c_{i,j} + \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([q]^{s_2}_{i,j})
\end{aligned}
$$

oder

$$
p^3_{m,i} = h \sum_{j=1}^{m} \overline{q}^c_{i,j} + \frac{h^2}{2} \sum_{j=1}^{m} \overline{q}^{s_1}_{i,j} + \frac{h^2}{8} \sum_{j=1}^{m} \operatorname{diam}([q]^{s_2}_{i,j})
$$

(jeweils für $i = 1, \ldots, m$) in Betracht kommen, erhalten wir

$$\begin{aligned}||\tilde{\mathcal{L}} - \mathcal{L}|| &= \max_{s\in[a,b]} \int_a^b |\tilde{k}(s,t) - \tilde{k}_m(s,t)|\,dt \\ &= \max_{i=1,\ldots,m} \max_{s\in[s]_{m,i}} \int_a^b |\tilde{k}(s,t) - \tilde{k}_m(s,t)|\,dt \\ &\leq \max_{i=1,\ldots,m} \max\{p^1_{m,i}, p^2_{m,i}, p^3_{m,i}\}.\end{aligned}$$

Setzen wir also

$$K_1 := \max_{i=1,\ldots,m} \max\{p^1_{m,i}, p^2_{m,i}, p^3_{m,i}\},$$

so ist Ungleichung (2.30) erfüllt.

2.5.4 Bestimmung einer Konstanten K_2

In diesem Abschnitt beschreiben wir ein Verfahren zur Bestimmung einer Konstanten K_2, für die

$$||\tilde{\mathcal{L}}^{-1}|| \leq K_2$$

gilt. Ist das Verfahren erfolgreich durchführbar, so ist automatisch auch die Invertierbarkeit von $\tilde{\mathcal{L}}$ gewährleistet. Bricht umgekehrt das Verfahren ab, so liegt das daran, dass $\tilde{\mathcal{L}}$ nicht invertierbar ist oder das Verfahren aufgrund von Rundungsfehlern und Einschließungsüberschätzungen die Invertierbarkeit nicht nachweisen kann. In beiden Fällen bestimmen wir gemäß Abschnitt 2.5.1 eine genauere Approximation $\tilde{\mathcal{L}}$ und versuchen erneut, mit diesem Verfahren eine Konstante K_2 zu bestimmen.

Wir untersuchen zunächst, welche Gestalt eine Lösung von $\tilde{\mathcal{L}}x = y$, also die Funktion $x = \tilde{\mathcal{L}}^{-1}y$, aufweist. Sei $y \in C([a,b])$ beliebig, fest. Dann gilt

$$\begin{aligned}& \tilde{\mathcal{L}}x = y \iff (\mathcal{I} - \tilde{\mathcal{K}}_m)x = y \\ \iff\quad & x(s) - \int_a^b \sum_{i=0}^m \sum_{j=0}^m \tilde{k}_{i,j} \Phi_{m,i}(s)\Phi_{m,j}(t)x(t)\,dt = y(s), \qquad s \in [a,b] \\ \iff\quad & x(s) - \sum_{i=0}^m \Phi_{m,i}(s) \underbrace{\int_a^b \sum_{j=0}^m \tilde{k}_{i,j}\Phi_{m,j}(t)x(t)\,dt}_{=\,\text{const}\,=:x_{m,i}} = y(s), \qquad s \in [a,b] \\ \iff\quad & x(s) - \sum_{i=0}^m x_{m,i}\Phi_{m,i}(s) = y(s), \qquad s \in [a,b] \\ \iff\quad & x(s) = y(s) + \sum_{i=0}^m x_{m,i}\Phi_{m,i}(s), \qquad s \in [a,b]. \qquad (2.32)\end{aligned}$$

Setzen wir diese Darstellung von x in die Integralgleichung $\tilde{\mathcal{L}}x = y$ ein, so erhalten wir ein lineares Gleichungssystem für die unbekannten Koeffizienten $x_{m,i}$, $i = 0, \ldots, m$:

$$\begin{aligned}
& \tilde{\mathcal{L}}x = y \\
\iff \quad & x(s) - \int_a^b \sum_{i=0}^m \sum_{j=0}^m \tilde{k}_{i,j} \Phi_{m,i}(s) \Phi_{m,j}(t) x(t)\, dt = y(s), \qquad s \in [a,b] \\
\iff \quad & y(s) + \sum_{i=0}^m x_{m,i}\Phi_{m,i}(s) - \int_a^b \sum_{i=0}^m \sum_{j=0}^m \tilde{k}_{i,j} \Phi_{m,i}(s)\Phi_{m,j}(t) \Big[y(t) + \sum_{k=0}^m x_{m,k}\Phi_{m,k}(t) \Big]\, dt \\
& = y(s), \qquad s \in [a,b] \\
\iff \quad & \sum_{i=0}^m x_{m,i}\Phi_{m,i}(s) - \sum_{i=0}^m \Phi_{m,i}(s) \int_a^b \sum_{j=0}^m \tilde{k}_{i,j}\Phi_{m,j}(t) \sum_{k=0}^m x_{m,k}\Phi_{m,k}(t)\, dt \\
& = \sum_{i=0}^m \Phi_{m,i}(s) \int_a^b \sum_{j=0}^m \tilde{k}_{i,j}\Phi_{m,j}(t) y(t)\, dt, \qquad s \in [a,b].
\end{aligned}$$

Da die Hütchenfunktionen $\Phi_{m,i}$, $i = 0, \ldots, m$, linear unabhängig sind, gilt

$$\begin{aligned}
\iff x_{m,i} - \int_a^b \sum_{j=0}^m \tilde{k}_{i,j}\Phi_{m,j}(t) \sum_{k=0}^m x_{m,k}\Phi_{m,k}(t)\, dt &= \int_a^b \sum_{j=0}^m \tilde{k}_{i,j}\Phi_{m,j}(t)y(t)\, dt, \quad i = 0, \ldots, m \\
\iff x_{m,i} - \sum_{j=0}^m \tilde{k}_{i,j} \sum_{k=0}^m \int_a^b \Phi_{m,j}(t)\Phi_{m,k}(t)\, dt\, x_{m,k} &= \sum_{j=0}^m \tilde{k}_{i,j} \int_a^b \Phi_{m,j}(t) y(t)\, dt, \quad i = 0, \ldots, m \\
\iff \qquad (\boldsymbol{I} - \tilde{\boldsymbol{K}}\boldsymbol{\Phi})\boldsymbol{x} &= \tilde{\boldsymbol{K}}\boldsymbol{y}
\end{aligned}$$

mit $(m+1, m+1)$-Matrizen

$$\tilde{\boldsymbol{K}} = (\tilde{k}_{i,j}), \qquad \boldsymbol{\Phi} = \left(\int_a^b \Phi_{m,i}(t)\Phi_{m,j}(t)\, dt \right)$$

und Vektoren

$$\boldsymbol{x} = (x_{m,i}), \qquad \boldsymbol{y} = \left(\int_a^b \Phi_{m,i}(t)y(t)\, dt \right). \tag{2.33}$$

Bemerkung 2.5.3. *Die Einträge der Matrix* $\boldsymbol{\Phi}$ *lassen sich explizit berechnen. Es gilt*

$$\int_a^b \Phi_{m,i}(t)\Phi_{m,j}(t)\, dt = \begin{cases} \frac{1}{3}h, & |i-j| = 0,\ i = 0, \\ \frac{2}{3}h, & |i-j| = 0,\ i = 1, \ldots, m-1, \\ \frac{1}{3}h, & |i-j| = 0,\ i = m, \\ \frac{1}{6}h, & |i-j| = 1, \\ 0 & \textit{sonst.} \end{cases}$$

Die Regularität der Matrix $\boldsymbol{A} := (\boldsymbol{I} - \tilde{\boldsymbol{K}}\boldsymbol{\Phi})$ und die Invertierbarkeit von $\tilde{\mathcal{L}}$ sind somit äquivalent. Falls wir die Regularität von $\boldsymbol{A}$ zeigen können ist die Existenz von $\tilde{\mathcal{L}}^{-1}$ also gesichert. Die Koeffizienten $x_{m,i}$ der Lösung $x = y + \sum_{i=0}^{m} x_{m,i}\Phi_{m,i}$ sind eindeutig bestimmt durch

$$(x_{m,i}) = \boldsymbol{x} = \boldsymbol{A}^{-1}\tilde{\boldsymbol{K}}\boldsymbol{y} = \boldsymbol{B}\boldsymbol{y} \qquad \text{mit} \qquad \boldsymbol{B} := \boldsymbol{A}^{-1}\tilde{\boldsymbol{K}}. \tag{2.34}$$

Da für $\tilde{k} \in C([a,b] \times [a,b])$ auf Grund der Approximationseigenschaft bilinearer Splines der Operator $\tilde{\mathcal{L}} = \mathcal{L}_m$ für $m \to \infty$ in der Operatornorm gegen $\mathcal{L}$ konvergiert und die Menge der invertierbaren Operatoren offen ist, wird, falls $\mathcal{L}$ invertierbar ist, der Operator $\mathcal{L}_m$ für genügend großes m invertierbar sein, d.h. die Matrix $\boldsymbol{A}$ wird regulär sein. Die Regularität der Matrix $\boldsymbol{A}$ können wir beispielsweise mit dem Intervall-Gauß-Algorithmus aus Abschnitt 1.6.1 oder dem Verfahren von Rump aus Abschnitt 1.6.2 nachweisen. Dazu sind die Einträge der Matrix $\boldsymbol{A}$ selbstverständlich in Intervallarithmetik zu berechnen, was voraussetzt, dass $\tilde{k}$ Komposition elementarer, in Intervallarithmetik auswertbarer Funktionen ist. Die Intervallmatrix, welche die Einschließungen der Einträge der Matrix $\boldsymbol{A}$ enthält, bezeichnen wir mit $[\boldsymbol{A}]$. Um die Einträge der Matrix $\boldsymbol{A}^{-1}$ einzuschließen berechnen wir mit dem Intervall-Gauß-Algorithmus oder dem Verfahren von Rump für $i = 0, \ldots, m$ Einschließungen $[\boldsymbol{z}]_i$ der Lösungsmenge von $[\boldsymbol{A}]\boldsymbol{z}_i = \boldsymbol{e}_i$. Die Intervallmatrix $[\boldsymbol{A}^{-1}] := ([\boldsymbol{z}]_0, [\boldsymbol{z}]_1, \ldots, [\boldsymbol{z}]_m)$ ist dann eine Einschließung der Matrix $\boldsymbol{A}^{-1}$. Eine Einschließung $[\boldsymbol{B}]$ der Matrix $\boldsymbol{B}$ ergibt sich durch $[\boldsymbol{B}] := [\boldsymbol{A}^{-1}]\tilde{\boldsymbol{K}}$.

Diese explizite Darstellung der Lösung der Integralgleichung $\tilde{\mathcal{L}}x = y$ verwenden wir nun, um die Norm des inversen Operators zu beschränken:

$$\begin{aligned}
||\tilde{\mathcal{L}}^{-1}|| &= \sup_{y \in C([a,b])\setminus\{0\}} \frac{||\tilde{\mathcal{L}}^{-1}y||}{||y||} = \sup_{\substack{y \in C([a,b])\setminus\{0\} \\ x \text{ aus } (2.32)}} \frac{||x||}{||y||} = \sup_{\substack{y \in C([a,b])\setminus\{0\} \\ x_{m,i} \text{ aus } (2.34)}} \frac{||y + \sum_{i=0}^{m} x_{m,i}\Phi_{m,i}||}{||y||} \\
&\leq \sup_{\substack{y \in C([a,b])\setminus\{0\} \\ x_{m,i} \text{ aus } (2.34)}} \frac{||y|| + ||\sum_{i=0}^{m} x_{m,i}\Phi_{m,i}||}{||y||} = 1 + \sup_{\substack{y \in C([a,b])\setminus\{0\} \\ \boldsymbol{x} \text{ aus } (2.34)}} \frac{||\boldsymbol{x}||}{||y||} \\
&\leq 1 + \sup_{\substack{y \in C([a,b])\setminus\{0\} \\ \boldsymbol{y} \text{ aus } (2.33)}} \frac{||\boldsymbol{B}||\,||\boldsymbol{y}||}{||y||} \\
&= 1 + \sup_{y \in C([a,b])\setminus\{0\}} \frac{||\boldsymbol{B}|| \max_{i=0,\ldots,n} \left| \int_a^b \Phi_{m,i}(t)y(t)\,dt \right|}{||y||} \\
&\leq 1 + \sup_{y \in C([a,b])\setminus\{0\}} \frac{||\boldsymbol{B}|| \max_{i=0,\ldots,n} \int_a^b |\Phi_{m,i}(t)y(t)|\,dt}{||y||}
\end{aligned}$$

$$
\begin{aligned}
&\leq 1 + \sup_{y \in C([a,b]) \setminus \{0\}} \frac{||\boldsymbol{B}|| \max\limits_{i=0,\dots,n} \int_a^b \Phi_{m,i}(t) ||y(t)|| \, dt}{||y||} \\
&= 1 + \sup_{y \in C([a,b]) \setminus \{0\}} \frac{||\boldsymbol{B}|| h ||y||}{||y||} \\
&= 1 + h||\boldsymbol{B}||.
\end{aligned}
$$

Dabei bezeichnen wir mit $||\boldsymbol{B}||$ die Zeilensummennorm

$$||\boldsymbol{B}|| := \max_{i=0,\dots,m} \sum_{j=0}^{m} |b_{i,j}|.$$

Als Konstante K_2 können wir somit

$$K_2 := \overline{1 + h \max_{i=0,\dots,m} \sum_{j=0}^{m} \operatorname{abs}([b]_{i,j})}$$

wählen.

