Skriptum zur Vorlesung
Numerische Analysis

von Walter Zulehner
überarbeitet von Ewald Lindner

Institut für Numerische Mathematik
Johannes Kepler Universität Linz

Wintersemester 2005/06

4.Auflage

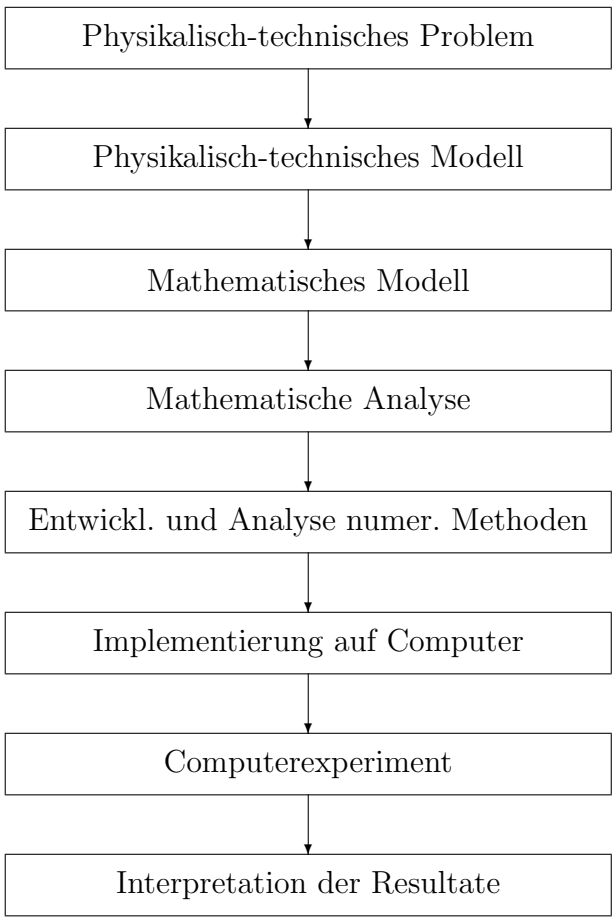
Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Der Problemlösungsprozess

Der Prozess des Lösens eines Anwendungsproblems mit numerischen Methoden lässt sich grob in folgende Stufen einteilen:



Das physikalisch-technische Modell berücksichtigt z.B. die relevanten Bilanzgleichungen, Materialgesetze, Minimalprinzipien, u.ä.

Dabei ist zu beachten, dass beim Übergang von einer realen Problemstellung auf ein Modell, also beim Modellieren, zwangsweise eine Reihe von Vereinfachungen zu treffen sind. Hier passiert bereits ein erster Fehler (**Modellfehler**) durch die gerechtfertigten oder ungerechtfertigten Modellannahmen.

Der Übergang vom physikalisch-technischen Modell zum mathematischen Modell ist fließend. Die getroffene Unterscheidung soll nur den unterschiedlichen Grad der Mathematisierung des Modells zum Ausdruck bringen. Am Ende steht im Idealfall ein wohl definiertes mathematisches Problem, wie z.B. ein Anfangsrandwertproblem für eine partielle Differentialgleichung mit vorgegebenen Eigenschaften.

Der zentrale Begriff der mathematischen Analyse ist das korrekt gestellte Problem, d.h., eine Lösung des Problems existiert, sie ist eindeutig und sie hängt stetig von den Daten ab. Die in vielen Fällen als selbstverständlich geltende Existenz und Eindeutigkeit einer Lösung des realen Problems bedeutet keineswegs, dass auch ein Modell diese Eigenschaften besitzen muss. Gelingt der Nachweis der Existenz und Eindeutigkeit einer Lösung, so ist eine Mindestanforderung an ein „sinnvolles“ Modell erfüllt. Die stetige Abhängigkeit von den Daten sichert die Richtigkeit einer Lösung auch bei kleinen Störungen in den Daten. Solche **Datenfehler** können z.B. durch Messfehler oder durch Fehler aufgrund einer vorangegangenen Rechnung entstanden sein. Die stetige Abhängigkeit muss nicht immer gegeben sein, siehe z. B. inkorrekt gestellte Probleme (Hadamard 1923).

Eine numerische Methode muss natürlich vor allem konstruktiv sein und in vernünftiger Zeit zu einem Ergebnis führen. Das hat häufig zur Folge, dass man sich mit einer Näherung der Lösung begnügen muss. Der in Kauf genommene Fehler wird **Verfahrensfehler** genannt (siehe z. B. iterative Verfahren in den Kapiteln 4, 5 und 6).

Die Realisierung einer numerischen Methode auf dem Computer führt zu einer weiteren Verfälschung der Ergebnisse. Zahlen lassen sich nicht exakt darstellen und die Operationen können nicht exakt durchgeführt werden. Fehler dieser Art werden als **Rundungsfehler** bezeichnet.

Steht schließlich ein Berechnungsprogramm zur Verfügung, so ist es ein (weiteres) Werkzeug, das zur Analyse oder Verbesserung von Produkten oder Prozessen eingesetzt werden kann. Durch Variation der Daten lassen sich Experimente am Computer durchführen, die gewonnenen Erkenntnisse ergänzen (gelegentlich ersetzen sogar) Kenntnisse aus realen Experimenten.

Schließlich erhält man konkrete Zahlen als Resultat eines Programms, deren Aussagekraft in Beziehung zu den gemachten Fehlern bewertet werden muss. Dies kann zu Modellmodifikationen, anderen numerischen Methoden oder deren besserer Implementierung auf einem Computer führen.

1.2 Ein Beispiel

Problemstellung:

Es soll die Temperatur in einem Metallstab bei stationärer Aufheizung bestimmt werden.

Modellierung:

Für die Modellierung wird angenommen, dass der Stab ein eindimensionales Kontinuum der Länge L ist, und dass der Wärmestrom nur in Längsrichtung zu berücksichtigen ist. Der Stab besitze eine konstante Wärmeleitzahl und innere stationäre Wärmequellen mit Dichte $q(x)$.

Die Energiebilanz in einem Kontrollvolumen am Ort x mit Querschnitt A und Dicke Δx muss folgende Terme berücksichtigen:

- Die aufgenommene Wärmemenge Q pro Zeit:

$$Q = \int_{x-\Delta x/2}^{x+\Delta x/2} q(\xi)\ d\xi \cdot A$$

- Die abgeführte Wärmemenge ΔJ pro Zeit durch Wärmeleitung im Stab:

$$\Delta J = \left[\sigma\left(x + \frac{\Delta x}{2}\right) - \sigma\left(x - \frac{\Delta x}{2}\right)\right] \cdot A,$$

  wobei $\sigma(x)$ die Wärmestromdichte bezeichnet. Für $\sigma(x)$ gilt nach dem Fourierschen Gesetz:

$$\sigma(x) = -\lambda \frac{dT}{dx}(x).$$

  $T(x)$ ist die Temperatur im Stab am Ort x, λ ist die Wärmeleitzahl des Stabes.

Unter diesen Annahmen führt die Bilanzierung der Energie

$$\Delta J = Q$$

auf die folgende Gleichung (Einsetzen der ersten drei Gleichungen):

$$-\lambda \frac{dT}{dx}\left(x + \frac{\Delta x}{2}\right) + \lambda \frac{dT}{dx}\left(x - \frac{\Delta x}{2}\right) = \int_{x-\Delta x/2}^{x+\Delta x/2} q(\xi)\ d\xi.$$

Falls die beteiligten Funktionen hinreichend glatt sind, erhält man nach Division durch Δx und dem Grenzübergang $\Delta x \to 0$ (zentraler Differenzenquotient für die 1.Ableitung und dem Mittelwertsatz der Integralrechnung) die stationäre Wärmeleitgleichung

$$-\lambda \frac{d^2T}{dx^2}(x) = q(x) \quad \text{für } x \in (0, L).$$

An den beiden Randpunkten des Stabes wird angenommen, dass gilt:

$$T(0) = T_a, \quad T(L) = T_b$$

mit bekannten Werten T_a und T_b.

Zusammenfassend erhält man zur Beschreibung der stationären Aufheizung eines Metallstabes ein so genanntes Randwertproblem für eine Differentialgleichung (siehe KV Gewöhnliche Differentialgleichungen und Dynamische Systeme 1).

Bei der Modellierung dieses Problems wurden viele Vereinfachungen vorgenommen, die zu Modellfehlern führen.

Mathematische Analyse:

Das Randwertproblem lässt sich auf folgende einfache Form transformieren (Skalierung):

Gesucht ist eine Funktion u mit

$$-u''(x) = f(x), \qquad x \in (0,1) \tag{1.1}$$
$$u(0) = u(1) = 0, \tag{1.2}$$

wobei f eine vorgegebene Funktion ist.

Im Rahmen der mathematischen Analyse ist die Korrektgestelltheit des Problems in geeigneten Funktionenräumen nachzuweisen. Dazu muss zuerst ein sinnvoller Lösungsbegriff festgelegt werden. Die obige Formulierung lässt erwarten, dass man die Lösung in einem Raum von zweimal differenzierbaren Funktionen suchen sollte. Dass das durchaus nicht notwendig ist, zeigt die folgende Neuformulierung der Problemstellung als so genannte Variationsgleichung:

Die obige Differentialgleichung für die Funktion u wird mit einer beliebigen stetig differenzierbaren Funktion v mit $v(0) = v(1) = 0$ multipliziert und über das Intervall $(0,1)$ integriert:

$$-\int_0^1 u''(x)v(x)\ dx = \int_0^1 f(x)v(x)\ dx.$$

Durch partielle Integration erhält man

$$\int_0^1 u'(x)v'(x)\ dx - \underbrace{u'(x)v(x)\Big|_0^1}_{=0} = \int_0^1 f(x)v(x)\ dx$$

und somit

$$\int_0^1 u'(x)v'(x)\ dx = \int_0^1 f(x)v(x)\ dx. \tag{1.3}$$

Daraus lässt sich die Problemstellung neu formulieren:

Gesucht ist eine Funktion u mit $u(0) = u(1) = 0$, sodass für alle Funktionen v mit $v(0) = v(1) = 0$ die Gleichung (1.3) gilt.

Diese Problemformulierung enthält nur noch erste Ableitungen. Man nennt ein solches Problem ein Variationsproblem. Die Korrektgestelltheit dieses Problems lässt sich relativ leicht in geeigneten Funktionenräumen von „differenzierbaren“ Funktionen (den so genannten Sobolev-Räumen) nachweisen (mit Hilfe des Satzes von Lax-Milgram).

Numerische Methode:

Das im Ort kontinuierliche (unendlichdimensionale) Problem (1.1), (1.2) wird durch ein im Ort diskretes (endlichdimensionales) Problem approximiert. Man spricht von Diskretisierung bezüglich der Variablen x:

Das kontinuierliche Intervall $[0, 1]$ wird durch eine endliche Punktmenge (Gitterpunkte), z.B. $x_i = ih$, $i = 0, 1, \ldots, N, N+1$ mit $h = 1/(N+1)$, ersetzt (nur als Beispiel, nicht notwendigerweise äquidistant).

Die 2. Ortsableitung in einem Gitterpunkt wird durch Differenzenquotienten approximiert (z. B.):

$$\begin{aligned} u''(x_i) &\approx \frac{u'\left(x_i + \frac{1}{2}h\right) - u'\left(x_i - \frac{1}{2}h\right)}{h} \\ &\approx \frac{1}{h}\left\{\frac{u(x_{i+1}) - u(x_i)}{h} - \frac{u(x_i) - u(x_{i-1})}{h}\right\} \\ &= \frac{1}{h^2}\left(u(x_{i-1}) - 2u(x_i) + u(x_{i+1})\right). \end{aligned}$$

Ersetzt man in der ursprünglichen Differentialgleichung die zweite Ortsableitung durch die obige Differenzenapproximation, so entsteht folgendes lineare Gleichungssystem:

$$-\frac{1}{h^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) = f(x_i) \quad \text{für } i = 1, 2, \ldots, N$$

mit den Randbedingungen

$$u_0 = u_{N+1} = 0$$

für die Unbekannten u_i, $i = 1, 2, \ldots, N$, die als Näherungen für die exakte Lösung $u(x_i)$ in den Punkten x_i interpretiert werden können.

Bemerkung: Die hier vorgestellte Methode der Diskretisierung bezüglich der Ortvariablen ist eine so genannte Finite Differenzen Methode (FDM). Sie ist nur eine von vielen Möglichkeiten, eine endliche Zahl von Ortsfreiheitsgraden zu erhalten. Eine andere Technik ist z.B. die Finite Elemente Methode (FEM) oder die Randelementmethode (BEM).

Beim Diskretisieren der Differentialgleichung entsteht ein Verfahrensfehler, von dem erwartet wird, dass er immer kleiner wird, je kleiner die Schrittweite h ist.

Eine kleine Schrittweite h bedeutet allerdings eine große Anzahl N von Unbekannten im Gleichungssystem. Die Systemmatrix ist außerdem dünnbesetzt (sehr viele Einträge sind 0). Daher sind geeignete Algorithmen für lineare Gleichungssysteme mit großer, dünnbesetzter Matrix gefragt.

Implementierung:

Bei der Umsetzung eines Algorithmus auf einem Computer ist zu berücksichtigen, dass bei vielen Operationen Rundungsfehler entstehen. Die Rechenzeit sollte erträglich sein.

Computerexperiment:

Steht ein effizientes Berechnungsprogramm einmal zur Verfügung, kann die Problemstellung am Computer simuliert werden. Anpassungen (z.B. Länge des Stabes, andere Wärmequellen) sind leicht möglich.

Interpretation der Resultate:

Die Interpretation der numerischen Resultate hat unter Berücksichtigung von möglichen Modellfehlern, Datenfehlern, Verfahrensfehlern und Rundungsfehlern zu erfolgen

1.3 Problemstellungen

Häufig führen Modelle auf Differentialgleichungsprobleme. Solche Problemstellungen können grob unterschieden werden in:

- stationäre und instationäre Probleme.

Aus den ursprünglich

- kontinuierlichen Problemen (unendlichdimensionalen Problemen)

entstehen nach der Diskretisierung

- diskrete Probleme (endlichdimensionale Probleme).

Die wichtigste Unterscheidung bezüglich der Komplexität von Problemen ist die Einteilung in

- lineare und nichtlineare Probleme.

Nach einem einführenden Kapitel über Besonderheiten des Numerischen Rechnens werden folgende typische Problemstellungen näher untersucht:

- Lineare Gleichungssysteme (Sie treten z.B. bei der Diskretisierung linearer stationärer Problemen auf, sie sind aber auch ein wichtiger Baustein bei komplizierteren Problemstellungen, siehe z. B. Linearisierung nichtlinearer Gleichungen.)
- Nichtlineare Gleichungen (zur Beschreibung nichtlinearer stationärer Probleme)
- Eigenwertprobleme (z.B. zur Beschreibung von zeitharmonischen Lösungen instationärer Probleme)
- Interpolation, Numerische Differentiation und Numerische Integration (als wichtige Hilfsproblemstellungen)

Kapitel 2

Besonderheiten des Numerischen Rechnens

2.1 Zahlendarstellung

Jede reelle Zahl $x \neq 0$ lässt sich im Dezimalsystem folgendermaßen darstellen:

$$x = \pm \left(\alpha_m 10^m + \alpha_{m-1} 10^{m-1} + \alpha_{m-2} 10^{m-2} + \ldots\right)$$

mit $m \in \mathbb{Z}$, $\alpha_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $\alpha_m \neq 0$.

Schreibweise: $x = \pm \alpha_m \ldots \alpha_1 \alpha_0, \alpha_{-1}\alpha_{-2} \ldots$ für $m \geq 0$, $x = \pm 0, 0 \ldots 0 \alpha_m \alpha_{m-1} \alpha_{m-2} \ldots$ für $m < 0$ mit $|m-1|$ Nullen zwischen Komma und α_m.

Diese Darstellung lässt sich nicht nur für die Zahl 10 sondern allgemein für eine beliebige positive ganze Zahl $B \neq 1$ durchführen:

$$x = \pm \left(\alpha_m B^m + \alpha_{m-1} B^{m-1} + \alpha_{m-2} B^{m-2} + \ldots\right)$$

mit $m \in \mathbb{Z}$, $\alpha_i \in \{0, 1, \ldots, B-1\}$, $\alpha_m \neq 0$. Die Darstellung ist im allgemeinen nicht eindeutig.

Man nennt B die Basis des Zahlensystems, für die Zahlen $0, 1, \ldots, B-1$ werden spezielle Symbole, die Ziffern des Zahlensystems, verwendet. Die Darstellung muss nicht eindeutig sein.

Neben dem Dezimalsystem ($B = 10$, Ziffern 0, 1, 2, 3, 4, 5, 6, 7, 8, 9) sind im Zusammenhang mit Computern vor allem das Dualsystem ($B = 2$, Ziffern 0, 1 oder 0, L) und das Hexadezimalsystem ($B = 16$, Ziffern 0, 1, ... ,9 , A, ... , F) in Verwendung.

Am Computer wird für reelle Zahlen $x \neq 0$ die normalisierte Gleitkommadarstellung verwendet:

$$x = \pm 0, \alpha_1 \alpha_2 \ldots \alpha_t \cdot B^e \tag{2.1}$$

mit $\alpha_i \in \{0, 1, \ldots, B-1\}$ und $\alpha_1 \neq 0$. Dabei ist $B \in \mathbb{N} - \{1\}$ die Basis des verwendeten Zahlensystems. $m = 0, \alpha_1 \alpha_2 \ldots \alpha_t$ heißt die Mantisse, t die Mantissenlänge. Der Exponent e ist eine ganze Zahl zwischen vorgegebenen Schranken: $U \leq e \leq O$.

Beispiel: Für den Datentyp `float` (einfache Genauigkeit) in C wird meistens eine Zahlendarstellung mit $B = 2$, $t = 24$, $U = -125$ und $O = 128$ verwendet. Das Rechnen mit doppelter Genauigkeit (Datentyp `double`) basiert auf der Setzung $B = 2$, $t = 53$, $U = -1021$ und $O = 1024$. 24 (53) Dualstellen entsprechen etwa 7 (15) Dezimalstellen.

Das Überschreiten bzw. Unterschreiten des Exponentenbereiches wird als overflow bzw. underflow bezeichnet. Während manchmal ein underflow nicht angezeigt wird und der Computer 0 oder die kleinste positive oder negative darstellbare Zahl verwendet, führt ein overflow im Allgemeinen zum Programmabbruch. Wir werden im Weiteren das Auftreten von underflows oder overflows nicht berücksichtigen.

Die Zahl 0 und alle Zahlen der Form (2.1) bilden die Menge $\mathcal{M}$ aller Maschinenzahlen. Nachdem $\mathcal{M}$ nur endlich viele Zahlen enthält, kann natürlich nicht jede reelle Zahl am Rechner exakt dargestellt werden. Man muss runden, d.h., jede reelle Zahl x wird durch eine geeignete Maschinenzahl $rd(x)$ approximiert.

Die bestmögliche Approximation $rd(x)$ einer reellen Zahl x erfüllt die Bedingung

$$|x - rd(x)| \leq |x - y| \quad \text{für alle Maschinenzahlen } y. \tag{2.2}$$

Diese Forderung lässt sich leicht realisieren: Für

$$x = \pm 0, \alpha_1 \alpha_2 \dots \alpha_t \alpha_{t+1} \dots \cdot B^e$$

mit $\alpha_1 \neq 0$ rundet man auf, falls $\alpha_{t+1} \geq B/2$, bzw. rundet man ab, falls $\alpha_{t+1} < B/2$ (kaufmännisches Runden).

Bezeichnung: Sei $\bar{x}$ eine Approximation einer reellen Zahl x. Dann heißt

$$\Delta x = \bar{x} - x$$

absoluter Fehler und, falls $x \neq 0$,

$$\varepsilon_x = \frac{\bar{x} - x}{x}$$

relativer Fehler. Natürlich gilt:

$$\Delta x = \varepsilon_x x \quad \text{und} \quad \bar{x} = x(1 + \varepsilon_x)$$

Wir werden auch den Betrag dieser Ausdrücke, d. h. , $|\bar{x} - x|$ als absoluten Fehler bzw. $|(\bar{x} - x)/x|$ als relativen Fehler bezeichnen.

Falls für den absoluten Fehler gilt:

$$|\bar{x} - x| \leq \frac{1}{2} B^{-s},$$

so heißt die Ziffer mit dem Stellenwert B^{-s} gültig. Die gültigen Ziffern ohne die führenden Nullen heißen signifikante Ziffern. Die gültigen Ziffern beschreiben den absoluten Fehler, die Anzahl der signifikanten Ziffern den relativen Fehler.

Der relative Fehler ist im Allgemeinen aussagekräftiger, weil er den Fehler relativ zum exakten Wert misst. Allerdings gibt es Schwierigkeiten in der Gegend von 0.

Es lässt sich leicht zeigen, dass für das Runden nach (2.2) gilt:

$$\frac{|rd(x) - x|}{|x|} \leq \text{eps} \tag{2.3}$$

mit $\text{eps} = \frac{1}{2}B^{1-t}$, der so genannten **Maschinengenauigkeit**, oder anders ausgedrückt:

$$rd(x) = x(1 + \varepsilon) \quad \text{mit } |\varepsilon| \leq \text{eps}.$$

Beweis. Der Einfachheit halber gelte $x > 0$. Sei $x = mB^e$ eine reelle Zahl mit normalisierter Mantisse m, also $B^{-1} \leq |m| < 1$, und sei $rd(x)$ die gerundete normalisierte Gleitkommazahl mit Mantissenlänge t. Man sieht sofort, dass gilt:

$$|rd(x) - x| \leq \frac{1}{2}B^{-t}B^e \quad \text{und} \quad |x| \geq B^{-1}B^e.$$

Also folgt durch Division:

$$\frac{|rd(x) - x|}{|x|} \leq \frac{1}{2}B^tB^eB^1B^{-e} = \frac{1}{2}B^{1-t} = \text{eps}$$

□

Der relative Fehler, der beim Runden entsteht, ist also durch die Maschinengenauigkeit eps nach oben beschränkt.

Beispiel: Für das Rechnen mit einfacher Genauigkeit und der obigen Form des Rundens erhält man $\text{eps} = 2^{-24} \approx 10^{-7}$, mit doppelter Genauigkeit $\text{eps} = 2^{-53} \approx 10^{-16}$.

Eine andere Möglichkeit des Rundens ist das Abschneiden der Mantisse nach t Ziffern, d.h., man rundet immer ab. In diesem Fall gilt die Abschätzung (2.3) für die Maschinengenauigkeit $\text{eps} = B^{1-t}$.

2.2 Gleitkommaarithmetik

Die Addition zweier Maschinenzahlen x, y muss nicht wieder eine Maschinenzahl ergeben. Man fordert nun, dass die auf der Maschine verfügbare „Addition“, die so genannte Gleitkommaaddition, die im Weiteren mit dem Symbol $+^*$ bezeichnet wird, so genau ist, dass gilt:

$$x +^* y = rd(x + y).$$

Diese Forderung lässt sich leicht konstruktiv realisieren: Gegeben seien zwei Maschinenzahlen $x = m_1 \cdot 10^{e_1}$ und $y = m_2 \cdot 10^{e_2}$. Der Einfachheit halber sei angenommen, dass $x \geq y > 0$. (Die anderen Fälle lassen sich analog behandeln.) Die Addition lässt sich in 4 Teilschritten durchführen:

1. Exponentenangleich: $d = e_1 - e_2$
2. Mantissenverschiebung: $m_2' = m_2 \cdot 10^{-d}$, falls $d \leq t$ bzw. $m_2' = 0$ sonst.
3. Mantissenaddition: $m = m_1 + m_2'$
4. Runden

Die Genauigkeit dieser Gleitkommaaddition lässt sich folgendermaßen abschätzen (für $x + y \neq 0$):

$$\frac{|(x +^* y) - (x + y)|}{|x + y|} = \frac{|rd(x + y) - (x + y)|}{|x + y|} \leq \text{eps},$$

oder anders ausgedrückt:

$$x +^* y = (x + y)(1 + \varepsilon) \quad \text{mit } |\varepsilon| \leq \text{eps}.$$

Der relative Fehler der Gleitkommaaddition ist also durch eps beschränkt.

Die Gleitkommaaddition erfüllt nicht alle Rechengesetze der „normalen“ Addition. So ist z.B. das Assoziativgesetz im Allgemeinen nicht mehr erfüllt:

$$x +^* (y +^* z) \neq (x +^* y) +^* z,$$

d.h., die Reihenfolge der Summanden beeinflusst das Resultat.

Auch alle anderen arithmetischen Operationen (Subtraktion, Multiplikation und Division) und weitere einfache binäre Operationen sind am Computer durch entsprechende Gleitkommaoperationen realisiert. Sei $\circ$ eine dieser Operationen und bezeichne $\circ^*$ die dazugehörige Gleitkommaoperation. Dann fordern wir:

$$x \circ^* y = rd(x \circ y),$$

woraus wie oben für die Genauigkeit folgt:

$$\frac{|(x \circ^* y) - (x \circ y)|}{|x \circ y|} \leq \text{eps}. \tag{2.4}$$

Am Computer stehen auch einfache Funktionen $f(x)$ wie $\sin x$, $\sqrt{x}$, $\log x$, ..., zur Verfügung. Es wird im Weiteren davon ausgegangen, dass die Realisierungen $f^*(x)$ dieser Funktionen die Abschätzung

$$\frac{|f^*(x) - f(x)|}{|f(x)|} \leq \text{eps} \tag{2.5}$$

erfüllen. Diese Abschätzung trifft in der Realität oft nicht zu, sondern gilt nur mit einer oberen Schranke von $K \cdot \text{eps}$ mit $K \leq 10$.

Man nennt die am Rechner realisierten Operationen mit den Abschätzungen (2.4), (2.5) **elementare Operationen**.

Bemerkung: Die in diesem und in dem vorigen Abschnitt gemachten Annahmen über die Genauigkeit der Zahlendarstellung und der am Rechner implementierten Operationen sind eine Idealisierung der realen Situation, die allerdings für die Zwecke einer Fehleranalyse zumindest größenordnungsmäßig zutreffende Aussagen erlauben.

2.3 Rechengeschwindigkeit

Eine Gleitkommaoperation (z.B.: $x = x + y$, aber auch eine zusammengesetzte Operation, z.B. $x = a * x + y$) wird als ein FLOP (floating point operation) bezeichnet. Für numerische Zwecke wird die Rechengeschwindigkeit durch die Anzahl der FLOPs, die eine Rechenanlage pro Sekunde durchführt, angegeben (kurz: FLOPS, floating point operations per second). Gebräuchlichere Einheiten sind 1 MFLOPS (MegaFLOPS) = 10^6 FLOPS und 1 GFLOPS (GigaFLOPS) = 1000 MFLOPS.

Die Angabe der Anzahl der MFLOPS charakterisiert die Rechengeschwindigkeit eines skalaren Rechners sehr gut. PCs erreichen FLOP-Raten der Größenordnung 10^2 MFLOPS bis 1 GFLOPS.

Auf einem Vektorrechner werden gewisse Operationenfolgen (vektorisierbare Operationen) wesentlich schneller durchgeführt.

Beispiel: (Pipeline-Prinzip bei der Addition) Unter der vereinfachenden Annahme, dass die Addition in 4 Segmente aufgeteilt wird (siehe vorher) und der Rechner für die Durchführung jedes Segments die gleiche Zeiteinheit (einen Takt) benötigt, dauert eine Addition 4 Zeiteinheiten. Verlässt ein Operandenpaar das erste Segment, kann ein zweites Operandenpaar bereits in das erste Segment nachrücken, u.s.w. Für eine Schleife

```
for (i = 1; i <= n; i++)
  x(i) = x(i) + y(i);
```

gilt daher: Die erste Addition benötigt 4 Zeiteinheiten, dann wird in jeder weiteren Zeiteinheit eine weitere Addition fertig.

