Kapitel 2

Numerische Lösung linearer Gleichungssysteme

Dieses Kapitel behandelt numerische Verfahren zur Lösung linearer Gleichungssysteme der Gestalt

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{n\times n},\ \mathbf{x}, \mathbf{b} \in \mathbb{R}^n \tag{2.1}$$

mit

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}.$$

Viele Methoden zur Lösung von Problemen erfordern im Kern die Lösung linearer Systeme der Form (2.1). Daher sind effiziente und zuverlässige Verfahren zur numerischen Lösung von (2.1) von großer Wichtigkeit.

In der Praxis unterscheidet man folgende Typen linearen Systemen:

1.) die Dimension n ist klein, $n \lesssim 10000$,
2.) die Dimension n ist groß, $10000 \lesssim n \lesssim 10^8$, und die Elemente von A sind überwiegend Null (A ist schwach besetzt),
3.) die Dimension von A ist groß und A ist nicht schwach besetzt.

In der Vorlesung werden vorwiegend Verfahren für Matrizen vom Typ 1 behandelt. Im letzten Abschnitt werden auch einige Verfahren kurz vorgestellt, die man prinzipiell für alle Typen verwenden kann.

2.1 Theorie

In diesem Abschnitt werden wichtige Eigenschaften von Matrizen und Vektoren kurz zusammengefasst und einige Begriffe eingeführt.

Aus der linearen Algebra ist folgender Sachverhalt bekannt.

Satz 2.1 *Sei $A \in \mathbb{R}^{n\times n}$. Die folgenden Aussagen sind äquivalent:*

- *A ist eine reguläre Matrix, das heißt der Rang von A ist $rg(A) = n$,*
- *alle Eigenwerte von A sind ungleich Null,*
- *das System (2.1) besitzt genau eine Lösung,*
- *das System (2.1) mit der rechten Seite $\mathbf{b} = \mathbf{0}$ besitzt nur die triviale Lösung $\mathbf{x} = \mathbf{0}$,*
- *die Determinante von A ist ungleich Null,* $\det(A) \neq 0$,
- *die Matrix A ist invertierbar.*

Damit das Problem (2.1) korrekt gestellt ist, muss die Lösung eindeutig sein. Also muss A regulär sein. Außerdem gilt für eine reguläre Matrix A und eine Störung $\delta\mathbf{b}$

$$A\mathbf{x} = \mathbf{b} + \delta\mathbf{b} \quad \Longleftrightarrow \quad \mathbf{x} = A^{-1}\mathbf{b} + A^{-1}(\delta\mathbf{b}).$$

Da die Matrizenmultiplikation stetig ist, folgt

$$\lim_{\delta\mathbf{b}\to\mathbf{0}} A^{-1}(\delta\mathbf{b}) = A^{-1}\mathbf{0} = \mathbf{0}.$$

Damit hängt die Lösung stetig von den Störungen der rechten Seite ab. Außerdem kann man zeigen, dass hinreichend kleine Störungen der Matrixkoeffizienten verändern nicht die Regularität der Matrix. Das wird am Ende dieses Abschnitts noch genauer angegeben. Es gilt, dass die Lösung auch stetig von Störungen der Matrixkoeffizienten abhängt. Damit ist gezeigt, dass Problem (2.1) genau dann korrekt gestellt ist, wenn A regulär ist.

Für lineare Gleichungssysteme (2.1) mit regulärer Matrix A soll auch der Begriff der Kondition des Problems genauer besprochen werden. Dazu benötigt man Vektor– und Matrixnormen.

Vektornormen. $\mathbf{x} \in \mathbb{R}^n$, l^p–Norm, $p \in [1, \infty)$:

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

$p = 1$ – Summennorm, $p = 2$ – Euklidische Norm; für $p = \infty$ hat man die Maximumnorm

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

Matrixnormen. $A \in \mathbb{R}^{m\times n}$. Die induzierte Matrixnorm ist gegeben durch

$$\|A\|_p := \max_{\mathbf{x}\in\mathbb{R}^n, \mathbf{x}\neq\mathbf{0}} \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p} = \max_{\|\mathbf{x}\|_p=1} \|A\mathbf{x}\|_p .$$

