

$$\begin{bmatrix} 1 & -2 & -1 & 2 \\ 0 & \mathbf{4} & 3 & -2 \\ 0 & 4 & 6 & -3 \\ 0 & 8 & 2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2 \\ 9 \\ 11 \\ 18 \end{bmatrix} \quad \begin{array}{l} \text{subtrahiere 1x Reihe 2 von Reihe 3} \\ \text{subtrahiere 2x Reihe 2 von Reihe 4} \end{array} \quad (2.2)$$

$$\begin{bmatrix} 1 & -2 & -1 & 2 \\ 0 & 4 & 3 & -2 \\ 0 & 0 & \mathbf{3} & -1 \\ 0 & 0 & -4 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2 \\ 9 \\ 2 \\ 0 \end{bmatrix} \quad \text{subtrahiere } -(4/3)\text{x Reihe 3 von Reihe 4} \quad (2.3)$$

$$\begin{bmatrix} 1 & -2 & -1 & 2 \\ 0 & 4 & 3 & -2 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & \mathbf{8/3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2 \\ 9 \\ 2 \\ 8/3 \end{bmatrix} \quad (2.4)$$

Die linke obere Zahl einer Teilmatrix heißt Pivotelement. Die durch die Umformungen (2.1) - (2.4) entstandene Matrix heißt obere Dreiecksmatrix \mathbf{U} . Das zugehörige obere Dreieckssystem $\mathbf{Ux} = \mathbf{g}$ wird mit Rückwärtssubstitution gelöst.

$$\begin{aligned} x_4 &= 1 \\ x_3 &= \frac{2 - (-1)1}{3} = 1 \\ x_2 &= \frac{9 - (-2)1 - (3)1}{4} = 2 \\ x_1 &= \frac{-2 - (2)1 - (-1)1 - (-2)2}{1} = 1 \end{aligned} \quad (2.5)$$

Aufwandsabschätzung

Zur Aufwandaufschätzung werden die Anzahl der Multiplikationen und Divisionen gezählt.

Für die Lösung eines Gleichungssystem mit N unbekanntem ergibt sich für die Gaußelimination:

1 Triangulation

$$\begin{aligned}
 (N+1)(N-1) + N(N-2) + \dots + (3)(1) &= \sum_{k=1}^{N-1} k(k+2) \quad (2.6) \\
 &= \sum_{k=1}^{N-1} k^2 + 2 \sum_{k=1}^{N-1} k = \frac{(N-1)N(2N-1)}{6} + N(N-1) \\
 &= \frac{N(N-1)(2N+5)}{6}
 \end{aligned}$$

2 Rückwärtssubstitution

$$1 + 2 + \dots + N = \frac{N(N+1)}{2} \quad (2.7)$$

Die Gesamtzahl der Operationen ergibt sich aus (2.6) und (2.7) zu

$$\frac{N(N-1)(2N+5)}{6} + \frac{N(N+1)}{2} = \frac{N^3}{3} + N^3 - \frac{N}{3} \quad (2.8)$$

N	Anzahl der Operationen
10	430
100	343 430
1000	336 333 430

Für große N dominierent der Term ist $N^3/3$. Der Algorithmus ist von der $O(N^3)$. Die erforderliche Rechenzeit ist proportional N^3 .

Hat man ein Programm mit einem voerst unbekannte Algorithmus so kann man die Abhängigkeit der Rechenzeit von N folgenermassen abschätzen.

- 1** Man führt voerst die Rechnung für ein kleines $N = N_1$ aus. Die dafür benötigte Rechenzeit ist $t = t_1(N_1)$.
- 2** Die Rechnung wird dann für $N_2 \gg N_1$ wiederholt wozu die Zeit $t = t_2(N_2)$ gebraucht wird.
- 3** Aus dem Ansatz $T = aN^b$ und den beiden Zeiten kann man den exponenten b abschätzen.

$$\frac{T_1}{T_2} = \frac{aN_1^b}{aN_2^b} = \left(\frac{N_1}{N_2}\right)^b \quad (2.9)$$

$$b = \frac{\log(N_1/N_2)}{\log(T_1/T_2)} \quad (2.10)$$

LU Zerlegung

Schreibt man die Schritte zur Lösung des Gleichungssystems $\mathbf{Ax} = \mathbf{f}$ aus dem vorangehenden Beispiel noch einmal in Matrixschreibweise an so ergibt sich

$$\begin{array}{c} \mathbf{I} \\ \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right] \end{array} \begin{array}{c} \mathbf{A} \\ \left[\begin{array}{ccc} 1 & -2 & -1 & 2 \\ 2 & 0 & 1 & 2 \\ 2 & 0 & 4 & 1 \\ 1 & 6 & 1 & 2 \end{array} \right] \end{array} \begin{array}{c} \mathbf{x} \\ \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \right] \end{array} = \begin{array}{c} \mathbf{I} \\ \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right] \end{array} \begin{array}{c} \mathbf{f} \\ \left[\begin{array}{c} -2 \\ -5 \\ 7 \\ 16 \end{array} \right] \end{array} \quad (2.11)$$

$$\begin{array}{c} \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array} \right] \end{array} \begin{array}{c} \left[\begin{array}{ccc} 1 & -2 & -1 & 2 \\ 0 & 4 & 3 & -2 \\ 0 & 4 & 6 & -3 \\ 0 & 8 & 2 & 0 \end{array} \right] \end{array} \begin{array}{c} \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \right] \end{array} = \begin{array}{c} \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{array} \right] \end{array} \begin{array}{c} \left[\begin{array}{c} -2 \\ 9 \\ 11 \\ 18 \end{array} \right] \end{array} \quad (2.12)$$

$$\begin{array}{c} \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{array} \right] \end{array} \begin{array}{c} \left[\begin{array}{ccc} 1 & -2 & -1 & 2 \\ 0 & 4 & 3 & -2 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & -4 & 4 \end{array} \right] \end{array} \begin{array}{c} \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \right] \end{array} = \begin{array}{c} \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{array} \right] \end{array} \begin{array}{c} \left[\begin{array}{c} -2 \\ 9 \\ 2 \\ 0 \end{array} \right] \end{array} \quad (2.13)$$

$$\begin{array}{c} \mathbf{L} \\ \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 1 & 2 & -4/3 & 1 \end{array} \right] \end{array} \begin{array}{c} \mathbf{U} \\ \left[\begin{array}{ccc} 1 & -2 & -1 & 2 \\ 0 & 4 & 3 & -2 \\ 0 & 0 & 3 & -1 \\ 0 & 0 & 0 & 8/3 \end{array} \right] \end{array} \begin{array}{c} \mathbf{x} \\ \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array} \right] \end{array} = \begin{array}{c} \mathbf{L} \\ \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 1 & 2 & -4/3 & 1 \end{array} \right] \end{array} \begin{array}{c} \mathbf{g} \\ \left[\begin{array}{c} -2 \\ 9 \\ 2 \\ 8/3 \end{array} \right] \end{array} \quad (2.14)$$