Die Beschleunigung wird allerdings nur dann voll wirksam, wenn es keine Datenabhängigkeiten der einzelnen Operationen gibt.

Man unterscheidet auf einem Vektorrechner zwischen der skalaren Leistung und der Spitzenleistung für vektorisierbare Operationen vom obigen Typ. Vektorrechner erreichen eine Spitzenleistung in der Größenordnung von GFLOPS. Je nach Anteil der vektorisierbaren Operationen (Vektorisierungsgrad) liegt die tatsächliche Rechengeschwindigkeit zwischen diesen Werten.

Ein Parallelrechner besteht aus mehreren Prozessoren. Zur Bewertung der Leistung eines Parallelrechners kommt es auf die Anzahl der Prozessoren, die Leistung der einzelnen Prozessoren und auf die Kommunikationsleistung zwischen den Prozessoren an. Die tatsächliche Rechengeschwindigkeit ergibt sich dann vor allem aus dem Anteil der parallel ausführbaren Programmteile (Parallelisierungsgrad), dem Kommunikationsaufwand und der Synchronität des Rechenablaufs.

2.4 Fehleranalyse

Man unterscheidet grob drei Fehlerarten:

- Verfahrensfehler,
- Datenfehler und
- Rundungsfehler.

Im Folgenden werden die Datenfehleranalyse und die Rundungsfehleranalyse näher diskutiert. Eine Behandlung von Verfahrensfehlern ist nur für jedes Verfahren individuell möglich und erfolgt daher in den einzelnen Kapiteln.

Ein(e) mathematische(s) Problem(stellung) lässt sich (zumindest formal) in folgende Form bringen: Die gesuchten Größen $y \in \mathbb{R}^m$ sollen aus gegebenen Größen (den Daten) $x \in \mathbb{R}^n$ über eine gegebene Vorschrift $\varphi\colon \mathbb{R}^n \to \mathbb{R}^m$ berechnet werden, kurz:

$$y = \varphi(x).$$

Wir untersuchen zuerst, wie sich eventuell vorhandene Störungen in den Daten auf die gesuchten Größen auswirken:

2.4.1 Datenfehleranalyse

Sei x der Vektor der exakten Daten und $\bar{x} = x + \Delta x$ ein Vektor von verfälschten Daten. Entsprechend bezeichnet $y = \varphi(x)$ das exakte Ergebnis und $\bar{y} = y + \Delta y = \varphi(\bar{x})$ das Ergebnis der verfälschten Daten.

Wir nehmen im Weiteren an, dass φ (2-mal) stetig differenzierbar ist. Dann gilt nach dem Satz von Taylor für den absoluten Fehler:

$$\Delta y = \varphi(x + \Delta x) - \varphi(x) \approx J_\varphi(x) \cdot \Delta x \tag{2.6}$$

bzw. komponentenweise

$$\Delta y_i = \varphi_i(x + \Delta x) - \varphi_i(x) \approx \sum_{j=1}^{n} \frac{\partial \varphi_i}{\partial x_j}(x) \cdot \Delta x_j \quad \text{für} \quad i = 1, \ldots, m \tag{2.7}$$

bzw. für den relativen Fehler, falls $y_i \neq 0$:

$$\varepsilon_{y_i} = \frac{\Delta y_i}{y_i} \approx \sum_{j=1}^{n} \frac{x_j}{\varphi_i(x)} \frac{\partial \varphi_i}{\partial x_j}(x) \cdot \varepsilon_{x_j}. \tag{2.8}$$

Die Verstärkungsfaktoren $k_{ij} = (x_j/\varphi_i(x))\partial\varphi_i/\partial x_j(x)$ heißen die Konditionszahlen des Problems.

Ein Problem heißt **gut konditioniert** (oder auch: **datenstabil**), wenn kleine Fehler der Daten nur kleine Störungen der Lösung bewirken, d.h. $\forall\, i, j \;:\; |k_{ij}| \leq 1$. Andernfalls heißt das Problem **schlecht konditioniert** (**dateninstabil**).

Nach der Fehlerformel (2.8) ist ein Problem dann gut (bzw. schlecht) konditioniert, wenn die Konditionszahlen des Problems klein (bzw. groß) im Vergleich mit 1 sind.

Bezeichnung: Die Fehlerformel (2.7) lässt sich in Matrix-Vektor-Schreibweise folgendermaßen kurz darstellen:

$$\Delta y \approx \varphi'(x)\Delta x$$

mit

$$\Delta y = \begin{pmatrix} \Delta y_1 \\ \Delta y_2 \\ \vdots \\ \Delta y_m \end{pmatrix},\ \Delta x = \begin{pmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{pmatrix},\ \varphi'(x) = \begin{pmatrix} \frac{\partial \varphi_1}{\partial x_1}(x) & \frac{\partial \varphi_1}{\partial x_2}(x) & \cdots & \frac{\partial \varphi_1}{\partial x_n}(x) \\ \frac{\partial \varphi_2}{\partial x_1}(x) & \frac{\partial \varphi_2}{\partial x_2}(x) & \cdots & \frac{\partial \varphi_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \varphi_m}{\partial x_1}(x) & \frac{\partial \varphi_m}{\partial x_2}(x) & \cdots & \frac{\partial \varphi_m}{\partial x_n}(x) \end{pmatrix}.$$

Beispiel: Für die Addition zweier Zahlen:

$$y = \varphi(x_1, x_2) = x_1 + x_2$$

folgt nach Formel (2.8):

$$\varepsilon_{x_1+x_2} = \frac{x_1}{x_1 + x_2} \cdot \varepsilon_{x_1} + \frac{x_2}{x_1 + x_2} \cdot \varepsilon_{x_2}.$$

Die Konditionszahlen sind besonders groß, wenn gilt: $x_1 \approx -x_2$. Diesen Effekt der großen Verstärkung von Datenfehlern bei der Addition zweier Zahlen mit $x_1 \approx -x_2$ nennt man **Auslöschung**.

Analog gilt für die Subtraktion:

$$\varepsilon_{x_1-x_2} = \frac{x_1}{x_1 - x_2} \cdot \varepsilon_{x_1} - \frac{x_2}{x_1 - x_2} \cdot \varepsilon_{x_2}.$$

Der kritische Fall der Auslöschung tritt hier für $x_1 \approx x_2$ ein.

In diesen Spezialfällen sind die Fehlerformeln sogar exakt. Es entsteht kein Fehler bei der Verwendung der Taylor-Formel, da höhere als erste Ableitungen von φ verschwinden.

Beispiel: Für die Multiplikation zweier Zahlen

$$y = \varphi(x_1, x_2) = x_1 \cdot x_2$$

folgt nach Formel (2.8):

$$\varepsilon_{x_1 \cdot x_2} \approx 1 \cdot \varepsilon_{x_1} + 1 \cdot \varepsilon_{x_2}.$$

Die Konditionszahlen sind hier 1, das Problem ist gut konditioniert.

Analog gilt für die Division:

$$\varepsilon_{x_1/x_2} \approx 1 \cdot \varepsilon_{x_1} - 1 \cdot \varepsilon_{x_2}.$$

Die Division zweier Zahlen ist ebenfalls ein gut konditioniertes Problem.

Beispiel: Die Berechnung einer Lösung der quadratischen Gleichung

$$y^2 + 2py - q = 0$$

führt auf das Problem:

$$y = \varphi(p, q) = \sqrt{p^2 + q} - p.$$

Aus der Fehlerformel (2.8) und der Identität

$$\sqrt{p^2 + q} - p = \frac{q}{\sqrt{p^2 + q} + p}$$

erhält man

$$\varepsilon_y \approx -\frac{p}{\sqrt{p^2 + q}} \cdot \varepsilon_p + \frac{\sqrt{p^2 + q} + p}{2\sqrt{p^2 + q}} \cdot \varepsilon_q.$$

Das Problem ist für den Fall $p > 0$ und $q > 0$ sicherlich gut konditioniert, da in diesem Fall die Konditionszahlen betragsmäßig durch 1 abgeschätzt werden können.

2.4.2 Rundungsfehleranalyse

Ein Algorithmus zur Lösung des Problems

$$y = \varphi(x)$$

lässt sich in elementare Operationen (Operationen, die am Rechner zur Verfügung stehen und eine relative Genauigkeit von eps besitzen) zerlegen:

$$x = x^{(0)} \mapsto x^{(1)} \mapsto x^{(2)} \mapsto \ldots \mapsto x^{(r)} \mapsto x^{(r+1)} = y$$

Die Abbildung, die das Zwischenergebnis $x^{(s)}$ auf y abbildet, wird mit $\psi^{(s)}$ bezeichnet und heißt Restabbildung, wobei $s = 0, 1, 2, \ldots, r$.

Beispiel: Ein möglicher Algorithmus zur Berechnung von

$$y = \varphi(p, q) = \sqrt{p^2 + q} - p$$

ist durch folgende Einzelschritte gegeben:
Algorithmus 1:

1. $r = p^2$
2. $s = r + q$
3. $t = \sqrt{s}$
4. $y = t - p$

In diesem Fall gibt es 3 Zwischenergebnisse:

$$x^{(1)} = r = p^2, \quad x^{(2)} = s = p^2 + q, \quad x^{(3)} = t = \sqrt{p^2 + q}.$$

Die dazugehörigen Restabbildungen lauten:

$$\psi^{(1)}(r(,p,q)) = \sqrt{r+q} - p, \quad \psi^{(2)}(s(,p)) = \sqrt{s} - p, \quad \psi^{(3)}(t(,p)) = t - p.$$

In der Parameterliste der Restabbildungen ist der Vollständigkeit halber in Klammern auch die jeweilige Abhängigkeit von p bzw. q angeführt. Für die Fehleranalyse ist aber nur die Abhängigkeit der Restabbildung $\psi^{(s)}$ vom Zwischenergebnis $x^{(s)}$ maßgeblich.

Nach der Fehlerformel (2.7) verfälschten Datenfehler $\Delta x^{(0)}$ das Endergebnis näherungsweise um den Beitrag $\varphi'(x)\Delta x^{(0)}$.

Bei der Berechnung des Zwischenergebnisses $x^{(s)}$ für $s = 1, 2, \ldots, r$ entsteht ein neuer absoluter Fehler $\Delta x^{(s)}$, der laut (2.7) zu einer Verfälschung des Endergebnisses um näherungsweise $(\psi^{(s)})'(x^{(s)})\Delta x^{(s)}$ führt, wenn man annimmt, dass alle anderen Operationen exakt ausgeführt werden.

Schließlich entsteht noch ein weiterer Fehler $\Delta x^{(r+1)}$ bei der letzten elementaren Operation.

In erster Ordnung ist es gerechtfertigt, den Gesamteinfluss der einzelnen Rundungsfehler durch die Addition der oben beschriebenen Einzeleffekte zu erfassen. Somit erhält man für den Gesamtrundungsfehler Δy^R näherungsweise:

$$\Delta y^R \approx \varphi'(x)\Delta x^{(0)} + \sum_{s=1}^{r} (\psi^{(s)})'(x^{(s)})\Delta x^{(s)} + \Delta x^{(r+1)}.$$

Der Gesamtrundungsfehler setzt sich aus zwei Anteilen zusammen, dem unvermeidbaren Fehler

$$\Delta y^0 = \varphi'(x)\Delta x^{(0)},$$

der sich aus der Datenfehlerfortpflanzung ergibt und der unabhängig vom gewählten Algorithmus ist, und dem restlichen Rundungsfehler

$$\Delta y^r = \sum_{s=1}^{r} (\psi^{(s)})'(x^{(s)})\Delta x^{(s)} + \Delta x^{(r+1)},$$

der vom gewählten Algorithmus abhängt.

Ein Algorithmus heißt **numerisch stabil**, wenn der restliche Rundungsfehler Δy^r den unvermeidbaren Fehler Δy^0 nicht dominiert.

Die Fehler $\Delta x^{(s)}$ für $s = 0, 1, \ldots, r, r+1$ lassen sich mit Hilfe der Maschinengenauigkeit abschätzen, siehe die Abschnitte über die Größe des Rundungsfehlers bei der Eingabe und bei elementaren Operationen:

$$|\Delta x^{(s)}| \leq \text{eps}\,|x^{(s)}| \quad \text{für } s = 0, 1, \ldots, r+1.$$

Daraus ergeben sich folgende Abschätzungen

$$|\Delta y^0| \leq \text{eps}\,|\varphi'(x)||x| \quad \text{und} \quad |\Delta y^r| \leq \text{eps}\left[\sum_{s=1}^{r} |(\psi^{(s)})'(x^{(s)})||x^{(s)}| + |y|\right].$$

Zur Beurteilung der numerischen Stabilität eines Algorithmus vergleicht man im Allgemeinen diese oberen Schranken für den unvermeidbaren Fehler bzw. den restlichen Rundungsfehler.

Beispiel: Der obige Algorithmus zur Berechnung von $y = \sqrt{p^2+q} - p$ führt im Fall $p > 0$, $q > 0$, $p^2 \gg q$ auf folgende Abschätzungen:

$$\begin{aligned} |\Delta y^0| &\leq \text{eps}\left(\left|\frac{p}{\sqrt{p^2+q}} - 1\right| \cdot p + \frac{1}{2\sqrt{p^2+q}} \cdot q\right) \\ &= \text{eps}\left(\frac{q\,p}{(\sqrt{p^2+q}+p)\sqrt{p^2+q}} + \frac{q}{2\sqrt{p^2+q}}\right) \\ &\approx \text{eps}\left(\frac{q}{2p} + \frac{q}{2p}\right) = \text{eps}\,\frac{q}{p} \end{aligned}$$

und

$$\begin{aligned} |\Delta y^r| &\leq \text{eps}\left(\frac{1}{2\sqrt{r+q}} \cdot r + \frac{1}{\sqrt{s}} \cdot s + t + y\right) \\ &= \text{eps}\left(\frac{p^2}{2\sqrt{p^2+q}} + \frac{1}{2}\sqrt{p^2+q} + \sqrt{p^2+q} + \frac{q}{\sqrt{p^2+q}+p}\right) \\ &\approx \text{eps}\left(\frac{1}{2}p + \frac{1}{2}p + p + \frac{q}{2p}\right) \approx 2\,\text{eps}\,p \end{aligned}$$

Für den Fall $p^2 \gg q$ folgt $p \gg q/p$. Es dominiert also der restliche Rundungsfehler den unvermeidbaren Fehler. Der Algorithmus ist in diesem Fall numerisch instabil.

Auch ohne detaillierte Fehleranalyse sieht man, dass es im letzten Schritt des Algorithmus zur Auslöschung kommt. Die Subtraktion selbst ist harmlos. Aber die an sich kleinen Rundungsfehler, die in den ersten 3 Schritten des Algorithmus entstehen, werden stark verstärkt.

Ein für den Fall $p^2 \gg q$ stabiler Algorithmus wird durch die mathematisch äquivalente Berechungsformel

$$y = \frac{q}{\sqrt{p^2+q}+p}$$

nahegelegt.

Bemerkung: Um den unterschiedlichen Rundungsfehlereinfluss zweier Algorithmen zu untersuchen, genügt es, die jeweiligen restlichen Rundungsfehler zu vergleichen.

Beispiel: (Numerische Differentiation) Die erste Ableitung einer Funktionen lässt sich z.B. mit Hilfe eines zentralen Differenzenquotienten approximieren:

$$f'(x) \approx \frac{1}{2h}[f(x+h) - f(x-h)].$$

Der Gesamtfehler setzt sich hier aus dem Verfahrensfehler und dem Rundungsfehler zusammen. Der Verfahrensfehler, also der Unterschied zwischen der exakten Ableitung und dem (rundungsfehlerfrei berechneten) Differenzenquotienten, wird umso kleiner, je kleiner die Schrittweite h ist, der Rundungsfehler steigt hingegen mit kleiner werdender Schrittweite (Auslöschung). Die Abbildung 2.1 zeigt dieses gegenläufige Verhalten am Beispiel

$$f(x) = \sin x.$$

Für den Differenzenquotienten gilt hier

$$\frac{1}{2h}[f(x+h) - f(x-h)] = \frac{1}{2h}[\sin(x+h) - \sin(x-h)] = \cos x \frac{\sin h}{h}.$$

Der letzte Ausdruck ermöglicht für dieses Beispiel die Vermeidung der Auslöschung und damit eine (fast) rundungsfehlerfreie Auswertung des Differenzenquotienten. Wie Abbildung 2.1 zeigt, fällt der Verfahrensfehler proportional zu h^2, während der Rundungsfehler proportional zu $1/h$ steigt. Die optimale Schrittweite liegt etwa bei $\text{eps}^{1/3}$, siehe Abbildung 2.2.

Abbildung 2.1: Numerische Differentiation: Verfahren- und Rundungsfehler

Abbildung 2.2: Numerische Differentiation: Gesamtfehler

Kapitel 3

Direkte Verfahren zur Lösung linearer Gleichungssysteme

Lineare Gleichungssysteme entstehen z.B. bei der **Diskretisierung** von linearen (partiellen) Differentialgleichungen, die bei der Beschreibung zahlreicher physikalisch-technischer Probleme auftreten. Aber auch die Lösung nichtlinearer Probleme wird häufig auf die Lösung einer Folge von linearen Gleichungssystemen zurückgeführt (**Linearisierung**).

3.1 Ein Beispiel

In der Einleitung wurde das Problem der Berechnung einer stationären Temperaturverteilung in einem Stab diskutiert, das auf ein Randwertproblem der folgender Art führt:

Gesucht ist eine Funktion u mit

$$\begin{aligned} -u''(x) &= f(x), \qquad x \in (0,1) \\ u(0) &= u(1) = 0, \end{aligned}$$

wobei f eine vorgegebene Funktion ist.

Durch Diskretisierung (Finite Differenzen Methode) entstand folgendes lineare Gleichungssystem:

$$-\frac{1}{h^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) = f(x_i) \quad \text{für } i = 1, 2, \ldots, N$$

mit den Randbedingungen

$$u_0 = u_{N+1} = 0$$

für die Unbekannten u_i, $i = 1, 2, \ldots, N$, die als Näherungen für die exakte Lösung $u(x_i)$ in den Punkten x_i interpretiert werden können.

Man erhält also ein lineares Gleichungssystem

$$K_h \underline{u}_h = \underline{f}_h$$

mit der Matrix

$$K_h = (K_{ij})_{i,j=1,2,\ldots,N} = \frac{1}{h^2} \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}$$

und den Vektoren

$$\underline{u}_h = (u_i)_{i=1,2,\ldots,N}, \quad \underline{f}_h = (f_i)_{i=1,2,\ldots,N} \text{ mit } f_i = f(x_i).$$

3.2 Datenstabilität

Wir betrachten nun folgende allgemeine Problemstellung:
Gegeben ist eine Matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ und ein Vektor $b = (b_1, b_2, \ldots, b_n)^T \in \mathbb{R}^n$. Gesucht ist ein Vektor $x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$ mit

$$Ax = b.$$

Im Sinne der allgemeinen Diskussion in Kapitel 2 ist das Problem, aus gegebenen Daten A und b die gesuchte Lösung x auszurechnen, rein formal durch

$$x = A^{-1}b = \varphi(A, b)$$

gegeben. Verfälschte Daten $\bar{b} = b+\Delta b$ und $\bar{A} = A+\Delta A$ führen auf ein verfälschtes Ergebnis $\bar{x} = x + \Delta x$

Die in Kapitel 2 vorgestellte Methode der Beurteilung der Datenstabilität würde es erforderlich machen, für jede der n Komponenten der Lösung und jede der $n^2 + n$ Komponenten der Daten eine Konditionszahl zu berechnen und abzuschätzen, eine völlig unübersichtliche Situation.

Besser ist es, mit Hilfe von Normen die Größe eines Vektors oder einer Matrix auf jeweils nur eine Zahl zu komprimieren.

So lässt sich statt der n komponentenweise gebildeten relativen Fehler $\varepsilon_{x_j} = (\bar{x}_j - x_j)/x_j$ die Abweichung im Ergebnis auch durch eine einzige Zahl (relativ gut) messen:
$\varepsilon_x = \|\bar{x} - x\|_2/\|x\|_2$.
Dabei bezeichnet $\|x\|_2$ die Euklidische Länge eines Vektors x:

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

Analog misst man den relativen Fehler der rechten Seite b: $\varepsilon_b = \|\bar{b} - b\|_2/\|b\|_2$.

Anstelle der Euklidischen Norm kann auch ein andere Vektornormen verwendet werden.

Beispiel: Will man vor allem den komponentenweisen maximalen Fehler erfassen, bietet sich die Maximumnorm an:

$$\|x\|_\infty = \max_{i=1,2,\ldots,n} |x_i|.$$

Beispiel: Lassen sich die Komponenten eines Vektors z.B. als einzelne Massendefekte interpretieren, so ist man gelegentlich am Gesamtmassendefekt interessiert. Das führt auf die Verwendung der Betragssummennorm:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|.$$

Beispiel: Alle oben genannten Normen sind Spezialfälle der l_p-Norm

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

für alle $p \in [1, \infty)$. Der Fall $p = \infty$ ist als Grenzfall $\|x\|_\infty = \lim_{p\to\infty} \|x\|_p$ zu verstehen.

Beispiel: Die Euklidische Norm eines Vektors lässt sich mit Hilfe des Euklidischen Skalarprodukts darstellen:

$$\|x\|_2 = \sqrt{(x,x)_2} \quad \text{mit} \quad (x,y)_2 = \sum_i x_i y_i.$$

Für jede symmetrische und positiv definite Matrix A lässt sich durch

$$(x,y)_A = (Ax,y)_2 = \sum_{i,j} a_{ij} x_i y_j$$

ein Skalarprodukt und damit eine Norm definieren:

$$\|x\|_A = \sqrt{(x,x)_A}$$

Alle diese Vektornormen in $\mathbb{R}^n$ erfüllen die drei für eine Norm charakteristischen Eigenschaften (vgl. VL Analysis I):

1. Definitheit: $\|v\| \geq 0$ und $\|v\| = 0$ nur wenn $v = 0$,

2. Homogenität: $\|\lambda v\| = |\lambda| \|v\|$ für alle reellen Zahlen λ,

3. Dreiecksungleichung: $\|v + w\| \leq \|v\| + \|w\|$.

Auch die Größe von Matrizen lässt sich durch Normen (Matrixnormen) messen. Wenn Matrix- und Vektornormen gemeinsam verwendet werden, fordert man gewisse Verträglichkeitsbedingungen, vor allem soll folgende Eigenschaft gelten:

$$\|Ax\| \leq \|A\| \, \|x\| \quad \text{für alle } x \in \mathbb{R}^n.$$

Man sieht sofort, dass diese Eigenschaft für folgende Definition von $\|A\|$ erfüllt ist:

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

Man nennt diese Norm die der entsprechenden Vektornorm zugeordnete Matrixnorm (Operatornorm).

Tatsächlich stellt sich heraus, dass durch diese Definition eine Matrixnorm entsteht, dass also die für eine Norm charakteristischen Eigenschaften (Definitheit, Homogenität und Dreiecksungleichung) erfüllt sind.

Darüber hinaus gelten zusätzlich noch die folgenden Rechenregeln für jede Vektornorm und die zugeordnete Matrixnorm:

1. Die Matrixnorm ist passend zur Vektornorm, d.h.:

$$\|Ax\| \leq \|A\| \, \|x\| \quad \text{für alle } A \in \mathbb{R}^{n \times n} \text{ und alle } x \in \mathbb{R}^n.$$

2. Die Matrixnorm ist submultiplikativ, d.h.:

$$\|AB\| \leq \|A\| \, \|B\| \quad \text{für alle } A, B \in \mathbb{R}^{n \times n}.$$

3. Für die Einheitsmatrix gilt: $\|I\| = 1$.

Beispiele:

1. Die der Euklidischen Norm $\|x\|_2$ zugeordnete Matrixnorm lässt sich folgendermaßen darstellen:

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)},$$

wobei $\lambda_{\max}(B)$ den größten Eigenwert einer Matrix B bezeichnet. Sie heißt Spektralnorm.

2. Die der Maximumsnorm $\|x\|_\infty$ zugeordnete Matrixnorm lässt sich folgendermaßen darstellen:

$$\|A\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}|.$$

Sie heißt Zeilenbetragssummennorm.

3. Die der Betragssummennorm $\|x\|_1$ zugeordnete Matrixnorm lässt sich folgendermaßen darstellen:

$$\|A\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{n} |a_{ij}|.$$

Sie heißt Spaltenbetragssummennorm.

Beispiel: Die Frobenius–Norm ist eine Matrixnorm, gegeben durch:

$$\|A\|_F = \left(\sum_{i,j=1}^{n} |a_{ij}|^2 \right)^{1/2}.$$

Sie entspricht der Euklidischen Norm, wenn man die Matrix $A \in \mathbb{R}^{n\times n}$ als Vektor in $\mathbb{R}^{n^2}$ interpretiert. Offensichtlich gilt:

$$\|I\|_F = \sqrt{n},$$

sie kann also nicht eine einer Vektornorm zugeordnete Matrixnorm sein. Sie ist aber trotzdem eine submultiplikative und zur Euklidischen Norm passende Matrixnorm und wesentlich einfacher berechenbar als die Spektralnorm.

Mit Hilfe einer dieser Matrixnormen lässt sich der relative Fehler in der Matrix A durch die Größe $\varepsilon_A = \|\bar{A} - A\|/\|A\|$ messen.

Mit diesen Vorbereitungen lässt sich nun die Datenstabilität eines linearen Gleichungssystems leicht untersuchen. Zuerst wird der Spezialfall $\bar{A} = A$ betrachtet:

Satz 3.1. *Seien $A \in \mathbb{R}^{n\times n}$ eine reguläre Matrix, $b \in \mathbb{R}^n$, $b \neq 0$, $\Delta b \in \mathbb{R}^n$ und $\bar{b} := b + \Delta b$. $x \in \mathbb{R}^n$ erfülle das Gleichungssystem $Ax = b$, $\bar{x} = x + \Delta x$ erfülle das Gleichungssystem $A\bar{x} = \bar{b}$. Dann gilt für jede Vektornorm und jede dazu passende Matrixnorm:*

$$\varepsilon_x \leq \kappa(A)\, \varepsilon_b$$

mit $\varepsilon_x = \|\Delta x\|/\|x\|$, $\varepsilon_b = \|\Delta b\|/\|b\|$ und $\kappa(A) = \|A\|\,\|A^{-1}\|$.

Beweis. Durch Subtraktion von $x = A^{-1}b$ und $\bar{x} = x + \Delta x = A^{-1}\,\bar{b} = A^{-1}\,(b + \Delta b)$ erhält man $\Delta x = A^{-1}\,\Delta b$. Also

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

Mit $\|b\| = \|A\,x\| \leq \|A\|\,\|x\|$ folgt

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

woraus nach Division mit $\|x\|$ die Behauptung folgt. □

Die Zahl $\kappa(A)$ heißt die Konditionszahl der Matrix A und hängt von der verwendeten Norm ab. Sie ist für die Größe der Auswirkung von Datenfehlern in b auf die Lösung x verantwortlich.

Berücksichtigt man auch Störungen in A, so erhält man:

Satz 3.2. *Für jede Vektornorm und jede dazu passende und submultiplikative Matrixnorm gelten folgende Aussagen:*

1. *(Satz von der Inversen benachbarter Operatoren)*
   Falls $A \in \mathbb{R}^{n\times n}$ regulär ist und $\Delta A \in \mathbb{R}^{n\times n}$ die Abschätzung $\|\Delta A\| < 1/\|A^{-1}\|$ erfüllt, dann ist auch $\bar{A} = A + \Delta A$ regulär.

