Lineare Gleichungssysteme

Lineare Gleichungssysteme treten oft als Teilprobleme bei numerischen Verfahren auf, z.B.:

- Lineare Ausgleichsprobleme (Normalgleichungen)
- Partielle Differentialgleichungen
  - bei manchen impliziten Verfahren muß in jedem Zeitschritt ein lineares Gleichungssystem gelöst werden
  - Randwertprobleme bei elliptischen Systemen (z.B. Potentialgleichung)
  - man hat oft spezielle, dünnbesetzte (*sparse*) Matrizen (*Bandmatrizen*)

Direkte Verfahren
Lösen das Problem durch geeignete Zerlegung der Koeffizientenmatrix nach endlich vielen Schritten von algebraischen Operationen, sodaß kein Verfahrensfehler auftritt. Abgesehen von Rundungsfehlern erhält man die exakte Lösung. Allerdings können wegen der großen Anzahl von Punktoperationen $\propto n^3$ die akkumulierten Rundungsfehler das Ergebnis erheblich verfälschen, sodaß die Lösung völlig unbrauchbar werden kann.
Beispiele: Gaußscher Algorithmus, LU-Zerlegung.

Iterationsverfahren
Ausgehend von einer Anfangsnäherung für die Lösung (Startvektor) wird diese schrittweise durch Iteration verbessert. Obwohl hier Verfahrens- und Rundungsfehler auftreten, sind Iterationsverfahren vor allem für große Systeme mit schwach besetzten Matrizen geeignet (pro Iterationsschritt ist die Anzahl der Punktoperationen $\propto n^2$).
Beispiele: Jacobi, Gauß-Seidel, SOR, ADI, CG.

Das Gaußsche Eliminationsverfahren

Wir betrachten ein System von n linearen Gleichungen in n Unbekannten $x_1, \ldots, x_n$:

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

oder in Matrixform:

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

Beispiel zum Gaußschen Eliminationsverfahren

$$
\begin{array}{rcrcrcrl}
2\,x_1 & + & x_2 & + & x_3 & = & 5 & \mathrm{G1}^{(0)} \\
4\,x_1 & - & 6\,x_2 & & & = & -2 & \mathrm{G2}^{(0)} \\
-2\,x_1 & + & 7\,x_2 & + & 2\,x_3 & = & 9 & \mathrm{G3}^{(0)} \\
\hline
2\,x_1 & + & x_2 & + & x_3 & = & 5 & \mathrm{G1}^{(0)} \\
 & - & 8\,x_2 & - & 2\,x_3 & = & -12 & \mathrm{G2}^{(1)} = \mathrm{G2}^{(0)} - 2 \cdot \mathrm{G1}^{(0)} \\
 & & 8\,x_2 & + & 3\,x_3 & = & 14 & \mathrm{G3}^{(1)} = \mathrm{G3}^{(0)} - (-\,\mathrm{G1}^{(0)}) \\
\hline
2\,x_1 & + & x_2 & + & x_3 & = & 5 & \mathrm{G1}^{(0)} \\
 & - & 8\,x_2 & - & 2\,x_3 & = & -12 & \mathrm{G2}^{(1)} \\
 & & & & x_3 & = & 2 & \mathrm{G3}^{(2)} = \mathrm{G3}^{(1)} - (-\,\mathrm{G2}^{(1)}) \\
\hline
\end{array}
$$

Damit ist das ursprüngliche Gleichungssystem in ein äquivalentes Gleichungssystem von oberer Dreiecksgestalt übergeführt, welches sich von rückwärts her auflösen läßt:

Aus der letzten Gleichung erhält man

$$x_3 = 2 \ .$$

Einsetzen von x_3 in die vorletzte Gleichung liefert

$$x_2 = (-12 + 4)/(-8) = 1 \ .$$

Aus Gleichung $\mathrm{G1}^{(0)}$ erhält man schließlich

$$x_1 = (5 - 1 - 2)/2 = 1 \ .$$

Die Lösung $\mathbf{x}$ der ursprünglichen Gleichung lautet also

$$\mathbf{x} \ = \ \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \ = \ \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \ .$$

