

Mehrdimensionale Iterationen, Gleichungssysteme

3. Vorlesung 170 004 Numerische Methoden I

Clemens Brand und Erika Hausenblas

Montanuniversität Leoben

8. März 2018

Mehrdimensionale Iterationen, Gleichungssysteme

① Wiederholung:

Vektoren, vektorwertige Funktionen

Fixpunkt-Iteration: Theorie

② Normen

Vektornorm

Norm misst Distanz

Matrixnorm

③ Konvergenz von Fixpunkt-Iterationen

Kontrahierende Abbildung

Jacobi-Matrix

④ Newton-Verfahren für nichtlineare Systeme

⑤ Fixpunkt-Iteration für Lineare Systeme

Jacobi-Verfahren

Beispiel: Wärmeleitungsgleichung

Gliederung 3. Vorlesung

① Wiederholung:

Vektoren, vektorwertige Funktionen
Fixpunkt-Iteration: Theorie

② Normen

Vektornorm
Norm misst Distanz
Matrixnorm

③ Konvergenz von Fixpunkt-Iterationen

Kontrahierende Abbildung
Jacobi-Matrix

④ Newton-Verfahren für nichtlineare Systeme

⑤ Fixpunkt-Iteration für Lineare Systeme

Jacobi-Verfahren
Beispiel: Wärmeleitungsgleichung

Wiederholung

Vektoren, vektorwertige Funktionen

- ▶ Mehrere Gleichungen und Unbekannte
- ▶ Unbekannte zu einem Vektor zusammenfassen
- ▶ Gleichungssystem als vektorwertige Funktion schreiben
- ▶ Begriffe Nullstelle und Fixpunkt lassen sich direkt verallgemeinern
- ▶ Auch Fixpunkt-Iteration geht analog

Bei geeigneter Schreibweise ändert sich fast nichts gegenüber dem Fall einer Gleichung und Unbekannten

Skalare und vektorwertige Funktionen

Schreibweise: Vektoren und vektorwertige Funktionen fett gedruckt

Reellwertige Funktionen, Skalare: $f : \mathbb{R} \rightarrow \mathbb{R}$, $y = f(x)$

Vektorwertige Funktionen, Vektoren: $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $\mathbf{y} = \mathbf{f}(\mathbf{x})$

Komponenten eines Vektors $\in \mathbb{R}^n$:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad \text{oder } \mathbf{x}^T = [x_1, x_2, \dots, x_n]$$

Normalerweise ist mit \mathbf{x} ein Spalten-, mit \mathbf{x}^T ein Zeilenvektor gemeint.

Iterationsindizes sind (um sie von Vektorkomponenten zu unterscheiden) in der Regel hochgestellt, in Klammern: $\mathbf{x}^{(k)}$, $k = 0, 1, 2, \dots$

Wiederholung

Wichtige Themen zur Fixpunkt-Iteration

- ▶ Konvergenzordnung: wie rasch konvergiert eine Iteration
- ▶ Was ist eine kontrahierende Abbildung
- ▶ Wann konvergiert Fixpunktiteration
 - ▶ anschaulich erklärt
 - ▶ mathematisch exakte Konvergenzbedingung
- ▶ Was bedeutet „lokale Konvergenz“
- ▶ Anschauliche Bedeutung von $|\Phi'| < 1$

Gliederung 3. Vorlesung

① Wiederholung:

Vektoren, vektorwertige Funktionen
Fixpunkt-Iteration: Theorie

② Normen

Vektornorm
Norm misst Distanz
Matrixnorm

③ Konvergenz von Fixpunkt-Iterationen

Kontrahierende Abbildung
Jacobi-Matrix

④ Newton-Verfahren für nichtlineare Systeme

⑤ Fixpunkt-Iteration für Lineare Systeme

Jacobi-Verfahren
Beispiel: Wärmeleitungsgleichung

Vektornormen: Motivation

Iterative Verfahren brauchen Abbruchbedingungen

Typische Bedingungen: Hör auf,

- ▶ wenn der Fehler klein genug ist; oder
- ▶ wenn der Funktionswert (fast) 0 ist; oder
- ▶ wenn sich Werte (fast) nicht mehr ändern.

$$|x^{(k)} - x^*| < \epsilon, \quad |f(x^{(k)})| < \epsilon, \quad |x^{(k+1)} - x^{(k)}| < \epsilon$$

Wie entscheidet man, ob ein *Vektor* klein genug ist; oder ob zwei Vektoren sich fast nicht mehr unterscheiden?

$$\|x^{(k)} - x^*\| < \epsilon, \quad \|f(x^{(k)})\| < \epsilon, \quad \|x^{(k+1)} - x^{(k)}\| < \epsilon$$

Norm $\| \cdot \|$ verallgemeinert für Vektoren den Betrag $| \cdot |$ von Skalaren

Vektornormen: Motivation

Iterative Verfahren brauchen Abbruchbedingungen

Typische Bedingungen: Hör auf,

- ▶ wenn der Fehler klein genug ist; oder
- ▶ wenn der Funktionswert (fast) 0 ist; oder
- ▶ wenn sich Werte (fast) nicht mehr ändern.

$$|x^{(k)} - x^*| < \epsilon, \quad |f(x^{(k)})| < \epsilon, \quad |x^{(k+1)} - x^{(k)}| < \epsilon$$

Wie entscheidet man, ob ein *Vektor* klein genug ist; oder ob zwei Vektoren sich fast nicht mehr unterscheiden?

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\| < \epsilon, \quad \|\mathbf{f}(\mathbf{x}^{(k)})\| < \epsilon, \quad \|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\| < \epsilon$$

Norm $\| \cdot \|$ verallgemeinert für Vektoren den Betrag $| \cdot |$ von Skalaren

Vektornormen

Norm $\| \cdot \|$ verallgemeinert Betrag $| \cdot |$

Eine *Norm* ist eine Maßzahl für die „Größe“ eines Vektors.
Für einen Vektor $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ ist

$$\|\mathbf{x}\|_1 = \sum_{i=1}^n |x_i| \quad \text{Einsnorm}$$

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n (x_i)^2} \quad \text{euklidische Norm, Zweinorm}$$

$$\|\mathbf{x}\|_\infty = \max_i |x_i| \quad \text{Unendlich-Norm, Maximums-Norm}$$

In MATLAB:

$\|\mathbf{x}\|_1 = \text{norm}(\mathbf{x}, 1)$, $\|\mathbf{x}\|_2 = \text{norm}(\mathbf{x})$ oder $\text{norm}(\mathbf{x}, 2)$,

$\|\mathbf{x}\|_\infty = \text{norm}(\mathbf{x}, \text{inf})$.