2.5.5 Zusammenfassung

In den vorangegangenen Abschnitten haben wir

$$
\begin{aligned}
K_1 &:= \max_{i=1,\dots,m} \max\{p^1_{m,i}, p^2_{m,i}, p^3_{m,i}\}, \\
K_2 &:= \overline{1 + h \max_{i=0,\dots,m} \sum_{j=0}^{m} \operatorname{abs}([b]_{i,j})}
\end{aligned}
$$

gesetzt. Gilt mit diesen Konstanten

$$K_1 < \frac{1}{K_2}, \tag{2.35}$$

so garantiert Satz 2.5.1 die Existenz des Operators $\mathcal{L}^{-1}$ und liefert die Abschätzung $||\mathcal{L}^{-1}|| \leq K$ mit

$$K := \frac{K_2}{1 - K_1 K_2}.$$

Gilt hingegen (2.35) nicht oder können wir K_2 nicht berechnen, weil wir die Inverse von $\boldsymbol{A}$ nicht einschließen können, so ist mit den hier benutzten Mitteln keine Aussage über die Existenz des Operators $\mathcal{L}^{-1}$ möglich. In diesem Fall approximieren wir $\mathcal{L}$ genauer, etwa mit $\mathcal{L}_{2m}$, und versuchen, K_1 und K_2 erneut zu berechnen. Die Approximation von $\mathcal{L}$ wird so lange verbessert (etwa durch Verdoppelung von m), bis wir Konstanten K_1 und K_2 erhalten, die (2.35) genügen, oder wir erfolglos abbrechen. Da für $\tilde{k} \in C([a,b] \times [a,b])$ der Operator $\mathcal{L}_m$ für $m \to \infty$ gegen $\mathcal{L}$ konvergiert, gibt es für invertierbares $\mathcal{L}$ und hinreichend großes m stets Konstanten K_1 und K_2, die (2.35) erfüllen. Da unsere Berechnungen aber nur mit begrenzter Rechengenauigkeit durchgeführt werden, können wir diese eventuell dennoch nicht berechnen.

2.5.6 Parallelisierung

Für die Parallelisierung der Berechnung der Konstanten

$$K_1 := \max_{i=1,\dots,m} \max\{p_{m,i}^1, p_{m,i}^2, p_{m,i}^3\}$$

bietet sich ein einfaches Vorgehen an, denn die Werte

$$\max\{p_{m,i}^1, p_{m,i}^2, p_{m,i}^3\}, \qquad i = 1, \dots, m$$

lassen sich unabhängig voneinander berechnen. Die Berechnung der einzelnen Werte wird vorab statisch oder während der Berechnung dynamisch mit Hilfe des Master-Slave-Verfahrens (siehe Abschnitt 2.4.1) oder des Lastverteilungsverfahrens mit zufälligem Anfragen (siehe Abschnitt 2.4.1) auf die verfügbaren Prozessoren verteilt. Die Konstante K_1 ergibt sich dann als globales Maximum über diese Werte.

Die Parallelisierung der Berechnung einer Konstanten K_2 ist etwas schwieriger. Zur Einschließung der Matrix $\boldsymbol{A}^{-1}$ wollen wir bei unserer Parallelisierung das Verfahren von Rump (siehe Abschnitt 1.6.2) verwenden, da dieses im Allgemeinen kleinere Fehlerschranken liefert als der Intervall-Gauß-Algorithmus. Die beiden Hauptaufgaben des in Algorithmus 1.6.5 beschriebenen Verfahrens von Rump sind zum einen die Invertierung der reellen Matrix $\mathrm{mid}([\boldsymbol{A}])$ und zum anderen viele Matrix-Vektor-Multiplikationen.

Die Invertierung der Matrix $\mathrm{mid}([\boldsymbol{A}])$ führen wir in zwei Schritten durch. Zuerst berechnen wir mit dem parallelisierten Algorithmus 2.3.16 eine LU-Faktorisierung der Matrix $\mathrm{mid}([\boldsymbol{A}])$. Mit dieser Faktorisierung lösen wir dann für $i = 0, \dots, m$ die Gleichungssysteme $\mathrm{mid}([\boldsymbol{A}])\boldsymbol{z}_i = \boldsymbol{e}_i$. Die zu berechnende Matrix $\boldsymbol{C} = (\mathrm{mid}\,[\boldsymbol{A}])^{-1}$ ist dann gegeben durch $\boldsymbol{C} = (\boldsymbol{z}_0, \boldsymbol{z}_1, \dots, \boldsymbol{z}_m)$. Die einzelnen zu lösenden Gleichungssystem werden dabei statisch oder dynamisch auf die zur Verfügung stehenden Prozessoren verteilt.

Die im Verfahren von Rump notwendigen Matrix-Vektor-Multiplikationen können wir bei einer Unterteilung der Matrix in Blockspalten gemäß Algorithmus 2.3.11 oder bei einer Unterteilung in Blockzeilen nach Algorithmus 2.3.12 parallelisieren.

Um die Matrix-Matrix-Multiplikation $[\boldsymbol{A}^{-1}]\tilde{\boldsymbol{K}}$ zu parallelisieren, helfen uns die gleichen Ideen wie bei der Parallelisierung der Matrix-Vektor-Multiplikation weiter.

Abbildung 2.10 zeigt die benötigte Rechenzeit des Programms zur Berechnung einer oberen Schranke K für die Norm der Inversen des linearen Operators $\boldsymbol{\mathcal{L}}$ für eine unterschiedliche Anzahl von Teilintervallen m und Prozessoren. Außer für die kleinste Problemgröße skaliert der Algorithmus sehr gut. Bei der kleinsten Problemgröße $m = 100$ skaliert der Algorithmus nur bis ca. 8 Prozessoren, danach tritt kaum noch eine Rechenzeitverkürzung ein. Der Rechenzeitbedarf bei dieser Problemgröße liegt aber sowieso im Bereich weniger Sekunden, so dass eine Parallelisierung nicht sinnvoll ist. Der Hauptrechenaufwand des Algorithmus ist durch das Verfahren von Rump verursacht, das ein $O(m^3)$-Algorithmus ist. Dies spiegelt sich auch in den gemessenen Rechenzeiten wieder. Eine Verdoppelung der Problemgröße führt zu einer Verachtfachung des Rechenzeitbedarfs.

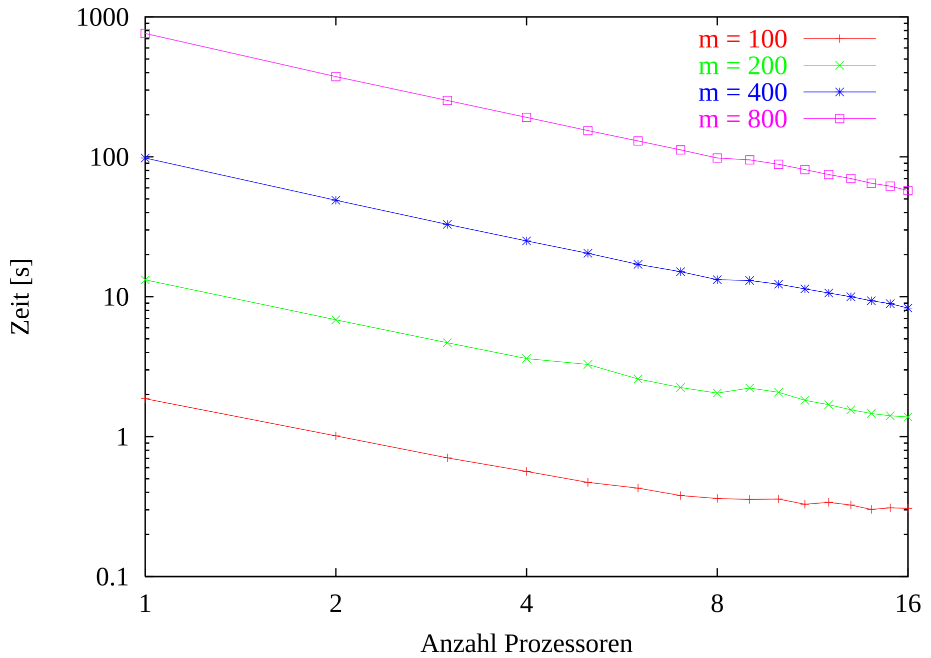


Abb. 2.10: Benötigte Rechenzeit des Programms zur Berechnung einer oberen Schranke K für die Norm der Inversen des linearen Operators $\mathcal{L}$ für eine unterschiedliche Anzahl von Teilintervallen m und Prozessoren. Die Berechnungen wurden auf einem Cluster mit 800 MHz Pentium III Prozessoren und Gigabit-Ethernet durchgeführt.

2.6 Bestimmung einer Funktion G

In diesem Abschnitt beschreiben wir, wie eine monoton wachsende Funktion $G : \mathbb{R}_+ \to \mathbb{R}_+$ bestimmt werden kann, die die Abschätzung

$$\left|\left| \int_a^b \left(k_3(.,t,\omega(t))x(t) - \left[k(.,t,\omega(t)+x(t)) - k(.,t,\omega(t))\right]\right) dt \right|\right| \leq G(\alpha) \qquad (2.36)$$

für alle $x \in C([a,b])$ mit $||x|| \leq \alpha$ erfüllt, und für die weiterhin $G(\alpha) = o(\alpha)$ für $\alpha \to 0$ gilt.

Die Näherungslösung ω liege in der speziellen Form

$$\omega(s) = y(s) + \sum_{j=1}^{n} w_{n,j} k(s, t_{n,j}, \omega_{n,j})$$

mit gegebenen Quadraturknoten $t_{n,j}$, $j = 1, \ldots, n$, und Quadraturgewichten $w_{n,j}$, $j = 1, \ldots, n$, vor (vergleiche Abschnitt 2.3.4).

Als erstes berechnen wir mit Hilfe der Intervallrechnung eine Einschließung für ω. Dazu unterteilen wir das Intervall $[a,b]$ in m gleichgroße Intervalle $[s]_{m,i}$, $i = 1, \ldots, m$, mit

Durchmesser $h = (b-a)/m$:

$$[s]_{m,i} := [a + (i-1)h, a + ih], \qquad i = 1, \ldots, m.$$

Um die Schreibweise zu vereinfachen legen wir $[t]_{m,i} := [s]_{m,i}$ für $i = 1, \ldots, m$ fest. Mit Hilfe der Intervallrechnung können wir durch die Intervalle

$$[\omega]_{m,i} := y([s]_{m,i}) + \sum_{j=1}^{n} w_{n,j} k([s]_{m,i}, t_{n,j},\ \omega_{n,j}), \qquad i = 1, \ldots, m$$

die Funktionswerte $\omega(s)$ für $s \in [s]_{m,i}$ einschließen.

Mit dem Mittelwertsatz in integraler Form formen wir den Integranden aus Gleichung (2.36) um:

$$k_3(s,t,\omega(t))x(t) - \big[k(s,t,\omega(t)+x(t)) - k(s,t,\omega(t))\big] = \\ \int_0^1 \big[k_3(s,t,\omega(t)) - k_3(s,t,\omega(t)+\theta x(t))\big]\, d\theta\, x(t).$$

Für die linke Seite in Ungleichung (2.36) gilt somit:

$$\begin{aligned}
&\Big|\Big| \int_a^b \big(k_3(.,t,\omega(t))x(t) - \big[k(.,t,\omega(t)+x(t)) - k(.,t,\omega(t))\big]\big)\, dt \Big|\Big| \\
&\quad = \Big|\Big| \int_a^b \int_0^1 \big[k_3(.,t,\omega(t)) - k_3(.,t,\omega(t)+\theta x(t))\big]\, d\theta\, x(t)\, dt \Big|\Big| \\
&\quad = \max_{s\in[a,b]} \Big| \int_a^b \int_0^1 \big[k_3(s,t,\omega(t)) - k_3(s,t,\omega(t)+\theta x(t))\big]\, d\theta\, x(t)\, dt \Big| \\
&\quad = \max_{i=1,\ldots,m} \max_{s\in[s]_{m,i}} \Big| \sum_{j=1}^{m} \int_{[t]_{m,j}} \int_0^1 \big[k_3(s,t,\omega(t)) - k_3(s,t,\omega(t)+\theta x(t))\big]\, d\theta\, x(t)\, dt \Big| \\
&\quad \le \max_{i=1,\ldots,m} \max_{s\in[s]_{m,i}} \sum_{j=1}^{m} \int_{[t]_{m,j}} \int_0^1 |k_3(s,t,\omega(t)) - k_3(s,t,\omega(t)+\theta x(t))|\, d\theta\, \underbrace{|x(t)|}_{\le\alpha}\, dt \\
&\quad \le \alpha \max_{i=1,\ldots,m} \max_{s\in[s]_{m,i}} \sum_{j=1}^{m} \int_{[t]_{m,j}} \int_0^1 |k_3(s,t,\omega(t)) - k_3(s,t,\omega(t)+\theta x(t))|\, d\theta\, dt.
\end{aligned}$$

Wir schließen nun den Ausdruck

$$|k_3(s,t,\omega(t)) - k_3(s,t,\xi)|, \qquad s \in [s]_{m,i},\ t \in [t]_{m,j},\ \xi \in \omega(t) + [-\alpha, \alpha]$$

möglichst eng ein. Bei der Einschließung dieses Ausdruckes mittels gewöhnlicher Intervallarithmetik kann es zu einer großen Überschätzung kommen, da die beiden Terme $k_3(s,t,\omega(t))$ und $k_3(s,t,\xi)$ von gleicher Größenordnung sind. Daher wollen wir im Folgenden stattdessen Steigungsarithmetik verwenden, um eine bessere Einschließung zu erhalten. Mit unserer eingeschlossenen Näherungslösung definieren wir ein Steigungstripel für die Funktion $f(x) = x$ (siehe Bemerkung 1.5.4):

$$\langle x \rangle_{m,j} := \begin{pmatrix} [\omega]_{m,j} + [-\alpha, \alpha] \\ [\omega]_{m,j} \\ 1 \end{pmatrix}, \qquad j = 1, \ldots, m.$$

Damit werten wir k_3 in Steigungsarithmetik aus:

$$\langle g \rangle_{i,j} := k_3([s]_{m,i}, [t]_{m,j}, \langle x \rangle_{m,j}), \qquad i, j = 1, \ldots, m.$$

Nach Definition des Steigungstripels erhalten wir damit die Einschließung

$$k_3(s, t, \omega(t)) - k_3(s, t, \xi) \in [g]^s_{i,j}(\omega(t) - \xi)$$

für alle $s \in [s]_{m,i}$, $t \in [t]_{m,j}$, $\xi \in \omega(t) + [-\alpha, \alpha]$. Der Betrag dieser Differenz wird somit eingeschlossen durch

$$|k_3(s, t, \omega(t)) - k_3(s, t, \xi)| \in \operatorname{abs}([g]^s_{i,j})|\omega(t) - \xi|,$$

was uns die obere Schranke

$$|k_3(s, t, \omega(t)) - k_3(s, t, \xi)| \leq \overline{\operatorname{abs}([g]^s_{i,j})}\alpha$$

für alle $s \in [s]_{m,i}$, $t \in [t]_{m,j}$, $\xi \in \omega(t) + [-\alpha, \alpha]$ liefert. Zuletzt schätzen wir durch die Verwendung von Riemannschen Obersummen ab und erhalten:

$$\begin{aligned}
&\left|\left| \int_a^b \big(k_3(., t, \omega(t))x(t) - \big[k(., t, \omega(t) + x(t)) - k(., t, \omega(t))\big]\big)\, dt \right|\right| \\
&\quad \leq \alpha \max_{i=1,\ldots,m} \max_{s \in [s]_{m,i}} \sum_{j=1}^m \int_{[t]_{m,j}} \int_0^1 |k_3(s, t, \omega(t)) - k_3(s, t, \omega(t) + \theta x(t))|\, d\theta\, dt \\
&\quad \leq \alpha \max_{i=1,\ldots,m} \max_{s \in [s]_{m,i}} \sum_{j=1}^m \int_{[t]_{m,j}} \overline{\operatorname{abs}([g]^s_{i,j})}\alpha\, dt \\
&\quad \leq \alpha \max_{i=1,\ldots,m} \sum_{j=1}^m h\overline{\operatorname{abs}([g]^s_{i,j})}\alpha \\
&\quad = \alpha^2 h \max_{i=1,\ldots,m} \sum_{j=1}^m \overline{\operatorname{abs}([g]^s_{i,j})}
\end{aligned}$$

mit von α abhängenden Werten $\overline{\operatorname{abs}([g]^s_{i,j})}$, $1 \leq i, j \leq m$. Definieren wir nun

$$G(\alpha) := \alpha^2 h \max_{i=1,\ldots,m} \sum_{j=1}^m \overline{\operatorname{abs}([g]^s_{i,j})},$$

so gilt offenbar Ungleichung (2.36) und $G(\alpha) = o(\alpha)$ für $\alpha \to 0$.

