Mehrdimensionale Gleichungen und Iterationen (2)

3. Vorlesung

170 004 Numerische Methoden I

Alexander Steinicke

Montanuniversität Leoben

19. Oktober 2023

Mehrdimensionale Gleichungen und Iterationen, Teil 2

1. Fixpunkt-Iteration: Theorie (Nachtrag 2. Vorl.)
   - Konvergenzordnung
   - Kontrahierende Abbildung, Konvergenz
   - Einfache Konvergenzbedingung
   - Jacobi-Matrix
   - Nette Beispiele von mehrdimensionalen Iterationen
2. Newton-Verfahren für nichtlineare Systeme (Nachtrag 2. Vorl.)
   - Beispiele, Aufgaben
3. Vorschau: Fixpunkt-Iteration für Lineare Systeme
   - Direkte und iterative Gleichungslöser
   - Jacobi-Verfahren
   - Beispiel: Wärmeleitungsgleichung

Gliederung 3. Vorlesung

1. Fixpunkt-Iteration: Theorie (Nachtrag 2. Vorl.)
   - Konvergenzordnung
   - Kontrahierende Abbildung, Konvergenz
   - Einfache Konvergenzbedingung
   - Jacobi-Matrix
   - Nette Beispiele von mehrdimensionalen Iterationen
2. Newton-Verfahren für nichtlineare Systeme (Nachtrag 2. Vorl.)
   - Beispiele, Aufgaben
3. Vorschau: Fixpunkt-Iteration für Lineare Systeme
   - Direkte und iterative Gleichungslöser
   - Jacobi-Verfahren
   - Beispiel: Wärmeleitungsgleichung

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 $|\Phi'| < 1$ oder $\|D_{\mathbf{\Phi}}\| < 1$

Konvergenzordnung

Angenommen, eine Iteration $\mathbf{x}^{(k+1)} = \mathbf{\Phi}(\mathbf{x}^{(k)})$ für $k = 0, 1, 2 \ldots$ konvergiert zu $\mathbf{x}^\star$, und die Fehlerschranke $\epsilon^{(k)}$ schätzt den Fehler:

$$\|\mathbf{x}^{(k)} - \mathbf{x}^\star\| \leq \epsilon^{(k)} \ .$$

Neue Fehlerschranke mindestens um Faktor C kleiner als. . .

- . . . alte Fehlerschranke: lineare Konvergenz (wenn $C < 1$), also

$$\epsilon^{(k+1)} \leq C\, \epsilon^{(k)}$$

- . . . das Quadrat des alten Fehlers: quadratische Konvergenz; typisch für Newton-Verfahren.

$$\epsilon^{(k+1)} \leq C(\epsilon^{(k)})^2$$

- . . . (allgemein) die p-te Potenz des alten Fehlers, $p \geq 1$: Konvergenz p-ter Ordnung. Bei Sekanten-Verfahren ist $p \approx 1.61$.

$$\epsilon^{(k+1)} \leq C(\epsilon^{(k)})^p$$

Konvergenzordnung

Angenommen, eine Iteration $\mathbf{x}^{(k+1)} = \mathbf{\Phi}(\mathbf{x}^{(k)})$ für $k = 0, 1, 2 \ldots$ konvergiert zu $\mathbf{x}^\star$, und die Fehlerschranke $\epsilon^{(k)}$ schätzt den Fehler:

$$\|\mathbf{x}^{(k)} - \mathbf{x}^\star\| \leq \epsilon^{(k)} \ .$$

Neue Fehlerschranke mindestens um Faktor C kleiner als. . .

- . . . alte Fehlerschranke: lineare Konvergenz (wenn $C < 1$), also

$$\epsilon^{(k+1)} \leq C\, \epsilon^{(k)}$$

- . . . das Quadrat des alten Fehlers: quadratische Konvergenz; typisch für Newton-Verfahren.

$$\epsilon^{(k+1)} \leq C(\epsilon^{(k)})^2$$

- . . . (allgemein) die p-te Potenz des alten Fehlers, $p \geq 1$: Konvergenz p-ter Ordnung. Bei Sekanten-Verfahren ist $p \approx 1.61$.

$$\epsilon^{(k+1)} \leq C(\epsilon^{(k)})^p$$

Konvergenzordnung

Angenommen, eine Iteration $\mathbf{x}^{(k+1)} = \mathbf{\Phi}(\mathbf{x}^{(k)})$ für $k = 0, 1, 2 \ldots$ konvergiert zu $\mathbf{x}^\star$, und die Fehlerschranke $\epsilon^{(k)}$ schätzt den Fehler:

$$\|\mathbf{x}^{(k)} - \mathbf{x}^\star\| \leq \epsilon^{(k)} \ .$$

Neue Fehlerschranke mindestens um Faktor C kleiner als. . .

- . . . alte Fehlerschranke: lineare Konvergenz (wenn $C < 1$), also

$$\epsilon^{(k+1)} \leq C\, \epsilon^{(k)}$$

- . . . das Quadrat des alten Fehlers: quadratische Konvergenz; typisch für Newton-Verfahren.

$$\epsilon^{(k+1)} \leq C(\epsilon^{(k)})^2$$

- . . . (allgemein) die p-te Potenz des alten Fehlers, $p \geq 1$: Konvergenz p-ter Ordnung. Bei Sekanten-Verfahren ist $p \approx 1.61$.

$$\epsilon^{(k+1)} \leq C(\epsilon^{(k)})^p$$

Konvergenzordnung

Angenommen, eine Iteration $\mathbf{x}^{(k+1)} = \mathbf{\Phi}(\mathbf{x}^{(k)})$ für $k = 0, 1, 2 \ldots$ konvergiert zu $\mathbf{x}^\star$, und die Fehlerschranke $\epsilon^{(k)}$ schätzt den Fehler:

$$\|\mathbf{x}^{(k)} - \mathbf{x}^\star\| \leq \epsilon^{(k)} \ .$$

Neue Fehlerschranke mindestens um Faktor C kleiner als. . .

- . . . alte Fehlerschranke: lineare Konvergenz (wenn $C < 1$), also

$$\epsilon^{(k+1)} \leq C\, \epsilon^{(k)}$$

- . . . das Quadrat des alten Fehlers: quadratische Konvergenz; typisch für Newton-Verfahren.

$$\epsilon^{(k+1)} \leq C(\epsilon^{(k)})^2$$

- . . . (allgemein) die p-te Potenz des alten Fehlers, $p \geq 1$: Konvergenz p-ter Ordnung. Bei Sekanten-Verfahren ist $p \approx 1.61$.

$$\epsilon^{(k+1)} \leq C(\epsilon^{(k)})^p$$

Konvergenzordnung: typisches Verhalten

Faustregeln

- Lineare Konvergenz braucht eine fixe Anzahl von Schritten pro gültiger Stelle. Je kleiner C, desto rascher nimmt Genauigkeit zu.
- Quadratische Konvergenz verdoppelt pro Schritt (ungefähr, hängt auch von C ab) die Zahl der korrekten Dezimalstellen. Beispiel:

$$\text{Fehler } \epsilon^{(k)} < 10^{-3} \Rightarrow \epsilon^{(k+1)} < C \cdot (10^{-3})^2 = C \cdot 10^{-6}$$
$$\epsilon^{(k)} < 10^{-6} \Rightarrow \epsilon^{(k+1)} < C \cdot (10^{-6})^2 = C \cdot 10^{-12}$$

- Sekanten-Regel: etwa 60% mehr korrekte Stellen pro Schritt.