2. *Seien zusätzlich $b \in \mathbb{R}^n$, $b \neq 0$, $\Delta b \in \mathbb{R}^n$ und $\bar{b} = b + \Delta b$. $x \in \mathbb{R}^n$ erfülle das Gleichungssystem $Ax = b$, $\bar{x} = x + \Delta x$ erfülle das Gleichungssystem $\bar{A}\bar{x} = \bar{b}$. Dann gilt:*

$$\varepsilon_x \leq \frac{\kappa(A)}{1 - \kappa(A)\,\varepsilon_A}\,(\varepsilon_A + \varepsilon_b)$$

   mit $\varepsilon_A = \|\Delta A\|/\|A\|$.

Für kleine Fehler ε_A gilt also näherungsweise:

$$\varepsilon_x \leq \kappa(A)\,(\varepsilon_A + \varepsilon_b).$$

$\kappa(A)$ ist somit auch im allgemeinen Fall der Verstärkungsfaktor der Datenfehler.

Bemerkung: Für jede submultiplikative Matrixnorm gilt: $\|I\| = \|A \cdot A^{-1}\| \leq \|A\| \cdot \|A^{-1}\|$, also: $\kappa(A) \geq \|I\|$, wobei I die Einheitsmatrix bezeichnet. Für Matrizennormen, die einer Vektornorm zugeordnet sind, gilt $\|I\| = 1$ und somit $\kappa(A) \geq 1$.

Bemerkung:

1. Die Konditionszahl einer Matrix A bezüglich der Spektralnorm lässt sich mit Hilfe der so genannten Singulärwerte von A darstellen:

   Die Eigenwerte von $A^T A$ sind immer reell und größer oder gleich 0. Die nichtnegativen Quadratwurzel aus diesen Eigenwerten heißen die Singulärwerte von A und werden im Weiteren mit

$$0 \leq \sigma_1 \leq \sigma_{\leq} \ldots \leq \sigma_n$$

   bezeichnet. Offensichtlich gilt: $\|A\|_2 = \sigma_n$.

   Im Falle einer regulären Matrix A gilt: $\sigma_1 > 0$.

   Für die Konditionszahl $\kappa(A)$ benötigt man noch die Wurzeln der Eigenwerte von $(A^{-1})^T A^{-1} = (AA^T)^{-1}$. Die Eigenwerte von $A^T A$ und AA^T stimmen überein. Somit erhält man für die Singulärwerte von A^{-1}:

$$\frac{1}{\sigma_n} \leq \frac{1}{\sigma_{n-1}} \leq \ldots \leq \frac{1}{\sigma_1}.$$

Also: $\|A^{-1}\|_2 = 1/\sigma_1$. Zusammenfassend folgt somit:

$$\kappa_2(A) = \frac{\sigma_n}{\sigma_1}.$$

Die Konditionszahl einer Matrix ist also gleich dem Verhältnis des größten zum kleinsten Singulärwert der Matrix.

2. Im Spezialfall $AA^T = A^TA$ (A nennt man dann eine normale Matrix, symmetrische Matrizen sind Beispiele von normalen Matrizen) sind die Singulärwerte gleich dem Betrag der Eigenwerte der Matrix A: $\sigma_i = |\lambda_i|$ mit

$$|\lambda_1| \leq |\lambda_2| \leq \ldots \leq |\lambda_n|,$$

woraus für die Konditionszahl folgt:

$$\kappa_2(A) = \frac{|\lambda_n|}{|\lambda_1|}.$$

Die Konditionszahl einer normalen Matrix ist also gleich dem Betrag des Verhältnisses des betragsgrößten zum betragskleinsten Eigenwert der Matrix.

3. Ist A symmetrisch und positiv definit, so sind alle Eigenwerte positiv und es gilt

$$\kappa_2(A) = \frac{\lambda_n}{\lambda_1}.$$

Beispiel: Für die Matrix K_h, die bei der diskutierten Diskretisierung entsteht, lassen sich die Eigenwerte explizit berechnen:

$$\lambda_k = \frac{4}{h^2}\sin^2\left(\frac{k}{N+1}\frac{\pi}{2}\right) \quad \text{für } k = 1, 2, \ldots, N.$$

Für den kleinsten und größten Eigenwert folgt:

$$\lambda_1 = \frac{4}{h^2}\sin^2\left(\frac{\pi}{2(N+1)}\right) \approx \frac{4}{h^2}\frac{\pi^2}{4(N+1)^2} = \pi^2$$

und

$$\lambda_N = \frac{4}{h^2}\sin^2\left(\frac{N}{N+1}\frac{\pi}{2}\right) \approx \frac{4}{h^2}.$$

Somit erhält man für die Konditionszahl:

$$\kappa(K_h) = \frac{\lambda_N}{\lambda_1} \approx \frac{4}{\pi^2}\frac{1}{h^2} \gg 1.$$

Dieses Ergebnis ist typisch für viele Matrizen K_h, die durch Diskretisierung von Differentialgleichungsproblemen 2. Ordnung entstehen: Die Konditionszahl ist von der Größenordnung $1/h^2$:

$$\kappa(K_h) = O\left(\frac{1}{h^2}\right).$$

Mit jeder Norm misst man die Größe eines Vektors oder einer Matrix anders. Allerdings gilt z.B.

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\,\|x\|_\infty.$$

Ist also ein Vektor klein bezüglich der Euklidischen Norm, so ist er auch klein bezüglich der Maximumnorm, und umgekehrt.

Bemerkung: Zwei Normen $\|.\|_\alpha$ und $\|.\|_\beta$ mit der Eigenschaft, dass Konstanten $c > 0$ und $C > 0$ existieren, sodass

$$c\,\|x\|_\alpha \leq \|x\|_\beta \leq C\,\|x\|_\alpha,$$

für alle Vektoren x gilt, heißen äquivalent. Es gilt: Alle Normen in endlichdimensionalen Räumen wie $\mathbb{R}^n$ und $\mathbb{R}^{n\times n}$ sind äquivalent. So gesehen, misst man mit jeder Norm in endlichdimensionalen Räumen etwa das gleiche. Allerdings können sich die Konstanten c und C in Abhängigkeit der Raumdimension n ändern. Für große n misst man u.U. doch wieder erheblich anders. Welche Norm wann am besten ist, hängt vom Zweck ab.

3.3 Das Gaußsche Eliminationsverfahren

Das klassische Verfahren zur Lösung eines linearen Gleichungssystems

$$Ax = b$$

oder

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad \text{für} \quad i = 1, \ldots, n$$

oder ausführlicher

$$\begin{array}{ccccccccc}
a_{11}\,x_1 & + & a_{12}\,x_2 & + & \ldots & + & a_{1n}\,x_n & = & b_1 \\
a_{21}\,x_1 & + & a_{22}\,x_2 & + & \ldots & + & a_{2n}\,x_n & = & b_2 \\
\vdots & & \vdots & & & & \vdots & & \vdots \\
a_{n1}\,x_1 & + & a_{n2}\,x_2 & + & \ldots & + & a_{nn}\,x_n & = & b_n
\end{array}$$

ist das Gaußsche Eliminationsverfahren.

Im Folgenden werden gelegentlich auch die Bezeichnungen $A^{(0)} = (a_{ij}^{(0)})$ für A und $b^{(0)} = (b_i^{(0)})$ für b verwendet.

Im ersten Schritt des Gaußschen Eliminationsverfahrens wird die Variable x_1 aus der 2. bis zur n-ten Gleichung eliminiert, indem ein jeweils geeignetes Vielfaches der 1. Gleichung von den restlichen Gleichungen abgezogen wird.

Dadurch entsteht ein neues Gleichungssystem mit unveränderter Lösung:

$$A^{(1)}x = b^{(1)}$$

mit

$$A^{(1)} = \left(\begin{array}{c|ccc} u_{11} & u_{12} & \cdots & u_{1n} \\ \hline 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{array}\right), \qquad b^{(1)} = \left(\begin{array}{c} c_1 \\ \hline b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{array}\right),$$

wobei für $a_{11}^{(0)} \neq 0$

$$\begin{aligned} u_{1j} &= a_{1j}, \quad j = 1, 2, \ldots, n, \\ l_{i1} &= \frac{a_{i1}}{u_{11}}, \quad i = 2, \ldots, n, \\ a_{ij}^{(1)} &= a_{ij} - l_{i1}\, u_{1j}, \quad i, j = 2, \ldots, n, \end{aligned}$$

und

$$\begin{aligned} c_1 &= b_1^{(0)}, \\ b_i^{(1)} &= b_i^{(0)} - l_{i1}\, c_1, \quad i = 2, \ldots, n. \end{aligned}$$

Im zweiten Schritt des Gaußschen Eliminationsverfahrens wird die Variable x_2 aus der 3. bis zur n-ten Gleichung analog eliminiert: Es entsteht das neue Gleichungssystem

$$A^{(2)} x = b^{(2)}$$

mit

$$A^{(2)} = \left(\begin{array}{cc|ccc} u_{11} & \ldots & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & \cdots & u_{2n} \\ \hline \vdots & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{array}\right), \qquad b^{(2)} = \left(\begin{array}{c} c_1 \\ c_2 \\ \hline b_3^{(2)} \\ \vdots \\ b_n^{(2)} \end{array}\right),$$

wobei für $a_{22}^{(1)} \neq 0$

$$\begin{aligned} u_{2j} &= a_{2j}^{(1)}, \quad j = 2, 3, \ldots, n, \\ l_{i2} &= \frac{a_{i2}^{(1)}}{u_{22}}, \quad i = 3, \ldots, n, \\ a_{ij}^{(2)} &= a_{ij}^{(1)} - l_{i2}\, u_{2j}, \quad i, j = 3, \ldots, n \end{aligned}$$

und

$$\begin{aligned} c_2 &= b_2^{(1)}, \\ b_i^{(2)} &= b_i^{(1)} - l_{i2}\, c_2, \quad i = 3, \ldots, n. \end{aligned}$$

Nach insgesamt $n-1$ Schritten erhält man schließlich ein Gleichungssystem der Form

$$Ux = c \tag{3.1}$$

mit

$$U = \begin{pmatrix} u_{11} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{pmatrix}, \qquad c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}.$$

U heißt rechte obere Dreiecksmatrix. Man nennt das Gleichungssystem (3.1) ein gestaffeltes System. Es lässt sich leicht von unten nach oben auflösen:

$$\begin{aligned} x_n &= \frac{1}{u_{nn}} c_n, \\ x_i &= \frac{1}{u_{ii}} \left(c_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) \quad i = n-1, n-2, \ldots, 1. \end{aligned}$$

Das Gaußsche Eliminationsverfahren ist genau dann **durchführbar**, wenn

$$u_{kk} = a_{kk}^{(k-1)} \neq 0, \quad k = 1, 2, \ldots, n.$$

Diese Bedingung ist erfüllt, falls die ersten $n-1$ Hauptminoren $M_i \neq 0$. Die eineindeutige Lösbarkeit des linearen Gleichungssystems ist gegeben, falls noch $M_n = \det(A) \neq 0$.

Der **Aufwand** zur Berechnung der rechten oberen Dreiecksmatrix U beträgt ungefähr

$$(n-1)^2 + (n-2)^2 + \cdots + 2^2 + 1^2 \approx \frac{n^3}{3}$$

Operationen der Form $z = x - a \cdot y$.

Zur Berechnung von c benötigt man ungefähr

$$(n-1) + (n-2) + \cdots + 2 + 1 \approx \frac{n^2}{2}$$

Operationen.

Zur Berechnung von x durch Auflösung des gestaffelten Systems benötigt man ebenfalls ungefähr

$$(n-1) + (n-2) + \cdots + 2 + 1 \approx \frac{n^2}{2}$$

Operationen.

Die **Abspeicherung** des Zwischenergebnisses nach $k-1$ Schritten lässt sich in der Form

$$\left(\begin{array}{ccc|ccc} u_{11} & \cdots & \cdots & \cdots & \cdots & u_{1n} \\ l_{21} & \ddots & & & & \vdots \\ \vdots & \ddots & u_{k-1,k-1} & \cdots & \cdots & u_{k-1,n} \\ \hline \vdots & & l_{k,k-1} & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} \\ \vdots & & \vdots & \vdots & & \vdots \\ l_{n1} & \cdots & l_{n,k-1} & a_{nk}^{(k-1)} & \cdots & a_{nn}^{(k-1)} \end{array}\right), \qquad \left(\begin{array}{c} c_1 \\ \vdots \\ c_{k-1} \\ \hline b_k^{(k-1)} \\ \vdots \\ b_n^{(k-1)} \end{array}\right),$$

in ungefähr n^2 Speicherplätzen realisieren, falls A und b überschrieben werden dürfen.

3.4 Dreieckszerlegungen

Die einzelnen Schritte des Gaußschen Eliminationsverfahrens lassen sich auch folgendermaßen interpretieren:

Die Operationen, die im 1. Schritt auszuführen sind, sind äquivalent zu den Formeln

$$\begin{aligned} a_{1j} &= 1 \cdot u_{1j}, \quad j = 1, \ldots, n, \\ a_{i1} &= l_{i1} \cdot u_{11}, \quad i = 2, \ldots, n, \\ a_{ij} &= l_{i1} \cdot u_{1j} + a_{ij}^{(1)}, \quad i, j = 2, \ldots, n. \end{aligned}$$

Die Operationen, die im 2. Schritt auszuführen sind, sind äquivalent zu den Formeln

$$\begin{aligned} a_{2j}^{(1)} &= 1 \cdot u_{2j}, \quad j = 2, \ldots, n, \\ a_{i2}^{(1)} &= l_{i2} \cdot u_{22}, \quad i = 3, \ldots, n, \\ a_{ij}^{(1)} &= l_{i2} \cdot u_{2j} + a_{ij}^{(2)}, \quad i, j = 3, \ldots, n. \end{aligned}$$

Aus den Formeln des 1. Schrittes folgt dann:

$$\begin{aligned} a_{2j} &= l_{21} \cdot u_{11} + 1 \cdot u_{2j}, \quad j = 2, \ldots, n, \\ a_{i2} &= l_{i1} \cdot u_{1j} + l_{i2} \cdot u_{22}, \quad i = 3, \ldots, n, \\ a_{ij} &= l_{i1} \cdot u_{1j} + l_{i2} \cdot u_{2j} + a_{ij}^{(2)}, \quad i, j = 3, \ldots, n, \end{aligned}$$

u.s.w.

Man erkennt, dass

$$A = L\,U$$

gilt, wobei

$$L = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{pmatrix}, \qquad U = \begin{pmatrix} u_{11} & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{pmatrix}.$$

L heißt linke untere Dreiecksmatrix. Man spricht von einer Dreieckszerlegung von A.

Die Operationen, die im 1. Schritt für die rechte Seite auszuführen sind, sind äquivalent zu den Formeln

$$\begin{aligned} b_1 &= 1 \cdot c_1, \\ b_i &= l_{i1} \cdot c_1 + b_i^{(1)}, \quad i = 2, \ldots, n. \end{aligned}$$

Die Operationen, die im 2. Schritt für die rechte Seite auszuführen sind, sind äquivalent zu den Formeln

$$\begin{aligned} b_2^{(1)} &= 1 \cdot c_2, \\ b_i^{(1)} &= l_{i2} \cdot c_2 + b_i^{(2)}, \quad i = 3, \ldots, n. \end{aligned}$$

Aus den Formeln des 1. Schrittes folgt dann:

$$\begin{aligned} b_2 &= l_{21} \cdot c_1 + 1 \cdot c_2, \\ b_i &= l_{i1} \cdot c_1 + l_{i2} \cdot c_2 + b_i^{(2)}, \quad i = 3, \ldots, n, \end{aligned}$$

u.s.w.

Man erkennt, dass

$$Lc = b$$

gilt, wobei $c = (c_1, c_2, \ldots, c_n)^T$.

Schließlich ist im letzten Teil das System

$$Ux = c$$

zu lösen.

Zusammenfassung: In der ersten Phase wird eine Dreieckszerlegeung $A = LU$ durchgeführt. Daraus lässt sich dann leicht die Lösung des linearen Gleichungssystems $Ax = b$ bestimmen. Denn mit der Bezeichnung $c = Ux$ ist die Auflösung des Gleichungssystems

$$Ax = L\,Ux = b$$

gleichbedeutend mit der sukzessiven Auflösung der beiden Gleichungssysteme

$$Lc = b \quad \text{und} \quad Ux = c.$$

Da L und U Dreiecksmatrizen sind, nennt man diese Systeme gestaffelte Systeme. Das erste gestaffelte System löst man leicht von oben nach unten auf, das zweite gestaffelte System von unten nach oben.

Bemerkung: Nach den Überlegungen zum Aufwand des Gaußschen Eliminationsverfahrens benötigt man zur Dreieckszerlegung von A etwa $n^3/3$ Operationen, zur Auflösung der beiden gestaffelten Systeme je $n^2/2$ Operationen.

3.5 Rundungsfehleranalyse für das Gaußsche Eliminationsverfahren

Bei der **Rückwärtsanalyse** für das Gaußsche Eliminationsverfahren zur Lösung des linearen Gleichungssystems

$$Ax = b$$

wird versucht, die tatsächlich berechnete und durch Rundungsfehler verfälschte Näherung $\bar{x}$ als exaktes Ergebnis künstlich gestörter Daten $\bar{A}$, $\bar{b}$ darzustellen.

Beispiel: Die Lösung einer einzelnen linearen Gleichung

$$a \cdot x = b$$

erfolgt durch den Algorithmus

$$\bar{x} = b /^* a.$$

Auf Grund der Genauigkeit der Gleitkommadivision weiß man, dass

$$\bar{x} = b /^* a = (b/a)(1+\varepsilon) \quad \text{mit } |\varepsilon| \leq \text{eps}.$$

Also

$$\bar{a} \cdot \bar{x} = b \quad \text{mit } \bar{a} = \frac{1}{1+\varepsilon} a.$$

Die Größe der Datenstörung in a lässt sich folgendermaßen abschätzen:

$$|\Delta a| = |\bar{a} - a| = |\varepsilon|\,|\bar{a}| \leq \text{eps}\,|\bar{a}| = \text{eps}\,|\bar{l}|\,|\bar{u}|$$

mit $\bar{l} = 1$ und $\bar{u} = \bar{a}$, der trivialen Dreieckszerlegung von a. Die tatsächlich berechnete Näherung lässt sich also als exaktes Ergebnis gestörter Daten interpretieren, durch die obige Abschätzung kennt man die Größenordnung der notwendigen Datenstörung in a.

Der folgende Satz enthält das Ergebnis der **Rückwärtsanalyse** des Rundungsfehlereinflusses für allgemeine lineare Gleichungssysteme. Wie im vorigen Beispiel wird angenommen, dass die Daten A und b selbst keine Rundungsfehler aufweisen:

Satz 3.3 (Sautter)**.** *Die mit Hilfe des Gaußschen Eliminationsverfahrens berechnete Näherung $\bar{x}$ des linearen Gleichungssystems $Ax = b$ ist exakte Lösung eines Gleichungssystems*

$$\bar{A}\bar{x} = b \quad \textit{mit} \quad |\Delta A| = |\bar{A} - A| \leq \frac{2(n+1) \cdot \textit{eps}}{1 - n \cdot \textit{eps}} |\bar{L}|\,|\bar{U}| \ \textit{ bis auf } O(\textit{eps}^2),$$

falls $n \cdot \textit{eps} < 1/2$. $\bar{L}$ und $\bar{U}$ bezeichnen die tatsächlich berechneten Dreiecksmatrizen.

Bezeichnung: $|M|$ bezeichnet jene Matrix, die aus M entsteht, indem man komponentenweise den Betrag bildet. $\|M\|$, die Norm einer Matrix, ist eine Zahl.

In der Terminologie des 2. Kapitels bedeutet das, dass der restliche Rundungsfehler beim Gaußschen Eliminationsverfahren einem künstlichen Datenfehler $\varepsilon_A^{(r)} = \|\Delta A\|/\|A\|$ entspricht. Somit erhält man (in erster Ordnung) folgende Abschätzung für den restlichen Rundungsfehler $\varepsilon_x^{(r)}$:

$$\varepsilon_x^{(r)} \leq \kappa(A)\varepsilon_A^{(r)}$$

mit

$$\varepsilon_A^{(r)} = \frac{\|\Delta A\|}{\|A\|} \leq \frac{2n \cdot \text{eps}}{1 - n \cdot \text{eps}} \frac{\||\bar{L}| \cdot |\bar{U}|\|}{\|A\|}.$$

Zur Beurteilung der numerischen Stabilität des Gaußschen Eliminationsverfahrens ist dieser restliche Rundungsfehler mit dem unvermeidbaren Rundungsfehler $\varepsilon_x^{(0)}$, verursacht durch die Rundungsfehler $\varepsilon_A^{(0)}$ und $\varepsilon_b^{(0)}$ bei der Dateneingabe, zu vergleichen. Es gilt (in erster Ordnung)

$$\varepsilon_x^{(0)} \leq \kappa(A)(\varepsilon_A^{(0)} + \varepsilon_b^{(0)}) \leq 2\,\text{eps}\,\kappa(A).$$

Durch Vergleich der Abschätzungen des unvermeidbaren Rundungsfehlers und des restlichen Rundungsfehlers erkennt man, dass das Gaußsche Eliminationsverfahren (für nicht zu große n) dann numerisch stabil ist, wenn die Komponenten von $\bar{L}$ und $\bar{U}$ im Vergleich zu den Komponenten von A nicht zu groß sind.

Durch geeignete Zeilen- und/oder Spaltenvertauschungen lässt sich stets garantieren, dass die Komponenten von $\bar{L}$ kleiner oder gleich 1 bleiben:

Im k-ten Schritt des Gaußschen Eliminationsverfahren wird die k-te Zeile der Matrix

$$A^{(k-1)} = \left(\begin{array}{ccc|ccc} u_{11} & \cdots & \cdots & \cdots & \cdots & u_{1n} \\ 0 & \ddots & & & & \vdots \\ \vdots & \ddots & u_{k-1,k-1} & \cdots & \cdots & u_{k-1,n} \\ \hline \vdots & & 0 & a_{kk}^{(k-1)} & \cdots & a_{kn}^{(k-1)} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & a_{nk}^{(k-1)} & \cdots & a_{nn}^{(k-1)} \end{array}\right)$$

zur Elimination der Variable x_k aus den restlichen Gleichungen verwendet. Das Element $a_{kk}^{(k-1)}$ heißt Pivotelement. Aus dem Pivotelement und den restlichen Komponenten der k-ten Spalte werden die Komponenten von L gebildet. Je kleiner das Pivotelement ist, desto größer können die Komponenten von L werden.

Es kann allerdings jedes beliebige Element $a_{ij}^{(k-1)} \neq 0$ mit $i, j = k, \ldots, n$ als Pivotelement verwendet werden, wenn man vor dem nächsten Eliminationsschritt zuerst die i-te mit der k-ten Zeile und die j-te mit der k-ten Spalte vertauscht. Die Vertauschung zweier Zeilen ändert nicht die Lösung eines Gleichungssystem. Die Vertauschung zweier Spalten bedeutet nur eine Umbenennung der Unbekannten.

Man unterscheidet vor allem zwei Strategien zur Wahl des Pivotelements:

1. Totalpivotsuche: Es wird das betragsgrößte Element unter allen möglichen Elementen $a_{ij}^{(k-1)}$, $i,j = k,\ldots,n$ der Restmatrix als nächstes Pivotelement verwendet. Bezeichnet man mit $\tilde{a}_{ij}^{(k-1)}$ die Elemente der Restmatrix nach der Zeilen- und Spaltenvertauschung, so bedeutet offensichtlich diese Wahl des Pivotelements:

$$|\tilde{a}_{kk}^{(k-1)}| = \max_{i,j=k,\ldots,n} |a_{ij}^{(k-1)}|.$$

$$\Longrightarrow \quad PA\,\Pi = LR$$

2. Spaltenpivotsuche: Es wird das betragsgrößte Element unter allen möglichen Elementen $a_{ik}^{(k-1)}$, $i = k,\ldots,n$ der k-ten Spalte als nächstes Pivotelement verwendet. Also

$$|\tilde{a}_{kk}^{(k-1)}| = \max_{i=k,\ldots,n} |a_{ik}^{(k-1)}|.$$

$$\Longrightarrow \quad PA = LR$$

In beiden Fällen gilt dann

$$|l_{ik}| = \frac{|\tilde{a}_{ik}^{(k-1)}|}{|\tilde{a}_{kk}^{(k-1)}|} \leq 1.$$

Man hat also das Wachstum der Elemente von L durch Total- oder Spaltenpivotsuche unter Kontrolle.

Nicht so einfach ist die Auswirkung der Pivotsuche auf das Wachstum der Elemente von U. Für den k-ten Eliminationsschritt gilt wegen $|l_{ik}| \leq 1$ jedenfalls die Abschätzung

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

also

$$\max|\tilde{a}_{ij}^{(k)}| \leq 2\,\max|\tilde{a}_{ij}^{(k-1)}|.$$

Es kommt also maximal zu einer Verdoppelung der Einträge pro Eliminationsschritt. Somit gilt zumindest folgende Abschätzung:

$$\max|u_{ij}| \leq 2^{n-1}\,\max|a_{ij}|. \tag{3.2}$$

Bei Verwendung der Totalpivotsuche vermutete man, dass

$$\max|u_{ij}| \leq n\,\max|a_{ij}|,$$

Diese Vermutung wurde 1991 widerlegt, siehe [Gould91, Edelman91]. Seither vermutet man, dass

$$\max|u_{ij}| \leq (\alpha n + \beta)\,\max|a_{ij}|,$$

dass also die Elemente von U nur moderat anwachsen. Diese Behauptung konnte bisher nicht bewiesen oder widerlegt werden. Alle praktischen Rechnungen bestätigten jedoch diese Vermutung. Wilkinson konnte jedoch folgende schwächere Abschätzung beweisen

$$\max|\tilde{a}_{ij}^{(k)}| \leq f(k)\,\max|a_{ij}^{(0)}| \tag{3.3}$$

mit

$$f(k) := \sqrt{k}\sqrt{2 \cdot 3^{1/2} \cdot 4^{1/3} \cdot \ldots \cdot k^{1/(k-1)}} \tag{3.4}$$

Daher gilt das Gaußsche Eliminationsverfahren mit Totalpivotsuche als numerisch stabil.

Bei Verwendung der Spaltenpivotsuche kann man durch konkrete Beispiele zeigen, dass die pessimistische Abschätzung (3.2) im allgemeinen Fall nicht verbessert werden kann, diese Abschätzung ist scharf. Das Gaußsche Eliminationsverfahren mit Spaltenpivotsuche ist also nicht immer numerisch stabil. Für spezielle Matrizenklassen erhält man allerdings wesentlich bessere Abschätzungen, und zwar ergeben sich fur Abschätzung (3.3):
- für rechte obere Hessenbergmatrizen: $f(k) = k + 1$
- für Tridiagonalmatrizen: $f(k) = 2$

Für reguläre Matrizen ist das Gaußsche Eliminationsverfahren für beide Varianten der Pivotsuche stets **durchführbar**, d.h., man erhält stets ein von Null verschiedenes Pivotelement. Die Durchführung des Verfahrens ohne Pivotsuche ist für reguläre Matrizen nicht in allen Fällen gesichert.

In der Praxis begnügt man sich aus Aufwandsgründen meistens mit Spaltenpivotsuche. Es ist allerdings empfehlenswert, vor Anwendung der Spaltenpivotsuche das Problem zu skalieren.

