Algorithmische Methoden in der Numerik

Vorlesungsskriptum

SS 2022

Helmut Gfrerer[1]

[1]Johannes Kepler Universität Linz, Institut für Numerische Mathematik, A-4040 Linz, Austria (helmut.gfrerer@jku.at)

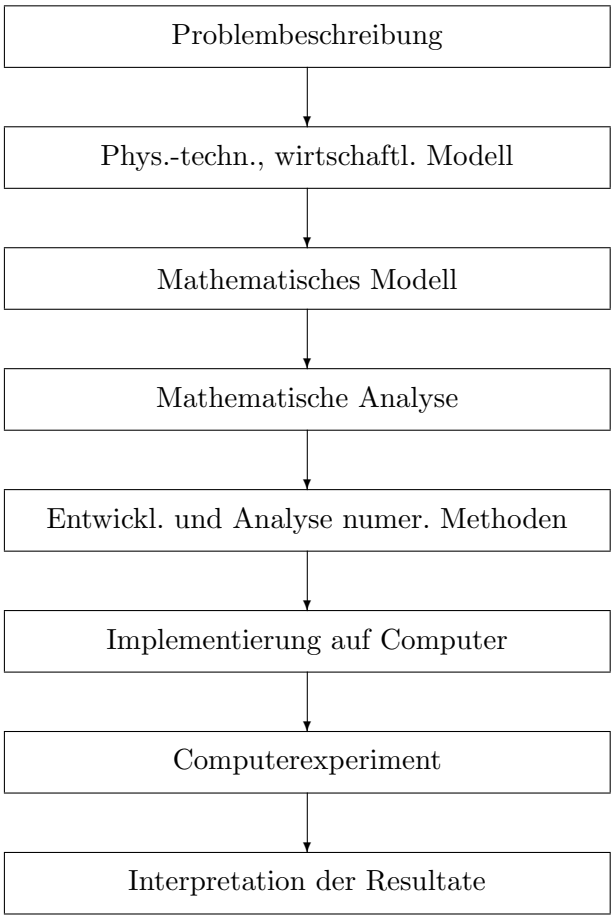
Inhaltsverzeichnis

Kapitel 1

Besonderheiten des numerischen Rechnens

1.1 Der Problemlösungsprozess

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



Das (physikalisch-technische oder wirtschaftl.) 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.

Früher war ein zentraler Begriff der mathematischen Analyse das *korrekt gestellte Problem*, d.h. nach Hadamard (1923)

1. eine Lösung des Problems existiert,
2. sie ist eindeutig und
3. sie hängt stetig von den Daten ab.

Die in vielen Fällen als selbstverständlich geltende Existenz (bzw. Eindeutigkeit) einer Lösung des realen Problems bedeutet keineswegs, dass auch ein Modell diese Eigenschaften besitzen muss. Gelingt in solchen Fällen der Nachweis der Existenz (bzw. 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.)

In der modernen Mathematik klammert man sich nicht mehr unbedingt an das Konzept des korrekt gestellten Problems hinsichtlich Existenz und Eindeutigkeit von Lösungen. Entscheidend ist bei Vorliegen eines mathematischen Problems die folgende Fragestellung: *Finde (mindestens) eine Lösung des Problems oder weise nach, dass keine Lösung existiert.* Damit verknüpft sind natürlich eine ganze Reihe weiterer Fragestellungen, z.B. falls wir als Anwort erhalten: "es gibt keine Lösung" die Frage, wie weit wir unsere Inputdaten verändern müssen, um die Lösbarkeit zu garantieren (und eine Lösung dann auch zu berechnen)

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. bei iterativen Verfahren).

Die Realisierung einer numerischen Methode am 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.

Wir müssen uns bei diesem Problemlösungsprozess vor Augen halten, dass die vielen Fehler, die auftreten können, ein vollkommen unbrauchbares Ergebnis liefern können. Man darf sich also keinesfalls darauf beschränken, nur zu rechnen und den Computerergebnissen blind zu vertrauen. Man muss also immer hinterfragen, ob die Rechenergebnisse auch wirklich richtig sind.

1.2 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.

Neben dieser Festkommadarstellung ist auch noch die normalisierte Gleitkommadarstellung von Bedeutung:

Satz 1.1. *Für jede Basis $2 \leq B \in \mathbb{N}$ und für jede reelle Zahl $x \neq 0$ gibt es genau eine Darstellung der Form*

$$x \;=\; sB^e \sum_{k=1}^{\infty} \alpha_k B^{-k},$$

die folgenden Bedingungen genügt:

1. $s \in \{-1, +1\}$,
2. $e \in \mathbb{Z}$,
3. $\alpha_k \in \{0, 1 \ldots, B-1\}$, $\alpha_1 \neq 0$,
4. $\forall k \in \mathbb{N} \exists i \geq k : \alpha_i \neq B - 1$ *(d.h., es sind nicht alle Ziffern gleich $B-1$ ab einem gewissen Index).*

Beweis. Sei $s = \text{sign}\,(x)$, $e = \lfloor \log_B(|x|) \rfloor + 1$, $z := y_0 := |x|B^{-e}$, $z_0 := 0$. Wegen der Monotonie des Logarithmus gilt für $z = y_0 = B^{\log_B(|x|) - \lfloor \log_B(|x|) \rfloor - 1}$ die Abschätzung $B^{-1} \leq z = y_0 < 1$. Wir definieren nun

$$\alpha_k := \lfloor B y_{k-1} \rfloor,\ z_k := z_{k-1} + B^{-k}\alpha_k,\ y_k := B y_{k-1} - \alpha_k,\ k = 1, 2, \ldots$$

Offensichtlich gilt dann $0 \leq y_k < 1$ für alle $k \in \mathbb{N}$ und daher $\alpha_k \in \{0, \ldots, B-1\}$. Wir zeigen nun mit Induktion die Beziehung $z = z_k + y_k B^{-k}$, $\forall k \in \mathbb{N}$. Dies ist offensichtlich der Fall für $k = 0$. Sei nun die Behauptung richtig für $k \geq 0$. Wegen

$$\begin{aligned} z &= z_k + y_k B^{-k} = z_k + \alpha_{k+1} B^{-(k+1)} + y_k B^{-k} - \alpha_{k+1} B^{-(k+1)} \\ &= z_{k+1} + B^{-(k+1)}(B y_k - \alpha_{k+1}) = z_{k+1} + B^{-(k+1)} y_{k+1} \end{aligned}$$

ist die Aussage richtig für $k+1$ und daher unsere Behauptung gezeigt. Gleichfalls zeigt man leicht mit Induktion, dass $z_k = \sum_{i=1}^{k} \alpha_i B^{-i}$ und wegen $y_k \in [0, 1)$ folgt $z = \lim_{k\to\infty} z_k = \sum_{i=1}^{\infty} \alpha_i B^{-i}$. Weiters ist

$$x = s|x| = sB^e z = sB^e \sum_{i=1}^{\infty} \alpha_i B^{-i}$$

und es bleibt nur mehr die letzte Eigenschaft sowie die Eindeutigkeit zu zeigen.

Wir zeigen die letzte Eigenschaft per Widerspruchsbeweis. Annamhe: $\exists k \forall i \geq k : \alpha_i = B - 1$. Dann gilt

$$y_{k-1} = B^{k-1}(z - z_{k-1}) = B^{k-1} \sum_{i=k} \alpha_i B^{-i} = B^{k-1}(B-1)B^{-k}\frac{1}{1 - 1/B} = 1$$

im Widerspruch zu $y_{k-1} < 1$, also ist die letzte Eigenschaft erfüllt.

Eindeutigkeit: Eindeutigkeit des Vorzeichens ist für $x \neq 0$ klar. Wegen $\alpha_1 \geq 1$ und 4) gilt

$$B^{-1} \leq \sum_{i=1}^{\infty} \alpha_i B^{-i} < \sum_{i=1}^{\infty} (B-1)B^{-i} = 1$$

und damit für den Exponenten

$$e - 1 \leq \log_B(|x|) < e$$

woraus sich mit $e \in \mathbb{Z}$ die eindeutige Festlegung $e = \lfloor \log_B(|x|) \rfloor + 1$ ergibt. Seien nun (α_i), $(\hat{\alpha}_i)$ zwei verschiedene Folgen mit $\sum_{i=1}^{\infty} \alpha_i B^{-i} = \sum_{i=1}^{\infty} \hat{\alpha}_i B^{-i}$ und k der erste auftretende Index mit $\alpha_k \neq \hat{\alpha}_k$, o.B.d.A. $\alpha_k < \hat{\alpha}_k \Rightarrow$

$$0 = \sum_{i=1}^{\infty} (\hat{\alpha}_i - \alpha_i) B^{-i} = \sum_{i=k}^{\infty} (\hat{\alpha}_i - \alpha_i) B^{-i} \geq B^{-k} + \sum_{i=k+1}^{\infty} (\hat{\alpha}_i - \alpha_i) B^{-i} \geq B^{-k} - \sum_{i=k+1}^{\infty} (B-1) B^{-i} = 0$$

und daher $\hat{\alpha}_i - \alpha_i = -(B-1)$, $\forall i \geq k+1 \Rightarrow \hat{\alpha}_i = 0$, $\alpha_i = B - 1$, $\forall i \geq k+1$ im Widerspruch zu 4). Damit ist auch die Eindeutigkeit gezeigt. □

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

$$x = \pm 0, \alpha_1 \alpha_2 \ldots \alpha_t \cdot B^e \tag{1.1}$$

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 (1.1) 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

$$|x - rd(x)| \leq |x - y| \quad \text{für alle Maschinenzahlen } y. \tag{1.2}$$

Diese Forderung lässt sich leicht realisieren: Für

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

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 und man sagt: $\bar{x}$ hat s gültige Nachkommastellen (Dezimalen). Die Ziffern an der Stelle s und davor (**ohne** die führenden Nullen) heißen signifikante Ziffern. Die gültigen Ziffern beschreiben den absoluten Fehler, die Anzahl der signifikanten Ziffern den relativen Fehler.
Beispiel: $x = 3.14159, \bar{x} = 3.142$: Dann hat $\bar{x}$ 3 gültige Nachkommastellen (Dezimalen) und 4 signifikante Ziffern
$x = 0.00345, \bar{x} = 0.0035$: $\bar{x}$ hat 4 gültige Nachkommastellen und 2 signifikante Ziffern.

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.

Satz 1.2. *Für das Runden nach (1.2) gilt:*

$$\frac{|rd(x) - x|}{|x|} \leq eps \tag{1.3}$$

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

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

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:

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

□

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 (1.3) für die Maschinengenauigkeit $\text{eps} = B^{1-t}$.

1.3 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:

$$x +^* y = rd(x + y),\ \forall x, y \in \mathcal{M}.$$

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:

$$x \circ^* y = rd(x \circ y), \ \forall x, y \in \mathcal{M},$$

woraus wie oben für die Genauigkeit folgt:

$$\frac{|(x \circ^* y) - (x \circ y)|}{|x \circ y|} \leq \text{eps}. \tag{1.4}$$

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

$$\frac{|f^*(x) - f(x)|}{|f(x)|} \leq \text{eps} \tag{1.5}$$

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 (1.4), (1.5) **elementare Operationen**.

Man beachte ferner, dass die Aussagen nur für Maschinenzahlen $x, y \in \mathcal{M}$ gültig sind. Die Verknüpfung reeller Zahlen $x, y \notin \mathcal{M}$ führt auf $rd(rd(x) \circ rd(y))$!

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.

1.4 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.

1.5 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:

1.5.1 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:

$$\Delta y = \varphi(x + \Delta x) - \varphi(x) \approx \varphi'(x) \cdot \Delta x \tag{1.6}$$

bzw. komponentenweise

$$\Delta y_i = \varphi_i(x + \Delta x) - \varphi_i(x) \approx \sum_{j=1}^{n} \frac{\partial \varphi_i}{\partial x_j}(x) \cdot \Delta x_j \quad \text{für} \quad i = 1, \ldots, m \tag{1.7}$$

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

$$\varepsilon_{y_i} = \frac{\Delta y_i}{y_i} \approx \sum_{j=1}^{n} \frac{x_j}{\varphi_i(x)} \frac{\partial \varphi_i}{\partial x_j}(x) \cdot \varepsilon_{x_j}. \tag{1.8}$$

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. Andernfalls, wenn kleine Fehler in den Daten große Fehler im Ergebnis bewirken können, heißt das Problem **schlecht konditioniert** (**dateninstabil**).

Nach der Fehlerformel (1.8) ist ein Problem dann gut (bzw. schlecht) konditioniert, wenn die Konditionszahlen $|k_{ij}|$ des Problems klein (bzw. groß) im Vergleich mit 1 sind. Die Begriffe "klein" und "groß sind hier im Sinne von Größenordnungen zu verstehen.

Daher ist obige Definition unscharf: Das Gegenteil von "gut konditioniert" ist nicht "schlecht konditioniert", vielmehr gibt es hier einen Graubereich, wo ein Problem nicht eindeutig gut oder schlecht konditioniert ist, z.B. wenn eine Konditionszahl 10 oder 100 ist.

Bezeichnung: Die Fehlerformel (1.6) ist in Matrix-Vektor-Schreibweise mit:

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

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 (1.8):

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

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 (1.8):

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

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 (1.8) 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.