Norm, formale Definition

Ergänzung: was Sie eigentlich aus Mathematik II schon wissen sollten

Eine *Norm* im \mathbb{R}^n ist eine Funktion, die jedem Vektor $\mathbf{x} \in \mathbb{R}^n$ eine nichtnegative reelle Zahl $\|\mathbf{x}\| \in \mathbb{R}_0^+$ zuordnet, wobei drei Bedingungen gelten müssen:

- ▶ Nur der Nullvektor hat Norm 0

$$\|\mathbf{x}\| = 0 \quad \Rightarrow \quad \mathbf{x} = \mathbf{0}$$

- ▶ Skalar α lässt sich als Betrag herausheben

$$\|\alpha \cdot \mathbf{x}\| = |\alpha| \cdot \|\mathbf{x}\|$$

- ▶ Die Dreiecksungleichung

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$$

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \forall \alpha \in \mathbb{R}$$

Norm, formale Definition

Ergänzung: was Sie eigentlich aus Mathematik II schon wissen sollten

Eine *Norm* im \mathbb{R}^n ist eine Funktion, die jedem Vektor $\mathbf{x} \in \mathbb{R}^n$ eine nichtnegative reelle Zahl $\|\mathbf{x}\| \in \mathbb{R}_0^+$ zuordnet, wobei drei Bedingungen gelten müssen:

- ▶ Nur der Nullvektor hat Norm 0

$$\|\mathbf{x}\| = 0 \quad \Rightarrow \quad \mathbf{x} = \mathbf{0}$$

- ▶ Skalar α lässt sich als Betrag herausheben

$$\|\alpha \cdot \mathbf{x}\| = |\alpha| \cdot \|\mathbf{x}\|$$

- ▶ Die Dreiecksungleichung

$$\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$$

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \forall \alpha \in \mathbb{R}$$

Vektornormen beim Fluggepäck

:-)

Ein Handgepäckstück darf maximal
55 cm × 40 cm × 23 cm groß und
nicht schwerer als 8 kg sein.

$$\left\| \begin{pmatrix} \ell/55 \\ b/40 \\ h/23 \\ m/8 \end{pmatrix} \right\|_{\infty} \leq 1$$

Ein Gepäckstück darf maximal
158 cm (Länge + Breite + Höhe)
lang sein.

$$\left\| \begin{pmatrix} \ell \\ b \\ h \end{pmatrix} \right\|_1 \leq 158$$

Norm und Distanz

Eine *Norm* kann auch die *Distanz* zwischen zwei Punkten \mathbf{x} und \mathbf{y} messen:

$$\text{dist}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|$$

- ▶ Taxis in Manhattan messen Strecken in der 1-Norm. deswegen heißt 1-Norm auch Taxi- oder Manhattan-Norm
- ▶ Abstand in der Luftlinie entspricht der 2-Norm.
- ▶ Größter Unterschied in den Komponenten: ∞ -Norm.

Matrixnormen

Achtung—Ergänzung zum Skriptum!

- ▶ Ein Hauptberuf von Matrizen ist, Vektoren zu multiplizieren.
- ▶ Das Ergebnis einer Matrix-Vektor-Multiplikation ist wieder ein Vektor; der ist i.a. länger/kürzer/verdreht gegenüber Ausgangsvektor
- ▶ Eine *Matrixnorm* misst als „Größe“ einer Matrix, wie „stark“ sie auf Vektoren wirkt.
- ▶ Eine gegebene Matrix kann Vektoren nicht beliebig stark verlängern. Es gibt für jede Matrix einen „Maximal-Verlängerungs-Faktor“

Der „Maximal-Verlängerungs-Faktor“ ist eine Matrixnorm

Matrixnormen: Beispiel

Wie stark „wirkt“ $A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$ auf verschiedene Vektoren?

Vergleichen Sie $\|\mathbf{x}\|$ mit $\|A\mathbf{x}\|$ für verschiedene \mathbf{x} und verschiedene Normen.

$$\mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \begin{pmatrix} 2 \\ -1 \end{pmatrix}$$

Welcher dieser Vektoren verlängert sich am meisten

- ▶ in der 1-Norm,
- ▶ in der ∞ -Norm,
- ▶ in der 2-Norm (aufwändiger zu rechnen wegen $\sqrt{\quad}$)

Verschiedene Matrixnormen

Achtung—Ergänzung zum Skriptum!

Die 1-, 2- und ∞ -Normen lassen sich von den entsprechenden Vektornormen ableiten: Sie geben für die Rechenoperation $\mathbf{y} = A \cdot \mathbf{x}$ an, um wieviel \mathbf{y} gegenüber \mathbf{x} *maximal vergrößert* wird.

$\|A\|_1$ *Einsnorm*: maximale Spaltenbetragssumme

$\|A\|_2$ *Zweinnorm*: größter Singulärwert

$\|A\|_\infty$ *Unendlich-Norm*: maximale Zeilenbetragssumme

Die Zweinnorm lässt sich nicht so einfach berechnen wie Eins- oder Unendlichnorm.

MATLAB: $\|A\|_1 = \text{norm}(A, 1)$, $\|A\|_2 = \text{norm}(A)$, $\|A\|_\infty = \text{norm}(A, \text{Inf})$.

Matrixnorm, Definition

Kurzfassung: Eine Matrixnorm ist eine Norm

Matrizen lassen sich addieren und mit Skalaren multiplizieren. In diesem Sinn verhalten sie sich genauso wie Vektoren des \mathbb{R}^n .

Alles, was sich wie ein Vektor verhält, ist ein „Vektor“: Die $m \times n$ -Matrizen bilden einen *Vektorraum*. Der Begriff „Norm“ wird genau so definiert wie die Norm von Vektoren des \mathbb{R}^n .