3.6 Orthogonalisierungsverfahren

Die einzelnen Schritte des Gaußschen Eliminationsverfahrens lassen sich auch durch Multiplikation mit entsprechenden Matrizen darstellen. So erhält man z.B. für den 1. Schritt:

$$A^{(1)} = G_1 A^{(0)}, \quad b^{(1)} = G_1 b^{(0)}$$

mit

$$G_1 = \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -l_{21} & 1 & \ddots & & \vdots \\ -l_{31} & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -l_{n1} & 0 & \cdots & 0 & 1 \end{pmatrix} = I - \begin{pmatrix} 0 \\ l_{21} \\ l_{31} \\ \vdots \\ l_{n1} \end{pmatrix} \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \end{pmatrix}.$$

Wie beim Gaußschen Eliminationsverfahren versucht man beim so genannten Householder–Verfahren im ersten Schritt eine Matrix $A^{(1)}$ zu erzeugen, die folgende Struktur hat:

$$A^{(1)} = \left(\begin{array}{c|ccc} * & * & \cdots & * \\ \hline 0 & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{array}\right)$$

Allerdings wählt man eine andere Transformationsmatrix:

$$A^{(1)} = P_1 A^{(0)}, \qquad b^{(1)} = P_1 b^{(0)}.$$

Die Matrix P_1 soll eine orthogonale Matrix sein, d.h.: $P_1^T P = I$. Das sichert, dass sich die Konditionszahl der Matrizen bezüglich der Spektralnorm nicht ändert:

$$\kappa_2(A^{(1)}) = \kappa_2(A^{(0)}).$$

Damit wirken sich die im 1. Schritt neu erzeugten Rundungsfehler genauso auf das Endergebnis aus wie die Rundungsfehler bei der Dateneingabe. Der Algorithmus ist dann sicherlich numerisch stabil (für nicht allzu große Dimensionen n).

Wenn a_j, $j = 1, 2, \ldots, n$ die Spalten der Matrix $A^{(0)}$ bezeichnen, so gilt:

$$A^{(0)} = \left(a_1 \mid a_2 \mid \cdots \mid a_n\right)$$

und somit:

$$P_1 A^{(0)} = \left(P_1 a_1 \mid P_1 a_2 \mid \cdots \mid P_1 a_n\right).$$

Gesucht ist also eine orthogonale Matrix P_1 mit der Eigenschaft

$$P_1 a_1 = k e_1,$$

wobei e_1 den ersten Einheitsvektor bezeichnet:

$$e_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

Geometrisch lässt sich eine solche Transformation in 2 und 3 Dimensionen leicht angeben: Der erste Spaltenvektor kann z.B. durch eine Spiegelung an einer Geraden bzw. Ebene auf die x-Achse, also auf ein Vielfaches des ersten Einheitsvektors, transformiert werden.

Eine solche Transformation wird für beliebige Raumdimensionen durch die Matrix

$$P_1 = I - \frac{2}{v^T v} v v^T \quad (= P_1^T) \tag{3.5}$$

beschrieben.

Der Vektor v lässt sich in 2 bzw. 3 Dimensionen als Normalvektor zur Spiegelungsgeraden bzw. Spiegelungsebene interpretieren.

Aus dieser Interpretation erhält man leicht für den allgemeinen Fall

$$v = a_1 + \operatorname{sign}(a_{11}) \|a_1\|_2 \, e_1.$$

Nach dem 1. Schritt erhält man das Gleichungssystem

$$A^{(1)} x = b^{(1)}$$

mit

$$A^{(1)} = \left(\begin{array}{c|ccc} * & * & \cdots & * \\ \hline 0 & & & \\ \vdots & & \tilde{A}^{(1)} & \\ 0 & & & \end{array}\right) \qquad \text{und} \qquad b^{(1)} = \left(\begin{array}{c} * \\ \hline \\ \tilde{b}^{(1)} \\ \\ \end{array}\right)$$

Im 2. Schritt des Householder-Verfahrens wird eine analoge Transformation wie im 1. Schritt für $\tilde{A}^{(1)}$ und $\tilde{b}^{(1)}$ durchgeführt, jedoch für ein um 1 in der Dimension reduziertes Problem, u.s.w.

Nach insgesamt $n-1$ Schritten erhält man schließlich ein gestaffeltes System

$$Rx = c.$$

Jeder einzelne Schritt lässt sich als Multiplikation mit einer orthogonalen Matrix P_i in faktorisierter Darstellung analog (3.5) darstellen. Zusammenfassend erhält man also:

$$P_{n-1} \cdots P_2 P_1 A = R \qquad \text{und} \qquad P_{n-1} \cdots P_2 P_1 b = c.$$

Daraus folgt die Darstellung

$$A = P_1^T P_2^T \cdots P_{n-1}^T R = QR$$

mit

$$Q = P_1^T P_2^T \cdots P_{n-1}^T = P_1 P_2 \cdots P_{n-1}. \tag{3.6}$$

Q ist ebenfalls eine orthogonale Matrix. Das Householder–Verfahren entspricht also einer Zerlegung der Matrix A in das Produkt einer orthogonalen Matrix und einer rechten oberen Dreiecksmatrix. Man spricht kurz von einer QR-Zerlegung.

Aus der QR-Zerlegung $A = Q \cdot R$ lässt sich dann leicht die Lösung des linearen Gleichungssystem $Ax = b$ bestimmen. Denn mit der Bezeichnung $c = Rx$ ist die Auflösung des Gleichungssystems

$$Ax = QRx = b$$

gleichbedeutend mit der sukzessiven Auflösung der beiden Gleichungssysteme

$$Qc = b \quad \text{und} \quad Rx = c.$$

Das erste System besitzt die Lösung $c = Q^T b$, das zweite System ist ein gestaffeltes System, das leicht von unten nach oben aufgelöst werden kann.

Der Aufwand zur Berechnung von Q in faktorisierter Darstellung (3.6) und von R kostet ungefähr $2n^3/3$ Operationen.

Der Aufwand zur Berechnung von c kostet rund n^2 Operationen.

Die Lösung des gestaffelten Systems $Rx = c$ kostet rund $n^2/2$ Operationen.

Bemerkung: Für die Lösung eines linearen Gleichungssystems mit dem Householder-Verfahren wird die Matrix Q nicht explizit benötigt. Man benötigt für jeden einzelnen Schritt jeweils nur einen Vektor v_i.

Will man stattdessen die Matrix Q explizit berechnen, so kostet das weitere $2n^3/3$ Operationen.

Die Durchführung des Verfahrens lässt sich so organisieren, dass man etwa n^2 Speicherplätze benötigt. Dabei wird A im wesentlichen durch R und die v_i überspeichert.

Zusammenfassend gilt also für das Householder-Verfahren: Es kostet etwa doppelt so viele Operationen wie das Gaußsche Eliminationsverfahren. Da sich im Laufe des Verfahrens die Konditionszahlen nicht verändern, ist es allerdings immer numerisch stabil.

3.7 Spezielle Gleichungssysteme

In Anwendungsproblemen treten häufig lineare Gleichungssysteme mit Matrizen spezieller Struktur auf, die eine effizientere Lösung erlauben.

Eine häufig auftretende Eigenschaft von Matrizen, die durch Diskretisierung von Differentialgleichungsproblemen entstehen, ist ihre Dünnbesetztheit, d.h., viele Einträge einer solchen Matrix sind 0. Durch geeignete Nummerierung kann man oft erreichen, dass vor allem Diagonalen, die von der Hauptdiagonale genügend weit entfernt sind, nur Nulleinträge besitzen. Solche Matrizen nennt man **Bandmatrizen**.

Ein weiterer wichtiger Fall sind lineare Gleichungssysteme mit **symmetrischen positiv definiten Matrizen**. Solche Systeme erhält man vor allem bei der Diskretisierung elliptischer Differentialgleichungsprobleme, die gewisse Symmetrieeigenschaften aufweisen. Lassen sich z.B. die Gleichungen aus einem Variationsprinzip (wie z.B. das Prinzip der virtuellen Arbeit) ableiten und werden sie geeignet diskretisiert, so erhält man Gleichungssysteme mit symmetrischen positiv definiten Matrizen.

Im Folgenden werden spezielle direkte Verfahren, das sind Verfahren, die abgesehen von Rundungsfehlern in endlich vielen Schritten die exakte Lösung eines Gleichungssystems liefern, für diese beiden Klassen von Matrizen diskutiert.

3.7.1 Dreieckszerlegungen für Bandmatrizen

Eine Bandmatrix ist eine Matrix, bei der bis auf ein Band von Diagonalen rund um die Hauptdiagonale alle Matrixelement gleich 0 sind. Genauer gesagt, unterscheidet man zwischen oberer und unterer Bandbreite:

Definition 3.1. *Eine Matrix $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ besitzt eine untere Bandbreite p bzw. eine obere Bandbreite q, falls $a_{ij} = 0$ für alle i, j mit $i > j + p$ bzw. mit $j > i + q$.*

Eine Bandmatrix hat also folgende Gestalt:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1,q+1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ a_{p+1,1} & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & a_{n-q,n} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n,n-p} & \cdots & a_{nn} \end{pmatrix}.$$

Beim Gaußschen Eliminationsverfahren ohne Pivotsuche überträgt sich diese Gestalt auf die Dreiecksmatrizen. L besitzt die gleiche untere Bandbreite wie A, U besitzt die gleiche obere Bandbreite wie A:

$$L = \begin{pmatrix} l_{11} & 0 & \cdots & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & & \vdots \\ l_{p+1,1} & \ddots & \ddots & \ddots & & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & l_{n,n-p} & \cdots & l_{nn} \end{pmatrix}, \quad U = \begin{pmatrix} u_{11} & \cdots & u_{1,q+1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & u_{n-q,n} \\ \vdots & & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 0 & u_{nn} \end{pmatrix}.$$

Dadurch lässt sich Aufwand sparen. Zur Durchführung des Gaußschen Eliminationsverfahrens für Matrizen mit unterer Bandbreite p und oberer Bandbreite q benötigt man ungefähr $p\,q\,n$ Operationen, falls $1 \ll p, q \ll n$.

Der Speicherbedarf zur Durchführung des Gaußschen Eliminationsverfahrens beträgt $(p + q + 1)n$ Speicherplätze.

Bemerkung: Elemente außerhalb des Bandes von insgesamt $p + q + 1$ Diagonalen rund um die Hauptdiagonale sind gleich 0 und bleiben auch während des Gaußschen Eliminationsverfahrens gleich 0. Dies gilt nicht für eventuell vorhandene Nulleinträge der Matrix A innerhalb des Bandes. Solche Nulleinträge bleiben im Allgemeinen während des Verfahrens nicht erhalten (fill in).

Matrizen mit $p = q = 1$ nennt man Tridiagonalmatrizen. Die Durchführung des Verfahrens erfordert nur $8n - 7$ Gleitkommaoperationen, der Rechenaufwand ist also proportional zu n, der Anzahl der Unbekannten. Zur Abspeicherung der Daten benötigt man $4n - 2$ Speicherplätze, die zur Durchführung des gesamten Algorithmus ausreichen, wenn die ursprünglichen Daten überschrieben werden dürfen. Der Algorithmus ist asymptotisch optimal.

3.7.2 Die Cholesky-Zerlegung für symmetrische positiv definite Matrizen

Für symmetrische positiv definite Matrizen A, also für Matrizen mit

$$A^T = A$$

und

$$x^T Ax > 0 \quad \text{für alle } x \in \mathbb{R}^n \text{ mit } x \neq 0,$$

lässt sich eine Dreieckszerlegung $A = LU$ mit $U = L^T$ durchführen:

$$A = LL^T.$$

Man spricht von einer Cholesky-Zerlegung. Im Gegensatz zum Gaußschen Eliminationsverfahren sind die Diagonalelemente von L nicht gleich 1, sondern ergeben sich im Laufe der Rechnung.

Der Aufwand des Cholesky-Verfahrens entspricht dem Aufwand des Gaußschen Eliminationsverfahrens, allerdings müssen alle Operationen aufgrund der Symmetrie nur für Indizes i, j mit $i \geq j$ durchgeführt werden. Man benötigt damit nur etwa die Hälfte der Operationen für das Gaußsche Eliminationsverfahren, also etwa $n^3/6$ Operationen.

3.8 Ergänzungen zu direkten Verfahren

Mit den bisher diskutierten Verfahren lassen sich auch weitere Problemstellungen lösen:

3.8.1 Gleichungssysteme mit mehreren rechten Seiten

Zur Lösung mehrerer Gleichungssysteme der Form

$$Ax_l = b_l, \qquad l = 1, \ldots, r$$

mit gleicher Matrix aber verschiedenen rechten Seiten, benötigt man nur einmal eine Dreieckszerlegung. Nur die gestaffelten Systeme müssen mehrmals gelöst werden. Damit ergibt sich ein Aufwand von $n^3/3 + rn^2$ Operationen.

Beispiel: Berechnung der Inversen einer regulären Matrix. Bezeichnet man die unbekannten Spaltenvektoren der Inversen A^{-1} mit x_l, so entspricht die Berechnung der Inversen der Lösung der Systeme

$$Ax_l = e_l, \qquad l = 1, \ldots, n.$$

Dabei bezeichnet e_l den l-ten natürlichen Einheitsvektor, dessen l-te Komponente gleich 1 ist, während alle anderen gleich 0 sind. Nach den obigen Aufwandsüberlegungen kostet diese Berechnung der Inversen insgesamt $n^3/3 + nn^2 = 4n^3/3$ Operationen. Eine etwas genauere Zählung ergibt tatsächlich nur n^3 Operationen.

Man könnte die Lösung eines linearen Gleichungssystems $Ax = b$ auch nach der Formel $x = A^{-1}b$ über die Berechnung der Inversen durchführen. Die obige Analyse zeigt allerdings, dass dazu wesentlich mehr Operationen benötigt werden als beim Gaußschen Eliminationsverfahren.

3.8.2 Berechnung der Determinante einer Matrix

Bei Vorliegen einer Dreieckszerlegung $A = LU$ lässt sich die Determinante von A leicht bestimmen:

$$\det A = \det L \cdot \det U = l_{11} \cdot l_{22} \cdots l_{nn} \cdot u_{11} \cdot u_{22} \cdots u_{nn}.$$

Dabei wurde verwendet, dass die Determinante einer Dreiecksmatrix gleich dem Produkt der Diagonalelemente ist.

3.8.3 Ausgleichsprobleme

Sei A eine $m \times n$-Matrix mit $m > n$, $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$. Das überbestimmte Gleichungssystem

$$Ax = b$$

ist in dieser Form nicht korrekt gestellt. Es gibt rechte Seiten, für die keine Lösung existiert. Stattdessen macht es oft Sinn (Beispiel: Ausgleichsgerade) nach einem Vektor zu suchen, der das Problem möglichst gut löst, z.B. in folgendem Sinn: Gesucht ist $x \in \mathbb{R}^n$ mit

$$\|Ax - b\|_2 \to \min .$$

Man spricht von einem Ausgleichsproblem. Falls $\operatorname{Rang} A = n$, ist dieses Problem korrekt gestellt. Man sieht dann leicht, dass die Lösung eindeutig ist und das folgende Gleichungssystem (Normalgleichungen) erfüllt:

$$A^T Ax = A^T b.$$

Das Ausgleichsproblem lässt sich leicht mithilfe einer QR-Zerlegung lösen: Sei

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

mit einer orthogonalen $m \times m$-Matrix Q und einer rechten oberen $n \times n$-Dreiecksmatrix R. Dann folgt aus der Normalgleichung sofort

$$Rx = b,$$

also wird das Ausgleichsproblem auf die Lösung eines gestaffelten Systems reduziert.

Bemerkung: Für den allgemeinen Fall ist das Ausgleichsproblem noch nicht korrekt gestellt. Es wird zu einem korrekt gestellten Problem durch die zusätzliche Forderung an eine Lösung der Normalgleichungen:

$$\|x\|_2 \to \min,$$

d.h., man wählt jene Lösung der Normalgleichungen aus, die die kleinste Euklidische Norm besitzt.

Kapitel 4

Iterative Verfahren zur Lösung linearer Gleichungssysteme

Als Motivation für typische Gleichungssystems, die mit iterativen Verfahren gelöst werden, wird im Folgenden die Finite-Differenzen-Methode für ein zweidimensionales Wärmeleitproblem näher diskutiert.

4.1 Ein Beispiel

Sei $\Omega = (0,1) \times (0,1)$ (das Einheitsquadrat in der Ebene), Γ bezeichnet den Rand der Menge Ω. Für vorgegebene Funktionen $f(x,y)$ in Ω und $g(x,y)$ auf Γ ist eine Funktion $u(x,y)$ auf $\Omega \cup \Gamma$ gesucht, die folgende Bedingungen erfüllt:

$$\begin{aligned} -u_{xx} - u_{yy} &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{auf } \Gamma. \end{aligned}$$

Man spricht von einem Randwertproblem. Diese Gleichungen beschreiben eine Vielzahl von physikalisch-technischen Problemstellungen, z.B. die Temperaturverteilung bei vorgegebenen Wärmequellen.

Diskretisiert man das obige Randwertproblem auf dem Gitter $\Omega_h = \{(x_i, y_j) | i, j = 1, 2, \ldots, N\}$ mit $h = 1/(N+1)$, $x_i = i \cdot h$, $y_j = j \cdot h$ unter Verwendung von zentralen Differenzenquotienten für die zweiten Ableitungen

$$\begin{aligned} u_{xx} &\approx \frac{1}{h^2} \left(u_{i-1,j} - 2u_{ij} + u_{i+1,j} \right), \\ u_{yy} &\approx \frac{1}{h^2} \left(u_{i,j-1} - 2u_{ij} + u_{i,j+1} \right), \end{aligned}$$

so entsteht in jedem Gitterpunkt (x_i, y_j) eine Differenzengleichung

$$\frac{1}{h^2}(-u_{i-1,j} - u_{i,j-1} + 4u_{ij} - u_{i+1,j} - u_{i,j+1}) = f_{ij},$$

wobei $f_{ij} = f(x_i, y_j)$. u_{ij} bezeichnet dabei die Näherung für $u(x_i, y_j)$.

Ordnet man die Unbekannten u_{ij} und die Differenzengleichungen zeilenweise von links nach rechts und die Zeilen von unten nach oben, so entsteht unter Verwendung der Randbedingungen

$$u_{0j} = 0, \quad u_{N+1,j} = 0, \quad u_{i0} = 0, \quad u_{i,N+1} = 0$$

das lineare Gleichungssystem

$$K_h \underline{u}_h = \underline{f}_h$$

mit

$$K_h = \frac{1}{h^2} \left(\begin{array}{cccc|cccc|cccc|cccc} 4 & -1 & & & -1 & & & & & & & & & & & \\ -1 & \ddots & \ddots & & & \ddots & & & & & & & & & & \\ & \ddots & \ddots & -1 & & & \ddots & & & & & & & & & \\ & & -1 & 4 & & & & -1 & & & & & & & & \\ \hline -1 & & & & 4 & -1 & & & \ddots & & & & & & & \\ & \ddots & & & -1 & \ddots & \ddots & & & \ddots & & & & & & \\ & & \ddots & & & \ddots & \ddots & -1 & & & \ddots & & & & & \\ & & & -1 & & & -1 & 4 & & & & \ddots & & & & \\ \hline & & & & \ddots & & & & \ddots & \ddots & & & -1 & & & \\ & & & & & \ddots & & & \ddots & \ddots & \ddots & & & \ddots & & \\ & & & & & & \ddots & & & \ddots & \ddots & \ddots & & & \ddots & \\ & & & & & & & \ddots & & & \ddots & \ddots & & & & -1 \\ \hline & & & & & & & & -1 & & & & 4 & -1 & & \\ & & & & & & & & & \ddots & & & -1 & \ddots & \ddots & \\ & & & & & & & & & & \ddots & & & \ddots & \ddots & -1 \\ & & & & & & & & & & & -1 & & & -1 & 4 \end{array} \right)$$

und

$$\underline{u}_h = \left(\begin{array}{c} u_{11} \\ u_{21} \\ \vdots \\ u_{N1} \\ \hline u_{12} \\ u_{22} \\ \vdots \\ u_{N2} \\ \hline \vdots \\ \vdots \\ \vdots \\ \hline u_{1N} \\ u_{2N} \\ \vdots \\ u_{NN} \end{array}\right), \qquad \underline{f}_h = \left(\begin{array}{c} f_{11} \\ f_{21} \\ \vdots \\ f_{N1} \\ \hline f_{12} \\ f_{22} \\ \vdots \\ f_{N2} \\ \hline \vdots \\ \vdots \\ \vdots \\ \hline f_{1N} \\ f_{2N} \\ \vdots \\ f_{NN} \end{array}\right).$$

Dabei sind die Vektoren $\underline{u}_h$ und $\underline{f}_h$ in N Blöcke von jeweils N Komponenten unterteilt. Die Matrix K_h besteht aus N^2 Teilmatrizen, die jeweils $N \times N$-Matrizen sind.

Das entstehende Gleichungssystem ist für große N sehr groß ($n = N^2$ Unbekannte und Gleichungen) und dünnbesetzt (maximal 5 Nichtnulleinträge pro Zeile). Als Bandmatrix besitzt es die untere und obere Bandbreite $p = q = N = \sqrt{n}$.

4.1.1 Typische Eigenschaften von Diskretisierungsmatrizen

Der Abstand h ist ein Maß für die Feinheit der Zerlegung. Offensichtlich gilt, dass die Anzahl n_h der Unbekannten proportional $1/h^d$ ist, wobei d die Raumdimension des Problems ist, kurz $n_h = O(1/h^d)$.

Bei der Diskretisierung entstehen Matrizen mit folgenden typische Eigenschaften:

1. Die Dimension n_h der Matrix K_h ist sehr groß, da man vor allem an feinen Zerlegungen interessiert ist. Je feiner die Zerlegung ist, desto kleiner ist im Allgemeinen der Diskretisierungsfehler. Die Dimension n_h der Matrix K_h ist also (sehr) groß:

$$n_h = O(\frac{1}{h^d}).$$

2. Die meisten Einträge der Matrix sind 0, die Matrix ist dünn besetzt. Die Gesamtanzahl der Nichtnulleinträge ist typisch von der Größenordnung $O(n_h) = O(1/h^d)$.

3. Bei geeigneter Nummerierung der Knoten entsteht eine Bandmatrix mit Bandbreite

$$p_h = q_h = O(\frac{1}{h^{d-1}}) = O(n_h^{\frac{d-1}{d}}).$$

4. Zumindest bei der konventionellen Diskretisierung von Randwertproblemen 2. Ordnung gilt für die Konditionszahl unabhängig von der Raumdimension:

$$\kappa(K_h) = O(\frac{1}{h^2}).$$

5. Im diskutierten Beispiel (aber nicht im Allgemeinen) ist die Matrix K_h symmetrisch: $K_h^T = K_h$.

6. Im diskutierten Beispiel (aber nicht im Allgemeinen) ist die Matrix K_h positiv definit: $\underline{v}_h^T K_h \underline{v}_h > 0$ für alle $\underline{v}_h \neq 0$.

Daraus ergeben sich bei Verwendung des Gaußschen Eliminationsverfahrens (für Bandmatrizen) folgende Aufwandsbetrachtungen:

Speicherbedarf:

$$n_h(p_h + q_h + 1) = O(\frac{1}{h^d}\frac{1}{h^{d-1}}) = O(\frac{1}{h^{2d-1}}) = O(n_h^{\frac{2d-1}{d}}).$$

Anzahl der Operationen:

$$p_h q_h n_h = O(\frac{1}{h^{d-1}}\frac{1}{h^{d-1}}\frac{1}{h^d}) = O(\frac{1}{h^{3d-2}}) = O(n_h^{\frac{3d-2}{d}}).$$

Also:

	$d=1$	$d=2$	$d=3$
Speicherbedarf	n_h	$n_h^{3/2}$	$n_h^{5/3}$
Anzahl der Operationen	n_h	n_h^2	$n_h^{7/3}$

Man sieht, dass der Aufwand an Speicher und Rechenzeit nur bei eindimensionalen Problemen proportional zur Anzahl der Unbekannten steigt. In diesem Fall nennt man das Verfahren (quasi-)optimal. Für zwei- oder dreidimensionale Probleme ist das Gaußsche Eliminationsverfahren nicht mehr optimal. Im Folgenden werden als Alternative zu einem direkten Verfahren, wie dem Gaußschen Eliminationsverfahren, iterative Verfahren konstruiert und deren Effizienz zur Lösung von linearen Gleichungssystemen mit großen dünnbesetzten Matrizen untersucht. Ziel ist es jedenfalls, (im obigen Sinn) optimale Verfahren zu konstruieren.

4.2 Konstruktion von Iterationsverfahren

Viele Iterationsverfahren zur Lösung eines linearen Gleichungssystems

$$Ax = b \tag{4.1}$$

beruhen auf einer Umformung von (4.1) als Fixpunktgleichung der Form

$$x = Mx + Nb. \tag{4.2}$$

Daraus gewinnt man ein Iterationsverfahren (Fixpunktiteration), indem man eine bekannte Näherung $x^{(k)}$ der exakten Lösung x^* von (4.1) in dieser Gleichung auf der rechten Seite einsetzt, um durch die linke Seite eine neue Näherung $x^{(k+1)}$ zu erhalten:

$$x^{(k+1)} = Mx^{(k)} + Nb.$$

Man startet die Iteration mit einem beliebigen Startwert $x^{(0)}$.

Beispiel: Das **Jacobi-Verfahren** beruht auf folgenden Umformungen der i-ten Zeile des Gleichungssystems (4.1):

$$\begin{aligned}\sum_{j=1}^{n} a_{ij}x_j = b_i \quad &\Longleftrightarrow \quad a_{ii}x_i + \sum_{j\neq i} a_{ij}x_j = b_i \\ &\Longleftrightarrow \quad x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j\neq i} a_{ij}x_j\right),\end{aligned}$$

unter der Voraussetzung, dass $a_{ii} \neq 0$. Das Iterationsverfahren lautet also:

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

Mit der Bezeichnung

$$A = D - E - F,$$

wobei

$$D = \begin{pmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{pmatrix}, \; E = -\begin{pmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \; F = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & \cdots & 0 & 0 \end{pmatrix},$$

lässt sich das Jacobi-Verfahren auch folgendermaßen darstellen:

$$Dx^{(k+1)} - (E + F)x^{(k)} = b,$$

also

$$x^{(k+1)} = D^{-1}(E + F)x^{(k)} + D^{-1}b.$$

Das Verfahren hängt nicht von der Nummerierung der Variablen ab.

Beispiel: Zum Zeitpunkt der Berechnung von $x_i^{(k+1)}$ stehen die Komponenten $x_j^{(k+1)}$ mit $j < i$ bereits zur Verfügung und könnten statt den entsprechenden Komponenten der alten Näherung verwendet werden. Diese Idee führt auf das Iterationsverfahren

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

das man **Gauß–Seidel–Verfahren** nennt. Mit den eben eingeführten Bezeichnungen lässt sich das Verfahren auch folgendermaßen darstellen

$$Dx^{(k+1)} - Ex^{(k+1)} - Fx^{(k)} = b,$$

also

$$x^{(k+1)} = (D-E)^{-1}Fx^{(k)} + (D-E)^{-1}b.$$

Das Verfahren hängt von der Nummerierung der Variablen ab.

Jacobi–Verfahren und Gauß–Seidel–Verfahren sind Beispiele für das folgende allgemeine Konstruktionsprinzip, basierend auf einer additiven Aufspaltung von A: Sei

$$A = W - R$$

mit W regulär. Dann erhält man aus $Ax = b$ das System

$$Wx - Rx = b,$$

also

$$x = W^{-1}Rx + W^{-1}b,$$

das auf das Iterationsverfahren

$$x^{(k+1)} = W^{-1}Rx^{(k)} + W^{-1}b = Mx^{(k)} + Nb \tag{4.3}$$

mit

$$M = W^{-1}R = W^{-1}(W-A) = I - W^{-1}A \quad \text{und} \quad N = W^{-1}.$$

führt.

Das Jacobi–Verfahren entspricht der Setzung $W = D$, das Gauß–Seidel–Verfahren der Setzung $W = D - E$.