Man findet

$$\begin{aligned}
\|A\|_1 &= \max_{j=1,\ldots,n} \sum_{i=1}^{m} |a_{ij}| \quad \text{Spaltensummennorm,} \\
\|A\|_2 &= \sqrt{\lambda_{\max}\left(A^T A\right)} \quad \text{Spektralnorm,} \\
\|A\|_\infty &= \max_{i=1,\ldots,m} \sum_{j=1}^{n} |a_{ij}| \quad \text{Zeilensummennorm.}
\end{aligned}$$

λ – Eigenwert. Außerdem

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 \right)^{1/2} \quad \text{Frobenius–Norm.}$$

Eine Vektornorm und eine Matrixnorm nennt man verträglich, falls für alle Matrizen und Vektoren gilt

$$\|A\mathbf{x}\|_{\text{vektor}} \leq \|A\|_{\text{matrix}} \|\mathbf{x}\|_{\text{vektor}} .$$

Man kann zeigen, dass eine l^p–Vektornorm und ihre induzierte Matrixnorm verträglich sind. *Übungsaufgabe*

Definition 2.2 Die Konditionszahl einer regulären Matrix $A \in \mathbb{R}^{n\times n}$ ist definiert als

$$\kappa(A) = \|A\| \left\|A^{-1}\right\|,$$

wobei $\|\cdot\|$ eine der oben angegebenen Matrixnormen ist. □

Unterschiedliche Normen ergeben unterschiedliche Konditionszahlen. Diese werden durch einen Index unterschieden.

Eigenschaften der Konditionszahl.

- $\kappa(A) \geq 1$, da (I – Einheitsmatrix)

$$1 = \|I\| = \left\|AA^{-1}\right\| \leq \|A\| \left\|A^{-1}\right\| = \kappa(A),$$

- $\kappa(A^{-1}) = \kappa(A)$,
- sei $\alpha \in \mathbb{R} \setminus \{0\}$

$$\kappa(\alpha A) = \|\alpha A\| \left\|(\alpha A)^{-1}\right\| = |\alpha| \left|\alpha^{-1}\right| \|A\| \left\|A^{-1}\right\| = \kappa(A),$$

- ist A eine orthogonale Matrix, dass heißt $A^T = A^{-1}$, dann ist $\kappa_2(A) = 1$, da

$$\|A\|_2 = \left\|A^T\right\|_2 = \left\|A^{-1}\right\|_2$$

und

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)} = \sqrt{\lambda_{\max}(I)} = 1,$$

$\kappa_2(\cdot)$ – Spektralkonditionszahl.

- ist A symmetrisch und positiv definit, das heißt alle Eigenwerte von A sind positiv, dann gilt *Übungsaufgabe*

$$\kappa_2(A) = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)},$$

Definiert man den relativen Abstand einer regulären Matrix A zur Menge der singulären Matrizen wie folgt

$$\operatorname{dist}_p(A) := \min\left\{ \frac{\|\delta A\|_p}{\|A\|_p} \ : \ A + \delta A \text{ ist singulär} \right\},$$

so kann man zeigen, dass

$$\operatorname{dist}_p(A) = \frac{1}{\kappa_p(A)}.$$

Dass bedeutet, dass bei Matrizen mit einer großen Konditionszahl schon kleine Störungen dazu führen können, dass die gestörte Matrix singular ist. Des weiteren ist die Matrix $A + \delta A$ auf jeden Fall regulär, falls

$$\frac{\|\delta A\|_p}{\|A\|_p} < \frac{1}{\kappa_p(A)} \quad \Longleftrightarrow \quad \|\delta A\|_p \left\|A^{-1}\right\|_p < 1.$$

Auf Grund der allgegenwärtigen Rundungsfehler wird ein numerisches Verfahren zur Lösung von (2.1) nur eine Näherungslösung berechen, die jedoch die Lösung eines gestörten linearen Systems ist

$$(A + \delta A)(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}.$$

In der Literutur findet man Abschätzungen für den relativen Fehler $\|\delta\mathbf{x}\| \,/\, \|\mathbf{x}\|$ mit Hilfe der Konditionszahl der Matrix A, siehe [QSS04].

2.2 Numerische Lösung linearer Systeme mit Dreiecksmatrix

Die Matrix $A \in \mathbb{R}^{n\times n}$ habe eine der beiden Formen

$$L = \begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & 0 & \\ \dots & \dots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix}, \quad U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & 0 & \ddots & \vdots \\ & & & u_{nn} \end{pmatrix},$$

L – untere Dreiecksmatrix (lower), U – obere Dreiecksmatrix (upper).