1.5.2 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:

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

Jeder Zwischenschritt entspricht formal einer Abbildung $x^{(i+1)}=\varphi^{(i)}(x^{(i)})$, $i=0,\ldots,r$, wobei jede Komponente von $\varphi^{(i)}: D_i\subset\mathbb{R}^{n_i}\to\mathbb{R}^{n_{i+1}}$ einer elementaren Operation entspricht. Die Abbildung, die das Zwischenergebnis $x^{(s)}$ auf y abbildet, wird mit $\psi^{(s)}$ bezeichnet und heißt Restabbildung, wobei $s=0,1,2,\ldots,r$. Es gilt $\psi^{(s)}=\varphi^{(r)}\circ\ldots\circ\varphi^{(s+1)}\circ\varphi^{(s)}$

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. $y=t-p$

Dies entspricht den Zwischenschritten

$$\begin{aligned}
x^{(0)} &= (p,q)\to\varphi^{(0)}(x^{(0)})=x^{(1)}=(p^2,p,q)=(r,p,q)\\
x^{(1)} &= (r,p,q)\to\varphi^{(1)}(x^{(1)})=x^{(2)}=(r+q,r,p,q)=(s,r,p,q)\\
x^{(2)} &= (s,r,p,q)\to\varphi^{(2)}(x^{(2)})=x^{(3)}=(\sqrt{s},s,r,p,q)=(t,s,r,p,q)\\
x^{(3)} &= (t,s,r,p,q)\to\varphi^{(3)}(x^{(3)})=t-p=y
\end{aligned}$$

Die dazugehörigen Restabbildungen lauten:

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

Der Einfachheit halber inkludieren wir hier in jedem Zwischenschritt auch jedes bis jetzt berechnete Zwischenergebnis, auch wenn man es nicht braucht. Eliminiert man die Zwischenresultate , die nicht mehr benötigt werden, lautet ein äquivalenter Algorithmus:

$$\tilde{\varphi}^{(0)} : (p,q) \to (p^2,p,q), \quad \tilde{\varphi}^{(1)} : (r,p,q) \to (r+q,p), \quad \tilde{\varphi}^{(2)} : (s,p) \to (\sqrt{s},p), \quad \tilde{\varphi}^{(3)} : (t,p) \to t-p$$

mit Restabbildungen

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

Nach der Fehlerformel (1.7) 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 (1.7) zu einer Verfälschung des Endergebnisses um näherungsweise $(\psi^{(s)})'(x^{(s)})\Delta x^{(s)}$ führt.

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

Die Fehler $\Delta x^{(s)}$ in den einzelnen Zwischenschritten lassen sich schreiben als

$$\Delta x^{(s)} = \begin{pmatrix} \Delta x_1^{(s)} \\ \vdots \\ \Delta x_{n_s}^{(s)} \end{pmatrix} = \begin{pmatrix} \varepsilon_1^{(s)} & & 0 \\ & \ddots & \\ 0 & & \varepsilon_{n_s}^{(s)} \end{pmatrix} \begin{pmatrix} x_1^{(s)} \\ \vdots \\ x_{n_s}^{(s)} \end{pmatrix} =: E^{(s)} x^{(s)}$$

mit $|\varepsilon_j^{(s)}| \leq \text{eps}$, $j = 1, \ldots, n_s$. Dabei muss man beachten, dass eventuell einige dieser $\varepsilon_j^{(s)}$ gleich 0 sind, nämlich für die Komponenten $x_j^{(s)}$, die durch Kopieren von Komponenten aus $x^{(s-1)}$ hervorgegangen sind.

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

$$\Delta y^0 = \varphi'(x)\Delta x^{(0)} + \Delta x^{(r+1)},$$

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

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

der vom gewählten Algorithmus abhängt.

Wir schreiben nun für Vektoren $x \in \mathbb{R}^n$ und $m \times n$ Matrizen A

$$|x| = \begin{pmatrix} |x_1| \\ \vdots \\ |x_n| \end{pmatrix}, \quad |A| = \begin{pmatrix} |a_{11}| & \ldots & |a_{1n}| \\ \vdots & \ddots & \vdots \\ |a_{m1}| & \ldots & |a_{mn}| \end{pmatrix} \quad \text{(komponentenweise)}$$

Bezeichnet $\hat{E}^{(s)}$, $s = 1, \ldots, r$ jene Diagonalmatrizen mit Diagonalelementen 0 oder 1, für die $|E^{(s)}| \leq \mathrm{eps}\hat{E}^{(s)}$ gilt, so erhält man die folgenden Fehlerschranken:

$$\begin{aligned} |\Delta y^0| &\leq \mathrm{eps}\,(|\varphi'(x^{(0)}|\,|x^{(0)}| + |y|) \\ |\Delta y^r| &\leq \mathrm{eps}\sum_{s=1}^{r} |(\psi^{(s)})'(x^{(s)})|\hat{E}^{(s)}|x^{(s)}|. \end{aligned}$$

Ein Algorithmus heißt **numerisch stabil**, wenn der restliche Rundungsfehler Δy^r den unvermeidbaren Fehler Δy^0 nicht dominiert, d.h. kleiner oder höchstens von der gleichen Größenordnung ist.

Ist dagegen der restliche Rundungsfehler "sehr viel" größer als der unvermeidbare Fehler, so heißt der Algorithmus numerisch instabil.

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:

$$\begin{aligned} |\Delta y^0| &\leq \mathrm{eps}\left(\left|\frac{p}{\sqrt{p^2+q}} - 1\right| \cdot p + \frac{1}{2\sqrt{p^2+q}} \cdot q + \left|\sqrt{p^2+q} - p\right|\right) \\ &= \mathrm{eps}\left(\frac{q\,p}{(\sqrt{p^2+q}+p)\sqrt{p^2+q}} + \frac{q}{2\sqrt{p^2+q}} + \frac{q}{\sqrt{p^2+q}+p}\right) \\ &\approx \mathrm{eps}\left(\frac{q}{2p} + \frac{q}{2p} + \frac{q}{2p}\right) = \mathrm{eps}\,\frac{3q}{2p} \end{aligned}$$

und wegen $\hat{E}^{(s)}_{11} = 1$, $\hat{E}^{(s)}_{ii} = 0$, $i = 2, \ldots, n_s$, $s = 1, 2, 3$

$$\begin{aligned} |\Delta y^r| &\leq \mathrm{eps}\sum_{s=1}^{r} \left|\frac{\partial\psi^{(s)}(x^{(s)})}{\partial x_1}\right| \left|x_1^{(s)}\right| \\ &= \mathrm{eps}\left(\frac{1}{2\sqrt{r+q}}\,r + \frac{1}{2\sqrt{s}}\,s + t\right) \\ &= \mathrm{eps}\left(\frac{p^2}{2\sqrt{p^2+q}} + \frac{1}{2}\sqrt{p^2+q} + \sqrt{p^2+q}\right) \\ &\approx \mathrm{eps}\left(\frac{1}{2}p + \frac{1}{2}p + p\right) \approx 2\,\mathrm{eps}\,p \end{aligned}$$

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.

1.6 Experimentelle Bestimmung des absoluten Fehlers

Bis jetzt sind wir davon ausgegangen, dass wir den Algorithmus zur Berechnung von

$$y = \varphi(x)$$

in seine Einzelschritte zerlegen können und somit seine numerische Stabilität bewerten können. Aber schon bei relativ einfachen Funktionen φ ist diese Vorgangsweise ziemlich aufwändig und schwierig durchzuführen. Unmöglich wird dies außerdem, wenn die Funktionsauswertung in einem (kompilierten) Computerprogramm vorliegt, dessen Sourcecode wir nicht kennen.

Wir betrachten also folgende Situation: Durch ein Computerprogramm ist gegeben eine Funktion $\hat{\varphi} : \mathcal{M} \to \mathcal{M}$, die für jedes reelle x eine Näherung $\hat{\varphi}(rd(x))$ für $\varphi(x)$ zurückliefert und wir suchen eine Fehlerschranke der Form

$$|\hat{\varphi}(rd(\bar{x})) - \varphi(\bar{x}) \leq \varepsilon_A,$$

die für alle $\bar{x}$ nahe einem gegebenen Datum x gültig ist.

In gewissen Fällen besteht die folgende Möglichkeit, diesen Berechnungsfehler aus den Funktionswerten mit statistischen Methoden abzuschätzen: Für eine kleine Schrittweite h (z.B. $h = 10^{-5}(0.1+|x|)$ berechnet man die Funktionswerte $\hat{\varphi}_i := \hat{\varphi}(rd(x_i))$, wobei $x_i = \bar{x}+ih$, $i = 0, \ldots, p$ und schreibt diese Werte in die 1.Spalte einer Differenzentabelle:

$$\begin{array}{llll}
\Delta^0\hat{\varphi}_0 := \hat{\varphi}_0 & \Delta^1\hat{\varphi}_0 := \Delta^0\hat{\varphi}_1 - \Delta^0\hat{\varphi}_0 & \ldots & \Delta^k\hat{\varphi}_0 := \Delta^{k-1}\hat{\varphi}_1 - \Delta^{k-1}\hat{\varphi}_0 \\
\Delta^0\hat{\varphi}_1 := \hat{\varphi}_1 & \Delta^1\hat{\varphi}_1 := \Delta^0\hat{\varphi}_2 - \Delta^0\hat{\varphi}_1 & \ldots & \Delta^k\hat{\varphi}_1 := \Delta^{k-1}\hat{\varphi}_2 - \Delta^{k-1}\hat{\varphi}_1 \\
\vdots & & \ddots & \\
\Delta^0\hat{\varphi}_{p-1} := \hat{\varphi}_{p-1} & \Delta^1\hat{\varphi}_{p-1} := \Delta^0\hat{\varphi}_p - \Delta^0\hat{\varphi}_{p-1} & & \\
\Delta^0\hat{\varphi}_p := \hat{\varphi}_p & & &
\end{array}$$

also das ist jetzten hier so eine differenzentabelle:

Die Elemente der $j+1$-ten Spalte dieser Tabelle berechnen sich nach der Vorschrift $\Delta^j\hat{\varphi}_i := \Delta^{j-1}\hat{\varphi}_{i+1} - \Delta^{j-1}\hat{\varphi}_i$, $i = 0, 1, 2, \ldots, p-j$.

Unter der Annahme, dass $\hat{\varphi}(rd(x_i)) = \varphi(x_i) + \epsilon_A\delta_i$, wobei δ_i unabhängige Zufallsvariable aus $(-1, 1)$ sind, und die Funktion φ hinreichend oft stetig differenzierbar ist, sind die Elemente der $k+1$-ten Spalte (k hinreichend groß, z.B. $k \geq 4$) ungefähr gleich in Größe, aber besitzen alternierendes Vorzeichen. Es kann dann die folgende Formel herangezogen werden:

$$\varepsilon_A \approx \frac{\max_{i\in\{0,1,\ldots,p-k\}} |\Delta^k\hat{\varphi}_i|}{\sqrt{\binom{2k}{k}}}.$$

Beispiel: Wir betrachten die Auswertung der Funktion $\varphi(x) = \frac{1-\cos x}{x^2}$ durch folgende MATLAB Funktion:

```
function [ f ] = tf1( x )
    if x ~= 0
        f=(1-cos(x))/(x*x);
    else
        f=0;
    end
end
```

Mit $h = 1.e-6$ erhält man für $x = 0.0001$ folgende Differenzentabelle

$10^0\times$	$10^{-8}\times$	$10^{-7}\times$	$10^{-7}\times$	$10^{-7}\times$	$10^{-7}\times$	$10^{-6}\times$
0.4999999969612	0.6210902	-0.132314	0.216933	-0.324399	0.533857	-0.1047379
0.5000000031721	-0.7020506	0.084619	-0.107465	0.209457	-0.513522	0.1220427
0.4999999961516	0.1441427	-0.022846	0.101991	-0.304065	0.706904	-0.1409971
0.4999999975930	-0.0843223	0.079145	-0.202073	0.402839	-0.703066	0.1103927
0.4999999967498	0.7071321	-0.122927	0.200766	-0.300227	0.400860	-0.0501556
0.5000000038211	-0.5221452	0.077838	-0.099461	0.100633	-0.100696	0.0190808
0.4999999985997	0.2562395	-0.021622	0.001171	-0.000063	0.090111	
0.5000000011621	0.0400122	-0.020450	0.001108	0.090048		
0.5000000015622	-0.1644966	-0.019342	0.091157			
0.4999999999172	-0.3579209	0.071814				
0.4999999963380	0.3602248					
0.4999999999403						

Ab der 5.Spalte (d.h. $k = 4$) haben die Werte alternierendes Vorzeichen und für $k = 4, 5, 6$ erhält man folgende Ergebnisse nach obiger Formel

$k = 4$: $\varepsilon_A \approx \frac{0.403\times 10^{-7}}{\sqrt{70}} = 0.48 \times 10^{-8}$

$k = 5$: $\varepsilon_A \approx \frac{0.707\times 10^{-7}}{\sqrt{252}} = 0.45 \times 10^{-8}$

$k = 6$: $\varepsilon_A \approx \frac{0.141\times 10^{-6}}{\sqrt{924}} = 0.46 \times 10^{-8}$

Der tatsächliche maximale Fehler in den berechneten Funktionswerten ist 0.42×10^{-8}, wir erhalten also eine sehr gute Übereinstimmung.

Kapitel 2

Direkte Verfahren zur Lösung linearer Gleichungssysteme

Das Lösen von linearen Gleichungssystemen ist eine fundamentale Problemstellung in der Numerischen Mathematik. 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**).

2.1 Ein Beispiel

Temperaturverteilung in einem Stab 2: Electric Boogaloo

Das Problem der Berechnung einer stationären Temperaturverteilung in einem Stab führt auf ein Randwertproblem der folgender Art:

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.

