8 Numerische Verfahren für große und dünn besetze lineare Gleichungssysteme

Die bei den Simulationsmethoden entstehenden linearen Gleichungssysteme sind von sehr großer Dimension, *dünn besetzt*, d.h. der größte Teil der Matrixelemente ist gleich Null, und weisen im Allgemeinen keine Symmetrie auf.

Bei der MOS-Technologie kommen in der Praxis iterative Verfahren, wie sie im nächsten Abschnitt beschrieben werden zum Einsatz, während bei der Bipolar-Technik direkte Verfahren, denen der Gaussalgorithmus zugrunde liegt eingesetzt werden.

8.1 Iterative Verfahren

Grundlegend für die meisten klassischen Iterationsverfahren zur Lösung eines linearen Gleichunssystems

$$\boldsymbol{A}\boldsymbol{x} \; = \; \boldsymbol{b} \tag{1}$$

mit regulärer Matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ und den Vektoren $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$ ist das *Fixpunktverfahren*

$$\boldsymbol{x}^{k+1} \; = \; \Phi(\boldsymbol{x}^k)\,, \quad \text{für } k = 0, 1, 2, \ldots$$

mit der Verfahrensfunktion Φ. Dazu formuliert man (1) durch Aufsplitten der Matrix $\boldsymbol{A} = \boldsymbol{G} + (\boldsymbol{A} - \boldsymbol{G})$ mit regulärer Matrix $\boldsymbol{G} \in \mathbb{R}^{n \times n}$ um, in ein lineares Fixpunktproblem:

$$\boldsymbol{A}\boldsymbol{x} \; = \; \boldsymbol{b} \;\; \Leftrightarrow \;\; \boldsymbol{G}\boldsymbol{x} + (\boldsymbol{A} - \boldsymbol{G})\boldsymbol{x} \; = \; \boldsymbol{b} \; \Leftrightarrow \; \boldsymbol{x} \; = \; \underbrace{\boldsymbol{G}^{-1}(\boldsymbol{G} - \boldsymbol{A})}_{=:\boldsymbol{B}}\,\boldsymbol{x} + \underbrace{\boldsymbol{G}^{-1}}_{=:\boldsymbol{C}}\boldsymbol{b} \tag{2}$$

$\boldsymbol{I}$ bezeichnet dabei die Einheitsmatrix. Die Verfahrensfunktion Φ, die durch dieses *Splitting-Verfahren* entsteht, ist linear bezüglich $\boldsymbol{x}$ und besitzt $\boldsymbol{b}$ als Parameter:

$$\Phi(\boldsymbol{x}, \boldsymbol{b}) \;\; := \;\; \boldsymbol{B}\boldsymbol{x} + \boldsymbol{C}\boldsymbol{b}\,.$$

Satz 1 Das *Iterationsverfahren* $\boldsymbol{x}^{k+1} = \Phi(\boldsymbol{x}^k, \boldsymbol{b})$ mit $k = 0, 1, 2, \ldots$ konvergiert genau dann gegen die Lösung des linearen Gleichungssystems (1) für jeden Startwert $\boldsymbol{x}^0 \in \mathbb{R}^n$, wenn gilt:

$$\rho(\boldsymbol{B}) < 1\,.$$

Dabei ist $\rho(\boldsymbol{B}) := \max\{|\lambda|, \lambda \text{ Eigenwert von } \boldsymbol{B}\}$ der *Spektralradius* von $\boldsymbol{B}$.

Beweisidee:
Der Beweis beruht auf dem Banachschen Fixpunktsatz für kontrahierende Abbildungen und ist beispielsweise in [**?**] ausgeführt. Die Kontraktionbedingung führt dann auf $\rho(\boldsymbol{B}) < 1$. □

Eine einfache Möglichkeit das Splittingverfahren durchzuführen, ist die Wahl von $\boldsymbol{G} = \boldsymbol{I}$, sie führt auf das sogenannte *Richardson-Verfahren*

$$\boldsymbol{x}^{k+1} \;=\; (\boldsymbol{I} - \boldsymbol{A})\boldsymbol{x}^k + \boldsymbol{b}\,. \tag{3}$$

Um Konvergenz zu erreichen muss nach Satz 1 die Bedingung

$$\rho(\boldsymbol{I} - \boldsymbol{A}) = \max\{|1 - \lambda_{\max}(\boldsymbol{A})|, |1 - \lambda_{\min}(\boldsymbol{A})|\} < 1$$

erfüllt sein. Dies führt beispielsweise auf

$$-1 < 1 - \lambda_{\max}(\boldsymbol{A}) < 1 \;\Leftrightarrow\; 0 < \lambda_{\max}(\boldsymbol{A}) < 2\,,$$

eine Bedingung die für die wenigsten Matrizen erfüllt sein wird.