Eine *Norm* im $\mathbb{R}^m \times \mathbb{R}^n$ ist eine Funktion, die jeder $m \times n$ -Matrix A eine nichtnegative reelle Zahl $\|A\| \in \mathbb{R}_0^+$ zuordnet, wobei gilt:

- ▶ Nur die Nullmatrix hat Norm 0: $\|A\| = 0 \Rightarrow A = 0$
- ▶ Skalar α lässt sich als Betrag herausheben: $\|\alpha \cdot A\| = |\alpha| \cdot \|A\|$
- ▶ Die Dreiecksungleichung $\|A + B\| \leq \|A\| + \|B\|$

Rechenregeln

für die 1-, 2- oder ∞ - Norm

Zusätzlich zu den Norm-Axiomen

$$\|A\| = 0 \quad \Rightarrow \quad A = O$$

$$\|\alpha \cdot A\| = |\alpha| \cdot \|A\|$$

$$\|A + B\| \leq \|A\| + \|B\|$$

gelten für die 1-, 2- oder ∞ - Norm auch

$$\|A \cdot B\| \leq \|A\| \cdot \|B\|$$

$$\|A \cdot \mathbf{x}\| \leq \|A\| \cdot \|\mathbf{x}\|$$

Vergleiche Absolutbetrag: $|a \cdot b| = |a| \cdot |b|$

Rechenregeln

für die 1-, 2- oder ∞ - Norm

Zusätzlich zu den Norm-Axiomen

$$\|A\| = 0 \quad \Rightarrow \quad A = O$$

$$\|\alpha \cdot A\| = |\alpha| \cdot \|A\|$$

$$\|A + B\| \leq \|A\| + \|B\|$$

gelten für die 1-, 2- oder ∞ - Norm auch

$$\|A \cdot B\| \leq \|A\| \cdot \|B\|$$

$$\|A \cdot \mathbf{x}\| \leq \|A\| \cdot \|\mathbf{x}\|$$

Vergleiche Absolutbetrag: $|a \cdot b| = |a| \cdot |b|$

Frobeniusnorm

... noch eine weitere Norm

Die Frobenius-Norm $\|A\|_F$ wird so ähnlich berechnet wie die Vektor-Zweinorm: *Quadrieren, summieren, Wurzel ziehen*

$$\text{Frobenius-Norm: } \|A\|_F = \sqrt{\sum a_{ij}^2}$$

Die Frobeniusnorm lässt sich leichter berechnen als die Matrix-Zweinorm und dient zu deren Abschätzung:

$$\|A\|_2 \leq \|A\|_F$$

Auch für $\|A\|_F$ gelten neben den Norm-Axiome noch die Rechenregeln $\|A \cdot B\|_F \leq \|A\|_F \cdot \|B\|_F$ und $\|A \cdot \mathbf{x}\|_2 \leq \|A\|_F \|\mathbf{x}\|_2$

MATLAB: $\|A\|_F = \text{norm}(A, 'fro')$.

Frobeniusnorm

... noch eine weitere Norm

Die Frobenius-Norm $\|A\|_F$ wird so ähnlich berechnet wie die Vektor-Zweinnorm: *Quadrieren, summieren, Wurzel ziehen*

$$\text{Frobenius-Norm: } \|A\|_F = \sqrt{\sum a_{ij}^2}$$

Die Frobeniusnorm lässt sich leichter berechnen als die Matrix-Zweinnorm und dient zu deren Abschätzung:

$$\|A\|_2 \leq \|A\|_F$$

Auch für $\|A\|_F$ gelten neben den Norm-Axiome noch die Rechenregeln $\|A \cdot B\|_F \leq \|A\|_F \cdot \|B\|_F$ und $\|A \cdot \mathbf{x}\|_2 \leq \|A\|_F \|\mathbf{x}\|_2$

MATLAB: $\|A\|_F = \text{norm}(A, 'fro')$.

Matrixnormen (das Kleingedruckte)

Was hier dasteht, ist nicht wichtig,
wenn 's nicht dastünd', wär's nicht richtig.

Die lockere Erklärung „*Matrixnorm ist maximaler Verlängerungsfaktor*“ ist mathematisch korrekt für 1-, 2- und ∞ -Norm, wenn Vektorlängen in den jeweiligen Normen gemessen werden.

Die Formulierung „*Matrixnorm ist obere Schranke für Verlängerungsfaktor*“ umfasst auch noch die Frobeniusnorm, wenn Vektorlängen in der 2-Norm gemessen werden.

Auch die Vorschrift $\|A\| = \max_{i,j} |a_{ij}|$ erfüllt die drei Bedingungen einer Norm, ist aber nicht immer eine obere Schranke für den Verlängerungsfaktor.

Gliederung 3. Vorlesung

① Wiederholung:

Vektoren, vektorwertige Funktionen
Fixpunkt-Iteration: Theorie

② Normen

Vektornorm
Norm misst Distanz
Matrixnorm

③ Konvergenz von Fixpunkt-Iterationen

Kontrahierende Abbildung
Jacobi-Matrix

④ Newton-Verfahren für nichtlineare Systeme

⑤ Fixpunkt-Iteration für Lineare Systeme

Jacobi-Verfahren
Beispiel: Wärmeleitungsgleichung

Kontrahierende Abbildung $\mathbb{R}^n \rightarrow \mathbb{R}^n$

Bildpunkte liegen näher beisammen als Originalpunkte

Definition

Die Funktion $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ heißt eine *kontrahierende Abbildung*, wenn für alle Punkte $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ die Bildpunkte $\Phi(\mathbf{x}), \Phi(\mathbf{y})$ näher beisammen liegen:

$$\|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \leq C\|\mathbf{x} - \mathbf{y}\|, \quad C < 1$$

(man müsste noch dazu sagen, welche Norm man verwendet – man kann jene wählen, mit der man am einfachsten rechnet)

Verallgemeinerung: Φ kann auch nur in einem Teilbereich $B \subset \mathbb{R}^n$ kontrahierend wirken.

Kontrahierende Abbildung $\mathbb{R}^n \rightarrow \mathbb{R}^n$

Bildpunkte liegen näher beisammen als Originalpunkte

Definition

Die Funktion $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ heißt eine *kontrahierende Abbildung*, wenn für alle Punkte $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ die Bildpunkte $\Phi(\mathbf{x}), \Phi(\mathbf{y})$ näher beisammen liegen:

$$\|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \leq C\|\mathbf{x} - \mathbf{y}\|, \quad C < 1$$

(man müsste noch dazu sagen, welche Norm man verwendet – man kann jene wählen, mit der man am einfachsten rechnet)

Verallgemeinerung: Φ kann auch nur in einem Teilbereich $B \subset \mathbb{R}^n$ kontrahierend wirken.

Fixpunkt-Iteration konvergiert für kontrahierende Abbildungen

Beweis-Idee

- ▶ Unterschiedliche Punkte liegen nach Anwendung von Φ näher beisammen
- ▶ Startwert und Fixpunkt liegen nach Anwendung von Φ näher beisammen
- ▶ Fortgesetzte Anwendung bringt Werte immer näher zum Fixpunkt

Fixpunkt-Iteration konvergiert für kontrahierende Abbildungen

Die Funktion $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ besitze einen Fixpunkt \mathbf{x}^* .