Die Faustregeln für Newton- und Sekantenverfahren gelten nur bei genügend kleinen Fehlern; umso besser, je mehr Stellen bereits korrekt sind.

Konvergenzordnung

ausführliche Definition

Definition

Ein Iterationsverfahren

$$\mathbf{x}^{(k+1)} = \mathbf{\Phi}(\mathbf{x}^{(k)}) \quad k = 0, 1, 2 \ldots$$

mit Iterationsfunktion $\mathbf{\Phi}: \mathbb{R}^n \to \mathbb{R}^n$, Fixpunkt $\mathbf{x}^\star \in \mathbb{R}^n$ und Fehlerschranken $||\mathbf{x}^{(k)} - \mathbf{x}^\star|| \leq \epsilon^{(k)}$ heißt lokal konvergent von (mindestens) p-ter Ordnung ($p \geq 1$), wenn für alle Startwerte $\mathbf{x}^{(0)}$, die genügend nahe an $\mathbf{x}^\star$ liegen, gilt

$$\epsilon^{(k+1)} \leq C \left[\epsilon^{(k)}\right]^p$$

und $C < 1$, falls $p = 1$.

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

Bildpunkte liegen näher beisammen als Originalpunkte

Definition

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

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

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

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

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

Bildpunkte liegen näher beisammen als Originalpunkte

Definition

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

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

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

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

Kontrahierende Abbildung $\Rightarrow$ Iteration konvergiert

Die Fixpunkt-Iteration konvergiert für kontrahierende Abbildungen

Beweis-Idee

- Unterschiedliche Punkte liegen nach Anwendung von $\mathbf{\Phi}$ näher beisammen
- Startwert und Fixpunkt liegen nach Anwendung von $\mathbf{\Phi}$ näher beisammen
- Fortgesetzte Anwendung bringt Werte immer näher zum Fixpunkt

Ein Konvergenzsatz

Voraussetzungen

- Die Funktion $\mathbf{\Phi} : \mathbb{R}^n \to \mathbb{R}^n$ hat einen Fixpunkt $\mathbf{x}^\star$.
- Um den Fixpunkt $\mathbf{x}^\star$ ist in einem Bereich $B = \{\mathbf{x} : \|\mathbf{x}^\star - \mathbf{x}\| < r\}$ die Funktion $\mathbf{\Phi}$ eine kontrahierende Abbildung.

Es gilt also für alle $\mathbf{x}, \mathbf{y} \in B$

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

Satz

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

Kurzfassung: Fixpunkt-Iteration konvergiert für kontrahierende Abbildungen

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.

Einfache Konvergenzbedingung für $\phi\colon \mathbb{R} \to \mathbb{R}$

Eine direkte Folgerung aus dem Konvergenz-Satz für kontrahierende Abbildungen

Das Fixpunktverfahren konvergiert lokal, falls $|\phi'(x^\star)| < 1$.

Genauer:
Ist $\phi(x)$ in einer Umgebung des Fixpunktes $x^\star$ stetig differenzierbar und $|\phi'(x^\star)| < 1$, so konvergiert die Fixpunkt-Iteration

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

mindestens linear mit $C \approx |\phi'(x^\star)|$ gegen $x^\star$ für alle $x^{(0)}$ in der Nähe des Fixpunktes.

Der Fehler nimmt $\approx$ um den Faktor C pro Iteration ab

Interpretation der Bedingung $|\phi'(x^\star)| < 1$.

- Locker gesagt: Fixpunkt-Iteration konvergiert,
  wenn $\phi(x)$ in einer Umgebung „nicht besonders stark“ von x abhängt.
- Ableitung ϕ' misst, wie stark sich $\phi(x)$ ändert, wenn sich x ändert.
- Der Konvergenzsatz quantifiziert, „wie stark“ ϕ von x abhängen darf, damit Iteration konvergiert.
- Bedingung $|\phi'(x^\star)| < 1$ bedeutet: ϕ ist kontrahierende Abbildung (zumindest in einer Umgebung von $x^\star$)

Achtung:

Wahl des Anfangspunkt ist wichtig! Ist im Interval $[x^\star, x]$ die Ableitung $|\phi'|$ irgendwo > 1, muss das Verfahren nicht konvergieren!

Beispiel: $\phi(x) = \frac{9}{4}x(1-x)$

Zwei Fixpunkte: $x_1^\star = 0, x_2^\star = \frac{5}{9}$.
Einsetzen der Fixpunkte in $\phi'(x) = \frac{9}{4}(1-2x)$ liefert

$$|\phi'(0)| = \frac{9}{4} > 1 \qquad \left|\phi'\left(\frac{5}{9}\right)\right| = \frac{1}{4} < 1$$

Folgerungen:

- Für Startwerte in der Nähe von $x_2^\star = \frac{5}{9}$ konvergiert die Fixpunkt-Iteration.
- $\phi(x)$ ändert sich dort nur etwa $1/4$ so stark, wenn sich x-Werte ändern.
- Ein Fehler im Eingabewert bewirkt einen $\approx 1/4$ so großen Fehler im Resultat.
- Wiederholtes Einsetzen macht den Fehler immer kleiner

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

(Jetzt geht es um mehrdimensionale Iterationen)

Die $n \times n$ Matrix der partiellen Ableitungen

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

heißt die Jacobi-Matrix der Abbildung $\mathbf{\Phi} : \mathbb{R}^n \to \mathbb{R}^n$. Ist in einem Fixpunkt von $\mathbf{\Phi}$ die Norm[1] $\|D_{\mathbf{\Phi}}\| < 1$, dann konvergiert die Fixpunkt-Iteration für Startwerte in einer Umgebung des Fixpunktes.

[1] 1-, 2-, ∞- oder F-Norm

Jacobi-Matrix D_{Φ} verallgemeinert Ableitung ϕ'

Wie ändern sich Funktionswerte bei kleinen Änderungen der Eingabewerte?

Der skalare Fall:

Ableitung ϕ' einer Funktion $\phi : \mathbb{R} \to \mathbb{R}$ beschreibt:

Zusammenhang Input-Änderung $\Delta x \to$ Ergebnis-Änderung Δf.

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

Mehrdimensionale Verallgemeinerung:

Matrix D_{Φ} der part. Ableitungen von $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ beschreibt:

Zusammenhang Input-Änderungs-Vektor $\Delta\mathbf{x} \to$ Änderungs-Vektor $\Delta\Phi$.

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

Jacobi-Matrix $D_{\mathbf{\Phi}}$ verallgemeinert Ableitung ϕ'

Wie ändern sich Funktionswerte bei kleinen Änderungen der Eingabewerte?

Der skalare Fall:

Ableitung ϕ' einer Funktion $\phi : \mathbb{R} \to \mathbb{R}$ beschreibt:

Zusammenhang Input-Änderung $\Delta x \to$ Ergebnis-Änderung Δf.

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

Mehrdimensionale Verallgemeinerung:

Matrix $D_{\mathbf{\Phi}}$ der part. Ableitungen von $\mathbf{\Phi} : \mathbb{R}^n \to \mathbb{R}^n$ beschreibt:

Zusammenhang Input-Änderungs-Vektor $\Delta\mathbf{x} \to$ Änderungs-Vektor $\Delta\mathbf{\Phi}$.