Die Koeffizientenmatrix $\mathbf{A}$ sei nicht singulär, $\det\mathbf{A} \neq 0$. Dann ist (1) eindeutig lösbar. Das Prinzip des Gaußschen Algorithmus besteht nun darin, (1) in ein Gleichungssystem von oberer Dreiecksgestalt (mit einer oberen Dreiecksmatrix $\mathbf{U}$ als neuer Koeffizientenmatrix) überzuführen, welches dieselbe Lösung wie (1) besitzt:

$$\left.\begin{array}{rcrcrcrcrcl} u_{11}x_1 & + & u_{12}x_2 & + & u_{13}x_3 & + & \cdots & + & u_{1n}x_n & = & b'_1 \\ & & u_{22}x_2 & + & u_{23}x_3 & + & \cdots & + & u_{2n}x_n & = & b'_2 \\ & & & \ddots & & & & & \vdots & & \vdots \\ & & & & & & u_{n-1,n-1}x_{n-1} & + & u_{n-1,n}x_n & = & b'_{n-1} \\ & & & & & & & & u_{nn}x_n & = & b'_n \end{array}\right\} \qquad (2)$$

Falls alle $u_{ii} \neq 0$ sind, läßt sich (2), angefangen von der letzten Gleichung, von rückwärts her sukzessive auflösen:

$$\begin{array}{rcl} x_n & = & b'_n \,/\, u_{nn} \\ x_{n-1} & = & (b'_{n-1} - u_{n-1,n}x_n) \,/\, u_{n-1,n-1} \\ & \vdots & \\ x_2 & = & (b'_2 - u_{23}x_3 - \ldots - u_{2n}x_n) \,/\, u_{22} \\ x_1 & = & (b'_1 - u_{12}x_2 - u_{13}x_3 - \ldots - u_{1n}x_n) \,/\, u_{11} \end{array}$$

Allgemein erhält man für das *Rückwärtseinsetzen* ($i = n-1, n-2, \ldots, 1$):

$$\boxed{\begin{array}{rcl} x_n & = & b'_n \,/\, u_{nn} \\ x_i & = & (\, b'_i \,-\, \displaystyle\sum_{j=i+1}^{n} u_{ij}\, x_j \,) \,/\, u_{ii} \end{array}} \qquad (3)$$

Alle nicht singulären Gleichungssysteme lassen sich auf die Form (2) bringen: Die Lösungsmenge eines linearen Gleichungssystems ändert sich nämlich nicht, wenn man

- die Reihenfolge der Gleichungen vertauscht
- zu einer Gleichung das Vielfache einer anderen Gleichung addiert und eine der beiden Gleichungen durch diese Summe ersetzt.

Wir verwenden nun diese Eigenschaften, um das ursprüngliche Gleichungssystem (1) so umzuformen, daß alle Elemente von $\mathbf{A}$ unterhalb der Hauptdiagonale verschwinden. In der Ausgangsform bezeichnen wir das gegebene Gleichungssystem (1) mit

$$\mathbf{A}^{(0)} \cdot \mathbf{x} \;=\; \mathbf{b}^{(0)} \;.$$

Im *ersten Schritt* definieren wir $n-1$ Multiplikatoren für die Zeilen $k=2$ bis $k=n$,

$$m_k^{(1)} \;=\; \frac{a_{k1}^{(0)}}{a_{11}^{(0)}} \qquad k=2,3,\ldots,n \quad , \quad a_{11}^{(0)} \neq 0 \;\;,$$

multiplizieren die erste Gleichung mit $m_2^{(1)}$ und subtrahieren sie von der zweiten, dann multiplizieren wir die erste Gleichung mit $m_3^{(1)}$ und subtrahieren sie von der dritten usw. Dadurch wird x_1 von der 2. bis zur n-ten Gleichung eliminiert:

$$\left.\begin{array}{ccccccccc}
a_{11}^{(0)}x_1 & + & a_{12}^{(0)}x_2 & + & \cdots & + & a_{1n}^{(0)}x_n & = & b_1^{(0)} \\
0 & & a_{22}^{(1)}x_2 & + & \cdots & + & a_{2n}^{(1)}x_n & = & b_2^{(1)} \\
\vdots & & \vdots & & & & \vdots & & \vdots \\
0 & & a_{n2}^{(1)}x_2 & + & \cdots & + & a_{nn}^{(1)}x_n & = & b_n^{(1)}
\end{array}\right\} \qquad (4)$$

