ADI-Verfahren

Sebastian Plitzko

Seminar Numerik und wissenschaftliches Rechnen
Technische Universität Darmstadt

14. Juni 2007

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Inhaltsverzeichnis

Übersicht

Grundlegendes

- implizite Behandlung mehrdimensionaler Probleme nicht einfach
- ADI $\equiv$ alternate direction implicit
- speziell für das Lösen parabolische DGL entwickelt
- Algorithmus ist dimensionsabhängig
- Matrix K wird in eine Summe von Matrizen zerlegt, die in tridiagonalform gebracht werden können

Übersicht

ADI-Verfahren

ADI-Verfahren in 2D ($u_t = u_{xx} + u_{yy}$)

1. Halbschritt

$$u_{k,l}^{(n+\frac{1}{2})} = u_{k,l}^{(n)} + \frac{\delta t}{2}\left[\frac{u_{k+1,l}^{(n+\frac{1}{2})} + 2u_{k,l}^{(n+\frac{1}{2})} + u_{k-1,l}^{(n+\frac{1}{2})}}{h_x^2} + \frac{u_{k+1,l}^{(n)} + 2u_{k,l}^{(n)} + u_{k-1,l}^{(n)}}{h_y^2}\right]$$

2. Halbschritt

$$u_{k,l}^{(n+1)} = u_{k,l}^{(n+\frac{1}{2})} + \frac{\delta t}{2}\left[\frac{u_{k+1,l}^{(n+\frac{1}{2})} + 2u_{k,l}^{(n+\frac{1}{2})} + u_{k-1,l}^{(n+\frac{1}{2})}}{h_x^2} + \frac{u_{k+1,l}^{(n+1)} + 2u_{k,l}^{(n+1)} + u_{k-1,l}^{(n+1)}}{h_y^2}\right]$$

- zwei Teilschritte $\frac{\delta t}{2}$
- zunächst wird nur x-Richtung implizit integriert
- danach wird die y-Richtung weiter integriert
- ein voller Schritt ist vollzogen

Verfahren

Verfahrensvorschrift

$$u_{k,l}^{(n+1)} = u_{k,l}^{(n)} + \delta t \frac{u_{k+1,l}^{(n+\frac{1}{2})} + 2u_{k,l}^{(n+\frac{1}{2})} + u_{k-1,l}^{(n+\frac{1}{2})}}{h_x^2}$$
$$+ \frac{\delta t}{2} \left[\frac{u_{k+1,l}^{(n+1)} + 2u_{k,l}^{(n+1)} + u_{k-1,l}^{(n+1)}}{h_y^2} + \frac{u_{k+1,l}^{(n)} + 2u_{k,l}^{(n)} + u_{k-1,l}^{(n)}}{h_y^2} \right]$$

- Integrationen:
  1. x-Richtung Tangententrapezregel: Fehlerordnung δt^2
  2. y-Richtung Sehnentrapezregel: Fehlerordnungδt^2

  $\Longrightarrow$ Gesamtfehlerordnung δt^3

Stabilitätsanalyse für Wärmeleitungsproblem

- Courant-Friedrichs-Levy Stabilitätsanalyse mit dem Ansatz

$$u_{k,l}^{(n)} = \lambda^n e^{i(q_x k h_x + q_y l h_y)}$$

- Ansatz folgt aus Eigensystem des Laplaceoperators
- für den ersten Halbschritt

$$\lambda_1^{\frac{1}{2}} = 1 + \frac{\delta t}{2}\left[\lambda_1^{\frac{1}{2}} \frac{e^{iq_x h_x} + e^{-iq_x h_x} - 2}{h_x^2} + \frac{e^{iq_y h_y} + e^{-iq_y h_y} - 2}{h_y^2}\right]$$

$$\Longrightarrow \qquad \lambda_1^{\frac{1}{2}} = 1 + 2\delta t\left[\lambda_1^{\frac{1}{2}} \frac{\sin^2 \frac{q_x h_x}{2}}{h_x^2} + \frac{\sin^2 \frac{q_y h_y}{2}}{h_y^2}\right]$$

Stabilitätsanalyse für Wärmeleitungsproblem

$$\lambda_1^{\frac{1}{2}} = \frac{1 - 2\delta t \frac{\sin^2 \frac{q_y h_y}{2}}{h_y^2}}{1 + 2\delta t \frac{\sin^2 \frac{q_x h_x}{2}}{h_x^2}} \qquad \lambda_2^{\frac{1}{2}} = \frac{1 - 2\delta t \frac{\sin^2 \frac{q_x h_x}{2}}{h_x^2}}{1 + 2\delta t \frac{\sin^2 \frac{q_y h_y}{2}}{h_y^2}}$$