2.6.1 Parallelisierung

Das Programm zur Berechnung einer monoton wachsenden Funktion G kann analog zu dem Vorgehen zur Parallelisierung des Programms zur Berechnung einer oberen Schranke δ für den Defekt in Abschnitt 2.4.1 parallelisiert werden.

Das Gesamtproblem wird wieder in viele unabhängig voneinander berechenbare Teilprobleme zerlegt. Solche Teilprobleme sind hier beispielsweise die Einschließung der Näherungslösung in Intervalle $[\omega]_{m,i}$, $i = 1, \dots m$, bzw. die Berechnung von oberen Schranken $\sum_{j=1}^{m} \overline{\text{abs}([g]_{i,j}^{s})}$, $i = 1, \dots, m$.

Die dynamische Lastverteilung kann entweder mit dem Master-Slave-Verfahren (vgl. Algorithmus 2.4.1) oder dem Lastverteilungsverfahren mit zufälligem Anfragen (vgl. Algorithmus 2.4.2) erfolgen.

Abbildung 2.11 zeigt die benötigte Rechenzeit des Programms zur Berechnung einer monoton wachsenden Funktion $G(\alpha)$ für eine unterschiedliche Anzahl von Teilintervallen m und Prozessoren. Die Lastverteilung erfolgte mittels Algorithmus 2.4.2. Wie schon bei der Berechnung des Defektes (siehe Abbildung 2.8) skaliert dieser Algorithmus sehr gut. Eine Verdoppelung der Problemgröße führt zu einer Vervierfachung des Rechenzeitbedarfs, der Algorithmus ist also ein $O(m^2)$-Algorithmus.

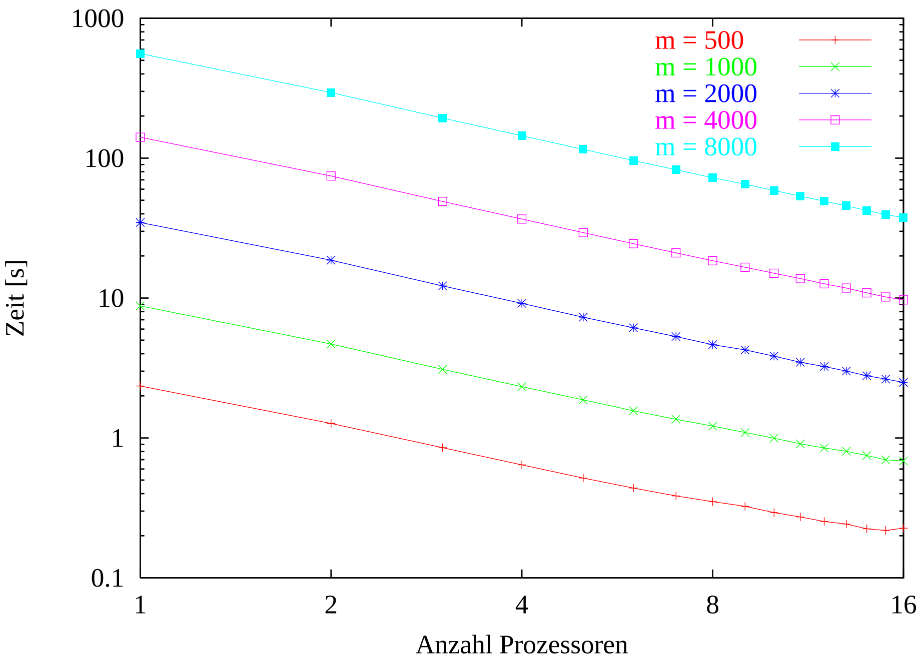


Abb. 2.11: Benötigte Rechenzeit des Programms zur Berechnung einer monoton wachsenden Funktion $G(\alpha)$ für eine unterschiedliche Anzahl von Teilintervallen m und Prozessoren. Die Lastverteilung erfolgte mittels Algorithmus 2.4.2. Die Berechnungen wurden auf einem Cluster mit 800 MHz Pentium III Prozessoren und Gigabit-Ethernet durchgeführt.

Da die benötigte Rechenzeit bei Verwendung des Master-Slave-Verfahrens zur Lastverteilung (Algorithmus 2.4.1) generell höher war als mit dem Lastverteilungsverfahren mit zufälligem Anfragen und da die einzelnen Kurven sehr ähnlich zu den Kurven in Abbildung 2.7 verlaufen, wird dieses Diagramm hier nicht wiedergegeben.

2.7 Bestimmung einer Konstanten α

In diesem Abschnitt beschäftigen wir uns mit dem Problem, eine Konstante α zu finden, so dass

$$\delta \leq \frac{\alpha}{K} - G(\alpha) \tag{2.37}$$

gilt. Abbildung 2.12 gibt den typischen Verlauf der Funktion $\alpha/K - G(\alpha)$ wieder.

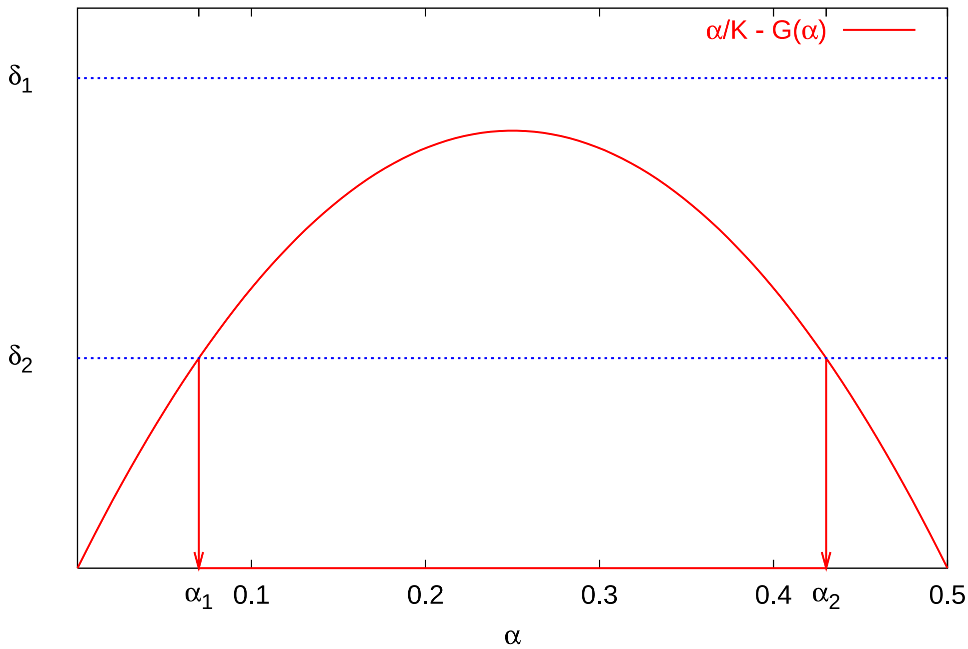


Abb. 2.12: Typischer Verlauf der Funktion $\alpha/K - G(\alpha)$. Mit der Defektschranke δ_1 ist keine Einschließung möglich. Mit der Defektschranke δ_2 gelingt die Einschließung für alle $\alpha \in [\alpha_1, \alpha_2]$. Für die praktische Einschließung sind wir an α_1 interessiert.

An diesem Schaubild erkennen wir, dass für die Defektschranke δ_1 kein α existiert, so dass Ungleichung (2.37) erfüllt ist. Für δ_2 existiert jedoch ein Intervall $[\alpha_1, \alpha_2]$, so dass für jedes $\alpha \in [\alpha_1, \alpha_2]$ Ungleichung (2.37) gilt. Falls wir also δ_1 als Defektschranke berechnet haben, können wir die Existenz einer Lösung nicht nachweisen, obwohl diese sehr wohl existieren kann. Erhalten wir hingegen δ_2 als Defektschranke, so genügt es, ein (möglichst kleines) $\alpha \in [\alpha_1, \alpha_2]$ anzugeben, um die Existenz der Lösung nachzuweisen und diese mit dem Fehler α einzuschließen.

Gelingt es nicht, für eine berechnete Defektschranke δ eine Konstante α zu bestimmen, so dass Ungleichung (2.37) erfüllt ist, so können wir versuchen, die Defektschranke zu verbessern. Dies ist oft möglich, da die berechnete Näherungslösung ω häufig in Wahrheit einen viel kleineren Defekt besitzt als den mit unserem mit vielen Überschätzungen behafteten Berechnungsverfahren eingeschlossenen. Üblicherweise führt eine doppelt so feine Intervallunterteilung bei der Berechnung der Defektschranke zu einer vier mal kleineren Defektschranke δ.

Trotz dieser Überlegungen haben wir bis jetzt keine Methode, um bei bekannter Defektschranke δ und Operatornormschranke K überhaupt ein α angeben zu können oder sogar ein möglichst kleines α wählen zu können. Dazu verwenden wir die folgende Heuristik:

Die Funktion $G(\alpha)$ in Ungleichung (2.37) können wir nicht nach α auflösen. Aus Abschnitt 2.6 wissen wir aber, dass

$$G(\alpha) \approx c_0\alpha^2$$

für ein $c_0 \in \mathbb{R}$ gilt. Setzen wir

$$c_0 := \frac{G(\alpha_0)}{\alpha_0^2}$$

für ein hinreichend kleines $\alpha_0 \in \mathbb{R}_+$, so lässt sich Ungleichung (2.37) approximieren durch die Ungleichung

$$\delta \leq \frac{\alpha}{K} - c_0\alpha^2. \tag{2.38}$$

Falls nun $1 - 4c_0K^2\delta > 0$ ist, so ist diese Ungleichung für alle $\alpha \in [\underline{\alpha}, \overline{\alpha}]$ mit

$$\underline{\alpha} := \frac{1}{2c_0K}\left(1 - \sqrt{1 - 4c_0K^2\delta}\right), \qquad \overline{\alpha} := \frac{1}{2c_0K}\left(1 + \sqrt{1 - 4c_0K^2\delta}\right)$$

erfüllt. Für $\alpha \approx \underline{\alpha}$ überprüfen wir nun, ob Ungleichung (2.37) erfüllt ist und wir somit die Existenz einer Lösung der Integralgleichung nachweisen konnten.

Falls hingegen $1-4c_0K^2\delta < 0$ ist, gibt es kein α, welches die approximierende Ungleichung (2.38) erfüllt. In diesem Fall müssen wir daher zuerst eine kleinere Defektschranke $\hat{\delta}$ berechnen. Aus der Forderung $1 - 4c_0K^2\hat{\delta} > 0$ ergibt sich $\hat{\delta} < 1/(4c_0K^2)$. Hat die berechnete Näherungslösung einen kleineren Defekt als $1/(4c_0K^2)$, so können wir durch Erhöhen der Anzahl von Teilintervallen bei der Defektberechnung die Defektschranke verbessern. Eine Verdoppelung der Teilintervalle führt bei der Verwendung der zentrierten Form mit Steigung üblicherweise zu einer vier mal kleineren Defektschranke. Wenn m bzw. $\hat{m}$ die Anzahl von Teilintervallen bei der Berechnung von δ bzw. $\hat{\delta}$ bezeichnen, sollten wir $\hat{m}$ also gemäß

$$\frac{\hat{m}}{m} \approx \sqrt{\frac{\delta}{\hat{\delta}}} > \sqrt{4c_0K^2\delta}$$

wählen und wie in dieser Heuristik beschrieben erneut ein α suchen, so dass Ungleichung (2.37) erfüllt ist.

Alternativ zu dieser Heuristik besteht auch die Möglichkeit, eine gewünschte Fehlerschranke α vorzugeben. Wir benötigen dann eine Methode um vorherzusagen, wie fein die Intervallunterteilung bei der Berechnung der Defektschranke δ zu wählen ist, um diese Fehlerschranke zu erreichen. Wir gehen wie folgt vor:

Für das gegebene α berechnen wir $G(\alpha)$. Die Operatornormschranke K, die wir ebenfalls berechnen, ist unabhängig von α. Ist $\alpha/K - G(\alpha) < 0$, so gibt es kein δ, so dass Ungleichung (2.37) erfüllt ist, wir können also keine Einschließung mit einem Fehler α gewinnen. Die von uns gewünschte Fehlerschranke α liegt somit rechts des in Abbildung 2.12 dargestellten Bereiches. Wollen wir dennoch eine Einschließung der Lösung der Integralgleichung gewinnen, so müssen wir ein kleineres $\hat{\alpha}$ wählen, für das

$$\frac{\hat{\alpha}}{K} - G(\hat{\alpha}) > 0$$

ist. Da die Funktion $G(\hat{\alpha})$ nicht formelmäßig gegeben ist, können wir diese Ungleichung wieder nicht nach $\hat{\alpha}$ auflösen. Wir verwenden wie schon zuvor die Approximation $G(\hat{\alpha}) \approx \frac{G(\alpha)}{\alpha^2}\hat{\alpha}^2$ und erhalten damit die approximierende Ungleichung

$$\frac{\hat{\alpha}}{K} - \frac{G(\alpha)}{\alpha^2}\hat{\alpha}^2 > 0,$$

die für

$$0 < \hat{\alpha} < \frac{\alpha^2}{G(\alpha)K}$$

erfüllt ist. Für dieses $\hat{\alpha}$ überprüfen wir erneut, ob $\hat{\alpha}/K - G(\hat{\alpha}) > 0$ ist.