Man erkennt, dass bei der Gaußelimination die Matrix \mathbf{A} in eine linke untere und eine rechte obere Dreiecksmatrix zerlegt wird

$$\mathbf{A} = \mathbf{LU}. \quad (2.15)$$

Die Einträge in \mathbf{L} sind die Multiplikationsfaktoren, die bei der Gaußelimination verwendet wurden. Es gilt

$$\mathbf{LUx} = \mathbf{Lg}, \quad (2.16)$$

wobei \mathbf{g} die Lösung des Systems

$$\mathbf{Lg} = \mathbf{f} \quad (2.17)$$

ist. Da \mathbf{L} nicht singular ist gilt auch

$$\mathbf{Ux} = \mathbf{g}. \quad (2.18)$$

Ist das Gleichungssystem $\mathbf{Ax}^{(l)} = \mathbf{f}^{(l)}$ bei fester Koeffizientenmatrix \mathbf{A} für mehrere rechte Seiten $\mathbf{f}^{(l)}$, mit $l = 1, 2, \dots, M$, zu lösen geht man wie folgt vor

- 1 Bestimme die LU Zerlegung der Matrix \mathbf{A}
- 2 Löse das System $\mathbf{Lg}^{(l)} = \mathbf{f}^{(l)}$
- 3 Löse das System $\mathbf{Ux}^{(l)} = \mathbf{g}^{(l)}$
- 4 Gehe zu Schritt 2 für die nächste rechte Seite ($l = l + 1$)

Die LU Zerlegung muß also nur einmal durchgeführt werden.

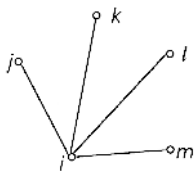
Cholesky Zerlegung. Im Falle einer symmetrischen, positiv definiten Matrix gilt $\mathbf{U} = \mathbf{L}^T$. Man erhält dann die Zerlegung

$$\mathbf{A} = \mathbf{LL}^T. \quad (2.19)$$

Bei symmetrischen, positiv definiten Matrizen halbiert sich der Speicheraufwand die Zahl der Rechenoperationen zur LU Zerlegung.

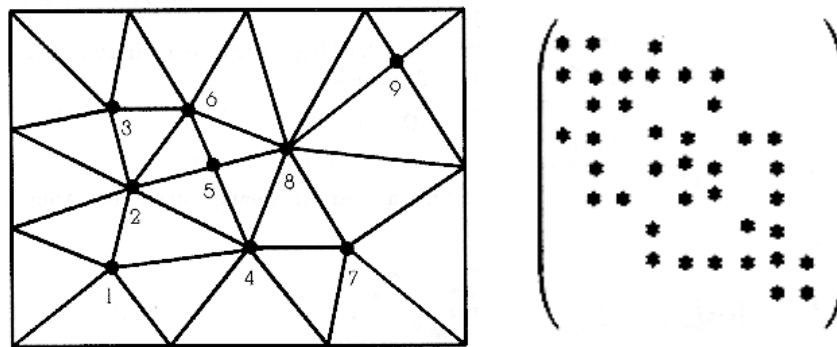
Dünn besetzte Matrizen

Die Gleichungssysteme die aus der Diskretisierung einer paritellen Differentialgleichung mit der Differenzen-, Finiten Volumen - oder finiten Elementmethode resultiert besitzt eine Koeffizientenmatrix mit spezieller Struktur. Ein Großteil der Einträge ist null, da nicht-Null-Einträge $a_{ij} \neq 0$ nur dann entstehen, wenn im Gitter der Knoten i mit dem Knoten j in Verbindung steht. Man spricht von sparse matrices. Für große Gitter besteht zwischen den meisten Knoten keine direkte Verbindung und die Anzahl der Nicht-Null-Elemente ist proportional der Gitterpunkte N .

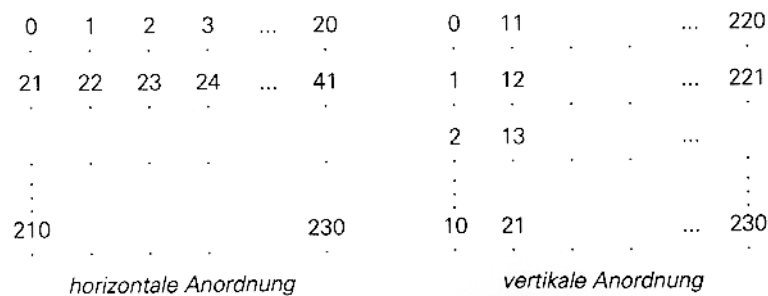


Für dünn besetzte Matrizen gibt es spezielle Speicherstrukturen, die das Wissen um die erzwungen Nullen ausnutzen. In den Programmpaketen für lineare Algebra mit sparse-matrices kann man meist zwischen verschiedenen Speicherstrukturen auswählen. Natürlich wird durch die Verwendung von sparse linear algebra routines nicht nur Speicherplatz sondern auch erheblich Rechenzeit gespart.

Bandmatrizen. Eine Möglichkeit zur Speicherung von dünn besetzten Matrizen ist die Speicherung des Bandes mit nichtnull Elementen entlang der Hauptdiagonalen. Matrizen mit Bandstruktur entstehen bei der Diskretisierung mit finite Differenzen beziehungsweise finiten Elementen. Die Bandbreite kann durch geeignete Wahl der Nummerierung der Gitterpunkte minimiert werden.



Wird zum Beispiel das Rechteck mit dem Achsenverhältnis 2:1 mit einem regelmässigen Netz von 200 Elementen überzogen kann man folgenden zwei Nummerierungen der Knotenpunkte wählen.