Die erste Zeile bleibt dabei unverändert und heißt *Pivotzeile*. Das aktuell zur Elimination benutzte Diagonalelement $a_{11}^{(0)}$ heißt *Pivotelement* (frz. pivot: Drehpunkt, Angelpunkt). Die neuen Koeffizienten $a_{kj}^{(1)}$ $(j=2,\ldots,n)$ und rechten Seiten $b_k^{(1)}$ der Zeilen $k=2,3,\ldots,n$ lauten

$$\begin{aligned}
a_{kj}^{(1)} &= a_{kj}^{(0)} - m_k^{(1)} a_{1j}^{(0)} \\
b_k^{(1)} &= b_k^{(0)} - m_k^{(1)}\, b_1^{(0)}
\end{aligned}$$

Dieses Verfahren wird nun mit den verbleibenden $(n-1)$ Gleichungen wiederholt, indem man wieder Multiplikatoren

$$m_k^{(2)} \;=\; \frac{a_{k2}^{(1)}}{a_{22}^{(1)}} \qquad k=3,4,\ldots,n \quad , \quad a_{22}^{(1)} \neq 0 \;\;,$$

definiert, die zweite Gleichung des Systems (4) als neue Pivotzeile verwendet, mit dem entsprechenden $m_k^{(2)}$ multipliziert, und sie dann von der k-ten Gleichung subtrahiert.
Im *i-ten Schritt* definiert man die Multiplikatoren

$$m_k^{(i)} \;=\; \frac{a_{ki}^{(i-1)}}{a_{ii}^{(i-1)}} \qquad k=i+1,\ldots,n \quad , \quad a_{ii}^{(i-1)} \neq 0 \;\;, \qquad (5)$$

und subtrahiert von der k-ten Gleichung $(k=i+1,\ldots,n)$ das $m_k^{(i)}$-fache der i-ten Gleichung (der aktuellen Pivotzeile). Die neuen Koeffizienten $a_{kj}^{(i)}$ $(j=i+1,\ldots,n)$ und rechten Seiten $b_k^{(i)}$ der Zeilen $k=i+1,\ldots,n$ sind dann

$$\boxed{\begin{aligned}
a_{kj}^{(i)} &= a_{kj}^{(i-1)} - m_k^{(i)} a_{ij}^{(i-1)} \\
b_k^{(i)} &= b_k^{(i-1)} - m_k^{(i)}\, b_i^{(i-1)}
\end{aligned}} \qquad (6)$$

Nach dem i-ten Schritt hat das System die Gestalt

$$\begin{array}{ccccccccccccc}
a_{11}^{(0)}x_1 & + & a_{12}^{(0)}x_2 & + & a_{13}^{(0)}x_3 & + & a_{14}^{(0)}x_4 & + & \cdots & + & a_{1n}^{(0)}x_n & = & b_1^{(0)} \\
 & & a_{22}^{(1)}x_2 & + & a_{23}^{(1)}x_3 & + & a_{24}^{(1)}x_4 & + & \cdots & + & a_{2n}^{(1)}x_n & = & b_2^{(1)} \\
 & & & \ddots & & & & & & & \vdots & & \vdots \\
 & & & & a_{ii}^{(i-1)}x_i & + & a_{i,i+1}^{(i-1)}x_{i+1} & + & \cdots & + & a_{in}^{(i-1)}x_n & = & b_i^{(i-1)} \\
 & & & & 0 & & a_{i+1,i+1}^{(i)}x_{i+1} & + & \cdots & + & a_{i+1,n}^{(i)}x_n & = & b_{i+1}^{(i)} \\
 & & & & \vdots & & \vdots & & & & \vdots & & \vdots \\
 & & & & 0 & & a_{n,i+1}^{(i)}x_{i+1} & + & \cdots & + & a_{nn}^{(i)}x_n & = & b_n^{(i)}
\end{array}$$

Damit wird x_i aus allen Zeilen $k = i+1, \ldots, n$ eliminiert und die ersten i Spalten enthalten unterhalb der Hauptdiagonale nur mehr Nullen.
Nach $n-1$ Schritten hat man schließlich die Form (2) mit einer oberen Dreiecksmatrix als Koeffizientenmatrix erreicht:

$$\left.\begin{array}{ccccccccc}
a_{11}^{(0)}x_1 & + & a_{12}^{(0)}x_2 & + & \cdots & + & a_{1n}^{(0)}x_n & = & b_1^{(0)} \\
 & & a_{22}^{(1)}x_2 & + & \cdots & + & a_{2n}^{(1)}x_n & = & b_2^{(1)} \\
 & & & \ddots & & & \vdots & & \vdots \\
 & & & & a_{n-1,n-1}^{(n-2)}x_{n-1} & + & a_{n-1,n}^{(n-2)}x_n & = & b_{n-1}^{(n-2)} \\
 & & & & & & a_{nn}^{(n-1)}x_n & = & b_n^{(n-1)}
\end{array}\right\} \qquad (7)$$