- numerische Diffusionsprobleme $\delta t > 0$
- für $\vec{q} \neq 0 \longrightarrow |\lambda_1|^{\frac{1}{2}} |\lambda_2|^{\frac{1}{2}} < 1$
- für positive Schritte unbedingt stabil
- unbedingte Stabilität ist spezielle Eigenschaft des 2D ADI-Verfahrens
- 3D ADI-Verfahren nur bedingt stabil

Übersicht

Sequentieller Algorithmus

Algorithmus

Choose u^0

$r := f - K \cdot u^0$

$\sigma := \sigma_0 := (r, r)$

$k := 0$

while $\sigma > tol^2 \cdot \sigma_0$ do

$$(H + \rho_{k+1} I) \cdot u^{k*} = (\rho_{k+1} I - V) \cdot u^k + f$$

$$(V + \rho_{k+1} I) \cdot u^{k+1} = (\rho_{k+1} I - H) \cdot u^{k*} + f$$

$$r := f - K \cdot u^{k+1}$$

$$\sigma = (r, r)$$

$$k := k + 1$$

end

- $K = H + V$ wobei sich H auf die Diskretisierung in x-Richtung und V in y-Richtung bezieht

Konvergenzbeschleunigung

- bekannt sind die Eigenwerte bzw. Eigenvektoren der Matrizen
- $K\mu_{jm} = \lambda_{jm}\mu_{jm}$
- $\mu_{jm} = \sin(j\pi y)\sin(\frac{m\pi x}{2})$
- $H\mu_{jm} = \lambda_m\mu_{jm}$
- $V\mu_{jm} = \lambda_j\mu_{jm}$
- $\lambda_{jm} = \lambda_j + \lambda_m$

Konvergenzbeschleunigung

- Zur Beschleunigung der Konvergenz wird eine Zeitentwicklungsmatrix betrachtet
- Aus dem Algerithmus:
  $T_\rho = (V + \rho I)^{-1}(H - \rho I)(H + \rho I)^{-1}(V - \rho I)$
- somit $T_\rho \mu_{jm} = \frac{(\lambda_j - \rho)(\lambda_m - \rho)}{(\lambda_j + \rho)(\lambda_m + \rho)} \mu_{jm}$
- wir erhalten ein Minimierungsproblem mit Variable ρ

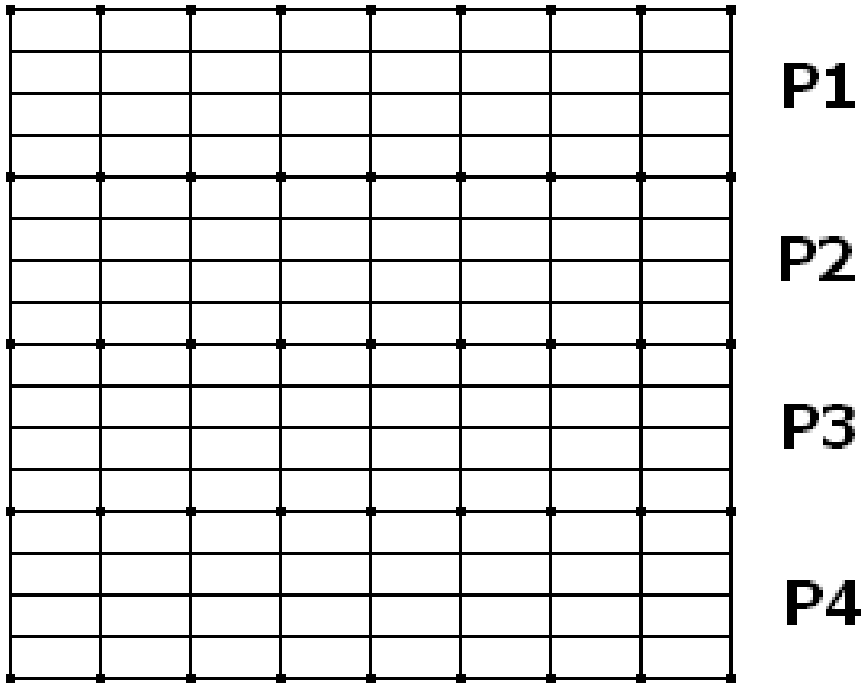
Konvergenzbeschleunigung

- $0 < \alpha < \lambda_j \ , \ \lambda_m \leq \beta$
- $c = \frac{\alpha}{\beta} \ , \ \delta = (\sqrt{2}-1)^2 \ , \ n = [log_\delta c] + 1$
- $\rho_j = \beta c^{\frac{j-1}{n-1}}$
- Für ein $(N_x + 1)x(N_y + 1)$-Gitter erhalten wir:
  $\alpha = \frac{1}{h^2}(2 - 2\cos(\pi h)) \approx \pi$
  $\beta = \frac{1}{h^2}(2 + 2\cos(\pi h)) \approx \frac{4}{h^2}$
- $\rho_j \approx \frac{4}{h^2}\delta^{j-1}$