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

Beispiel im Skriptum

Seite 21

Die Funktion $\mathbf{\Phi}$ 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.

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

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

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

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

Mehrdimensionale Iterationen: nette Beispiele

Newton-Fraktal

Das Polynom $z^3 - 1$ hat in $\mathbb{C}$ drei Nullstellen: $z_1 = 1,\ z_{2,3} = -\frac{1}{2} \pm \mathrm{i}\frac{\sqrt{3}}{2}$.
Zu welcher Nullstelle konvergiert das Newton-Verfahren für einen bestimmten Startwert $z^{(0)} \in \mathbb{C}$?

Barnsley-Farn

Ein Farn-ähnliches Fraktal entsteht durch Iteration von Funktionen der Art

$$\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2, \quad \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

Mandelbrot-Menge

Iteriere in $\mathbb{C}$, ausgehend von $z^{(0)} = 0$, mit der Funktion

$$f : \mathbb{C} \to \mathbb{C}, \quad z \mapsto z^2 + c\ .$$

Für welche $c \in \mathbb{C}$ konvergiert diese Iteration (fast)?

Mehrdimensionale Iterationen: nette Beispiele

Newton-Fraktal

Das Polynom $z^3 - 1$ hat in $\mathbb{C}$ drei Nullstellen: $z_1 = 1,\ z_{2,3} = -\frac{1}{2} \pm \mathrm{i}\frac{\sqrt{3}}{2}$.
Zu welcher Nullstelle konvergiert das Newton-Verfahren für einen bestimmten Startwert $z^{(0)} \in \mathbb{C}$?

Barnsley-Farn

Ein Farn-ähnliches Fraktal entsteht durch Iteration von Funktionen der Art

$$\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2, \quad \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

Mandelbrot-Menge

Iteriere in $\mathbb{C}$, ausgehend von $z^{(0)} = 0$, mit der Funktion

$$f : \mathbb{C} \to \mathbb{C}, \quad z \mapsto z^2 + c\ .$$

Für welche $c \in \mathbb{C}$ konvergiert diese Iteration (fast)?

Mehrdimensionale Iterationen: nette Beispiele

Newton-Fraktal

Das Polynom $z^3 - 1$ hat in $\mathbb{C}$ drei Nullstellen: $z_1 = 1,\ z_{2,3} = -\frac{1}{2} \pm \mathrm{i}\frac{\sqrt{3}}{2}$.
Zu welcher Nullstelle konvergiert das Newton-Verfahren für einen bestimmten Startwert $z^{(0)} \in \mathbb{C}$?

Barnsley-Farn

Ein Farn-ähnliches Fraktal entsteht durch Iteration von Funktionen der Art

$$\mathbf{f} : \mathbb{R}^2 \to \mathbb{R}^2, \quad \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