8.1.1 Gesamt- und Einzelschrittverfahren

Da das Richardson-Verfahren (3) nur auf eine sehr eingeschränkte Klasse von Matrizen angewendet werden kann, sind weitere besser einsetzbare Verfahren erforderlich.

In diesem Abschnitt splitten wir die Matrix $\boldsymbol{A} = (a_{ij})$ folgendermaßen auf

$$\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{R}\,,$$

dabei ist $\boldsymbol{D} := \mathrm{diag}(a_{11}, \ldots, a_{nn})$ die Diagonale von $\boldsymbol{A}$. Das linke untere Dreieck von $\boldsymbol{A}$ wird mit $\boldsymbol{L} = (\ell_{ij})$ bezeichet, d.h. $\ell_{ij} := a_{ij}$ für $i > j$ und Null sonst. Entsprechend wird das rechte obere Dreieck mit $\boldsymbol{R} = (r_{ij})$ bezeichnet, d.h. $r_{ij} := a_{ij}$ für $i < j$ und Null sonst. Die Bezeichnungsweise hier darf später **nicht** mit der $\boldsymbol{LR}$-Zerlegung der Matrix $\boldsymbol{A}$, die aus dem Gaußschen-Eliminationsverfahren entsteht, verwechselt werden.

Wählt man $\boldsymbol{G} = \boldsymbol{D}$, so erhält man das *Jacobi-Verfahren*

$$\boldsymbol{x}^{k+1} \;=\; -\boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{R})\boldsymbol{x}^k + \boldsymbol{D}^{-1}\boldsymbol{b}\,. \tag{4}$$

In Komponentenschreibweise mit $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ und $\boldsymbol{b} = (b_1, \ldots, b_n)^T$ besitzt das Jacobi-Verfahren (4) die Darstellung

$$x_i^{k+1} \;=\; \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^k \right) / a_{ii} \quad \text{für} \quad i = 1, \ldots, n\,. \tag{5}$$

Vorausgesetzt werden muss hier natürlich $a_{ii} \neq 0$, was äquivalent mit der Invertierbarkeit von $\boldsymbol{D}$ ist. Die Darstellung (5) ist auch folgendermaßen interpretierbar: Man nimmt sich die i-te Gleichung des Gleichungssystems (1)

$$\sum_{j=1}^{n} a_{ij} x_j \quad = \quad b_i \tag{6}$$

her und eliminiert aus dieser die Variable x_i. Die entstehende Fixpunktgleichung wird dann mit dem Fixpunktverfahren gelöst. Dabei ist es gleichgültig, in welcher Reihenfolge die Einzelgleichungen $i = 1, \ldots, n$ ausgewertet werden. Es werden die Komponenten der neuen Iterierten $\boldsymbol{x}^{k+1}$ aufgrund der gesamten Kenntnis der alten Komponenten, also der Iterierten $\boldsymbol{x}^k$ berechnet. Deshalb bezeichnet man das Jacobi-Verfahren auch als *Gesamtschrittverfahren*.

Satz 2 Das Gesamtschrittverfahren (4)/(5) konvergiert gegen die Lösung des linearen Gleichungssystems (1) für jeden Startwert $\boldsymbol{x}^0 = (x_1^0, \ldots, x_n^0)^T$ und jede rechte Seite $\boldsymbol{b}$, falls das *starke Zeilensummenkriterium*

$$\max_{i=1,\ldots,n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| \quad < \quad 1 \tag{7}$$

oder das *starke Spaltensummekriterium*

$$\max_{j=1,\ldots,n} \sum_{\substack{i=1 \\ i \neq j}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| \quad < \quad 1 \tag{8}$$

erfüllt ist.

Beweis:
Die Iterationsmatrix $\boldsymbol{B} = -\boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{R})$ ist von folgender Gestalt:

$$\boldsymbol{B} \quad = \quad - \begin{pmatrix} 0 & a_{12}/a_{11} & a_{13}/a_{11} & a_{14}/a_{11} & \cdots & a_{1n}/a_{11} \\ a_{21}/a_{22} & 0 & a_{23}/a_{22} & a_{24}/a_{22} & \cdots & a_{2n}/a_{22} \\ a_{31}/a_{33} & a_{32}/a_{33} & 0 & a_{34}/a_{33} & \cdots & a_{3n}/a_{33} \\ & & & \vdots & & \\ a_{n1}/a_{nn} & a_{n2}/a_{nn} & a_{n3}/a_{nn} & \cdots & a_{n,n-1}/a_{nn} & 0 \end{pmatrix}$$