Die oberen Indizes (i) geben an, wie oft die Koeffizienten $a_{kj}^{(i)}$ bzw. Elemente $b_k^{(i)}$ der rechten Seite während der Eliminationsrechnung höchstens verändert wurden. Die Gesamtanzahl der Multiplikationen und Divisionen bei der Elimination und beim Rückwärtseinsetzen ist $(n^3 + 3n^2 - n)/3 = O(n^3)$. Der Zeitaufwand für die Lösung eines allgemeinen linearen Gleichungssystems steigt beim Gaußverfahren also mit der *dritten* Potenz der Anzahl n der Gleichungen: Bei einer Verdopplung von n erhöht sich die Rechenzeit auf ca. das Achtfache. Wegen der großen Anzahl von Rechenoperationen besteht die Gefahr, daß sich Rundungsfehler akkumulieren. Man sollte daher die Rechnungen nur in doppelter Genauigkeit (`double`) durchführen.

Um unnötigen Rechenaufwand (und damit eine Anhäufung von Rundungsfehlern) zu vermeiden, sollte man auf keinen Fall ein Gleichungssystem $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ lösen, indem man zuerst die zu $\mathbf{A}$ inverse Matrix $\mathbf{A}^{-1}$ berechnet (dies erfordert die Lösung von n linearen Gleichungssystemen mit den n Spalten der $(n \times n)$-Einheitsmatrix als rechte Seiten) und dann durch Multiplikation die Lösung $\mathbf{x} = \mathbf{A}^{-1} \cdot \mathbf{b}$ ermittelt!

Durchführbarkeit des Gaußschen Eliminationsverfahrens

Bei den Eliminationsschritten wurde vorausgesetzt, daß die jeweiligen Diagonalelemente (Pivotelemente) $a_{ii}^{(i-1)} \neq 0$ sind. Ist das nicht der Fall, wie zum Beispiel bei

$$\begin{aligned} x_2 &= 1 \\ x_1 + x_2 &= 2\,, \end{aligned}$$

so versagt der Gaußsche Algorithmus in seiner bisherigen Form. Wenn das System (1) nicht singulär ist, kann man durch *Zeilenvertauschungen* immer erreichen, daß das gerade aktuelle Pivotelement $a_{ii}^{(i-1)} \neq 0$ wird. Man wählt jedoch als neue Pivotzeile im i-ten Schritt nicht die erstbeste Zeile des verbleibenden Systems mit $a_{ki}^{(i-1)} \neq 0$ $(k = i, \ldots, n)$, sondern diejenige Zeile k_0 mit dem *betragsgrößten* Element in Spalte i (Abb. 1):

$$\left| a_{k_0 i}^{(i-1)} \right| = \max_{k=i,\ldots,n} \left| a_{ki}^{(i-1)} \right|$$

Die so gefundene neue Pivotzeile k_0 wird dann mit der aktuellen Pivotzeile i vertauscht. Damit bleiben die Multiplikatoren $|m_k^{(i)}| \leq 1$ und Rundungsfehler werden besonders klein gehalten. Dieses Verfahren heißt *Spaltenpivotsuche* oder *teilweise Pivotsuche*:

- Die Zeilenvertauschungen brauchen in einem Programm nicht tatsächlich ausgeführt zu werden, es genügt eine Indexvertauschung: Man definiert einen zusätzlichen Merkvektor (*Permutationsvektor*) `p[1],..,p[n]`, in den man alle Vertauschungen einträgt. Er wird zu Beginn mit `p[1]=1,..,p[n]=n` initialisiert und jeder Zeilenindex `i` wird durch `p[i]` ersetzt, z.B: `a[i][j]` → `a[p[i]][j]`.

- Ist **A** *diagonaldominant*, $|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|$ $(i = 1, \ldots, n)$, bzw. *symmetrisch* und *positiv definit*, so ist das Gaußverfahren ohne Pivotsuche durchführbar. Um Rundungsfehler zu minimieren, sollte aber eine Pivotsuche immer durchgeführt werden.