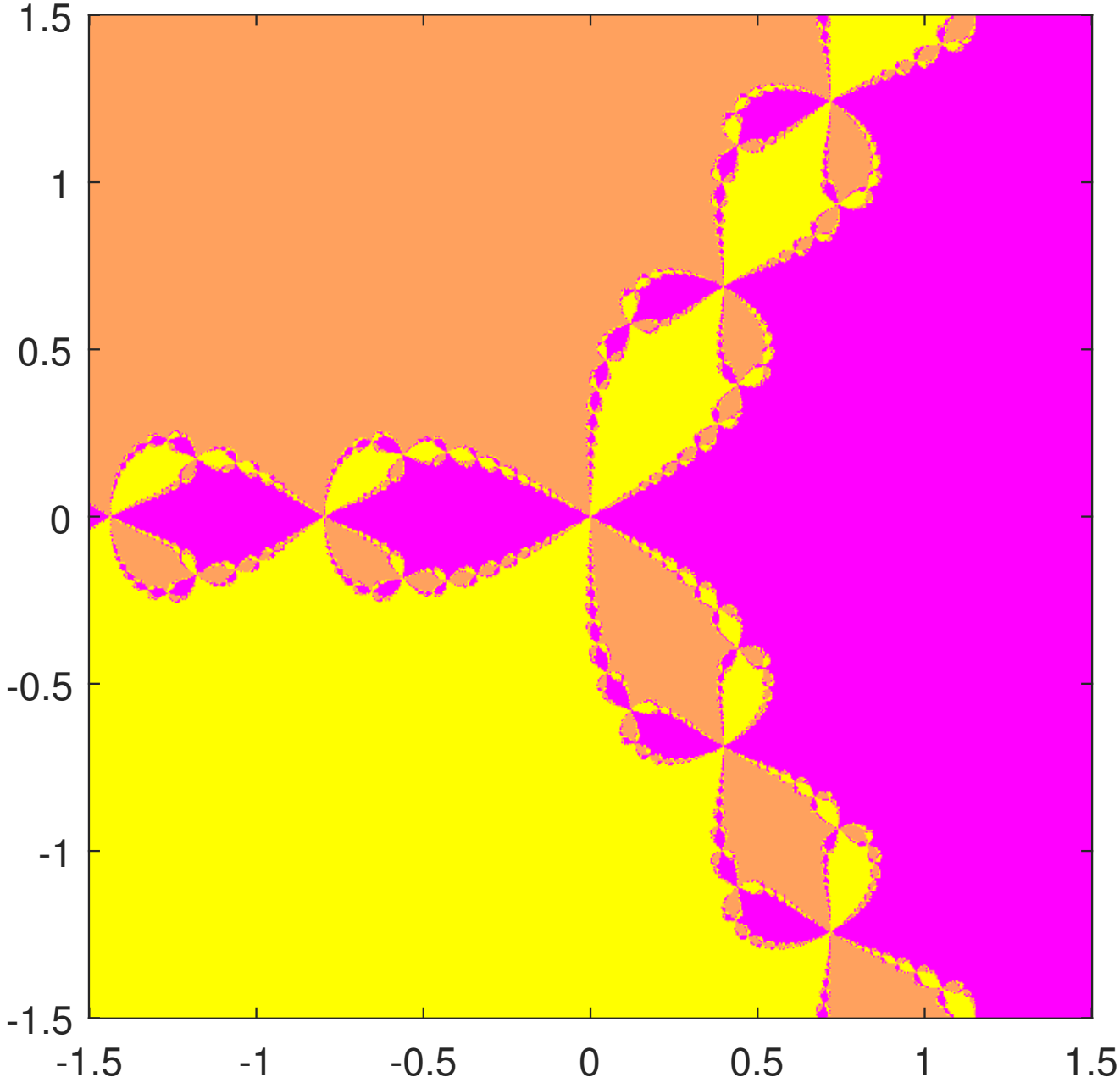
Mandelbrot-Menge

Iteriere in $\mathbb{C}$, ausgehend von $z^{(0)} = 0$, mit der Funktion

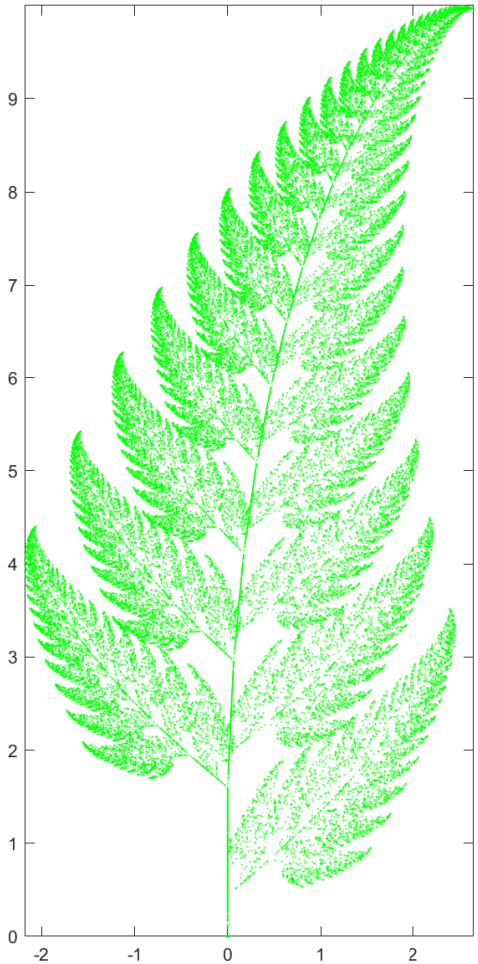
$$f : \mathbb{C} \to \mathbb{C}, \quad z \mapsto z^2 + c\ .$$

Für welche $c \in \mathbb{C}$ konvergiert diese Iteration (fast)?