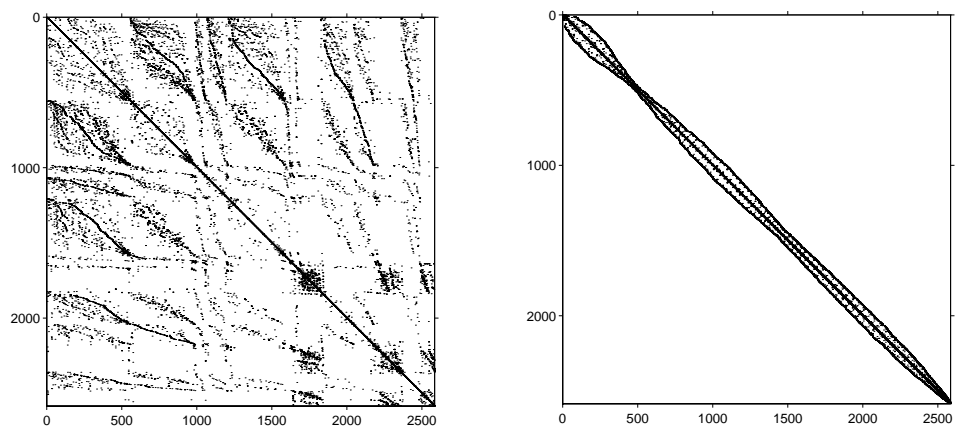
Bei der Diskretisierung der Laplace Gleichung mit dem Differenzenverfahren entsteht bei der horizontalen Anordnung das Gleichungssystem

$$u_{i-21} + u_{i-1} + u_{i+1} + u_{i+21} = f_i \tag{2.20}$$

und in der vertikalen Anordnung das Gleichungssystem

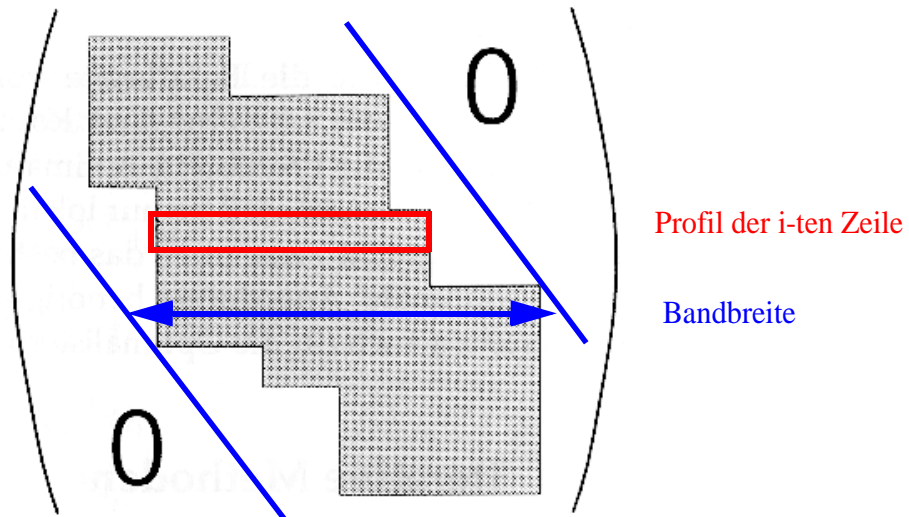
$$u_{i-11} + u_{i-1} + u_{i+1} + u_{i+11} = f_i \quad (2.21)$$

Die Systemmatrix des Gleichungssystems (2.20) hat eine Bandbreite von 43 die Matrix des Systems (2.21) eine Bandbreite von . Zählt man in jeder Zeile der Matrix die Anzahl der Elemente zwischen dem ersten und letzten Nichtnull-Element (dem Nichtnull-Element mit dem niedrigsten und dem Nichtnull-Element mit dem höchsten Spaltenindex) erhält man die Bandbreite der Matrix. Die meisten mesh generatoren besitzen routinen zur Bandbreitenreduktion durch Ummummerierung der Knoten.



Bandbreitenreduktion durch Ummummerierung der Knoten.

Profilspeicherung. Dabei wird die Matrix zeilenweise gespeichert. Zum Profil der i -ten Zeile gehören alle Elemente die zwischen den den äußersten Nichtnullen-Elementen liegen, einschließlich dieser Nichtnullen selbst. Bei der LU Zerlegung bleibt das Profil der Matrix erhalten. Es kommen keine Nichtnull Elemente außerhalb das Profils hinzu. Das linke Profil der L-Matrix und das rechte Profile der U-Matrix sind mit dem Profil der ursprünglichen Matrix gleich.



Band und Profil eine Matrix.

Profil der i -ten Zeile:

000000000000xxxxxx000x0x000xxxxx00xxxxx00000000000000



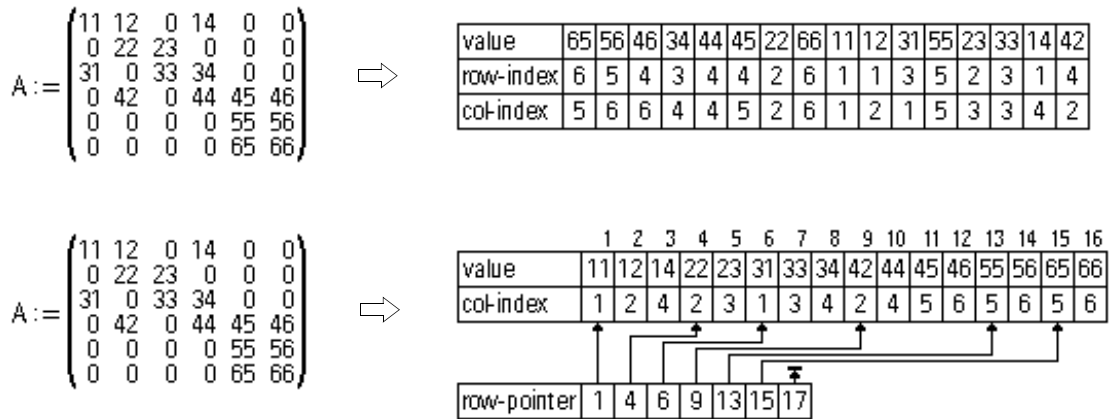
Speichere für jede Zeile: das Profil und den Spaltenindex des ersten und letzten Nichtnull-Elementes.

Die Kenntnis des Profils hat bei direkten Methoden den Vorteil, daß im Vergleich zu Bandmethoden weniger Eliminierungsschritte ausgeführt werden müssen, weil die variable Bandbreite noch in Betracht gezogen wird.

Das Profil ist ebenso wie die Bandbreite von der Nummerierung der Knoten abhängig. Algorithmen zur Reduktion der Bandbreite oder des Profils sind in der Zeitschrift ACM Transactions on Mathematical Software publiziert. Der Fortran-Code ist public domain (<http://www.acm.org/toms>).

Sparse storage schemes. Wie wir im Abschnitt über iterative Verfahren zur Lösung von linearen Gleichungssysteme sehen werden, ist für deren Lösung die explizite Speicherung der Matrix A gar nicht nötig. Es genügt

ein Verfahren zur Berechnung des Matrix-Vektor Produkt \mathbf{Ax} . Es gibt verschiedene Speicherstrukturen, die nur die Nichtnull-Elemente speichern. Natürlich sind dann auch integer-arrays zur Verwaltung der Indizes nötig.



Anzahl der Rechenoperationen. Bei Bandmatrizen reduziert sich der Aufwand für die Rückwärtssubstitution auf $O(NB)$, falls B die Bandbreite bezeichnet. Die folgende Tabellen gibt Abschätzungen für die typische Bandbreite bei mit FD oder FEM diskretisierten Gleichungen.