Typisch für ein numerisches Verfahren ist nun, dass die Bestimmung von Funktionswerten $u(x)$ an unendlich vielen Stellen (nämlich an allen $x \in [0,1]$) durch die Bestimmung an endlich vielen Stellen $x_i, i = 0, \ldots, N$ ersetzt wird. Diesen Vorgang nennt man *Diskretisierung*: Wir ersetzen dabei das Intervall $[0,1]$ durch eine endliche Punktmenge $\{x_0, x_1, \ldots, x_N\}$ (z.B. äquidistante Diskretisierung $x_i = ih$, $i = 0, \ldots, N$, mit $h = 1/N$, es gibt aber auch andere)

Die 2.Ableitung wird nun durch einen Differenzenquotienten ersetzt. Wegen

$$u''(x) = \lim_{h \to 0} \frac{1}{h^2}(u(x-h) - 2u(x) + u(x+h)$$

approximieren wir bei einer äquidistanten Diskretisierung

$$u''(x_i) \approx \frac{1}{h^2}(u(x_{i-1}) - 2u(x_i) + u(x_{i+1}))$$

Es entsteht folgendes lineares Gleichungssystem:

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

mit den Randbedingungen

$$u_0 = u_N = 0$$

für die Unbekannten u_i, $i = 1, 2, \ldots, N-1$, 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

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

und den Vektoren

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

2.2 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 über Datenfehler ist das Problem, aus gegebenen Daten A und b die gesuchte Lösung x auszurechnen, rein formal durch

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

gegeben. Verfälschte Daten $\bar{b} = b + \Delta b$ und $\bar{A} = A + \Delta A$ führen auf ein verfälschtes Ergebnis $\bar{x} = x + \Delta x$. Den Fehler komponentenweise abzuschätzen, würde auf eine unübersichtliche Situation führen.

Bemerkung: Achtung: Die Schreibweise $x = A^{-1}b$ ist nur ein Formalismus, um auszudrücken: Bestimme eine Lösung des Gleichungssystems $Ax = b$. Normalerweise wird beim numerischen Rechnen **nie** eine Inverse berechnet!

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:

$$\|x\|_2 = \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2}. \tag{2.1}$$

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 Vektornorm verwendet werden.

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

$$\|x\|_\infty = \max_{i=1,2,\ldots,n} |x_i|. \tag{2.2}$$

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:

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|. \tag{2.3}$$

Beispiel: Die Normen (2.1), (2.2), (2.3) 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:

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

wenn A eine $m \times n$ Matrix ist und $_m\|\cdot\|$ bzw. $_n\|\cdot\|$ jene Normen auf $\mathbb{R}^m$ bzw. $\mathbb{R}^n$ sind, mit denen wir gerade arbeiten. Man nennt in diesem Fall die Matrixnorm $\|A\|$ zu den Vektornormen $_m\|\cdot\|$ und $_n\|\cdot\|$ passend.

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

$$\|A\| := \sup_{x\neq 0} \frac{{}_m\|Ax\|}{{}_n\|x\|} = \sup\{{}_m\|Ax\| \mid {}_n\|x\| = 1\}. \tag{2.4}$$

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. Man nennt diese Norm die den entsprechenden Vektornormen zugeordnete Matrixnorm (Operatornorm).

Darüber hinaus gelten zusätzlich noch die folgenden Rechenregeln für Vektornormen und zugeordnete Matrixnormen:

1. Seien ${}_n\|\cdot\|$, ${}_p\|\cdot\|$, ${}_m\|\cdot\|$ Vektornormen auf $\mathbb{R}^n$, $\mathbb{R}^p$ und R^m und ${}_{p,n}\|\cdot\|$, ${}_{m,p}\|\cdot\|$ und ${}_{m,n}\|\cdot\|$ die zugeordneten Matrixnormen für Matrizen $A \in \mathbb{R}^{p\times n}$, $B \in \mathbb{R}^{m\times p}$ und $C \in \mathbb{R}^{m\times n}$. Dann gilt
$$ {}_{m,n}\|BA\| \leq {}_{m,p}\|B\| \; {}_{p,n}\|A\|.$$
Dies ergibt sich sofort aus der Beziehung (falls $A \neq 0$)
$$ {}_{m,n}\|BA\| = \sup_{x\neq 0} \frac{{}_m\|BAx\|}{{}_n\|x\|} = \sup_{\substack{x\neq 0\\ Ax\neq 0}} \frac{{}_m\|B(Ax)\|}{{}_p\|Ax\|} \frac{{}_p\|Ax\|}{{}_n\|x\|} \leq \sup_{y\neq 0} \frac{{}_m\|By\|}{{}_p\|y\|} \sup_{x\neq 0} \frac{{}_p\|Ax\|}{{}_n\|x\|}$$

2. Eine Matrixnorm $\|\cdot\|$ für quadratische $n \times n$ Matrizen heißt submultiplikativ, wenn
$$\|AB\| \leq \|A\| \, \|B\| \quad \text{für alle } A, B \in \mathbb{R}^{n\times n}.$$
Ist $m = n$ und ${}_m\|\cdot\| = {}_n\|\cdot\|$, so ist die zugeordnete Matrixnorm submultiplikativ.

3. Für die Einheitsmatrix gilt: $\|I\| = 1$, falls die Matrixnorm einer Vektornorm zugeordnet ist.

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 2.1. *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 2.2. *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 $\varepsilon_A = \|\Delta A\|/\|A\|$.

Beweis. 1. Widerspruchsbeweis. Annahme: $A + \Delta A$ nicht regulär $\Rightarrow (A + \Delta A)\tilde{x} = 0$ für ein $\tilde{x} \neq 0 \Rightarrow \Delta A\tilde{x} = -A\tilde{x}$, Da A regulär ist, folgt $A^{-1}\Delta A\tilde{x} = -\tilde{x}$ und

$$\|A^{-1}\|\|\Delta A\| \geq \|A^{-1}\Delta A\| \geq \frac{\|A^{-1}\Delta A\tilde{x}\|}{\|\tilde{x}\|} = 1,$$

ein Widerspruch zur Voraussetzung.

2. Aus $(A+\Delta A)(x+\Delta x) = b+\Delta b$ und $Ax = b$ folgt $(A+\Delta A)\Delta x = \Delta b - \Delta A x$ und daher

$$\|\Delta x\| = \|(A + \Delta A)^{-1}(\Delta b - \Delta A x\| \leq \|(A + \Delta A)^{-1}\|(\|\Delta b\| + \|\Delta A\|\|x\|) \tag{2.5}$$

Wegen $(A + \Delta A)(A + \Delta A)^{-1} = I$ folgt $A(A + \Delta A)^{-1} = I - \Delta A(A + \Delta A)^{-1}$ und nach Multiplikation mit A^{-1}

$$(A + \Delta A)^{-1} = A^{-1} - A^{-1}\Delta A(A + \Delta A)^{-1} \tag{2.6}$$

Nimmt man auf beiden Seiten die Norm, ergibt sich

$$\|(A + \Delta A)^{-1}\| = \|A^{-1} - A^{-1}\Delta A(A + \Delta A)^{-1}\| \leq \|A^{-1}\| + \|A^{-1}\Delta A\|\|(A + \Delta A)^{-1}\|$$

und daher

$$\|(A + \Delta A)^{-1}\| \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\Delta A\|} \leq \frac{\|A^{-1}\|}{1 - \|A^{-1}\|\|\Delta A\|} = \frac{\|A^{-1}\|}{1 - \kappa(A)\,\varepsilon_A}$$

Wegen

$$\|\Delta b\| + \|\Delta A\|\|x\| = \|Ax\|\varepsilon_b + \|A\|\varepsilon_A\|x\| \leq \|A\|(\varepsilon_b + \varepsilon_A)\|x\|$$

folgt aus (2.5) die Behauptung.

□

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: Aus dem Beweis ergibt sich dass die Aussage 1. unter der schwächeren Bedingung $\|A^{-1}\Delta A\| < 1$ gültig bleibt. Weiters erhalten wir aus (2.6) die Abschäzung

$$\|(A+\Delta A)^{-1} - A^{-1}\| \leq \|A^{-1}\Delta A(A+\Delta A)^{-1}\| \leq \|A^{-1}\Delta A\|\|(A+\Delta A)^{-1}\| \leq \frac{\|A^{-1}\Delta A\|\|A^{-1}\|}{1 - \|A^{-1}\Delta A\|}.$$

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. Weiters gilt $\|I\| = \|I \cdot I\| \leq \|I\|^2$ und daher $\kappa(A) \geq \|I\| \geq 1$.

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

   $$\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$$

   bezeichnet. Wir haben bereits hergeleitet: $\|A\|_2 = \sigma_1$.

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

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

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

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

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

   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

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

   woraus für die Konditionszahl folgt:

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

   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

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

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

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

Für den kleinsten und größten Eigenwert folgt:

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

und

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

Somit erhält man für die Konditionszahl:

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

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.

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

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

Es stehen alle Normen in einer gewissen Abhängigkeit zueinander:

Satz 2.3. *In einem endlichdimensionalen Raum E sind alle Normen äquivalent, d.h. für jedes Paar ${}_\alpha\|\cdot\|$, ${}_\beta\|\cdot\|$ von Normen gibt es Konstanten c, C mit $C \ge c > 0$ sodass*

$$c\ {}_\alpha\|x\| \ \le\ {}_\beta\|x\| \ \le\ C\ {}_\alpha\|x\|,\ \forall x \in E$$

Beweis. 1. Sei $n = \dim E$ und $e_1, \dots, e_n$ eine beliebige Basis von E. Für jedes x existiert dann eine eindeutige Darstellung als Linearkombination $x = \sum_{i=1}^n y_i e_i$ mit $y_i \in \mathbb{R}$, d.h. $(y_1, \dots, y_n)^T \in \mathbb{R}^n$. Wir definieren nun

$${}_2\|\sum_{i=1}^n y_i e_i\| := \sqrt{\sum_{i=1}^n |y_i|^2} = \|y\|_2$$

Wie man leicht nachrechnet, ist ${}_2\|\cdot\|$ eine Norm auf E.

2. Wir zeigen nun die Behauptung für ${}_\alpha\|\cdot\| = {}_2\|\cdot\|$. Für $x = \sum_{i=1}^n y_i e_i$ gilt offensichtlich nach Anwendung der Cauchy-Schwarz'schen Ungleichung

$${}_\beta\|x\| \le \sum_{i=1}^n |y_i|\,{}_\beta\|e_i\| \le \|y\|_2 \underbrace{\sqrt{\sum_{i=1}^n {}_\beta\|e_i\|^2}}_{=:C} = C\ {}_2\|x\|$$

Für die Abbildung $\varphi : \mathbb{R}^n \to \mathbb{R}$, $\varphi(y) := {}_\beta\|\sum_{i=1}^n y_i e_i\|$ gilt damit wegen der 2. Dreiecksungleichung

$$|\varphi(y) - \varphi(z)| \le \varphi(y - z) \le C\|y - z\|_2.$$

φ is also stetig. Die Menge $\mathcal{S} := \{y \in \mathbb{R}^n \mid \|y\|_2 = 1\}$ ist abgeschlossen und beschränkt und daher nach dem Satz von Heine-Borel kompakt. Die Funktion φ nimmt also auf $\mathcal{S}$ ein Minimum an einer Stelle $\bar{y}$ an. Dann ist $\bar{y} \neq 0$ und damit auch $\bar{x} := \sum_{i=1}^n \bar{y}_i e_i \neq 0$, woraus sich aus der Definitheit $c := \varphi(\bar{y}) = {}_\beta\|\bar{x}\| > 0$ ergibt. Daher folgt mit Hilfe der Homogenität für alle $x = \sum_{i=1}^n y_i e_i \neq 0$

$$c \leq \varphi(\frac{y}{\|y\|_2}) = {}_\beta\|\frac{1}{\|y\|_2}\sum_{i=1}^n y_i e_i\| = \frac{1}{\|y\|_2}{}_\beta\|\sum_{i=1}^n y_i e_i\| = \frac{1}{{}_2\|x\|}{}_\beta\|x\|$$

Es folgt für alle $x \neq 0$ die Ungleichung $c\ {}_2\|x\| \leq {}_\beta\|x\|$, die natürlich auch für $x = 0$ richtig ist.

3. Für zwei beliebige Normen ${}_\alpha\|\cdot\|$, ${}_\beta\|\cdot\|$ E seien c_1, c_2, C_1, C_2 so gewählt, dass

$$c_1\ {}_\alpha\|x\| \leq {}_2\|x\| \leq C_1\ {}_\alpha\|x\|,\ c_2\ {}_\beta\|x\| \leq {}_2\|x\| \leq C_2\ {}_\beta\|x\|$$

Dann folgt sofort

$$\frac{c_1}{C_2}\ {}_\alpha\|x\| \leq {}_\beta\|x\| \leq \frac{C_1}{c_2}\ {}_\alpha\|x\|$$

□

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.

Wir haben unsere Betrachtungen nur für den $\mathbb{R}^n$ bzw. reelle Matrizen durchgeführt. Wie man sich leicht überlegt, gelten alle Ausführungen auch im $\mathbb{C}^n$ bzw. für komplexwertige Matrizen.

2.3 Rückwärtsanalyse

Per Definition wissen wir, dass $x \in \mathbb{R}^n$ genau dann Lösung des Gleichungssystems $Ax = b$ wenn $Ax - b = 0$ wissen. Es ist daher naheliegend, $\bar{x} \in \mathbb{R}^n$ als Näherungslösung ("numerische Lösung") zu akzeptieren, wenn $\|A\bar{x} - b\|$ "klein " ist. Wir bezeichnen für beliebiges $\bar{x} \in \mathbb{R}^n$ die Größe $r(\bar{x}) := A\bar{x} - b$ als das *Residuum* von $\bar{x}$.