Mehrdimensionale Iterationen: Newton-Fraktal

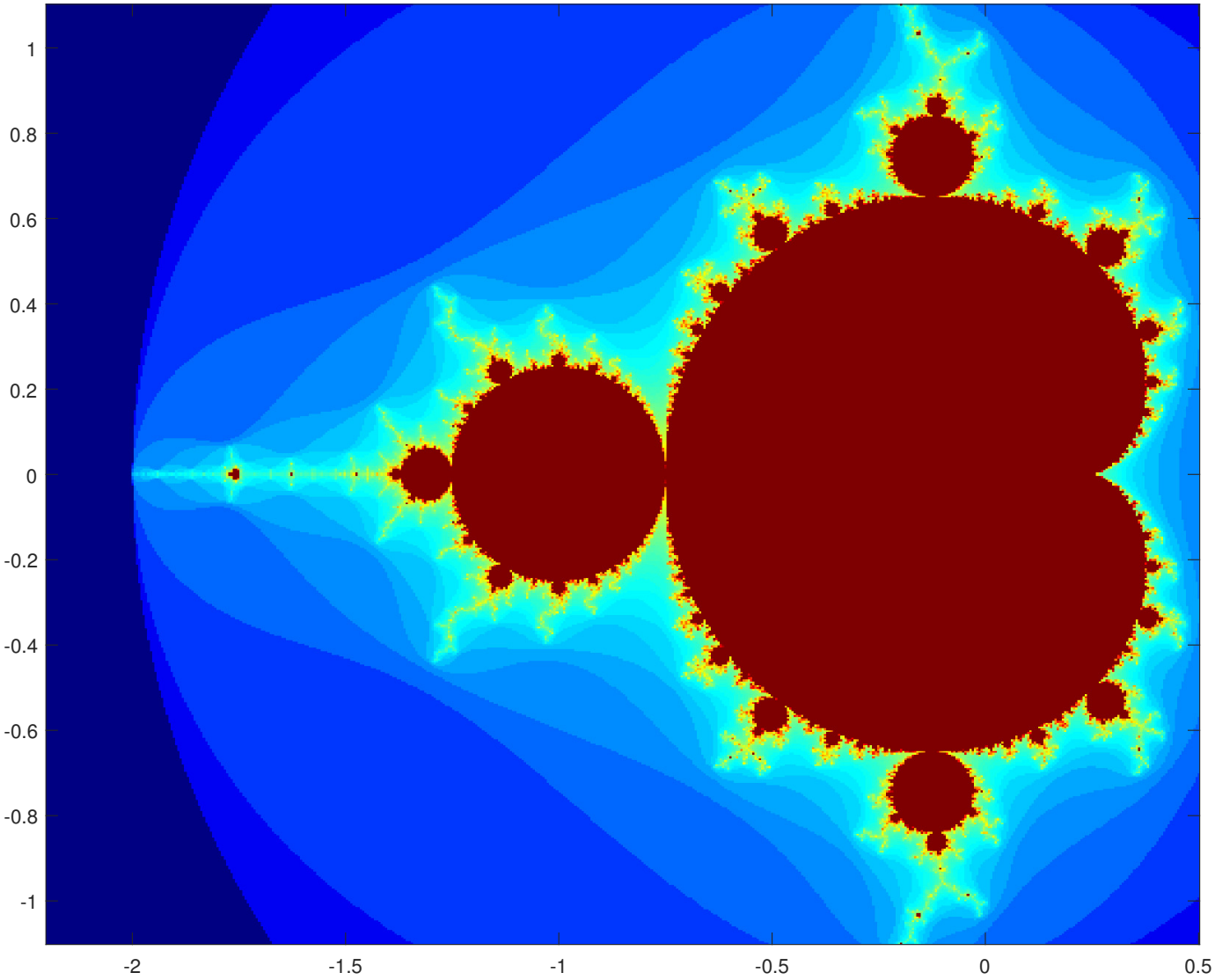


Mehrdimensionale Iterationen: Barnsley-Farn

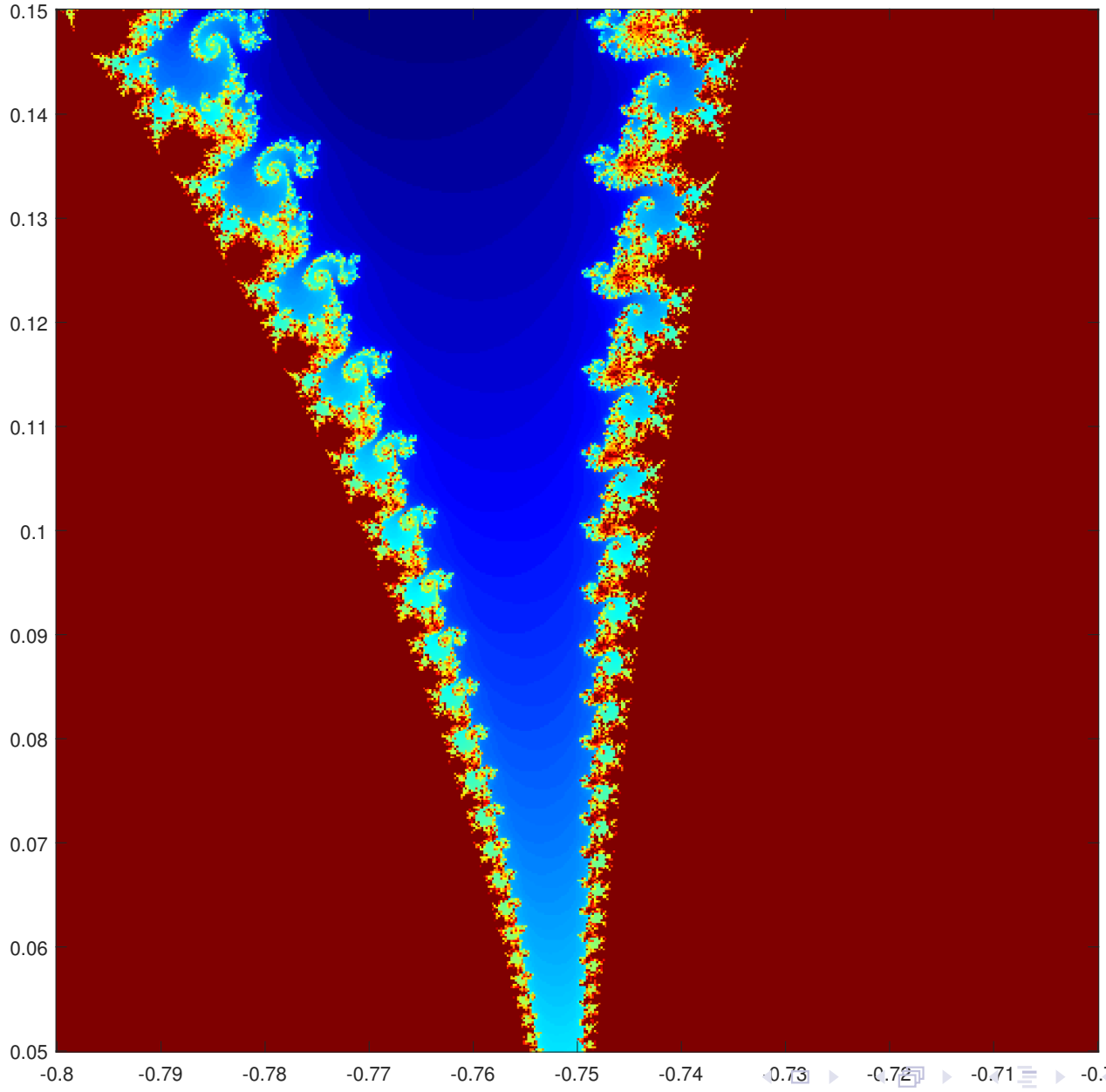


Mehrdimensionale Iterationen: Barnsley-Farn

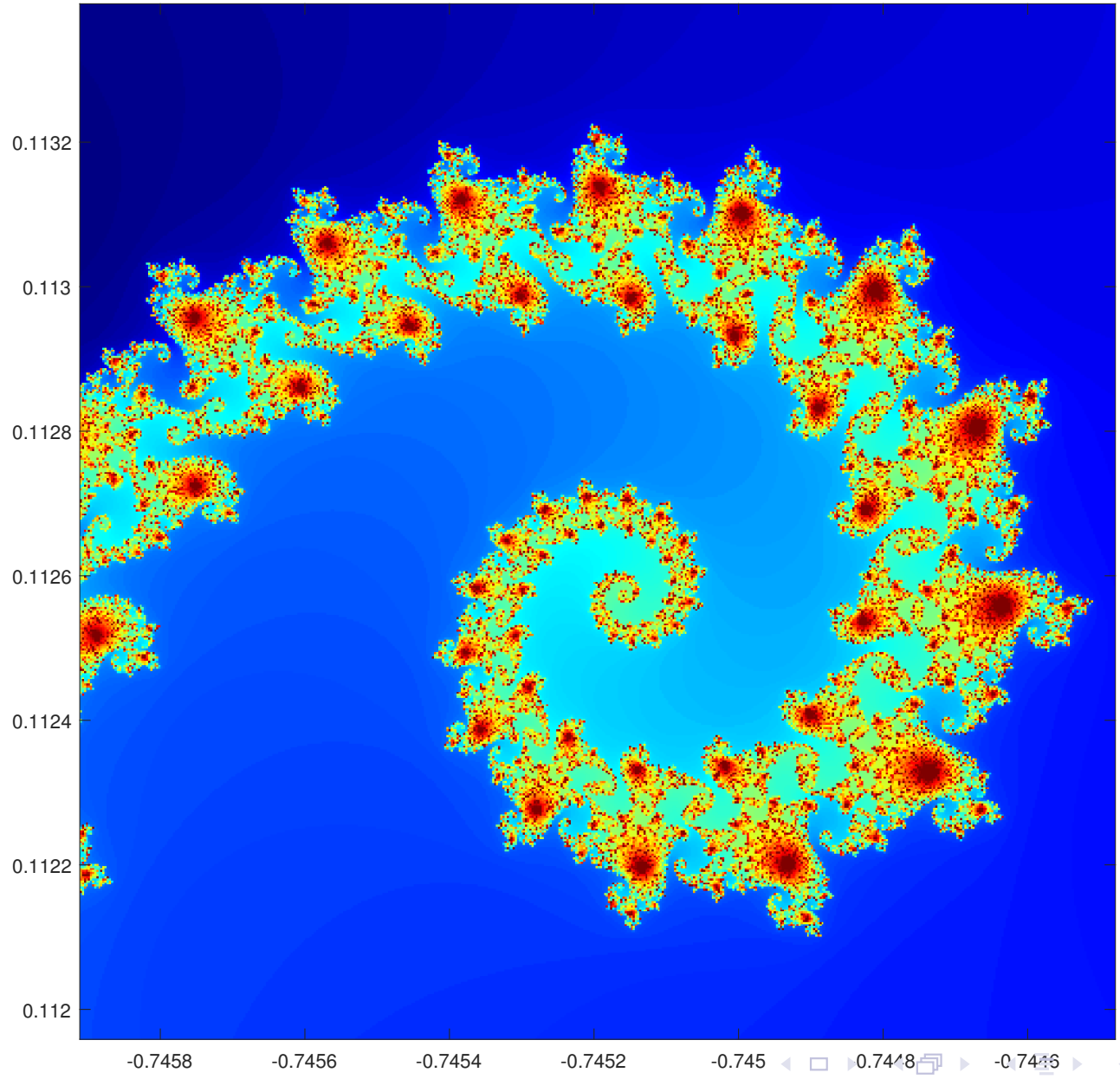
Mehrdimensionale Iterationen: Mandelbrot-Menge



Mehrdimensionale Iterationen: Mandelbrot-Menge



Mehrdimensionale Iterationen: Mandelbrot-Menge



Gliederung 3. Vorlesung

Newton-Verfahren für nichtlineare Systeme

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

Iterationsvorschrift

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

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

Quadratische Konvergenz bei Startwerten nahe einfachen Nullstellen.

Siehe Abschnitt 2.5 im Skript!

Beispiel:

Skript Seite 26

Gesucht ist eine Nullstelle von $\mathbf{f}$ in der Nähe eines Startwertes $\mathbf{x}^{(0)}$.
Die Funktion $\mathbf{f}$ und die Jacobi-Matrix D_f sind hier

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} 4x - y + xy - 1 \\ -x + 6y + \log(xy) - 2 \end{bmatrix}, \qquad D_f = \begin{bmatrix} 4 + y & -1 + x \\ -1 + \frac{1}{x} & 6 + \frac{1}{y} \end{bmatrix}.$$

Newton-Verfahren: Varianten

Gedämpftes Newton-Verfahren

reduziert den Korrekturvektor um einen Faktor $\omega < 1$.

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

Konvergiert bei schlechten Startwerten verlässlicher – aber nur linear.

Vereinfachtes Newton-Verfahren

wertet die Jacobi-Matrix nicht für jeden Schritt erneut aus. Schnellere Rechnung, aber nur lineare Konvergenz.

Die Lösung des Systems $D_f(\mathbf{x}^{(k)})\Delta \mathbf{x}^{(k)} = -\mathbf{f}(\mathbf{x}^{(k)})$ ist rechenaufwändig. Sobald Näherung einigermaßen brauchbar: Speichere D_f und behalte sie für die restlichen Schritte bei.