Im Folgenden wird eine andere Interpretation für (4.3) entwickelt:

Angenommen, eine Näherung $x^{(k)}$ der exakten Lösung x^* ist bekannt. Man wäre in einem Schritt am Ziel, wenn man jenen Zuwachs $p^{(k,*)}$ kennen würde, für den gilt:

$$x^{(k)} + p^{(k,*)} = x^*.$$

Durch Multiplikation mit A erhält man daraus die Gleichung

$$Ax^{(k)} + Ap^{(k,*)} = Ax^* = b.$$

Also sollte man den idealen Zuwachs aus der Gleichung

$$Ap^{(k,*)} = r^{(k)} \tag{4.4}$$

mit $r^{(k)} = b - Ax^{(k)}$, dem Residuum, berechnen.

Nun möchte man allerdings bei einem iterativen Verfahren gerade die Auflösung einer Gleichung mit der Matrix A vermeiden. Daher ersetzt man in der Gleichung (4.4) die Matrix A durch eine Näherung W und erhält damit das Iterationsverfahren

$$x^{(k+1)} = x^{(k)} + p^{(k)} \quad \text{mit} \quad W p^{(k)} = r^{(k)}. \tag{4.5}$$

Dieses Verfahren ist ident mit (4.3).

Es sind also zwei Forderungen an W zu stellen:

1. W soll A möglichst gut approximieren.
2. Gleichungssysteme der Form $Wp = r$ sollen leicht lösbar sein.

Die Setzung $W = A$ ist zwar optimal bezüglich der ersten Forderung, aber unbrauchbar wegen der zweiten Forderung. Annähernd umgekehrt ist die Situation beim Jacobi–Verfahren mit $W = D$.

In diesem Sinne lässt sich W als Approximation von A und $N = W^{-1}$ als approximative Inverse von A interpretieren.

Wird in (4.5) ein zusätzlicher Skalierungsfaktor $\tau > 0$ eingeführt, so erhält man die folgende Variante:

$$x^{(k+1)} = x^{(k)} + \tau\, p^{(k)} \quad \text{mit} \quad W p^{(k)} = r^{(k)} \tag{4.6}$$

oder kürzer

$$x^{(k+1)} = x^{(k)} + \tau\, W^{-1}(b - Ax^{(k)}). \tag{4.7}$$

Für $\tau < 1$ spricht man von einem gedämpften Verfahren, für $\tau > 1$ von einem extrapolierten Verfahren, für $\tau = 1$ erhält man das ursprüngliche (ungedämpfte) Verfahren. Meist wird jedoch (etwas ungenau) in allen Fällen von einem gedämpften Verfahren gesprochen.

Die Durchführung des Verfahrens (4.6) kann man in folgende Schritte trennen:

1. Berechne $r^{(k)} = b - Ax^{(k)}$.
2. Löse $Wp^{(k)} = r^{(k)}$.
3. Setze $x^{(k+1)} = x^{(k)} + \tau\, p^{(k)}$.

Beispiel: Das gedämpfte Jacobi–Verfahren lässt sich folgendermaßen darstellen:

$$x^{(k+1)} = x^{(k)} + \tau\, D^{-1}(b - Ax^{(k)})$$

Beispiel: Das einfachste (gedämpfte) Iterationsverfahren, das **Richardson-Verfahren**, entsteht für die Setzung $W = I$:

$$x^{(k+1)} = x^{(k)} + \tau\,(b - Ax^{(k)})$$

In gewisser Weise ist es der Prototyp aller bisheriger Iterationsverfahren: Wendet man das Richardson–Verfahren nicht direkt auf das System $Ax = b$ sondern auf das äquivalente System

$$\hat{A}x = \hat{b} \quad \text{mit} \quad \hat{A} = W^{-1}A,\ \hat{b} = W^{-1}b,$$

an, so erhält man (4.7).

4.3 Konvergenzanalyse

Die Vorteile von iterativen Verfahren gegenüber direkten Verfahren sind:

- der geringe Speicherbedarf;
- der geringe Aufwand pro Iteration;
- der unbedeutende Rundungsfehlereinfluss.

Diesen Vorteilen steht der Nachteil

- eines Verfahrensfehlers (durch Abbruch der Iteration)

gegenüber.

Um einen kleinen Verfahrensfehler mit relativ geringem Aufwand zu erreichen, muss das Iterationsverfahren schnell konvergieren. Daher wird nun die Konvergenz von Iterationsverfahren diskutiert.

Ein Iterationsverfahren der Form (4.7) lässt sich auch folgendermaßen schreiben:

$$x^{(k+1)} = Mx^{(k)} + Nb \quad \text{mit} \quad M = I - \tau W^{-1}A,\ N = \tau W^{-1}.$$

Für die exakte Lösung x^* gilt: $Ax^* = b$, also

$$x^* = x^* + \tau W^{-1}(b - Ax^*) = \left(I - \tau W^{-1}A\right)x^* + \tau W^{-1}b = Mx^* + Nb.$$

Durch Subtraktion folgt für den Fehler $e^{(l)} = x^{(l)} - x^*$:

$$e^{(k+1)} = Me^{(k)}.$$

Die Matrix M, die so genannte Iterationsmatrix, steuert also das Fehlerverhalten.

Für eine beliebige Vektornorm $\|.\|$ bzw. der dazugehörigen Matrixnorm gilt nun:

Satz 4.1. *Falls $\|M\| = q < 1$, konvergiert das Iterationsverfahren für beliebige Startwerte $x^{(0)}$ und es gilt*

1. $\|e^{(k+1)}\| \le q\,\|e^{(k)}\|$ *(q–lineare Konvergenz),*
2. $\|e^{(k)}\| \le C\,q^k$ *(r–lineare Konvergenz).*

Beweis. Aus $e^{(k+1)} = Me^{(k)}$ folgt

$$\|e^{(k+1)}\| \le \|M\|\,\|e^{(k)}\| = q\,\|e^{(k)}\|.$$

Durch mehrmalige Anwendung dieser Ungleichung erhält man

$$\|e^{(k)}\| \le q\,\|e^{(k-1)}\| \le q^2\,\|e^{(k-2)}\| \le \cdots \le q^k\,\|e^{(0)}\| = C\,q^k.$$

□

Bemerkung: Nach diesem Satz ist die Bedingung $\|M\| < 1$ hinreichend für die Konvergenz des Iterationsverfahrens für alle Startwerte. Notwendig und hinreichend für die Konvergenz des Iterationsverfahrens für alle Startwerte ist die (im Allgemeinen) schwächere Bedingung

$$\rho(M) < 1,$$

wobei $\rho(M) = \max\{|\lambda| : \lambda \text{ ist Eigenwert von } M\}$ den Spektralradius von M bezeichnet. Unter dieser schwächeren Bedingung lässt sich nur die r–lineare Konvergenz zeigen.

Aufwand eines Iterationsverfahrens:

Wie viele Iterationen werden nun benötigt, um einen Anfangsfehler um einen Faktor ε zu verkleinern? Gesucht ist also k mit

$$\|e^{(k)}\| \leq \varepsilon \|e^{(0)}\|.$$

Wegen $\|e^{(k)}\| \leq q^k \|e^{(0)}\|$, ist diese Bedingung sicher erfüllt, wenn k so groß ist, dass

$$q^k \leq \varepsilon.$$

Das ist gleichbedeutend mit der Forderung

$$\ln q^k \leq \ln \varepsilon,$$

also

$$k \ln q \leq \ln \varepsilon,$$

und somit

$$k \geq \frac{\ln \varepsilon}{\ln q} = \frac{-\ln \varepsilon}{-\ln q}.$$

Man benötigt also rund

$$k = \frac{-\ln \varepsilon}{-\ln q}$$

Iterationen.

Anwendung auf das Richardson-Verfahren für symmetrische Matrizen A

$$x^{(k+1)} = (I - \tau A)x^{(k)} + \tau b$$

mit $\tau > 0$, also

$$M = I - \tau A.$$

Zunächst wird die Euklidische Norm als Vektornorm, deren dazugehörige Matrixnorm die Spektralnorm ist, verwendet. Da M symmetrisch ist, gilt

$$\begin{aligned} \|M\|_2 &= \rho(M) = \max\{|\lambda| : \lambda \text{ ist Eigenwert von } M\} \\ &= \rho(I - \tau A) = \max\{|1 - \tau\lambda| : \lambda \text{ ist Eigenwert von } A\} \end{aligned}$$

Um die Bedingung $\|M\| < 1$ zu erfüllen, ist es offensichtlich notwendig, dass alle Eigenwerte von A positiv sind, dass also A positiv definit ist.

Für symmetrische positiv definite Matrizen A ist die Bedingung $\|M\|_2 < 1$ genau dann erfüllt, wenn

$$0 < \tau < \frac{2}{\lambda_{\max}(A)}.$$

Das Richardson-Verfahren konvergiert also, wenn der Dämpfungsparameter τ hinreichend klein gewählt wird. Um sicher zu sein, wie klein man den Dämpfungsparameter wählen muss, sollte man (eine gute obere Schranke für) den größten Eigenwert von A kennen. Zu kleine Parameter verschlechtern allerdings die Konvergenz. Eine in gewissem Sinne optimale Wahl für den Parameter τ erhält man durch die Forderung, die obere Schranke $q(\tau) = \max\{|1 - \tau\,\lambda| \ : \ \lambda \text{ ist Eigenwert von } A\}$ möglichst klein zu halten. Das führt auf die folgende Bedingung für den optimalen Parameter τ_{opt}:

$$1 - \tau_{\text{opt}}\lambda_{\min}(A) = \tau_{\text{opt}}\lambda_{\max}(A) - 1,$$

also

$$\tau_{\text{opt}} = \frac{2}{\lambda_{\min}(A) + \lambda_{\max}(A)}.$$

Für diese Wahl erhält man für den dazugehörigen Konvergenzfaktor q_{opt}:

$$\begin{aligned} q_{\text{opt}} &= q(\tau_{\text{opt}}) = 1 - \tau_{\text{opt}}\lambda_{\min}(A) = 1 - \frac{2\lambda_{\min}(A)}{\lambda_{\min}(A) + \lambda_{\max}(A)} \\ &= \frac{\lambda_{\max}(A) - \lambda_{\min}(A)}{\lambda_{\max}(A) + \lambda_{\min}(A)} = \frac{\lambda_{\max}(A)/\lambda_{\min}(A) - 1}{\lambda_{\max}(A)/\lambda_{\min}(A) + 1} = \frac{\kappa(A) - 1}{\kappa(A) + 1}. \end{aligned}$$

Für die Bestimmung des optimalen Dämpfungsparameters sollte man also (gute Näherungen für) den kleinsten und den größten Eigenwert von A kennen.

Es gilt

$$\frac{1}{-\ln q_{\text{opt}}} = \frac{1}{-\ln\left[1 - \frac{2}{\kappa(A)+1}\right]} \approx \frac{1}{\frac{2}{\kappa(A)+1}} = \frac{\kappa(A)+1}{2}$$

für große Konditionszahlen $\kappa(A)$. Daraus erhält man für die Anzahl der benötigten Iterationen

$$k = O(\kappa(A)).$$

Die Gesamtzahl der Operationen für dünnbesetzte Matrizen mit $O(n)$ Nichtnulleinträgen ist dann von der Größenordnung $O(\kappa(A)\,n)$.

Beispiel: Für die diskutierten FE-Diskretisierungen erhält man wegen

$$\kappa(K_h) = O(\frac{1}{h^2}) \quad \text{und} \quad n_h = O(\frac{1}{h^d})$$

eine Gesamtzahl von $O(\kappa(K_h)\,n_h) = O(1/h^{d+2}) = O(n_h^{1+2/d})$ Operationen. Für $d = 2$ benötigt man also $O(n_h^2)$ Operationen, das ist die gleiche Größenordnung des Aufwands

wie beim Gaußschen Eliminationsverfahren. Für $d = 3$ ergeben sich $O(n_h^{5/3})$ Operationen, also (asymptotisch) weniger Operationen als beim Gaußschen Eliminationsverfahren.

Für die bisherige Analyse wurde die Euklidische Norm verwendet. Dann ist allerdings das Abbruchkriterium

$$\|e^{(k)}\|_2 \leq \varepsilon \, \|e^{(0)}\|_2$$

nicht konstruktiv, da die Euklidische Norm des Fehlers ohne Kenntnis der exakten Lösung nicht berechenbar ist.

Eine andere Norm, die zu den gleichen Aussagen führt, ist die Energienorm $\|.\|_A$ (für den Fall, dass A symmetrisch und positiv definit ist): Beachte

$$\begin{aligned} \|M\|_A^2 &= \sup_x \frac{\|Mx\|_A^2}{\|x\|_A^2} = \sup_x \frac{(AMx, Mx)_2}{(Ax, x)_2} \\ &= \sup_x \frac{(MA^{1/2}x, MA^{1/2}x)_2}{(A^{1/2}x, A^{1/2}x)_2} = \sup_y \frac{(My, My)_2}{(y, y)_2} = \|M\|_2^2. \end{aligned}$$

Aussagen über die Konvergenz des Richardson–Verfahren in der Energienorm werden im nächsten Abschnitt benötigt. Diese Norm liefert ebenfalls kein konstruktives Abbruchkriterium.

Eine weitere interessante Norm ist die Residuumsnorm $\|.\|_{A^TA}$, die für jede reguläre Matrix A eingeführt werden kann. Der Fehler in dieser Norm ist ohne Kenntnis der exakten Lösung berechenbar:

$$\|e^k\|_{A^TA}^2 = (A^TAe^k, e^k)_2 = (Ae^k, Ae^k)_2 = (Ax^k - b, Ax^k - b)_2 = (r^k, r^k)_2 = \|r^k\|_2^2$$

Für den Fall A symmetrisch und positiv definit stimmt die Residuumsnorm natürlich mit $\|.\|_{A^2}$ überein. Wie oben lässt sich zeigen, dass $\|M\|_{A^2} = \|M\|_2$, woraus wieder alle früher gezeigten Aussagen folgen.

Die drei diskutierten Normen für den Fall A symmetrisch und positiv definit sind die Spezialfälle $s = 0$, $s = 1$ und $s = 2$ aus der Familie von Normen $\|.\|_{(s)} = \|.\|_{A^s}$, für die gleichen Konvergenzaussagen folgen.

Die Aussagen über die Konvergenz des Richardson-Verfahrens lassen sich auf allgemeinere präkonditionierte (und gedämpfte) Richardson-Verfahren der Form

$$x^{(k+1)} = (I - \tau \, W^{-1}A)x^{(k)} + \tau \, W^{-1}b$$

mit $\tau > 0$ übertragen:

Satz 4.2. *1. Für symmetrische und positiv definite Matrizen W und A konvergiert das Verfahren, falls*

$$0 < \tau < \frac{2}{\lambda_{max}(W^{-1}A)}.$$

2. Für die (optimale) Parameterwahl

$$\tau_{opt} = \frac{2}{\lambda_{min}(W^{-1}A) + \lambda_{max}(W^{-1}A)}$$

gelten die Fehlerabschätzungen

$$\|e^{(k+1)}\|_{(s)} \leq q_{opt} \|e^{(k)}\|_{(s)} \quad \textit{und} \quad \|e^{(k)}\|_{(s)} \leq q_{opt}^k \|e^{(0)}\|_{(s)}$$

mit

$$q_{opt} = \frac{\kappa(W^{-1}A) - 1}{\kappa(W^{-1}A) + 1}$$

und den folgende Normen für $s = 0, 1, 2$:

$$\|e\|_{(0)} = (We, e)^{1/2}, \quad \|e\|_{(1)} = (Ae, e)^{1/2}, \quad \|e\|_{(2)} = (W^{-1}Ae, Ae)^{1/2}.$$

3. Für den Fehler $e^{(k)} = x^{(k)} - x^*$ gilt:

$$\|e^{(k)}\|_{(2)}^2 = (W^{-1}r^{(k)}, r^{(k)}) = (p^{(k)}, r^{(k)}).$$

Beweis. Es gilt

$$\begin{aligned}
\|M\|_W^2 &= \sup_x \frac{(WMx, Mx)_2}{(Wx, x)_2} = \sup_x \frac{(W^{1/2}MW^{-1/2}W^{1/2}x, W^{1/2}MW^{-1/2}W^{1/2}x)_2}{(W^{1/2}x, W^{1/2}x)_2} \\
&= \sup_y \frac{(W^{1/2}MW^{-1/2}y, W^{1/2}MW^{-1/2}y)_2}{(y, y)_2} \\
&= \sup_y \frac{\|W^{1/2}MW^{-1/2}y\|_2^2}{\|y\|_2^2} = \|W^{1/2}MW^{-1/2}\|_2^2.
\end{aligned}$$

Die Matrix $W^{1/2}MW^{-1/2} = I - \tau\, W^{-1/2}AW^{-1/2}$ ist symmetrisch. Daher gilt:

$$\|W^{1/2}MW^{-1/2}\|_2 = \rho(W^{1/2}MW^{-1/2}) = \rho(M).$$

Alles Weitere folgt aus der Analyse von vorhin.

Die Analyse bezüglich der beiden anderen Normen verläuft völlig analog. □

Nach diesem Satz ist es also zweckmäßig, die Matrix W so zu wählen, dass die Konditionszahl von $W^{-1}A$ möglichst klein ist. Man nennt diese Strategie Präkonditionierung und nennt in diesem Zusammenhang W eine Präkonditionierungsmatrix oder einen Präkonditionierer.

Der Idealfall $W = A$ führt auf die kleinstmögliche Konditionszahl 1, ist aber unrealistisch, da das Verfahren die Lösung einer Gleichung der Form $Wp = r$ erfordert. Die bisherigen klassischen Iterationsverfahren, auf die der letzte Satz anwendbar ist (Jacobi-Verfahren, Gauß-Seidel-Verfahren) führen zu keiner wesentlichen Verkleinerung der Konditionszahl von $W^{-1}A$.

Gute Präkonditioner kann man z.B. durch additive Aufspaltungen der Form

$$A = LU - R$$

gewinnen, wobei L eine linke untere Dreiecksmatrix und U eine rechte obere Dreiecksmatrix ist. Falls $R = 0$ gewählt wird, spricht man von vollständiger Dreieckszerlegung (Gaußsche Elimination), die Berechnung von L und U und die Lösung von $Wp = r$ sind in diesem Fall allerdings zu aufwendig. Es gibt Bedingungen an R, die eine wesentlich effizientere Zerlegung erlauben.

Es gibt Verallgemeinerungen des Jacobi– und des Gauß–Seidel–Verfahrens, so genannte additive und multiplikative Schwarz-Methoden, die zu einer erheblichen Verkleinerung der Konditionszahl führen können.

4.4 Das Gradientenverfahren und das cg-Verfahren

Der für Anwendungsprobleme wichtige Spezialfall eines Gleichungssystems

$$Ax = b \tag{4.8}$$

mit einer symmetrischen und positiv definiten Matrix A ermöglicht die Durchführung eines besonders schnellen Iterationsverfahrens.

Für symmetrische und positiv definite Matrizen ist das Problem (4.8) zum Minimierungsproblem

$$J(x) = \frac{1}{2}(Ax, x)_2 - (b, x)_2 \longrightarrow \min!$$

äquivalent, da Gradient und Hessematrix von J in einem Punkt x durch

$$\nabla J(x) = Ax - b, \quad \nabla^2 J(x) = A$$

gegeben sind. J wird als Energiefunktional bezeichnet.

Die Richtung des steilsten Abstiegs von J in einem Punkt x ist durch $-\nabla J(x) = b - Ax = r$, also dem Residuum gegeben.

Ausgehend von einer Näherung $x^{(0)}$ ist es naheliegend, zuerst in dieser Richtung nach der nächsten Näherung zu suchen:

$$x^{(1)} = x^{(0)} + \alpha^{(0)} r^{(0)},$$

wobei die Zahl $\alpha^{(0)}$ so gewählt wird, dass $J(x^{(1)})$ minimal wird.

Für eine allgemeine Suchrichtung $p^{(0)}$ gilt:

$$\begin{aligned} J(x^{(0)} + \alpha\, p^{(0)}) &= \frac{1}{2}(A(x^{(0)} + \alpha\, p^{(0)}), x^{(0)} + \alpha\, p^{(0)}) - (b, x^{(0)} + \alpha\, p^{(0)})_2 \\ &= \frac{1}{2}(Ax^{(0)}, x^{(0)})_2 - (b, x^{(0)})_2 + \frac{\alpha}{2}\,(Ap^{(0)}, x^{(0)})_2 + \frac{\alpha}{2}\,(Ax^{(0)}, p^{(0)})_2 \\ &\quad - \alpha\,(b, p^{(0)})_2 + \frac{\alpha^2}{2}\,(Ap^{(0)}, p^{(0)})_2 \\ &= \frac{\alpha^2}{2}\,(Ap^{(0)}, p^{(0)})_2 - \alpha\,(r^{(0)}, p^{(0)})_2 + \frac{1}{2}(Ax^{(0)}, x^{(0)})_2 - (b, x^{(0)})_2. \end{aligned}$$

$J(x^{(0)} + \alpha p^{(0)})$ ist genau dann minimal, wenn

$$\frac{d}{d\alpha} J(x^{(0)} + \alpha\, p^{(0)})\Big|_{\alpha=\alpha^{(0)}} = 0,$$

d.h.:

$$\alpha^{(0)}\,(Ap^{(0)}, p^{(0)}) - (r^{(0)}, p^{(0)}) = 0. \tag{4.9}$$

Das führt auf die Setzung

$$\alpha^{(0)} = \frac{(r^{(0)}, p^{(0)})}{(Ap^{(0)}, p^{(0)})}.$$

und man erhält die nächste Näherung

$$x^{(1)} = x^{(0)} + \alpha^{(0)} r^{(0)}.$$

Die Richtung des steilsten Abstiegs von J im Punkt $x^{(1)}$ lässt sich leicht berechnen:

$$r^{(1)} = b - Ax^{(1)} = b - Ax^{(0)} - \alpha^{(0)} Ar^{(0)} = r^{(0)} - \alpha^{(0)} Ar^{(0)}.$$

Setzt man diese Strategie fort, erhält man das so genannte Gradientenverfahren:

Für einen gegebenen Startwert $x^{(0)}$ setzt man $r^{(0)} = b - Ax^{(0)}$, dann führt man für $k = 0, 1, \ldots$ die folgende Iteration durch:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \alpha^{(k)}\, r^{(k)} \quad \text{mit} \quad \alpha^{(k)} = \frac{(r^{(k)}, r^{(k)})}{(Ar^{(k)}, r^{(k)})}, \\ r^{(k+1)} &= r^{(k)} - \alpha^{(k)}\, Ar^{(k)}. \end{aligned}$$

Konvergenzeigenschaften des Gradientenverfahrens

Ein Schritt des Gradientenverfahrens entspricht einem Schritt des Richardson-Verfahrens mit der Parameterwahl $\tau = \alpha^{(k)}$. Die Konvergenz lässt sich besonders einfach mit Hilfe der Norm $\|x\|_A = \sqrt{(Ax, x)}$ untersuchen: Es gilt folgender Zusammenhang zwischen dem Fehler in der Energienorm und dem Energiefunktional:

$$\begin{aligned} \|x - x^*\|_A^2 &= (A(x - x^*), x - x^*) \\ &= (Ax, x) - (Ax, x^*) - (Ax^*, x) + (Ax^*, x^*) \\ &= (Ax, x) - 2\,(Ax^*, x) + (Ax^*, x^*) \\ &= [(Ax, x) - 2\,(b, x)] - [(Ax^*, x^*) - 2\,(b, x^*)] = 2\,[J(x) - J(x^*)], \end{aligned}$$

Daraus folgt:

$$\begin{aligned} \|x^{(1)} - x^*\|_A^2 &= 2\,[J(x^{(1)}) - J(x^*)] \le 2\,[J(x^{(0)} + \tau_{\text{opt}}\, r^{(0)}) - J(x^*)] \\ &= \|(x^{(0)} + \tau_{\text{opt}}\, r^{(0)}) - x^*\|_A^2 \\ &\overset{(S.4.2.2)}{\le} q_{\text{opt}}^2\, \|x^{(0)} - x^*\|_A^2. \end{aligned}$$

Man beachte, dass $x^{(0)} + \tau_{\text{opt}} r^{(0)}$ der nächste Iterationspunkt des Richardson-Verfahrens mit optimaler Parameterwahl ist. Die letzte Abschätzung folgt dann aus Satz 4.2.

Ein Schritt des Gradientenverfahrens ist also nicht schlechter als ein Schritt des Richardson-Verfahrens mit optimaler Parameterwahl. Während man allerdings für die Durchführung des Richardson-Verfahrens mit optimaler Parameterwahl den kleinsten und größten Eigenwert (oder zumindest sehr gute Näherungen dafür) kennen muss, ist die Durchführung des Gradientenverfahrens ohne Kenntnis von Eigenwerten möglich.

Damit ist folgender Satz bewiesen:

Satz 4.3. *Sei A eine symmetrische und positiv definite Matrix mit kleinstem Eigenwert $\lambda_{min}(A)$ und größtem Eigenwert $\lambda_{max}(A)$. Dann konvergiert das Gradientenverfahren und es gelten die Fehlerabschätzungen*

$$\|x^{(k+1)} - x^*\|_A \leq q \, \|x^{(k)} - x^*\|_A \quad \textit{und} \quad \|x^{(k)} - x^*\|_A \leq q^k \, \|x^{(0)} - x^*\|_A$$

mit

$$q = \frac{\lambda_{max}(A) - \lambda_{min}(A)}{\lambda_{max}(A) + \lambda_{min}(A)} = \frac{\kappa(A) - 1}{\kappa(A) + 1}.$$

Daraus ergibt sich für die Anzahl der Iterationen bei großer Konditionszahl die gleiche Abschätzung wie beim Richardson-Verfahren mit optimaler Parameterwahl:

$$k \approx \frac{-\ln \varepsilon}{-\ln q} \approx (-\ln \varepsilon) \frac{\kappa(A) + 1}{2} = O(\kappa(A))$$

Der erste Schritt des **cg-Verfahrens** stimmt mit dem ersten Schritt des Gradientenverfahrens überein:

$$x^{(1)} = x^{(0)} + \alpha^{(0)} p^{(0)}$$

mit $p^{(0)} = r^{(0)}$. Wie vorhin erhält man für das nächste Residuum:

$$r^{(1)} = r^{(0)} - \alpha^{(0)} \, A p^{(0)}.$$

Die beste Wahl für die nächste Suchrichtung wäre $p^{(1,*)} = x^* - x^{(1)}$, die natürlich nicht verfügbar ist. Aus (4.9) folgt

$$(A p^{(1,*)}, p^{(0)})_2 = (r^{(1)}, p^{(0)})_2 = 0.$$

Diese Eigenschaft motiviert die folgende Bedingung, die für die tatsächliche nächste Suchrichtung $p^{(1)}$ gefordert wird:

$$(A p^{(1)}, p^{(0)}) = 0. \tag{4.10}$$

Zwei Richtungen, die (4.10) erfüllen, heißen konjugierte Richtungen bezüglich $(.,.)_A$ oder A–orthogonal.