Damit erhält man

$$||\boldsymbol{B}||_\infty = \max_{i=1,\ldots,n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| \quad \text{und} \quad ||\boldsymbol{B}||_1 = \max_{j=1,\ldots,n} \sum_{\substack{i=1 \\ i \neq j}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right| .$$

Gilt nun $||\boldsymbol{B}||_\infty < 1$ oder $||\boldsymbol{B}||_1 < 1$, so folgt die Behauptung, wegen $\rho(\boldsymbol{B}) \leq ||\boldsymbol{B}||_\infty$ bzw. $\rho(\boldsymbol{B}) \leq ||\boldsymbol{B}||_1$ nach Satz 1. □

Eine Matrix wird als *strikt diagonaldominant* bezeichnet, wenn sie das starke Zeilensummenkriterium erfüllt, denn:

$$\max_{i=1,\dots,n} \sum_{\substack{j=1\\ j\neq i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1 \iff \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| < |a_{ii}| \quad i = 1,\cdots,n$$

Wählt man $\boldsymbol{G} = \boldsymbol{D} + \boldsymbol{L}$, so erhält man das *Gauss-Seidel-Verfahren*

$$\boldsymbol{x}^{k+1} = -(\boldsymbol{D}+\boldsymbol{L})^{-1}\boldsymbol{R}x^k + (\boldsymbol{D}+\boldsymbol{L})^{-1}\boldsymbol{b}\,. \tag{9}$$

In Komponentenschreibweise besitzt das Gauss-Seidel-Verfahren (9) die Darstellung

$$x_i^{k+1} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k\right)/a_{ii} \quad \text{für} \quad i = 1,\dots,n\,. \tag{10}$$

Vorausgesetzt werden muss hier natürlich wiederum $a_{ii} \neq 0$, was äquivalent mit der Invertierbarkeit von $\boldsymbol{D} + \boldsymbol{L}$ ist. Die Darstellung (10) ist auch folgendermaßen interpretierbar: Man nimmt sich die i-te Gleichung (6) des Gleichungssystems (1) her und eliminiert aus dieser die Variable x_i. Im Gegensatz zum Gesamtschrittverfahren, werden die Komponenten nun einzeln und nacheinander in aufsteigender Reihenfolge berechnet. Dabei geht bei der Berechnung der i-ten Komponente im Schritt $k+1$ die jeweils aktuelle Version der j-ten Komponente ein, d.h. für die Komponeneten $1,\cdots i-1$ wird der im $k+1$-ten Schritt schon neu berechnete Wert verwendet, während die restlichen Komponenten noch in der Version des k-ten Schrittes einfließen. Man bezeichnet das Gauss-Seidel-Verfahren (9) bzw. (10) aus diesem Grund auch als *Einzelschrittverfahren*.

Da beim Einzelschrittverfahren die jeweils aktuelle Version der i-ten Komponente verwendet wird, besteht die Hoffnung, dass man eine schnellere Konvergenz als bei Gesamtschrittverfahren erhält. Man kann jedoch keine allgemeine Aussage dieser Art formulieren, da es Matrizen gibt, für die zwar das Gesamtschrittverfahren nicht aber das Einzelschrittverfahren konvergiert (vgl. [?]). Gesichert wird die Konvergenz des Einzelschrittverfahrens durch den dem Gesamtschrittverfahren analogen

Satz 3 Das Einzelschrittverfahren (9)/(10) konvergiert gegen die Lösung des linearen Gleichungssystems (1) für jeden Startwert $\boldsymbol{x}^0 = (x_1^0,\dots,x_n^0)^T$ und jede rechte Seite $\boldsymbol{b}$, falls die Matrix $\boldsymbol{A}$ strikt diagonaldominant ist.

Der Beweis zu Satz 3 kann bei [?] nachgelesen werden. Dort wird auch eine detailiertere Konvergenzuntersuchung duchgeführt.

Die folgende *Fehlerabschätzung* für die Iterierte $\boldsymbol{x}^k = (x_1^k, \cdots, x_n^k)^T$ und die Lösung $\boldsymbol{x} = (x_1, \cdots, x_n)^T$ des Gleichungssystems (1) geht auf den, den Beweisen zugrundeliegenden Fixpunktsatz zurück (vgl. [?]):

$$\max_{i=1,\cdots,n} \left| x_i^k - x_i \right| \quad \leq \quad \max_{i=1,\cdots,n} \frac{L^k}{1-L} \max_{i=1,\cdots,n} \left| x_i^1 - x_i^0 \right| \,, \tag{11}$$

mit $L := \max_{i=1,\dots,n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left| \frac{a_{ij}}{a_{ii}} \right|$.