Da A regulär sein soll, folgt $l_{ii} \neq 0$ beziehungsweise $u_{ii} \neq 0$ für $i = 1, \dots, n$. Die Komponenten des Lösungsvektors können in diesen Fällen nacheinander berechnet werden, zum Beispiel für $L\mathbf{x} = \mathbf{b}$:

$$\begin{aligned} x_1 &= b_1/l_{11}, \\ x_2 &= (b_2 - l_{21}x_1)/l_{22}, \\ &\vdots \\ x_i &= \frac{1}{l_{ii}}\left(b_i - \sum_{j=1}^{i-1} l_{ij}x_j\right), \quad i = 1, \dots, n. \end{aligned}$$

Diese Vorgehensweise wird auch in dieser Form implementiert. *Übungsaufgabe* Die Durchführung dieses Verfahrens erfordert $n(n+1)/2$ Multiplikationen/Divisionen und $n(n-1)/2$ Additionen/Subtraktionen. *Übungsaufgabe* Die Gesamtzahl der benötigten Flops ist somit n^2.

Man beachte, dass man nach der Berechnung von x_i den Wert b_i der rechten Seite nicht mehr benötigt. Deshalb kann man die Lösungskomponente x_i sofort auf dem Speicherplatz von b_i speichern. Man benötigt also keinen zusätzlichen Speicherplatz für die Lösung.

Für $L\mathbf{x} = \mathbf{b}$ ist die Reihenfolge der berechneten Komponenten $x_1 \to x_2 \to \cdots \to x_n$ und man spricht von Vorwärtssubstitution. Löst man $U\mathbf{x} = \mathbf{b}$, so erhält man $x_n \to x_{n-1} \to \cdots \to x_1$ und dies wird Rückwärtssubstitution genannt.

Benötigt man den Vektor $\mathbf{b}$ nicht mehr, so kann man im Verfahren die berechneten Komponenten x_i auf dem Platz von b_i speichern.

Führt man eine Rundungsfehleranalyse für die Vorwärts–(Rückwärts–)substitution durch, erhält man folgendes Ergebnis, [KS88, S. 146f.]: Die mit der Vorwärtssubstitution berechnete Lösung $\mathbf{x}$ genügt der Gleichung

$$(L + \delta L)\mathbf{x} = \mathbf{b}$$

mit einer unteren Dreiecksmatrix δL, die komponentenweise klein ist im Sinne von

$$|(\delta L)_{ij}| \leq \varepsilon_M j\, |l_{ij}|, \quad i, j = 1, \dots, n,\ i \geq j,$$

wobei ε_M das Maschinenepsilon (siehe Übungen) ist. Man beachte, dass nur die Matrix L mit durch Rundungsfehler hervorgerufene Störungen behaftet ist. Die Aussage bedeutet, dass die Vorwärts–(Rückwärts–)substitution numerisch gutartige (gut konditionierte) Verfahren sind. Diese sollen im folgenden für die Lösung eines allgemeinen linearen Gleichungssystems der Form (2.1) genutzt werden.

2.3 Das Gauß–Verfahren und die LU–Zerlegung

Das bekannte Gauß–Verfahren ist bei richtiger Anwendung ein stabiles und effizientes Verfahren zur Lösung kleiner linearer Gleichungssysteme.

Vorgehen. Man formt das System $A\mathbf{x} = \mathbf{b}$ in ein äquivalentes System $U\mathbf{x} = \tilde{\mathbf{b}}$ um, wobei U eine obere Dreiecksmatrix ist, welches sich durch Rückwärtssubstitution leicht lösen lässt. Bei den Umformungen nutzt man aus, dass sich die Lösung des Systems $A\mathbf{x} = \mathbf{b}$ nicht ändert, wenn

- ein Vielfaches einer Gleichung zu einer anderen Gleichung addiert wird,
- zwei Gleichungen vertauscht werden,
- eine Gleichung mit einer rellen Zahl $\neq 0$ multipliziert wird.

Man hat also zwei Phasen beim Lösen von $A\mathbf{x} = \mathbf{b}$, zuerst die Erzeugung des Dreieckssystems (Vorwärtselimination) und dann die Rückwärtssubstitution.