Das Residuum $r^{(1)}$ ist orthogonal aber im Allgemeinen nicht A–orthogonal zu $p^{(0)}$. Aber die Modifikation

$$p^{(1)} = r^{(1)} - \beta^{(0)} \, p^{(0)}$$

erfüllt (4.10), falls

$$(A(r^{(1)} - \beta^{(0)}\, p^{(0)}), p^{(0)})_2 = 0,$$

also

$$\beta^{(0)} = \frac{(r^{(1)}, Ap^{(0)})_2}{(Ap^{(0)}, p^{(0)})_2}.$$

Setzt man diese Strategie analog fort, so erhält man das so genannte cg–Verfahren:

Mit den Anfangssetzungen $r^{(0)} = b - Ax^{(0)}$ und $p^{(0)} = r^{(0)}$ lautet der allgemeine Schritt des cg-Verfahrens für $k = 0, 1, 2 \ldots$:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} + \alpha^{(k)} p^{(k)} \quad \text{mit} \quad \alpha^{(k)} = \frac{(r^{(k)}, p^{(k)})}{(Ap^{(k)}, p^{(k)})}, \\ r^{(k+1)} &= r^{(k)} - \alpha^{(k)} Ap^{(k)}, \\ p^{(k+1)} &= r^{(k+1)} - \beta^{(k)} p^{(k)} \quad \text{mit} \quad \beta^{(k)} = \frac{(r^{(k+1)}, Ap^{(k)})}{(Ap^{(k)}, p^{(k)})}. \end{aligned}$$

Konvergenzeigenschaften des cg-Verfahrens:

Angenommen, $\bar{x}^{(1)}$, $\bar{x}^{(2)}$, ... bezeichnen die Näherungen, die man mit Hilfe des Richardson–Verfahrens gewinnt, wenn man mit $\bar{x}^{(0)} = x^{(0)}$ startet. Es lässt sich leicht zeigen, dass die k–te Näherung des cg–Verfahrens als Linearkombination des Startwerts und der ersten k Näherungen des Richardson–Verfahrens geschrieben werden kann:

$$x^{(k)} = \omega_0^{(k)}\, \bar{x}^{(0)} + \omega_1^{(k)}\, \bar{x}^{(1)} + \ldots + \omega_k^{(k)}\, \bar{x}^{(k)} \quad \text{mit} \quad \omega_0^{(k)} + \omega_1^{(k)} + \ldots + \omega_k^{(k)} = 1$$

und dass es jene Linearkombination dieser Form ist, die den kleinstmöglichen Wert von J ergibt. Daraus folgt: Bei exakter Rechnung führt das cg-Verfahren nach n Schritten zur exakten Lösung des Gleichungssystems. In diesem Sinne ist es ein direktes Verfahren.

Wichtiger allerdings für die Lösung großer Gleichungssysteme mit dünnbesetzten Matrizen ist das cg-Verfahren als Iterationsverfahren, da es bereits nach wesentlich weniger als n Schritten eine gute Näherung der exakten Lösung liefern kann. Es gilt nämlich:

Satz 4.4. *Sei $A \in \mathbb{R}^{n\times n}$ eine symmetrische, positiv definite Matrix. Dann konvergiert das cg-Verfahren und es gilt die Fehlerabschätzung*

$$\|x^{(k)} - x^*\|_A \leq 2\frac{q^k}{1+q^{2k}}\|x^{(0)} - x^*\|_A \leq 2\, q^k\, \|x^{(0)} - x^*\|_A$$

mit

$$q = \frac{\sqrt{\lambda_{max}(A)} - \sqrt{\lambda_{min}(A)}}{\sqrt{\lambda_{max}(A)} + \sqrt{\lambda_{min}(A)}} = \frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}.$$

Um den Anfangsfehler $\|x^{(0)} - x^*\|_A$ um einen Faktor ε zu reduzieren, muss nach diesem Satz gelten:

$$2\, q^k \leq \varepsilon.$$

Also genügen für große Konditionszahlen

$$k \approx \frac{-\ln(\varepsilon/2)}{-\ln q} = \frac{-\ln(\varepsilon/2)}{-\ln\left[1 - \frac{2}{\sqrt{\kappa(A)}+1}\right]} \approx \frac{-\ln(\varepsilon/2)}{\frac{2}{\sqrt{\kappa(A)}+1}} = (-\ln(\varepsilon/2))\frac{\sqrt{\kappa(A)}+1}{2} = O(\sqrt{\kappa(A)})$$

Iterationen.

Beispiel: Für die diskutierten FE-Diskretisierungen erhält man wegen

$$\kappa(K_h) = O(\frac{1}{h^2}) \quad \text{und} \quad n_h = O(\frac{1}{h^d})$$

eine Gesamtzahl von $O(\sqrt{\kappa(K_h)}\, n_h) = O(1/h^{d+1}) = O(n_h^{(1+1/d)})$ Operationen. Für $d = 2$ benötigt man $O(n_h^{3/2})$ Operationen. Für $d = 3$ ergeben sich $O(n_h^{4/3})$ Operationen, also in beiden Fällen (asymptotisch) weniger Operationen als beim Gaußschen Eliminationsverfahren.

Bemerkung: Das cg-Verfahren kann noch weiter verbessert werden, wenn man es nicht auf das ursprüngliche Gleichungssystem $Ax = b$ sondern auf das vorkonditioniertes Problem

$$W^{-1}Ax = W^{-1}b$$

anwendet, wobei W so zu wählen ist, dass $\kappa(W^{-1}A)$ möglichst klein ist.

Bemerkung: Es gibt Varianten des cg-Verfahrens auch für symmetrische, nicht positiv definite Matrizen und für nichtsymmetrische Matrizen.

Bemerkung: Zu den effizientesten Verfahren für diskretisierte Gleichungen gehören die so genannten Mehrgitterverfahren, für die sich für viele Anwendungen zeigen läßt, dass sie einen Konvergenzfaktor $q < 1$ unabhängig von h besitzen. Damit erhält man für die Anzahl k der notwendigen Iterationen, um einen Anfangsfehler um einen Faktor ε zu reduzieren:

$$k \approx \frac{-\ln \varepsilon}{-\ln q} = O(1).$$

Somit benötigt man nur $O(k\, n_h) = O(n_h)$ Operationen für die Durchführung des Mehrgitterverfahrens, es ist also ein optimales Verfahren.

Kapitel 5

Nichtlineare Gleichungssysteme

Lineare Probleme sind oft Idealisierungen eigentlich nichtlinearer Probleme.

Beispiel: Die Wärmeleitgleichung ist nur linear, wenn man annimmt, dass die Materialeigenschaften, wie die Wärmeleitfähigkeit, unabhängig von der Temperatur sind. Tatsächlich ist in vielen Situationen diese Temperaturabhängigkeit zu berücksichtigen. Dann besitzt die stationäre Wärmeleitgleichung die Form

$$-\left(\lambda(u(x))u'(x)\right)' = f(x).$$

Durch Diskretisierung solcher nichtlinearer Differentialgleichungsprobleme (mit entsprechenden Randbedingungen) erhält man dann nichtlineare Gleichungssysteme, die im Allgemeinen von folgender Form sind:

Gesucht sind n Zahlen $x_1, x_2, \ldots, x_n \in \mathbb{R}$ mit

$$\begin{aligned} F_1(x_1, x_2, \ldots, x_n) &= 0, \\ F_2(x_1, x_2, \ldots, x_n) &= 0, \\ &\vdots \\ F_n(x_1, x_2, \ldots, x_n) &= 0, \end{aligned}$$

wobei die Funktionen $F_i \colon \mathbb{R}^n \to \mathbb{R}$ für $i = 1, 2, \ldots, n$ vorgegeben sind, oder kurz

$$F(x) = 0$$

mit $x = (x_1, x_2, \ldots, x_n)^T$, $F \colon \mathbb{R}^n \to \mathbb{R}^n$, $F(x) = (F_1(x), F_2(x), \ldots, F_n(x))^T$. Man spricht von einem nichtlinearen Gleichungssystem. Etwas genauer müsste man eigentlich von einem nicht notwendig linearen Gleichungssystem sprechen, da der Spezialfall eines linearen Gleichungssystems durch die Setzung $F(x) = Ax - b$ nicht ausgeschlossen ist.

Bis auf vereinzelte Spezialfälle stehen keine direkten Verfahren zur Lösung nichtlinearer Gleichungen zur Verfügung. Es kommen daher im Allgemeinen nur Iterationsverfahren zur Berechnung einer Näherungslösung eines nichtlinearen Gleichungssystems in Frage.

Wie im linearen Fall erhält man durch Umwandlung eines nichtlinearen Gleichungssystems

$$F(x) = 0$$

in Fixpunktform

$$x = G(x)$$

ein Iterationsverfahren der Form

$$x^{(k+1)} = G(x^{(k)}), \quad k = 1, 2, \ldots, \tag{5.1}$$

für einen geeignet gewählten Startwert $x^{(0)}$. Ein Iterationsverfahren der Form (5.1) heißt ein stationäres Einschrittverfahren. (Ein Verfahren heißt Einschrittverfahren, wenn zur Berechnung der nächsten Näherung nur der aktuelle Näherungswert verwendet wird. Ein Verfahren heißt stationär, wenn die Berechnungsformel unabhängig vom Iterationsindex ist.) Man spricht auch von einer Fixpunktiteration oder einem Verfahren der sukzessiven Approximation.

Beispiel: Gesucht ist die Lösung der Gleichung

$$F(x) = e^{\frac{x}{2}} + x - 2 = 0.$$

Da die Gleichung $F(x) = 0$ auch in der Form

$$x = 2 - e^{\frac{x}{2}} = G(x)$$

geschrieben werden kann, bietet sich als eine Alternative zum Newton-Verfahren auch folgende Fixpunktiteration an:

$$x^{(k+1)} = G(x^{(k)}) = 2 - e^{\frac{x^{(k)}}{2}}.$$

Für den Startwert $x^{(0)} = 1$ erhält man

$$\begin{aligned}
x^{(0)} &= 1.000, \\
x^{(1)} &= 0.351, \\
x^{(2)} &= 0.808, \\
x^{(3)} &= 0.502, \\
x^{(4)} &= 0.715, \\
x^{(5)} &= 0.571, \\
x^{(6)} &= \underline{0.6}70, \\
x^{(7)} &= \underline{0.6}02, \\
x^{(8)} &= \underline{0.6}48, \\
x^{(9)} &= \underline{0.6}16, \\
x^{(10)} &= \underline{0.6}38, \\
x^{(11)} &= \underline{0.62}4, \\
x^{(12)} &= \underline{0.63}4,
\end{aligned}$$

also eine sehr langsam gegen die exakte Lösung $x^* = 0.629\,846\,115\,690\,812\,2$ konvergente Folge von Näherungen (eine richtige Dezimalstelle in 5 Iterationen).

Wichtige Aussagen über die Konvergenz von stationären Einschrittverfahren liefert der Banachsche Fixpunktsatz:

Satz 5.1 (Banach)**.** *Sei D eine abgeschlossene Menge und sei $G\colon D \to \mathbb{R}^n$ eine Abbildung mit $G(D) \subset D$ und*

$$\|G(y) - G(x)\| \leq q\,\|y - x\| \quad \textit{für alle } x, y \in D$$

mit $q < 1$. Dann existiert ein eindeutiger Fixpunkt $\bar{x}$ in D, die Folge

$$x^{(k+1)} = G(x^{(k)})$$

konvergiert gegen diesen Fixpunkt $\bar{x}$ für alle Startwerte $x^{(0)} \in D$ und es gelten die folgenden Fehlerabschätzungen:

1. $\|e^{(k+1)}\| \leq q \cdot \|e^{(k)}\|$ *(q–lineare Konvergenz),*

2. $\|e^{(k)}\| \leq Cq^k$ *(r–lineare Konvergenz)*

für eine Konstante C.

Bemerkung: Sei x^* ein Fixpunkt von G im Inneren des Definitionsbereiches und sei G differenzierbar in x^*. Dann folgt aus der Bedingung $\|G'(x^*)\| < 1$ die q–lineare Konvergenz, aus der Bedingung $\rho(G'(x^*)) < 1$ zumindest die r–lineare Konvergenz für die Startwerte, die hinreichend nahe bei x^* liegen (Satz von Ostrowski). Im Spezialfall $n = 1$ erhält man lokal monotone Konvergenz für $0 < G'(x^*) < 1$ bzw. alternierende Konvergenz für $-1 < G'(x^*) < 0$.

Beispiel: Durch Taylor-Entwicklung sieht man sofort, dass die Iterationsfolge aus dem obigen Beispiel q–linear konvergiert:

$$x^{(k+1)} - x^* = G(x^{(k)}) - G(x^*) \approx G'(x^*)(x^{(k)} - x^*),$$

also

$$|e^{(k+1)}| \leq q\,|e^{(k)}|$$

mit $q \approx |G'(x^*)| = e^{x^*/2}/2 < 1$.

5.1 Das Newton-Verfahren

Das wichtigste Iterationsverfahren zur Lösung nichtlinearer Gleichungen ist das Newton-Verfahren, das im Folgenden abgeleitet wird.

Angenommen, eine Näherung $x^{(k)}$ der exakten Lösung x^* des nichtlinearen Gleichungssystems

$$F(x) = 0$$

ist bekannt. Man wäre in einem Schritt am Ziel, wenn man jenen Zuwachs $p^{(k,*)}$ kennen würde, für den gilt:

$$x^{(k)} + p^{(k,*)} = x^*.$$

Daraus erhält man die Bedingung

$$F(x^{(k)} + p^{(k,*)}) = 0. \tag{5.2}$$

Nun wird die linke Seite durch Taylor-Entwicklung linearisiert:

$$F(x^{(k)} + p^{(k,*)}) \approx F(x^{(k)}) + F'(x^{(k)})p^{(k,*)}.$$

wobei $F'(x)$ die Jacobi-Matrix von F an der Stelle x bezeichnet:

$$F'(x) = \begin{pmatrix} \frac{\partial F_1}{\partial x_1}(x) & \frac{\partial F_1}{\partial x_2}(x) & \cdots & \frac{\partial F_1}{\partial x_n}(x) \\ \frac{\partial F_2}{\partial x_1}(x) & \frac{\partial F_2}{\partial x_2}(x) & \cdots & \frac{\partial F_2}{\partial x_n}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1}(x) & \frac{\partial F_n}{\partial x_2}(x) & \cdots & \frac{\partial F_n}{\partial x_n}(x) \end{pmatrix}.$$

Aus der idealen Bedingungsgleichung (5.2), die allerdings nicht konstruktiv ist, wird durch diese Linearisierung die Bedingung

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

Man erhält somit die Iterationsvorschrift:

$$x^{(k+1)} = x^{(k)} + p^{(k)} \quad \text{mit} \quad F'(x^{(k)})p^{(k)} = -F(x^{(k)}). \tag{5.3}$$

Dieses Iterationsverfahren heißt Newton-Verfahren.
Geometrische Interpretation: Offensichtlich gilt:

$$F(x^{(k)}) + F'(x^{(k)})(x^{(k+1)} - x^{(k)}) = 0,$$

also

$$L_k(x^{(k+1)}) = 0.$$

mit

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

Die (affin) lineare Funktion $L_k(x)$ besitzt an der Stelle $x^{(k)}$ den gleichen Funktionswert und die gleiche erste Ableitung wie $F(x)$.

$$F(x) \approx F(x^{(k)}) + F'(x^{(k)})(x - x^{(k)}) = L_k(x).$$

Die entscheidende Idee des Newton-Verfahrens ist also die Approximation der nichtlinearen Funktion $F(x)$ durch die (affin) lineare Funktion $L_k(x)$ in jedem Iterationsschritt (Linearisierung), deren Nullstelle dann die nächste Näherung für x^* bestimmt.

Im Fall $n = 1$ erhält man also den nächsten Iterationspunkt als Schnittpunkt der Tangente an den Graphen von F im Punkt $x^{(k)}$ mit der x-Achse.

Durch Elimination von $p^{(k)}$ lässt sich das Newton–Verfahren auch folgendermaßen darstellen:

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

Das Newton–Verfahren ist also ein stationäres Einschrittverfahren

$$x^{(k+1)} = G(x^{(k)})$$

mit

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

Für den Spezialfall $n = 1$ erhält man die einfachere Darstellung:

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

Man beachte allerdings, dass zur Durchführung des Verfahrens nicht die Inverse der Jacobi-Matrix $F'(x)$ gebildet wird, sondern der Zuwachs $p^{(k)}$ durch Lösen des linearen Gleichungssystems $F'(x^{(k)})p^{(k)} = -F(x^{(k)})$ bestimmt wird:

Die Durchführung des Verfahrens (5.3) kann man in folgende Schritte trennen:

1. Berechne $r^{(k)} = -F(x^{(k)})$, $W^{(k)} = F'(x^{(k)})$.
2. Löse $W^{(k)} p^{(k)} = r^{(k)}$.
3. Setze $x^{(k+1)} = x^{(k)} + p^{(k)}$.

Konvergenzeigenschaften des Newton-Verfahrens

Satz 5.2. *Sei $F\colon \mathbb{R}^n \to \mathbb{R}^n$ zweimal differenzierbar. $x^* \in \mathbb{R}^n$ sei Nullstelle von F, also $F(x^*) = 0$, und $F'(x^*)$ sei regulär. Dann konvergiert das Newton-Verfahren für Startwerte $x^{(0)}$, die hinreichend nahe bei x^* liegen und es gelten die folgenden Fehlerabschätzungen:*

1. *$\|e^{(k+1)}\| \leq C \cdot \|e^{(k)}\|^2$ (q–quadratische Konvergenz).*

2. $\|e^{(k)}\| \leq C' q^{2^k-1}$ *(r–quadratische Konvergenz).*

für Konstanten C*,* C' *und* $q < 1$*.*

Beweis. (Für den Fall $n = 1$) Das Newton-Verfahren lässt sich folgendermaßen darstellen:

$$x^{(k+1)} = G(x^{(k)}) \quad \text{mit} \quad G(x) = x - \frac{F(x)}{F'(x)}.$$

Für x^* gilt natürlich $x^* = G(x^*)$.

Es gilt:

$$G'(x) = 1 - \frac{F'(x)^2 - F(x)F''(x)}{F'(x)^2},$$

also

$$G'(x^*) = 0.$$

Für den Fehler $e^{(k)} = x^{(k)} - x^*$ erhält man damit:

$$\begin{aligned} e^{(k+1)} &= x^{(k+1)} - x^* = G(x^{(k)}) - G(x^*) \\ &\approx G'(x^*)(x^{(k)} - x^*) + \frac{1}{2}G''(x^*)(x^{(k)} - x^*)^2 = \frac{1}{2}G''(x^*)\left(e^{(k)}\right)^2 \end{aligned}$$

Also folgt (korrekterweise erst durch Restgliedabschätzung) die erste Fehlerabschätzung:

$$|e^{(k+1)}| \leq C\,|e^{(k)}|^2.$$

Daraus ergibt sich leicht die zweite Abschätzung:

$$\begin{aligned} |e^{(k)}| &\leq C\,|e^{(k-1)}|^2 \leq C\,C^2\,|e^{(k-2)}|^4 \leq C\,C^2\,C^4\,|e^{(k-3)}|^8 \leq \cdots \\ &\leq C\,C^2\,C^4 \cdots C^{2^{k-1}}\,|e^{(0)}|^{2^k} = C^{1+2+4+\cdots+2^{k-1}}\,|e^{(0)}|^{2^k} \\ &= C^{2^k-1}\,|e^{(0)}|^{2^k} = \left(C|e^{(0)}|\right)^{2^k-1}|e^{(0)}| = C'\,q^{2^k-1} \end{aligned}$$

□

Bemerkung: Die Bedingung, dass $F'(x^*)$ regulär ist, reduziert sich für $n = 1$ auf die Forderung $F'(x^*) \neq 0$. Es wird also vorausgesetzt, dass die Tangente an den Graphen von F in der Nullstelle x^* nicht waagrecht ist. Solche Nullstellen heißen einfache Nullstellen. Im Falle $F'(x^*) = 0$ spricht man von mehrfachen Nullstellen.

Zur Illustration dieser Begriffe betrachte man folgendes Zahlenbeispiel:

1. Für $C = 1$ und $q = 0.1$ erhält man für die Fehler einer r–linear konvergenten Folge:

$$\begin{aligned} \|e^{(0)}\| &\leq 1.000\ 000 \\ \|e^{(1)}\| &\leq 0.100\ 000 \\ \|e^{(2)}\| &\leq 0.010\ 000 \\ \|e^{(3)}\| &\leq 0.001\ 000 \\ \|e^{(4)}\| &\leq 0.000\ 100 \\ \|e^{(5)}\| &\leq 0.000\ 010 \end{aligned}$$

Bei einer r–linear konvergenten Folge nimmt also die Anzahl der „richtigen“ Stellen linear zu, hier eine Stelle pro Iteration.

2. Für $C' = 1$ und $q = 0.1$ erhält man für die Fehler einer r–quadratisch konvergenten Folge:

$$\begin{aligned} \|e^{(0)}\| &\leq 1.000\ 000\ 000 \\ \|e^{(1)}\| &\leq 0.010\ 000\ 000 \\ \|e^{(2)}\| &\leq 0.000\ 100\ 000 \\ \|e^{(3)}\| &\leq 0.000\ 000\ 010 \end{aligned}$$

Bei einer r–quadratisch konvergenten Folge nimmt also die Anzahl der „richtigen“ Stellen quadratisch zu.

Beispiel: Das Newton-Verfahren für die Gleichung

$$F(x) = e^{\frac{x}{2}} + x - 2 = 0$$

lautet

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

Für den Startwert $x^{(0)} = 1$ erhält man

$$\begin{aligned} x^{(0)} &= 1., \\ x^{(1)} &= \underline{0.6}44, \\ x^{(2)} &= \underline{0.629\ 8}67, \\ x^{(3)} &= \underline{0.629\ 846\ 115\ 7}38, \\ x^{(4)} &= \underline{0.629\ 846\ 115\ 690\ 812}, \end{aligned}$$

also eine sehr schnell gegen die exakte Lösung $x^* = 0.629\ 846\ 115\ 690\ 812\ \ldots$ konvergente Folge von Näherungen.

Beispiel: Die Berechnung der positiven Wurzel aus einer positiven Zahl a, also die Lösung der nichtlinearen Gleichung

$$x^2 - a = 0$$

ist ein weiteres einfaches Beispiel für das Newton-Verfahren:

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

Jede positive (Maschinen-)Zahl a kann man in der Form $a = m \cdot 2^e$ mit einer geraden Zahl e und $m \in [0.5, 2)$ darstellen. Dann gilt: $\sqrt{a} = \sqrt{m} \cdot 2^{e/2}$. Verwendet man zur Berechnung von $\sqrt{m}$ das Newton-Verfahren mit Startwert $m^{(0)} = 1$ für $m \in [0.5, 1]$ und Startwert $m^{(0)} = 1.5$ für $m \in [1, 2)$, so benötigt man nur 5 Iterationen, um den relativen Fehler unterhalb der Maschinengenauigkeit $\text{eps} = 2^{-53}$ zu halten. Die Wurzelfunktion ist (etwa) auf diese Weise am Computer implementiert.

5.2 Varianten des Newton-Verfahrens

Der Satz 5.2 liefert nur die lokale Konvergenz, d.h., der Startwert muss hinreichend nahe bei einer einfachen Nullstelle sein.

Beispiel: Das Newton-Verfahren zur Lösung der Gleichung

$$\arctan x = 0$$

konvergiert nur für Startwerte $x^{(0)}$ mit $|x^{(0)}| < x_{krit}$ mit $x_{krit} \approx 1.4$.

Zusammenfassend lässt sich also sagen: Der große Vorteil des Newton-Verfahrens ist die schnelle Konvergenz. Diesem Vorteil stehen als Nachteile die nur lokale Konvergenz und der hohe Aufwand zur Durchführung des Newton-Verfahrens gegenüber.

In den nächsten beiden Abschnitten werden Modifikationen des Newton-Verfahrens diskutiert, die die oben genannten Nachteile abschwächen ohne dabei den Vorteil der schnellen Konvergenz vollständig zu verlieren.

Zunächst wird kurz eine Strategie vorgestellt, die im Allgemeinen die Menge der Startwerte, für die das Iterationsverfahren konvergiert, vergrößert.

5.2.1 Das gedämpfte Newton-Verfahren

Das Newton-Verfahren lässt sich auch als ein Liniensuchverfahren interpretieren. Ausgehend von der aktuellen Näherung $x^{(k)}$ berechnet man eine so genannte Suchrichtung $p^{(k)}$ als Lösung der Gleichung

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

und wählt als nächste Näherung jenen Punkt, den man von $x^{(k)}$ aus in Richtung $p^{(k)}$ mit Schrittweite 1 erreicht. Beim so genannten gedämpften Newton-Verfahren verallgemeinert man nun dieses Vorgehen, indem man für den nächsten Iterationspunkt auch Schrittweiten $\alpha^{(k)} \in (0, 1]$ zulässt:

$$x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}.$$

Dabei wird der Parameter $\alpha^{(k)}$ so gewählt, dass zumindest (vgl. KV Optimierung)

$$\|F(x^{(k+1)})\|_2^2 < \|F(x^{(k)})\|_2^2. \tag{5.4}$$

Man erhofft sich dadurch, dass der nächste Iterationspunkt „näher“ bei der Nullstelle liegt, da für die Nullstelle x^* natürlich gilt: $\|F(x^*)\|_2^2 = 0$.

Eine praktische Durchführung wäre z.B. das sukzessive Durchprobieren der Werte

$$1,\ \frac{1}{2},\ \frac{1}{4},\ \frac{1}{8}, \ldots, \qquad \text{Bisektion}$$

für $\alpha^{(k)}$, bis das Kriterium (5.4) schließlich erfüllt ist.

Der folgende Satz zeigt die grundsätzliche Durchführbarkeit dieser Strategie:

Satz 5.3. *Sei $F\colon \mathbb{R}^n \to \mathbb{R}^n$ differenzierbar. Für $x^{(k)} \in \mathbb{R}^n$ gelte: $F(x^{(k)}) \neq 0$ und $F'(x^{(k)})$ sei regulär. Dann gilt für $p^{(k)} = -F'(x^{(k)})^{-1}F(x^{(k)})$: Es gibt eine Zahl $\alpha_0 > 0$ mit*

$$\|F(x^{(k)} + \alpha p^{(k)})\|^2 < \|F(x^{(k)})\|^2$$

für alle $\alpha \in (0, \alpha_0)$.

Beweis. Durch Taylor-Entwicklung erhält man für $f(x) = \|F(x)\|^2$:

$$f(x^{(k)} + \alpha p^{(k)}) \approx f(x^{(k)}) + \alpha f'(x^{(k)})p^{(k)}.$$

Die Ableitung von $f(x) = \|F(x)\|_2^2 = F(x)^T F(x)$ ist durch

$$f'(x) = 2F(x)^T F'(x)$$

gegeben. Somit folgt:

$$f'(x^{(k)})p^{(k)} = -2F(x^{(k)})^T F'(x^{(k)})F'(x^{(k)})^{-1}F(x^{(k)}) = -2F(x^{(k)})^T F(x^{(k)}) = -2f(x^{(k)}) < 0.$$

Also gilt

$$f(x^{(k)} + \alpha p^{(k)}) \approx (1 - 2\alpha)f(x^{(k)}) < f(x^{(k)})$$

für hinreichend kleine Werte von α. □

Dieser Satz zeigt also, dass durch sukzessive Verkleinerung von α (z.B. durch Halbierung) das Kriterium (5.4) immer erfüllt werden kann.

Bemerkung: Das Kriterium (5.4) ist etwas zu schwach, um unter geeigneten Voraussetzungen die Konvergenz des gedämpften Newton-Verfahrens nachweisen zu können. Man verwendet stattdessen das (durch die Taylor-Entwicklung im Beweis des Satzes 5.3 motivierte) Kriterium

$$f(x^{(k+1)}) < (1 - 2\mu\alpha)f(x^{(k)})$$

für die Wahl der nächsten Schrittweite $\alpha^{(k)}$, wobei der Parameter $\mu \in (0, 1)$ vorzugeben ist, z.B. $\mu = 0.1$.