Ist hingegen $\alpha/K - G(\alpha) > 0$, so wollen wir vorhersagen, mit wie vielen Teilintervallen m die Defektschranke δ_m zu berechnen ist, um eine Einschließung mit Fehler α zu erhalten. Dazu berechnen wir für ein beliebiges $m_0 \in \mathbb{N}$ die Defektschranke δ_{m_0} mit einer Unterteilung in m_0 Teilintervalle. Gilt

$$\delta_{m_0} \leq \frac{\alpha}{K} - G(\alpha),$$

so sind wir fertig, denn wir konnten die Lösung der Integralgleichung mit einem Fehler α einschließen. Ist $\delta_{m_0} > \alpha/K - G(\alpha)$ und der tatsächliche Defekt unserer Näherungslösung ω kleiner als $\alpha/K - G(\alpha)$, so können wir durch eine größere Anzahl m von Teilintervallen die Defektschranke δ_m verbessern. Da das Verfahren zur Berechnung der Defektschranke quadratisch konvergiert, erhalten wir

$$\frac{m}{m_0} \approx \sqrt{\frac{\delta_{m_0}}{\delta_m}} > \sqrt{\frac{\delta_{m_0}}{\alpha/K - G(\alpha)}}.$$

Erneut prüfen wir, ob für δ_m Ungleichung (2.37) gilt, und wiederholen gegebenenfalls diese Heuristik.

Wenn der linearisierte Operator $\mathcal{L}$ fast singulär ist, können bei der Einschließung der Lösung Probleme auftreten, denn wir erhalten dann eine sehr große Schranke K für $||\mathcal{L}^{-1}||$ und müssen aufgrund des flachen Verlaufs der Kurve in Abbildung 2.12 eine besonders kleine Defektschranke δ bestimmen.

Um die Existenz einer Lösung der Integralgleichung nachzuweisen und eine Einschließung dieser Lösung zu berechnen, benötigen wir somit eine hinreichend genaue Näherungslösung ω und eine genügend kleine obere Schranke δ für den Defekt. Dies scheint eine natürliche Forderung zu sein.

Kapitel 3

Beispiele

In diesem Kapitel stellen wir zunächst die entwickelte Software vor und erläutern anhand einiger Beispiele ihre Funktionsweise.

3.1 Software

Die einzelnen Schritte des Einschließungsverfahrens aus Kapitel 2 haben wir in einem Softwarepaket realisiert. Um die Bedienung dieses Pakets möglichst komfortabel und intuitiv zu gestalten, haben wir eine grafische Oberfläche integriert (siehe Abbildung 3.1). Diese Oberfläche fungiert als Steuerzentrale für die anderen Programmteile und implementiert auch deren gegenseitige Abhängigkeiten.

Um eine möglichst gute Portabilität des Softwarepakets zu erreichen, erfolgt eine strikte Trennung zwischen der graphischen Oberfläche und den Programmteilen, die Berechnungen durchführen. Die graphische Oberfläche ist in der Programmiersprache Java implementiert, die auf vielen Rechnerplattformen verfügbar ist. Die Programmteile, welche keine Intervallarithmetik benötigen, sind in der Programmiersprache C verwirklicht, diejenigen, die sie benötigen, in der Programmiersprache C++. Um Intervallarithmetik zu nutzen, steht beispielsweise die Bibliothek `C-XSC` (siehe Klatte et al. [22] oder Hofschuster et al. [20]) zur Verfügung. Wir verwenden im Softwarepaket die Intervallarithmetik aus der auf Rechengeschwindigkeit optimierten Bibliothek `filib++` (siehe Lerch et al. [26]), die allerdings nicht immer die kleinstmögliche Einschließung liefert.

Sämtliche rechenintensiven Programmteile liegen in einer seriellen und in einer parallelen Variante vor. Beide Varianten sind mit dem gleichen Namen bezeichnet, liegen aber in den unterschiedlichen Verzeichnissen `serial` bzw. `parallel`. Die Parallelisierung der Programmteile erfolgt mit der Bibliothek `MPI` (Message Passing Interface) (siehe [29], [30]). Diese Standardbibliothek wird von vielen Herstellern unterstützt, so dass eine gute Portabilität, bei gleichzeitig hoher Effizienz, garantiert ist. Die Bibliothek erfordert eine explizite Programmierung der Parallelität (vgl. Abschnitt 1.8), d. h. die Daten werden vom Programmierer auf die einzelnen Prozesse verteilt und der Datenaustausch erfolgt durch explizites Versenden von Nachrichten zwischen diesen Prozessen. Bei der Parallelisierung mittels MPI wird ein einziges Programm mehrfach gestartet (SPMD - Single Program Multiple Data). Damit dieses Programm auf den einzelnen Prozessoren unterschiedliche Aufgaben erfüllt, ist der Programmablauf abhängig von einer Prozessnummer, die bei jedem Aufruf des Programms unterschiedlich ist. Eine Einführung in die Benutzung der MPI-Bibliothek ist beispielsweise in MacDonald et al. [27], Pacheco [34] oder Sanders und Worsch [43] zu finden. In [29] und [30] wird die Spezifikation der Bibliothek beschrieben.

Im Folgenden beschreiben wir zuerst die einzelnen Berechnungsprogramme des Softwarepakets und danach die Verwendung der graphischen Oberfläche. Die Programme sind jeweils so gestaltet, dass sie auch direkt von der Textkonsole, also ohne Verwendung einer graphischen Oberfläche, aufgerufen werden können. Dies ist insbesondere dann wichtig, wenn die Berechnung auf einem Parallelrechner erfolgen soll, der keine graphische Oberfläche unterstützt.

Das Programm omega berechnet eine Näherungslösung ω der Fredholmschen Integralgleichung zweiter Art vom Urysohntyp nach dem in Abschnitt 2.3 beschriebenen Verfahren. Als Parameter werden dem Programm die Anzahl der Quadraturknoten n und das zu verwendende Quadraturverfahren übergeben. Als Quadraturverfahren stehen die Trapezsumme, die Simpson-Summe und das Gaußsche Quadraturverfahren zur Verfügung. Um die Näherungslösung ω graphisch, beispielsweise mit dem Programm gnuplot, ausgeben zu können, schreibt das Programm die Werte $t_{n,i}$ und $\omega(t_{n,i})$ für $i = 1, \ldots, n$ in die Datei <>_omega.txt. Des weiteren erzeugt es die Datei <>_omega.dat, die die Näherungslösung in einem binären Format enthält, welches für die anderen Programmteile besser geeignet ist.

Das Programm defekt berechnet eine obere Schranke δ für den Defekt der Integralgleichung für die Näherungslösung ω. Dazu benötigt es die Datei <>_omega.dat und die Anzahl m der Teilintervalle als Eingabe. Das Ergebnis schreibt es in menschenlesbarer Form in die Datei <>_delta.txt und binär in die Datei <>_delta.dat.

Das Programm const_k berechnet eine obere Schranke K für die Norm der Inversen des am Punkt ω linearisierten Operators $\mathcal{I} - \mathcal{K}$. Als Eingangsparameter benötigt das Programm m, die Anzahl der zu verwendenden Teilintervalle, und ebenfalls die Datei <>_omega.dat. Die Konstante K wird in die Dateien <>_const_k.txt bzw. <>_const_k.dat geschrieben.

Das Programm func_g berechnet eine monoton wachsende Funktion $G : \mathbb{R}_+ \to \mathbb{R}_+$ mit den Eigenschaften $||\mathcal{G}x|| \leq G(||x||)$ und $G(\alpha) = o(\alpha)$, $\alpha \to 0$. Als Eingabe benötigt es die Anzahl m von Teilintervallen, die bei der Berechnung verwendet werden sollen, und die Stelle α, für die der Funktionswert $G(\alpha)$ berechnet werden soll. Auch hier wird die Näherungslösung aus der Datei <>_omega.dat gelesen. Der Funktionswert $G(\alpha)$ wird in die Dateien <>_func_g.txt bzw. <>_func_g.dat gespeichert.

Im letzten Schritt verifiziert das Programm alpha, ob die Ungleichung

$$\delta \leq \frac{\alpha}{K} - G(\alpha)$$

für die zuvor berechneten Konstanten erfüllt ist. Ist sie erfüllt, so existiert eine Lösung der Integralgleichung in einer α-Umgebung um die Näherungslösung ω. Ist die Ungleichung

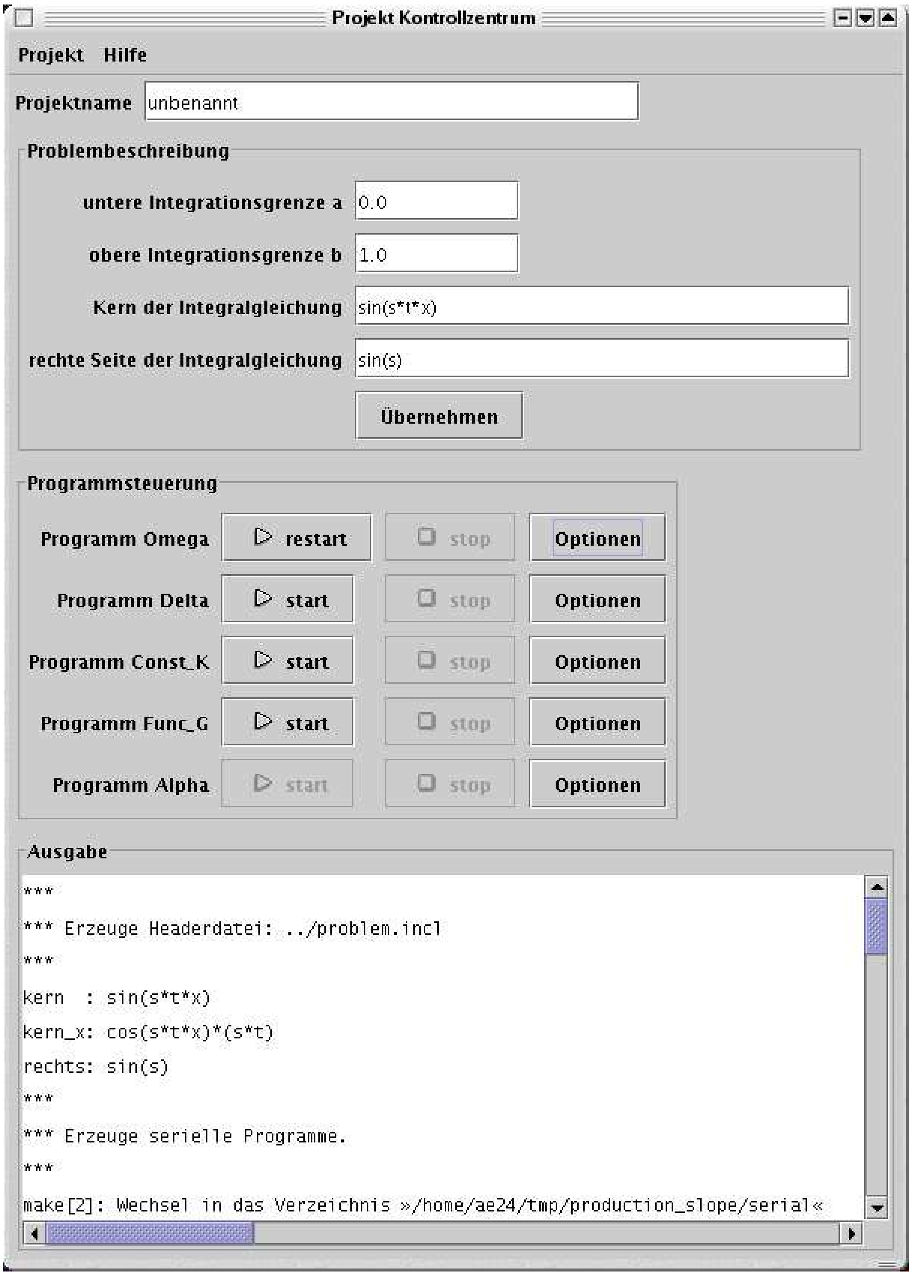


Abb. 3.1: Steuerzentrale `project_control` des Programmpakets zum Existenznachweis und zur Einschließung von Lösungen der nichtlinearen Fredholmschen Integralgleichung zweiter Art.

nicht erfüllt, so gibt das Programme Hinweise, wie vorgegangen werden kann, um eventuell doch noch eine Einschließung berechnen zu können (vgl. Abschnitt 2.7). Als Eingabe für das Programm `alpha` wird ausschließlich der Wert α benötigt. Die berechnete Defektschranke δ wird aus der Datei `<>_delta.dat`, die berechnete Konstante K aus der Datei `<>_const_k.dat` und der berechnete Funktionswert von G aus der Datei `<>_func_g.dat` gelesen.

In der nachfolgenden Tabelle haben wir zur Übersicht die benötigten Eingaben und die erzeugten Ausgaben der einzelnen Teilprogramme zusammengestellt.

Programm	Parameter	Eingabedatei	Ausgabedatei
`omega`	$m \hat{=}$ # Quadraturknoten Quadraturverfahren	-	`<>_omega.txt` `<>_omega.dat`
`defekt`	$m \hat{=}$ # Teilintervalle	`<>_omega.dat`	`<>_delta.txt` `<>_delta.dat`
`const_k`	$m \hat{=}$ # Teilintervalle	`<>_omega.dat`	`<>_const_k.txt` `<>_const_k.dat`
`func_g`	$m \hat{=}$ # Teilintervalle α	`<>_omega.dat`	`<>_func_g.txt` `<>_func_g.dat`
`alpha`	α	`<>_delta.dat` `<>_const_k.dat` `<>_func_g.dat`	-

Die vorgestellten Programme können zentral über die graphische Oberfläche `project_control` (siehe Abbildung 3.1) gesteuert werden. Für jedes Programm gibt es einen Knopf `start`, um es zu starten, einen Knopf `stop`, um die Berechnungen zu unterbrechen und einen Knopf `Optionen`, um die Aufrufparameter einzustellen. Bei den parallelisierten Programmen `omega`, `defekt`, `const_k` und `func_g` kann in den Optionen zusätzlich die Anzahl der zu verwendenden Prozessoren angegeben werden. Bei den Programmen `defekt` und `func_g` kann der Benutzer auch zwischen den Lastverteilungsverfahren durch Master-Slave-Verfahren und durch zufälliges Anfragen auswählen (vgl. Abschnitt 2.4.1). Die Knöpfe zum Starten der Programme haben wir so gestaltet, dass ein Aufruf des Programms nur möglich ist, wenn die notwendigen Abhängigkeiten erfüllt sind. Beispielsweise kann das Programm `defekt` erst dann aufgerufen werden, wenn bereits eine Näherungslösung ω berechnet wurde.