Natürlich wissen wir schon auf Grund unserer Überlegungen bezüglich der Datenstabilität, dass aus der Kleinheit des Residuums nicht unbedingt auf die Brauchbarkeit des Ergebnisses geschlossen werden kann: Offensichtlich ist jedes $\bar{x} \in \mathbb{R}^n$ exakte Lösung des Gleichungssystem $A\bar{x} = \bar{b}$ mit $\bar{b} = b + r(\bar{x})$ und daher gilt für den absoluten Fehler

$$\|\bar{x} - x\| \leq \|A^{-1}\|\|r(\bar{x})\|$$

bzw. nach Satz 2.1 für den relativen Fehler

$$\varepsilon_x = \frac{\|\bar{x} - x\|}{\|x\|} \leq \kappa(A)\frac{\|r(\bar{x})\|}{\|b\|},$$

wobei x die exakte Lösung des Gleichungssystems $Ax = b$ bezeichnet. Daher ist die Größe des Residuums nur bedingt aussagekräftig über den Fehler in der Lösung.

Eine andere Möglichkeit, auf die Brauchbarkeit einer Lösung zu schließen, besteht in der sogenannten Rückwärtsanalyse: Dabei geht es darum, festzustellen, ob die erhaltene Lösung $\bar{x}$ als exakte Lösung eines "benachbarten" Gleichungssystems aufgefasst werden kann.

Satz 2.4 (Prager-Öttli)**.** *Seien $\bar{A}, \Delta A \in \mathbb{R}^{m\times n}$ 2 Matrizen mit $\Delta A \geq 0$, $\bar{b}, \Delta b \in \mathbb{R}^m$ 2 Vektoren mit $\Delta b \geq 0$ und $\bar{x} \in \mathbb{R}^n$ mit Residuum $r(\bar{x}) = \bar{A}\bar{x} - \bar{b}$.*

Dann gibt es genau dann eine Matrix $A \in \mathbb{R}^{m\times n}$ mit $|A - \bar{A}| \leq \Delta A$ und einen Vektor $b \in \mathbb{R}^m$ mit $|b - \bar{b}| \leq \Delta b$, sodass $\bar{x}$ exakte Lösung von $A\bar{x} = b$ ist, wenn $|r(\bar{x})| \leq \Delta A|\bar{x}| + \Delta b$.

Bezeichnung: $|\ |$, $\leq$ und $\geq$ sind hier für Matrizen und Vektoren komponentenweise zu verstehen: z.B.

$$\Delta A \geq 0 \ \Leftrightarrow\ \Delta A_{ij} \geq 0,\ \forall i = 1,\dots,m;\ j = 1,\dots,n$$

$$|\bar{x}| = \begin{pmatrix} |\bar{x}_1| \\ \vdots \\ |\bar{x}_n| \end{pmatrix}, \quad |b - \bar{b}| \leq \Delta b \ \Leftrightarrow\ |b_i - \bar{b}_i| \leq \Delta b_i, \forall i = 1,\dots,m$$

Wir betrachten nun das Gegenteil von der Aussage $\Delta b \geq 0$ für Vektoren: Es gilt $\neg(\Delta b \geq 0) \Leftrightarrow \exists i : \Delta b_i < 0$ (und **nicht** $\Delta b < 0$, wie wir es von rellen Zahlen gewohnt sind)

Beweis. "$\Rightarrow$": Es existiere $A \in \mathbb{R}^{m\times n}$ mit $|A - \bar{A}| \leq \Delta A$ und $b \in \mathbb{R}^m$ mit $|b - \bar{b}| \leq \Delta b$ sodass $A\bar{x} = b$. Dann gilt für das Residdum

$$|r(\bar{x})| = |\bar{A}\bar{x} - \bar{b}| = |\bar{A}\bar{x} - \bar{b} - (A\bar{x} - b)| = |(\bar{A} - A)\bar{x} - (\bar{b} - b)| \leq |\bar{A} - A||\bar{x}| + |\bar{b} - b| \leq \Delta A|\bar{x}| + \Delta b$$

"$\Leftarrow$": Es gelte $|r(\bar{x})| \leq \Delta A|\bar{x}| + \Delta b$. Wir setzen $r := r(\bar{x}) \in \mathbb{R}^m$, $s := \Delta A|\bar{x}| + \Delta b \in \mathbb{R}^m$ sowie für $i = 1,\dots,m$, $j = 1,\dots,n$

$$\delta a_{ij} := \begin{cases} -\frac{r_i}{s_i}\Delta a_{ij}\operatorname{sign}(\bar{x}_j) & \text{falls } s_i \neq 0 \\ 0 & \text{falls } s_i = 0, \end{cases} \qquad \delta b_i := \begin{cases} \frac{r_i}{s_i}\Delta b_i & \text{falls } s_i \neq 0 \\ 0 & \text{falls } s_i = 0. \end{cases}$$

Wegen $|r_i| \leq s_i$ folgt $|\delta a_{ij}| \leq \Delta a_{ij}$ und $|\delta b_i| \leq \Delta b_i$, $\forall i = 1,\dots,m$, $j = 1,\dots,n$. Seien nun $\delta A := (\delta a_{ij})$, $A := \bar{A} + \delta A$ und $\delta b := (\delta b_i)$, $b := \bar{b} + \delta b$. Dann gilt $|A - \bar{A}| = |\delta A| \leq \Delta A$ und $|b - \bar{b}| \leq |\delta b| \leq \Delta b$ sowie für beliebiges $i \in \{1,\dots,m\}$

$$(A\bar{x} - b)_i = \sum_{j=1}^{n} a_{ij}\bar{x}_j - b_i = \sum_{j=1}^{n}(\bar{a}_{ij} + \delta a_{ij})\bar{x}_j - (\bar{b}_i + \delta b_i) = r_i + \sum_{j=1}^{n}\delta a_{ij}\bar{x}_j - \delta b_i.$$

Ist nun $s_i = 0$, folgt $\delta a_{ij} = 0$, $\forall j = 1,\dots,n$ und $\delta b_i = 0$ sowie $r_i = 0$ wegen $|r_i| \leq s_i$. Dann ist aber auch $(A\bar{x} - b)_i = 0$. Ist dagegen $s_i \neq 0$, folgt ebenfalls

$$(A\bar{x} - b)_i = r_i + \sum_{j=1}^{n}(-\frac{r_i}{s_i})\Delta a_{ij}\operatorname{sign}(x_j)|x_j| - \frac{r_i}{s_i}\Delta b_i = r_i - \frac{r_i}{s_i}(\sum_{j=1}^{n}\Delta a_{ij}\bar{x}_j + \Delta b_i) = r_i - \frac{r_i}{s_i}s_i = 0.$$

Damit gilt $(A\bar{x} - b)_i = 0$, $\forall i = 1,\dots,m$ und daher $A\bar{x} = b$. □

Dieser Satz besitzt unter anderem folgende Interpretation: Auf Grund des unvermeidbaren Rundungsfehlers können wir unter Umständen ja gar nicht mit exakten Daten rechnen. Beim Rechnen am Computer stehen ja nur eine Matrix $\bar{A}$ und eine rechte Seite $\bar{b}$ zur Verfügung, sodass für die exakten Daten A und b gilt

$$|\bar{a}_{ij} - a_{ij}| \leq \text{eps}|\bar{a}_{ij}|,\ i,j = 1,\dots,n, \quad |\bar{b}_i - b_i| \leq \text{eps}|\bar{b}_i|,\ i = 1,\dots,n$$

Gilt also für die berechnete Lösung $\bar{x}$

$$|\bar{A}\bar{x} - \bar{b}| \leq \text{eps}(|\bar{A}||\bar{x}| + |b|), \tag{2.7}$$

so müssen wir $\bar{x}$ als Lösung akzeptieren, denn dann könnte es ja exakte Lösung eines Gleichungssystems $A\bar{x} = b$ sein, dessen Daten A und b bei Darstellung am Computer genau $\bar{A}$ und $\bar{b}$ ergeben. In der Praxis wird man die rechte Seite von (2.7) noch mit einem kleinen Vorfaktor (z.B. 10) versehen, um $\bar{x}$ als Lösung zu akzeptieren.

Akzeptieren wir $\bar{x}$ im Sinne der Rückwärtsanalyse, so kann durchaus ein großer (absoluter und/oder relativer) Fehler im Ergebnis auftreten! Man beachte dass in Satz 2.4 die Matrix $\bar{A}$ weder quadratisch noch invertierbar zu sein braucht.

2.4 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 ersten Schritt des Gaußschen Eliminationsverfahrens wird die Variable x_1 aus der i-ten Gleichung durch Subtraktion des $\frac{a_{i1}}{a_{11}}$-fachen der ersten Gleichung von der i-ten Gleichung eliminiert, $i = 2, \ldots, n$.

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 $a_{11} \neq 0$

$$\begin{aligned} u_{1j} &= a_{1j}, \quad j = 1, 2, \ldots, n, \\ l_{i1} &= \frac{a_{i1}}{u_{11}}, \quad i = 2, \ldots, n, \\ a_{ij}^{(1)} &= a_{ij} - l_{i1}\,u_{1j}, \quad i, j = 2, \ldots, n, \end{aligned} \tag{2.8}$$

und

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

Im zweiten Schritt des Gaußschen Eliminationsverfahrens wird die Variable x_2 aus der 3. bis zur n-ten Gleichung analog eliminiert, d.h. Subtraktion des $\frac{a_{i2}^{(1)}}{a_{22}^{(1)}}$-fachen des zweiten Gleichung. Es entsteht das neue Gleichungssystem

$$A^{(2)} x = b^{(2)} \tag{2.9}$$

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 weiteren $n-3$ Schritten erhält man schließlich ein Gleichungssystem der Form $Ux = c$ oder

$$\begin{aligned} u_{11}x_1 + u_{12}x_2 + \ldots + u_{1,n-1}x_{n-1} + u_{1,n}x_n &= c_1 \\ u_{22}x_2 + \ldots + u_{2,n-1}x_{n-1} + u_{2,n}x_n &= c_2 \\ \vdots &= \vdots \\ u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n &= c_{n-1} \\ u_{n,n}x_n &= c_n. \end{aligned} \tag{2.10}$$

Durch Rückwärtselimination lassen sich aus (2.10) die x_i bestimmen. Wir erhalten dann

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

Das Gaußsche Eliminationsverfahren ist in dieser Form genau dann **durchführbar**, wenn

$$u_{kk} \neq 0, \quad k = 1, 2, \ldots, n.$$

Diese Bedingung ist erfüllt, falls die Hauptminoren $M_k \neq 0$, $k = 1, 2, \ldots, n$.

Der Aufwand zur Berechnung von $A^{(k)}$ im k-ten Eliminationsschritt erfordert $(n-k)^2$ Operationen der Form $z = x - a \cdot y$, insgesamt also

$$\sum_{k=1}^{n-1}(n-k)^2 = \sum_{k=1}^{n-1} k^2 = \frac{n(2n-1)(n-1)}{6} = \frac{1}{3}n^3 + \mathcal{O}(n^2)$$

Operationen. Zur Berechnung von c und x aus (2.10) sind je $\sum_{k=1}^{n-1} k = \mathcal{O}(n^2)$ Operationen erforderlich.

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.

2.5 Dreieckszerlegungen

Sei $Ax = b$. Die einzelnen Gaußschen Eliminationsschritte lassen sich auch als Multiplikation von Matrizen L_k^{-1} von links ausdrücken. Der erste Eliminationsschritt entspricht der Operation

$$A^{(1)}x = L_1^{-1}Ax = L_1^{-1}b = b^{(1)}$$

mit

$$L_1^{-1}A = \begin{pmatrix} 1 & 0 & \ldots & & 0 \\ -l_{21} & 1 & 0 & \ldots & 0 \\ -l_{31} & 0 & 1 & 0 & \vdots \\ \ldots & \ldots & & \ddots & \\ -l_{n1} & 0 & \ldots & 0 & 1 \end{pmatrix} \begin{pmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \ldots & a_{3n} \\ \ldots & & & \ddots & \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nn} \end{pmatrix}$$

mit den l_{i1} aus (2.8). Im zweiten Schritt multiplizieren wir $A^{(1)}x = b^{(1)}$ mit der Matrix

$$L_2^{-1} = \begin{pmatrix} 1 & 0 & \ldots & & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & -l_{32} & 1 & 0 & \vdots \\ \ldots & \ldots & & \ddots & \\ 0 & -l_{n2} & \vdots & 0 & 1 \end{pmatrix}$$

um das System $A^{(2)}x = b^{(2)}$ zu erhalten. Man erhält schließlich

$$Ux = L_{n-1}^{-1} \cdots L_2^{-1} L_1^{-1} Ax = L_{n-1}^{-1} \cdots L_2^{-1} L_1^{-1} b = c$$

mit $L_{n-1}^{-1} \cdots L_2^{-1} L_1^{-1} A = U$. Dies ist äquivalent zu $A = LU$ mit $L = L_1 L_2 \cdots L_{n-1}$.

Was ist nun L? Aufgrund der Blockmatrixbeziehungen

$$\begin{pmatrix} I & \mathbf{0} \\ X & I \end{pmatrix}^{-1} = \begin{pmatrix} I & \mathbf{0} \\ -X & I \end{pmatrix} \quad \text{und}$$

$$\begin{pmatrix} I & \mathbf{0} \\ X & I \end{pmatrix} \begin{pmatrix} I & \mathbf{0} \\ \mathbf{0} & B \end{pmatrix} = \begin{pmatrix} I & \mathbf{0} \\ X & B \end{pmatrix}$$

folgt nun

$$L_1 = \begin{pmatrix} 1 & 0 & \dots & & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & 0 & 1 & 0 & \vdots \\ \dots & \dots & & \ddots & \\ l_{n1} & 0 & \dots & 0 & 1 \end{pmatrix}$$

usw. und damit

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

Damit enthalten L und U die durch den Gauß-Algorithmus berechneten Zahlen l_{ij} bzw. u_{ij}. U ist dabei eine obere und L eine untere Dreiecksmatrix und die Zerlegung $A = LU$ heißt LU-Zerlegung. Die Berechnung von x aus $Ax = LUx = b$ (insbedondere bei mehreren rechten Seiten) erfolgt nun nach dem folgenden Verfahren:

1. Berechne L und U mit $A = LU$.
2. Löse das System $Lc = b$ mit Vorwärtselimination.
3. Löse das System $Ux = c$ mittels Rückwärtselimination.