	2D	3D
Maschenweite h	$1/L$	$1/L$
Anzahl der Knotenpunkte N	$N = L^3$	$N = L^2$
Bandbreite B	$L^2 = N^{2/3}$	$L = N^{1/2}$

Meist hat man mehrere Gleichungssystem mit verschiedenen rechten Seiten zu lösen. Die LU Zerlegung muß nur einmal durchgeführt werden. Der Aufwand pro System entspricht dem Aufwand der Rücksubstitution $O(NB)$ und ist $N^{5/3}$ für 3D Probleme und $N^{3/2}$ für 2D Probleme.

Kondition eines Gleichungssystems, Rundungsfehlern

Das System

$$\mathbf{Ax} = \mathbf{f} \tag{2.22}$$

soll gelöst werden. Nun wird die rechte Seite durch $\mathbf{f} + \mathbf{r}$ ersetzt und das System

$$\frac{\|\mathbf{x} - \mathbf{y}\|}{\|\mathbf{x}\|} \leq \frac{\|\mathbf{A}^{-1}\| \|\mathbf{A}\| \|\mathbf{x}\| \|\mathbf{r}\|}{\|\mathbf{x}\| \|\mathbf{f}\|} \quad (2.31)$$

$$\frac{\|\mathbf{x} - \mathbf{y}\|}{\|\mathbf{x}\|} \leq \|\mathbf{A}^{-1}\| \|\mathbf{A}\| \frac{\|\mathbf{r}\|}{\|\mathbf{f}\|} = K(\mathbf{A}) \frac{\|\mathbf{r}\|}{\|\mathbf{f}\|} \quad (2.32)$$

Bei der Diskretisierung von partiellen Differentialgleichungen zweiter Ordnung ist die Konditionszahl proportional zu $1/h^2$, wobei h die Maschenweite ist

$$K(\mathbf{A}) \sim \frac{1}{h^2} \quad (2.33)$$

Dadurch ergeben sich folgende Abschätzungen für die Konditionszahl mit der Zahl der Knotenpunkte N und der Anzahl der Zellen in einer Raumrichtung $L = 1/h$.

Konditionszahl	
allgemein	$K \sim L^2$
2D: $N = L^2, N = L^{1/2}$	$K \sim N$
3D: $N = L^3, N = L^{1/3}$	$K \sim N^{2/3}$

Iterative Methoden

Differenzenverfahren und die Methode der finiten Element führen zu großen, linearen Gleichungssystemen mit schwach besetzter Koeffizientenmatrix. Zur Lösung dieser Systeme werden in der Praxis fast ausschließlich iterative Verfahren verwendet. Im Vergleich zu direkten Methoden ergibt sich bei großen Systemen eine erhebliche Rechenzeiterparnis, wenn an die Lösungsgenauigkeit keine zu hohen Anforderungen gestellt werden. In der Anwendung ist das fast immer der Fall, da das Gleichungssystem selbst eine Näherung für einen Differentialoperator darstellt. Gleichungssysteme, die aus der Diskretisierung von Randwertproblemen entstehen haben meist spezielle Eigenschaften, die die Konvergenz eines iterativen Verfahrens sicherstellen.

Es lassen sich drei Arten von iterativen Verfahren unterscheiden: Standardmethoden, Gradientenmethoden und Mehrgitterverfahren.

Standardmethoden

Zu lösen ist das lineare System

$$\mathbf{A} \mathbf{u} = \mathbf{f} \quad (2.34)$$

oder

$$\sum_{j=1}^N a_{ij} u_j = f_i, \quad i = 1, \dots, N \quad (2.35)$$

mit nichtsingulärer Systemmatrix $\mathbf{A} \in \mathfrak{R}^{N \times N}$. Ein einfaches Iterationsverfahren zur Lösung von (2.35) läßt sich wie folgt konstruieren.

- 1 **Dividiere die i-te Zeile durch a_{ii}** $u_i + \frac{1}{a_{ii}} \sum_{j \neq i} a_{ij} u_j = \frac{f_i}{a_{ii}}$
- 2 **Löse nach u_i auf** $u_i = \frac{f_i}{a_{ii}} - \frac{1}{a_{ii}} \sum_{j \neq i} a_{ij} u_j$
- 3 **Iteriere die Gleichung (2.36)**

$$u_i^{(k+1)} = u_i^{(k)} + \frac{1}{a_{ii}} \left(f_i - \sum_{j=1}^N a_{ij} u_j^k \right), \quad k = 0, 1, \dots \quad (2.36)$$

Das ist Iterationsvorschrift des Jacobi-Verfahrens. Werden auf der rechten Seite alle bereits bekannten $u_j^{(k+1)}$ verwendet erhält man das Gauss-Seidel-Verfahren

$$u_i^{(k+1)} = u_i^{(k)} + \frac{1}{a_{ii}} \left(f_i - \sum_{j=1}^{i-1} a_{ij} u_j^{(k+1)} - \sum_{j=i+1}^N a_{ij} u_j^{(k)} \right) \quad (2.37)$$

Der Index i läuft jeweils von 1 bis N . Wird der Klammerausdruck in (2.36) oder (2.37) mit einem Relaxationsparameter ω und $u_i^{(k)}$ mit $(1-\omega)$ gewichtet, spricht man vom gedämpften Jacobi-Verfahren beziehungsweise SOR-Verfahren. Ist $0 < \omega < 1$, spricht man von Unterrelaxation, bei $\omega > 1$ von (sukzessiver) Überrelaxation..

Gradienten Methoden

- allgemeines Iterationsschema zur iterativen Lösung von linearen Gleichungen

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \mathbf{C}(\mathbf{f} - \mathbf{A}\mathbf{u}_i), \quad i = 1, 2, \dots$$

- die Größe $(\mathbf{f} - \mathbf{A}\mathbf{u}_i)$ ist das i -te Residuum der Gleichung

$$\mathbf{u}_i = \mathbf{u}_i + \mathbf{C}\mathbf{r}_i, \quad i = 1, 2, \dots$$

- betrachte die Ergebnisse der einzelnen Schritte

$$\begin{aligned} & \mathbf{u}_0 \\ \mathbf{u}_1 &= \mathbf{u}_0 + \mathbf{C}\mathbf{r}_0 \\ \mathbf{u}_2 &= \mathbf{u}_1 + \mathbf{C}(\mathbf{f} - \mathbf{A}\mathbf{u}_1) \\ &= \mathbf{u}_1 + \mathbf{C}(\mathbf{f} - \mathbf{A}\mathbf{u}_0 - \mathbf{A}\mathbf{C}\mathbf{r}_0) \\ &= \mathbf{u}_0 + \mathbf{C}\mathbf{r}_0 + \mathbf{C}(\mathbf{f} - \mathbf{A}\mathbf{u}_0) + \mathbf{C}\mathbf{A}\mathbf{C}\mathbf{r}_0 \\ &= \mathbf{u}_0 + 2\mathbf{C}\mathbf{r}_0 + \mathbf{C}\mathbf{A}\mathbf{C}\mathbf{r}_0 \end{aligned}$$