Die Ausgabe der Teilprogramme und die Informationen über die durchgeführten Schritte sind zentral im großen Ausgabefenster im unteren Bereich der graphischen Oberfläche zu finden.

Die Eingabe der Parameter der zu lösenden Integralgleichung

$$x(s) - \int_a^b k(s,t,x(t))\,dt = y(s), \qquad s \in [a,b],$$

also der Integrationsgrenzen a und b, des Kerns k und der rechten Seite y, erfolgt im oberen Bereich der graphischen Oberfläche. Durch Drücken des Knopfes `Übernehmen` erzeugt die Oberfläche aus dieser Eingabe die Headerdatei `problem.incl`. Da die Ableitung k_3 von einigen Teilprogrammen benötigt wird, wird sie hier durch symbolisches Differenzieren berechnet und ebenfalls in `problem.incl` abgespeichert. Die einzelnen Teilprogramme binden dann diese Datei beim Übersetzen ein, um die Berechnungen durchführen zu können.

Alle Einstellungen und Programmausgaben können über den Menüpunkt `Speichern` im Menü `Projekt` gespeichert werden. Mit dem Menüpunkt `Laden` können diese Daten dann wieder eingelesen werden. Der Menüpunkt `Drucken` ermöglicht den Ausdruck des Ausgabefensters. Im Menü `Hilfe` finden wir eine kurze Anleitung zur Bedienung der graphischen Oberfläche.

Wir beschreiben nun kurz die grundlegenden Ideen, die beim symbolischen Differenzieren benötigt werden. Dazu zeigen wir in Algorithmus 3.1.1 in Javacode eine Rumpfklasse für Formelausdrücke, die eine Methode zum Differenzieren des Formelausdrucks nach der Variablen x und eine Methode zur Ausgabe des Formelausdrucks, bietet.

Algorithmus 3.1.1 (Klasse für Formelausdrücke).

```
abstract class FAusdruck {
  public abstract String print();        (Ausgabe des Formelausdrucks)
  public abstract FAusdruck diffx();     (Differenzieren des Formelausdrucks)
}
```

In Algorithmus 3.1.2 zeigen wir, wie die Implementierung dieser Rumpfklasse für die Summe zweier Formelausdrücke aussehen kann. In der Variablen `operand1` speichern wir den ersten Summanden, in der Variablen `operand2` den zweiten. Der Aufruf der Methode `print` gibt zuerst den ersten Summanden, dann ein Pluszeichen und danach den zweiten Summanden aus. Die Methode `diffx` erzeugt einen neuen Formelausdruck der die Summe der Ableitung von `operand1` und der Ableitung von `operand2` ist $((u+v)' = u' + v')$.

Algorithmus 3.1.2 (Klasse für Summe zweier Formelausdrücke).

```
class Summe_FAusdruck extends FAusdruck {
  private FAusdruck operand1;
  private FAusdruck operand2;

  public String print() {        (Ausgabe des Formelausdrucks)
    return(operand1.print() + '' + '' + operand2.print());
  }

  public FAusdruck diffx() {        (Differenzieren des Formelausdrucks)
    return(new Summe_FAusdruck(operand1.diffx(), operand2.diffx()));
  }

  Summe_FAusdruck(FAusdruck wert1, FAusdruck wert2) {        (Konstruktor)
    operand1 = wert1;
    operand2 = wert2;
  }
}
```

3.2 Eine lineare Integralgleichung

In diesem Abschnitt untersuchen wir als erstes Beispiel die lineare Fredholmsche Integralgleichung zweiter Art

$$x(s) - \int_a^b \hat{k}(s,t)x(t)\,dt = y(s), \qquad s \in [a,b]$$

als Spezialfall der allgemeinen nichtlinearen Gleichung

$$x(s) - \int_a^b k(s,t,x(t))\,dt = y(s), \qquad s \in [a,b].$$

Mit anderen Worten: Der Kern hat die spezielle Form

$$k(s,t,x) := \hat{k}(s,t)x.$$

Diese Voraussetzung vereinfacht das in Kapitel 2 beschriebene Einschließungsverfahren wie folgt:

Schritt 1: Berechnung einer Näherungslösung ω

Das im Abschnitt 2.3 vorgestellte Verfahren reduziert sich im linearen Fall auf das Nyström-Verfahren, denn im ersten Homotopieschritt lösen wir auf Grund unserer speziellen Schrittweitensteuerung die Operatorgleichung

$$\mathcal{F}_1 x = \mathcal{F}x = x - \int_a^b k(.,t,x(t))\,dt - y = 0,$$

die mit der gegebenen linearen Integralgleichung identisch ist. Für lineare Gleichungen liefert das Newton-Verfahren aber bekanntlich bereits nach einem Schritt das Ergebnis.

Schritt 2: Berechnung einer Konstanten δ

Bei der Berechnung einer Defektschranke δ ergibt sich keine Vereinfachung zur nichtlinearer Integralgleichung.

Schritt 3: Berechnung einer Konstanten K

Wir berechnen eine Konstante K, so dass

$$||\mathcal{L}^{-1}|| \leq K$$

gilt. Dabei ist

$$\mathcal{L}x = x - \int_a^b k_3(.,t,\omega(t))x(t)\,dt = x - \int_a^b \hat{k}(.,t)x(t)\,dt$$

der am Punkt ω linearisierte Operator $\mathcal{I} - \mathcal{K}$. Aufgrund dieser Gleichung sehen wir, dass der Operator $\mathcal{L}$, und damit auch die Konstante K, im linearen Fall nicht von

ω abhängt. Wir brauchen deshalb für unterschiedliche rechte Seiten y keine neuen Konstanten K zu berechnen. Eine Beschleunigung des in Abschnitt 2.5 vorgestellten Verfahrens können wir erreichen, weil wir darin nun keine Einschließung von ω mehr benötigen.

Schritt 4: Bestimmung einer Funktion G

Für den Operator

$$(\mathcal{G}x)(s) := \int_a^b \big(k_3(s,t,\omega(t))x(t) - \big[k(s,t,\omega(t)+x(t)) - k(s,t,\omega(t))\big]\big)\, dt$$

gilt im linearen Fall

$$\begin{aligned}(\mathcal{G}x)(s) &= \int_a^b \big(k_3(s,t,\omega(t))x(t) - \big[k(s,t,\omega(t)+x(t)) - k(s,t,\omega(t))\big]\big)\, dt \\ &= \int_a^b \big(\hat{k}(s,t)x(t) - \big[\hat{k}(s,t)(\omega(t)+x(t)) - \hat{k}(s,t)\omega(t)\big]\big)\, dt \\ &= 0.\end{aligned}$$

Definieren wir somit $G(\alpha) \equiv 0$, so ist G trivialerweise eine monoton wachsende Funktion $\mathbb{R}_+ \to \mathbb{R}_+$ mit $G(\alpha) = o(\alpha)$ für $\alpha \to 0$ und

$$||\mathcal{G}x|| \leq G(||x||) \qquad \text{für alle } x \in C([a,b]).$$

Schritt 5: Bestimmung einer Konstanten α

Wählen wir $G(\alpha) \equiv 0$, so können wir

$$\alpha := \delta \cdot K \geq 0$$

setzen, um die benötigte Ungleichung

$$\delta \leq \frac{\alpha}{K} - G(\alpha)$$

automatisch zu erfüllen.

In einem konkreten Beispiel wenden wir das so vereinfachte Einschließungsverfahren auf die lineare Gleichung

$$x(s) - \int_0^1 \frac{\gamma^2 - 1}{1 + \gamma^2 - 2\gamma\cos(2\pi(s+t))} x(t)\, dt = (1+\gamma)\cos(2\pi s), \qquad s \in [0,1] \qquad (3.1)$$

mit Parameter $0 \leq \gamma < 1$ (Beispiel 13 aus Gienger [11]) an.

Diese Gleichung ergibt sich bei der Umformulierung des Dirichletproblems für die Laplacegleichung $\triangle u = 0$ auf einer Ellipse E mit Rand

$$\partial E : \quad \begin{cases} x_1 = a\cos t, \\ x_2 = b\sin t, \end{cases} \qquad 0 \leq t \leq 2\pi,\ a \geq b$$

in eine Integralgleichung mit Hilfe des *Doppelschichtpotentials*, wenn wir

$$\gamma := \frac{a-b}{a+b}$$

setzen.

Die rechte Seite von (3.1) haben wir dabei so gewählt, dass

$$x(s) = \cos(2\pi s)$$

für alle $\gamma \in [0,1)$ Lösung von (3.1) ist.

In der nachfolgenden Tabelle haben wir die Berechnungen für unterschiedliche Werte γ zusammengestellt. Die in den Klammern angegebenen Werte entsprechen der Anzahl der Teilintervalle, die bei der Berechnung verwendet wurden. Wir haben immer so viele Intervalle gewählt, dass eine Einschließung mit einem Fehler der Größenordnung 10^{-5} gelungen ist. Für den Wert $\gamma = 0.9$ konnte Gienger in [11] keine Einschließung berechnen. Auch bei der Verwendung unseres Algorithmus wächst der Rechenaufwand bei diesem Problem deutlich. Die von uns berechneten Konstanten K sind durchweg kleiner als die von Gienger in [11] berechneten Konstanten. Die von Gienger angegebenen Fehlerschranken sind hingegen, bis auf den Fall $\gamma = 0.9$, kleiner. Mit einer größeren Anzahl von Teilintervallen bei der Berechnung der Defektschranke δ könnten wir unsere Fehlerschranken α aber noch weiter verbessern.

γ	K		δ		α
0.1	1.508 533	(100)	$10.301\,052 \cdot 10^{-6}$	(1000)	$1.6 \cdot 10^{-5}$
0.2	1.513 019	(100)	$20.942\,492 \cdot 10^{-6}$	(1000)	$3.2 \cdot 10^{-5}$
0.3	1.607 899	(100)	$38.379\,319 \cdot 10^{-6}$	(1000)	$6.2 \cdot 10^{-5}$
0.4	1.868 775	(100)	$17.233\,064 \cdot 10^{-6}$	(2000)	$3.3 \cdot 10^{-5}$
0.5	2.285 680	(100)	$31.549\,901 \cdot 10^{-6}$	(2000)	$7.3 \cdot 10^{-5}$
0.6	2.991 673	(100)	$15.243\,432 \cdot 10^{-6}$	(4000)	$4.6 \cdot 10^{-5}$
0.7	4.573 610	(100)	$8.185\,739 \cdot 10^{-6}$	(8000)	$3.8 \cdot 10^{-5}$
0.8	6.896 123	(200)	$5.441\,546 \cdot 10^{-6}$	(16000)	$3.8 \cdot 10^{-5}$
0.9	18.822 504	(400)	$1.575\,517 \cdot 10^{-6}$	(64000)	$3.0 \cdot 10^{-5}$

3.3 Eine einfache nichtlineare Integralgleichung

Als nächstes Beispiel betrachten wir die sehr einfache nichtlineare Fredholmsche Integralgleichung zweiter Art vom Urysohntyp

$$x(s) - \int_0^1 x^2(t)\,dt = \lambda, \qquad s \in [0,1] \tag{3.2}$$

mit Parameter $\lambda \in \mathbb{R}$, für die wir alle im Einschließungsverfahren auftretenden Konstanten explizit berechnen können. Wir zeigen sogar, dass die berechneten Konstanten kleinstmöglich sind. Anschließend vergleichen wir diese optimalen Konstanten mit denen, die das Softwarepaket berechnet.

Schritt 1: Berechnung einer Lösung x

Üblicherweise berechnen wir im ersten Schritt eine Näherungslösung ω für die nichtlineare Integralgleichung. In diesem speziellen Fall ist die Gleichung aber einfach genug, um eine exakte Lösung zu bestimmen. Diese vergleichen wir dann mit der vom Softwarepaket bestimmten Näherungslösung. Es gilt:

$$(3.2) \quad \Longleftrightarrow \quad x(s) = \lambda + \underbrace{\int_0^1 x^2(t)\,dt}_{=:\tilde{c}\text{ (const)}} = \underbrace{\lambda + \tilde{c}}_{=:c\text{ (const)}} = c.$$

Also ist die Lösung eine konstante Funktion $x(s) = c$. Wir setzen $x(s) = c$ in Gleichung (3.2) ein, um die Konstante c zu bestimmen:

$$\begin{aligned}(3.2) \quad &\Longleftrightarrow \quad c - \int_0^1 c^2\,dt = \lambda \quad \Longleftrightarrow \quad c - c^2 = \lambda \\ &\Longleftrightarrow \quad c_{1,2} = \frac{1}{2} \pm \frac{1}{2}\sqrt{1-4\lambda}.\end{aligned}$$

Gleichung (3.2) besitzt somit für $\lambda < 0.25$ zwei reelle Lösungen, für $\lambda = 0.25$ eine reelle Lösung und für $\lambda > 0.25$ keine reellen Lösungen.