(Über Rechenaufwand und effizientes Lösen linearer Gleichungssysteme werden wir in den nächsten Einheiten sprechen...)

Muster-Prüfungsaufgabe

Gegeben sei das nichtlineare Gleichungssystem

$$\begin{aligned} -\frac{x^3}{48} + y^2 x + 2 &= 0 \\ x^3 - x + y^3 - y - 1 &= 0 \end{aligned}$$

1. Wie lautet die Jacobi-Matrix für das gegebene Gleichungssystem?
2. Ausgehend von der Näherungslösung $x^{(0)} = 0;\ y^{(0)} = 1$ bestimme man mit Hilfe des Newton-Raphson Verfahrens $x^{(2)}$ und $y^{(2)}$! (Wenn Sie korrekt rechnen, lassen sich auftretende lineare Gleichungssysteme „einfach“ lösen)
3. Verwenden Sie nun für den Schritt 2 das vereinfachte Newton-Verfahren.

Lösung nichtlinearer Gleichungssysteme: Übersicht der Methoden

- Fixpunkt-Iteration: Allgemeine Formulierung; kein Rezept, um günstiges $\mathbf{\Phi}$ 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: `fsolve` löst nichtlineare Gleichungssysteme — mehrdimensionale Verallgemeinerung von `fzero`.

Ein Prüfungsbeispiel

Die Funktion

$$\phi(x) = \frac{18 - 30\,x + 23\,x^2 - 4\,x^3}{9}$$

hat Fixpunkte für $x = 3/4, 2$ und 3.
Überprüfen Sie mithilfe der Konvergenzsätze für die verschiedenen Fixpunkte: Konvergiert die Fixpunkt-Iteration $x^{(k+1)} = \phi(x^{(k)})$, und wenn ja, mit welcher Konvergenzordnung?

Prüfungsbeispiel

Gegeben sei die Funktion

$$\phi(x) = ax(1-x) \qquad \text{für ein } a \neq 0$$

1. Zeigen Sie: $x = 0$ und $x = (a-1)/a$ sind Fixpunkte von ϕ.
2. In welchem Bereich muss a liegen, damit eine Fixpunkt-Iteration lokal zu $x = 0$ konvergiert?
3. In welchem Bereich muss a liegen, damit eine Fixpunkt-Iteration lokal nach $x = (a-1)/a$ konvergiert?
4. Für welchen Wert von a folgt lokal quadratische Konvergenz zum Fixpunkt $x = (a-1)/a$?

Prüfungsbeispiel

Gegeben sei die Funktion $f(x) = x^3 - 1$.

1. Wie lautet die reelle Nullstelle von f?
2. Zeigen Sie: Das Newton-Verfahren zur Nullstellenbestimmung führt auf die Iterationsvorschrift

$$x = \frac{1}{3\,x^2} + \frac{2\,x}{3}$$

3. Leiten Sie die Konvergenzordnung dieser Iteration her.

Gliederung 3. Vorlesung

Lineares Gleichungssystem in n Gleichungen und Unbekannten

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

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: einfache Fixpunkt-Iteration

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

Standard-Form, 3×3-Beispiel

$$\begin{aligned} a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 &= b_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 &= b_2 \\ a_{31}\,x_1 + a_{32}\,x_2 + a_{33}\,x_3 &= b_3 \end{aligned}$$

Auflösen nach Diagonal-Term $\rightarrow$ Fixpunkt-Form

$$\begin{aligned} a_{11}\,x_1 &= b_1 - a_{12}x_2 - a_{13}x_3 \\ a_{22}\,x_2 &= b_2 - a_{21}x_1 - a_{23}x_3 \\ a_{33}\,x_3 &= b_3 - a_{31}x_1 - a_{32}x_2 \end{aligned}$$

Jacobi-Verfahren: einfache Fixpunkt-Iteration

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

Standard-Form, 3×3-Beispiel

$$\begin{aligned} a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 &= b_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 &= b_2 \\ a_{31}\,x_1 + a_{32}\,x_2 + a_{33}\,x_3 &= b_3 \end{aligned}$$

Auflösen nach Diagonal-Term $\rightarrow$ Fixpunkt-Form

$$\begin{aligned} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{aligned}$$

Jacobi-Verfahren: einfache Fixpunkt-Iteration

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

Standard-Form, 3×3-Beispiel

$$\begin{aligned} a_{11}\,x_1 + a_{12}\,x_2 + a_{13}\,x_3 &= b_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + a_{23}\,x_3 &= b_2 \\ a_{31}\,x_1 + a_{32}\,x_2 + a_{33}\,x_3 &= b_3 \end{aligned}$$

Auflösen nach Diagonal-Term $\rightarrow$ Fixpunkt-Form

$$\begin{aligned} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{aligned}$$

setze Startwerte ein, iteriere

Stationäre Wärmeleitungs-Aufgabe

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

Randtemperaturen sind vorgegeben, gesucht sind $T_1, \ldots, T_9$.

	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-Volums-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 großteils gar nicht mehr aufgeschrieben).