- \mathbf{u}_i ist ein Element des von folgenden Vektoren aufgespannten Raumes

$$\mathbf{u}_0, \mathbf{C}\mathbf{r}_0, \mathbf{C}\mathbf{A}\mathbf{C}\mathbf{r}_0, \dots, (\mathbf{C}\mathbf{A})^{k-1}\mathbf{C}\mathbf{r}_0$$

Krylow-Raum Methoden

ist \mathbf{A} eine $N \times N$ Matrix, dann heißt der von den Funktionen

$$\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, (\mathbf{A})^{i-1}\mathbf{r}_0$$

aufgespannte Raum Krylow-Raum $K^i(\mathbf{A}, \mathbf{r}^{(0)})$ der Dimension i

- Optimales iteratives Verfahren zur Lösung von $\mathbf{A}\mathbf{u} = \mathbf{f}$
 suche Iterationspunkte \mathbf{u}_i im Krylow Raum so,
 daß der Abstand zu exakten Lösung \mathbf{u} minimal wird

$$\|\mathbf{u} - \mathbf{u}_i\|_2 \rightarrow \text{minimal}$$

- zum Beispiel: $\mathbf{u}_0 = 0$ dann ist $\mathbf{u}_1 \in \{\mathbf{r}_0\}$
- $\mathbf{u}_1 = \alpha_0 \mathbf{r}_0$

$$\begin{aligned} \|\mathbf{u} - \mathbf{u}^{(1)}\|_2^2 &= (\mathbf{u} - \alpha_0 \mathbf{r}_0, \mathbf{u} - \alpha_0 \mathbf{r}_0) \\ &= (\mathbf{u}, \mathbf{u}) - 2\alpha_0(\mathbf{u}, \mathbf{r}_0) + \alpha_0^2(\mathbf{r}_0, \mathbf{r}_0) \end{aligned}$$

minimiere nach α_0

$$\alpha_0 = \frac{(\mathbf{r}_0, \mathbf{u})}{(\mathbf{r}_0, \mathbf{r}_0)}$$

\mathbf{u} ist unbekannt, α_0 ?

- Verwende das Innere Produkt

$$(\mathbf{x}, \mathbf{x})_A = (\mathbf{x}, \mathbf{Ax})$$

- \mathbf{Ax} ist bekannt ($\mathbf{Au} = \mathbf{f}$), somit läßt sich α_0 berechnen

$$\alpha_0 = \frac{(\mathbf{r}_0, \mathbf{Au})}{(\mathbf{r}_0, \mathbf{Ar}_0)}$$

- Wähle $\mathbf{u}_i \in K^i(A, \mathbf{r}_0)$ so, daß der Abstand $\|\mathbf{u} - \mathbf{u}_i\|_A$ minimal wird

$$\mathbf{f} - \mathbf{Au}_i \perp K^i(A, \mathbf{r}_0) \quad \text{Krylow Unterraum}$$

\Rightarrow Methode der konjugierten Gradienten

Othogonalität der Residuen

- Kette von Unterräumen

$$K^1(\mathbf{A}, \mathbf{r}_0) \subset K^2(\mathbf{A}, \mathbf{r}_0) \subset \dots \subset K^N(\mathbf{A}, \mathbf{r}_0)$$

$$\{\mathbf{r}_0\} \subset \{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0\} \subset \dots \subset \{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \dots, \mathbf{A}^{N-1}\mathbf{r}_0\}$$

$$\mathbf{u}_1 \in K^1(\mathbf{A}, \mathbf{r}_0), \mathbf{u}_2 \in K^2(\mathbf{A}, \mathbf{r}_0), \dots, \mathbf{u}_N \in K^{N-1}(\mathbf{A}, \mathbf{r}_0)$$

- Wähle \mathbf{u}_i so, daß

$$\mathbf{r}_i = \mathbf{f} - \mathbf{A}\mathbf{u}_i \perp K^i(\mathbf{A}, \mathbf{r}_0)$$

für $\mathbf{u}_0 = 0$ folgt $\mathbf{r}_0 = \mathbf{f}$, und somit $\mathbf{f} \in \{\mathbf{r}_0\} = K^1(\mathbf{A}, \mathbf{r}_0)$

$$\mathbf{u}_1 = \alpha \mathbf{r}_0,$$

$$\mathbf{r}_1 = \mathbf{f} - \mathbf{A}\mathbf{u}_1 = \mathbf{f} - \alpha \mathbf{A}\mathbf{r}_0 \in \{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0\} = K^2(\mathbf{A}, \mathbf{r}_0)$$

$$\Leftrightarrow \mathbf{r}_0 \perp \mathbf{r}_1 \quad \Leftrightarrow \quad \{\mathbf{r}_0, \mathbf{r}_1\} \quad \text{bilden eine orthogonale Basis von } K^2(\mathbf{A}, \mathbf{r}_0)$$

- Nach N Schritten: N orthogonale Basisvektoren für \mathbf{u}_i
jeder Vektor aus \mathbb{R}^N läßt sich darstellen
 \mathbf{u}_N ist die Lösung des Gleichungssystems

Weitere Gradientenverfahren

- CG – Algorithmus
($\mathbf{x}, \mathbf{A}\mathbf{x}$) ist nur dann ein inneres Produkt, wenn \mathbf{A} symmetrisch und positiv definit ist
- LSQR an algorithm for sparse linear equations and sparse least square problems
 \mathbf{A} ist nicht positiv definit: löse das System $\mathbf{A}^T \mathbf{A} = \mathbf{A}^T \mathbf{f}$
- GMRES a generalized miniam residual algorithm for non-symmetric linear systems
suche Iterationspunkte im Krylow Unterraum, sodaß