- In der Praxis kann es vorkommen, daß sich die Koeffizienten a_{ij} um Größenordnungen unterscheiden. Damit in solchen Fällen die Pivotsuche trotzdem wirkungsvoll ist, führt man eine sogenannte *skalierte Spaltenpivotsuche* durch: Man definiert für jede Zeile k einen Skalierungsfaktor $s_k := \max\limits_{j=1,\ldots,n} |a_{kj}|$ und verwendet zur Bestimmung der Pivotzeile im i-ten Schritt $\max\limits_{k=i,\ldots,n} (|a_{ki}^{(i-1)}|/s_k)$. Damit wird sichergestellt, daß das betragsgrößte Element jeder Zeile die *relative* Größe von 1 besitzt. Die Skalierung wird in einem Programm nur zum Vergleich bei der Suche nach Pivotzeilen durchgeführt; die Koeffizienten und rechten Seiten selbst bleiben dabei unverändert, um keine zusätzlichen Rundungsfehler zu erzeugen. Zeilenvertauschungen müssen im Vektor `s[]` der Skalierungsfaktoren s_k mitberücksichtigt werden: `s[i]` → `s[p[i]]`.

Ein *singuläres* Gleichungssystem wird vom Gaußschen Algorithmus daran erkannt, daß an einer bestimmten Stelle die Pivotsuche erfolglos ist, z.B. ist dann im i-ten Schritt $a_{ki}^{(i-1)} = 0$ für alle $i \leq k \leq n$. Durch Rundungsfehler kann es aber passieren, daß ein singuläres System als lösbar erscheint. In der Praxis erklärt man daher ein lineares Gleichungssystem für nicht behandelbar, wenn ein Pivotelement betragsmäßig kleiner als ein vorgegebenes ϵ wird (z.B. $\epsilon = 10^{-8}$). Die Matrix **A** ist dann *numerisch singulär*.

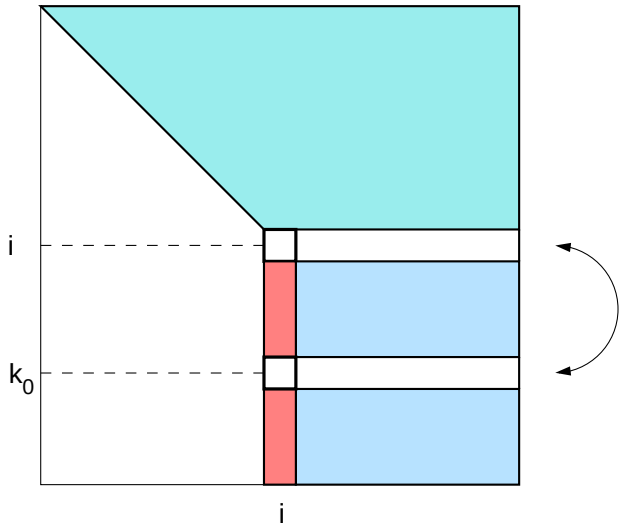

Abb. 1: Spaltenpivotsuche im i-ten Schritt.

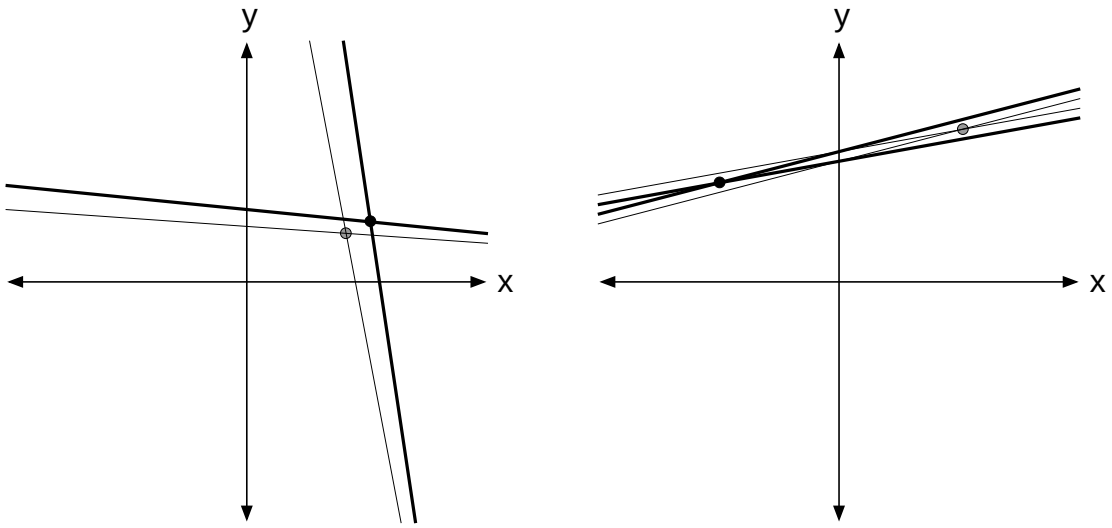

Abb. 2: Gut und schlecht konditioniertes 2×2-System.

