

Numerisches Rechnen

(für Informatiker)

M. Grepl

P. Esser & G. Welper & L. Zhang

Institut für Geometrie und Praktische Mathematik
RWTH Aachen

Wintersemester 2011/12

Problemstellung

Lineare Gleichungssysteme, iterative Verfahren

- ▶ geg.: $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $n \gg 1$, A dünnbesetzt;
ges.: $x \in \mathbb{R}^n$, so dass $Ax = b$, wobei

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

und die meisten $a_{i,j} = 0$ sind.

- ▶ Annahme: A nichtsingulär, d.h. $\det(A) \neq 0$
 \Rightarrow Spalten von A bilden eine Basis in \mathbb{R}^n
 $\Rightarrow Ax = b$ eindeutig lösbar.

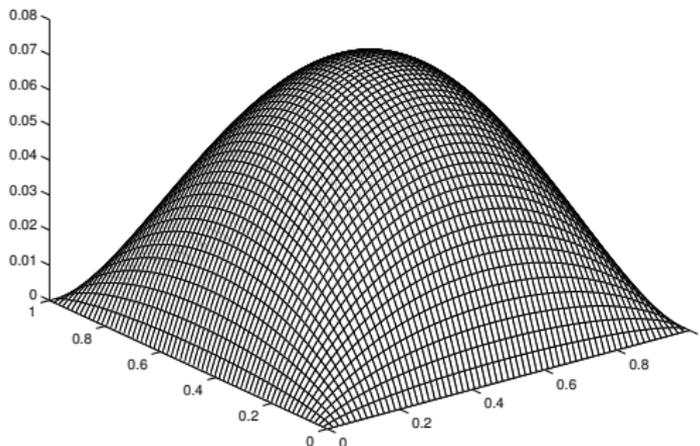
Poisson-Gleichung

Gesucht $u \in C^2(\Omega) \cap C(\bar{\Omega})$, $\Omega = [0, 1]^2$, so daß

$$-\Delta u = 1 \quad \text{in } \Omega,$$

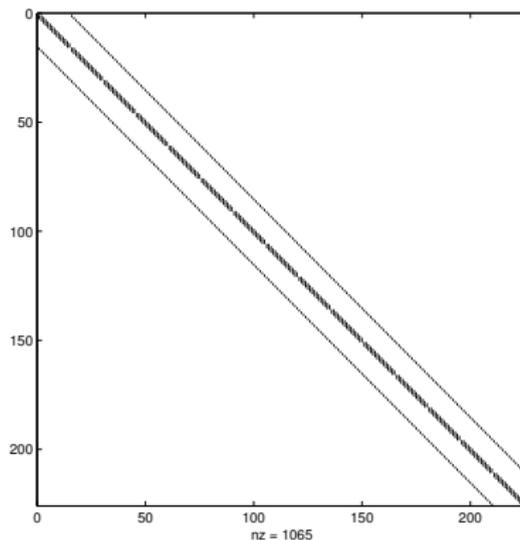
$$u = 0 \quad \text{auf } \partial\Omega.$$

Typisches Beispiel: Wärmeleitungsgleichung, Diffusionsgleichung.



Poisson-Gleichung

Diskretisierung mit Finite Differenzen Verfahren: $\text{spy}(A)$



Elemente: 50625, $\text{nnz}(G) = 1065$, $\text{density} = 0.021$

Poisson-Gleichung

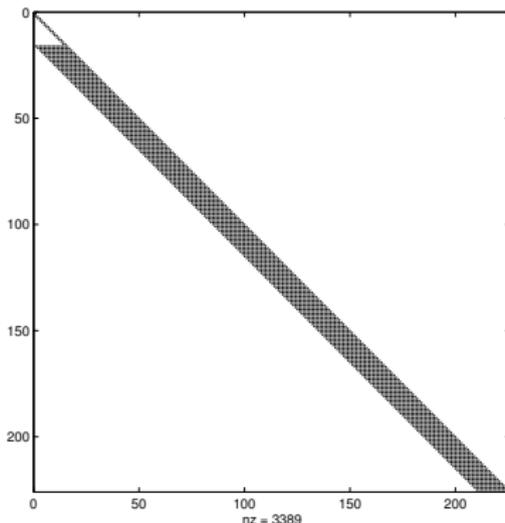
Eigenschaften von Steifigkeitsmatrizen:

- ▶ Hochdimensionalität
Anzahl der Unbekannten $\sim h^{-d}$ bei d Ortsvariablen
- ▶ Dünnbesetztheit
- ▶ Blockstruktur
Bei Diskretisierung auf regelmäßigen Rechteckgittern
- ▶ Schlechte Kondition
Diskrete Poisson-Gleichung: $\kappa(\mathbf{A}) = \left(\frac{2}{\pi h}\right)^2$
- ▶ Symmetrie, Positiv-Definitheit
- ▶ Diagonaldominanz (strikt für $<$)

$$\sum_{j \neq i} |a_{i,j}| \leq |a_{i,i}| \text{ für alle } i,$$

Poisson-Gleichung

Problem des "fill in" bei Cholesky-Zerlegung $A = LDL^T$: $\text{spy}(L)$



h	1/16	1/32	1/64	1/128	
$\text{nz}(A_1)$	1065	4681	19593	80137	$\approx 5h^{-2}$
$\text{nz}(L)$	3389	29821	250109	2048509	$\approx h^{-3}$

Dünnbesetzte Matrizen in Matlab

Löse $Ax = b$

- ▶ $x = \text{inv}(A)*b$
- ▶ $x = A^{(-1)}*b$
- ▶ $x = A \setminus b$

Sparse Matrices

- ▶ “Matlab is slow only if you use it incorrectly.” (Mathworks)
- ▶ Wenn A dünnbesetzt ist, ist der Code langsam, wenn man den folgenden Befehl zu oft benutzt :

$$A(i, j) = \dots$$

- ▶ Warum? Hat damit zu tun, wie Matlab Matrizen speichert ...

Dünnbesetzte Matrizen in Matlab

Beispiel, wie dünnbesetzte Matrizen in Matlab gespeichert werden:

```
C = [
4.5 0.0 3.2 0.0
3.1 2.9 0.0 0.9
0.0 1.7 3.0 0.0
3.5 0.4 0.0 1.0 ] ;
C = sparse(C)
```

```
C =
(1,1) 4.5000
(2,1) 3.1000
(4,1) 3.5000
(2,2) 2.9000
(3,2) 1.7000
(4,2) 0.4000
(1,3) 3.2000
(3,3) 3.0000
(2,4) 0.9000
(4,4) 1.0000
```

```
column 1:
k      row index  value
1.0000 1.0000    4.5000
2.0000 2.0000    3.1000
3.0000 4.0000    3.5000
column 2:
k      row index  value
4.0000 2.0000    2.9000
5.0000 3.0000    1.7000
6.0000 4.0000    0.4000
column 3:
k      row index  value
7.0000 1.0000    3.2000
8.0000 3.0000    3.0000
column 4:
k      row index  value
9.0000 2.0000    0.9000
10.0000 4.0000    1.0000
```

Was passiert, wenn wir den folgenden Befehl eingeben?

$$C(3,1) = 42$$

Dünnbesetzte Matrizen in Matlab

Vorher:

```
column 1:
  k      row index  value
  1.0000  1.0000    4.5000
  2.0000  2.0000    3.1000
  3.0000  4.0000    3.5000
column 2:
  k      row index  value
  4.0000  2.0000    2.9000
  5.0000  3.0000    1.7000
  6.0000  4.0000    0.4000
column 3:
  k      row index  value
  7.0000  1.0000    3.2000
  8.0000  3.0000    3.0000
column 4:
  k      row index  value
  9.0000  2.0000    0.9000
 10.0000  4.0000    1.0000
```

Nachher:

```
column 1:
  k      row index  value
  1.0000  1.0000    4.5000
  2.0000  2.0000    3.1000
  3.0000  3.0000   42.0000
  4.0000  4.0000    3.5000
column 2:
  k      row index  value
  5.0000  2.0000    2.9000
  6.0000  3.0000    1.7000
  7.0000  4.0000    0.4000
column 3:
  k      row index  value
  8.0000  1.0000    3.2000
  9.0000  3.0000    3.0000
column 4:
  k      row index  value
 10.0000  2.0000    0.9000
 11.0000  4.0000    1.0000
```

Dünnbesetzte Matrizen in Matlab

- ▶ Der Befehl $C(3,1) = 42$ benötigt Zeit, die proportional zu den Einträgen in C ist, die ungleich null sind
- ▶ Besseres Vorgehen: drei Vektoren generieren, die die Reihen und Spalten und zugehörige Einträge beinhalten:

$$I(\dots) = \dots;$$

$$J(\dots) = \dots;$$

$$X(\dots) = \dots;$$

- ▶ Zusammenbau der Matrix mit:

$$A = \text{sparse}(I, J, X, n, n);$$

- ▶ Doppelte Einträge werden summiert (z.B. beim Finite Differenzen Code)
- ▶ Preallocate memory (`spalloc`).