8.1.2 Relaxationsverfahren

Eine Umformulierung des Fixpunktproblems (2) liefert:

$$\boldsymbol{x}^{k+1} \;=\; \boldsymbol{G}^{-1}(\boldsymbol{G}-\boldsymbol{A})\boldsymbol{x}^k + \boldsymbol{G}^{-1}\boldsymbol{b} \;=\; \boldsymbol{x}^k + \boldsymbol{G}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^k)\,. \tag{12}$$

Dies kann so interpretiert werden, dass der Vektor $\boldsymbol{x}^{k+1}$ aus dem Vektor $\boldsymbol{x}^k$ hervorgeht durch den Korrekturvektor $\boldsymbol{r}^k := \boldsymbol{G}^{-1}(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^k)$.

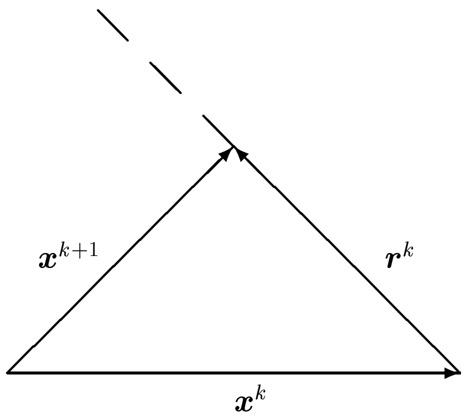

Bild: Korrekturvektor $\boldsymbol{r}^k$ und Relaxationsrichtung

Wird im Iterationsschritt nur die Richtung des Korrekturvektors $\boldsymbol{r}^k$ verwendet, also $\omega\boldsymbol{r}^k$, so spricht man von einem *Relaxationsverfahren.* Den Fall $\omega > 1$ nennt man *Überrelaxation*, $\omega = 1$ ist das Ausgangsverfahren, $0 < \omega < 1$ heißt *Unterrelaxation* und für $\omega = 0$ wird keine Korrektur durchgeführt.

Der *Relaxationsparameter* ω wird mit dem Ziel eingeführt die Konvergenz des Verfahrens zu beschleunigen.

Das *Richardson-Relaxationsverfahren*, d.h. $\boldsymbol{G} = \boldsymbol{I}$:

$$\boldsymbol{x}^{k+1} \;=\; \boldsymbol{x}^k + \omega(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^k) \;=\; (\boldsymbol{I} - \omega\boldsymbol{A})\boldsymbol{x}^k + \omega\boldsymbol{b} \tag{13}$$

in komponentenschreibweise

$$x_i^{k+1} \;=\; x_i^k + \omega\left(b_i - \sum_{j=1}^{n} a_{ij}x_j^k\right) \;. \tag{14}$$

Für das Richardson-Verfahren (3) lag Konvergenz höchstens für Matrizen mit der Eigenschaft $0 < \lambda_{\max}(\boldsymbol{A}) < 2$ vor. Anhand des relaxierten Richardson-Verfahrens soll nun untersucht werden, wie der Parameter ω zu wählen ist, damit sich Konvergenz einstellt (vgl. auch [?] und [?]).

Für die Untersuchung wird vorausgesetzt, dass $\boldsymbol{A}$ reelle Eigenwerte

$$\lambda_{\min} := \lambda_1 \le \lambda_2 \le \cdots \le \lambda_{m-1} \le \lambda_m =: \lambda_{\max}$$

hat, d.h. dass für das *Spektrum von* $\boldsymbol{A}$ gilt $\sigma(\boldsymbol{A}) \in \mathbb{R}$.

Dann besitzt die Iterationsmatrix $\boldsymbol{B} = \boldsymbol{I} - \omega\boldsymbol{A}$ (vgl. Satz 1) im Fall $\omega > 0$ die Eigenwerte

$$1 - \omega\lambda_{\min} = 1 - \omega\lambda_1 \ge 1 - \omega\lambda_2 \ge \cdots \ge 1 - \omega\lambda_{m-1} \ge 1 - \omega\lambda_m = 1 - \omega\lambda_{\max}$$

und im Fall $\omega < 0$ die Eigenwerte

$$1 - \omega\lambda_{\min} = 1 - \omega\lambda_1 \le 1 - \omega\lambda_2 \le \cdots \le 1 - \omega\lambda_{m-1} \le 1 - \omega\lambda_m = 1 - \omega\lambda_{\max}\,.$$