Bemerkung: Oft enthält das zu lösende Gleichungssystem einen Parameter t, also

$$F(x, t) = 0. \tag{5.5}$$

Angenommen, das Gleichungssystem soll für die Setzung $t = t^*$ gelöst werden. Weiters wird angenommen, für eine andere Setzung $t = t^{(0)} < t^*$ sei das Gleichungssystem leicht lösbar. Die Lösung wird mit $x^{(0)}$ bezeichnet.

Man erhöht nun t geringfügig: $t = t^{(1)} > t^{(0)}$ und kann hoffen, dass das Newton-Verfahren, angewendet auf die Gleichung (5.5) für $t = t^{(1)}$, mit Startwert $x^{(0)}$ gegen einen Wert $x^{(1)}$ konvergiert, da sich das Problem nur geringfügig geändert hat.

Man erhöht nun t erneut: $t = t^{(2)} > t^{(1)}$, verwendet wieder das Newton-Verfahren für (5.5) mit $t = t^{(2)}$ und Startwert $x^{(1)}$.

Man setzt diese Prozedur solange fort, bis man schließlich nach insgesamt N Schritten bei $t = t^{(N)} = t^*$, also bei der Gleichung

$$F(x) = 0$$

ankommt und einen guten Startwert $x^{(N-1)}$ für das abschließende Newton-Verfahren kennt.

Bietet sich bei einem Problem kein Parameter an, so kann man den Parameter t künstlich durch die Setzung

$$F(x,t) = F(x) - (1-t)F(x^{(0)})$$

einführen. Für $t = t^{(0)} = 0$ besitzt die $F(x,t) = 0$, also

$$F(x) = (1-t)\,F(x^{(0)})$$

natürlich die Lösung $x = x^{(0)}$. Mit $t = t^* = 1$ erhält man die ursprüngliche Gleichung (Homotopiemethoden, Einbettungsverfahren).

5.2.2 Inexakte Newton-Verfahren

Für den Fall, dass zwar die Berechnung der Jacobi-Matrix $F'(x^{(k)})$ wenig Rechenzeit kostet, dafür aber die Lösung des linearen Gleichungssystems

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

zu aufwendig ist, bieten sich iterative Verfahren zur näherungsweisen Lösung des linearen Gleichungssystems an.

Wichtig dabei ist die richtige Wahl eines Abbruchkriteriums für diese (innere) Iteration. Wird zu früh abgebrochen, kostet zwar eine inexakte Newton-Iteration wenig, es ist jedoch zu befürchten, dass durch die relativ große Abweichung von der exakten Newton-Iteration die schnelle Konvergenz des Verfahrens verloren geht. Andererseits kostet eine zu genaue Berechnung der nächsten Näherung zuviel Rechenzeit.

Für den Fall, dass die Berechnung der Jacobi-Matrix $F'(x^{(k)})$ explizit nicht möglich oder zu aufwendig ist, bieten sich Differenzenquotienten oder anderes zur Approximation der Jacobi–Matrix an.

Eine besonders effiziente Strategie, die der Einfachheit halber zunächst nur für den Fall $n = 1$ diskutiert wird, führt auf das so genannte Sekantenverfahren:

Man ersetzt im Newton-Verfahren die Ableitung $F'(x^{(k+1)})$ durch die Approximation

$$A^{(k+1)} = \frac{F(x^{(k+1)}) - F(x^{(k)})}{x^{(k+1)} - x^{(k)}}.$$

Das bedeutet geometrisch, dass man statt der Tangente im Punkt $x^{(k+1)}$ die Sekante in den Punkten $x^{(k+1)}$ und $x^{(k)}$ zur Linearisierung verwendet. Der Schnittpunkt dieser Sekante mit der x-Achse bestimmt die nächste Näherung.

Als Iterationsvorschrift erhält man

$$\begin{aligned} x^{(k+2)} &= x^{(k+1)} - \frac{F(x^{(k+1)})}{A^{(k+1)}} = x^{(k+1)} - \frac{F(x^{(k+1)})}{\frac{F(x^{(k+1)}) - F(x^{(k)})}{x^{(k+1)} - x^{(k)}}} \\ &= \frac{x^{(k)} F(x^{(k+1)}) - x^{(k+1)} F(x^{(k)})}{F(x^{(k+1)}) - F(x^{(k)})}. \end{aligned}$$

Eine Verallgemeinerung dieses Verfahrens für den allgemeinen Fall $n \geq 1$ ist das so genannte Broyden-Rang-1-Verfahren, bei dem mit folgender Approximation der Jacobi-Matrix gerechnet wird:

$$A^{(k+1)} = A^{(k)} + u^{(k)} {v^{(k)}}^T$$

mit

$$u^{(k)} = \frac{1}{{s^{(k)}}^T s^{(k)}} \left(y^{(k)} - A^{(k)} s^{(k)} \right), \quad v^{(k)} = s^{(k)}$$

und

$$s^{(k)} = x^{(k+1)} - x^{(k)}, \quad y^{(k)} = F(x^{(k+1)}) - F(x^{(k)}).$$

Ein Iterationsschritt des Broyden-Rang-1-Verfahrens besteht also aus folgenden Teilschritten:

1. Löse die Gleichung $A^{(k)} p^{(k)} = -F(x^{(k)})$.
2. Berechne $x^{(k+1)} = x^{(k)} + p^{(k)}$.
3. Berechne $A^{(k+1)} = A^{(k)} + u^{(k)} {v^{(k)}}^T$.

Beim Broyden-Rang-1-Verfahren benötigt man zur Berechnung einer Approximation der Jacobi-Matrix keine zusätzlichen Funktionsauswertungen. $A^{(k+1)}$ erhält man aus $A^{(k)}$ und den Funktionswerten $F(x^{k+1})$ und $F(x^{(k)})$, die man für die Berechnung von $x^{(k+1)}$ und $x^{(k)}$ ohnehin benötigt, durch einige wenige zusätzliche arithmetische Operationen pro Matrixelement.

Auch die Lösung des linearen Gleichungssystems, das in jedem Iterationsschritt anfällt, lässt sich schneller als beim gewöhnlichen Newton-Verfahren durchführen. So erhält man z.B. eine LU–Zerlegung von $A^{(k+1)}$ auf Grund der speziellen Struktur aus der LU–Zerlegung von $A^{(k)}$ in $O(n^2)$ Operationen. Analoges gilt auch für andere Zerlegungen aus Kapitel 3. Im Gegensatz dazu würde der Aufwand zur Lösung des linearen Gleichungssystems im Allgemeinen proportional zu n^3 steigen.

Das Broyden-Rang-1-Verfahren konvergiert trotz der Abweichungen vom Newton-Verfahren immer noch lokal und q–superlinear, d.h.

$$\lim_{k \to \infty} \frac{\|e^{(k+1)}\|}{\|e^{(k)}\|} = 0.$$

Das ist umso überraschender, als die Folge der Approximationen $A^{(k)}$ im Allgemeinen nicht gegen die exakte Jacobi-Matrix $F'(x^*)$ konvergiert.

Bemerkung: Bei der praktischen Durchführung des Broyden-Rang-1-Verfahrens wird nach allen K Schritten ein so genannter „restart“ mit der exakten Jacobi-Matrix durchgeführt. Dadurch verhindert man zu schlechte Approximationen $A^{(k)}$ der Jacobi-Matrix.

Kapitel 6

Eigenwertprobleme

Wir betrachten folgende Problemstellung: Gegeben ist eine Matrix $A \in \mathbb{C}^{n\times n}$. Gesucht sind Zahlen $\lambda \in \mathbb{C}$ und Vektoren $x \in \mathbb{C}^n$ mit $x \neq 0$ und

$$Ax = \lambda x.$$

λ heißt ein Eigenwert von A. x nennt man einen (zum Eigenwert λ gehörigen) Eigenvektor.

Probleme dieser Art treten z.B. bei der Diskussion von Systemen linearer gewöhnlicher Differentialgleichungen mit konstanten Koeffizienten

$$\dot{x} = Ax.$$

auf. Die Menge der Lösungen solcher Systeme lässt sich mit Hilfe der Eigenwerte und Eigenvektoren der Matrix A angeben. Stabilitätsuntersuchungen solcher Systeme lassen sich über die Eigenwerte von A führen. So ist z.B. die Ruhelage $x_s = 0$ des obigen Systems genau dann global asymptotisch stabil, wenn alle Eigenwerte von A negative Realteile besitzen.

Die Analyse von Schwingungsvorgängen führt auf Eigenwertprobleme. So erhält man z.B. zur Bestimmung der für ein Bauelement kritischen Eigenfrequenzen (nach einer Ortsdiskretisierung bei unendlich vielen Freiheitsgraden) ein Eigenwertproblem der obigen Art.

6.1 Theoretische Grundlagen

Einen ersten Ansatz zur Berechnung der Eigenwerte einer Matrix liefert das so genannte charakteristische Polynom:

Definition 6.1. *Sei $A \in \mathbb{C}^{n\times n}$. Das Polynom $p(\lambda)$, gegeben durch*

$$p(\lambda) = \det(A - \lambda I),$$

heißt das charakteristische Polynom von A.

$p(\lambda)$ ist ein Polynom vom Grade n. Es gilt:

Satz 6.1. *$\lambda \in \mathbb{C}$ ist genau dann ein Eigenwert von A, wenn $p(\lambda) = 0$.*

Die Nullstellen (oder Wurzeln) des charakteristischen Polynoms sind also genau die Eigenwerte von A. Dieser Satz kann daher als Grundlage einer numerischen Berechnung der Eigenwerte über die Nullstellen der Gleichung $p(\lambda) = 0$ dienen.

Definition 6.2. *Zwei Matrizen A und B heißen ähnlich, wenn es eine reguläre Matrix X gibt, sodass*

$$A = XBX^{-1}.$$

Für ähnliche Matrizen gilt folgende wichtige Aussage:

Satz 6.2. *Ähnliche Matrizen haben die gleichen Eigenwerte.*

Es bietet sich daher folgende weitere Strategie zur Berechnung von Eigenwerten an: Man konstruiere (zumindest näherungsweise) eine zu A ähnliche Matrix, deren Eigenwerte leicht zu bestimmen sind. So sind z.B. die Diagonalelemente einer Dreiecksmatrix ihre Eigenwerte.

Dass diese Strategie zumindest theoretisch erfolgversprechend ist, zeigt folgende Aussage:

Satz 6.3 (Schur)**.** *Zu jeder Matrix $A \in \mathbb{C}^{n\times n}$ gibt es eine unitäre Matrix U mit*

$$A = UTU^H$$

und

$$T = \begin{pmatrix} \lambda_1 & * & \cdots & * \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ 0 & \cdots & 0 & \lambda_n \end{pmatrix}.$$

T heißt Schur-Normalform, $\lambda_1, \lambda_2, \ldots, \lambda_n$ bezeichnen die Eigenwerte von A.

Der folgende Satz liefert eine Aussage über die ungefähre Lage der Eigenwerte:

Satz 6.4 (Gerschgorin)**.** *Die Vereinigung der Kreisscheiben*

$$K_i = \{\mu \in \mathbb{C} : |\mu - a_{ii}| \leq \sum_{j\neq i} |a_{ij}|\}$$

enthält alle Eigenwerte der Matrix $A = (a_{ij}) \in \mathbb{C}^{n\times n}$.

Beweis. Sei λ ein Eigenwert von A und kein Eigenwert von D. Da $A - \lambda I = (A - D) + (D - \lambda I)$ singulär ist, ist auch $I + (D - \lambda I)^{-1}(A - D)$ singulär. Dann folgt:

$$1 \leq \|(D - \lambda I)^{-1}(A - D)\|_\infty = \max_i \frac{\sum_{j\neq i} |a_{ij}|}{|\lambda - a_{ii}|},$$

also $|\lambda - a_{ii}| \leq \sum_{j\neq i} |a_{ij}|$ für ein $i \in \{1, 2, \ldots, n\}$. □

Der folgende Satz beschreibt die Auswirkung einer Störung der Matrix A auf die Eigenwerte:

Satz 6.5. *Sei $A \in \mathbb{C}^{n\times n}$ eine diagonalisierbare Matrix, d.h., es gibt eine reguläre Matrix X mit*

$$A = XDX^{-1},$$

wobei D eine Diagonalmatrix ist. D ist dann von der Form

$$D = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{pmatrix},$$

wobei $\lambda_1, \lambda_2, \ldots, \lambda_n$ die Eigenwerte von A bezeichnen. Weiters sei $\Delta A \in \mathbb{C}^{n\times n}$. Dann gibt es zu jedem Eigenwert $\bar{\lambda}$ von $\bar{A} = A + \Delta A$ einen Eigenwert λ von A mit

$$|\bar{\lambda} - \lambda| \leq \kappa_2(X) \cdot \|\Delta A\|_2.$$

Beweis. Sei $\bar{\lambda}$ ein Eigenwert von $A + \Delta A$ und nicht Eigenwert von A. Da $X^{-1}(A + \Delta A - \bar{\lambda}\, I)X = (D - \lambda\, I) + X^{-1}\Delta AX$ singulär ist, ist auch $I + (D - \lambda\, I)^{-1}X^{-1}\Delta AX$ singulär. Dann folgt:

$$\begin{aligned} 1 &\leq \|(D - \bar{\lambda}\, I)^{-1}X^{-1}\Delta AX\|_2 \leq \|(D - \bar{\lambda}\, I)^{-1}\|_2 \|X^{-1}\|_2 \|\Delta A\|_2 \|X\|_2 \\ &= \kappa(A)\, \|\Delta A\|_2 \max_{\lambda\in\sigma(A)} \frac{1}{|\bar{\lambda} - \lambda|}, \end{aligned}$$

also $|\bar{\lambda} - \lambda| \leq \kappa(A)\, \|\Delta A\|_2$ für ein $\lambda \in \sigma(A)$. □

Fehler in den Daten werden also mit der Konditionszahl von X (nicht von A) verstärkt. Ein besonders günstiger Fall liegt vor, wenn X eine unitäre Matrix ist, denn dann gilt $\kappa(X) = 1$ und $|\lambda_{max}(A)| = \rho(A) = \|D\|_2 = \|A\|_2$. Somit folgt

$$\varepsilon_\lambda \leq \varepsilon_A$$

mit $\varepsilon_\lambda = |\bar{\lambda} - \lambda|/|\lambda_{max}(A)|$ und $\varepsilon_A = \|\Delta A\|_2/\|A\|_2$. Das Eigenwertproblem ist in diesem Fall also gut konditioniert.

Jene Matrizen A, für die X unitär gewählt werden kann, die also durch eine Ähnlichkeitstransformation mit einer unitären Matrix diagonalisiert werden können, sind die normalen Matrizen, d.h.:

$$A^H A = AA^H.$$

Eine wichtige Teilklasse bilden die symmetrischen reellen Matrizen. Das Eigenwertproblem ist also für solche Matrizen gut konditioniert.

6.2 Der QR-Algorithmus

Das Ziel des QR-Algorithmus ist die Konstruktion einer Folge von Matrizen $A^{(k)}$, die alle zu A ähnlich sind und in gewisser Weise gegen eine Schur-Normalform von A konvergieren.

Die Basisversion des Algorithmus besteht aus folgender Iteration:

QR-Algorithmus:

$A^{(1)} = A$

Für $k = 1, 2, \ldots$ berechne:

1. $Q_k R_k = A^{(k)}$ (QR-Zerlegung)
2. $A^{(k+1)} = R_k Q_k$

Der erste Teilschritt besteht aus einer QR-Zerlegung von $A^{(k)}$ z.B. mit dem Householder-Verfahren, mit dem modifizierten Gram-Schmidt-Verfahren oder mittels Givens-Rotationen. Im zweiten Teilschritt werden die Faktoren der QR-Zerlegung in umgekehrter Reihenfolge miteinander multipliziert, um die nächste Iterierte $A^{(k+1)}$ zu erhalten.

Auf diese Weise entsteht eine Folge ähnlicher Matrizen, denn

$$A^{(k+1)} = R_k Q_k = Q_k^{-1} A^{(k)} Q_k.$$

Der nächste Satz zeigt, dass unter gewissen Voraussetzungen diese Folge für $k \to \infty$ der Form nach gegen eine obere Dreiecksmatrix, also gegen eine Schur-Normalform, konvergiert.

Satz 6.6. *Sei $A \in \mathbb{C}^{n \times n}$ eine Matrix mit folgenden Eigenschaften:*

1. *Die Eigenwerte von A sind betragsmäßig verschieden:*

$$|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n| > 0.$$

2. *$Y = X^{-1}$ besitzt eine Dreieckszerlegung: $Y = L_Y \cdot R_Y$, wobei X eine Matrix ist, für die gilt: $A = XDX^{-1}$ mit*

$$D = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \lambda_n \end{pmatrix}.$$

Dann gilt für die Folge der Matrizen $A^{(k)} = (a_{ij}^{(k)})$, die durch den QR-Algorithmus erzeugt werden:

1. $\lim_{k \to \infty} a_{ij}^{(k)} = 0$ *für alle i, j mit $i > j$ bzw. genauer* $a_{ij}^{(k)} = O(|\frac{\lambda_i}{\lambda_j}|^k)$.

2. $\lim_{k\to\infty} a_{ii}^{(k)} = \lambda_i$ *für* $i = 1, 2, \ldots, n$.

Bemerkung: Die Konvergenz des QR-Algorithmus folgt auch ohne die Voraussetzung (2), allerdings sind dann die Eigenwerte als Grenzwerte der Diagonalelemente von $A^{(k)}$ nicht mehr der Größe nach angeordnet.

6.2.1 Reduktion auf einfache Gestalt

Der QR-Algorithmus ist für allgemeine Matrizen A viel zu aufwendig. Der Aufwand für einen Iterationsschritt steigt im Allgemeinen proportional zu n^3. Daher führt man für A zuerst eine Reduktion auf einfache Gestalt durch:

Definition 6.3. *Eine Matrix* $A = (a_{ij}) \in \mathbb{C}^{n\times n}$ *heißt* ***(rechte obere) Hessenberg-Matrix****, genau dann wenn* $a_{ij} = 0$ *für* $i > j + 1$.

Eine Hessenberg-Matrix A ist also von folgender Form:

$$A = \begin{pmatrix} * & \cdots & \cdots & \cdots & * \\ * & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & * \\ 0 & \cdots & 0 & * & * \end{pmatrix}$$

Es gilt nun:

Satz 6.7. *Zu jeder Matrix* $A \in \mathbb{C}^{n\times n}$ *existiert eine unitäre Matrix* U *mit* $U^H AU = H$, *wobei* H *eine Hessenberg-Matrix ist.*

Beweis. Konstruktiver Beweis:

Angenommen, A sei von der Form

$$A = \left(\begin{array}{c|c} * & * \\ \hline a_1 & * \\ & \end{array}\right)$$

mit $a_1 \in \mathbb{C}^{n-1}$.

Man wähle im 1. Schritt eine unitäre Matrix $P_1 \in \mathbb{C}^{(n-1)\times(n-1)}$ mit

$$P_1 a_1 = k e_1,$$

siehe z.B. das Householder-Verfahren. Für die unitäre Matrix

$$U_1 = \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & P_1^H & \\ 0 & & & \end{array}\right)$$

gilt dann:

$$
\begin{aligned}
U_1^H A U_1 &= \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & P_1 & \\ 0 & & & \end{array}\right) \cdot \left(\begin{array}{c|c} * & * \\ \hline a_1 & * \end{array}\right) \cdot \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & P_1^H & \\ 0 & & & \end{array}\right) \\
&= \left(\begin{array}{c|ccc} * & * & \cdots & * \\ \hline ke_1 & & * & \end{array}\right) \cdot \left(\begin{array}{c|ccc} 1 & 0 & \cdots & 0 \\ \hline 0 & & & \\ \vdots & & P_1^H & \\ 0 & & & \end{array}\right) \\
&= \left(\begin{array}{c|ccc} * & * & \cdots & * \\ \hline ke_1 & & \tilde{A}_1 & \end{array}\right)
\end{aligned}
$$

Die erste Spalte der transformierten Matrix hat also bereits die richtige Struktur.

Man setzt das Verfahren in analoger Weise für die Restmatrix $\tilde{A}_1$ fort, bis man nach insgesamt $n-2$ Schritten eine Hessenberg-Matrix H erhält mit

$$U_{n-2}^H \cdots U_2^H U_1^H A U_1 U_2 \cdots U_{n-2} = H.$$

Mit $U = U_1 U_2 \cdots U_{n-2}$ folgt dann die Behauptung. □

Für den Spezialfall einer symmetrischen Matrix A folgt daraus sofort, dass auch die zu A ähnliche Hessenberg-Matrix symmetrisch ist, also eine Tridiagonalmatrix ist. Es gilt daher:

Satz 6.8. *Zu jeder symmetrischen Matrix $A \in \mathbb{R}^{n\times n}$ existiert eine orthogonale Matrix Q mit $Q^T A Q = T$, wobei T eine Tridiagonalmatrix ist.*

Es lässt sich leicht nachweisen, dass die QR-Zerlegung so durchgeführt werden kann, dass auch alle weiteren Matrizen $A^{(k)}$ des QR-Algorithmus Hessenberg-Matrizen bzw. Tridiagonalmatrizen bleiben, wobei der Aufwand für einen Schritt des QR-Algorithmus nur noch proportional zu n^2, im symmetrischen Fall nur noch proportional zu n, steigt.

Bemerkung: Für große dünnbesetzte Matrizen ist die beschriebene Methode zur Reduktion auf einfache Gestalt zu aufwendig und benötigt zuviel Speicherplatz. Es gibt jedoch ein Verfahren, das so genannte Arnoldi-Verfahren, das es erlaubt, die Elemente einer zu A ähnlichen Hessenberg-Matrix auch ohne Zwischenergebnisse direkt zu berechnen. Der analoge Algorithmus für symmetrische Matrizen heißt Lanczos-Verfahren.

Eine vollständige Reduktion einer großen dünnbesetzten Matrix auf Hessenberg-Gestalt mit dem Arnoldi-Verfahren kostet zwar wenig Speicherplatz aber immer noch zu viel Rechenzeit. Man begnügt sich daher mit der Erzeugung von Teilmatrizen der vollständigen

zu A ähnlichen Hessenberg-Matrix und wendet darauf z.B. einen speziell adaptierten QR-Algorithmus an.

Im symmetrischen Fall wird eine entsprechende Strategie auf der Basis des Lanczos-Verfahrens durchgeführt.

Bemerkung:

1. Der QR–Algorithmus liefert eine genährte Schur–Normalform T, die sich leicht diagonalisieren lässt: Es gibt eine Dreiecksmatrix Y der Form

$$Y = \begin{pmatrix} 1 & * & \cdots & * \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ 0 & \cdots & 0 & 1 \end{pmatrix}$$

   mit $Y^{-1}TY = D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$: Die Gleichung $TY = YD$ lässt sich leicht spaltenweise auflösen. Die Matrix $X = QY$ enthält dann spaltenweise die Eigenvektoren.

2. Die folgende Variante des QR-Algorithmus besteht aus der Anwendung des ursprünglichen QR-Algorithmus auf eine Matrix der Form $A - \mu_k I$ und anschließender Rücktransformation durch Addition mit $\mu_k I$:

   QR-Algorithmus mit einfacher Shift-Strategie:

   $A^{(1)} = A$

   Für $k = 1, 2, \ldots$ berechne:

   1. $Q_k R_k = A^{(k)} - \mu_k I$ (QR-Zerlegung)
   2. $A^{(k+1)} = R_k Q_k + \mu_k I$

   Durch eine geeignete Wahl für μ_k lässt sich erreichen, dass die Matrizen $A^{(k)}$ auch für betragsgleiche Eigenwerte der Form nach gegen eine Schur-Normalform konvergieren und dass die Konvergenz beschleunigt wird. Konkret wird z.B. vorgeschlagen, den Shift-Parameter nach der Formel

$$\mu_k = a_{nn}^{(k)}$$

   zu wählen. Das führt auf eine besonders schnelle Konvergenz von $a_{n,n-1}^{(k)}$ gegen 0. Anschließend kann das Eigenwertproblem um eine Dimension verkleinert werden. Man setzt dann die gleiche Strategie für die reduzierte Matrix fort, bis man sie wieder um eine Dimension reduzieren kann, u.s.w.

   Eigenwerte für reelle Matrizen sind entweder reell oder sie treten in Paaren von konjugiert komplexen Zahlen auf. Im zweiten Fall können die Eigenwerte nicht mehr betragsmäßig getrennt sein. Eine einfache Shift-Strategie mit reellen Shift-Parametern

würde zu keiner Trennung der Eigenwerte führen. Es lässt sich allerdings eine so genannte Doppel-Shift-Strategie konstruieren, die es erlaubt, die Konvergenz zu beschleunigen, und die nur reelle Arithmetik verwendet.

6.3 Berechnung einzelner Eigenwerte und Eigenvektoren

Einzelne Eigenwerte lassen sich als Nullstellen des charakteristischen Polynoms berechnen, z.B. mit Hilfe des Newton–Verfahrens. Allerdings ist zu beachten, dass die Aufstellung des charakteristischen Polynoms und seiner Ableitung über die Berechnung der Koeffizienten des Polynoms und die anschließende Berechnung der Nullstellen im Allgemeinen kein numerisch stabiler Algorithmus ist. Besser geeignet sind spezielle Rekursionsformeln zur Berechnung von $p(\lambda)$ bzw. $p'(\lambda)$ für einen gegebenen Wert von λ.

So lässt sich das charakteristische Polynom an einer Stelle λ für symmetrische reelle Tridiagonalmatrizen

$$A = \begin{pmatrix} b_1 & a_2 & & \\ a_2 & b_2 & \ddots & \\ & \ddots & \ddots & a_n \\ & & a_n & b_n \end{pmatrix}$$

durch folgende Rekursion berechnen: $p(\lambda) = p_n(\lambda)$ mit

$$\begin{aligned} p_0(\lambda) &= 1 \\ p_1(\lambda) &= b_1 - \lambda \\ p_k(\lambda) &= (b_k - \lambda)\, p_{k-1}(\lambda) - a_k^2\, p_{k-2}(\lambda), \quad k = 2, \ldots, n. \end{aligned}$$

Durch Ableiten der Rekursion für $p(\lambda)$ erhält man eine entsprechende Rekursion für $p'(\lambda)$.

Geeignete Startwerte für das Newton–Verfahren erhält man aus dem Satz von Gerschgorin.

Inverse Vektoriteration (Wielandt)

Sei $\bar{\lambda}$ eine Näherung eines Eigenwertes λ_i der Matrix A. Ausgehend von einem Startvektor $q^{(0)}$ berechnet man eine Folge von Vektoren mit

$$(A - \bar{\lambda}\, I) z^{(k+1)} = q^{(k)}, \quad q^{(k+1)} = z^{(k+1)} / \|z^{(k+1)}\|_2$$

Sei A diagonalisierbar und sei $\{x_1, x_2, \ldots, x_n\}$ eine Basis von Eigenvektoren mit $Ax_j = \lambda_j\, x_j$. Für einen Startwert

$$q^{(0)} = \sum_j \alpha_j\, x_j$$

gilt offensichtlich, dass $q^{(k)}$ parallel zu

$$\begin{aligned}(A-\bar{\lambda}\,I)^{-k}q^{(0)} &= \sum_j \frac{\alpha_j}{(\lambda_j-\bar{\lambda})^k}\,x_j \\ &= \frac{1}{(\lambda_i-\bar{\lambda})^k}\left[\alpha_i\,x_i+\sum_{j\neq i}\alpha_j\left(\frac{\lambda_i-\bar{\lambda}}{\lambda_j-\bar{\lambda}}\right)^k x_j\right].\end{aligned}$$

liegt. Unter den Bedingungen $\alpha_i \neq 0$ und

$$0<|\lambda_i-\bar{\lambda}|<|\lambda_j-\bar{\lambda}| \quad \text{für alle } j\neq i$$

folgt die Konvergenz von $q^{(k)}$ gegen einen Eigenvektor zum Eigenwert λ_i.