Der Rechenaufwand für die Schritte 2 und 3 beträgt $\mathcal{O}(n^2)$ Operationen, der Rechenaufwand für Schritt 1 jedoch $\mathcal{O}(n^3)$ Operationen.

2.6 Rundungsfehleranalyse für das Gaußsche Eliminationsverfahren

Es gilt der folgende Satz.

Satz 2.5 (Sautter)**.** *Sei $Ax = b$ und $\bar{x}$ die mit Hilfe des Gaußschen Eliminationsverfahrens berechnete Näherungslösung. $\bar{x}$ ist exakte Lösung des Gleichungssystems*

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

falls $n \cdot eps < 1/2$. $\bar{L}$ 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.

Im Sinne der Rückwärtsanalyse ist also das Gaußsche Eliminationsverfahren numerisch stabil, wenn die Einträge in der Störungsmatrix die gleiche Größenordnung wie der unvermeidbare Datenfehler eps$|A|$ besitzen.

Speziell in unserem Fall sollte also $|\bar{L}| \cdot |\bar{U}| \approx |A|$ gelten, d.h. die Matrizen $|\bar{L}|$ und $|\bar{U}|$ sollten nicht zu große Einträge haben.

Im k-ten Schritt des Gaußschen Eliminationsverfahren haben wir die Matrix

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

gegeben. (Im Falle $k = 0$ setzen wir $a_{ij}^{(0)} = a_{ij}$).

Betrachten wir zunächst die Matrix $\bar{L}$. Falls $|a_{ik}^{(k-1)}| \leq |a_{kk}^{(k-1)}|$ für $i = k+1, \ldots, n$, $k = 1, \ldots, n$, so folgt die Beziehung $|l_{ik}| \leq 1$ unmittelbar aus $l_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$. Im Allgemeinfall ist dies nicht erfüllt. Um die Bedingung $|l_{ij}| \leq 1$ zu garantieren, führt man die sogenannte Spaltenpivotsuche ein, d.h. wir suchen unter den Elementen $a_{jk}^{(k-1)}$, $j = k, \ldots, n$ das Element $a_{ik}^{(k-1)}$ mit

$$|a_{ik}^{(k-1)}| \geq |a_{jk}^{(k-1)}|, \quad j = k, \ldots, n.$$

Nun tauschen wir die i-te Zeile mit der k-ten Zeile in $A^{(k-1)}$ und wenden dann den üblichen Gaußeliminationsschritt an. Damit sind die Eliminationsfaktoren stets betragsmäßig nicht größer als 1.

Leider können die Einträge in $\bar{U}$ noch relativ stark anwachsen, wie z.B. bei Zerlegung der Matrix

$$\begin{pmatrix} 1 & 0 & \cdots & 0 & 1 \\ -1 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots \\ \vdots & \ddots & -1 & 1 & \vdots \\ -1 & \cdots & \cdots & -1 & 1 \end{pmatrix}.$$

Deshalb führt man manchmal auch eine *Totalpivotsuche* durch, d.h. unter allen Elementen $a_{ij}^{(k-1)}$, $i, j = k, \ldots, n$ wird das betragsgrößte bestimmt und durch maximal je einen Tausch von Zeilen und Spalten auf die Position (k, k) gebracht. Dadurch sind die Faktoren $|l_{ij}|$ wieder nicht größer als 1, allerdings wachsen nun die Einträge in $\bar{U}$ weniger stark an. Der Nachteil dieser Methode besteht darin, daß nun die Komponenten in x ebenfalls umgeordnet werden müssen.

Der Zeilentausch von i-ter mit k-ter Zeile kann auch als Multiplikation mit einer Permutationsmatrix P_k von links aufgefasst werden, wobei

$$P_k = \begin{pmatrix} 1 & & & & & & & & & \\ & \ddots & & & & & & & & \\ & & 1 & & & & & & & \\ & & & 0 & & & 1 & & & \\ & & & & 1 & & & & & \\ & & & & & \ddots & & & & \\ & & & & & & 1 & & & \\ & & & 1 & & & & 0 & & \\ & & & & & & & & 1 & \\ & & & & & & & & & \ddots \\ & & & & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \\ k \\ \\ \\ i \\ \\ \\ \end{matrix}$$

Wir betrachten nun diese einzelnen Zeilenvertauschungen: Im k-ten Iterationsschritt kennen wir eine Zerlegung der Form

$$P_{k-1} \dots P_1 A = L^{(k-1)} A^{(k-1)}$$

wobei

$$L^{(k-1)} = \left(\begin{array}{cccc|ccc} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & \ddots & \vdots & 0 & \cdots & 0 \\ \vdots & & \ddots & 0 & \vdots & & \vdots \\ l_{k-1,1} & l_{k-1,2} & \cdots & 1 & 0 & \cdots & 0 \\ \hline l_{k1}^{(k-1)} & l_{k2}^{(k-1)} & \cdots & l_{k,k-1}^{(k-1)} & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & & \ddots & \\ l_{n1}^{(k-1)} & l_{n2}^{(k-1)} & \cdots & l_{n,k-1}^{(k-1)} & 0 & \cdots & 1 \end{array}\right)$$

Der k-te Schritt besteht nun in einer Zeilenvertauschung (Multiplikation mit P_k) und der Elimination (Multiplikation mit L_k^{-1}), also

$$A^{(k)} = L_k^{-1} P_k A^{(k-1)} \;\Rightarrow\; P_k L^{(k-1)} P_k^T L_k A^{(k)} = P_k P_{k-1} \dots P_1 A$$

und daher

$$L^{(k)} = P_k L^{(k-1)} P_k^T L_k$$

Wie man sich leicht überzeugt, erhält man die Matrix $P_k L^{(k-1)} P_k^T$ dadurch, dass man im linken unteren Block von $L^{(k-1)}$ die i-te mit der k-ten Zeile vertauscht.

Fasst man alle Zeilenvertauschungen von Zeilen in der Matrix $P = P_{n-1} \cdots P_1$ zusammen, so löst man eigentlich das System

$$PAx = Pb,$$

d.h. wir berechnen die LU-Zerlegung von PA. Bei der totalen Pivotsuche löst man das System

$$PAQQ^T x = Pb,$$

d.h. wir berechnen die LU-Zerlegung von PAQ, wobei Q das Produkt aller Permutationsmatrizen des Spaltentauschs ist.

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, z.B. durch Multiplikation mit einer Diagonalmatrix D von links, sodass alle Zeilenbetragssummen 1 sind, d.h. $d_{ii} = 1/\sum_{j=1}^{n} |a_{ij}|$. Die Multiplikation mit D wird aber nicht tatsächlich ausgeführt sondern nur bei der Pivotsuche berücksichtigt: Es wird das Element $a_{rk}^{(k-1)}$ als Pivotelement gewählt sodass

$$|a_{rk}^{(k-1)}|d_{rr} = \max_{j \geq k} |a_{jk}^{(k-1)}|d_{jj}.$$

In dem folgenden Algorithmnen geben wir einen Pseudocode für LU-Zerlegung eine Matrix mit Spaltenpivotsuche und Lösen der gestaffelten System an:

Algorithmus 2.1 (LU-Zerlegung mit Spaltenpivotsuche)**.**
Input: $n \times n$ Matrix $A = (a_{ij})$
Output: LU-Zerlegung der Matrix PA, wobei Permutationsmatrix P durch vektor π gegeben ist (die i-te Zeile von PA entspricht der π_i-ten zeile von A, Flag $sing$ das angibt, ob A singulär ist. Die Matrix A wird dabei überschrieben.

```
{   for i = 1,...,n do π_i := i, s_i := Σ_{j=1}^{n} |a_ij|, if s_i == 0 then sing =true, return;
    sing :=false;
    for k = 1,...,n − 1 do
    {   piv := |a_kk|/s_{π_k}, ipiv := k
        for i = k + 1,...,n do if |a_ik|/s_{π_i} > piv then piv := |a_ik|/s_{π_i}, ipiv := i;
        if piv ≤ eps then sing :=true, return;
        if k ≠ ipiv then
        {   π_k ↔ π_ipiv;
            for j = 1,...,n do a_kj ↔ a_ipiv,j;
        }
        for i = k + 1,...,n do
        {   f := a_ik := a_ik/a_kk;
            for j = k + 1,...,n do a_ij := a_ij − f ∗ a_kj;
        }
    }
}
```

Algorithmus 2.2 (Lösen der gestaffelten Systeme)**.**
Input: LU-Zerlegung von Matrix $A = (a_{ij})$ und Permutationsvektor π als Output von Algorithmus 2.1, Vektor b
Output: Lösung x des Gleichungssystems $Ax = b$.

```
{   for i = 1,...,n do c_i := b_{π_i} − Σ_{j=1}^{i−1} a_ij c_j;
    for i = n,...,1 do x_i := (c_i − Σ_{j=i+1}^{n} a_ij x_j)/a_ii;
}
```

2.7 Orthogonalisierungsverfahren

Beim Gaußschen Verfahren haben wir das System $Ax = b$ mit einer unteren Dreiecksmatrix der Form

$$L^{-1} = \begin{pmatrix} 1 & 0\dots & & & 0 \\ l_{21} & 1 & 0 & \dots & 0 \\ l_{31} & l_{32} & 1 & \ddots & \vdots \\ \vdots & \vdots & & \ddots & 0 \\ l_{n1} & l_{n2} & \dots & l_{n,n-1} & 1 \end{pmatrix}^{-1}$$

multipliziert, um das gestaffelte System $Ux = c$ zu erhalten. Dadurch kann die Konditionszahl von U unter Umständen größer sein als die von A. Eine weitere Klasse von Matrizen, bei denen die Inverse leicht berechenbar ist, sind orthogonale Matrizen. Da wegen $Q^TQ = I$ stets $\| Q \|_2 = \| Q^{-1} \|_2 = \kappa_2(Q) = 1$ gilt, ändert die Multiplikation mit einer orthogonalen Matrix Q die Konditionszahl von A nicht, denn es gilt

$$\begin{aligned} \| QA \|_2^2 &= \lambda_{max}(A^TQ^TQA) = \lambda_{max}(A^TA) = \| A \|_2^2 \\ \| (QA)^{-1} \|_2^2 &= \| A^{-1}Q^T \|_2^2 = \lambda_{max}(QA^{-T}A^{-1}Q^T) \\ &= \lambda_{max}(A^{-T}A^{-1}Q^TQ) = \lambda_{max}(A^{-T}A^{-1}) = \| A^{-1} \|_2^2 . \end{aligned}$$

Damit sind orthogonale Transformationen stets stabil.

2.7.1 Das Householderverfahren

Es seien a_j, $j = 1, 2, \dots, n$ die Spalten der Matrix A und P_1 eine orthogonale Matrix. Dann gilt:

$$A = \left(a_1 \mid a_2 \mid \cdots \mid a_n\right)$$

und somit:

$$P_1A = \left(P_1a_1 \mid P_1a_2 \mid \cdots \mid P_1a_n\right).$$

In einem ersten Schritt ist nun eine orthogonale Matrix P_1 mit der Eigenschaft

$$P_1a_1 = \beta e_1,$$

gesucht, wobei e_1 den ersten Einheitsvektor bezeichnet:

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

Geometrisch lässt sich eine solche Transformation als Spiegelung von a_1 an einer geeignet zu wählenden Geraden angeben (Spiegelungsmatrix). In Formeln ausgedrückt, hat jede Spiegelungsmatrix die Form

$$P_i = I - \frac{2}{v_i^Tv_i}v_iv_i^T \quad (= P_i^T) \tag{2.11}$$

mit zu bestimmendem Vektor v_i. Im ersten Schritt folgt aus der Bedingung $P_1a_1 = \beta e_1$, daß

$$\beta e_1 = a_1 - \gamma v_1 \quad \text{mit } \gamma = 2\frac{v_1^Ta_1}{v_1^Tv_1}, \tag{2.12}$$

Wir können nun die Länge von v_1 so wählen, dass $\gamma = 1$, d.h. $v_1 = -\beta e_1 + a_1$. Wegen $\beta^2 = \|\beta e_1\|_2^2 = \|P_1 a_1\|_2^2 = \|a_1\|_2^2$ erhält man $\beta = -\text{sign}(a_{11})\|a_1\|_2$, d.h. wir wählen

$$v_1 = a_1 + \text{sign}(a_{11})\|a_1\|_2\, e_1.$$

(das Vorzeichen von β wird so gewählt dass bei der Berechnung der 1. Komponente von v_1 Auslöschung vermieden wird.) Hier ist sign definiert als

$$\text{sign}(x) := \begin{cases} +1 & \text{falls } x \geq 0 \\ -1 & \text{falls } x < 0. \end{cases}$$

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 wählen wir

$$v_2 = \begin{pmatrix} 0 \\ \tilde{a}_2 \end{pmatrix} + \text{sign}(\tilde{a}_{22}) \parallel \tilde{a}_2 \parallel_2 e_2.$$

usw. 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 (2.11) 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

$$Q = P_1^T P_2^T \cdots P_{n-1}^T = P_1 P_2 \cdots P_{n-1}. \tag{2.13}$$

Q ist ebenfalls eine orthogonale Matrix. Das 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. Das eben beschriebene Verfahren zur Bestimmung der Matrizen P_i heißt Householder-Verfahren.