Schritt 2: Berechnung einer Konstanten δ

Für die zuvor berechnete exakte Lösung x ist der Defekt natürlich Null. Für die mit dem Softwarepaket berechneten konstanten Näherungslösungen ω berechnen wir die optimale Defektschranke durch

$$||d(\omega)|| = \max_{s\in[a,b]} \left|\omega(s) - \int_0^1 \omega^2(t)\,dt - \lambda\right| = |\omega - \omega^2 - \lambda| =: \delta. \tag{3.3}$$

Schritt 3: Berechnung einer Konstanten K

Wir berechnen die kleinste Konstante K, so dass

$$||\mathcal{L}^{-1}|| \leq K$$

gilt. Für dieses Beispiel ist

$$(\mathcal{L}x)(s) = x(s) - \int_a^b k_3(s,t,\omega(t))x(t)\,dt = x(s) - \int_0^1 2\omega(t)x(t)\,dt$$

der am Punkt ω linearisierte Operator $\mathcal{I} - \mathcal{K}$. Um die kleinstmögliche Konstante K zu berechnen, für die $||\mathcal{L}^{-1}|| \leq K$ gilt, lösen wir $\mathcal{L}x = y$ nach x auf.

$$\mathcal{L}x = y \quad \Longleftrightarrow \quad x - \underbrace{\int_0^1 2\omega(t)x(t)\,dt}_{=:\bar{c}\text{ (const)}} = y \quad \Longleftrightarrow \quad x = y + \bar{c}.$$

Um die Konstante $\bar{c}$ zu bestimmen, setzen wir $x = y + \bar{c}$ in $\mathcal{L}x = y$ ein.

$$\begin{aligned}
\mathcal{L}x = y \qquad &\Longleftrightarrow \qquad y + \bar{c} - \int_0^1 2\omega(t)(y(t) + \bar{c})\,dt = y \\
&\overset{\omega=\text{const}}{\Longleftrightarrow} \qquad \bar{c}(1-2\omega) = 2\omega \int_0^1 y(t)\,dt \\
&\Longleftrightarrow \qquad \bar{c} = \frac{2\omega}{1-2\omega}\int_0^1 y(t)\,dt \qquad \text{falls } \omega \neq \frac{1}{2}.
\end{aligned} \tag{3.4}$$

Damit können wir die gesuchte Norm nach oben abschätzen:

$$\begin{aligned}
||\mathcal{L}^{-1}|| &= \sup_{y\in C([0,1])\setminus\{0\}} \frac{||\mathcal{L}^{-1}y||}{||y||} = \sup_{\substack{y\in C([0,1])\setminus\{0\} \\ \bar{c} \text{ aus } (3.4)}} \frac{||y+\bar{c}||}{||y||} \leq \sup_{\substack{y\in C([0,1])\setminus\{0\} \\ \bar{c} \text{ aus } (3.4)}} \frac{||y|| + |\bar{c}|}{||y||} \\
&= 1 + \sup_{y\in C([0,1])\setminus\{0\}} \frac{\left|\frac{2\omega}{1-2\omega}\int_0^1 y(t)\,dt\right|}{||y||} = 1 + \left|\frac{2\omega}{1-2\omega}\right| \underbrace{\sup_{y\in C([0,1])\setminus\{0\}} \frac{\left|\int_0^1 y(t)\,dt\right|}{||y||}}_{=1} \\
&= 1 + \left|\frac{2\omega}{1-2\omega}\right|.
\end{aligned}$$

Die Norm schätzen wir nach unten ab, indem wir speziell $y = 1$ wählen:

$$||\mathcal{L}^{-1}|| = \sup_{y\in C([0,1])\setminus\{0\}} \frac{||\mathcal{L}y||}{||y||} \geq \frac{||\mathcal{L}^{-1}1||}{||1||} = \left|\left|1 + \frac{2\omega}{1-2\omega}\int_0^1 1\,dt\right|\right| = \left|1 + \frac{2\omega}{1-2\omega}\right|.$$

Für $0 \leq \omega < \frac{1}{2}$ ist $|\frac{2\omega}{1-2\omega}| = \frac{2\omega}{1-2\omega}$ und somit $||\mathcal{L}^{-1}|| = 1 + \frac{2\omega}{1-2\omega}$ nach obiger Rechnung. Als kleinstmögliche Schranke für $0 \leq \omega < \frac{1}{2}$ erhalten wir somit:

$$K := 1 + \frac{2\omega}{1-2\omega}. \tag{3.5}$$

Schritt 4: Bestimmung einer Funktion G

Wir bestimmen die punktweise kleinste monoton wachsende Funktion $G : \mathbb{R}_+ \to \mathbb{R}_+$ mit den Eigenschaften

$$||\mathcal{G}x|| \leq G(||x||) \qquad \text{für alle } x \in C([a,b])$$

und $G(\alpha) = o(\alpha)$ für $\alpha \to 0$. Wegen

$$\begin{aligned}
||\mathcal{G}x|| &= \left|\left| \int_a^b k_3(.,t,\omega(t))x(t) - \left[k(.,t,\omega(t)+x(t)) - k(.,t,\omega(t))\right] dt \right|\right| \\
&= \left|\left| \int_0^1 2\omega(t)x(t) - \left[(\omega(t)+x(t))^2 - \omega^2(t)\right] dt \right|\right| \\
&= \left|\left| \int_0^1 2\omega(t)x(t) - \left[\omega^2(t) + 2\omega(t)x(t) + x^2(t) - \omega^2(t)\right] dt \right|\right| \\
&= \left|\left| \int_0^1 -x^2(t)\,dt \right|\right| \leq ||x||^2 \leq \alpha^2
\end{aligned}$$

setzen wir

$$G(\alpha) := \alpha^2. \tag{3.6}$$

Dieses $G(\alpha)$ ist optimal, denn für $x(s) = \alpha$ gilt Gleichheit in obiger Abschätzung.

Schritt 5: Bestimmung einer Konstanten α

Wir suchen die kleinste Konstante α, so dass

$$\delta \leq \frac{\alpha}{K} - G(\alpha)$$

erfüllt ist. Für $G(\alpha) = \alpha^2$ ist diese durch

$$\alpha := \frac{1}{2K} - \sqrt{\left(\frac{1}{2K}\right)^2 - \delta} \tag{3.7}$$

gegeben.

Wir berechnen nun für die drei Fälle $\lambda = 0.24$, $\lambda = 0.25$ und $\lambda = 0.26$ die optimalen Konstanten und vergleichen diese mit den Ergebnissen des Programmpakets.

$\boldsymbol{\lambda = 0.24}$ Gleichung (3.2) besitzt die beiden Lösungen $x_1(s) = 0.4$ und $x_2(s) = 0.6$. Das Programm `omega` liefert nach einem Homotopieschritt mit 5 Newton-Schritten die Näherungslösung $\omega(s) = 0.399\,999\,995\ldots$.

Aus Gleichung (3.3) erhalten wir $\delta = 9.564\ 549\ 437\ldots \cdot 10^{-10}$ als kleinste obere Schranke für den Defekt. Das Programm `delta` liefert bei einer Unterteilung in $m = 1000$ Teilintervalle die Schranke $\delta_{1000} = 9.564\,971\,870\ldots \cdot 10^{-10}$, welche offenbar sehr nahe an δ liegt.

Nach Gleichung (3.5) berechnen wir $K = 4.999\,999\,760\ldots$ als optimale Schranke. Das Programm `const_k` liefert bei der Verwendung von nur $m = 100$ Teilintervallen schon die recht gute Schranke $K_{100} = 5.039\,999\,758\ldots$.

Das Programm `func_g` berechnet bei der Verwendung von $m = 1000$ Teilintervallen für $\alpha = 5 \cdot 10^{-9}$ den Funktionswert $G_{1000}(\alpha) = 5.000\ 000\ 000\ 000\ 561 \cdot 10^{-17}$, der ziemlich genau um den Faktor 2 größer ist als der optimale Wert $G(\alpha) = \alpha^2 = 2.5 \cdot 10^{-17}$. Dieser Faktor ist dadurch bedingt, dass das in Abschnitt 2.6 beschriebene Vorgehen den Term mit Hilfe des Mittelwertsatzes abschätzt:

$$\begin{aligned}
||\mathcal{G}x|| &= \left|\left| \int_0^1 k_3(.,t,\omega(t))x(t) - [k(.,t,\omega(t)+x(t)) - k(.,t,\omega(t))]\,dt \right|\right| \\
&\leq \alpha \max_{i=1,\ldots,m} \max_{s\in[s]_{m,i}} \sum_{j=1}^{m} \int_{[t]_{m,j}} |k_3(s,t,\omega(t)) - k_3(s,t,\xi(t))|\,dt \\
&= \alpha \max_{i=1,\ldots,m} \max_{s\in[s]_{m,i}} \sum_{j=1}^{m} \int_{[t]_{m,j}} 2\underbrace{|\omega(t) - \xi(t)|}_{\leq\alpha}\,dt.
\end{aligned}$$

Dabei kann es zu einer Überschätzung kommen, die sich hier in diesem Faktor 2 äußert. Wie wir im nächsten Absatz sehen werden, hat der durch das Softwarepaket berechnete größere Funktionswert $G(\alpha)$ aber kaum Einfluss auf die Qualität der Einschließung.

Die nach Gleichung (3.7) berechnete optimale Fehlerschranke $\alpha = 4.782\,274\,604\ldots \cdot 10^{-9}$ entspricht genau dem tatsächlichen Fehler unserer Näherungslösung:

$$||\omega - x|| = |0.399\,999\,995\ldots - 0.4| = 4.782\,274\,604\ldots \cdot 10^{-9}.$$

Das Programm `alpha` verifiziert, dass für $\alpha = 5 \cdot 10^{-9}$ die Ungleichung

$$\delta_{1000} \leq \frac{\alpha}{K_{100}} - G_{1000}(\alpha)$$

erfüllt ist. Damit können wir mit unserem Softwarepaket die Existenz einer Lösung von (3.2) für $\lambda = 0.24$ nachweisen und diese mit einem Fehler von $\alpha = 5 \cdot 10^{-9}$ einschließen.

$\boldsymbol{\lambda = 0.25}$ Gleichung (3.2) besitzt genau eine reelle Lösung, $x(s) = 0.5$. Das Programm `omega` liefert nach einem Homotopieschritt mit 12 Newton-Schritten die Näherungslösung $\omega(s) = 0.499\,908\,455\ldots$

Nach Gleichung (3.3) erhalten wir $\delta = 8.380\,436\,475\ldots \cdot 10^{-9}$ als optimale Defektschranke. Das Programm `delta` berechnet mit $m = 1000$ Teilintervalle die Schranke $\delta_{1000} = 8.380\,491\,856\ldots \cdot 10^{-9}$.

Für $\omega = 0.5$ ist der Operator $\mathcal{L}$ singulär. Da die Näherungslösung in der Nähe dieser Singularität liegt, ist die optimale Schranke $K = 5461.811\,218\,204\ldots$ für $||\mathcal{L}^{-1}||$ relativ groß. Das Programm `const_k` liefert bei Verwendung von $m = 100$ Teilintervallen die Schranke $K_{100} = 5516.419\,335\,071\ldots$, bei $m = 1600$ Teilintervalle die Schranke $K_{1600} = 5465.224\,241\,305\ldots$.

Mit den optimalen Schranken δ und K und der punktweise kleinsten Funktion $G(\alpha) = \alpha^2$ ist die Ungleichung

$$\delta \leq \frac{\alpha}{K} - G(\alpha) = \frac{\alpha}{K} - \alpha^2$$

nur für $\alpha = 9.154\,472\,390\ldots \cdot 10^{-5}$ erfüllt. Diese Schranke α entspricht auch diesmal wieder genau dem Fehler $||x - \omega||$. Mit unserem Softwarepaket können wir diese Ungleichung aber nicht erfüllen, da durch die endliche Rechengenauigkeit des Computers die vom Softwarepaket berechneten Konstanten die optimalen immer überschätzen werden.

Auch wenn wir hier schon wissen, dass wir keine Einschließung mit unserem Softwarepaket berechnen können, sei der Vollständigkeit wegen erwähnt, dass das Programm `func_g` den optimalen Funktionswert um den Faktor 2 überschätzt, wie schon im Fall $\lambda = 0.24$.

Unser Softwarepaket ist somit nicht in der Lage eine Lösung einzuschließen, obwohl eine exakte Lösung von (3.2) existiert und wir eine Näherungslösung berechnen konnten, die nur einen Fehler der Größenordnung 10^{-5} aufweist. Dies ist auf Grund der Umkehrpunkteigenschaft von Gleichung (3.2) für den Parameter $\lambda = 0.25$ aber auch nicht anders zu erwarten.

$\boldsymbol{\lambda = 0.26}$ Gleichung (3.2) besitzt keine reellen Lösungen. Dem Programm `omega` gelingt es erwartungsgemäß nicht, innerhalb der maximal zulässigen Schrittzahl eine Näherungslösung zu berechnen.

Auch wenn wir schon wissen, dass keine Lösung von Gleichung (3.2) existiert, wollen wir die nachfolgenden Teile des Einschließungsalgorithmus mit einer gewählten Näherungslösung durchführen, um zu sehen, welcher Teilschritt eine Einschließung verhindert.

Die Näherungslösung $\omega = 0.5$ liefert unter allen konstanten Funktionen den geringsten Defekt $\delta = 0.1$. Für diese Näherungslösung ist der Operator $\mathcal{L}$ jedoch singulär (vergleiche dazu auch den Fall $\lambda = 0.25$).

Stattdessen wählen wir $\omega = 0.42$ als Näherungslösung, welche $\delta = 0.016\ 400$ als obere Schranke für den Defekt liefert. Das Programm `delta` berechnet bei der Unterteilung in $m = 1000$ Teilintervalle die Schranke $\delta_m = 0.016\ 400\ 000\ 000\ 042\ldots$.

Die optimale Schranke für $||\mathcal{L}^{-1}||$ ist gegeben durch $K = 6.25$. Mit einer Unterteilung in $m = 100$ Teilintervalle berechnet das Programm `const_k` eine Schranke $K_{100} = 6.276\ 250\ 000\ldots$.

Für die optimalen Konstanten δ und K und die punktweise kleinste Funktion $G(\alpha) = \alpha^2$ existiert aber kein $\alpha > 0$, für das die Ungleichung

$$\delta \leq \frac{\alpha}{K} - G(\alpha) = \frac{\alpha}{K} - \alpha^2$$

erfüllt ist, so dass unser Einschließungsalgorithmus an dieser Stelle abbricht und keine falsche Lösung eingeschlossen wird.

3.4 Eine nichtlineare Integralgleichung

Wir betrachten als weiteres Beispiel die nichtlineare Fredholmsche Integralgleichung zweiter Art vom Urysohntyp

$$x(s) - \int_0^1 \sin(stx(t))\,dt = \sin(10s). \tag{3.8}$$

Mit dem Algorithmus aus Kapitel 2 untersuchen wir die Gleichung (3.8) auf Existenz einer Lösung und schließen diese gegebenenfalls ein. Die angegebenen Rechenzeiten im ersten Schritt beziehen sich auf ein serielles System mit 800 MHz Pentium III Prozessor, alle anderen Rechenzeiten beziehen sich auf einen Cluster mit 16 Pentium III Prozessoren mit 800 MHz, die durch ein Gigabit Ethernet verbunden waren.

Schritt 1: Berechnung einer Näherungslösung ω

Um die Näherungslösung ω (siehe Abbildung 3.2) zu erhalten, haben wir den Integraloperator mit $n = 500$ Quadraturknoten diskretisiert. Nur ein einziger Homotopieschritt war erforderlich, der selbst wiederum nur drei Newton-Schritte benötigte. Die gesamte Berechnung dauerte 0.59 Sekunden.