Sei ferner B ein Bereich um den Fixpunkt \mathbf{x}^* in der Form $B = \{\mathbf{x} : \|\mathbf{x}^* - \mathbf{x}\| < r\}$, sodass Φ in B eine *kontrahierende Abbildung* ist. Es gilt also

$$\|\Phi(\mathbf{x}) - \Phi(\mathbf{y})\| \leq C\|\mathbf{x} - \mathbf{y}\|, \quad C < 1$$

für alle $\mathbf{x}, \mathbf{y} \in B$.

Dann konvergiert die Fixpunkt-Iteration $\mathbf{x}^{(k+1)} = \Phi(\mathbf{x}^{(k)})$ mindestens linear gegen \mathbf{x}^* für alle $\mathbf{x}^{(0)} \in B$.

Bemerkungen

Die Formulierung des Satzes auf der vorigen Folie setzt die *Existenz* eines Fixpunktes voraus. Dadurch wird der Konvergenz-Beweis kurz und schmerzlos.

Eine etwas allgemeinere Formulierung und ein technisch aufwändigerer Beweis zeigen, dass aus der Kontraktions-Eigenschaft auch schon die *Existenz und Eindeutigkeit* eines Fixpunktes folgen. Das ist der berühmte *Fixpunktsatz von Banach*.

Fixpunkt-Iteration konvergiert für $\|D_\phi\| < 1$

Achtung—Ergänzung zum Skriptum!

Die Matrix der partiellen Ableitungen

$$D_\phi = \begin{bmatrix} \frac{\partial \phi_1}{\partial x_1} & \frac{\partial \phi_1}{\partial x_2} & \cdots & \frac{\partial \phi_1}{\partial x_n} \\ \frac{\partial \phi_2}{\partial x_1} & \frac{\partial \phi_2}{\partial x_2} & \cdots & \frac{\partial \phi_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \phi_n}{\partial x_1} & \frac{\partial \phi_n}{\partial x_2} & \cdots & \frac{\partial \phi_n}{\partial x_n} \end{bmatrix}$$

heißt die *Jacobi-Matrix* der Abbildung $\Phi : \mathbb{R}^n \rightarrow \mathbb{R}^n$. Ist in einem Fixpunkt von Φ die Norm¹ $\|D_\phi\| < 1$, dann konvergiert die Fixpunkt-Iteration für Startwerte in einer Umgebung des Fixpunktes.

¹1-, 2-, ∞ - oder F -Norm

Jacobi-Matrix D_f verallgemeinert Ableitung f'

Ableitung misst Änderung Funktionswerte im Verhältnis zu Änderung Eingabewerte

Ableitung f' einer Funktion $f : \mathbb{R} \rightarrow \mathbb{R}$

$$\Delta f = f' \cdot \Delta x \quad (+\text{Terme höherer Ordnung in } \Delta x)$$

Matrix D_f der part. Ableitungen von $f : \mathbb{R}^n \rightarrow \mathbb{R}^k$

Zusammenhang zwischen Input-Vektor $\Delta \mathbf{x}$ und Output-Vektor $\Delta \mathbf{f}$.

$$\Delta \mathbf{f} = D_f \cdot \Delta \mathbf{x} \quad (+\text{Terme höherer Ordnung in } \|\Delta \mathbf{x}\|)$$

Beispiel im Skriptum

Seite 19

Die Funktion Φ ist hier ein Vektor aus zwei reellwertigen Funktionen ϕ_1 und ϕ_2 , der Vektor \mathbf{x} hat zwei Komponenten x_1 und x_2 .

$$\Phi(\mathbf{x}) = \begin{bmatrix} \phi_1(x_1, x_2) \\ \phi_2(x_1, x_2) \end{bmatrix} = \begin{bmatrix} \frac{1}{4}(x_2 - x_1 x_2 + 1) \\ \frac{1}{6}(x_1 - \log(x_1 x_2) + 2) \end{bmatrix}$$

$$D_\phi = \begin{bmatrix} \frac{-x_2}{4} & \frac{1-x_1}{4} \\ \frac{1-\frac{1}{x_1}}{6} & \frac{-1}{6x_2} \end{bmatrix}$$

Ausgewertet für $\mathbf{x} = \begin{bmatrix} 0,35 \\ 0,64 \end{bmatrix}$ (\approx Fixpunkt) $D_\phi = \begin{bmatrix} -0,160 & 0,163 \\ -0,310 & -0,260 \end{bmatrix}$

$$\|D_\phi\|_1 = 0,4695 \quad \|D_\phi\|_2 = 0,4051 \quad \|D_\phi\|_\infty = 0,5699 \quad \|D_\phi\|_F = 0,4644$$

Gliederung 3. Vorlesung

① Wiederholung:

Vektoren, vektorwertige Funktionen
Fixpunkt-Iteration: Theorie

② Normen

Vektornorm
Norm misst Distanz
Matrixnorm

③ Konvergenz von Fixpunkt-Iterationen

Kontrahierende Abbildung
Jacobi-Matrix

④ Newton-Verfahren für nichtlineare Systeme

⑤ Fixpunkt-Iteration für Lineare Systeme

Jacobi-Verfahren
Beispiel: Wärmeleitungsgleichung

Newton-Verfahren für nichtlineare Systeme

Gegeben eine differenzierbare vektorwertige Funktion $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ und ein Startwert $\mathbf{x}^{(0)}$. Gesucht eine Nullstelle von \mathbf{f} .

Iterationsvorschrift

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta\mathbf{x}^{(k)}$$

mit $\Delta\mathbf{x}^{(k)}$ als Lösung von $D_{\mathbf{f}}(\mathbf{x}^{(k)})\Delta\mathbf{x}^{(k)} = -\mathbf{f}(\mathbf{x}^{(k)})$