1. Vorwärtselimination. (Erzeugung von $U\mathbf{x} = \tilde{\mathbf{b}}$)

- Falls $a_{11} \neq 0$, dann eliminiert man x_1 aus den letzten $(n-1)$ Gleichungen, indem man von der i–ten Gleichung das

$$m_{i1} = \frac{a_{i1}}{a_{11}} \; - \; \text{fache}$$

der 1. Gleichung subtrahiert, $i = 2, \ldots, n$, subtrahiert. Dann erhält man das modifizierte System

$$\left(\begin{array}{c|ccc} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ \hline 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{array}\right) \left(\begin{array}{c} x_1 \\ \hline x_2 \\ \vdots \\ x_n \end{array}\right) = \left(\begin{array}{c} b_1^{(1)} \\ \hline b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{array}\right)$$

mit

$$\begin{aligned} a_{ij}^{(1)} &= a_{ij}, \quad b_i^{(1)} = b_i, \\ a_{ij}^{(2)} &= a_{ij}^{(1)} - m_{i1} a_{1j}^{(1)}, \quad b_i^{(2)} = b_i^{(1)} - m_{i1} b_1^{(1)}. \end{aligned}$$

Das System ohne erste Zeile und Spalte wird mit $A^{(2)}\mathbf{x}^{(2)} = \mathbf{b}^{(2)}$ bezeichnet.

- Falls $a_{22}^{(2)} \neq 0$, so wendet man das gleiche Vorgehen auf $A^{(2)}\mathbf{x}^{(2)} = \mathbf{b}^{(2)}$ an und eliminiert x_2. Dieses Vorgehen wird fortgesetzt, falls $a_{ii}^{(i)} \neq 0$, $i = 3, \ldots n-1$. Als Ergebnis erhält man ein oberes Dreieckssystem

$$\left(\begin{array}{ccccc} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1,n-1}^{(1)} & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2,n-1}^{(2)} & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{n-1,n-1}^{(n-1)} & a_{n-1,n}^{(n-1)} \\ 0 & 0 & \cdots & 0 & a_{nn}^{(n)} \end{array}\right) \left(\begin{array}{c} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{array}\right) = \left(\begin{array}{c} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_{n-1}^{(n-1)} \\ b_n^{(n)} \end{array}\right). \quad (2.2)$$

2. Rückwärtssubstitution. Die Lösung $\mathbf{x}$ wird nun durch Rückwärtssubstitution aus (2.2) berechnet.

Dieses Vorgehen funktioniert, solange die Elemente $a_{ii}^{(i)} \neq 0$, $i = 1, \ldots, n$, sind. Das ist aber für reguläre Matrizen nicht selbstverständlich, zum Beispiel bei

$$A = \left(\begin{array}{cc} 0 & 1 \\ 1 & 1 \end{array}\right)$$

ist $a_{11}^{(1)} = 0$.

Sei beim k–ten Schritt das Element $a_{kk}^{(k)} = 0$. Eines der Elemente der restlichen Spalte $a_{ik}^{(k)}$, $i = k+1, \ldots, n$, muss dann ungleich Null sein, etwa $a_{k'k}^{(k)}$. Ansonsten

wäre die k–te Spalte von den vorhergehenden Spalten linear abhängig und A wäre damit singulär. Man vertausche die Zeilen k und k', auch in der rechten Seite, und setzte die Vorwärtselimination fort. Mit dieser Vertauschung vertauscht man zwei Gleichungen des linearen Systems.

Das Diagonalenelement $a_{kk}^{(k)}$, welches man zur Definition der Multiplikatoren

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \tag{2.3}$$

verwendet, und welches man eventuell erst durch eine Zeilenvertauschung erhält, nennt man Pivotelement. Zur Fehlerdämpfung stellt es sich als günstig heraus, wenn die Multiplikatoren (2.3) vom Betrage her möglichst klein sind. Das bedeutet, $\left|a_{kk}^{(k)}\right|$ sollte möglichst groß sein. Die gebräuchlichste Strategie um betragsmäßig kleine Multiplikatoren zu bekommen ist die sogenannte Spaltenpivotsuche: Suche das betragsmäßig größte Element in der restlichen k–ten Spalte

$$\left|a_{k'k}^{(k)}\right| \geq \left|a_{ik}^{(k)}\right|, \quad i \geq k.$$

Dann tauscht man die Zeilen k und k', auch in $\mathbf{b}$. Es gibt auch Pivotstrategien, bei denen man Spalten vertauscht.

Bei dieser Pivotstrategie gilt $|m_{ik}| \leq 1$. Erhält man zum Schluss der Vorwärtselimination auch mit Pivotsuche kleine Pivotelemente, so ist das ein Hinweis darauf, dass die Matrix A schlecht konditioniert ist.