Übersicht

Paralleler Algorithmus

- Aufteilung der Knoten zu den 4 Prozessoren in Streifen



- $K = \begin{pmatrix} K_E & K_{EI} \\ K_{IE} & K_I \end{pmatrix}$
- E für Randknoten und I für innere Knoten

K-Matrix für unseren Fall von 4 Prozessoren

$$
\begin{pmatrix}
K_{E_{01}} & 0 & 0 & 0 & 0 & K_{E_{01}I_1} & 0 & 0 & 0 \\
0 & K_{E_{12}} & 0 & 0 & 0 & K_{E_{12}I_1} & K_{E_{12}I_2} & 0 & 0 \\
0 & 0 & K_{E_{23}} & 0 & 0 & 0 & K_{E_{23}I_2} & K_{E_{23}I_3} & 0 \\
0 & 0 & 0 & K_{E_{34}} & 0 & 0 & 0 & K_{E_{34}I_3} & K_{E_{34}I_4} \\
0 & 0 & 0 & 0 & K_{E_{40}} & 0 & 0 & 0 & K_{E_{40}I_4} \\
K_{I_1E_{01}} & K_{I_1E_{12}} & 0 & 0 & 0 & K_{I_1} & 0 & 0 & 0 \\
0 & K_{I_2E_{12}} & K_{I_2E_{23}} & 0 & 0 & 0 & K_{I_2} & 0 & 0 \\
0 & 0 & K_{I_3E_{23}} & K_{I_3E_{34}} & 0 & 0 & 0 & K_{I_3} & 0 \\
0 & 0 & 0 & K_{I_4E_{34}} & K_{I_4E_{40}} & 0 & 0 & 0 & K_{I_4}
\end{pmatrix}
$$

Parallleler Algorithmus

mit Richtungsunterscheidung

- neue Aufteilung in $\mathfrak{K} = \mathfrak{V} + \mathfrak{H}$
  $\mathfrak{V} = \begin{pmatrix} \mathfrak{K}_{E,y} & \mathfrak{K}_{IE} \\ \mathfrak{K}_{EI} & \mathfrak{K}_{I,y} \end{pmatrix}$ und $\mathfrak{H} = \begin{pmatrix} \mathfrak{K}_{E,x} & 0 \\ 0 & \mathfrak{K}_{I,x} \end{pmatrix}$
- $\mathfrak{K}_{E,y}$ Diagonalmatrix
- $\mathfrak{K}_{I,y}, \mathfrak{K}_{E,x}, \mathfrak{K}_{I,x}$ parallel invertierbar
- erstes Ansatz für den Algorithmus

Erster Algorithmus(einfache Übersetzung)

Deklarationsteil

Choose u^0

$$\mathfrak{r} := \begin{pmatrix} f_E & - & \mathfrak{K}_E \cdot u_E^0 & - & \sum_{i=1}^{P} A_i^T \mathfrak{K}_{EI,i} \cdot u_{I,i}^0 \\ f_I & - & \mathfrak{K}_{IE} \cdot u_E^0 & - & \mathfrak{K}_{IE} \cdot u_E^0 \end{pmatrix}$$

$\sigma := \sigma_0 := (\mathfrak{r}, R^{-1}\mathfrak{r})$

$k := 0$

Erster Algorithmus(einfache Übersetzung)