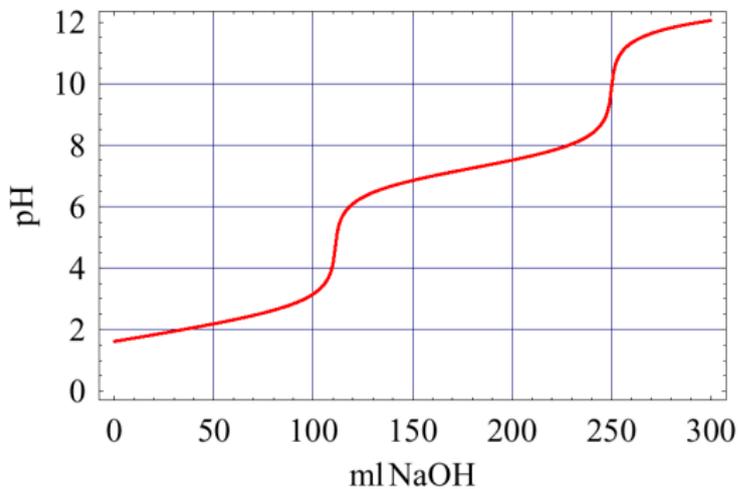
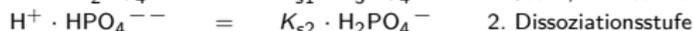
Auch dieses Verfahren ist ein Fixpunktverfahren. Die Iterationsfunktion lässt sich formal schreiben (**Vorsicht—explizites Ausrechnen der Inversen ist nicht ratsam!**)

$$\Phi(\mathbf{x}) = \mathbf{x} - D_{\mathbf{f}}(\mathbf{x})^{-1}\mathbf{f}(\mathbf{x})$$

Beispiel:

Titration von 0,1m-Phosphorsäure mit 1m-NaOH

Ein System sechs nichtlinearer Gleichungen bestimmt die Konzentration der undissoziierten H_3PO_4 ; der Kationen H^+ , Na^+ ; und Anionen OH^- , H_2PO_4^- , HPO_4^{2-} , PO_4^{3-} .



Die Lösung dieses Gleichungssystems für gegebene Na^+ -Konzentration bestimmt den pH-Wert. Im Verlauf der Titration schwanken die einzelnen Konzentrationen über viele Zehnerpotenzen. Vorgefertigte Lösungsverfahren haben damit große Schwierigkeiten.

Lösung nichtlinearer Gleichungssysteme: Übersicht der Methoden

- ▶ Fixpunkt-Iteration: Allgemeine Formulierung; kein Rezept, um günstiges Φ zu finden.
- ▶ Newton-Raphson: Standard-Verfahren. Varianten:
 - ▶ *gedämpft*: langsamere, aber verlässlichere Konvergenz.
 - ▶ *fixe Jacobi-Matrix*: lin. Konvergenz, weniger Rechenaufwand
 - ▶ *genäherte Jacobi-Matrix*: wenn exakte Ableitungen nicht verfügbar sind, Näherung durch Differenzenquotienten.
- ▶ MATLAB Optimization Toolbox: `fzero` löst nichtlineare Gleichungssysteme — mehrdimensionale Verallgemeinerung von `fzero`.

Gliederung 3. Vorlesung

① Wiederholung:

Vektoren, vektorwertige Funktionen
Fixpunkt-Iteration: Theorie

② Normen

Vektornorm
Norm misst Distanz
Matrixnorm

③ Konvergenz von Fixpunkt-Iterationen

Kontrahierende Abbildung
Jacobi-Matrix

④ Newton-Verfahren für nichtlineare Systeme

⑤ Fixpunkt-Iteration für Lineare Systeme

Jacobi-Verfahren
Beispiel: Wärmeleitungsgleichung

Lineares Gleichungssystem in n Gleichungen und Unbekannten

$$\begin{array}{ccccccc} a_{11} x_1 + a_{12} x_2 + & \dots & + a_{1n} x_n & = & b_1 \\ a_{21} x_1 + a_{22} x_2 + & \dots & + a_{2n} x_n & = & b_2 \\ \vdots & & & & \vdots \\ a_{n1} x_1 + a_{n2} x_2 + & \dots & + a_{nn} x_n & = & b_n \end{array}$$

In Matrixschreibweise: $A\mathbf{x} = \mathbf{b}$.

Gleichungssysteme lassen sich in Matrix-Schreibweise übersichtlich und prägnant formulieren.

Machen Sie sich mit den Bezeichnungen und Regeln der Matrizenrechnung vertraut!

Lösungsverfahren für lineare Gleichungssysteme

Grundlegende Unterscheidung: direkte und iterative Verfahren

Direkte Verfahren sind Varianten des Gaußschen Eliminationsverfahrens (LR -Zerlegung). Üblich bis $n \approx 10.000$ Unbekannten. Direkte Verfahren sind allgemeiner anwendbar und rechnen zumeist schneller, sofern die Matrix im schnell zugänglichen Speicher des Rechners Platz hat.

Iterative Verfahren finden schrittweise verbesserte Näherungslösungen. Üblich für $n \gg 10.000$. Iterative Methoden sind nur für spezielle Matrixtypen anwendbar, die beispielsweise bei partiellen Differentialgleichungen auftreten.

Jacobi-Verfahren

Idee: Löse jede Gleichung nach ihrem Diagonal-Term auf.

Standard-Form, 3×3 -Beispiel

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 = b_1$$

$$a_{21} x_1 + a_{22} x_2 + a_{23} x_3 = b_2$$

$$a_{31} x_1 + a_{32} x_2 + a_{33} x_3 = b_3$$

Auflösen nach Diagonal-Term \rightarrow Fixpunkt-Form

$$a_{11} x_1 = b_1 - a_{12} x_2 - a_{13} x_3$$

$$a_{22} x_2 = b_2 - a_{21} x_1 - a_{23} x_3$$

$$a_{33} x_3 = b_3 - a_{31} x_1 - a_{32} x_2$$

Jacobi-Verfahren

Idee: Löse jede Gleichung nach ihrem Diagonal-Term auf.

Standard-Form, 3×3 -Beispiel

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 = b_1$$

$$a_{21} x_1 + a_{22} x_2 + a_{23} x_3 = b_2$$

$$a_{31} x_1 + a_{32} x_2 + a_{33} x_3 = b_3$$

Auflösen nach Diagonal-Term \rightarrow Fixpunkt-Form

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

Jacobi-Verfahren

Idee: Löse jede Gleichung nach ihrem Diagonal-Term auf.

Standard-Form, 3×3 -Beispiel

$$a_{11} x_1 + a_{12} x_2 + a_{13} x_3 = b_1$$

$$a_{21} x_1 + a_{22} x_2 + a_{23} x_3 = b_2$$

$$a_{31} x_1 + a_{32} x_2 + a_{33} x_3 = b_3$$

Auflösen nach Diagonal-Term \rightarrow Fixpunkt-Form

$$x_1 = (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11}$$

$$x_2 = (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22}$$

$$x_3 = (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33}$$

setze Startwerte ein, iteriere

Stationäre Wärmeleitungs-Aufgabe

Bilanzgleichungen für Temperaturen in neun Zellen eines Finite-Volumen-Rechengitters

Randtemperaturen sind vorgegeben, gesucht sind T_1, \dots, T_9 .