Lineare Iterationsverfahren

Aufgabe

Für $A \in \mathbb{R}^{n \times n}$ (nichtsingulär) und $b \in \mathbb{R}^n$ ist das Gleichungssystem

$$Ax = b$$

zu lösen. Wir nehmen an, daß n groß ($n \gtrsim 1000$) und A dünnbesetzt ist.

Beispiele:

- ▶ Google-Matrix
- ▶ Markov-Ketten
- ▶ Netzwerkanalyse
- ▶ Diskretisierung von (partiellen) DGL (Finite Differenzen, Finite Elemente, Finite Volumen)
- ▶ ...

Lineare Iterationsverfahren

Erinnerung: Nachiteration

- ▶ Sei x^0 eine näherungsweise Lösung von $Ax = b$ mit Residuum $r^0 = b - Ax^0$.
- ▶ Der Fehler, $\delta^0 = x - x^0$, erfüllt

$$A\delta^0 = Ax - Ax^0 = b - Ax^0 = r^0$$

- ▶ **Exakte Lösung** ergibt sich aus

$$x = x^0 + \delta^0 = x^0 + A^{-1}(b - Ax^0)$$

- ▶ **Problem:** Lösung des LGS

$$A\delta^0 = b - Ax^0$$

benötigt.

- ▶ aber ...

Lineare Iterationsverfahren

Annahme: wir haben eine näherungsweise Lösung x^0

$$x = x^0 + A^{-1}(b - Ax^0)$$

$$x = x^0 + C(b - Ax^0)$$

$$x^{k+1} = x^k + C(b - Ax^k), \quad k = 0, 1, 2, \dots$$

Idee:

- ▶ Ersetze A^{-1} durch C , wobei
 - ▶ A^{-1} durch C "gut" approximiert wird, und
 - ▶ der Aufwand der Matrix-Vektor Multiplikation Cy gering ist.
- ▶ Führe Iteration für $k = 0, 1, 2, \dots$ ein.

Fixpunktiteration:

$$x^{k+1} = x^k + C(b - Ax^k) := \Phi(x^k), \quad k = 0, 1, 2, \dots$$

Lineare Iterationsverfahren

- ▶ Fixpunktgleichung

$$\Phi(x) := x + C(b - Ax)$$

mit geeignetem $C \in \mathbb{R}^{n \times n}$ nichtsingulär.

- ▶ Fixpunktiteration

$$\begin{aligned}x^{k+1} &= \Phi(x^k) = x^k + C(b - Ax^k) \\ &= (I - CA)x^k + Cb, \quad k = 0, 1, 2, \dots\end{aligned}$$

- ▶ Für den Fehler $e^k := x^k - x^*$ ergibt sich

$$e^{k+1} = x^{k+1} - x^* = \Phi(x^k) - \Phi(x^*) = (I - CA)e^k$$

also

$$e^k = (I - CA)^k e^0, \quad k = 0, 1, 2, \dots$$

Satz 13.3

Das Verfahren konvergiert für jeden Startwert x^0 gegen die Lösung x^* des Gleichungssystems genau dann, wenn

$$\rho(I - CA) < 1$$

gilt, wobei $\rho(I - CA)$ der Spektralradius, also der betragsmäßig größte Eigenwert, von $I - CA$ ist.

Folgerung 13.5

Für eine beliebige Vektornorm $\|\cdot\|$ mit zugehöriger Operatornorm gilt:

$$\|x^k - x^*\| \leq \|I - CA\|^k \|x^0 - x^*\|, \quad k = 0, 1, 2, \dots$$

Die Konvergenz ist für jeden Startwert x^0 gesichert, falls für *irgendeine* Norm die Bedingung

$$\|I - CA\| < 1$$

erfüllt ist.