Hauptteil

while $\sigma > tol^2 \cdot \sigma_0$ do

$$\mathfrak{T}_x := \mathfrak{H} + \rho_{k+1}\mathfrak{I}$$

$$u^{k*} := \begin{pmatrix} \mathfrak{T}_{E,x}^{-1}[f_E & + & \rho_{k+1}u_E^k & - & \mathfrak{K}_{E,y} \cdot u_E^k & - & \sum_{i=1}^{P} A_i^T \mathfrak{K}_{EI,i} \cdot u_{I,i}^k \\ \mathfrak{T}_{I,x}^{-1}[f_I & + & \rho_{k+1}u_I^k & - & \mathfrak{K}_{IE} \cdot u_E^k & - & \mathfrak{K}_{I,y} \cdot u_I^k \end{pmatrix}$$

$$\mathfrak{T}_y := \mathfrak{V} + \rho_{k+1}\mathfrak{I}$$

$$\mathfrak{g} := \begin{pmatrix} f_E & - & \rho_{k+1}u_E^{k*} & - & \mathfrak{K}_{E,x} \cdot u_E^{k*} \\ f_I & - & \rho_{k+1}u_I^{k*} & - & \mathfrak{K}_{I,x} \cdot u_I^{k*} \end{pmatrix}$$

Solve $\mathfrak{T}_y u^{k+1} = \mathfrak{g}$

$$\mathfrak{r} := \begin{pmatrix} f_E & - & \mathfrak{K}_E \cdot u_E^{k+1} & - & \sum_{i=1}^{P} A_i^T \mathfrak{K}_{EI,i} \cdot u_{I,i}^{k+1} \\ f_I & - & \mathfrak{K}_{IE} \cdot u_E^{k+1} & - & \mathfrak{K}_{IE} \cdot u_E^{k+1} \end{pmatrix}$$

$$\sigma := \sigma_0 := (\mathfrak{r}, R^{-1}\mathfrak{r})$$

$$k := k + 1$$

end

Verbesserter Algorithmus

Deklarationsteil

- Versuch eines verbesserten Algorithmus
- Reduzierung der Kommunikationsschritte

Choose u^0

$$\mathfrak{v} := \begin{pmatrix} f_E & - & \sum_{i=1}^{P} A_i^T \mathfrak{K}_{EI,i} \cdot u_{I,i}^0 \\ f_I & - & \mathfrak{K}_{IE} \cdot u_E^0 \end{pmatrix}$$

$$\mathfrak{r} := \mathfrak{v} - K \begin{pmatrix} \mathfrak{K}_E & 0 \\ 0 & \mathfrak{K}_I \end{pmatrix} \cdot u^0$$

$$\sigma := \sigma_0 := (\mathfrak{r}, R^{-1}\mathfrak{r})$$

$$k := 0$$

Verbesserter Algorithmus

while $\sigma > tol^2 \cdot \sigma_0$ do

$$\mathfrak{T}_x := \mathfrak{H} + \rho_{k+1}\mathfrak{I}$$

$$u^{k*} := \mathfrak{T}_x^{-1} \cdot \left[\mathfrak{v} + \rho_{k+1} u^k - \begin{pmatrix} \mathfrak{K}_{E,y} & 0 \\ 0 & \mathfrak{K}_{I,y} \end{pmatrix} \cdot u^k \right]$$

$$\mathfrak{T}_y := \mathfrak{V} + \rho_{k+1}\mathfrak{I}$$

$$\mathfrak{g} := f + \rho_{k+1} u^{k*} - \begin{pmatrix} \mathfrak{K}_{E,x} & 0 \\ 0 & \mathfrak{K}_{I,x} \end{pmatrix} \cdot u^{k*}$$

Solve $\mathfrak{T}_y u^{k+1} = \mathfrak{g}$

$$\mathfrak{v} := \begin{pmatrix} f_E & - & \sum_{i=1}^{P} A_i^T \mathfrak{K}_{EI,i} \cdot u_{I,i}^{k+1} \\ f_I & - & \mathfrak{K}_{IE} \cdot u_E^{k+1} \end{pmatrix}$$

$$\mathfrak{r} := \mathfrak{v} - K \begin{pmatrix} \mathfrak{K}_E & 0 \\ 0 & \mathfrak{K}_I \end{pmatrix} \cdot u^{k+1}$$

$$\sigma := \sigma_0 := (\mathfrak{r}, R^{-1}\mathfrak{r})$$

$$k := k + 1$$

end

Paralleler Algorithmus

- Einführung eines Hilfsvektors $\mathfrak{v}$
- Somit nur ein Kommunikationsschritt pro Iteration
- Inversion $\mathfrak{T}_x^{-1}$ mit einem sequentiellen Invertierer
- Einsetzen eines parallelen Algorithmus für das zu lösende GLS (parallel Gauß-Seidel)

Übersicht

Zusammenfassung

- sehr gutes Verfahren für parabolische Probleme
- in zwei Dimensionen unbedingt stabil
- nur ein Kommunikationsschritt pro Iteration
- anwendbar auf Schrödinger-Probleme aus der QM (imaginärzeitliche Diffusionsprobleme)