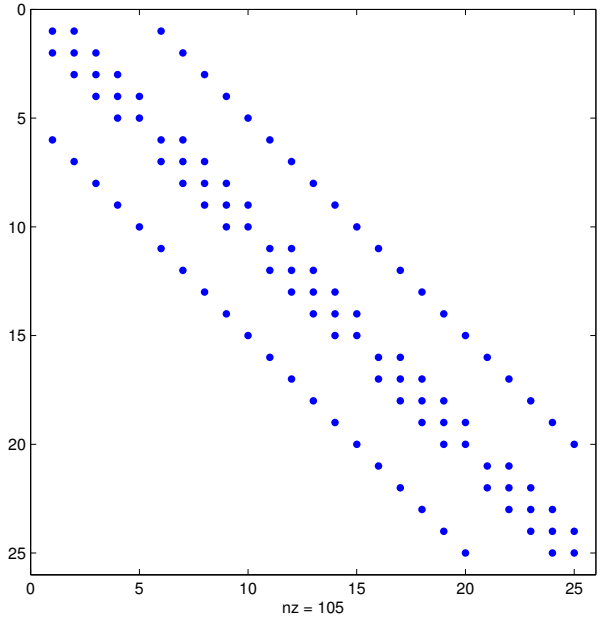
Schema

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

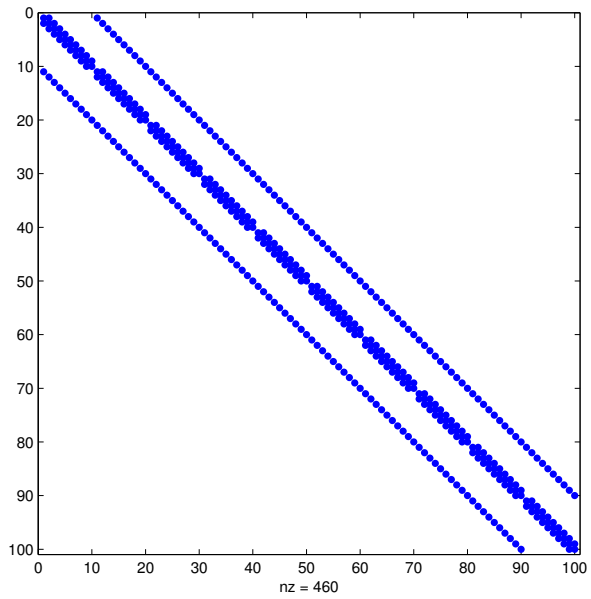
Matrix-Struktur bei größeren Gittern

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

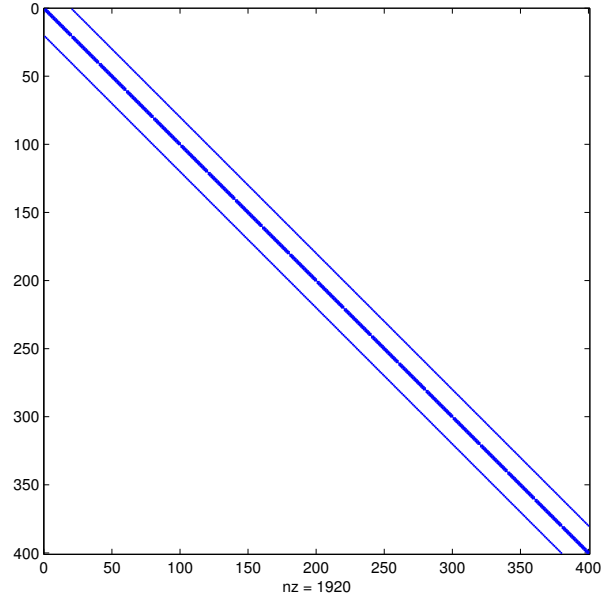
5×5



10×10



20×20



Spärlich besetzte Matrix

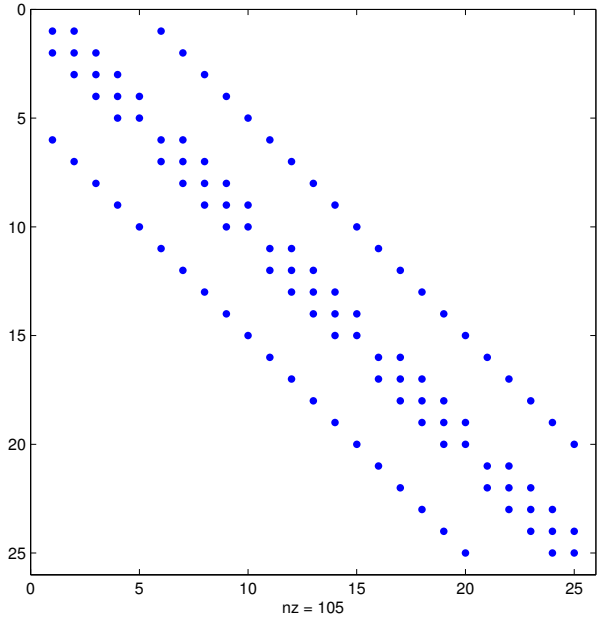
Die meisten Positionen sind mit Nullen besetzt

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

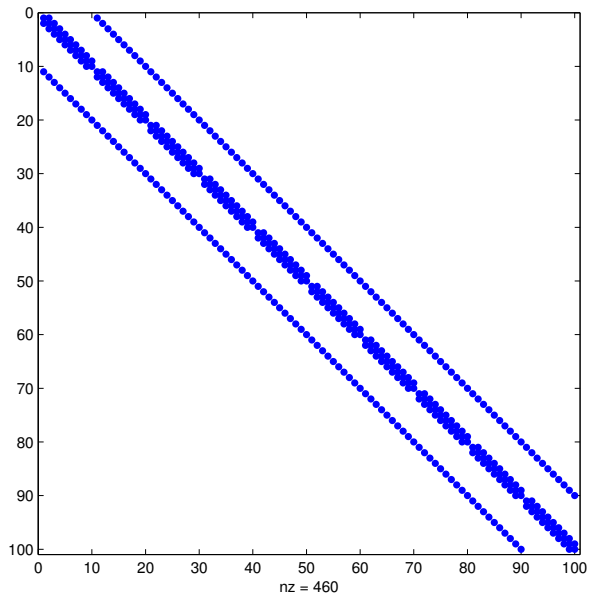
Matrix-Struktur bei größeren Gittern

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

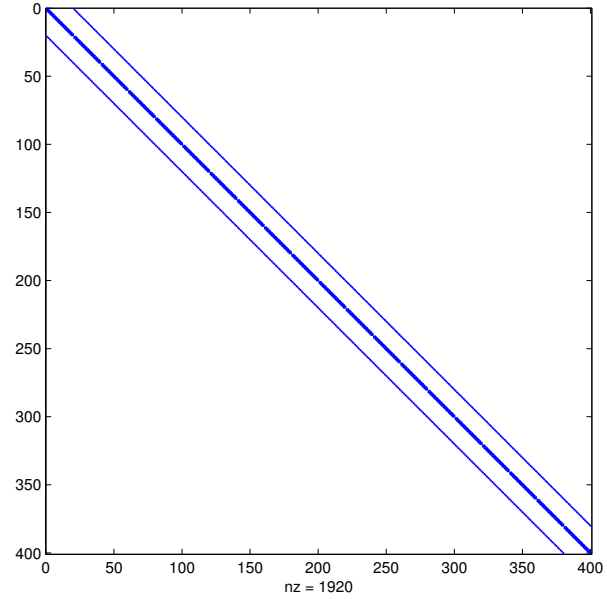
5×5



10×10



20×20



Spärlich besetzte Matrix

Die meisten Positionen sind mit Nullen besetzt

Für spärlich 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

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

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

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

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

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

Iterative Lösungsverfahren

Wir lösen die Gleichungen nach den Diagonal-Termen auf

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

Einsetzen von Startwerten rechts liefert verbesserte Werte links $\rightarrow$ 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