Multigrid Verfahren

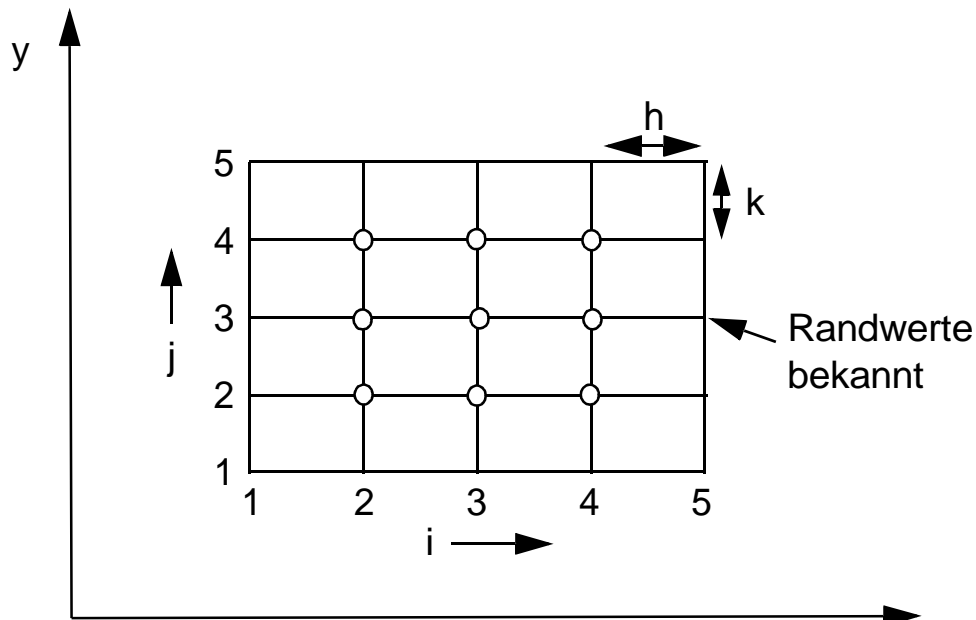
- Konvergenzrate der Iterationsverfahren

$$\|\mathbf{u}_{k+1} - \mathbf{u}\| \leq (1 - O(h^2)) \|\mathbf{u}_k - \mathbf{u}\|$$

- h klein notwendig damit *Ortsdiskretisierungsfehler* klein wird
- Iterationsverfahren konvergieren immer *langsamer*, je kleiner h
- Mehrgitterverfahren
löse die Gleichung auf einer Reihe von Gittern, $h_1 > h_2 > \dots > h_n$
- Grundidee: schwach variierende Fehler Anteile können auch auf größeren Gittern beseitigt werden
- Gesamtrechenzeit \sim Anzahl der Unbekannten

Beispiel: Lösung der Poissongleichung mit finiten Differenzen

$$u_{xx} + u_{yy} = f(x, y)$$



$$\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2} + \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{k^2} = f_{i,j}$$

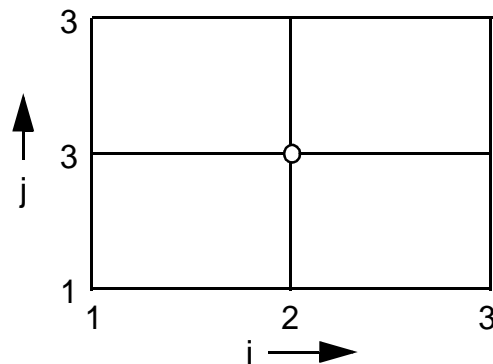
- Jacobi Verfahren

$$u_{i,j} = \frac{1}{2} \left(\frac{u_{i-1,j} + u_{i+1,j}}{h^2} + \frac{u_{i,j-1} + u_{i,j+1}}{k^2} - f_{i,j} \right) / \left(\frac{1}{h^2} + \frac{1}{k^2} \right)$$

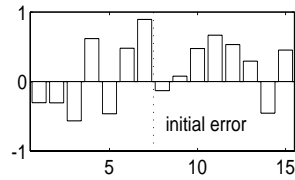
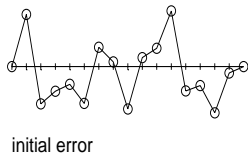
- setze auf der rechte Seite bekannte Näherungen für u ein
- berechne damit eine verbesserten Wert für $u_{i,j}$
- und so weiter
- möglicher Startwert ist $u = 0$
- Rechenaufwand $\sim N^2$
- Einfluß der Randbedingungen wandert nur langsam ins Innere hinein, da auf $u_{i,j}$ nur die benachbarten Knoten einwirken

Verbesserungsvorschläge

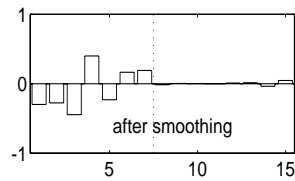
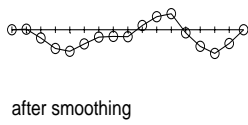
- Nur ein Innenknoten, dann ist der Einfluß des Randes sofort im Inneren wirksam



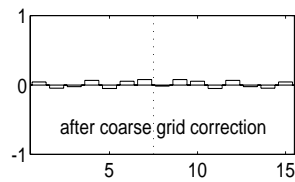
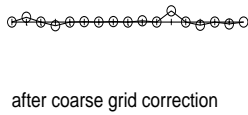
- $u_{2,2}$ nach **einem** Iterationsschritt
- Einfluß der Randwerte nach **einem** Schritt im Inneren wirksam
- halbiere das Gitter
- Lösung auf dem groben Gitter \rightarrow Startwert für Iterationen am feinen Gitter



Zweigitterverfahren



1 Glättung
wenige Iterationen auf
feinem Gitter



2 Grobgitterkorrektur
wenige Iterationen auf
grobem Gitter

Rechnen mit Matrixen–Bibliotheksroutinen

- Matrix Speicherung in Rechnern
- Praktische Tips
- scientific libraries, WWW
- wie finde ich verfügbare Bibliotheksroutine

Vektoren und Matrizen

- **Beschränkter Speicher**
Matrizen und Felder benötigen Speicherplatz: $a(10000,10000) \rightarrow 1 \text{ GB}$
 - **Rechenzeit**
Matrix-Vektor Multiplikationen N^2 : $N = 2N \rightarrow 4$ -fache Rechenzeit
Algorithmen mit Aufwand N^3 : $N = 2N \rightarrow 8$ -fache Rechenzeit
 - **Paging**
Cache \rightarrow fast memory \rightarrow slow memory \rightarrow disk space
falls der einen Rechenvorgang benötigte Speicher *eine Grenze überschreitet*,
kann die Rechenzeit um Größenordnungen steigen
 - **Matrix Speicherung**
Fortran: $a(1,1) a(2,1) a(3,1) a(1,2) a(2,2) a(3,2) a(1,3) a(2,3) a(3,3)$
C: $a(0,0) a(0,1) a(0,2) a(1,0) a(1,1) a(1,2) a(2,0) a(2,1) a(2,2)$
 - **Physikalische und logische Dimension**
 $\text{real}*8 a(3,3)$ reserviert Speicherplatz für 3×3 Matrix,
verwende nur 2×2 Matrix in der Rechnung, dann belegt die Matrix
keinen zusammenhängende Speicherplatz:
 $a(1,1) a(2,1) a(3,1) a(1,2) a(2,2) a(3,2) a(1,3) a(2,3) a(3,3)$
- an Unterprogrammen muß immer die physikalische und logische Dimension einer Matrix übergeben werden

Virtual memory

- Speicher ist in pages unterteilt
- wird ein Matrix-Element $a(i,j)$ benötigt
 - Nummer der Seite
 - Platz von $a(i,j)$ innerhalb der Seite
- page faults
 - gewünschte Seite befindet sich auf der Platte und nicht im RAM