LU-Zerlegung

Bei diesem Verfahren zur Lösung von (1) wird die Koeffizientenmatrix $\mathbf{A}$ zunächst in ein Produkt von zwei Dreiecksmatrizen zerlegt,

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} \ ,$$

wobei $\mathbf{L}$ eine untere (*lower*) Dreiecksmatrix und $\mathbf{U}$ eine obere (*upper*) Dreiecksmatrix ist:

$$\mathbf{L} = \begin{pmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{pmatrix} , \qquad \mathbf{U} = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix} .$$

Schreibt man nun das ursprüngliche Gleichungssystem $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ als

$$\mathbf{L} \cdot (\mathbf{U} \cdot \mathbf{x}) = \mathbf{b} \ ,$$

so gewinnt man die Lösung von (1) in einem zweistufigen Vorgang: Man löst zuerst

$$\mathbf{L} \cdot \mathbf{y} = \mathbf{b} \tag{8}$$

nach $\mathbf{y}$ und verwendet den Lösungsvektor $\mathbf{y}$ als rechte Seite in

$$\mathbf{U} \cdot \mathbf{x} = \mathbf{y} \ . \tag{9}$$

Wegen der Dreiecksform von $\mathbf{L}$ und $\mathbf{U}$ sind die beiden letzten Gleichungen leicht zu lösen. Man bestimmt zunächst den Hilfsvektor $\mathbf{y}$ in (8) durch *Vorwärtseinsetzen* ($i = 2, \ldots, n$):

$$\boxed{\begin{aligned} y_1 &= b_1 \,/\, l_{11} \\ y_i &= (\, b_i - \sum_{j=1}^{i-1} l_{ij}\, y_j \,) \,/\, l_{ii} \end{aligned}} \tag{10}$$

Den gesuchten Lösungsvektor $\mathbf{x}$ erhält man dann aus (9) durch *Rückwärtseinsetzen* wie im Gaußverfahren, vgl. (3). Vorteile der LU-Aufspaltung:

- Kennt man eine Faktorisierung $\mathbf{A} = \mathbf{L} \cdot \mathbf{U}$, so läßt sich die Lösung von $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$ in $O(n^2)$ Rechenoperationen gewinnen.
- Die Faktoren $\mathbf{L}$ und $\mathbf{U}$ sind unabhängig von $\mathbf{b}$ und können, einmal bestimmt, für verschiedene rechte Seiten $\mathbf{b}$ verwendet werden.
- Als Nebenprodukt einer LU-Zerlegung erhält man $\det\mathbf{A} = (-1)^k u_{11} u_{22} \ldots u_{nn}$, wobei k die Anzahl der Zeilenvertauschungen ist (siehe unten).

Wie findet man nun die Matrizen $\mathbf{L}$ und $\mathbf{U}$?

Wenn das Gaußsche Eliminationsverfahren ohne Pivotsuche durchführbar ist, dann liefert es eine Dreieckszerlegung von $\mathbf{A}$,

$$\mathbf{A} = \mathbf{L} \cdot \mathbf{U} \ ,$$

mit

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ l_{21} & 1 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ l_{n-1,1} & l_{n-1,2} & \cdots & 1 & 0 \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{pmatrix} \quad \text{und} \quad \mathbf{U} = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \cdots & a_{1n}^{(0)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n-1)} \end{pmatrix} ,$$

wobei die l_{ki} genau die im i-ten Schritt definierten Multiplikatoren $m_k^{(i)}$ sind, vgl. (5), und $\mathbf{U}$ die nach $(n-1)$ Eliminationsschritten erhaltene Koeffizientenmatrix $\mathbf{A}^{(n-1)}$ ist, vgl. (7). Die in $\mathbf{L}$ auftretenden Zahlen l_{ki} unterhalb der Hauptdiagonale können in dem (für die Elimination nicht benötigten) unteren Teil von $\mathbf{A}^{(i)}$ gespeichert werden (die $l_{ii} = 1$ werden nicht abgespeichert); damit wird die untere Dreiecksmatrix $\mathbf{L}$ im Laufe der Elimination spaltenweise aufgebaut.