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 (2.13) 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.

Für die Durchführung des Householderverfahrens muss A nicht unbedingt quadratisch sein. Ist z.B. A eine $m \times n$ Matrix, $m > n$, dann erhält man nach n Schritten eine Zerlegung der Form $A = QR$, wobei $Q = P_1^T P_2^T \cdots P_n^T$ eine orthogonale $m \times m$ Matrix und

$$R = \begin{pmatrix} \tilde{R} \\ \hline 0 \end{pmatrix} = \left(\begin{array}{ccc} r_{11} & \cdots & r_{1n} \\ 0 & \ddots & \vdots \\ 0 & 0 & r_{nn} \\ \hline 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{array}\right)$$

eine $m \times n$ rechte obere Dreiecks Matrix ist. Dies ist wichtig für sogenannte least squares Lösungen von überbestimmten Gleichungssystemen.

Definition 2.1. *Sei $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. $x \in \mathbb{R}^n$ heißt* least-squares Lösung *von $Ax = b$:⇔ x ist Lösung von*

$$\min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2 \tag{2.14}$$

Für jede orthogonale Matrix $Q \in \mathbb{R}^{m\times m}$ und jedes $x \in \mathbb{R}^n$ gilt nämlich $\|Ax - b\|_2 = \|Q^T(Ax-b)\|_2$. Daher ist x genau dann Lösung von (2.14), wenn x auch Lösung von $\min_x \|Q^T(Ax-b)\|_2^2$ ist. Letzteres Problem ist einfach zu lösen, wenn $A = QR$ gilt. Bezeichnen wir $c = Q^T b$, $e = (c_1, \ldots, c_n)^T$, $f = (c_{n+1}, \ldots, c_m)^T$, so folgt $\|Q^T(Ax - b)\|_2^2 = \|\tilde{R}x - e\|_2^2 + \|0x - f\|_2^2$. Eine least-squares Lösung ist also durch eine Lösung des Gleichungssystems $\tilde{R}x = e$ gegeben, wenn $\tilde{R}$ Vollrang besitzt.

Eine andere Berechnungsmöglichkeit ist durch die sogenannten *Normalgleichungen* gegeben: Es gilt, dass x genau dann least-squares Lösung von $Ax = b$ ist, wenn $A^T Ax = A^T b$. $A^T A$ ist stets positiv (semi)definit, man könnte also versuchen, diesen Gleichungssystem mittels Choleskyzerlegung (siehe unten) zu lösen. Es gilt aber $\kappa(A^T A) = \kappa(\tilde{R})^2$ und daher ist die Lösung über die Normalgleichungen i.a. numerisch instabil.

Für die praktische Durchführung des Householderverfahrens empfiehlt sich eine *Pivotsuche*, da damit festgestellt werden kann, ob eine Matrix (numerisch) singulär ist oder nicht.

Des weiteren kann damit auch der (Zeilen)-Rang zuverlässig abgeschätzt werden: Oft (z.B. in MATLAB) geschieht die Berechnung des Rangs über eine Berechnung der Singulärwerte. Dies ist allerdings relativ aufwändig und kann über eine orthogonale Faktorisierung mit Pivotsuche genauso gut erreicht werden

Algorithmus 2.3 (QR-Zerlegung mit Pivotsuche). schleife benötigt
Input: $m \times n$ Matrix $A = (a_{ij})$
Output: Abschätzung für den Rang k der Matrix sowie eine Zerlegung der Form $P_k \dots P_1 A P = R$, mit Householdermatrizen $P_j = I - \frac{2}{v_j^T v_j} v_j v_j^T$, $j = 1, \dots, k$, P Permutationsmatrix dargestellt durch Vektor π und R rechte obere Dreiecksmatrix

{ $\bar{n} = n$; for $j = 1, \dots, n$ do $\pi_j = j$; colon-notation
matlab funktion deal oder Dreieckstausch
for $j = n, \dots, 1$ do $\sigma_j := \bar{\sigma}_j := \sum_{i=1}^{m} a_{ij}^2$, if $\sigma_j == 0$ then $\pi_j \leftrightarrow \pi_{\bar{n}}$, $\bar{n}$--;
for $k = 1, \dots, \bar{n}$ do
{ $piv := k - 1$, $val := -1$;
for $j = k, \dots, \bar{n}$ do
if $\sigma_{\pi_j} / \bar{\sigma}_{\pi_j} > val$ then $val = \sigma_{\pi_j} / \bar{\sigma}_{\pi_j}$, $piv = j$;
finde das größte element si/siq
Matlab:
[M,I] = max(__)
[val,piv] = //gibt also maximum und Index zurück
max(si(pi(k:nq))./siq(pi(k:nq))) // colon notation k:nq
If $piv < k$ then return $k := k - 1$;
dot funktion
$\pi_k \leftrightarrow \pi_{piv}$, $\sigma_{\pi_k} := \sum_{i=k}^{m} a_{i\pi_k}^2$, if $\sigma_{\pi_k} < m\text{eps}^2 \bar{\sigma}_{\pi_k}$ then return $k := k - 1$;
$diag_k := -\text{sign}(a_{k\pi_k}) \sqrt{\sigma_{\pi_k}}$, $a_{k\pi_k} := a_{k\pi_k} + \text{sign}(a_{k\pi_k}) \sqrt{\sigma_{\pi_k}}$;
sign(0) = 1, matlab arbeitet aber mit sign(0) = 0!!!! entweder eigenes sign oder mit if
= - diag(k)
for $j = k + 1, \dots \bar{n}$ do
{ $\gamma := (\sum_{i=k}^{m} a_{i\pi_j} a_{i\pi_k}) / (-diag_k a_{k\pi_k})$, for $i = k \dots, m$ do $a_{i\pi_j} = a_{i\pi_j} - \gamma a_{i\pi_k}$;
$\sigma_{\pi_j} := \sigma_{\pi_j} - a_{k\pi_j}^2$;
skalarprodukt
kann man sich sparen durch:
a(k:n, pi(j)) = a(k:n, pi(j)) - gamma * a(k:n, pi(j))
if $\sigma_{\pi_j} < m\text{eps} \bar{\sigma}_{\pi_j}$ then $\sigma_{\pi_j} := \sum_{i=k+1}^{m} a_{i\pi_j}^2$;
summe mit colon notation
}
}
return $k := \bar{n}$;
}
matlab funktion sum kann man verwenden

In Algorithmus 2.3 wird die Matrix A überschrieben: die Spalte π_j, $1 \le j \le n$ von A beinhaltet die j-te Spalte von R (bis auf das Diagonalelement, das auf $diag_j$ gespeichert ist), sowie den Vektor v_j für $1 \le j \le k$:

$$a_{i\pi_j} = \begin{cases} r_{ij} & \text{falls } i < \min\{j, k+1\} \\ v_{ij} & \text{falls } i \ge j,\ j \le k \end{cases}$$

Die Elemente $a_{i\pi_j}$, $i > k$, $j > k$ sind alle "klein" und werden als Null betrachtet.

2.7.2 Givensrotationen

Seien $1 \leq k \neq l \leq n$ und $c, s \in \mathbb{R}$ mit $c^2 + s^2 = 1$. Dann heißt die Matrix

$$G^{kl} = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow k \\ \\ \leftarrow l \\ \\ \\ \end{matrix}$$
$$\begin{matrix} & & \uparrow & & \uparrow & & \\ & & k & & l & & \end{matrix}$$

Givensrotation (in der (k,l) Ebene). Die Matrix G^{kl} ist also bis auf die durch die k-te und l-te Zeile bzw. Spalte festgelegte 2×2 Teilmatrix gleich der Einheitsmatrix und

$$\begin{pmatrix} G^{kl}_{kk} & G^{kl}_{kl} \\ G^{kl}_{lk} & G^{kl}_{ll} \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

Ist $u \in \mathbb{R}^n$ ein Vektor, so kann eine Givensrotation dazu verwendet werden, um das l-te Element zu Null zu machen: Mit

$$a := u_k,\ b := u_l,\ (c,s) := \begin{cases} (\frac{a}{\sqrt{a^2+b^2}}, \frac{b}{\sqrt{a^2+b^2}}) & \text{falls } a^2 + b^2 \neq 0 \\ (1,0) & \text{sonst} \end{cases}$$

folgt $(G^{kl}u)_l = 0$.

Givensrotationen werden oft dazu verwendet, um speziell strukturierte Matrizen, z.B. dünnbesetzte Matrizen auf einfachere Gestalt zu bringen. Besitzt z.B. die Matrix A die Struktur

$$A = \begin{pmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & * & * & * & * & * \\ & * & & * & * & * \\ & * & & & * & * \\ & * & & & & * \end{pmatrix},$$

so können zuerst Givensrotationen G^{56}, G^{45}, und G^{34} angewandt werden, um in der 2.Spalte ab der 4.Zeile Nullen zu erzeugen. Damit erhält man folgende Struktur

$$G^{56}A = \begin{pmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & * & * & * & * & * \\ & * & & * & * & * \\ & * & & & * & * \\ & 0 & & & * & * \end{pmatrix},\ G^{45}G^{56}A = \begin{pmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & * & * & * & * & * \\ & * & & * & * & * \\ & 0 & & * & * & * \\ & 0 & & & * & * \end{pmatrix},$$

$$\tilde{A} := G^{34}G^{45}G^{56}A = \begin{pmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & * & * & * & * & * \\ & 0 & * & * & * & * \\ & 0 & & * & * & * \\ & 0 & & & * & * \end{pmatrix}.$$

Man beachte dass die Multiplikation von links mit der Givensrotation G^{kl} die k-te und l-te Zeile modifiziert: Durch Multiplikation mit G^{56} wird zwar das Element in Position $(6,2)$ eliminiert, aber dafür in Position $(6,5)$ ein neuer Eintrag erzeugt, usw. Anschließend werden durch Multiplikation von links mit Givensmatrizen $\tilde{G}^{23}, \tilde{G}^{34}, \tilde{G}^{45}, \tilde{G}^{56}$ die Subdiagonalelemente elimiert

$$\tilde{A} \underset{\tilde{G}^{23}}{\to} \begin{pmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & 0 & * & * & * & * \\ & & * & * & * & * \\ & & & * & * & * \\ & & & & * & * \end{pmatrix} \underset{\tilde{G}^{34}}{\to} \begin{pmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & 0 & * & * & * & * \\ & & 0 & * & * & * \\ & & & * & * & * \\ & & & & * & * \end{pmatrix} \underset{\tilde{G}^{45}}{\to} \cdots \underset{\tilde{G}^{56}}{\to} \begin{pmatrix} * & * & * & * & * & * \\ & * & * & * & * & * \\ & 0 & * & * & * & * \\ & & 0 & * & * & * \\ & & & 0 & * & * \\ & & & & 0 & * \end{pmatrix},$$

d.h. die Matrix $\tilde{G}^{56}\tilde{G}^{45}\tilde{G}^{34}\tilde{G}^{23}G^{34}G^{45}G^{56}A$ besitzt Dreiecksgestalt.

2.8 Spezielle Gleichungssysteme

In Anwendungsproblemen treten häufig lineare Gleichungssysteme $Ax = b$ mit Matrizen spezieller Struktur auf, die eine effizientere Lösung erlauben. Beispiele sind

- eine Bandmatrix A,
- eine symmetrisch positiv definite Matrix A,

siehe Abschnitt 2.1. Dann gibt es schnellere Verfahren als den einfachen Gauß-Algorithmus.

2.8.1 Dreieckszerlegungen für Bandmatrizen

Definition 2.2. *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:

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

Ohne Pivotsuche haben die L und U ebenfalls Bandstruktur:

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

Der Rechenaufwand unter Beachtung der Nullen in A, L und U beträgt dann nur noch ungefähr $p\,q\,n$ Operationen. 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).

Falls $p = q = 1$ gilt, spricht man von einer tridiagonalen Matrix. Dann betragen Rechenaufwand und Speicherplatzbedarf jeweils $\mathcal{O}(n)$.

2.8.2 Die Cholesky-Zerlegung für symmetrische positiv definite Matrizen

Für symmetrische positiv definite Matrizen A, also für Matrizen mit

$$A^T = A$$

und

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

kann der Rechenaufwand der Gaußelimination auf etwa die Hälfte reduziert werden.

