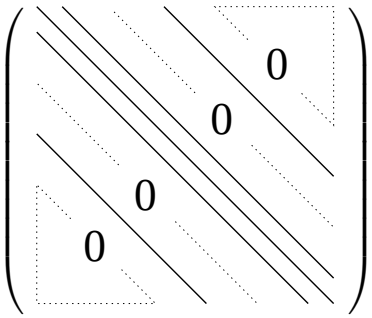
38 Iterative Verfahren für lineare Gleichungssysteme

38.1 Motivation

- Viele praktische Probleme führen auf sehr große lineare Gleichungssysteme, bei denen die Systemmatrix **dünn besetzt** ist, d. h. nur wenige von Null verschiedene Einträge aufweist.

  Beispiel: Pentadiagonalmatrix bei der Berechnung von 2-D-Diffusionsfiltern in der Bildverarbeitung

- Aus Speicherplatzgründen will man oft nur die von 0 verschiedenen Elemente abspeichern.

  Beispiel: Ein Grauwertbild mit 512×512 Pixeln führt zu $512^2 = 262\,144$ Gleichungen mit ebenso vielen Unbekannten. Bei 4 Byte je Eintrag (Datentyp `float`) und vollem Abspeichern benötigt die Pentadiagonalmatrix $4 \cdot 512^4$ Byte ≈ 275 GB, bei effizientem Abspeichern nur $5 \cdot 4 \cdot 512^2$ Byte $\approx$ 5,2 MB!

- Direkte Verfahren wie der Gauß-Algorithmus können die Nullen auffüllen und so zu einem enormen Speicherplatzbedarf führen.

  Zudem ist ihr Rechenaufwand oft zu hoch: $\mathrm{O}(n^3)$ Operationen für ein $n \times n$-Gleichungssystem.

- Daher verwendet man oft iterative Näherungsverfahren, die kaum zusätzlichen Speicherplatz benötigen und nach wenigen Schritten eine brauchbare Approximation liefern.

38.2 Grundstruktur klassischer iterativer Verfahren

Gegeben: $A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n$

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

Falls $A = S - T$ mit einer einfach zu invertierenden Matrix S ist (z. B. Diagonalmatrix, Dreiecksmatrix), so kann man $A\,x = b$ umformen in

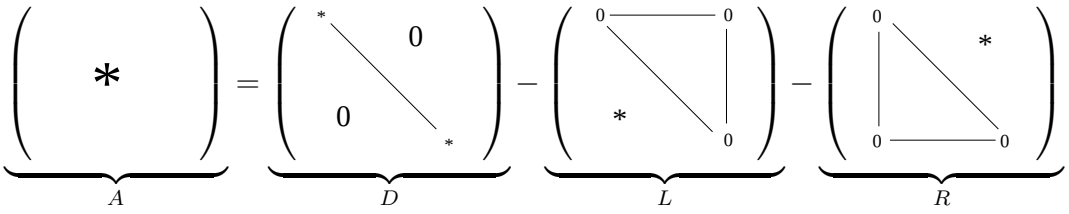
$$S\,x = T\,x + b$$

und mit einem Startvektor $x^{(0)} \in \mathbb{R}^n$ die Fixpunktiteration

$$S\,x^{(k+1)} = T\,x^{(k)} + b\,, \quad k = 0, 1, 2, \ldots$$

anwenden.

Wir wollen nun drei verschiedene Aufspaltungen $A = S - T$ untersuchen. Dazu sei $A = D - L - R$ eine Aufspaltung von A in eine Diagonalmatrix D, eine strikte untere Dreiecksmatrix (also eine untere Dreiecksmatrix, die auf der Diagonale nur Nullen hat) und eine strikte obere Dreiecksmatrix R.



38.3 Das Jacobi-Verfahren (Gesamtschrittverfahren)

Hier wählt man $S := D$ und $T := L + R$. In jeder Iteration wird also nur das Diagonalsystem

$$D\,x^{(k+1)} = (L + R)\,x^{(k)} + b$$

nach $x^{(k+1)}$ aufgelöst, d. h. es wird nur durch die Diagonalelemente dividiert:

$$x^{(k+1)} = D^{-1}\Big((L + R)\,x^{(k)} + b\Big)$$

oder explizit:

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

38.4 Beispiel

$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$, $b = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$ führt auf das Diagonalsystem

$$2x_1^{(k+1)} = x_2^{(k)} + 3$$
$$2x_2^{(k+1)} = x_1^{(k)} + 4 \, .$$

Bei vierstelliger Genauigkeit und Startvektor $x^{(0)} = (0, 0)^{\mathrm{T}}$ erhält man

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	3,563
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

Die exakte Lösung ist $x_1 = 3\,\frac{1}{3}$, $x_2 = 3\,\frac{2}{3}$.

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

38.5 Das Gauß-Seidel-Verfahren (Einzelschrittverfahren)

Hier setzt man $S := D - L$, $T := R$.

In jeder Iteration wird daher das Dreieckssystem

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

durch einfache Rückwärtssubstitution gelöst:

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

d. h. neue Werte werden weiterverwendet, sobald sie berechnet wurden.

