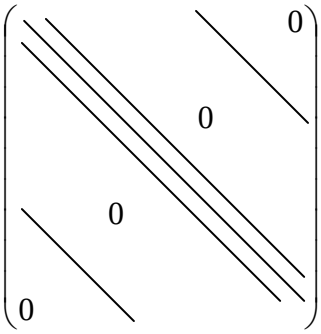
§18 Iterative Verfahren für lineare Gleichungssysteme

18.1 Motivation

- Viele praktische Probleme führen auf sehr große lineare Gleichungssysteme, bei denen die Systemmatrix **dünn besetzt** ist, d.h. nur wenige Einträge von 0 verschieden sind.
  Beispiel:



Pentadiagonalmatrix, z.B. bei 2D – Wärmeleitungsproblem

- Aus Speicherplatzgründen will man oft nur die von 0 verschiedenen Elemente abspeichern. Eine $n \times n$-Pentadiagonalmatrix mit $n = 100.000$ benötigt bei 8 Byte/Eintrag und vollem Abspeichern 80 GByte, bei effizientem Abspeichern 800 KByte.
- Direkte Verfahren wie der Gauß-Algorithmus können die Nullen auffüllen und zu einem hohen Speicherplatzbedarf führen. Zudem ist oft auch der Rechenaufwand zu hoch:
  $O(n^3)$ Operationen für $n \times n$-Gleichungssystem
- Man sucht effiziente iterative Verfahren, die kaum zusätzlichen Speicherplatz benötigen und nach wenigen Schritten eine brauchbare Approximation liefern.

18.2 Klassische iterative Verfahren

Gegeben: $A \in \mathrm{Mat}_{n\times n}(\mathbb{R})$, $b \in \mathbb{R}^n$

Gesucht: $x \in \mathbb{R}^n$ mit $Ax = b$

Falls $A = S - T$ mit einer einfach zu invertierenden Matrix S (z.B. Diagonalmatrix, Dreiecksmatrix), kann man $Ax = b$ umformen in $Sx = Tx + b$, und mit einem Startvektor $x^{(0)}$ die Iteration $\boxed{Sx^{(k+1)} = Tx^{(k)} + b}$ $k = 0,1,2,...$ ausprobieren.

Wie sehen geeignete Splittings $A = S - T$ aus?
Sei $A = D - L - R$ Aufspaltung in eine Diagonalmatrix D, eine strikte untere Dreiecksmatrix L und eine strikte obere Dreiecksmatrix R.

a) **Jacobi-Verfahren (Gesamtschritt-Verfahren)**
(nicht verwechseln mit Jacobi-Verfahren zur Eigenwertapproximation, vgl. §17)
$S := D$, $T := L + R$

In jeder Iteration löst man nur das Diagonalsystem

$$Dx^{(k+1)} = (L+R)x^{(k)} + b$$

explizit: $$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(-\sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x_j^{(k)} + b_i \right) \qquad (i = 1,\ldots,n)$$

Beispiel:

$$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 4 \end{pmatrix}, \qquad x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$x_1^{(k+1)} = \frac{1}{2}\left[x_2^{(k)} + 3\right]$$

$$x_2^{(k+1)} = \frac{1}{2}\left[x_1^{(k)} + 4\right]$$

bei vierstelliger Genauigkeit:

k	$x_1^{(k)}$	$x_2^{(k)}$
0	0	0
1	1,5	2
2	2,5	2,75
3	2,875	3,25
4	3,125	3,438
5	3,219	3,568
6	3,282	3,610
7	3,305	3,641
8	3,321	3,653
9	3,327	3,661
10	3,331	3,664
11	3,332	3,666
12	3,333	3,666

exakte Lösung:

$$x_1 = 3\frac{1}{3}, \quad x_2 = 3\frac{2}{3}$$

Das Jacobi-Verfahren ist gut geeignet für Parallelrechner, da $x_i^{(k+1)}$ nicht von $x_j^{(k+1)}$ abhängt.

b) **Gauß-Seidel-Verfahren (Einzelschritt-Verfahren)**

$S := D - L$, $\qquad T := R$

In jeder Iteration wird das Dreieckssystem $(D-L)x^{(k+1)} = Rx^{(k)} + b$ durch einfache Vorwärtssubstitution gelöst:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(-\sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{r=i+1}^{n} a_{ir} x_r^{(k)} + bi \right) \qquad (i = 1,\ldots,n)$$

d.h. neue Werte werden in der Iteration verwandt, sobald sie berechnet wurden.