Satz 2.6. *Sei A eine symmetrische, positiv definite $n \times n$ Matrix. Dann gibt es eine eindeutig bestimmte rechte obere Dreiecksmatrix R mit positiven Diagonalelementen, $r_{ii} > 0$, $i = 1, \ldots, n$, sodass*

$$A = R^T R \tag{2.15}$$

*(*Choleskyzerlegung*).*

Beweis. Da A symmetrisch ist, gibt es eine orthogonale Matrix V (gebildet aus den Eigenvektoren von A) und eine Diagonalmatrix Λ (Diagonalelemente sind Eigenwerte λ_i), sodass $V^T A V = \Lambda$ und daher $A = V \Lambda V^T$. Wir definieren nun $A^{\frac{1}{2}} := V \Lambda^{\frac{1}{2}} V^T$, wobei $\Lambda^{\frac{1}{2}}$ jene Diagonalmatrix ist, deren Diagonalelemente die (positive) Wurzel der Diagonalelemente von Λ ist, also $\Lambda = \Lambda^{\frac{1}{2}} \Lambda^{\frac{1}{2}}$. Wir können die Wurzel ziehen, da auf Grund der Positiv Definitheit die Eigenwerte alle positiv sind. $A^{\frac{1}{2}}$ ist offensichtlich symmetrisch. Für $A^{\frac{1}{2}}$ existiert aber eine orthogonale Faktorisierung der Form $A^{\frac{1}{2}} = QR$ mit Q orthogonal und $\tilde{R}$ rechte obere Dreiecksmatrix. Damit folgt

$$A = V \Lambda V^T = V \Lambda^{\frac{1}{2}} V^T V \Lambda^{\frac{1}{2}} V^T = A^{\frac{1}{2}} A^{\frac{1}{2}} = {A^{\frac{1}{2}}}^T A^{\frac{1}{2}} = \tilde{R}^T Q^T Q \tilde{R} = \tilde{R}^T \tilde{R}$$

Da A vollen Rang besitzt, muss auch $\tilde{R}$ vollen Rang besitzen, d.h. die Diagonalelemente sind alle ungleich 0. Sei nun $\tilde{I}$ Diagonalmatrix mit $\tilde{i}_{jj} = \operatorname{sgn} \tilde{r}_{jj}$ und $R := \tilde{I}\tilde{R}$. Dann ist offensichtlich R wiederum rechte obere Dreiecksmatrix mit positiven Diagonalelementen r_{ii} und $R^T R = \tilde{R}^T \tilde{I}^T \tilde{I} \tilde{R} = \tilde{R}^T \tilde{R} = A$. Es existiert also mindestens eine Choleskyzerlegung. Für die Matrix

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1n} \\ 0 & r_{22} & r_{23} & \dots & r_{2n} \\ 0 & 0 & r_{33} & \dots & r_{3n} \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \dots & 0 & r_{nn} \end{pmatrix}.$$

muss gelten

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & & & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} = \begin{pmatrix} r_{11} & 0 & 0 & \dots & 0 \\ r_{12} & r_{22} & 0 & \dots & 0 \\ r_{13} & r_{23} & r_{33} & 0 & \\ \vdots & & & \ddots & 0 \\ r_{1n} & r_{2n} & r_{3n} & \dots & r_{nn} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & \dots & r_{1n} \\ 0 & r_{22} & r_{23} & \dots & r_{2n} \\ 0 & 0 & r_{33} & \dots & r_{3n} \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \dots & 0 & r_{nn} \end{pmatrix}$$

d.h. wir erhalten die Formeln

$$\begin{aligned} a_{11} = r_{11}^2 &\Rightarrow r_{11} = \sqrt{a_{11}} \\ a_{1j} = a_{j1} = r_{11} r_{1j} &\Rightarrow r_{1j} = a_{1j}/r_{11}, \quad j = 2, \dots, n \\ a_{22} = r_{12}^2 + r_{22}^2 &\Rightarrow r_{22} = \sqrt{a_{22} - r_{12}^2} \\ a_{2j} = a_{j2} = r_{12} r_{1j} + r_{22} r_{2j} &\Rightarrow r_{2j} = \frac{a_{2j} - r_{12} r_{1j}}{r_{22}}, \quad j = 3, \dots, n \\ &\dots \\ a_{ii} = \sum_{k=1}^{i} r_{ki}^2 &\Rightarrow r_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} r_{ki}^2}, \\ a_{ij} = a_{ji} = \sum_{k=1}^{i} r_{ki} r_{kj} &\Rightarrow r_{ij} = (a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj})/r_{ii}, \quad j = i+1, \dots, n \end{aligned}$$

die eindeutig auflösbar sind. □

Mit diesen Betrachtungen haben wir bereits einen Algorithmus zur Berechnung von R gewonnen:

Algorithmus 2.4 (Choleskyzerlegung)**.**
Input: Symmetrische $n \times n$ Matrix $A = (a_{ij})$
Output: $R^T R$–Zerlegung von A mit $r_{ii} > 0$, falls A positiv definit, ansonsten Fehlermeldung

```
{   for i = 1,...,n do
    {   d_i := a_ii - sum_{k=1}^{i-1} r_ki^2;
        if d_i <= 0 then return A nicht positiv definit;
        r_ii := sqrt(d_i);
        for j = i+1,...,n do r_ij := (a_ij - sum_{k=1}^{i-1} r_ki r_kj)/r_ii;
    }
}
```

Der Rechenaufwand beträgt $\frac{1}{6}n^3 + \mathcal{O}(n^2)$ Operationen, der Speicherplatz in etwa $\frac{1}{2}n^2$. Die im vorherigen Abschnitt geschilderten Ideen für Bandmatrizen gelten auch für die Cholesky-Zerlegung.

Für die Matrix $F = |R^T| \cdot |R|$ gilt für $j \geq i$

$$f_{ij} = f_{ji} = \sum_{k=1}^{i} |r_{ki}||r_{kj}| \leq \sqrt{\sum_{k=1}^{i} r_{ki}^2} \sqrt{\sum_{k=1}^{i} r_{kj}^2} \leq \sqrt{a_{ii} a_{jj}},$$

die Choleskyzerlegung ist daher auch ohne Pivotsuche numerisch stabil.

2.9 Ergänzungen zu direkten Verfahren

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

2.9.1 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. So läßt sich durch die Lösung der Systeme

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

die Inverse von A in $\mathcal{O}(n^3)$ Operationen bestimmen.

2.9.2 Berechnung der Determinante einer Matrix

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

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

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

Bemerkung: Benützen Sie **niemals** die Abfrage $\det A == 0$ um festzustellen, ob eine Matrix Vollrang besitzt.

Kapitel 3

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 die zweidimensionale Laplacegleichung näher diskutiert.

3.1 Ein Beispiel

Die Modellierung eines stationären Wärmeleitproblems führt auf eine Laplacegleichung. Wir betrachten ein solches Problem im $\mathbb{R}^2$.

Der Einfachheit halber sei nun $\Omega = (0,1) \times (0,1)$ (das Einheitsquadrat in der Ebene), Γ bezeichnet den Rand der Menge Ω. Für eine vorgegebene Funktion $f : \Omega \mapsto \mathbb{R}$ ist eine Funktion $u : \overline{\Omega} \mapsto \mathbb{R}$ gesucht, die folgende Bedingungen erfüllt:

$$-u_{xx} - u_{yy} = f \quad \text{in } \Omega, \tag{3.1}$$

$$u = 0 \quad \text{auf } \Gamma = \partial\Omega. \tag{3.2}$$

Man spricht von einem Randwertproblem. Diskretisiert man (3.1) auf dem Gitter $\Omega_h = \{(x_i, y_j) | i, j = 1, 2, \ldots, N-1\}$ mit $h = 1/N$, $x_i = i \cdot h$, $y_j = j \cdot h$ 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

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

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

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

Dabei sind die Vektoren $\underline{u}_h$ und $\underline{f}_h$ in $N-1$ Blöcke von jeweils N Komponenten unterteilt. Die Matrix K_h besteht aus $(N-1)^2$ Teilmatrizen, die jeweils $(N-1) \times (N-1)$-Matrizen sind.

Das entstehende Gleichungssystem ist für große N sehr groß ($n = (N-1)^2$ Unbekannte und Gleichungen) und dünnbesetzt (maximal 5 Nichtnulleinträge pro Zeile). Als Bandmatrix

besitzt es die untere und obere Bandbreite $p = q = N - 1 = \sqrt{n}$.

3.1.1 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 $n_h = \mathcal{O}(1/h^d)$.

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:
$$n_h = \mathcal{O}\left(\frac{1}{h^d}\right).$$

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

3. Bei geeigneter Nummerierung der Knoten entsteht eine Bandmatrix mit Bandbreite
$$p_h = q_h = \mathcal{O}\left(\frac{1}{h^{d-1}}\right) = \mathcal{O}\left(n_h^{\frac{d-1}{d}}\right).$$

4. Zumindest bei der konventionellen Diskretisierung von Randwertproblemen 2. Ordnung gilt für die Konditionszahl unabhängig von der Raumdimension:
$$\kappa_2(K_h) = \mathcal{O}\left(\frac{1}{h^2}\right).$$

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

Anzahl der Operationen:
$$p_h q_h n_h = \mathcal{O}\left(\frac{1}{h^{d-1}}\frac{1}{h^{d-1}}\frac{1}{h^d}\right) = \mathcal{O}\left(\frac{1}{h^{3d-2}}\right) = \mathcal{O}\left(n_h^{\frac{3d-2}{d}}\right).$$

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.

3.2 Konstruktion von Iterationsverfahren

Viele Iterationsverfahren zur Lösung eines linearen Gleichungssystems

$$Ax = b \tag{3.3}$$

beruhen auf einer Umformung von (3.3) als Fixpunktgleichung der Form

$$x = Mx + Nb. \tag{3.4}$$

Daraus gewinnt man ein Iterationsverfahren (Fixpunktiteration), indem man eine bekannte Näherung $x^{(k)}$ der exakten Lösung x^* von (3.3) 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 bzw. Gesamtschrittverfahren** beruht auf folgenden Umformungen der i-ten Zeile des Gleichungssystems (3.3):

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

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

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j<i} a_{ij} x_j^{(k+1)} - \sum_{j>i} a_{ij} x_j^{(k)} \right), \quad i = 1, \ldots, n, \tag{3.5}$$

das man **Gauß–Seidel–Verfahren bzw. Einzelschrittverfahren** nennt. Mit den eben eingeführten Bezeichnungen lässt sich das Verfahren (3.5) 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

$$x^{(k+1)} = W^{-1}Rx^{(k)} + W^{-1}b = Mx^{(k)} + Nb \tag{3.6}$$

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 (3.6) 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

$$Ap^{(k,*)} = r^{(k)} \tag{3.7}$$

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 (3.7) die Matrix A durch eine Näherung W und erhält damit das Iterationsverfahren

$$x^{(k+1)} = x^{(k)} + p^{(k)} \quad \text{mit} \quad Wp^{(k)} = r^{(k)}. \tag{3.8}$$

Dieses Verfahren ist identisch mit (3.6).

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

$$x^{(k+1)} = x^{(k)} + \tau\, p^{(k)} \quad \text{mit} \quad Wp^{(k)} = r^{(k)} \tag{3.9}$$

oder kürzer

$$x^{(k+1)} = x^{(k)} + \tau\, W^{-1}(b - Ax^{(k)}). \tag{3.10}$$

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 (3.9) 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 (3.10).

3.3 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 (3.10) lässt sich auch folgendermaßen schreiben:

$$x^{(k+1)} = Mx^{(k)} + Nb \quad \text{mit} \quad M = I - \tau\, W^{-1}A,\ N = \tau\, W^{-1}. \tag{3.11}$$

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

$$x^* = x^* + \tau\, W^{-1}\,(b - Ax^*) = \left(I - \tau\, W^{-1}A\right) x^* + \tau\, W^{-1}b = Mx^* + Nb. \tag{3.12}$$

Durch Subtraktion von (3.11) von (3.12) folgt für den Fehler $e^{(l)} = x^{(l)} - x^*$:

$$e^{(k+1)} = Me^{(k)}. \tag{3.13}$$

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

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

Satz 3.1. *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) mit $C = \|e^{(0)}\|$.*
3. $\| e^{(k)} \| \le \frac{q^k}{1-q} \| x^{(1)} - x^{(0)} \|$

Beweis. Aus (3.13) folgt

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

Durch mehrmalige Anwendung dieser Ungleichung erhält man

$$\|e^{(k)}\| \le q\,\|e^{(k-1)}\| \le q^2\,\|e^{(k-2)}\| \le \cdots \le q^k\,\|e^{(0)}\| = C\,q^k. \tag{3.14}$$

Zum Beweis von 3.) nutzen wir die Dreiecksungleichung und die erste Behauptung

$$\begin{aligned} \| e^{(0)} \| = \| x^{(0)} - x^* \| &= \| x^{(0)} - x^{(1)} + x^{(1)} - x^* \| \\ &\le \| x^{(0)} - x^{(1)} \| + \| x^{(1)} - x^* \| \\ &\le \| x^{(0)} - x^{(1)} \| + q \| e^{(0)} \| . \end{aligned}$$

Umstellen liefert nun

$$\| e^{(0)} \| \leq \frac{1}{1-q} \| x^{(1)} - x^{(0)} \|, \tag{3.15}$$

also erhalten wir unter Verwendung von (3.14) und (3.15)

$$\| e^{(k)} \| \leq q^k \| e^{(0)} \| \leq \frac{q^k}{1-q} \| x^{(1)} - x^{(0)} \|,$$

was die gesuchte Behauptung ist. □

Nach diesem Satz ist die Bedingung $\|M\| < 1$ hinreichend für die Konvergenz des Iterationsverfahrens für alle Startwerte. Es gilt jedoch die folgende Verschärfung:

Satz 3.2. *Das Iterationsverfahren konvergiert für beliebige Startwerte $x^{(0)}$ genau dann, wenn*

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

wobei $\rho(M) = \max\{|\lambda| : \lambda$ ist Eigenwert von $M\}$ den Spektralradius von M bezeichnet. Unter dieser schwächeren Bedingung lässt sich nur mehr $\| e^{(k)} \| \leq Cq^k$ für beliebiges $q > \rho(M)$ zeigen.