Konvergenzrate

- ▶ Maß für die Konvergenzgeschwindigkeit: $\rho(I - CA)$.
- ▶ Je kleiner $\rho(I - CA)$ ist, desto schneller wird der Fehler reduziert.
- ▶ Mittlere Fehlerreduktion in k Iterationsschritten

$$\sigma_k = \left(\frac{\|e^k\|}{\|e^0\|} \right)^{\frac{1}{k}} \rightarrow |\lambda_1| = \rho(I - CA) \text{ für } k \rightarrow \infty$$

- ▶ Reduktion des Anfangfehlers $\|e^0\|$ um Faktor $\frac{1}{e}$ benötigt $k = (-\ln \sigma_k)^{-1}$ Iterationsschritte
- ▶ Asymptotische Konvergenzrate

$$-\ln(\rho(I - CA))$$

Wahl von C

Zu $A \in \mathbb{R}^{n \times n}$ wähle $C \in \mathbb{R}^{n \times n}$ so, daß

- ▶ A^{-1} durch C genügend gut in dem Sinne approximiert wird, daß $\rho(I - CA)$ möglichst klein ist.
- ▶ die Operation

$$y \rightarrow Cy$$

mit möglichst geringem Aufwand durchführbar ist.

Beachte:

Bei den meisten iterativen Verfahren wird die Operation $y \rightarrow Cy$ durchgeführt, ohne daß die Matrix C explizit berechnet wird.

Komplexität

Zum Vergleich der unterschiedliche Verfahren, gehen wir davon aus, daß

- ▶ eine Klasse von Gleichungssystemen $Ax = b$ vorliegt
- ▶ vorgegeben ist, mit welchem Faktor R ein (beliebiger) Startfehler reduziert werden soll ($\|e^k\| = \frac{1}{R}\|e^0\|$)

Komplexität eines iterativen Verfahrens

Größenordnung der Anzahl arithmetischer Operationen, die benötigt werden, um für das vorliegende Problem eine Fehlerreduktion um den Faktor R zu erreichen.

Lineare Iterationsverfahren

- ▶ Jakobi
- ▶ Gauß-Seidel
- ▶ SOR
- ▶ Conjugate Gradients (später)

Problemstellung

Lineare Gleichungssysteme, iterative Verfahren

geg.: $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$, $\det(A) \neq 0$,
und $n \gg 1$, A dünnbesetzt;

ges.: $x \in \mathbb{R}^n$, so dass $Ax = b$, wobei

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

und die meisten $a_{i,j} = 0$ sind.

Vorgehen

$$x^{k+1} = x^k + C(b - Ax^k), \quad k = 0, 1, 2, \dots$$

Definition

Definiere die Zerlegung

$$A = D - L - U$$

wobei

$$D = \begin{bmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{n,n} \end{bmatrix},$$

$$L = - \begin{bmatrix} 0 & \cdots & \cdots & 0 \\ a_{2,1} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n-1} & 0 \end{bmatrix}, \quad U = - \begin{bmatrix} 0 & a_{1,2} & \cdots & a_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{bmatrix}.$$

Definition

Diagonaldominanz

Eine Matrix $A \in \mathbb{R}^{n \times n}$ heißt strikt diagonaldominant, falls gilt

$$\sum_{\substack{j=1 \\ j \neq i}}^n |a_{i,j}| < |a_{i,i}|.$$

Zeilen-/Spaltensummenkriterium

Eine Matrix $A \in \mathbb{R}^{n \times n}$ erfüllt das starke Zeilensummenkriterium (bzw. Spaltensummenkriterium), wenn

$$\max_i \sum_{\substack{j=1 \\ j \neq i}}^n \frac{|a_{i,j}|}{|a_{i,i}|} < 1, \quad \left(\text{bzw.} \max_j \sum_{\substack{i=1 \\ i \neq j}}^n \frac{|a_{i,j}|}{|a_{i,i}|} < 1 \right).$$

Jacobi-Verfahren

Beim Jacobi- oder Gesamtschrittverfahren:

$$C = (\text{diag}(A))^{-1} = D^{-1}.$$

Die Iterationsvorschrift lautet

$$x^{k+1} = (I - D^{-1}A) x^k + D^{-1}b,$$

d.h. in Komponentenschreibweise ergibt sich:

Algorithmus 13.7 (Jacobi-Verfahren)

Gegeben: Startvektor $x^0 \in \mathbb{R}^n$. Für $k = 0, 1, \dots$ berechne:

$$x_i^{k+1} = a_{i,i}^{-1} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{i,j} x_j^k \right), \quad i = 1, 2, \dots, n$$

Jacobi-Verfahren

Vorgehen:

- ▶ Löse i -te Gleichung nach der i -ten Unbekannte x_i auf.
- ▶ Verwende Werte aus dem vorherigen Iterationsschritt für die übrigen Unbekannten ($x_j, j \neq i$).
- ▶ Beachte:
 - ▶ x^k und x^{k+1} müssen gespeichert werden
 - ▶ Verfahren leicht parallelisierbar

Rechenaufwand. Der Rechenaufwand pro Iterationsschritt ist beim Jacobi-Verfahren angewandt auf eine dünnbesetzte Matrix $A \in \mathbb{R}^{n \times n}$ vergleichbar mit einer Matrix-Vektor-Multiplikation Ax , beansprucht also $O(n)$ Operationen.