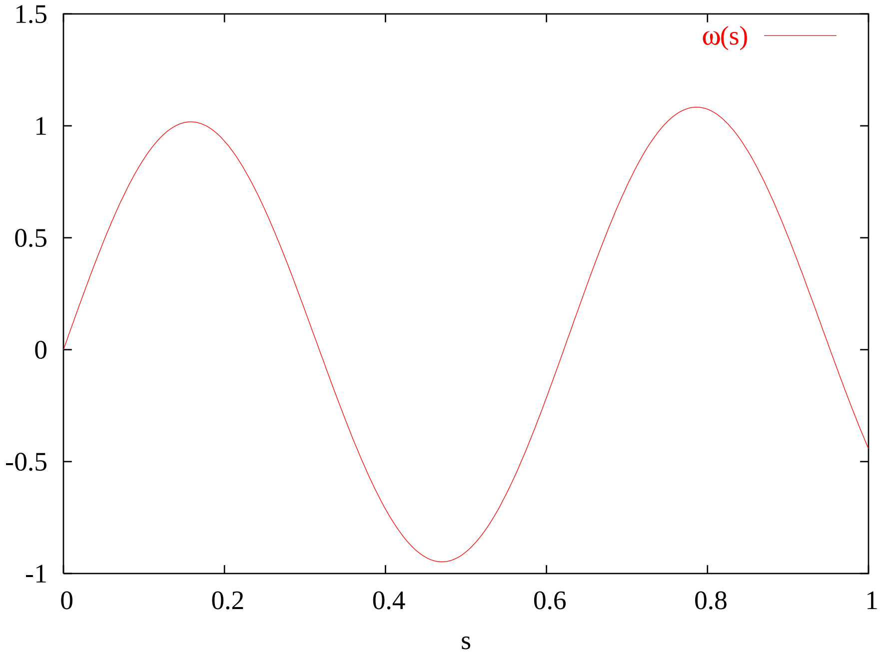


Abb. 3.2: Näherungslösung ω berechnet mit $n = 500$ Quadraturknoten.

Schritt 2: Berechnung einer Konstanten δ

Die Defektschranke δ haben wir mit dem in Abschnitt 2.4 vorgestellten $O(m^2)$-Algorithmus für eine unterschiedliche Anzahl m von Teilintervallen berechnet. Die Ergebnisse sind in der folgenden Tabelle zusammengestellt. Im Einzelnen ist bemerkenswert, dass eine Verdoppelung von m (der Anzahl der verwendeten Teilintervalle) zu einer vier mal kleineren Defektschranke δ führt (siehe letzte Spalte der Tabelle). Der Grund hierfür liegt darin, dass der tatsächliche Defekt der Integralgleichung sehr klein ist und deswegen die Qualität der Integraleinschließung hauptsächlich die Größe der Defektschranke bestimmt. Bei der Verwendung der zentrierten Form mit Steigung konvergiert die Integraleinschließung quadratisch gegen den tatsächlichen Wert des Integrals.

#Teilintervalle m	δ_m	Rechenzeit (s)	$\delta_{m/2}/\delta_m$
1 000	$6.307\,497\,909 \cdot 10^{-6}$	0.80	-
2 000	$1.577\,159\,448 \cdot 10^{-6}$	2.46	4
4 000	$0.394\,322\,831 \cdot 10^{-6}$	8.39	4
8 000	$0.098\,582\,814 \cdot 10^{-6}$	30.56	4
16 000	$0.024\,644\,026 \cdot 10^{-6}$	116.03	4
32 000	$0.006\,158\,956 \cdot 10^{-6}$	448.97	4
64 000	$0.001\,537\,840 \cdot 10^{-6}$	1 766.24	4
128 000	$0.000\,382\,978 \cdot 10^{-6}$	7 009.16	4

Schritt 3: Berechnung einer Konstanten K

Eine Konstante K mit

$$||\mathcal{L}^{-1}|| \leq K \qquad \text{mit } \mathcal{L} = (\mathcal{I} - \mathcal{K})'[\omega]$$

haben wir für unterschiedliche Werte von m (m wieder die Anzahl der Teilintervalle) mit dem $O(m^3)$-Algorithmus aus Abschnitt 2.5 berechnet. Selbst für kleine m erhalten wir bereits gute obere Schranken K_m, die sich bei wachsendem m nur unwesentlich verbessern. Die Tabelle fasst die Ergebnisse zusammen.

#Teilintervalle m	K_m	Rechenzeit (s)
25	1.696 993 868	0.17
50	1.662 145 117	0.15
100	1.650 305 158	0.32
200	1.645 727 330	1.36
400	1.643 770 615	8.41
800	1.642 874 918	58.59
1 600	1.642 447 690	467.52

Schritt 4: Bestimmung einer Funktion G

Die Funktion $G(\alpha)$, die wir für unterschiedliche Werte von m mit dem $O(m^2)$-Algorithmus aus Abschnitt 2.6 berechnet haben, wird in Abbildung 3.3 veranschaulicht. Hier erkennen wir, dass wie gefordert $G(\alpha) = o(\alpha)$ für $\alpha \to 0$ gilt. Wir haben hier $G(\alpha)$ für Intervallunterteilungen bis $m = 128\,000$ berechnet, um das Verhalten von $G(\alpha)$ für unterschiedliche Werte von m bei festem α zu zeigen. Zur Einschließung hätte jedoch schon $G_{1\,000}(10^{-9})$ genügt. Mit wachsendem m verbessert sich der Funktionswert $G_m(10^{-9})$ nur noch unwesentlich.

#Teilintervalle m	$G_m(10^{-9})$	Rechenzeit (s)
1 000	$1.363\,386\,531 \cdot 10^{-19}$	0.68
2 000	$1.358\,656\,470 \cdot 10^{-19}$	2.53
4 000	$1.356\,288\,201 \cdot 10^{-19}$	9.65
8 000	$1.355\,102\,084 \cdot 10^{-19}$	37.44
16 000	$1.354\,509\,170 \cdot 10^{-19}$	148.02
32 000	$1.354\,212\,584 \cdot 10^{-19}$	584.29
64 000	$1.354\,064\,285 \cdot 10^{-19}$	2 319.52
128 000	$1.353\,990\,136 \cdot 10^{-19}$	9 252.98

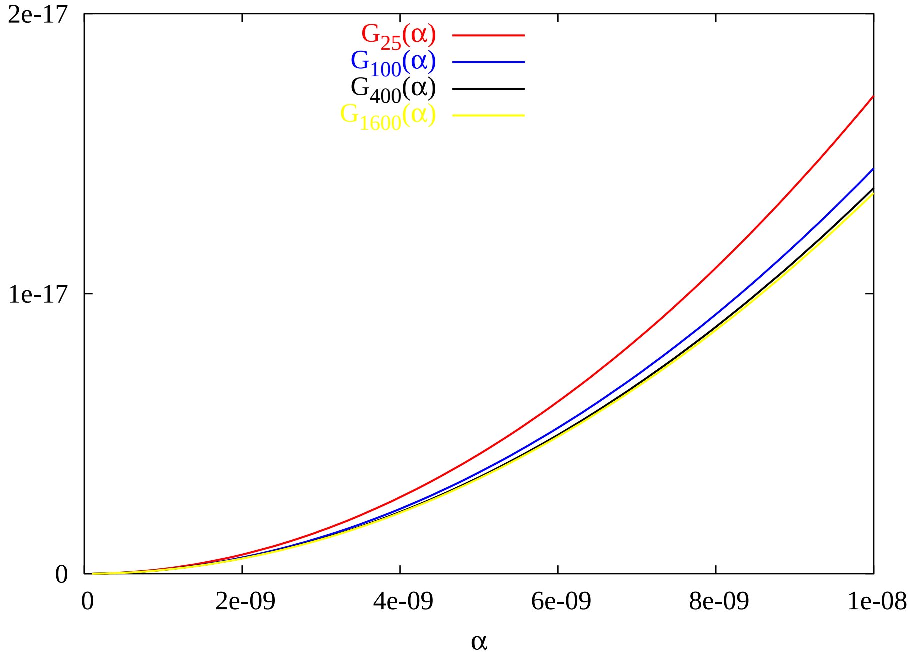


Abb. 3.3: Verlauf der Funktion G, berechnet mit $m = 25,\ 100,\ 400,\ 1600$ Teilintervallen.

Schritt 5: Bestimmung einer Konstanten α

Für $\alpha = 10^{-9}$ gilt offenbar

$$\begin{aligned}\delta_{128\,000} &= 3.829\,78 \cdot 10^{-10} \\ &\leq 6.088\,473\,964 \ldots \cdot 10^{-9} = \frac{10^{-9}}{1.642\,447\,690} - 1.353\,990\,136 \cdot 10^{-19} \\ &= \frac{\alpha}{K_{1\,600}} - G_{128\,000}(\alpha).\end{aligned}$$

Dies sichert nach Theorem 2.2.1 die Existenz einer Lösung x der Gleichung (3.8) und liefert die Einschließung

$$||x - \omega|| \leq 10^{-9}.$$

In Abbildung 3.4 zeigen wir den Verlauf von $\alpha/K_{1\,600} - G_{1\,600}(\alpha)$ für $\alpha \in [0, 10^{-8}]$. Für alle $\alpha \in [10^{-9}, 10^{-8}]$ ist die Ungleichung $\delta_{128\,000} \leq \alpha/K_{1\,600} - G_{1\,600}(\alpha)$ erfüllt und somit die Existenz einer Lösung von Gleichung (3.8) gesichert. Für eine optimale Einschließung wählen wir natürlich $\alpha = 10^{-9}$.

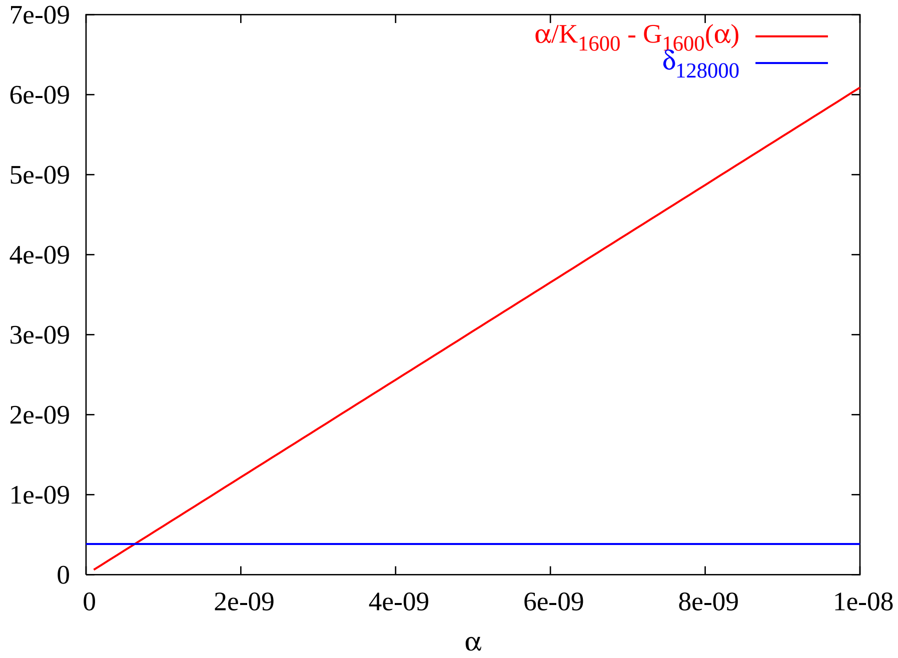


Abb. 3.4: Die Defektschranke $\delta_{128\,000}$ ist so klein, dass für alle $\alpha \in [10^{-9}, 10^{-8}]$ die Ungleichung $\delta_{128\,000} \leq \alpha/K_{1\,600} - G_{1\,600}(\alpha)$ erfüllt ist. Die Einschließung der Lösung gelingt somit für $\alpha = 10^{-9}$.

3.5 Eine nichtlineare Integralgleichung, bei der die zweite Fréchetableitung nicht existiert

Bei den bisher vorgestellten Beispielen existierte immer die zweite Fréchetableitung des Operators

$$\mathcal{F}x = x - \int_a^b k(.,t,x(t))\,dt - y,$$

so dass prinzipiell auch der Satz von Kantorovič 1.2.5 zur Lösungsverifikation anwendbar wäre. Im letzten Beispiel betrachten wir nun eine Integralgleichung, bei der die zweite Fréchetableitung

$$(\mathcal{F}''[x_0]h_1)h_2 = \int_a^b k_{3,3}(.,t,x_0(t))h_1(t)h_2(t)\,dt$$

nicht für alle $x_0 \in C([a,b])$ existiert, der Algorithmus aus Kapitel 2 aber anwendbar ist. Wir untersuchen die Integralgleichung

$$x(s) - \int_{-1}^{1} k(s,t,x(t))\,dt = y(s), \qquad s \in [-1,1] \tag{3.9}$$

mit

$$k(s,t,x) := \begin{cases} -st\cos(stx) + st, & x < 0, \\ \ \ st\cos(stx) - st, & x \geq 0, \end{cases}$$

$$y(s) := 2s - \sin s.$$

Durch Differenzieren erhalten wir:

$$k_3(s,t,x) = \begin{cases} \ \ (st)^2 \sin(stx), & x < 0, \\ -(st)^2 \sin(stx), & x > 0, \end{cases}$$

$$k_{1,3}(s,t,x) = \begin{cases} \ \ 2st^2 \sin(stx) + s^2t^3x\cos(stx), & x < 0, \\ -2st^2 \sin(stx) - s^2t^3x\cos(stx), & x > 0, \end{cases}$$

$$k_{3,3}(s,t,x) = \begin{cases} \ \ (st)^3 \cos(stx), & x < 0, \\ -(st)^3 \cos(stx), & x > 0. \end{cases}$$

Da

$$\lim_{(s,t,x)\to(s_0,t_0,0-)} k(s,t,x) = 0 = \lim_{(s,t,x)\to(s_0,t_0,0+)} k(s,t,x),$$

$$\lim_{(s,t,x)\to(s_0,t_0,0-)} k_3(s,t,x) = 0 = \lim_{(s,t,x)\to(s_0,t_0,0+)} k_3(s,t,x),$$

$$\lim_{(s,t,x)\to(s_0,t_0,0-)} k_{1,3}(s,t,x) = 0 = \lim_{(s,t,x)\to(s_0,t_0,0+)} k_{1,3}(s,t,x)$$

für alle $s_0, t_0 \in [-1,1]$ gilt, sind k, k_3 und $k_{1,3}$ stetig. Die rechte Seite y ist offensichtlich stetig differenzierbar und somit erfüllt die Integralgleichung (3.9) die Glattheitsvoraussetzungen, die für die Anwendung des Algorithmus nötig sind.

Da aber

$$\lim_{(s,t,x)\to(s_0,t_0,0-)} k_{3,3}(s,t,x) = (st)^3 \neq -(st)^3 = \lim_{(s,t,x)\to(s_0,t_0,0+)} k_{3,3}(s,t,x)$$

für alle $s_0, t_0 \in [-1,1]\backslash\{0\}$ gilt, ist der Kern nur einmal stetig nach x differenzierbar und die zweite Fréchetableitung existiert somit nicht für jedes $x_0 \in C([a,b])$, z.B. nicht für $x_0 = 0$.