Beispiel:

$$A=\begin{pmatrix}2 & -1\\ -1 & 2\end{pmatrix},\quad b=\begin{pmatrix}3\\4\end{pmatrix},\qquad x^{(0)}=\begin{pmatrix}0\\0\end{pmatrix}$$

k	$x_1^{(k)}$	$x_2^{(k)}$
0	0	0
1	1,5	2,75
2	2,875	3,348
3	3,174	3,587
4	3,294	3,647
5	3,324	3,662
6	3,331	3,666

$$x_1^{(k+1)}=\frac{1}{2}\left[x_2^{(k)}+3\right]$$
$$x_2^{(k+1)}=\frac{1}{2}\left[x_1^{(k+1)}+4\right]$$

Konvergiert etwa doppelt so schnell wie Jacobi.

c) **SOR-Verfahren (SOR = successive overrelaxation)**

Beschleunigung des Gauß-Seidel-Verfahrens durch Extrapolation

$x^{(k+1)}=x^{(k)}+\varpi\left(\tilde{x}^{(k+1)}-x^{(k)}\right)$

wobei $\varpi\in(1,2)$ und $\tilde{x}^{(k+1)}$ die Gauß-Seidel-Iteration zu $x^{(k)}$ ist.

Für große Gleichungssysteme kann damit die Iterationszahl für ein geeignetes w oft um eine Zehnerpotenz gesenkt werden.

Satz 18.3 (Allgemeine Konvergenzbedingung)

Seien $A\in \mathrm{Mat}_{n\times n}(\mathbb{R})$ invertierbar, $b\in\mathbb{R}^n$ und $A=S-T$ mit einer invertierbaren Matrix S. Falls $\left\|S^{-1}T\right\|<1$ in einer zu einer Vektornorm verträglichen Matrixnorm, so konvergiert das Iterationsverfahren $x^{(k+1)}=S^{-1}\left(Tx^{(k)}+b\right)$ für jeden beliebigen Startvektor $x^{(0)}\in\mathbb{R}^n$ gegen die exakte Lösung von $Ax=b$.

Beweis:

Aus $Sx=Tx+b$ und

$Sx^{(k+1)}=Tx^{(k)}+b$ folgt durch Subtraktion für den Fehler $e^{(k)}:=x-x^{(k)}$:

$$Se^{(k+1)}=Te^{(k)} \qquad \Rightarrow e^{(k+1)}=S^{-1}Te^{(k)}=\ldots=\left(S^{-1}T\right)^k e^{(0)}$$

$$\Rightarrow \left\|e^{(k+1)}\right\|=\left\|\left(S^{-1}T\right)^k e^{(0)}\right\|\le\left\|\left(S^{-1}T\right)^k\right\|\cdot\left\|e^{(0)}\right\|$$

$$=\left\|S^{-1}T\right\|^k\cdot\left\|e^{(0)}\right\|$$

Für $\left\|S^{-1}T\right\|<1$ konvergiert der Fehler gegen 0. q.e.d.

Man kann zeigen:

Satz 18.4 (Spezielle Konvergenzaussagen)

a) Ist die Systemmatrix $A = (a_{ij}) \in \mathrm{Mat}_{n\times n}(\mathsf{R})$ strikt **diagonaldominant** (d.h. $|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|$), so konvergieren Jacobi- und Gauß-Seidel-Verfahren.

b) Für positiv definite (oder negativ definite) Matrizen konvergiert das SOR-Verfahren für $\varpi \in (0,2)$. Insbesondere konvergiert damit das Gauß-Seidel-Verfahren ($\varpi = 1$).