Das Gaußsche Verfahren soll nun so formuliert werden, dass man die Matrix A überspeichern kann, ohne sie damit zu verlieren. Man wird sie indirekt behalten. Sei $n = 2$. Wir definieren mit $m_{21} = a_{21}/a_{11}$

$$L = \begin{pmatrix} 1 & 0 \\ m_{21} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} \\ 0 & a_{22}^{(2)} \end{pmatrix},$$

mit $a_{22}^{(2)} = a_{22}^{(1)} - m_{21}a_{12}^{(1)}$. Lässt man der Einfachheit halber die oberen Indizes weg sofern sie (1) sind, so erhält man

$$LU = \begin{pmatrix} a_{11} & a_{12} \\ m_{21}a_{11} & m_{21}a_{12} + a_{22}^{(2)} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & m_{21}/a_{12} + a_{22} - m_{21}/a_{12} \end{pmatrix} = A.$$

Diese Beobachtung lässt sich für $A \in \mathbb{R}^{n\times n}$ verallgemeinern. Seien

$$L = \begin{pmatrix} 1 & & & \\ m_{21} & 1 & 0 & \\ \vdots & & \ddots & \\ m_{n1} & \cdots & m_{n,n-1} & 1 \end{pmatrix}, \quad U = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n)} \end{pmatrix},$$

dann gilt

$$PA = LU, \tag{2.4}$$

wobei P eine Permutationsmatrix ist, die die Zeilen– (und eventuellen Spalten–) vertauschungen beschreibt, welche bei der Vorwärtselimination erfolgen. Man braucht diese Matrix nicht zu speichern, falls alle erforderlichen Vertauschungen in der rechten Seite gleich ausgeführt werden $\mathbf{b} \to P\mathbf{b}$. Beschränkt man sich auf Zeilenvertauschungen, so braucht man für die Beschreibung der Vertauschungen nur einen Vektor zu speichern. Hat man keine Vertauschungen, so gilt $A = LU$. Die Zerlegung (2.4) nennt man LU–Zerlegung oder LR–Zerlegung von A.

Bei der allgemeinen numerischen Herangehensweise zur Lösung von (2.1) entkoppelt man die Vorwärtselimination der Matrix A und die Anwendung dieser Elimination auf die rechte Seite $\mathbf{b}$. Somit hat man drei Schritte:

1. berechne die LU–Zerlegung von A: $PA = LU$,
2. berechne $\mathbf{y}$ aus
$$PA\mathbf{x} = L(U\mathbf{x}) = L\mathbf{y} = P\mathbf{b}$$
durch Vorwärtssubstitution,
3. berechne $\mathbf{x}$ aus
$$U\mathbf{x} = \mathbf{y}$$
durch Rückwärtssubstitution.

Wenn die Matrix A nicht mehr explizit benötigt wird, können L und U sofort auf dem Speicherplatz von A gespeichert werden. Man beachte, dass man die Diagonale von L nicht zu speichern braucht, da die Einträge $(= 1)$ sowieso bekannt sind. Mit Hilfe von L, U und der Permutationsmatrix P kann man jedoch gegebenenfalls A wieder rekonstruieren. Braucht man die rechte Seite $\mathbf{b}$ nicht mehr, so kann dieser Speicherplatz sofort für $\mathbf{y}$ und dann für die Lösung $\mathbf{x}$ verwendet werden. Man kann das Gauß–Verfahren also so implementieren, dass man keinen zusätzlichen Speicher benötigt für die LU–Zerlegung von A und für die Lösung benötigt.

Die Kosten zur Lösung von $A\mathbf{x} = \mathbf{b}$ sind wie folgt

1. $\frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{7}{6}n$ Flops, *Übungsaufgabe*
2. n^2 Flops,
3. n^2 Flops,

woraus Gesamtkosten von
$$\frac{2}{3}n^3 + \frac{3}{2}n^2 - \frac{7}{6}n \text{ Flops} \approx \frac{2}{3}n^3 \text{ Flops}$$
resultieren.

Man kann nicht mathematisch beweisen, dass die LU–Zerlegung mit Spaltenpivotsuche stabil ist. Im Gegenteil, man kann (pathologische) Beispiele konstruieren, bei denen eine Fehlerschranke, die man beweisen kann, mit wachsender Dimension n des Gleichungssystems exponentiell anwächst. In den allermeisten praktischen Fällen ist dies jedoch nicht der Fall. Auf Grund dieser Erfahrungen wird die Gauß–Elimination mit Spaltenpivotsuche als praktisch stabil angesehen und im allgemeinen verwendet.