Der Kern und die rechte Seite sind gerade so gewählt, dass $x(s) = s$, $s \in [-1,1]$, die Integralgleichung löst. Somit treten die beiden Fälle $x < 0$ und $x \geq 0$ in der Definition des Kerns auch auf.

Alle folgenden Rechenzeiten beziehen sich auf die Rechnung auf einem seriellen Rechner mit AMD Athlon XP 2000+ Prozessor. Bei Rechenzeiten, die mit einem '-' gekennzeichnet sind, ist es in der Zeitmessroutine von Linux zu einem Überlauf gekommen.

Schritt 1: Berechnung einer Näherungslösung ω

Wir verwenden im Programm `omega` nur den ersten Homotopieschritt von $\lambda_0 = 0$ nach $\lambda_1 = 1$

$$\begin{aligned} x^*(1,s) &= x^*(0,s) + 1\frac{\partial x^*}{\partial \lambda}(0,s) + \frac{1^2}{2}\frac{\partial^2 x^*}{\partial \lambda^2}(0,s) \\ &= y(s) + \int_a^b k(s,t,y(t))\,dt + \frac{1}{2}\int_a^b 2k_3(s,\bar{t},y(\bar{t}))\int_a^b k(\bar{t},t,y(t))\,dt\,d\bar{t}, \end{aligned}$$

um eine Startnäherung für das Newton-Verfahren für $\mathcal{F}x = \mathcal{F}_1 x = 0$ zu bestimmen, da bei Zwischenschritten des Homotopieverfahrens auch die Ableitung $k_{3,3}$ benötigt würde. Dennoch wird nach nur 4 Newton-Schritten in 0.29 Sekunden eine Näherungslösung ω berechnet.

Schritt 2: Berechnung einer Konstanten δ

Das Programm `delta` liefert für unterschiedliche Anzahlen von Teilintervallen m die nachfolgenden Defektschranken. Deutlich ist die quadratische Konvergenz des Algorithmus zu sehen.

#Teilintervalle m	δ_m	Rechenzeit (s)	$\delta_{m/2}/\delta_m$
1 000	$1.685\,576\,796 \cdot 10^{-5}$	10.47	-
2 000	$0.421\,421\,807 \cdot 10^{-5}$	34.75	4
4 000	$0.105\,358\,841 \cdot 10^{-5}$	124.74	4
8 000	$0.026\,340\,117 \cdot 10^{-5}$	469.77	4
16 000	$0.006\,585\,075 \cdot 10^{-5}$	1 824.15	4
32 000	$0.001\,646\,287 \cdot 10^{-5}$	-	4
64 000	$0.000\,411\,622 \cdot 10^{-5}$	-	4
128 000	$0.000\,103\,033 \cdot 10^{-5}$	-	4

Schritt 3: Berechnung einer Konstanten K

Das Programm `const_k` berechnet für eine unterschiedliche Anzahl von Teilintervallen m eine obere Schranke K_m. Die Ergebnisse verbessern sich mit wachsendem m nur noch geringfügig, weswegen sich hier eine Rechnung für große m nicht rentiert.

#Teilintervalle m	K_m	Rechenzeit (s)
10	1.998 951 393	0.05
25	1.420 719 334	0.14
50	1.340 676 737	0.32
100	1.314 488 328	1.05
200	1.304 577 142	5.87
400	1.300 396 755	42.85
800	1.298 498 589	343.76
1 600	1.297 597 340	-

Schritt 4: Bestimmung einer Funktion G

In der nachfolgenden Tabelle befinden sich für eine verschiedene Anzahl von Teilintervallen m, aber festes $\alpha = 2 \cdot 10^{-9}$, die vom Programm `func_g` berechneten Funktionswerte $G_m(\alpha)$. Wie schon bei der Berechnung der Konstanten K verbessern sich auch hier mit wachsendem m die Funktionswerte nur noch unwesentlich.

#Teilintervalle m	$G_m(2 \cdot 10^{-9})$	Rechenzeit (s)
1 000	$1.531\,908\,834 \cdot 10^{-18}$	4.60
2 000	$1.528\,998\,792 \cdot 10^{-18}$	17.05
4 000	$1.527\,877\,675 \cdot 10^{-18}$	66.16
8 000	$1.527\,481\,017 \cdot 10^{-18}$	259.56
16 000	$1.527\,287\,169 \cdot 10^{-18}$	1024.14
32 000	$1.527\,190\,223 \cdot 10^{-18}$	-
64 000	$1.527\,141\,710 \cdot 10^{-18}$	-
128 000	$1.527\,117\,441 \cdot 10^{-18}$	-

Schritt 5: Bestimmung einer Konstanten α

Da für $\alpha = 2 \cdot 10^{-9}$ die Ungleichung

$$\begin{aligned} \delta_{128\,000} &= 1.030\,33 \cdot 10^{-9} \\ &\leq 1.541\,310\,186\ldots \cdot 10^{-9} = \frac{2 \cdot 10^{-9}}{1.297\,597\,340} - 1.527\,117\,441 \cdot 10^{-18} \\ &\qquad = \frac{\alpha}{K_{1\,600}} - G_{128\,000}(\alpha) \end{aligned}$$

erfüllt ist, existiert nach Theorem 2.2.1 eine Lösung x von Gleichung (3.9), die wir durch

$$||x - \omega|| \leq 2 \cdot 10^{-9}$$

einschließen können.

Literaturverzeichnis

[1] G. Alefeld, J. Herzberger; *Introduction to Interval Computations*; Academic Press, New York, 1983.

[2] G. Alefeld, I. Lenhardt, H. Obermaier; *Parallele numerische Verfahren*; Springer, Heidelberg, 2002.

[3] E. L. Allgower, K. Georg; *Numerical Continuation Methods: An Introduction*; Springer, Berlin, 1990.

[4] K. E. Atkinson; *The Numerical Solution of Fredholm Integral Equations of the Second Kind*; Preprint, Iowa City, 1994.

[5] M. Bräuer, W. Hofschuster, W. Krämer; *Steigungsarithmetiken in C-XSC*; Preprint, Bergische Universität GH Wuppertal, Wuppertal, 2001.
Erhältlich von:
`http://www.math.uni-wuppertal.de/org/WRST/preprints/prep_01_3.ps`

[6] C. G. Broyden; *Recent developments in solving nonlinear algebraic systems*; in: *Numerical Methods for Nonlinear Algebraic Equations*; P. Rabinowitz (Ed.); Gordon and Breach Science Publishers, 1969, S. 61–74.

[7] H.-J. Dobner; *Numerische Methoden zur verifizierten Lösung von Integralgleichungen*; Habilitationsschrift, Universität Karlsruhe, Karlsruhe, 1992.

[8] H.-J. Dobner; *An error controlling Nyström method*; Calcolo 31, 1994, S. 167 - 177.

[9] A. Frommer; *Lösung linearer Gleichungssysteme auf Parallelrechnern*; Vieweg & Sohn, Braunschweig, 1990.

[10] A. Geist, A Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam; *PVM: Parallel Virtual Machine*; MIT Press, Cambridge, 1994.

[11] A. Gienger; *Zur Lösungsverifikation bei Fredholmschen Integralgleichungen zweiter Art*; Dissertation, Universität Karlsruhe, Karlsruhe, 1997.

[12] A. Gienger; *Enclosing the solution of linear Fredholm integral equations of the second kind*; Z. Angew. Math. Mech. 79, 1999, S. 241 - 244.

[13] G. H. Golub, J. A. Welsch; *Calculation of Gauss quadrature rules*; Math. Comp. 23, 1969, S. 221 - 230.

[14] G. H. Golub, C. F. van Loan; *Matrix Computations (Third Edition)*; The Johns Hopkins University Press, Baltimore, 1996.

[15] A. Griewank; *Evaluating Derivatives. Principles and Techniques of Algorithmic Differentiation*; SIAM, Philadelphia, 2000.

[16] W. Hackbusch; *Integralgleichungen – Theorie und Numerik*; B. G. Teubner, Stuttgart, 1989.

[17] H. Heuser; *Funktionalanalysis*; B. G. Teubner, Stuttgart, 1975.

[18] H. Heuser; *Lehrbuch der Analysis Teil 1, 13. Auflage*; B. G. Teubner, Stuttgart, 2000.

[19] H. Heuser; *Lehrbuch der Analysis Teil 2, 11. Auflage*; B. G. Teubner, Stuttgart, 2000.

[20] W. Hofschuster, W. Krämer, S. Wedner, A. Wiethoff; *C-XSC 2.0 - A C++ Class Library for Extended Scientific Computing*; Preprint 2001/1, Universität Wuppertal, Wuppertal, 2001.
Erhältlich von:
`http://www.math.uni-wuppertal.de/org/WRST/preprints/prep_01_1.ps`

[21] A. J. Jerri; *Introduction to Integral Equations with Applications (Second Edition)*; John Wiley & Sons, New York, 1999.

[22] R. Klatte, U. W. Kulisch, A. Wiethoff, C. Lawo, M. Rauch; *C-XSC. A C++ Class Library for Extended Scientific Computing*; Springer, Berlin, 1993.

[23] W. Klein; *Zur Einschließung der Lösung von linearen und nichtlinearen Fredholmschen Integralgleichungssystemen zweiter Art*; Dissertation, Universität Karlsruhe, Karlsruhe, 1990.

[24] W. Klein; *Enclosure methods for linear and nonlinear systems of Fredholm integral equations of the second kind*; in: *Scientific Computing with Automatic Result Verification*; E. Adams (Ed.); Academic Press, Boston, 1993, S. 255 - 282.

[25] U. W. Kulisch, W. L. Miranker; *Computer Arithmetic in Theory and Practice*; Academic Press, New York, 1981.

[26] M. Lerch, G. Tischler, J. W. von Gudenberg, W. Hofschuster, W. Krämer; *The Interval Library filib++ 2.0 - Design, Features and Sample Programs*; Preprint 2001/4, Universität Wuppertal, Wuppertal, 2001.
Erhältlich von:
`http://www.math.uni-wuppertal.de/org/WRST/preprints/prep_01_4.ps`

[27] N. MacDonald, E. Minty, T. Harding, S. Brown; *Writing Message-Passing Parallel Programms with MPI*; Course Notes, Edinburgh Parallel Computing Center, University of Edinburgh, Edinburgh, 1995.
Erhältlich von:
`http://www.epcc.ed.ac.uk/epcc-tec/documents/coursemat.html`

[28] R. H. Moore; *Newton's method and variations*; in: *Nonlinear Integral Equations*; P. M. Anselone (Ed.); University of Wisconsin Press, Madison, 1964, S. 65 - 98.

[29] Message Passing Interface Forum; *MPI: A Message-Passing Interface Standard*; University of Tennessee, Knoxville, 1995.
Erhältlich von: `http://www.mpi-forum.org/docs/mpi-11.ps`

[30] Message Passing Interface Forum; *MPI-2: Extensions to the Message-Passing Interface*; University of Tennessee, Knoxville, 1997.
Erhältlich von: `http://www.mpi-forum.org/docs/mpi-20.ps`

[31] J. Nocedal, S. J. Wright; *Numerical Optimization*; Springer, New York, 1999.

[32] J. M. Ortega, W. C. Rheinboldt; *On discretization and differentiation of operators with application to Newton's method*; SIAM J. Numer. Anal. 3, 1966, S. 143 - 156.

[33] J. M. Ortega, W. C. Rheinboldt; *Iterative Solution of Nonlinear Equations in Several Variables*; Academic Press, New York, 1970.

[34] P. S. Pacheco; *Parallel Programming with MPI*; Morgan Kaufmann Publishers, San Francicso, 1997.

[35] M. Plum; *Verified existence and inclusion results for two-point boundary value problems*; in: *Contributions to Computer Arithmetic and Self-Validating numerical Methods*; C. Ullrich (Ed.); IMACS Ann. Comput. Appl. Math. 7, 1990, S. 341 - 355.

[36] M. Plum; *Computer-assisted existence proofs for two-point boundary value problems*; Computing 46, 1991, S. 19 - 34.

[37] M. Plum; *Computer-assisted enclosure methods for elliptic differential equations*; Linear Algebra Appl. 324, 2001, S. 147-187.

[38] L. B. Rall; *Computational Solution of Nonlinear Operator Equations*; Robert E. Kreiger Publishing Company, New York, 1979.

[39] L. B. Rall; *Automatic Differentiation: Techniques and Applications*; Springer, New York, 1981.

[40] D. Ratz; *Automatic Slope Computation and its Application in Nonsmooth Global Optimization*; Shaker Verlag, Aachen, 1998.

[41] T. Rauber, G. Rünger; *Parallele und verteilte Programmierung*; Springer, Berlin, 2000.

[42] S. Rump; *Kleine Fehlerschranken bei Matrixproblemen*; Dissertation, Universität Karlsruhe, Karlsruhe, 1980.

[43] P. Sanders, Th. Worsch; *Parallele Programmierung mit MPI – ein Praktikum*; Logos Verlag, Berlin, 1997.

[44] H. Schwetlick; *Numerische Lösung nichtlinearer Gleichungen*; R. Oldenbourg Verlag, München, 1979.

[45] J. Stoer; *Numerische Mathematik 1*; Springer, Berlin, 1993.

Stichwortverzeichnis

Lebenslauf

Name: Holger Obermaier

Geburtsdatum: 19.08.1973

Geburtsort: Mosbach/Baden

Eltern: Herbert Obermaier
Roswitha Obermaier, geb. Schmidt

Schulbildung: 1981 – Jul. 1984:

Grundschule Haßmersheim

1984 – Jun. 1990:

Realschule Obrigheim
Mittlere Reife am 22. Juni 1990

1990 – Mai 1993:

Technisches Gymnasium Mosbach
Abiturprüfung am 11. Mai 1993

Studium: Okt. 1993 – Jan. 1999:

Technomathematik mit Nebenfach Maschinenbau und Informatik
an der Universität Karlsruhe
Diplomprüfung am 20. Januar 1999

Berufstätigkeit: Okt. 1994 – März 1995:

Wissenschaftliche Hilfskraft
am Institut für Angewandte Mathematik
der Universität Karlsruhe

Apr. 1995 – Dez. 1998:

Wissenschaftliche Hilfskraft
am Institut für Mechanik
der Universität Karlsruhe

Okt. 1996 – März 1997:

Wissenschaftliche Hilfskraft
am Mathematischen Institut I
der Universität Karlsruhe

seit Feb. 1999:

Wissenschaftlicher Angestellter
am Institut für Angewandte Mathematik
der Universität Karlsruhe