Nach Satz 1 liegt Konvergenz vor, wenn

$$\rho(\boldsymbol{I} - \omega\boldsymbol{A}) = \max\{|1 - \omega\lambda_{\min}|, |1 - \omega\lambda_{\max}|\} < 1$$

gilt. Äquivalent dazu ist

$$-1 < 1 - \omega\lambda_{\max} < 1 \wedge -1 < 1 - \omega\lambda_{\min} < 1 \Leftrightarrow 0 < \omega\lambda_{\min}, \omega\lambda_{\max} < 2\,.$$

Für den Fall $0 < \lambda_{\min}$, d.h. $\boldsymbol{A}$ besitzt nur positive reelle Eigenwerte, ergibt sich also Konvergenz, wenn man den Relaxationsparameter in folgender Weise wählt:

$$0 < \omega < \frac{2}{\lambda_{\max}}.$$

Die anderen Fälle führen bei analoger Vorgehensweise zu entsprechenden Bedingungen.

Der optimale Relaxationsparameter ω^* für den Fall $0 < \lambda_{\min}$, in dem Sinne, dass $\max\{|1 - \omega\lambda_{\min}|, |1 - \omega\lambda_{\max}|\}$ minimiert wird, ergibt sich anschaulich aus folgendem Bild,

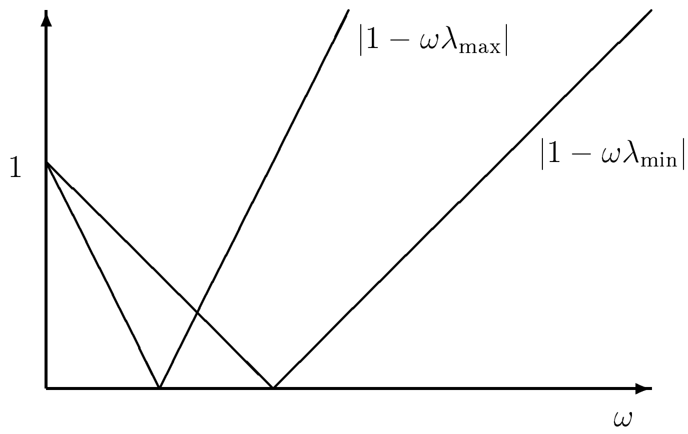

Bild: Die Funktionen $|1-\omega\lambda_{\max}|$ und $|1-\omega\lambda_{\min}|$

d.h. durch die Bedingung

$$-(1-\omega^*\lambda_{\max}) = 1-\omega^*\lambda_{\min} \quad \Longleftrightarrow \quad \omega^* = \frac{2}{\lambda_{\min}+\lambda_{\max}}\,.$$

Der die Konvergenz bestimmende Spektralradius ergibt sich dann durch

$$\rho(\boldsymbol{I}-\omega^*\boldsymbol{A}) \;=\; 1-\omega^*\lambda_{\min} \;=\; \frac{\lambda_{\max}-\lambda_{\min}}{\lambda_{\min}+\lambda_{\max}}\,.$$

Das *Gesamtschritt-Relaxationsverfahren*, d.h. $\boldsymbol{G}=\boldsymbol{D}$:

$$\begin{aligned} \boldsymbol{x}^{k+1} &= \boldsymbol{x}^k + \omega\boldsymbol{D}^{-1}(\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}^k) \;=\; (\boldsymbol{I}-\omega\boldsymbol{D}^{-1}\boldsymbol{A})\boldsymbol{x}^k + \omega\boldsymbol{D}^{-1}\boldsymbol{b} \\ &= (1-\omega)\boldsymbol{x}^k + \omega\boldsymbol{D}^{-1}(\boldsymbol{b}-(\boldsymbol{L}+\boldsymbol{R})\boldsymbol{x}^k) \qquad (15) \end{aligned}$$

in komponentenschreibweise

$$x_i^{k+1} \;=\; (1-\omega)x_i^k + \omega\left(b_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j^k\right)/a_{ii}\,. \qquad (16)$$

Untersuchungen zur optimalen Wahl des Relaxartionsparameters können in [?] nachgelesen werden.

Das *Einzelschritt-Relaxationsverfahren* in komponentenschreibweise:

$$x_i^{k+1} \;=\; (1-\omega)x_i^k + \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k\right)/a_{ii}\,. \qquad (17)$$

Das Einzelschritt-Relaxationsverfahren ist von der Idee her aus dem Gesamtschritt-Relaxationsverfahren in komponentenweiser Darstellung abgeleitet. Verwendet man den Korrekturvektoransatz $\boldsymbol{r}^k$ mit $\boldsymbol{G} = \boldsymbol{D} + \boldsymbol{L}$, so ergibt sich ein anderes Verfahren. Da für den optimalen Relaxationsparameter im Sinne der Konvergenzbeschleunigung $\omega \in (1, 2)$ gilt (vgl. [**?**]), wird das Verfahren (17) auch als *SOR-Verfahren (succsessive overrelaxation method)* bezeichnet.