Die LU–Zerlegung ohne Pivotsuche ist jedoch im allgemeinen instabil. Es gibt allerdings Klassen von Matrizen, für die man zeigen kann, dass die LU–Zerlegung auch ohne Pivotsuche stabil ist, beispielsweise symmetrische und positiv definite Matrizen. Dann kann man sich die Pivotsuche sparen und spart damit Rechenzeit.

Analysiert man das Gesamtverfahren 1)–3) zur Lösung des linearen Gleichungssystems (2.1), so erhält man folgende Aussagen:

Satz 2.3 *Löst man das lineare Gleichungssystem $A\mathbf{x} = \mathbf{b}$ mittels Gauß–Elimination mit Spaltenpivotsuche, so ist die berechnete Lösung $\mathbf{x}$ die Lösung des gestörten Systems*
$$(A + \delta A)\mathbf{x} = \mathbf{b}$$
und für die Fehlermatrix δA gilt die Abschätzung
$$\|\delta A\|_\infty \leq (\varepsilon_M n^3 + 3n^2)\rho\varepsilon_M \|A\|_\infty ,$$
wobei ε_M das Maschinenepsilon und
$$\rho = \max_{i,j,k} \left(\frac{\left|a_{ij}^{(k)}\right|}{\|A\|_\infty} \right)$$
sind.

Sei $\mathbf{x}^*$ *die Lösung von* $A\mathbf{x} = \mathbf{b}$*, dann gilt für den Fehler*

$$\|\mathbf{x}^* - \mathbf{x}\|_\infty \leq \kappa_\infty(A)(\varepsilon_M n^3 + 3n^2)\rho\varepsilon_M \|\mathbf{x}\|_\infty .$$

Die Abschätzung von $\|\delta A\|_\infty$ im Satz stellt für große n im allgemeinen eine starke Überschätzung dar. In der Praxis ist $\|\delta A\|_\infty$ kaum größer als $\varepsilon_M n \|A\|_\infty$. Ebenso ist die Abschätzung für den Fehler im allgemeinen zu pessimistisch. Man erkennt jedoch den Einfluss der Konditionszahl von A.

Bemerkung 2.4

1. Hat man mehrere Systeme mit der Matrix A und unterschiedlichen rechten Seiten $\mathbf{b}_1, \ldots, \mathbf{b}_l$ zu lösen, so führt man zuerst die LU–Zerlegung wie oben beschrieben durch ($\approx 2n^3/3$ Flops). Man speichert L, U und P. Danach wendet man Vorwärts– und Rückwärtssubstitution für alle rechten Seiten an (je $2n^2$ Flops). Sind alle rechten Seiten direkt nach der LU–Zerlegung bekannt, kann man die Substitutionen für diese rechten Seiten sogar gleichzeitig durchführen. Ist dies nicht der Fall so braucht man ab der zweiten rechten Seite die teure LU–Zerlegung nicht mehr zu berechnen.
2. Die inverse Matrix erhält man, falls man die in Bemerkung 2.4.1 beschriebene Vorgehensweise mit $\mathbf{b}_j = \mathbf{e}_j$ (j–ter Einheitsvektor), $j = 1, \ldots, n$, durchführt. Die Berechnung der inversen Matrix ist jedoch praktisch fast nie von Interesse. Hat man in einer Aufgabenstellung „$A^{-1}\mathbf{b}$" zu berechnen, so heißt das „löse das System $A\mathbf{x} = \mathbf{b}$". Die Matrix A^{-1} wird hierbei nicht gebraucht.
3. Sind die Matrizen L und U auf dem Speicherplatz von A gespeichert und braucht man den Wert des Produkts $\mathbf{x} = A\mathbf{y}$, so erhält man diesen durch

   $$\mathbf{z} = U\mathbf{y}, \quad \mathbf{w} = L\mathbf{z}, \quad \mathbf{x} = P^{-1}\mathbf{w}.$$

   Bei Spaltenpivotsuche ist P^{-1} nur eine Vertauschung der Komponenten. Die Berechnung des Produkts auf diesem Weg benötigt die gleiche Anzahl an Flops wie die Berechnung von $A\mathbf{y}$ bei abgespeicherter Matrix A.
4. Die Determinante der Matrix A erhält man direkt aus ihrer LU–Zerlegung. Für Permutationsmatrizen gilt $\det(P) = 1$. Damit hat man

   $$\det(A) = \det(PA) = \det(LU) = \det(L)\det(U) = \det(U) = \prod_{i=1}^{n} a_{ii}^{(i)}.$$