Ist das Gaußsche Eliminationsverfahren nur mit Pivotsuche durchführbar, so liefert es eine Zerlegung von $\mathbf{P} \cdot \mathbf{A}$,

$$\mathbf{P} \cdot \mathbf{A} \ = \ \mathbf{L} \cdot \mathbf{U} \ ,$$

wo $\mathbf{P}$ die *Permutationsmatrix* ist, welche die im Laufe des Verfahrens durchgeführten Zeilenvertauschungen beschreibt.

Beispiel: Multiplikation einer (3×3)-Matrix $\mathbf{A}$ mit

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

vertauscht die erste und zweite Zeile von $\mathbf{A}$.

Man hat also in diesem Fall $(\mathbf{P} \cdot \mathbf{A}) \cdot \mathbf{x} \ = \ \mathbf{P} \cdot \mathbf{b} \ =: \ \mathbf{c}$ und löst

$$\mathbf{L} \cdot \mathbf{U} \cdot \mathbf{x} \ = \ \mathbf{c}$$

wie oben mit Vorwärts- und Rückwärtseinsetzen. Programmiertechnisch bedeutet das, daß man dabei auf die Zeilenindizes der ermittelten LU-Matrix und der originalen rechten Seite über den Permutationsvektor `p[]` zugreift; dazu muß während der Elimination die untere Dreiecksmatrix $\mathbf{L}$ in der ursprünglichen Zeilenordnung gespeichert werden, also `a[p[k]][j] = zmult`, und beim Vorwärts- und Rückwärtseinsetzen muß `p[]` zur Verfügung stehen.

Neben dem Gaußschen Algorithmus gibt es noch direkte Verfahren zur Berechnung von $\mathbf{L}$ und $\mathbf{U}$ (Crout, Doolittle, Banachiewicz), welche nur halb so viele Rechenoperationen benötigen. Allerdings ist hier die Pivotierung komplizierter als beim Gaußverfahren.

Fehler und Kondition

Die mit Hilfe direkter Methoden ermittelte Lösung $\mathbf{x}^*$ eines linearen Gleichungssystems ist meist nicht die exakte Lösung $\mathbf{x}$, da (1) Rundungsfehler akkumuliert werden, und da (2) Ungenauigkeiten in den Ausgangsgrößen $\mathbf{A}$ und $\mathbf{b}$ auch Ungenauigkeiten in der Lösung hervorrufen können. Leider ist das *Residuum* $\mathbf{r} := \mathbf{b} - \mathbf{A} \cdot \mathbf{x}^*$ als Maß für die Güte einer Näherungslösung $\mathbf{x}^*$ wenig geeignet: Ist nämlich die Länge $||\mathbf{r}||$ des Residuums klein, so kann man daraus nicht schließen, daß auch die Länge $||\mathbf{x} - \mathbf{x}^*||$ des unbekannten Fehlers $\mathbf{x} - \mathbf{x}^*$ klein ist. Zur Beurteilung der Güte von $\mathbf{x}^*$ ist also eine Probe, d.h. das Einsetzen der Näherungslösung $\mathbf{x}^*$ in das gegebene Gleichungssystem $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, nicht hinreichend.

Beispiel: Das System

$$\begin{pmatrix} 0 & 1 \\ \alpha & -1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

beschreibt den Schnittpunkt der Geraden $y = 0$ mit der Geraden $y = \alpha x$ und hat die exakte Lösung $\mathbf{x} = (0, 0)^T$. Für ein beliebiges $\mathbf{x}^* = (x^*, 0)^T$ ist das Residuum $\mathbf{r} = (0, -\alpha x^*)^T$. Ist etwa $x^* = 10^5$, $\alpha = 10^{-10}$, so ist zwar $||\mathbf{r}|| = \alpha x^* \approx 10^{-5}$, trotzdem ist der Fehler $||\mathbf{x} - \mathbf{x}^*|| = 10^5$. Die Lösung $\mathbf{x}^*$ hat also mit der exakten Lösung fast nichts zu tun, produziert aber beim Einsetzen nur ein kleines Residuum.