8.1.3 Überblick über weitere Verfahren

Ein Überblick über weitere iterative Verfahren wird in [**?**] gegeben. Ohne auf Details einzugehen sollen hier einige der Projektionsmethoden und Krylov-Unterraumverfahren genannt werden.

- Arnoldi-Algorithmus (1951)
- GMRES-Verfahren (1986, Saad, Schulz)
- BiCG-Verfahren (1975, Fletcher)
- CGS-Verfahren (1989, Sonneveld)
- BiCGSTAB (1992, van der Vorst)

8.1.4 Nutzung der dünnen Besetztheit

Die dargestellten iterativen Verfahren erfordern je Iterationsschritt, d.h. bei der Berechnung von $\boldsymbol{x}^{k+1}$ aus $\boldsymbol{x}^k$, eine Matrix-Vektor-Multiplikation. Bei der Berechnung der i-ten Komponente von $\boldsymbol{x}^{k+1}$ ist also im Wesentlichen das Skalarprodukt $< \boldsymbol{a}^i, \boldsymbol{x}^k >$ zu berechnen, wobei $\boldsymbol{a}^i$ die i-te Zeile von $\boldsymbol{A}$ ist. Treten in $\boldsymbol{a}^i$ nur wenige von Null verschiedene Elemente auf, so lässt sich der Rechenaufwand erheblich reduzieren.

Am Beispiel des Gesamtschrittverfahrens in komponentenweiser Darstellung

$$x_i^{k+1} = \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^k \right) / a_{ii} \quad \text{für} \quad i = 1, \ldots, n \,. \tag{18}$$

soll der Rechenaufwand betrachtet werden.

Die Anzahl der Operationen des Gesamtschrittverfahrens (18) lautet:

Komponenten x (Multipl. + Subtr. + Div.) $= n(n - 1 + 1 + 1) = O(n^2)$.

Ist die Matrix $\boldsymbol{A}$ dünn besetzt, sind beispielsweise je Zeile maximal $m \leq K$ der Matrixelemente a_{ij} von Null verschieden, wobei K eine (kleine) Konstante ist, so sind diese nicht in der Summe von (18) zu berücksichtigen, und man erhält nur $n(K+1+1) = O(n)$ Operationen. Jeder Iterationsschritt ist also selbst bei sehr großen Systemen schnell durchführbar.

Am folgenden Beispiel soll, wiederum für das Gesamtschrittverfahren, eine mögliche kompakte Speicherung der von Null verschiedenen Nichtdiagonalelemente der Matrix $\boldsymbol{A}(1:n, 1:n)$ in der Matrix $\tilde{\boldsymbol{A}}(1:n, 1:m)$, in Verbindung mit der Indexmatrix $\boldsymbol{S}(1:n, 1:m)$ dargestellt werden:

$$\underbrace{\begin{pmatrix} 11 & 5 & 0 & -2 & 0 & 0 & 0 & 0 \\ 3 & -21 & -7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 3 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 74 & -8 & 4 & 0 & 0 \\ 0 & 7 & 0 & 0 & -6 & 9 & 0 & 0 \\ 6 & 0 & -5 & 0 & 0 & 33 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 57 & -1 \\ 0 & 0 & 2 & 0 & 0 & 0 & 4 & -89 \end{pmatrix}}_{\boldsymbol{A}} \rightarrow \underbrace{\begin{pmatrix} 5 & -2 \\ 3 & -7 \\ 3 & 1 \\ -8 & 4 \\ 7 & 9 \\ 6 & -5 \\ 3 & -1 \\ 2 & 4 \end{pmatrix}}_{\tilde{\boldsymbol{A}}}, \underbrace{\begin{pmatrix} 2 & 4 \\ 1 & 3 \\ 4 & 8 \\ 5 & 6 \\ 2 & 6 \\ 1 & 3 \\ 4 & 8 \\ 3 & 7 \end{pmatrix}}_{\boldsymbol{S}}.$$

Das Gesamtschrittverfahren mit reduziertem Rechenaufwand lautet dann

$$x_i^{k+1} = \left(b_i - \sum_{r=1}^{m} \tilde{a}(i,r) x_{s(i,r)}^k \right) / a(i,i) \quad \text{für} \quad i = 1, \ldots, n\,.$$

8.2 Direkte Verfahren

Als direktes Verfahren kommt nur der Gauss-Algorithmus in Verbindung mit der LR-Zerlegung zur Anwendung. Im Standardfall sind, d.h. ohne Ausnutzung spezieller Struktureigenschaften und bei einer Dimension von n des Gleichungssystem, $O(n^3)$ wesentliche Operationen (= Additionen und Multiplikationen) erforderlich. Bei sehr großer Dimension ist dieses Verfahren daher auf heutigen Rechnern nicht durchführbar. Wird hingegen die dünne Besetzung der Systemmatrix ausgenutzt, so wird eine Berechnung insbesondere auch auf Parallelrechnern möglich.