Konvergenz

Satz 13.9

Für das Jacobi-Verfahren gelten folgende Konvergenzkriterien:

- ▶ Falls A als auch $2D - A$ symmetrisch positiv definit sind, folgt

$$\rho(I - D^{-1}A) < 1.$$

- ▶ Falls A irreduzibel diagonaldominant ist, gilt

$$\rho(I - D^{-1}A) \leq \|I - D^{-1}A\|_{\infty} < 1.$$

Satz: hinreichende Bedingung

Das Jacobi-Verfahren konvergiert, falls A das starke Zeilensummen- oder Spaltensummenkriterium erfüllt.

Beispiel 13.10

Das Diffusionsproblem mit der Matrix \mathbf{A}_1 . Es gilt

$$\begin{aligned}\rho(\mathbf{I} - \mathbf{CA}) &= \rho(\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}_1) \\ &= \max\{|1 - \frac{1}{4}h^2\lambda| \mid \lambda \text{ Eigenwert von } \mathbf{A}_1\} \\ &= 1 - 2 \sin^2(\frac{1}{2}\pi h) = \cos(\pi h) \approx 1 - \frac{1}{2}\pi^2 h^2.\end{aligned}$$

Für die asymptotische Konvergenzrate gilt somit

$$-\ln(\rho(\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}_1)) \approx -\ln(1 - \frac{1}{2}\pi^2 h^2) \approx \frac{1}{2}\pi^2 h^2.$$

Um einen Startfehler um einen Faktor R zu reduzieren, sind etwa

$$K = \frac{-\ln R}{\ln \rho(\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}_1)} \approx \frac{2}{\pi^2 h^2} \ln R$$

Iterationsschritte erforderlich.

Beachte: Die geschätzte Anzahl der notwendigen Iterationsschritte wächst in Abhängigkeit der Schrittweite h wie die Konditionszahlen der betreffenden Matrizen.

Beispiel 13.10

Sei $\#$ die Anzahl von Iterationsschritten, die zur Reduktion des Startfehlers um einen Faktor $R = 10^3$ benötigt werden, und K die theoretische Schätzung von $\#$ aus:

$$K = \frac{-\ln 10^3}{\ln(\cos \pi h)} \approx \frac{2}{\pi^2 h^2} \ln 10^3.$$

Ergebnisse:

h	1/40	1/80	1/160	1/320
$\#$	2092	8345	33332	133227
K	2237	8956	35833	143338

Komplexität. Da $m \sim h^{-2}$, folgt, daß $K \sim m$ Iterationsschritte benötigt werden. Da jeder Schritt einen zu m proportionalen Aufwand erfordert, ergibt sich, daß die Komplexität des Jacobi-Verfahrens (für diese Problemstellung) etwa cm^2 ist.

Beispiel

► Gleichungssystem

$$\begin{bmatrix} 4 & -1 & 0 & 1 & 0 \\ -1 & 4 & -1 & 0 & 1 \\ 0 & -1 & 4 & -1 & 0 \\ 1 & 0 & -1 & 4 & -1 \\ 0 & 1 & 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \\ 100 \end{bmatrix}$$

Beispiel

Konvergenz

Table 1.3. Solution by the Jacobi Iteration Method

k	x_1	x_2	x_3	x_4	x_5
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	25.000000	25.000000	25.000000	25.000000	25.000000
2	25.000000	31.250000	37.500000	31.250000	25.000000
3	25.000000	34.375000	40.625000	34.375000	25.000000
4	25.000000	35.156250	42.187500	35.156250	25.000000
5	25.000000	35.546875	42.578125	35.546875	25.000000
...
16	25.000000	35.714284	42.857140	35.714284	25.000000
17	25.000000	35.714285	42.857142	35.714285	25.000000
18	25.000000	35.714285	42.857143	35.714285	25.000000