	0	0	0	
100	T_7	T_8	T_9	0
100	T_4	T_5	T_6	0
100	T_1	T_2	T_3	0
	0	0	0	

Die Finite-Volumen-Diskretisierung der Wärmeleitungsgleichung liefert neun Bilanzgleichungen.

$$\begin{aligned}4T_1 - T_2 - T_4 &= 100 \\-T_1 + 4T_2 - T_3 - T_5 &= 0 \\-T_2 + 4T_3 - T_6 &= 0 \\-T_1 + 4T_4 - T_5 - T_7 &= 100 \\-T_2 - T_4 + 4T_5 - T_6 - T_8 &= 0 \\-T_3 - T_5 + 4T_6 - T_9 &= 0 \\-T_4 + 4T_7 - T_6 &= 100 \\-T_5 - T_7 + 4T_8 - T_6 &= 0 \\-T_6 - T_8 + 4T_9 &= 0\end{aligned}$$

Stationäre Wärmeleitung: Matrix-Struktur

So eine Matrix-Struktur tritt in vielen Modellproblemen auf

$$\begin{bmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & & & 4 & -1 & 0 & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & 0 & -1 & 4 & & & -1 \\ & & & -1 & & & 4 & -1 & 0 \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & 0 & -1 & 4 \end{bmatrix} \cdot \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{bmatrix} = \begin{bmatrix} 100 \\ 0 \\ 0 \\ 100 \\ 0 \\ 0 \\ 100 \\ 0 \\ 0 \end{bmatrix}$$

(Matrix-Elemente 0 sind größtenteils gar nicht mehr aufgeschrieben).

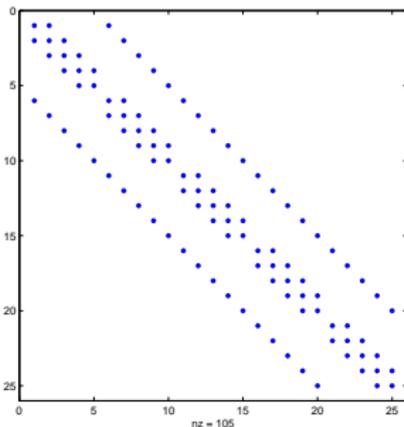
Schablone

Jede Gleichung verknüpft höchstens 5 Unbekannte nach dem Muster „4-mal der Wert im Punkt minus Werte im Süden, Westen, Norden, Osten“

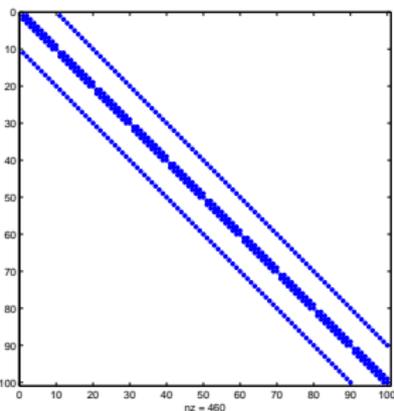
Matrix-Struktur bei größeren Gittern

Die typische Fünf-Band-Struktur eines diskreten Poisson-Problems

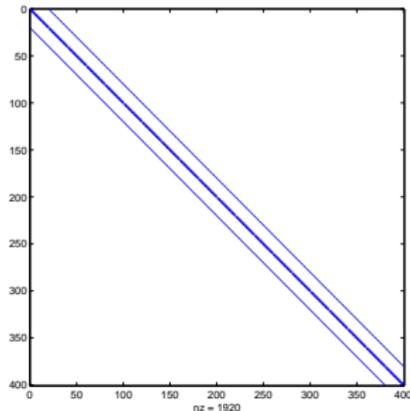
5×5



10×10



20×20



Schwach besetzte Matrix

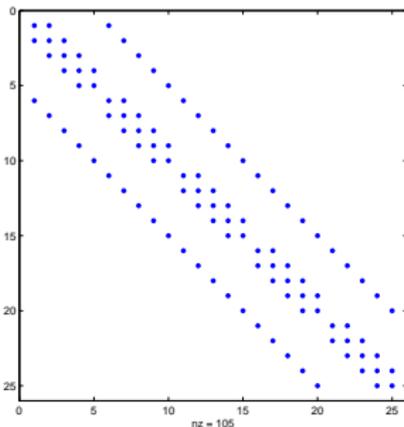
Die meisten Positionen sind mit Nullen besetzt

Für schwach besetzte Matrizen lassen sich auch Systeme mit Millionen von Unbekannten lösen. Bei voll besetzten Matrizen dieser Größe wäre Gauß-Elimination völlig unmöglich.

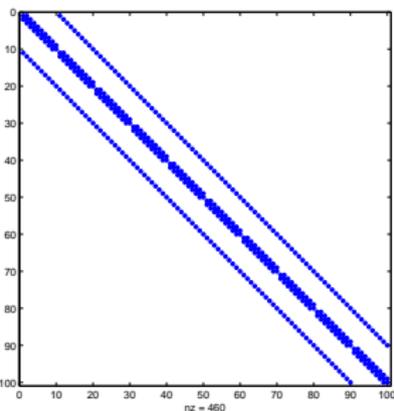
Matrix-Struktur bei größeren Gittern

Die typische Fünf-Band-Struktur eines diskreten Poisson-Problems

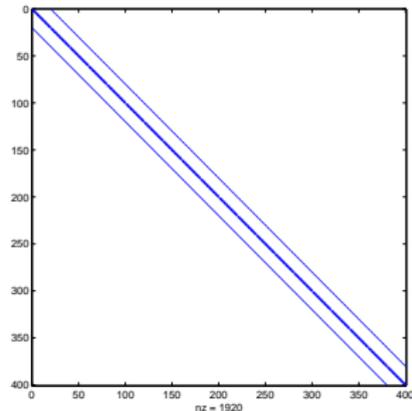
5×5



10×10



20×20



Schwach besetzte Matrix

Die meisten Positionen sind mit Nullen besetzt

Für schwach besetzte Matrizen lassen sich auch Systeme mit Millionen von Unbekannten lösen. Bei voll besetzten Matrizen dieser Größe wäre Gauß-Elimination völlig unmöglich.