Faustregeln

- 1 Denke über die folgenden Punkte nur bei großen Programmen nach

3 C: Ändere den rechten Index eines Feldes in der innersten Schleife

BLAS – Basic Linear Algebra

- ein Programm soll auf verschiedenen Plattformen möglichst optimal laufen
- Grundbausteine jedes Algorithmus sind Operationen der linearen Algebra
 - Vektor-Vektor,
 - Matrix-Vektor und
 - Matrix-Matrix Operationen
- verwende an die hardware angepaßte Bibliotheksroutinen
→ BLAS
- verwende BLAS routinen für eigene Algorithmen
- benutze Programmbibliotheken, die auf BLAS routine aufbauen

Implementierungen

- ESSL Engineering and Scientific Subroutine Library (IBM)
- DXML Advanced Mathematical Library (DEC)
- SPARSE BLAS Basic Linear Algebra for sparse matrices
- Vektorrechner
- Parallelrechner

Beispiel: Level 1 BLAS

Vektor – Vektor Operationen

- interchange $\mathbf{x} \leftrightarrow \mathbf{y}$
- scale $\mathbf{y} \leftarrow a \mathbf{x}$
- copy $\mathbf{y} \leftarrow \mathbf{x}$
- add $\mathbf{y} \leftarrow a \mathbf{x} + \mathbf{y}$
- dot product $\mathbf{x}^T \mathbf{y}$
- two-norm $\|\mathbf{x}\|_2$
- one-norm $\|\mathbf{x}\|_1 = \text{Summe}(|x_i|)$
- maximum modulus lowest index with $\max(|x_i|)$

Zahlendarstellung

Gleitkommadarstellung

Bei dieser Darstellung wird die Position des Dezimalpunktes einer Zahl zusätzlich dynamisch geregelt.

Unangenehmerweise ist es möglich, ein und dieselbe Zahl auf verschiedenste Arten darzustellen. Zum Beispiel: $0,123 = 123 * 10^{-3} = 1230 * 10^{-4} = 12300 * 10^{-5}$ Darum wurde für die Gleitkommadarstellung die normierte Form entwickelt: Die Mantisse ist eine Zahl, deren Vorkommateil 1 ist. Einzigster Problemfall: Die Zahl 0. Die restliche Information der Zahl ist im Nachkommateil der Mantisse und im Exponenten der Zahl.

Normiert man z.B. die binäre Zahl +11001,1011 sieht das Ergebnis folgendermaßen aus: $+1,10011011 * 10^{100}$ Da die Ziffer vor dem Komma also fast immer 1 ist (Ausnahme: 0), kann man auf deren Abspeicherung verzichten und gewinnt so ein zusätzliches Mantissenbit, wodurch die Genauigkeit erhöht werden kann. Diese Genauigkeit wird allerdings mit der Sonderbehandlung von 0 erkauft.

Es muß also nur das Vorzeichen (1 Bit), die sogenannte Mantisse und der Exponent abgespeichert werden.

$$x = (-1)^s * \text{Mantisse} * 2^{\text{Exponent} - \text{Bias}}$$

$$10^{-44} \leq \text{single precision} \leq 10^{38}$$

$$10^{-322} \leq \text{double precision} \leq 10^{308}$$

IEEE Das IEEE Format (Institute of Electrical and Electronics Engineers) ist eine genormte Gleitkommadarstellung und wird in den meisten Rechnersystemen verwendet. Bei beiden Formaten wird nur der Nachkommateil der Mantisse abgespeichert (1 Bit gespart)

Länge	Vorzeichen	Exponent	Mantisse	Bias	C/C++	F77
32 Bit	1 Bit	8 Bit	23 Bit	127	float	real*4
64 Bit	1 Bit	11 Bit	52 Bit	1023	double	real*8

Maschinen-Genauigkeit

größte positive Zahl, die zu 1 addiert, als Ergebnis 1 liefert

$$1_c + \varepsilon_m = 1_c$$

Der Index c gibt an das die Zahl in einem Computer gespeichert ist. Wird eine Zahl x im Computer dargestellt so gilt daher

$$x_c = x(1+\varepsilon)$$

Test der Maschinengenauigkeit

```
eps = 1
begin do N times
  eps = eps/2.
  one = 1. + eps
  write out: loop number, one, eps
enddo
```

Zahlendarstellung	Maschinen-Genauigkeit
einfache Genauigkeit (32 bit)	$\sim 10^{-7}$
doppelte Genauigkeit (64 bit)	$\sim 10^{-16}$

Daher: Zahlen im Gleitkommaformat (float,double,...) nie auf Gleichheit prüfen! Statt dessen Prüfung auf $>$ oder $<$ oder auf einen kleinen Bereich.

```
int main()
{
  float i=0.0;
  for (i=0.0; i != 1.0; i += 0.1)
  {
    /* Endlosschleife, weil 0.1 nicht exakt dargestellt werden kann. */
    printf("%f",i);
  };
  return(0);
};

int main()
{
  float i=0.0;
  for (i=0.0; i<0.9999 || i>1.0001; i += 0.1)
  {
    /* Durch die Prüfung auf den E-Bereich von 0,0001 ist es jetzt keine Endlosschleife
    mehr. */
    printf("%f",i);
  };
  return(1);
};
```

Rundungsfehler

Eine Rechnung mit einfacher Genauigkeit ergibt:

$$2(1/3) - 2/3 = 0.6666666 - 0.6666667 = -0.0000001 \text{ das ist ungleich } 0$$

Subtraktion zweier Zahlen $a = b - c$

$$a_c = b_c - c_c$$

$$a_c = b(1 + \varepsilon_b) - c(1 + \varepsilon_c)$$

$$a_c/a = 1 + \varepsilon_b(b/a) - \varepsilon_c(c/a)$$

Der mittlere Fehler in a ist das gewichtete Mittel der Fehler in b und c . Sind jedoch die beiden Zahlen b und c ungefähr gleich groß so ist a klein und es gilt:

$$a_c/a \sim 1 + (b/a)(\varepsilon_b - \varepsilon_c)$$

Auch wenn sich die beiden Fehler ε_b und ε_c teilweise kompensieren ist (b/a) groß und somit kann sich a_c deutlich von a unterscheiden.

Falls zwei Zahlen subtrahiert werden und das Ergebnis eine kleine Zahl ist, dann hat das Ergebnis weniger signifikante Stellen.

Multiplikation zweier Zahlen $a = b \times c$

$$a_c = b_c \times c_c$$

$$a_c/a = (1 + \varepsilon_b)(1 + \varepsilon_c) \sim 1 + \varepsilon_b + \varepsilon_c$$

Die Fehler ε_b und ε_c können positives oder negatives Vorzeichen haben, somit ist der Fehler in a_c manchmal kleiner und manchmal größer als die Fehler in b_c und c_c .