Gauß-Seidel-Verfahren

Beim Gauß-Seidel- oder Einzelschrittverfahren:

$$C = (D - L)^{-1}.$$

Die Iterationsvorschrift lautet

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

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

d.h. in Komponentenschreibweise ergibt sich:

Algorithmus 13.12 (Gauß-Seidel-Verfahren)

Gegeben: Startvektor $x^0 \in \mathbb{R}^n$. Für $k = 0, 1, \dots$ berechne:

$$x_i^{k+1} = a_{i,i}^{-1} \left(b_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{k+1} - \sum_{j=i+1}^n a_{i,j} x_j^k \right), \quad i = 1, 2, \dots, n.$$

Gauß-Seidel-Verfahren

Vorgehen:

- ▶ Löse i -te Gleichung nach der i -ten Unbekannte x_i auf.
- ▶ Verwende die bereits vorher berechneten Komponenten der neuen Annäherung.
- ▶ Beachte:
 - ▶ Nur ein Vektor muss gespeichert werden
 - ▶ Verfahren nicht mehr einfach parallelisierbar
 - ▶ Ergebnis hängt von der Reihenfolge der Berechnung ab.

Rechenaufwand. Für eine dünnbesetzte Matrix $A \in \mathbb{R}^{n \times n}$ ist der Rechenaufwand pro Iterationsschritt beim Gauß-Seidel-Verfahren vergleichbar mit dem Aufwand beim Jacobi-Verfahren, beträgt also $O(n)$ Operationen.

Konvergenz

Satz 13.13

Für das Gauß-Verfahren gelten folgende Konvergenzkriterien:

- ▶ Falls A symmetrisch positiv definit sind, folgt

$$\rho(I - (D - L)^{-1}A) < 1$$

- ▶ Falls A irreduzibel diagonaldominant ist, gilt

$$\rho(I - (D - L)^{-1}A) \leq \|I - (D - L)^{-1}A\|_{\infty} < 1.$$

Satz: hinreichende Bedingung

Das Gauß-Seidel-Verfahren konvergiert, falls A das starke Zeilensummen- oder Spaltensummenkriterium erfüllt.

Beispiel 13.15

Das Diffusionsproblem mit der Matrix \mathbf{A}_1 . Dafür gilt:

$$\rho(\mathbf{I} - \mathbf{CA}_1) = \rho(\mathbf{I} - (\mathbf{D} - \mathbf{L})^{-1}\mathbf{A}_1) = (\rho(\mathbf{I} - \mathbf{D}^{-1}\mathbf{A}_1))^2$$

Hieraus ergibt sich:

$$\rho(\mathbf{I} - \mathbf{CA}_1) = \cos^2(\pi h) \approx 1 - \pi^2 h^2 .$$

$$-\ln(\rho(\mathbf{I} - (\mathbf{D} - \mathbf{L})^{-1}\mathbf{A}_1)) \approx -\ln(1 - \pi^2 h^2) \approx \pi^2 h^2 .$$

Um einen Startfehler um einen Faktor R zu reduzieren sind (asymptotisch) etwa K Iterationsschritte erforderlich, wobei

$$K = \frac{-\ln R}{\ln(\rho(\mathbf{I} - (\mathbf{D} - \mathbf{L})^{-1}\mathbf{A}_1))} \approx \frac{1}{\pi^2 h^2} \ln R .$$

Ergebnisse:

h	1/40	1/80	1/160	1/320
$\#$	1056	4193	16706	66694
K	1119	4478	17916	71669

Beispiel 13.15

Komplexität

Für das Gauß-Seidel-Verfahren angewandt auf die diskrete Poisson-Gleichung

$$\mathbf{A}_1 \mathbf{x} = \mathbf{b}, \quad \mathbf{A}_1 \in \mathbb{R}^{m \times m}, \quad m = (n-1)^2, \quad n = \frac{1}{h}$$

ist die Komplexität cm^2 Operationen.