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

$$4T_1 - T_2 - T_4 = 100$$

$$-T_1 + 4T_2 - T_3 - T_5 = 0$$

$$-T_2 + 4T_3 - T_6 = 0$$

$$-T_1 + 4T_4 - T_5 - T_7 = 100$$

$$-T_2 - T_4 + 4T_5 - T_6 - T_8 = 0$$

$$-T_3 - T_5 + 4T_6 - T_9 = 0$$

$$-T_4 + 4T_7 - T_6 = 100$$

$$-T_5 - T_7 + 4T_8 - T_6 = 0$$

$$-T_6 - T_8 + 4T_9 = 0$$

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

$$4T_1 - T_2 - T_4 = 100$$

$$-T_1 + 4T_2 - T_3 - T_5 = 0$$

$$-T_2 + 4T_3 - T_6 = 0$$

$$-T_1 + 4T_4 - T_5 - T_7 = 100$$

$$-T_2 - T_4 + 4T_5 - T_6 - T_8 = 0$$

$$-T_3 - T_5 + 4T_6 - T_9 = 0$$

$$-T_4 + 4T_7 - T_6 = 100$$

$$-T_5 - T_7 + 4T_8 - T_6 = 0$$

$$-T_6 - T_8 + 4T_9 = 0$$

$$4T_1 = 100 + T_2 + T_4$$

$$4T_2 = 0 + T_1 + T_3 + T_5$$

$$4T_3 = 0 + T_2 + T_6$$

$$4T_4 = 100 + T_1 + T_5 + T_7$$

$$4T_5 = 0 + T_2 + T_4 + T_6 + T_8$$

$$4T_6 = 0 + T_3 + T_5 + T_9$$

$$4T_7 = 100 + T_4 - T_6$$

$$4T_8 = 0 + T_5 + T_7 + T_6$$

$$4T_9 = 0 + T_6 + T_8$$

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

$$4T_1 - T_2 - T_4 = 100$$

$$-T_1 + 4T_2 - T_3 - T_5 = 0$$

$$-T_2 + 4T_3 - T_6 = 0$$

$$-T_1 + 4T_4 - T_5 - T_7 = 100$$

$$-T_2 - T_4 + 4T_5 - T_6 - T_8 = 0$$

$$-T_3 - T_5 + 4T_6 - T_9 = 0$$

$$-T_4 + 4T_7 - T_6 = 100$$

$$-T_5 - T_7 + 4T_8 - T_6 = 0$$

$$-T_6 - T_8 + 4T_9 = 0$$

$$T_1 = (100 + T_2 + T_4)/4$$

$$T_2 = (T_1 + T_3 + T_5)/4$$

$$T_3 = (T_2 + T_6)/4$$

$$T_4 = (100 + T_1 + T_5 + T_7)/4$$

$$T_5 = (T_2 + T_4 + T_6 + T_8)/4$$

$$T_6 = (T_3 + T_5 + T_9)/4$$

$$T_7 = (100 + T_4 + T_6)/4$$

$$T_8 = (T_5 + T_7 + T_6)/4$$

$$T_9 = (T_6 + T_8)/4$$

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

$$\begin{array}{rcl} 4T_1 - T_2 - T_4 & = & 100 \\ -T_1 + 4T_2 - T_3 - T_5 & = & 0 \\ -T_2 + 4T_3 - T_6 & = & 0 \\ -T_1 + 4T_4 - T_5 - T_7 & = & 100 \\ -T_2 - T_4 + 4T_5 - T_6 - T_8 & = & 0 \\ -T_3 - T_5 + 4T_6 - T_9 & = & 0 \\ -T_4 + 4T_7 - T_6 & = & 100 \\ -T_5 - T_7 + 4T_8 - T_6 & = & 0 \\ -T_6 - T_8 + 4T_9 & = & 0 \end{array} \quad \begin{array}{l} T_1 = (100 + T_2 + T_4)/4 \\ T_2 = (T_1 + T_3 + T_5)/4 \\ T_3 = (T_2 + T_6)/4 \\ T_4 = (100 + T_1 + T_5 + T_7)/4 \\ T_5 = (T_2 + T_4 + T_6 + T_8)/4 \\ T_6 = (T_3 + T_5 + T_9)/4 \\ T_7 = (100 + T_4 + T_6)/4 \\ T_8 = (T_5 + T_7 + T_6)/4 \\ T_9 = (T_6 + T_8)/4 \end{array}$$

**Einsetzen von Startwerten rechts liefert verbesserte Werte links →
iteriere!**

Iterative Lösungsverfahren

Einfaches Prinzip: ersetze Wert der Gitterzelle durch Mittelwert der Nachbarn

- ▶ Einfach zu programmieren, wenn T -Werte (inklusive Randwerte) in 2D-Feld gespeichert sind:

```
for i=2:n-1
    for j=2:n-1
        T(i,j) = 0.25*(T(i-1,j)+T(i+1,j)...
                    +T(i,j-1)+T(i,j+1));
    end
end
```

- ▶ In dieser Form ein *Gauß-Seidel-Verfahren*.
- ▶ Es fehlen noch Konvergenztests, Abbruchbedingungen,...
- ▶ Allerdings nach heutigen Standards *viel* zu langsam

Iterative Lösungsverfahren

Einfaches Prinzip: ersetze Wert der Gitterzelle durch Mittelwert der Nachbarn

- ▶ Einfach zu programmieren, wenn T -Werte (inklusive Randwerte) in 2D-Feld gespeichert sind:

```
for i=2:n-1
    for j=2:n-1
        T(i,j) = 0.25*(T(i-1,j)+T(i+1,j)...
                    +T(i,j-1)+T(i,j+1));
    end
end
```

- ▶ In dieser Form ein *Gauß-Seidel-Verfahren*.
- ▶ Es fehlen noch Konvergenztests, Abbruchbedingungen,...
- ▶ Allerdings nach heutigen Standards *viel* zu langsam

Iterative Lösungsverfahren

Einfaches Prinzip: ersetze Wert der Gitterzelle durch Mittelwert der Nachbarn

- ▶ Einfach zu programmieren, wenn T -Werte (inklusive Randwerte) in 2D-Feld gespeichert sind:

```
for i=2:n-1
    for j=2:n-1
        T(i,j) = 0.25*(T(i-1,j)+T(i+1,j)...
                    +T(i,j-1)+T(i,j+1));
    end
end
```

- ▶ In dieser Form ein *Gauß-Seidel-Verfahren*.
- ▶ Es fehlen noch Konvergenztests, Abbruchbedingungen,...
- ▶ Allerdings nach heutigen Standards *viel* zu langsam