8.2.1 Der Gauss Algorithmus

Ausgangspunkt der Untersuchungen ist wiederum das Gleichungssystem (1)

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

mit regulärer Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ und gegebener rechter Seite $\boldsymbol{b} \in \mathbb{R}^n$. Das Gausssche Eliminationsverfahern liefert für $\boldsymbol{A}$ bzw. $\boldsymbol{PA}$, wobei $\boldsymbol{P}$ eine $n \times n$ Permutationsmatrix ist, eine LR-Zerlegung

$$\boldsymbol{PA} = \boldsymbol{LR}$$

mit

$$\boldsymbol{L} = \begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ & \vdots & & \ddots & \\ l_{n1} & & \cdots & l_{n,n-1} & 1 \end{pmatrix} \tag{19}$$

und

$$\boldsymbol{R} = \begin{pmatrix} r_{11} & & \cdots & r_{1n} \\ 0 & r_{22} & & r_{2n} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & r_{n,n} \end{pmatrix}. \tag{20}$$

Aus

$$\boldsymbol{PAx} = \boldsymbol{LRx} = \boldsymbol{Pb}$$

erhält man durch *Vorwärtseinsetzen*

$$\boldsymbol{Ly} = \boldsymbol{Pb}$$

und *Rückwärtseinsetzen*

$$\boldsymbol{Rx} = \boldsymbol{y}$$

die Lösung des Gleichungssystems (1).

8.2.2 LR-Zerlegung

In diesem Abschnitt wird vorausgesetzt, dass keine Permutationsmatrix $\boldsymbol{P}$ erforderlich ist, also bereits für $\boldsymbol{A}$ eine LR-Zerlegung existiert. Der Gausssche Algorithmus liefert die LR-Zerlegung in $n-1$ Stufen.

$$\begin{aligned} \boldsymbol{A}^{(1)} &:= \boldsymbol{A} \\ \boldsymbol{A}^{(k+1)} &:= \boldsymbol{L}_k \boldsymbol{A}^{(k)} \quad k = 1, \cdots, n-1 \end{aligned} \tag{21}$$

mit

$$\boldsymbol{A}^{(k)} = \begin{pmatrix} a_{11}^{(1)} & & \cdots & & a_{1n}^{(1)} \\ & \ddots & & & \vdots \\ & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \boldsymbol{0} & & \vdots & & \vdots \\ & & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{pmatrix}.$$

und

$$\boldsymbol{L}_k = \begin{pmatrix} 1 & & & & & \\ & \ddots & & \boldsymbol{0} & & \\ & & 1 & & & \\ & & -l_{k+1,k} & 1 & & \\ & \boldsymbol{0} & \vdots & \boldsymbol{0} & \ddots & \\ & & -l_{nk} & & & 1 \end{pmatrix}.$$

Mit

$$\boldsymbol{R} := \boldsymbol{A}^{(n)}$$

erhält man aus (21) nämlich

$$\boldsymbol{R} = \boldsymbol{L}_{n-1} \cdots \boldsymbol{L}_1 \boldsymbol{A}$$

und damit

$$\boldsymbol{A} = \underbrace{\boldsymbol{L}_1^{-1} \cdots \boldsymbol{L}_{n-1}^{-1}}_{=:\boldsymbol{L}} \boldsymbol{R}.$$

Die Berechnungsvorschriften aus (21) mit k als *Stufenindex*, i als *Zeilenindex* und j als *Spaltenindex* sehen im Einzelnen wie folgt aus:

Berechne für $k = 1, \cdots, n-1$

$$l_{ik} := a_{ik}^{(k)} / a_{kk}^{(k)}, \quad i = k+1, \cdots, n \tag{22}$$

$$a_{ij}^{(k+1)} := \begin{cases} a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}, & j = k, \cdots, n \\ a_{ij}^{(k)}, & \text{sonst.} \end{cases}$$

Bemerkungen

a) Da vorausgesetzt wurde, dass die LR-Zerlegung von $\boldsymbol{A}$ existiert, ist die Division in (22) durchführbar, d.h. $a_{kk}^{(k)} \neq 0$ für $k = 1, \cdots, n$.