Eine verbesserte Näherung für diesen Eigenwert erhält man durch den Rayleigh–Quotienten:

$$\lambda_i^{(k)}=\frac{(Aq^{(k)},q^{(k)})_2}{(q^{(k)},q^{(k)})_2}.$$

Bemerkung: Unter der direkten Vektoriteration (von Mises) versteht man die Iteration

$$z^{(k+1)}=Aq^{(k)}, \quad q^{(k+1)}=z^{(k+1)}/\|z^{(k+1)}\|_2.$$

Sei A diagonalisierbar und sei $\{x_1,x_2,\ldots,x_n\}$ eine Basis von Eigenvektoren mit $Ax_j=\lambda_j\,x_j$. Für einen Startwert

$$q^{(0)}=\sum_j \alpha_j\,x_j$$

gilt offensichtlich, dass $q^{(k)}$ parallel zu

$$\begin{aligned}A^k q^{(0)} &= \sum_j \alpha_j\lambda_j^k\,x_j \\ &= \lambda_n^k\left[\alpha_n\,x_n+\sum_{j<n}\alpha_j\left(\frac{\lambda_j}{\lambda_j}\right)^k x_j\right].\end{aligned}$$

liegt. Unter den Bedingungen $\alpha_n \neq 0$ folgt die Konvergenz gegen einen Eigenvektor zum betragsgrößten λ_n.

Kapitel 7

Interpolation, Numerische Differentiation und Integration

Bei der klassischen Interpolation versucht man eine Funktion aus ihren Werten in vorgegebenen Stützstellen zu konstruieren: Gesucht ist eine Funktion $\varphi(x)$ mit

$$\varphi(x_i) = f_i \quad \text{für } i = 0, 1, \ldots, n.$$

für gegebene paarweise verschiedene Stützpunkte x_i und Werte f_i.

Allgemeiner versteht man unter einem Interpolationsproblem folgende Aufgabenstellung: Gesucht ist eine Funktion $\varphi(x)$ mit

$$l_i(\varphi) = f_i \quad \text{für } i = 0, 1, \ldots, n.$$

für gegebene lineare Funktionale l_i und Werte f_i. Die klassische Interpolation entspricht der Setzung: $l_i(\varphi) = \varphi(x_i)$.
Voraussetzung: $x_i \neq x_j$ für $i \neq j,\ i, j = 0, \ldots, n$.

7.1 Interpolation mit Polynomen

Bei der Interpolation mit Polynomen wählt man den Ansatz

$$\varphi(x) = a_0 + a_1\, x + \ldots + a_n\, x^n$$

Es gilt folgende Existenz- und Eindeutigkeitsaussage:

Satz 7.1. *Es gibt genau ein Polynom $P(x)$ vom Grade $\leq n$, das das Interpolationsproblem löst.*

Beweis. Die Abbildung $P \mapsto (P(x_0), P(x_1), \ldots, P(x_n)$ ist eine lineare Abbildung vom $(n+1)$–dimensionalen Raum der Polynome vom Grad $\leq n$ in den $\mathbb{R}^{n+1}$. Die Abbildung ist injektiv, da aus $P(x_0) = P(x_1) = \ldots = P(x_n) = 0$ folgt, dass $P \equiv 0$. Eine injektive Abbildung zwischen $(n+1)$–dimensionalen Räumen ist auch surjektiv. □

Für theoretische Überlegungen ist die Darstellung mit Hilfe von Lagrange–Polynomen

$$P(x) = \sum_{i=0}^{n} f_i L_i(x) \quad \text{mit} \quad L_i(x) = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$$

und die Beschreibung der Polynomkoeffizienten als Lösung eines Vandermonde Systems

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}$$

von Bedeutung. Für die praktische Auswertung eines Interpolationspolynoms hingegen ist der Neville–Algorithmus geeignet: Sei $P_{i_0,i_1,\dots,i_k}(x)$ das Interpolationspolynom vom Grad $\leq k$ mit

$$P_{i_0,i_1,\dots,i_k}(x_{i_j}) = f_{i_j}, \quad j = 0, 1, \dots, k.$$

Dann gilt:

Satz 7.2.

$$\begin{aligned} P_i(x) &= f_i \\ P_{i_0,i_1,\dots,i_k}(x) &= \frac{(x - x_{i_0})\, P_{i_1,\dots,i_k}(x) - (x - x_{i_k})\, P_{i_0,i_1,\dots,i_{k-1}}(x)}{x_{i_k} - x_{i_0}} \end{aligned}$$

Beweis. $P_{i_0,i_1,\dots,i_k}(x_{i_0}) = P_{i_0,i_1,\dots,i_{k-1}}(x_{i_0}) = f_{i_0}$, $P_{i_0,i_1,\dots,i_k}(x_{i_k}) = P_{i_1,\dots,i_k}(x_{i_k}) = f_{i_k}$.

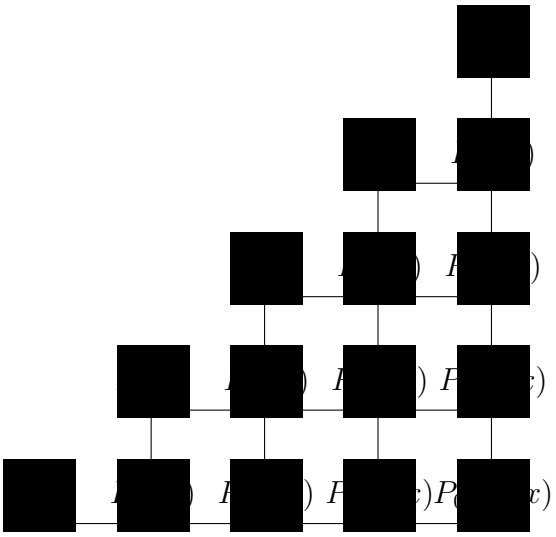
Für $j = 1, \dots, k-1$ gilt:

$$\begin{aligned} P_{i_0,i_1,\dots,i_k}(x_{i_j}) &= \frac{(x_{i_j} - x_{i_0})\, P_{i_1,\dots,i_k}(x_{i_j}) - (x_{i_j} - x_{i_k})\, P_{i_0,i_1,\dots,i_{k-1}}(x_{i_j})}{x_{i_k} - x_{i_0}} \\ &= \frac{(x_{i_j} - x_{i_0})\, f_{i_j} - (x_{i_j} - x_{i_k})\, f_{i_j}}{x_{i_k} - x_{i_0}} = f_{i_j} \end{aligned}$$

□

Dieser Satz legt folgendes Schema zur Auswertung eines Interpolationspolynoms $P(x) =$

$P_{0,1,\ldots,n}(x)$ nahe (Neville–Schema):



Das Neville–Schema erfordert $O(n^2)$ Operationen zur Auswertung des Interpolationspolynoms an einer Stelle x.

Zur expliziten Aufstellung des Interpolationspolynoms eignet sich die Newton–Darstellung:

$$P(x) = a_0 + a_1\,(x - x_0) + a_2\,(x - x_0)(x - x_1) + \ldots + a_n\,(x - x_0)(x - x_1) \cdot \ldots \cdot (x - x_{n-1})$$

Die Koeffizienten erfüllen das folgende gestaffeltes System:

$$\begin{array}{lcl} a_0 & = & f_0 \\ a_0 + a_1\,(x_1 - x_0) & = & f_1 \\ \vdots & & \\ a_0 + a_1\,(x_n - x_0) + \ldots + a_n\,(x_n - x_0) \cdot \ldots \cdot (x_n - x_{n-1}) & = & f_n \end{array}$$

Der folgende Satz erlaubt eine explizite Darstellung der Koeffizienten:

Satz 7.3.

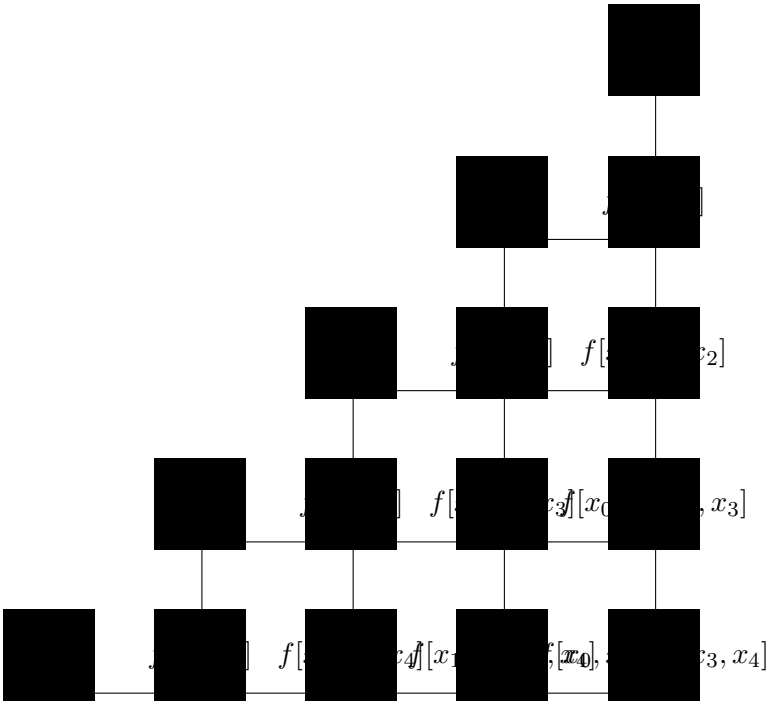
$$P_{i,\ldots,i+k}(x) = f[x_i] + f[x_i, x_{i+1}](x - x_i) + \ldots + f[x_i, \ldots, x_{i+k}](x - x_i) \cdot \ldots \cdot (x - x_{i+k-1})$$

mit den rekursiv definierten dividierten Differenzen

$$\begin{array}{rcl} f[x_i] & = & f_i \\ f[x_i, x_{i+1}, \ldots, x_{i+k}] & = & \dfrac{f[x_{i+1}, x_{i+1}, \ldots, x_{i+k}] - f[x_i, x_{i+1}, \ldots, x_{i+k-1}]}{x_{i+k} - x_i} \end{array}$$

Beweis. Folgt durch Induktion bezüglich k aus dem vorigen Satz. □

Nach diesem Satz erhält man die dividierten Differenzen aus folgendem Schema:



Daraus ergeben sich die Koeffizienten der Newton–Darstellung:

$$a_i = f[x_0, x_1, \ldots, x_i]$$

Die Auswertung des Interpolationspolynoms in Newton–Darstellung erfolgt durch das Horner–Schema:

$$P(x) = (\ldots((a_n\,(x - x_{n-1}) + a_{n-1})\,(x - x_{n-2}) + \ldots + a_1)\,(x - x_0) + a_0$$

Die Berechnung der Koeffizienten erfordert $O(n^2)$ Operationen, für das Horner–Schema benötigt man weitere $O(n)$ Operationen pro Auswertung.

Interpolationsfehler

Satz 7.4. *Sei $f \in C^{n+1}[a, b]$ und sei P_n das Interpolationspolynom vom Grad $\leq n$ bezüglich der paarweise verschiedenen Punkte $a \leq x_0 < x_1 < \ldots < x_n \leq b$ und den Werten $f(x_i)$.*

Dann gibt es zu jedem $\bar{x} \in [a,b]$ ein $\xi \in (a,b)$ mit

$$f(\bar{x}) - P_n(\bar{x}) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\,\omega(\bar{x})$$

und

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

Beweis. Sei $\bar{x} \neq x_i$ für alle $i = 0, 1, \ldots, n$. Die Funktion $F(x) = f(x) - P_n(x) - K\,\omega(x)$ mit

$$K = \frac{f(\bar{x}) - P_n(\bar{x})}{\omega(\bar{x})}$$

besitzt $n+2$ verschiedene Nullstellen. Nach dem Satz von Rolle besitzt $F^{(n+1)}(x)$ mindestens eine Nullstelle $\xi \in (a,b)$. Also

$$0 = F^{(n+1)}(\xi) = f^{(n+1)}(\xi) - K\,\omega^{(n+1)}(\xi) = f^{(n+1)}(\xi) - \frac{f(\bar{x}) - P_n(\bar{x})}{\omega(\bar{x})}\,(n+1)!$$

□

Dieser Satz besagt, dass der Interpolationsfehler wesentlich von der Größe der Funktion $\omega(x)$ abhängt. Es liegt nahe, die Stützpunkte so zu wählen, dass

$$\|\omega\|_\infty = \max_{x\in[a,b]} |\omega(x)| \to \min.$$

Es lässt sich zeigen, dass im Intervall $[a,b] = [-1,1]$ die in diesem Sinne günstigste Wahl der Stützpunkte durch die Nullstellen des $(n+1)$–ten Tschebyscheff–Polynoms T_{n+1} gegeben ist:

$$x_i = \cos\left(\frac{2i+1}{2n+2}\,\pi\right) \quad \text{für } i = 0, 1, \ldots, n.$$

Dann erhält man $\omega(x) = 2^{-n}\,T_{n+1}(x)$ mit $T_k(x) = \cos(k\,\arccos(x))$.

Diese Wahl der Stützpunkt führt häufig zu wesentlich kleineren Interpolationsfehler als die naheliegendere Wahl von äquidistanten Stützpunkten.

Der Satz von Faber zeigt allerdings, dass es für keine Stützpunktewahl eine Garantie auf Konvergenz gibt: Zu jeder Folge von Stützpunktsätzen $a \le x_0^{(k)} < x_1^{(k)} < \ldots < x_{n_k}^{(k)} \le b$ gibt es eine stetige Funktion f, so dass die Folge der Interpolationspolynome P_k nicht gleichmäßig gegen f konvergiert, ja sogar nicht einmal punktweise bei äquidistanter Stützpunktewahl.

7.2 Interpolation mit Splines

Polynome hohen Grades neigen zu Oszillationen , was letztlich die Ursache für die schlechte Konvergenz der Polynominterpolation ist. Interpolationsfunktionen, die stückweise aus Polynomen niedrigen Grades zusammengesetzt sind, weisen oft bessere Konvergenzeigenschaften auf. Besonders gute Approximationseigenschaften besitzen sogenannte Spline–Funktionen:

Definition 7.1. *Gegeben seien Gitterpunkte $a = x_0 < x_1 < \ldots < x_n = b$. Eine Spline–Funktion vom Grad $k-1$ (Ordnung k) ist eine Funktion $S \in C^{k-2}[a,b]$, die auf jedem Intervall $[x_i, x_{i+1}]$ mit einem Polynom vom Grad $\leq k-1$ übereinstimmt.*

Die wichtigsten Beispiele sind lineare Splines (Ordnung 2) und kubische Splines (Ordnung 4).

Die Dimension aller Splines der Ordnung k ist $n+k-1$.

Die Interpolationsaufgabe lautet nun: Gesucht ist eine Spline–Funktion $S(x)$ mit

$$S(x_i) = f_i, \quad i = 0, 1, \ldots, n.$$

Für lineare Splines reichen diese Bedingungen, um die Spline–Funktion eindeutig festzulegen.

Für kubische Splines fehlen noch zwei Bedingungen: Je nach Wahl der zwei fehlenden Bedingungen unterscheidet man vollständige, natürliche und periodische Spline–Funktionen:

1. Vollständige Splines: $S'(a) = f_0'$ und $S'(b) = f_n'$.

2. Natürliche Splines: $S''(a) = S''(b) = 0$.

3. Periodische Splines (falls $f_0 = f_n$): $S'(a) = S'(b)$ und $S''(a) = S''(b)$.

Konstruktion der kubischen Splines:

$S(x)$ stimmt im Teilintervall $[x_i, x_{i+1}]$ mit einem kubischen Polynom überein, das sich mit Hilfe der Funktionswerte $S(x_i) = f_i$ und $S(x_{i+1}) = f_{i+1}$ und der (noch unbekannten) zweiten Ableitungen $S''(x_i) = c_i$ und $S''(x_{i+1}) = c_{i+1}$ darstellen lässt:

$$S(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3 \quad \text{für } x \in [x_i, x_{i+1}]$$

mit $h_{i+1} = x_{i+1} - x_i$ und

$$a_i = f_i, \quad b_i = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{h_{i+1}}{6}(2c_i + c_{i+1}), \quad d_i = \frac{1}{h_{i+1}}(c_{i+1} - c_i).$$

Durch diese Darstellung ist die Stetigkeit von $S(x)$ und $S''(x)$ in den einzelnen Stützpunkten und somit im gesamten Intervall $[a, b]$ gesichert. Die noch notwendige Forderung der Stetigkeit von $S'(x)$ in jedem inneren Knoten x_i, $i = 1, \ldots, n-1$ führt auf die Bedingungen:

$$b_{i-1} + c_{i-1}h_i + \frac{d_{i-1}}{2}h_i^2 = b_i,$$

also

$$\frac{h_i}{6}\, c_{i-1} + \frac{h_i + h_{i+1}}{3}\, c_i + \frac{h_{i+1}}{6}\, c_{i+1} = \frac{f_{i+1} - f_i}{h_{i+1}} - \frac{f_i - f_{i-1}}{h_i} \quad \text{für } i = 1, \ldots, n-1.$$

Zwei weitere Bedingungen erhält man aus den Randbedingungen, z.B.:

$$c_0 = 0 \quad \text{und} \quad c_n = 0$$

für die natürlichen Splines. Man erhält also insgesamt ein tridiagonales Gleichungssystem, das sich effizient in $O(n)$ Operationen auflösen lässt.

Die kubischen Spline–Funktionen besitzen eine interessanten Extremaleigenschaft:

Satz 7.5. *Gegeben seien Gitterpunkte $a = x_0 < x_1 < \ldots < x_n = b$ und Werte $f_0, f_1, \ldots, f_n$. Sei $S(x)$ die interpolierende Spline–Funktion (d.h.: $S(x_i) = f_i, i = 0, 1, \ldots, n$) mit einer der drei oben genannten Randbedingungen und sei $\varphi(x) \in C^2[a, b]$ eine beliebige weitere interpolierende Funktion mit den gleichen Randbedingungen. Dann gilt:*

$$\int_a^b S''(x)^2 \, dx \leq \int_a^b \varphi''(x)^2 \, dx.$$

Das Integral $\int_a^b \varphi''(x)^2 \, dx$ lässt sich als Maß für die mittlere Krümmung von $\varphi(x)$ interpretieren. Der obige Satz besagt also, dass die Spline–Funktionen jene Interpolationsfunktionen sind, die die kleinste mittlere Krümmung besitzen.

Folgende Interpolationsfehlerabschätzung lässt sich zeigen:

Satz 7.6. *Gegeben seien Gitterpunkte $a = x_0 < x_1 < \ldots < x_n = b$, $h = \max_{1 \leq i \leq n} h_i$. Sei $f \in C^4[a, b]$ und $S(x)$ die interpolierende Spline–Funktion (d.h.: $S(x_i) = f(x_i), i = 0, 1, \ldots, n$) mit $S'(x_0) = f'(x_0)$ und $S'(x_n) = f'(x_n)$. Dann gibt es Konstanten c_ν mit*

$$\|f^{(\nu)} - S^{(\nu)}\|_\infty \leq c_\nu h^{4-\nu} \|f^{(4)}\|_\infty \quad \textit{für } \nu = 0, 1, 2.$$

Dieser Satz sichert die gleichmäßige Konvergenz der Spline–Funktion $S(x)$ (und der ersten und zweiten Ableitung) gegen $f(x)$ (und deren erster und zweiter Ableitung), wenn die Feinheit h der Unterteilung gegen 0 konvergiert.

7.3 Numerische Differentiation

Man kennt von einer Funktion $f(x)$ die Werte an Stützpunkten $a \leq x_0 < x_1 < \ldots < x_n \leq b$. Gesucht sind Näherungen von $f'(x)$.

Grundidee: $f(x)$ wird durch eine Interpolationsfunktion $\varphi(x)$ ersetzt. Man verwendet $\varphi'(x)$ als Näherungen für $f'(x)$.

Beispiel: Bei Verwendung von linearer Polynominterpolation in den Punkten x und $x+h$ erhält man den so genannten Vorwärtsdifferenzenquotienten:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

Bei Verwendung von linearer Polynominterpolation in den Punkten $x-h$ und x erhält man den so genannten Rückwärtsdifferenzenquotienten:

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}$$

Bei Verwendung von quadratischer Polynominterpolation in den Punkten $x-h$, x und $x+h$ erhält man den so genannten zentralen Differenzenquotienten:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

Aus dem entsprechenden Satz über den Interpolationsfehler bei Polynominterpolation erhält man leicht eine Aussage über den Fehler bei der Approximation der ersten Ableitung in den Stützpunkten:

Satz 7.7. *Sei $f \in C^{n+1}[a,b]$ und sei P_n das Interpolationspolynom vom Grad $\leq n$ bezüglich der paarweise verschiedenen Punkte $a \leq x_0 < x_1 < \ldots < x_n \leq b$ und den Werten $f(x_i)$. Dann gibt es zu jedem Stützpunkt x_i ein $\xi_i \in (a,b)$ mit*

$$f'(x_i) - P_n'(x_i) = \frac{f^{(n+1)}(\xi_i)}{(n+1)!} \prod_{j \neq i} (x_i - x_j)$$

Damit erhält man sofort folgende Fehlerabschätzungen für einfache Differenzenquotienten:

$$\begin{aligned}
\frac{f(x+h) - f(x)}{h} &= f'(x) + O(h) \\
\frac{f(x) - f(x-h)}{h} &= f'(x) + O(h) \\
\frac{f(x+h) - f(x-h)}{2h} &= f'(x) + O(h^2)
\end{aligned}$$

Natürlich kann auch Spline–Interpolation zur Approximation der ersten und zweiten Ableitung verwendet werden, siehe die entsprechenden Fehlerabschätzungen.

Extrapolation:

Ein grundsätzliches Problem bei der Auswertung einer Approximation für eine Ableitung ist das unvermeidbare Auftreten von Auslöschung. Also sollten zu kleine Schrittweiten h vermieden werden. Um trotzdem ausreichend genaue Approximationen zu erhalten, kann man die Technik der Extrapolation verwenden:

Verwendet man zum Beispiel

$$T(h) = \frac{f(x+h) - f(x-h)}{2h}$$

als Approximation für die erste Ableitung $f'(x)$, so lässt sich leicht durch eine Taylor–Entwicklung nachweisen, dass gilt:

$$T(h) = f'(x) + a_1 h^2 + a_2 h^4 + \dots .$$

Diese Entwicklung legt nahe, dass man zunächst $T(h)$ für einige nicht zu kleine h berechnet und anschließend durch diese Werte ein Interpolationspolynom in h (hier aufgrund der speziellen Struktur der Entwicklung besser ein Interpolationspolynom in h^2) legt. Der Wert dieses Interpolationspolynoms an der Stelle $h = 0$ ergibt dann eine bessere Näherung. Da $h = 0$ außerhalb des Intervalls der Stützpunkte liegt, spricht man nicht von Interpolation sondern von Extrapolation.

7.4 Numerische Integration

Man kennt von einer Funktion $f(x)$ die Werte an Stützpunkten $a \le x_0 < x_1 < \ldots < x_n \le b$. Gesucht sind Näherungen von $\int_a^b f(x)\ dx$.

Grundidee: $f(x)$ wird durch eine Interpolationsfunktion $\varphi(x)$ ersetzt. Man verwendet $\int_a^b \varphi(x)\ dx$ als Näherungen für $\int_a^b f(x)\ dx$.

Beispiel: Bei Verwendung von Polynominterpolation erhält man aus der Lagrangeschen Darstellung des Interpolationspolynom:

$$\int_a^b P(x)\ dx = \sum_{i=0}^{n} \alpha_i f(x_i)$$

mit

$$\alpha_i = \int_a^b L_i(x)\, dx.$$

Man nennt Approximationen eines Integrals Quadraturformeln.

Wählt man im Speziellen äquidistante Stützpunkte $x_i = a + ih$ mit $h = (b-a)/n$, so erhält man die Newton–Cotes–Formeln. Einige Beispiel:

- n = 1: Trapezregel:

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

  Die Formel ist exakt für Polynome vom Grad 1.

- n = 2: Simpson–Regel, Keplersche Fassregel:

$$\int_a^b f(x)\ dx = (b-a)\left[\frac{1}{6}f(a) + \frac{2}{3}f((a+b)/2) + \frac{1}{6}f(b)\right]$$

  Die Formel ist exakt für Polynome vom Grad 3.

Von entscheidender Bedeutung für die Konvergenz von Quadraturformeln ist folgender Satz:

Satz 7.8. *(Banach-Steinhaus) Angenommen, eine Folge von Quadraturformeln hat positive Gewichte und diese Folge von Quadraturformeln konvergiert für $n \to \infty$ gegen den exakten Wert $\int_a^b f(x)\ dx$ für alle f aus einer dichten Teilmenge von $C[a,b]$. Dann konvergiert die Folge gegen $\int_a^b f(x)\ dx$ für alle $f \in C[a,b]$.*

Die Newton–Cotes–Formeln konvergieren trivialerweise für alle Polynome, allerdings treten bald negative Gewichte auf. Es lässt sich zeigen, dass die Newton–Cotes–Formel nicht immer konvergieren. Einen Ausweg bietet die Verwendung von Spline–Interpolation oder die summierte Newton–Cotes–Formeln, die entstehen, wenn man eine Newton–Cotes–Formel niedriger Ordnung auf Teilintervalle anwendet:

Beispiel: Summierte Trapezregel für äquidistanten Unterteilung $x_i = a + ih$ mit $h = (b-a)/n$:

$$\begin{aligned}\int_a^b f(x)\ dx = \sum_{i=0}^{n-1}\int_{x_i}^{x_{i+1}} f(x)\ dx &\approx \sum_{i=0}^{n-1}\frac{1}{2}h[f(x_i) + f(x_{i+1}] \\ &= h\left[\frac{1}{2}f(x_0) + f(x_1) + \ldots + f(x_{n-1}) + \frac{1}{2}f(x_n)\right]\end{aligned}$$

Gaußsche Quadraturformeln:

Wählt man als Stützpunkte die Nullstellen des Legrende–Polynoms vom Grad $n+1$, so erhält man besonders genaue Formeln, die Gaußschen Quadraturformeln:

Im Intervall $[a,b] = [-1,1]$ sind die Legrende–Polynome z.B. durch folgende Definition gegeben:

$$P_k(x) = \frac{k!}{(2k)!}\frac{d^k}{dx^k}(x^2-1)^k$$

Die im Zusammenhang mit den Quadraturformeln wichtigste Eigenschaft dieser Polynome ist die Orthogonalitätsbedingung:

$$\int_{-1}^{1} P_k(x)P_l(x)\ dx = 0 \quad \text{für } k \neq l.$$

Einige Beispiele von Gaußschen Quadraturformeln:

- $n = 0$: $P_1(x) = x$, $x_0 = 0$, $\alpha_0 = 2$. Man erhält die Mittelpunktsregel

  $$\int_{-1}^{1} f(x)\ dx = 2f(0).$$

  Die Formel ist exakt für Polynome vom Grad 1.

- $n = 1$: $P_2(x) = x^2 - 1/3$, $x_0 = -1/\sqrt{3}$, $x_1 = -1/\sqrt{3}$, $\alpha_0 = \alpha_1 = 1$:

  $$\int_{-1}^{1} f(x)\ dx = f(-1/\sqrt{3}) + f(1/\sqrt{3})$$

  Die Formel ist exakt für Polynome vom Grad 3.

Die Gaußschen Quadraturformeln konvergieren für alle Polynome und besitzen positive Gewichte. Also konvergieren sie für alle stetigen Funktionen.

Kapitel 8

Literaturzitate

Gould91 Gould N.: SIAM J.Matrix Anal.Appl., 12 (1991), 354–361

Edelman91 Edelman Alan.: Note to the Editor, SIAM J.Matrix Anal.Appl., 12 (1991)