38.6 Beispiel

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

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

k	$x_1^{(k)}$	$x_2^{(k)}$
0	0	0
1	1,5	2,75
2	2,875	3,438
3	3,219	3,609

k	$x_1^{(k)}$	$x_2^{(k)}$
4	3,305	3,652
5	3,326	3,663
6	3,332	3,666
7	3,333	3,666

Das Verfahren konvergiert also etwa doppelt so schnell wie das Jacobi-Verfahren.

38.7 Überrelaxation und SOR-Verfahren

Grundgedanke der Überrelaxation: Aufeinanderfolgende Iterationsschritte des Gauß-Seidel-Algorithmus korrigieren den Lösungsvektor oftmals näherungsweise „in die gleiche Richtung“. Extrapoliert man also die Korrektur im einzelnen Iterationsschritt (geht also immer „etwas weiter“ in die vorgegebene Richtung, als der Gauß-Seidel-Algorithmus vorgibt), kann unter Umständen die Konvergenz des Verfahrens beschleunigt werden.

Successive over-relaxation (SOR):

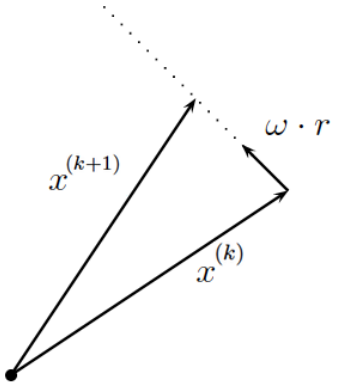
Man wählt einen *Extrapolationsparameter* $\omega \in (1, 2)$ und verwendet $S := D - \omega L$ und $T := [(1 - \omega)D + \omega R]$. Für $\omega = 1$ ist das SOR-Verfahren damit identisch zu dem Gauß-Seidel-Verfahren. Eine explizite Formel für das SOR-Verfahren ist gegeben durch

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

Erklärung: Man kann den SOR-Schritt auch betrachten in der Form

$$x_i^{(k+1)} = x_i^{(k)} + \omega \underbrace{\left(\tilde{\tilde{x}}_i^{(k+1)} - x_i^{(k)} \right)}_{r} ,$$

wobei $\omega \in (1,2)$ und $\tilde{\tilde{x}}_i^{(k+1)}$ die Gauß-Seidel-Iteration für die i-te Variable aus $(x_1^{(k+1)}, \ldots, x_{i-1}^{(k+1)}, x_i^{(k)}, \ldots, x_n^{(k)})$ ist.



Hier wird also der Gauß-Seidel-Schritt für jede einzelne Variable extrapoliert. Für große Gleichungssysteme kann damit die Iterationsanzahl für ein geeignetes ω oft um eine Zehnerpotenz gesenkt werden. Wird der Parameter < 1 gewählt, spricht man von einer Unterrelaxation.

38.8 Beispiel

$A = \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$, $b = \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, $x^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ und $\omega = 1.15$ seien gegeben.

k	$x_1^{(k)}$	$x_2^{(k)}$
0	0	0
1	1,725	3,292
2	3,359	3,738
3	3,370	3,677
4	3,334	3,665
3	3,333	3,666

38.9 Konvergenzresultate

a) Die Jacobi-, Gauß-Seidel- und SOR-Verfahren können als Fixpunktiterationen interpretiert werden. Ihre Konvergenz kann unter anderem mittels des Banach'schen Fixpunktsatzes (vgl. MfI 1, 22.3) untersucht werden; danach liegt Konvergenz vor, wenn die Abbildung kontrahierend ist. Dies ist nicht in allen Fällen erfüllt.

b) Für wichtige Spezialfälle existieren jedoch Konvergenzaussagen, z. B.:

- Ist die Systemmatrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ streng **diagonaldominant,** d. h. für jedes i gilt

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \ ,$$

so konvergieren das Jacobi- und das Gauß-Seidel-Verfahren.

c) Man kann zeigen, das generelle Iterationsverfahren für lineare Gleichungssysteme konvergieren, falls der betragsmäßig größte Eigenwert der Systemmatrix $S^{-1}T$ echt kleiner als 1 ist. Die Geschwindigkeit der Konvergenz hängt dabei von der Größe dieses Eigenwertes ab, d.h. je kleiner der Eigenwert, desto schneller das Verfahren. Damit ist es u.a. möglich einen optimalen Parameter ω zu finden.

38.10 Effizienz

a) Die vorgestellten Verfahren sind die einfachsten, allerdings nicht die effizientesten iterativen Verfahren zum Lösen linearer Gleichungssysteme.

b) Effizientere, aber kompliziertere Verfahren:

- Projektionsmethoden und Krylow-Unterraum-Verfahren (u.a. CG = conjugate gradient)
- Verwendung von präkonditionierten Matrizen
- Mehrgitterverfahren (Multigrid)

Vgl. dazu numerische Spezialliteratur.

c) Hocheffiziente Ansätze wie etwa die Mehrgitterverfahren verwenden oft das Gauß-Seidel- oder SOR-Verfahren als Grundbaustein. Mit ihnen ist es z. T. möglich, lineare Gleichungssysteme in *optimaler* Komplexität (d. h. $O(n)$) zu lösen.