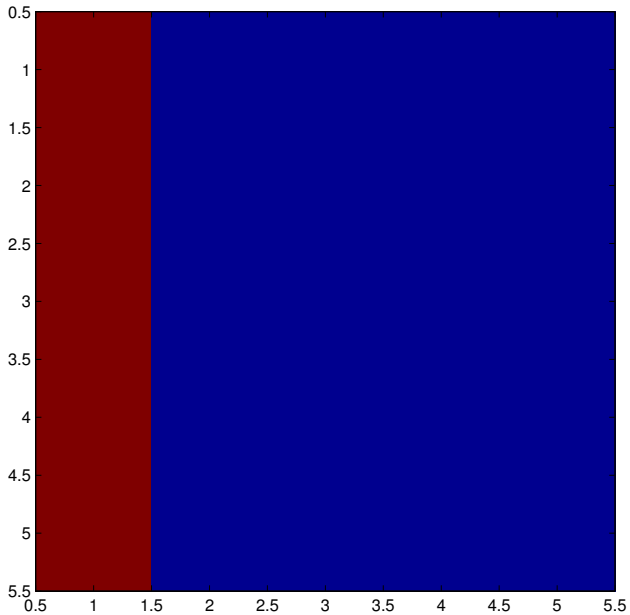
Ablauf der Iteration

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



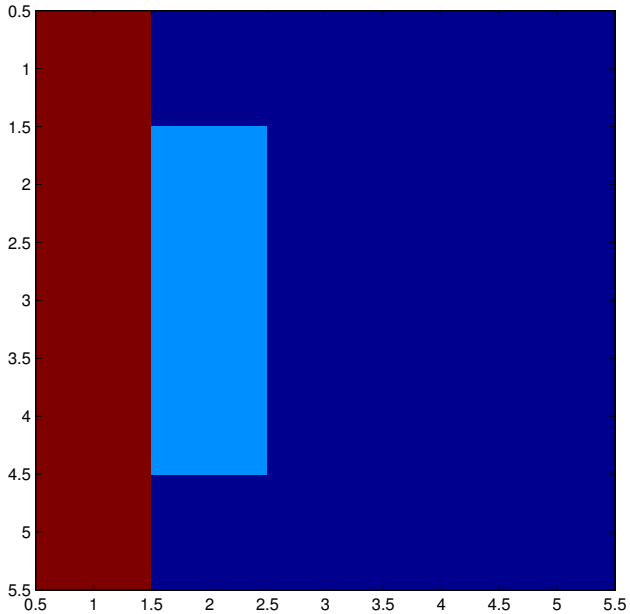
Ablauf der Iteration

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



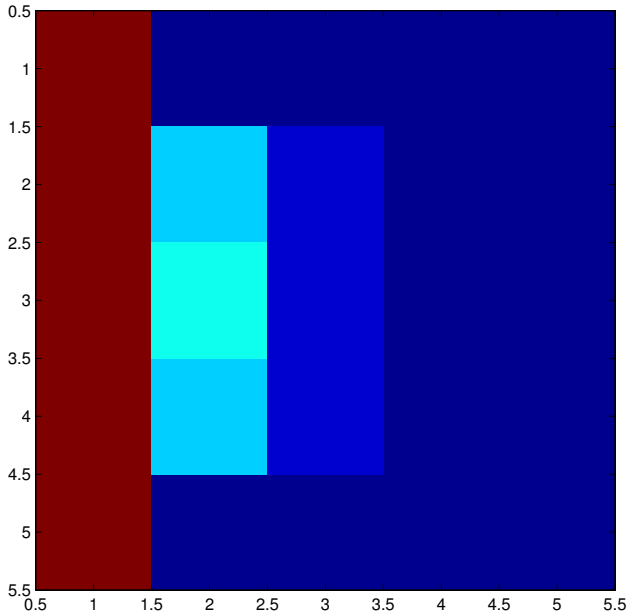
Ablauf der Iteration

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



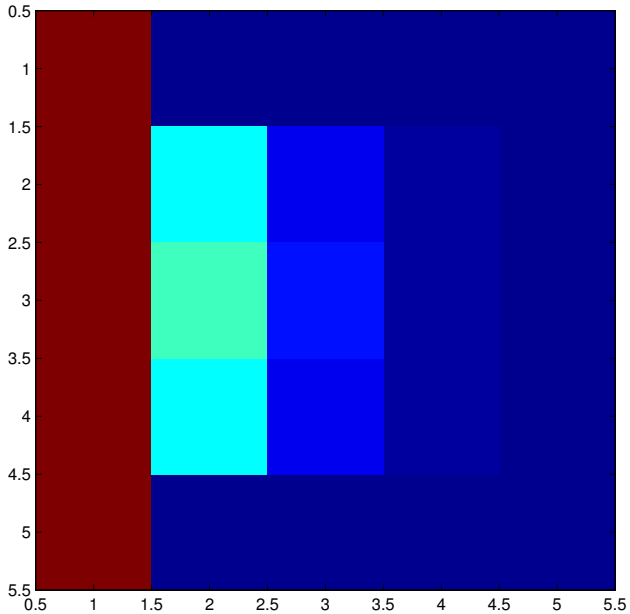
Ablauf der Iteration

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



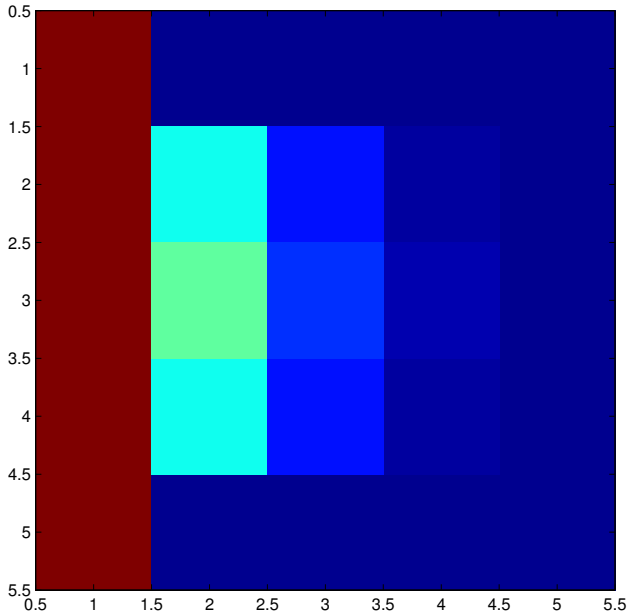
Ablauf der Iteration

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



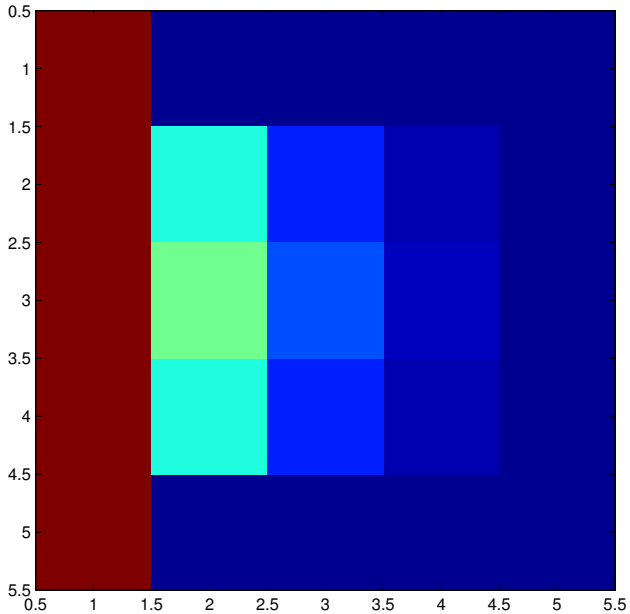
Ablauf der Iteration

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



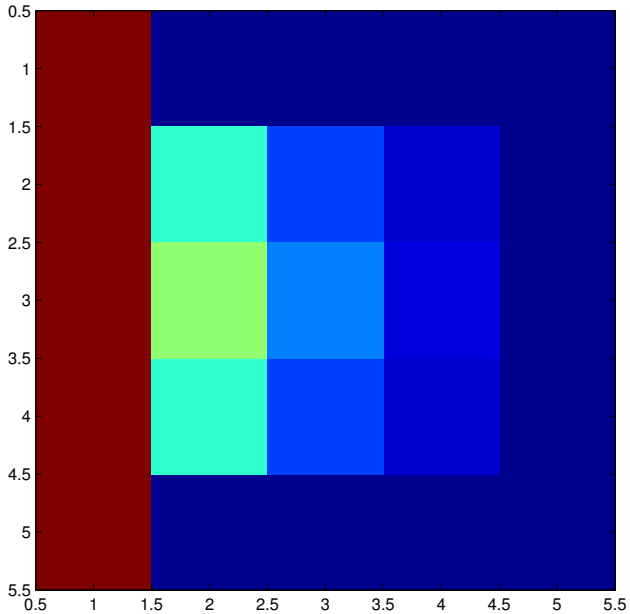
Ablauf der Iteration

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