Beispiel

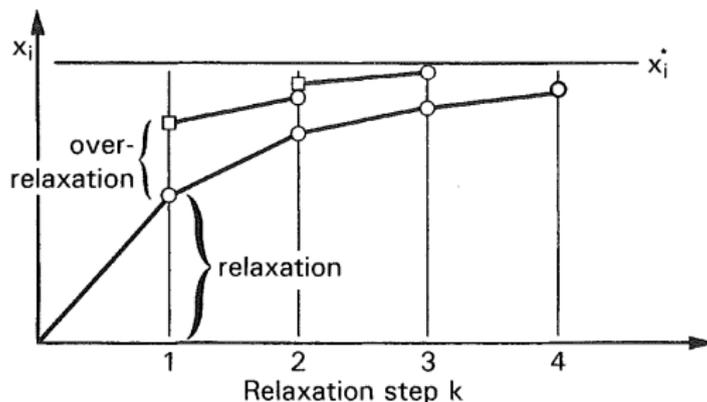
Konvergenz

Table 1.4. Solution by the Gauss-Seidel Iteration Method

k	x_1	x_2	x_3	x_4	x_5
0	0.000000	0.000000	0.000000	0.000000	
1	25.000000	31.250000	32.812500	26.953125	23.925781
2	26.074219	33.740234	40.173340	34.506226	25.191498
3	24.808502	34.947586	42.363453	35.686612	25.184757
4	24.815243	35.498485	42.796274	35.791447	25.073240
5	24.926760	35.662448	42.863474	35.752489	25.022510
...
13	25.000002	35.714287	42.857142	35.714285	25.999999
14	25.000001	35.714286	42.857143	35.714285	25.000000
15	25.000000	35.714286	42.857143	35.714286	25.000000

Successive Over-Relaxation

- ▶ In vielen Fällen ist die Veränderung der x_i von Iteration zu Iteration immer in die gleiche Richtung
- ▶ Korrektur ("over-relaxing") des Iterationsschritts kann die Konvergenz verbessern



Successive Over-Relaxation

Iterationsvorschrift bei Gauß-Seidel-Verfahren:

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

Algorithmus 13.17 (SOR-Verfahren)

Gegeben: Startvektor $x^0 \in \mathbb{R}^n$, Parameter $\omega \in (0, 2)$. Für $k = 0, 1, \dots$ berechne:

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

Dies entspricht der Iterationsvorschrift

$$\left(\frac{1}{\omega} D - L \right) x^{k+1} = \left(\frac{1}{\omega} D - L \right) x^k - A x^k + b,$$

$$\text{d.h. } C = \left(\frac{1}{\omega} D - L \right)^{-1}.$$

Successive Over-Relaxation

Rechenaufwand. Beim SOR-Verfahren ist der Rechenaufwand pro Iterationsschritt vergleichbar dem Aufwand beim Gauß-Seidel-Verfahren, also $O(n)$ Operationen für eine dünnbesetzte Matrix $A \in \mathbb{R}^{n \times n}$.

Satz 13.18. Sei $M_\omega := I - (\frac{1}{\omega}D - L)^{-1}A$ die Iterationsmatrix des SOR-Verfahrens. Es gilt:

- $\rho(M_\omega) \geq |\omega - 1|$.
- Für A s.p.d. gilt:

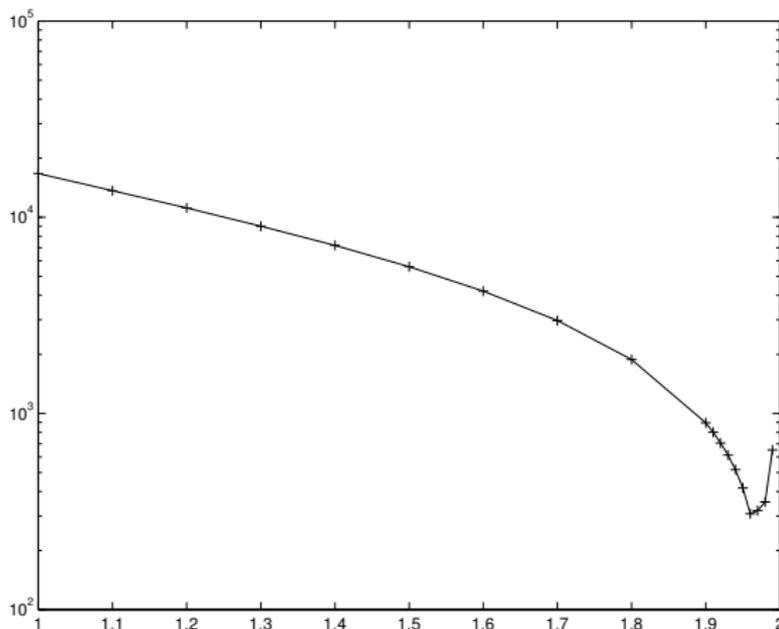
$$\rho(M_\omega) < 1 \quad \text{für alle } \omega \in (0, 2).$$