Es stellt sich heraus, daß die sogenannte *Kondition* $\kappa(\mathbf{A}) := ||\mathbf{A}||\,||\mathbf{A}^{-1}|| \geq 1$ der Koeffizientenmatrix $\mathbf{A}$ entscheidend dafür ist, ob man bei der numerischen Lösung des linearen Gleichungssystems mit Schwierigkeiten zu rechnen hat. $\kappa(\mathbf{A})$ ist der Faktor, um den die relativen Fehler in $\mathbf{A}$ und $\mathbf{b}$ zum relativen Fehler der Lösung verstärkt werden:

> Wenn kleine Änderungen in den Ausgangsdaten $\mathbf{A}$ und $\mathbf{b}$ große Änderungen in der Lösung $\mathbf{x}^*$ hervorrufen, wenn also $\kappa(\mathbf{A})$ groß ist, so spricht man von einem *schlecht konditionierten* System.

Im obigen Beispiel erhält man für kleines α für $\kappa(\mathbf{A}) \approx 2/\alpha$, d.h. das System ist für $\alpha \ll 1$ extrem schlecht konditioniert. Anschaulich bedeutet das, daß die beiden Geraden $g_1 : y = 0$ und $g_2 : y = \alpha x$ nahezu parallel sind, sodaß auf Grund des "schleifenden" Schnitts von g_1 und g_2 eine kleine Änderung in den Ausgangsdaten eine große Änderung im Schnittpunkt (in der Lösung) nach sich zieht (Abb. 2).

Das Paradebeispiel für eine schlecht konditionierte Matrix ist die sogenannte *Hilbertmatrix*

$$\mathbf{H}_n := \begin{pmatrix} 1 & \frac{1}{2} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n+1} \\ \vdots & & & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \cdots & \frac{1}{2n-1} \end{pmatrix} = (h_{ij}) = \left(\frac{1}{i+j-1}\right) \quad \text{für} \quad i, j = 1, \ldots, n\,.$$

Obwohl diese Matrix zunächst gutartig aussieht (symmetrisch, positiv definit, regulär), sind Gleichungssysteme mit dieser Matrix nur schwer zu lösen: $\kappa(\mathbf{H}_n)$ wächst exponentiell mit n (Abb. 3). Zum Beispiel hat das System $\mathbf{H}_n \cdot \mathbf{x} = \mathbf{b}$ für $b_i = \sum_{j=1}^{n} h_{ij}$ die exakte Lösung $\mathbf{x} = (1, \ldots, 1)^T$. Ein durch Rundungsfehler leicht verändertes System hat jedoch eine Lösung, die weit von $\mathbf{x}$ abweicht.

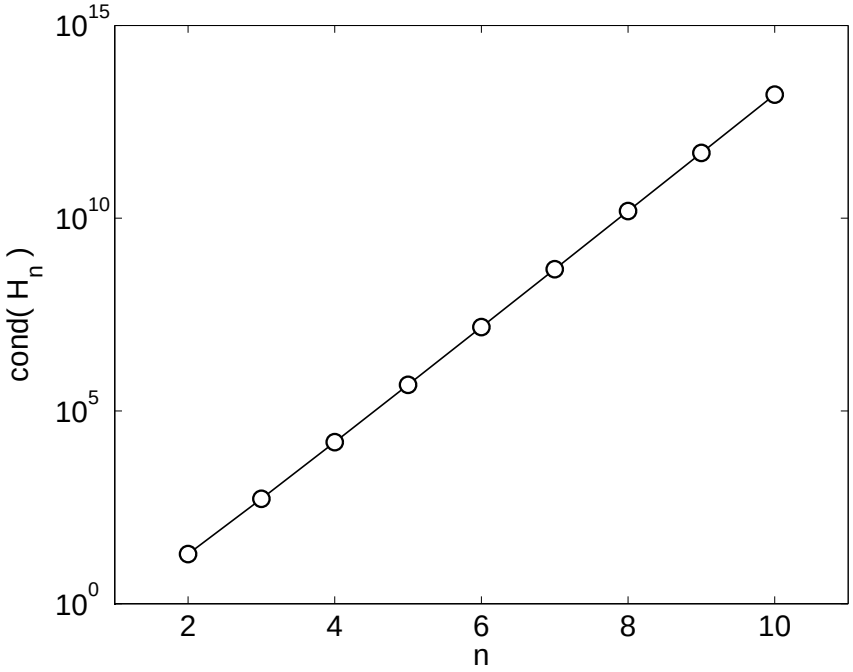

Abb. 3: Konditionszahlen der Hilbertmatrizen $\mathbf{H}_n$.