b) Nachdem die l_{ik} in (22) berechnet worden sind, können sie im nicht mehr benötigten Element a_{ik} von $\boldsymbol{A}$ abgespeichert werden, d.h. $\boldsymbol{L}$ wird am Ende des Gaussschen Algorithmus auf dem unteren Dreieck von $\boldsymbol{A}$ abgespeichert sein.

c) Ist das Gleichungsssystem nur für eine rechte Seite $\boldsymbol{b}$ zu lösen, so kann das Vorwärtseinsetzen, d.h.

$$\begin{aligned} \boldsymbol{b}^{(1)} &:= \boldsymbol{b} \\ \boldsymbol{b}^{(k+1)} &:= \boldsymbol{L}_k \boldsymbol{b}^{(k)} \quad k = 1, \cdots, n-1 \\ \boldsymbol{y} &= \boldsymbol{b}^{(n-1)} \end{aligned}$$

in (22) integriert werden:

$$b_i^{(k+1)} := b_i^{(k)} - l_{ik} b_k^{(k)} \quad i = k+1, \cdots, n.$$

8.2.3 LR-Zerlegung mit hüllenorientierte Rechentechnik

Um die *Hülle* einer Matrix $\boldsymbol{A}$ zu bestimmen wird zunächst in der i-ten Zeile der kleinste Spaltenindex $h(i)$ ermittelt, für den $a_{i,h(i)} \neq 0$ gilt. Die Hülle von $\boldsymbol{A}$ umfasst nun alle a_{ij} mit $j \geq h(i)$.

Am obigen Beispiel ergibt sich die Hülle folgendermaßen

$$(h(1), h(2), h(3), h(4), h(5), h(6), h(7), h(8)) = (1, 1, 3, 4, 2, 1, 4, 3)$$

$$\underbrace{\begin{pmatrix} 11 & 5 & 0 & -2 & 0 & 0 & 0 & 0 \\ 3 & -21 & -7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 3 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 74 & -8 & 4 & 0 & 0 \\ 0 & 7 & 0 & 0 & -6 & 9 & 0 & 0 \\ 6 & 0 & -5 & 0 & 0 & 33 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 57 & -1 \\ 0 & 0 & 2 & 0 & 0 & 0 & 4 & -89 \end{pmatrix}}_{\boldsymbol{A}} \rightarrow \underbrace{\begin{pmatrix} 11 & 5 & 0 & -2 & 0 & 0 & 0 & 0 \\ 3 & -21 & -7 & 0 & 0 & 0 & 0 & 0 \\ & & 5 & 3 & 0 & 0 & 0 & 1 \\ & & & 74 & -8 & 4 & 0 & 0 \\ & 7 & 0 & 0 & -6 & 9 & 0 & 0 \\ 6 & 0 & -5 & 0 & 0 & 33 & 0 & 0 \\ & & & 3 & 0 & 0 & 57 & -1 \\ & & 2 & 0 & 0 & 0 & 4 & -89 \end{pmatrix}}_{\text{Hülle von } \boldsymbol{A}}$$

Aus der Berechnungsvorschrift (22) ergibt sich, dass der Gaussalgorithmus nur auf der Hülle durchführbar ist. Denn man erhält für $k < h(i)$ induktiv

$$a_{ik}^{(k)} = 0 \Rightarrow l_{ik} = 0 \Rightarrow a_{ik}^{(k+1)} = 0\,.$$

Der nur auf der Hülle operierende Algorithmus (22) lautet daher:

Führe für $k = 1, \cdots, n-1$ und $k \geq h(i)$ folgende Berechnung durch:

$$l_{ik} := a_{ik}^{(k)} / a_{kk}^{(k)}\,, \quad i = k+1, \cdots, n \tag{23}$$

$$a_{ij}^{(k+1)} := \begin{cases} a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}\,, & j = k, \cdots, n \\ a_{ij}^{(k)}\,, & \text{sonst.} \end{cases}$$

Als *fill in* bezeichnet man den Sachverhalt, dass die Nullelemente a_{ij} innerhalb der Hülle von $\boldsymbol{A}$ im Laufe des Algorithmus im Allgemeinen zu Nichtnullelementen umgeformt werden. Im Sinne einer Reduktion der Anzahl der Rechenoperationen sind diese Nullen einer dünn besetzten Matrix somit nicht nutzbar. Man wird also vor dem Start des Gauss-Algorithmus bestrebt sein, Verfahren zur Hüllenreduktion anzuwenden.

Die hüllenorientierte Rechentechnik wird am Beispiel symmetrischer Matrizen ausführlich in [**?**] dargestellt.