- Sei A irreduzibel diagonaldominant mit $a_{i,j} \leq 0$ für alle $i \neq j$ und $a_{i,i} > 0$ für alle i . Dann gilt:

$$\rho(M_\omega) < 1 \quad \text{für alle } \omega \in (0, 1].$$

Beispiel 13.19

Anzahl der Iterationen in Abhängigkeit von ω ($h = \frac{1}{160}$)



Optimaler Wert von ω

- ▶ Der optimale Wert von ω ist stark problemabhängig
- ▶ Für die meisten Probleme ist der optimal Wert nicht bekannt
- ▶ Ausnahme: diskrete Poisson-Gleichung (Matrix A_1)

Satz 13.20.

Wir betrachten die diskretisierte Poisson-Gleichung $A_1 x = b$.

Sei $\mu := \rho(I - D^{-1}A_1) < 1$ der Spektralradius der Iterationsmatrix des Jacobi-Verfahrens.

Sei M_ω die Iterationsmatrix des SOR-Verfahrens.

Dann ist $\rho(M_\omega)$ für den Relaxationsparameter

$$\omega_{\text{opt}} := \frac{2}{1 + \sqrt{1 - \mu^2}} = 1 + \left(\frac{\mu}{1 + \sqrt{1 - \mu^2}} \right)^2$$

optimal und

$$\rho(M_{\omega_{\text{opt}}}) = \omega_{\text{opt}} - 1.$$

Komplexität

Komplexität.

Für die diskretisierte Poisson-Gleichung $\mathbf{A}_1 \mathbf{x} = \mathbf{b}$ ergibt sich beim SOR-Verfahren mit dem *optimalen* ω -Wert *eine sehr große Komplexitätsverbesserung im Vergleich zum Jacobi- und zum Gauß-Seidel-Verfahren.*

$$\rho(\mathbf{M}_{\omega_{\text{opt}}}) = \omega_{\text{opt}} - 1 = \left(\frac{\cos(\pi h)}{1 + \sin(\pi h)} \right)^2 = \frac{1 - \sin(\pi h)}{1 + \sin(\pi h)} \approx 1 - 2\pi h.$$

$$K = -\frac{\ln R}{\ln \rho(\mathbf{M}_{\omega_{\text{opt}}})} \approx \frac{1}{2\pi h} \ln R \approx \frac{\ln R}{2\pi} \sqrt{m}.$$

Da der Aufwand pro Iteration proportional zu m ist, hat dieses Verfahren eine Komplexität (für die vorliegende Problemstellung) von

$$c m^{1.5} \text{ Operationen.}$$

Beispiel 13.21

Poisson-Gleichung $A_1 x = b$ mit x^0 , b und R wie in Beispiel 13.10.

Wir verwenden das SOR-Verfahren mit $\omega = \omega_{\text{opt}}$.

Ergebnisse:

h	1/40	1/80	1/160	1/320
#	73	146	292	585
K	44	88	176	352

Beispiel

Konvergenz

Table 1.5. Solution by the SOR Method

k	x_1	x_2	x_3	x_4	x_5
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	27.500000	35.062500	37.142188	30.151602	26.149503
2	26.100497	34.194375	41.480925	35.905571	25.355629
3	24.419371	35.230346	42.914285	35.968342	25.167386
4	24.855114	35.692519	42.915308	35.790750	25.010375
5	24.987475	35.726188	42.875627	35.717992	24.996719
...
11	24.999996	35.714285	42.857145	35.714287	25.000000
12	25.000000	35.714286	42.857143	35.714286	25.000000
13	25.000000	35.714286	42.857143	35.714286	25.000000

Zusammenfassung

- ▶ Viele Problem in der Praxis führen auf große dünnbesetzte lineare Gleichungssysteme
- ▶ Direkte Verfahren
 - ▶ Berechnung der Lösung bis auf Rundungsfehler in $O(n^3)$ Operationen
 - ▶ Problem des "fill-in" bei dünnbesetzten Matrizen
- ▶ Iterative Verfahren
 - ▶ Bestimmung der Lösung näherungsweise
 - ▶ Konvergenz hängt von Spektralradius $\rho(I - CA)$ ab
- ▶ Wenn A das strenge Zeilen- oder Spaltensummenkriterium erfüllt, konvergieren das **Jacobi-Verfahren**, das **Gauss-Seidel-Verfahren**, und **Succesive Over-Relaxation**.