Iterative Lösungsverfahren

Einfaches Prinzip: ersetze Wert der Gitterzelle durch Mittelwert der Nachbarn

- ▶ Einfach zu programmieren, wenn T -Werte (inklusive Randwerte) in 2D-Feld gespeichert sind:

```
for i=2:n-1
    for j=2:n-1
        T(i,j) = 0.25*(T(i-1,j)+T(i+1,j)...
                    +T(i,j-1)+T(i,j+1));
    end
end
```

- ▶ In dieser Form ein *Gauß-Seidel-Verfahren*.
- ▶ Es fehlen noch Konvergenztests, Abbruchbedingungen,...
- ▶ Allerdings nach heutigen Standards *viel* zu langsam

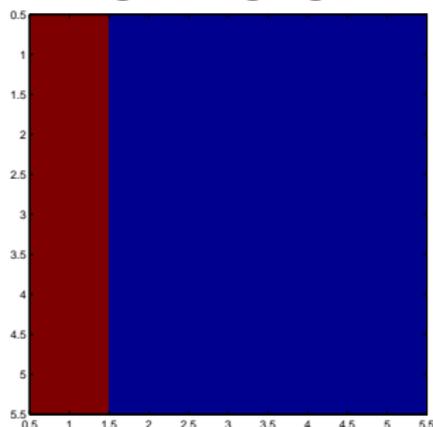
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein *Jacobi-Verfahren*: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	0	0	0	0
100	0	0	0	0
100	0	0	0	0
	0	0	0	

Anfangsbedingung



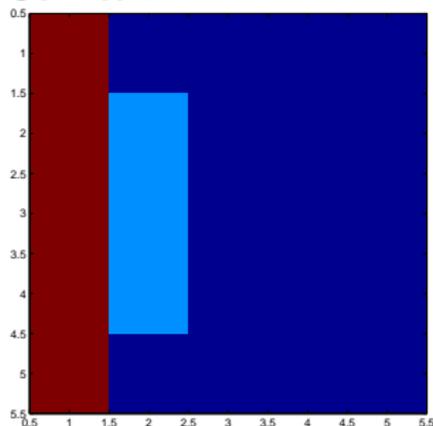
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein *Jacobi-Verfahren*: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	25	0	0	0
100	25	0	0	0
100	25	0	0	0
	0	0	0	

Schritt 1



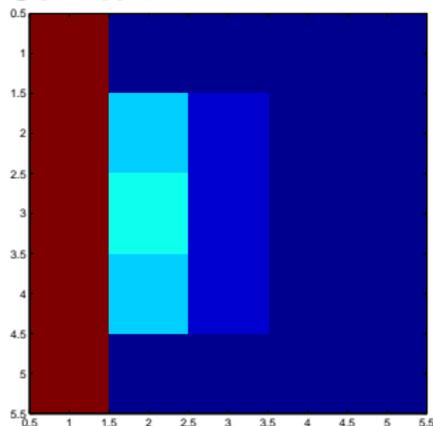
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein *Jacobi-Verfahren*: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	31	6	0	0
100	38	6	0	0
100	31	6	0	0
	0	0	0	

Schritt 2



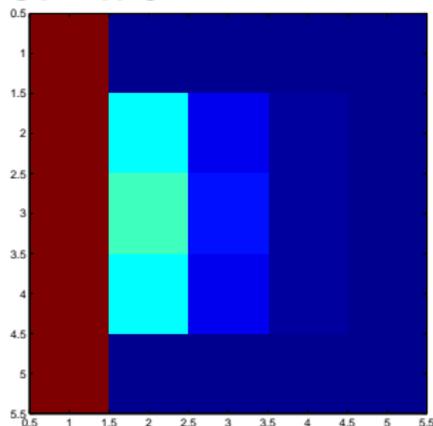
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein *Jacobi-Verfahren*: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	36	9	2	0
100	42	13	2	0
100	36	9	2	0
	0	0	0	

Schritt 3



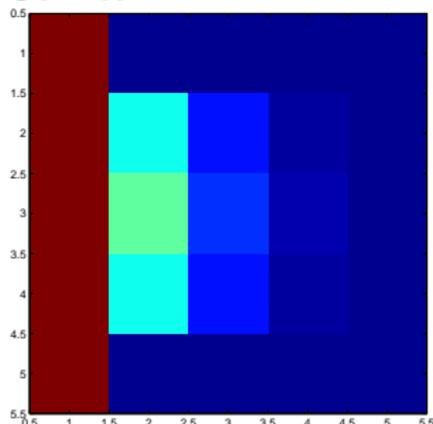
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein *Jacobi-Verfahren*: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	38	13	3	0
100	46	16	4	0
100	38	13	3	0
	0	0	0	

Schritt 4



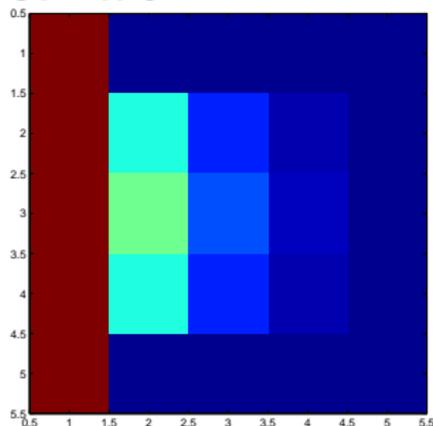
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein *Jacobi-Verfahren*: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	40	14	4	0
100	48	19	5	0
100	40	14	4	0
	0	0	0	

Schritt 5



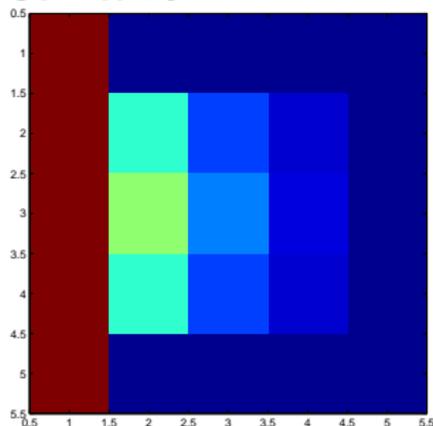
Ablauf der Iteration

bei 3×3 inneren Zellen

In dieser Form ein *Jacobi-Verfahren*: alle Gitter-Werte werden gleichzeitig aktualisiert. Noch langsamer als das Gauß-Seidel-Verfahren, aber für Parallel-Rechner geeignet.

	0	0	0	
100	42	18	7	0
100	52	24	9	0
100	42	18	7	0
	0	0	0	

Schritt 10



Stationäre Temperaturverteilung

100 × 100-Gitter, insgesamt 10.000 Jacobi-Iterationen