   Die Determinanten der Dreiecksmatrizen L und U ist jeweils das Produkt der Hauptdiagonalelemente.
5. Ist A eine symmetrische und (positiv) definite Matrix, so verwendet man eine Variante des Gauß–Verfahrens, bei welcher man eine Zerlegung der Gestalt

   $$A = C^T C, \quad C - \text{ obere Dreiecksmatrix},$$

   erhält, das sogenannte Cholesky–Verfahren. Wenn A symmetrisch ist, reicht es, das obere Dreieck von A zu speichern. Bei der normalen LU–Zerlegung müsste man jedoch sowohl L als auch U speichern, also eine volle quadratische Matrix. Beim Cholesky–Verfahren ist dies nicht nötig und man kann C auf dem Speicherplatz von A speichern.
6. In MATLAB steckt das Gaußsche Verfahren mit Spaltenpivotsuche hinter dem Befehl `x = A\b`. (*Demo)*

□

2.4 Klassische Iterationsverfahren zur Lösung linearer Gleichungssysteme

Bei der Diskretisierung partieller Differentialgleichungen, die zum Beispiel physikalische Prozesse beschreiben, muss man letztlich oft große lineare Systeme mit schwach besetzten Matrizen lösen. Dazu sind direkte Verfahren, wie das Gauß–Verfahren, im allgemeinen ungeeignet. Zum einen ist die Rechenzeit zu lang. Ist beispielsweise $n = 10^6$ und hat man einen Computer mit einem Gigaflop (10^9 Flops/s), dann braucht man mit dem Gauß–Verfahren zur Lösung des linearen Systems ungefähr 21.1 Jahre. Wendet man das Gauß–Verfahren auf eine schwach besetzte Matrix an, dann muss man im allgemeinen damit rechnen, dass die Faktoren L und U nicht mehr schwach besetzt sind. Deshalb ist der Speicherbedarf im allgemeinen n^2. Für $n = 10^6$ wären das etwa 7450 Gigabyte (mit 8 Byte double Zahlen).

Für große Systeme nutzt man iterative Verfahren. Dabei kann die bekannte Idee der Fixpunktiteration verwendet werden. Dazu wird die Matrix $A \in \mathbb{R}^{n\times n}$ in die Form

$$A = M - N, \quad M, N \in \mathbb{R}^{n\times n}$$

zerlegt, wobei M invertierbar sein soll. Aus (2.1) erhält man dann die äquivalente Fixpunktgleichung

$$M\mathbf{x} = \mathbf{b} + N\mathbf{x} \quad \Longleftrightarrow \quad \mathbf{x} = M^{-1}\left(\mathbf{b} + N\mathbf{x}\right).$$

Ist eine Startnäherung $\mathbf{x}^{(0)} \in \mathbb{R}^n$ gegeben, so lautet die Fixpunktiteration zur Lösung von $A\mathbf{x} = \mathbf{b}$:

$$\mathbf{x}^{(k+1)} = M^{-1}\left(\mathbf{b} + N\mathbf{x}^{(k)}\right), \quad k = 0, 1, 2, \ldots \tag{2.5}$$

Über die Konvergenz dieser Iteration gibt der Banachsche Fixpunktsatz, siehe später Satz 3.4, Auskunft. Für den Spezialfall (2.5) erhält man:

Satz 2.5 *Das iterative Verfahren (2.5) konvergiert genau dann für alle $\mathbf{x}^{(0)} \in \mathbb{R}^n$ gegen die Lösung von $A\mathbf{x} = \mathbf{b}$, wenn für die zugehörige Iterationsmatrix $M^{-1}N$ gilt*

$$\max\left\{|\lambda| \ : \ \lambda \text{ ist Eigenwert von } M^{-1}N\right\} < 1.$$