Beweis. 1. Wir zeigen zuerst, dass für $\rho(M) \geq 1$ das Iterationsverfahren nicht für alle Startwerte konvergiert. Sei dazu λ ein (eventuell komplexer) Eigenwert von M mit $\rho(M) = |\lambda|$ und u ein zugehöriger Eigenvektor mit $\|u\|_\infty = 1$. Dann ist $M^k u = \lambda^k u$ und daher $\|M^k u\|_\infty = |\lambda|^k \|u\| \infty \geq \|u\|_\infty = 1$. Dann gibt es für jedes k einen Zeilenindex $i(k)$ mit

$$1 \leq \|M^k u\|_\infty = |\sum_{j=1^n} m^{(k)}_{i(k),j} u_j| \leq \sum_{j=1}^{n} |m^{(k)}_{i(k),j}| \|u\|_\infty,$$

wobei $m^{(k)}_{ij}$ das Element in der i-ten Zeile und j-ten Spalte von $M^(k)$ bezeichnet. Dann gibt es aber auch einen Index $j(k)$ mit $m^{(k)}_{i(k),j(k)} \geq \frac{1}{n}$. Nachdem es nur n^2 Paare von Zeilen und Spaltenindizes gibt, muss es ein Paar $(\bar{i}, \bar{j})$ geben, sodass für eine Folge (l_k) von natürlichen Zahlen mit $|m^{(l_k)}_{\bar{i}\bar{j}}| \geq \frac{1}{n}$. Wählen wir $x^{(0)} = x^* + e_{\bar{j}}$, also $e^{(0)} = e_{\bar{j}}$, folgt

$$\|e^{(l_k)}\|_\infty = \|M^k e_{\bar{j}}\|_\infty \geq |m^{(l_k)}_{\bar{i}\bar{j}}| \geq \frac{1}{n},$$

d.h., das Iterationsverfahren konvergiert nicht für diesen Startwert.

2. Die Konvergenz für beliebige Startwerte folgt aus dem nachfolgenden Satz 3.3 sowie Satz 3.1

□

Satz 3.3. *Sei M eine $n \times n$ Matrix. Dann gibt es für jedes $\epsilon > 0$ eine Vektornorm $\|\cdot\|$, sodass für die zugeordnete Operatornorm*

$$\|M\| \leq \rho(M) + \epsilon$$

gilt.

Beweis. Sei

$$TMT^{-1} = J$$

die Jordan'sche Normalform von M, d.h. T ist regulär (eventuell komplex) und J wird aus Diagonalblöcken der Form

$$C_\nu(\lambda_i) = \begin{pmatrix} \lambda_i & 1 & & 0 \\ & \ddots & \ddots & \\ & & & 1 \\ 0 & & & \lambda_i \end{pmatrix}$$

gebildet.

Für beliebiges $\epsilon > 0$ sei nun $J_\epsilon := D_\epsilon^{-1} J D_\epsilon$, wobei

$$D_\epsilon := \begin{pmatrix} 1 & & & \\ & \epsilon & & \\ & & \ddots & \\ & & & \epsilon^{n-1} \end{pmatrix}$$

Dadurch werden die Diagonalblöcke $C_\nu(\lambda_i)$ zu

$$\begin{pmatrix} \lambda_i & \epsilon & & 0 \\ & \ddots & \ddots & \\ & & & \epsilon \\ 0 & & & \lambda_i \end{pmatrix}$$

transformiert. Damit folgt

$$\|J_\epsilon\|_\infty = \|D_\epsilon^{-1} T M T^{-1} D_\epsilon\|_\infty = \rho(M) + \epsilon.$$

Sei nun $p(x) = \|Sx\|_\infty$ mit $S := D_\epsilon^{-1} T$. Wir zeigen, dass p eine Norm auf $\mathbb{C}^n$ ist. Aus $p(x) = 0$ folgt $Sx = 0$ und, da S regulär, $x = 0$. Für beliebiges $\alpha \in \mathbb{C}$ bzw. $x, y \in C^n$ folgt die Homogenität bzw die Dreiecksungleichung wegen

$$\begin{aligned} p(\alpha x) &= \|S(\alpha x)\|_\infty = \|\alpha S x\|_\infty = |\alpha| \|Sx\|_\infty = |\alpha| p(x), \\ p(x+y) &= \|Sx + Sy\|_\infty \leq \|Sx\|_\infty + \|Sy\|_\infty = p(x) + p(y) \end{aligned}$$

und die Normeigenschaft ist gezeigt. Für die zugeordnete Operatornorm ${}_p\|\cdot\|$ gilt

$${}_p\|M\| = \sup_{x \neq 0} \frac{{}_p\|Mx\|}{{}_p\|x\|} = \sup_{x \neq 0} \frac{\|SMx\|_\infty}{\|Sx\|_\infty} = \sup_{y \neq 0} \frac{\|SMS^{-1}y\|_\infty}{\|y\|_\infty} = \|J_\epsilon\|_\infty = \rho(M) + \epsilon,$$

wobei das Supremum über komplexwertige Vektoren gebildet wird. Bilden wir das Supremum nur über reellwertige Vektoren, so wird es sicher nicht größer. Damit folgt die Behauptung des Satzes. □

Aufwand eines Iterationsverfahrens:

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

Das Richardson-Verfahren ist durch

$$x^{(k+1)} = (I - \tau A)x^{(k)} + \tau b \tag{3.16}$$

gegeben, wobei $\tau > 0$. Damit ist also

$$M = I - \tau A, \quad W = \tau^{-1} I.$$

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

$$0 < \tau < \frac{2}{\lambda_{\max}(A)}. \tag{3.17}$$

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}:

$$\begin{aligned} q_{\text{opt}} &= q(\tau_{\text{opt}}) = 1 - \tau_{\text{opt}}\lambda_{\min}(A) = 1 - \frac{2\lambda_{\min}(A)}{\lambda_{\min}(A) + \lambda_{\max}(A)} \\ &= \frac{\lambda_{\max}(A) - \lambda_{\min}(A)}{\lambda_{\max}(A) + \lambda_{\min}(A)} = \frac{\lambda_{\max}(A)/\lambda_{\min}(A) - 1}{\lambda_{\max}(A)/\lambda_{\min}(A) + 1} = \frac{\kappa_2(A) - 1}{\kappa_2(A) + 1}. \end{aligned} \tag{3.18}$$

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

Es gilt mit $\ln(1 - x) \approx -x$ für kleine x

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

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

$$k = \mathcal{O}(\kappa(A)).$$

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

Beispiel: Für die diskutierten FE-Diskretisierungen erhält man wegen

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

eine Gesamtzahl von $\mathcal{O}(\kappa(K_h)\, n_h) = \mathcal{O}(1/h^{d+2}) = \mathcal{O}(n_h^{1+2/d})$ Operationen. Für $d = 2$ benötigt man also $\mathcal{O}(n_h^2)$ Operationen, das ist die gleiche Größenordnung des Aufwands wie beim Gaußschen Eliminationsverfahren. Für $d = 3$ ergeben sich $\mathcal{O}(n_h^{5/3})$ Operationen, also (asymptotisch) weniger Operationen als beim Gaußschen Eliminationsverfahren.

Ein Nachteil der Euklidischen Norm besteht darin, daß das Abbruchkriterium

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

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

Deshalb nutzt man oftmals andere Normen. Wir nehmen nun an, daß A symmetrisch und positiv definit ist und das Iterationsverfahren über

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

gegeben ist, d.h. $M = I - \tau W^{-1}A$, wobei die Matrix W ebenfalls symmetrisch und positiv definit ist. Dies ist z.B. beim Jacobi-Verfahren der Fall. Wir nutzen dann die energetische Norm

$$\|M\|_A^2 = \sup_x \frac{\|Mx\|_A^2}{\|x\|_A^2} = \sup_x \frac{(AMx, Mx)_2}{(Ax, x)_2}.$$

Zunächst gilt offenbar

$$M^T A = (I - \tau A W^{-1})A = A(I - \tau W^{-1}A) = AM. \tag{3.19}$$

Dann gilt mittels (3.19)

$$\begin{aligned} \|M\|_A^2 = \sup_x \frac{(AMx, Mx)_2}{(Ax, x)_2} &= \sup_x \frac{\langle M^T AMA^{-1/2}A^{1/2}x, A^{-1/2}A^{1/2}x\rangle_2}{\langle A^{1/2}x, A^{1/2}x\rangle_2} \\ &= \sup_y \frac{\langle M^T AMA^{-1/2}y, A^{-1/2}y\rangle_2}{\langle y, y\rangle_2} \\ &= \sup_y \frac{\langle AMMA^{-1/2}y, A^{-1/2}y\rangle_2}{\langle y, y\rangle_2} = \rho(A^{1/2}M^2A^{-1/2}). \end{aligned} \tag{3.20}$$

Die Matrix $F = A^{-1/2}M^T AMA^{-1/2} = A^{1/2}M^2A^{-1/2}$ ist offenbar symmetrisch und positiv definit. Damit besitzt F nur reelle Eigenwerte. Weiterhin ist

$$\lambda(A^{1/2}M^2A^{-1/2}) = \lambda(M^2A^{-1/2}A^{1/2}) \quad = \quad \lambda(M^2) = (\lambda(M))^2\,,$$

wobei $\lambda(M)$ einen Eigenwert der Matrix M bezeichnet. Aus (3.20) und (3.21) folgt nun

$$\| M \|_A = \rho(M). \tag{3.21}$$

Jedoch ist $\langle Ae^{(k)}, e^{(k)}\rangle_2$ immer noch nicht berechenbar. Deshalb betrachten wir nun die durch das Skalarprodukt

$$\langle x, x\rangle_{A^T W^{-1}A} = \langle W^{-1}Ax, Ax\rangle_2$$

induzierte Norm. Nun gilt

$$\begin{aligned} \langle e^{(k)}, e^{(k)}\rangle_{A^T W^{-1}A} &= \langle W^{-1}A(x^{(k)} - x^*), A(x^{(k)} - x^*)\rangle_2 \\ &= \langle W^{-1}(Ax^{(k)} - b), Ax^{(k)} - b\rangle_2 \\ &= \langle W^{-1}r^{(k)}, r^{(k)}\rangle_2 = \langle w^{(k)}, r^{(k)}\rangle_2, \end{aligned} \tag{3.22}$$

mit $w^{(k)} = W^{-1}r^{(k)}$, d.h. in dieser Norm ist der Fehler berechenbar. Desweiteren ist

$$A^T W^{-1}A = AW^{-1}A = WW^{-1}AW^{-1}A = Wp_2(W^{-1}A)$$

mit dem Polynom 2. Grades $p_2(y) = y^2$. Analog zum Beweis von (3.21) gilt nun mit (3.19) und $(W^{-1}A)M = M(W^{-1}A)$

$$\| M \|_{AW^{-1}A} = \rho(M). \tag{3.23}$$

Zusammenfassend haben wir nun gezeigt.

Satz 3.4. *Sei*

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

das vorkonditionierte Richardson-Verfahren mit symmetrisch positiv definiten Matrizen W und A. Weiterhin sei $M = I - \tau W^{-1}A$ der Fehlerübergangsoperator. Dann gilt

1. $|||x|||_s^2 = \langle W p_s(W^{-1}A)x, x\rangle_2$ *mit* $p_s(y) = y^s$ *definiert eine Norm mit* $|||M|||_s = \rho(M)$. *Speziell gilt*

$$\begin{aligned} |||e|||_0 &= \langle We, e\rangle_2 \\ |||e|||_1 &= \langle Ae, e\rangle_2 \\ |||e|||_2 &= \langle W^{-1}Ae, Ae\rangle_2 \end{aligned}$$

2. *Das Verfahren konvergiert für*

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

3. *Für die (optimale) Parameterwahl*

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

gelten die Fehlerabschätzungen

$$|||e^{(k+1)}|||_s \leq q_{opt}\,|||e^{(k)}|||_s \quad \textit{und} \quad |||e^{(k)}|||_s \leq \frac{q_{opt}^k}{1-q_{opt}}\,|||x^{(1)} - x^{(0)}|||_s$$

mit

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

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

$$|||e^{(k)}|||_2^2 = (W^{-1}r^{(k)}, r^{(k)}) = (w^{(k)}, r^{(k)}).$$

Beweis. 1. Die Definitheit der Norm folgt aus der positiven Definitheit von W und A. Die Aussage $|||M|||_s = \rho(M)$ kann wie in (3.21) für alle $s \in \mathbb{N}_0$ bewiesen werden.

2. Es folgt dann $q = \rho(M)$ für den Konvergenzfaktor. Die Matrix $M = I - \tau W^{-1}A$ besitzt die gleichen Eigenwerte wie die symmetrische Matrix $I - \tau W^{-1/2}AW^{-1/2}$, d.h. alle Eigenwerte sind reell. Desweiteren gelten alle Abschätzungen aus Satz 3.1 mit dem eben berechneten q, d.h. es folgt dann die Konvergenz für

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

aus (3.17).

3. Die Behauptung folgt nun mit (3.18) für $\tau = \tau_{\text{opt}}$.

4. Beziehung (3.22).

□

Damit ist es zweckmäßig, die Matrix W so zu wählen, daß die Konditionszahl von $W^{-1}A$ klein ist, d.h. es müssen die Beziehungen

$$c_1\langle Wu, u\rangle_2 \leq \langle Au, u\rangle_2 \leq c_2\langle Wu, u\rangle_2$$

mit Konstanten c_1 und c_2 gezeigt werden. Dann ist $\kappa(W^{-1}A) = \frac{c_2}{c_1}$. Die Matrix W heißt der Vorkonditionierer von A.

Die konkrete Wahl hängt dabei von der konkreten Problemklasse ab.

Bemerkung: Die beschriebenen Verfahren (Jacobi, Gauß-Seidel) werden heute nur noch als sogenannte Glätter in komplexeren Verfahren (Multigrid) genutzt. Im Falle einer symmetrisch positiven Systemmatrix A wird meistens das Verfahren des konjugierten Gradienten bevorzugt. Dann läßt sich zeigen, daß die Iterationszahl sich wie $\sqrt{\kappa(A)}$ verhält, was zu einer erheblich geringeren Iterationszahl bei gleicher relativer Genauigkeit führt, siehe Vorlesungen Numerik partieller DGL bzw. Optimierung. Im Falle unsymmetrischer Systeme gibt es z.B. GMRES, BiCG Stab.