Für eine Folge von Multiplikationen kann man davon ausgehen, dass der aktuelle Fehler zufällig ist. Der mittlere Fehler nach N Schritten ist dann

$$\varepsilon \sim N^{1/2} \varepsilon_m$$

Einfache oder Doppelte Genauigkeit. Ein Rechner mit 10^{10} floating point Operationen pro Sekunden führt in drei Stunden ca. 10^{14} Operationen durch. Bei zufälligen Rundungsfehlern hat sich daher ein relative Fehler von $10^7 \varepsilon_m$ akkumuliert. Damit der Fehler kleiner als die Antwort ist muss die Maschinengenauigkeit ein kleiner als 10^{-7} sein. Eine Rechnung mit einfacher Genauigkeit, die mehrere Stunden dauert, enthält wahrscheinlich viel numerisches Rauschen.

Verwende doppelte Genauigkeit für lange Rechnungen.

Matrixnormen

In numerischen Mathematik ist es wichtig Abschätzungen der Form $\|\mathbf{A}\mathbf{u}\| \leq Q\|\mathbf{u}\|$ mit einer reellen positiven Konstante Q zu kennen. Dazu kann man wie bei Vektoren auf für Matrizen eine Norm definieren. Eine Matrixnorm bildet die Matrix $\mathbf{A} \in \mathfrak{R}^N$ auf einen Skalar ab [26].

Matrixnorm

Jeder Matrix $\mathbf{A} \in \mathfrak{R}^{N \times N}$ werde eine reelle Zahl $\|\mathbf{A}\|$ mit folgenden Eigenschaften zugeordnet:

- 1 $\|\mathbf{A}\| > 0$, für $\mathbf{A} \neq \mathbf{0}$
- 2 $\|\mathbf{A} + \mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$, für alle $\mathbf{A}, \mathbf{B} \in \mathfrak{R}^{N \times N}$
- 3 $\|\alpha\mathbf{A}\| = |\alpha|\|\mathbf{A}\|$, für jede Zahl $\alpha \in \mathbb{R}$
- 4 $\|\mathbf{A} \cdot \mathbf{B}\| \leq \|\mathbf{A}\| \cdot \|\mathbf{B}\|$

Dann heißt $\|\mathbf{A}\|$ Matrixnorm von \mathbf{A} .

Eine Matrixnorm $\|\mathbf{A}\|_M$ heißt verträglich mit der Vektornorm $\|\mathbf{u}\|_V$, wenn

$$\|\mathbf{A}\mathbf{u}\| \leq \|\mathbf{A}\|_M \|\mathbf{u}\|_V \quad (\text{A.1})$$

für jede beliebige Matrix $\mathbf{A} \in \mathfrak{R}^{N \times N}$ und jeden beliebigen Vektor $\mathbf{u} \in \mathbb{R}^N$ gilt. Setzt man in (A.1) $\|\mathbf{A}\|_M = Q$ so erhält man eine Abschätzung $\|\mathbf{A}\mathbf{u}\| \leq Q\|\mathbf{u}\|$ in der geforderten Form.

Besonders häufig verwendete Matrixnormen sind die folgenden:

Matrixnorm

Norm der maximalen Spaltenbetragssumme $\|\mathbf{A}\|_1 = \max_j \left(\sum_{i=1}^N |a_{ij}| \right)$

Spektralnorm $\|\mathbf{A}\|_2 = +\sqrt{\text{größter Eigenwert von } \mathbf{A}^* \mathbf{A}}$

Norm der maximalen Zeilenbetragssumme $\|\mathbf{A}\|_\infty = \max_i \left(\sum_{j=1}^N |a_{ij}| \right)$

\mathbf{A}^* ist die zu \mathbf{A} konjugiert komplexe Matrix.

Allgemein gibt es zu jeder Vektornorm eine verträgliche Matrixnorm, die sogenannte zugeordnete Matrixnorm

$$\|\mathbf{A}\| = \max_{\|\mathbf{u}\|=1} \|\mathbf{A}\mathbf{u}\| = \max_{\mathbf{u} \neq \mathbf{0}} \frac{\|\mathbf{A}\mathbf{u}\|}{\|\mathbf{u}\|}, \quad \mathbf{u} \in \mathbb{C}^N. \quad (\text{A.2})$$

Eigenwerte—Spektralradius

Sei $\mathbf{A} \in \mathbb{R}^{N \times N}$ eine $N \times N$ Matrix, λ eine komplexe Zahl und \mathbf{v} ein komplexer Vektor so, daß

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}, \quad \mathbf{v} \neq \mathbf{0}. \quad (\text{A.3})$$

Dann heißt λ Eigenwert und \mathbf{v} Eigenvektor von \mathbf{A} . Die Wurzeln des charakteristischen Polynomes

$$p(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}), \quad (\text{A.4})$$

wobei \mathbf{I} die $N \times N$ -Einheitsmatrix ist, sind die Eigenwerte von \mathbf{A} . Eine reell-symmetrische oder hermitesche Matrix besitzt nur reelle Eigenwerte. Weiters gilt

$$\|\mathbf{A}\|_2 = +\sqrt{\text{größter Eigenwert von } \mathbf{A}^* \mathbf{A}} = +\sqrt{[\rho(\mathbf{A})]^2} = \rho(\mathbf{A}). \quad (\text{A.5})$$

Im Falle einer hermiteschen (beziehungsweise reell-symmetrischen) Matrix ist also die Spektralnorm gleich dem Spektralradius.

Folgender Satz ist hilfreich zur Abschätzung des Spektralradius einer Matrix.

Abshätzung für den Spektralradius

Es seien $\mathbf{A} \in \mathfrak{R}^{N \times N}$ und $\|\mathbf{A}\|$ irgendeine einer Vektornorm zugeordnete Matrixnorm. Dann ist

$$\rho(\mathbf{A}) \leq \|\mathbf{A}\|.$$

Eigenschaften Matrizen

Numerische Verfahren der linearen Algebra können für Matrizen mit speziellen Eigenschaften optimiert werden.

Eine Matrix heißt wenn

symmetrisch $\mathbf{A} = \mathbf{A}^T$

hermitesch $\mathbf{A} = \mathbf{A}^*$

positiv definit $\mathbf{u}^* \mathbf{A} \mathbf{u} > 0$, für alle $\mathbf{u} \in \mathbb{R}^N$

diagonal dominant $\sum_{\substack{j=1 \\ j \neq i}}^N |a_{ij}| \leq |a_{ii}|, k = 1, 2, \dots, N$

M-Matrix \mathbf{A} nicht singulär ist und reell ist, sämtliche Elemente von \mathbf{A}^{-1} nicht negativ sind, $a_{ij} \leq 0, i \neq j; i, j = 1, 2, \dots, N$