In jedem Iterationsschritt von (2.5) muss man ein lineares Gleichungssystem der Gestalt

$$M\mathbf{y} = \mathbf{b} + N\mathbf{x}^{(k)}$$

lösen. Das soll natürlich einfach gehen. Im einfachsten Fall wählt man

$$M = \omega^{-1}D = \omega^{-1}\mathrm{diag}(A),$$

wobei $\mathrm{diag}(A)$ die Diagonalmatrix mit den Diagonalelementen von A ist und $\omega > 0$ ein Parameter (Dämpfungsfaktor). Man muss bei diesem Fall natürlich voraussetzen, dass alle Diagonalenelemente von A ungleich Null sind. Es folgt

$$N = M - A = \omega^{-1}D - A$$

und man erhält die Iteration

$$\begin{aligned}\mathbf{x}^{(k+1)} &= M^{-1}\left(\mathbf{b} + N\mathbf{x}^{(k)}\right) = \omega D^{-1}\left(\mathbf{b} + \omega^{-1}D\mathbf{x}^{(k)} - A\mathbf{x}^{(k)}\right)\\ &= \mathbf{x}^{(k)} + \omega D^{-1}\left(\mathbf{b} - A\mathbf{x}^{(k)}\right).\end{aligned}$$

Dieses Verfahren wird für $\omega = 1$ Jacobi–Verfahren und für $\omega \in (0, 1)$ gedämpftes Jacobi–Verfahren genannt.

Eine andere Möglichkeit besteht darin, M als Dreiecksmatrix zu wählen. Zerlege

$$A = L + D + U,$$

wobei D die Diagonalmatrix mit den Diagonalelementen von A ist, L das strikte untere Dreieck von A und U das strikte obere Dreieck. Man muss wieder voraussetzen, dass alle Diagonalelemente von A ungleich Null sind. Wählt man

$$M = L + \omega^{-1} D,$$

so ergibt sich folgendes Iterationsverfahren

$$\begin{aligned}
\left(L + \omega^{-1} D\right) \mathbf{x}^{(k+1)} &= \mathbf{b} + \left(L + \omega^{-1} D\right) \mathbf{x}^{(k)} - (L + D + U)\, \mathbf{x}^{(k)} \quad \Longleftrightarrow \\
\omega^{-1} D \mathbf{x}^{(k+1)} &= \mathbf{b} + \omega^{-1} D \mathbf{x}^{(k)} - (D + U)\, \mathbf{x}^{(k)} - L \mathbf{x}^{(k+1)} \quad \Longleftrightarrow \\
\mathbf{x}^{(k+1)} &= \mathbf{x}^{(k)} + \omega D^{-1} \left(\mathbf{b} - L \mathbf{x}^{(k+1)} - (D + U)\, \mathbf{x}^{(k)} \right). \qquad (2.6)
\end{aligned}$$

Schreibt man dieses Verfahren in Komponentenschreibweise, so sieht man, dass man die rechte Seite berechnen kann, obwohl sich darin die neue Iterierte $\mathbf{x}^{(k+1)}$ befindet:

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

Man nutzt die neu berechneten Komponenten von $\mathbf{x}^{(k+1)}$ sofort zur Berechnung weiterer Komponenten. Dieses Verfahren heißt für $\omega = 1$ Gauß–Seidel–Verfahren und für $\omega > 0, \omega \neq 1$, spricht man vom SOR–Verfahren (successive overrelaxation, aufeinander folgende Entspannung). (*MATLAB–Demo*)

Es gibt viele Untersuchungen zur Konvergenz dieser Iterationsverfahren. Ein bemerkenswertes Ergebnis ist das folgende:

Satz 2.6 *Sei $A \in \mathbb{R}^{n \times n}$ symmetrisch und positiv definit. Dann konvergiert das SOR–Verfahren für alle $\mathbf{x}^{(0)} \in \mathbb{R}^n$ genau dann, wenn $\omega \in (0, 2)$.*

Bemerkung 2.7

1. Der Satz gibt keine Aussage über die Konvergenzgeschwindigkeit und über die Existenz oder Wahl eines optimalen Parameters ω. Es gibt Fälle, in denen man zeigen kann, dass ein optimaler Parameter ω existiert. Die Berechnung dieses Parameters erfordert jedoch im allgemeinen Informationen über die Matrix A, deren Beschaffung mit großem Aufwand verbunden ist.
2. Es zeigt sich, dass die Lösung großer Gleichungssysteme mit den hier vorgestellten Iterationsverfahren zwar im Prinzip geht, jedoch im allgemeinen sehr ineffizient ist (sehr viele Iterationen, sehr lange Rechenzeiten). Man hat inzwischen wesentlich bessere Iterationsverfahren zur Lösung linearer Systeme konstruiert. Die hier vorgestellten einfachen Iterationsverfahren können jedoch eine wichtige Teilkomponente komplizierterer Iterationsverfahren sein.

□