Georg-August-Universität
Göttingen

Numerische Strömungsmechanik

Simulation mit FTCS- und BTCS-Verfahren

Author:
Philip Marszal

Prüfer:
Prof. Dr. Andreas
Tilgner

4. August 2016

Zusammenfassung

Your abstract.

1 Einleitung

2 Problemstellung

$$\frac{\partial T(\boldsymbol{x},t)}{\partial t} + \mathrm{Pe}\ \boldsymbol{u}_0 \cdot \nabla T(\boldsymbol{x},t) = \nabla^2 T(\boldsymbol{x},t). \tag{1}$$

3 Methoden

Zur Simulation von Strömungsmechanikproblemen kommen eine Vielzahl verschiedener Verfahren in Frage. Ein konzeptionell simples Verfahren, das häufig Anwendung findet, ist die *Finite-Differenzen-Methode*. Diese wird zum Lösen des hier gestellten Problems verwendet. Andere Verfahren die zur Lösung dieser Art von Problem eingesetzt werden können sind *Finite-Volumen-Methoden*, *Finite-Elemente-Verfahren* und *spektrale Verfahren*. Bei fast allen Verfahren liegt die Diskretisierung des integrierten Systems durch ein Gitter zu Grunde, welches die theoretisch kontinuierliche Lösungsfunktion approximiert.

3.1 Finite-Differenzen-Verfahren

Um eine partielle Differentialgleichung numerisch zu integrieren, müssen die Ableitungen ersetzt werden durch diskret berechenbare Terme. Ein naiver Ansatz ist die Näherung der Ableitung $T'(x)$ einer Funktion an einem Punkt durch den Differenzenquotienten.

$$T'(x) = \lim_{h \to 0} \frac{T(x+h) - T(x)}{h}. \tag{2}$$

Bei der Verwendung eines Gitters entspricht hier h dem Abstand der zwei benachbarten Gitterpunkte. Der Differenzenquotient in Gleichung (2) entspricht dem Vorwärtsdifferenzenquotienten erster Ordnung in h.

Eine Approximation höherer Ordnung erhält man z.B. durch den zentralen Differenzenquotienten

$$T'(x) = \frac{T(x+h) - T(x-h)}{2h} + O(h^2), \tag{3}$$

oder für die zweite Ableitung als Vorwärtsdifferenzenquotient von $T'(x)$:

$$T''(x) = \frac{T(x+h) - 2T(x) + T(x-h)}{h^2} + O(h^2). \tag{4}$$

Durch Kombinationen unterschiedlicher Differenzenquotienten können Näherungen höherer Ordnung erhalten werden.

3.2 FTCS-Variante

Verwendet man zur Diskretisierung der PDGL eine Vorwärtsdifferenz für die Zeitableitung und eine zentrale Differenz für die Ortsableitungen, so spricht man von einem *FTCS*-Verfahren (**F**orwards **T**ime **C**entral **S**pace). Diskretisiert man Gleichung (1) auf diese Weise so erhält man die Differenzengleichung:

$$\begin{aligned} \frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t} + \mathrm{Pe}\ u_{i,j} \frac{T_{i+1,j}^{n} - T_{i-1,j}^{n}}{\Delta x} + \mathrm{Pe}\ v_{i,j} \frac{T_{i,j+1}^{n} - T_{i,j-1}^{n}}{\Delta y} \\ = \frac{T_{i+1,j}^{n} - 2T_{i,j}^{n} + T_{i-1,j}^{n}}{\Delta x^2} + \frac{T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}}{\Delta y^2} \end{aligned} \tag{5}$$

Diese Gleichung kann nun numerisch integriert werden, indem $T_{i,j}^{n+1}$ für jeden Gitterpunkt berechnet wird. Allerdings ist darauf zu achten ob man $T_{i,j}^{n+1}$ am Rand berechnet, denn hier müssen die Randbedingungen in die Gleichung eingebaut werden.
Die Dirichlet Randbedingungen lassen sich unproblematisch für $T_{i,0}^{n}$ und T_{i,N_y+1}^{n} einsetzen. Diese Gitterpunkte verändern sich nicht im Laufe der Zeit, müssen also nicht integriert werden. Anders sieht dies allerdings für die Neumann Randbedingungen aus. Die Gitterpunkte mit $i = 0$ und $i = N_x + 1$ verändern sich sehr wohl im Laufe der Zeit.
Um zu vermeiden, dass man auf Gitterpunkte außerhalb des Gitters zugreift, muss man eine Vorwärtsdifferenz verwenden. Eine Näherung zweiter Ordnung für die Ableitung $\frac{\partial T(\boldsymbol{x},t)}{\partial x}$ bei $x = 0$ und $x = 1$ lässt sich mit Gleichung (6)

berechnen:

$$\begin{aligned} T_{0,j}^{n+1} &= -\frac{1}{3}\left(T_{2,j}^{n+1} - 4T_{1,j}^{n+1}\right), \\ T_{N_x+1,j}^{n+1} &= -\frac{1}{3}\left(T_{N_x-1,j}^{n+1} - 4T_{N_x,j}^{n+1}\right). \end{aligned} \tag{6}$$

Mit Gleichung (5) und Gleichung (6) ist man nun in der Lage das das Problem zu simulieren.

3.3 Die BTCS-Variante

Im Gegensatz zu dem FTCS-Verfahren, welches ein explizites Verfahren zu Lösung von partiellen Differentialgleichungen darstellt, handelt es sich bei der *BTCS*-Methode (**B**ackwards **T**ime **C**entral **S**pace) um ein implizites Verfahren. Implizite Verfahren erfordern die Lösung eines Gleichungssystems ehe ein Zeitschritt durchgeführt werden kann.
Im BTCS-Verfahren wird die Zeitableitung durch eine Rückwärtsdifferenz diskretisiert. Dadurch entsteht in dem vorliegenden Problem zunächst ein lineares Gleichungssystem aus $(N_x + 1) \cdot (N_y + 1)$ Gleichungen für die $T_{i,j}^n$.

$$\begin{aligned} &\frac{T_{i,j}^n - T_{i,j}^{n-1}}{\Delta t} + \mathrm{Pe}\ u_{i,j}\frac{T_{i+1,j}^n - T_{i-1,j}^n}{\Delta x} + \mathrm{Pe}\ v_{i,j}\frac{T_{i,j+1}^n - T_{i,j-1}^n}{\Delta y} \\ &\qquad = \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{\Delta x^2} + \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{\Delta y^2} \end{aligned} \tag{7}$$

Führt man die Dirichlet-Randbedingungen ein so reduziert sich das Gleichungssystem auf $N_x \cdot (N_y - 1)$ Gleichungen. Um die Neumann-Randbedingungen mit dem Gleichungssystem zu vereinigen betrachtet man den zentralen Differenzenquotienten an $x = 0$ und $x = 1$:

$$\left.\frac{\partial T}{\partial x}\right|_{x=0} = \frac{T_{1,j}^n - T_{-1,j}^n}{\Delta x} = 0,$$

und einem analogen Quotienten für $x = 1$. Mithilfe dieser Gleichungen lassen sich die Terme $T_{-1,j}^n$ und $T_{N_x+2,j}^n$ aus dem Gleichungssystem eliminieren und somit die Neumann-Randbedingungen in die Gleichungen einbauen.
Um dieses System zu lösen bietet es sich an das Gitter $T_{i,j}^n$ als $N_x \cdot (N_y - 1)$ dimensionalen Vektor umzuschreiben. Die Lösung des Systems entspricht

dann einer Invertierung der Matrix M, die sich aus den Gleichungen als Gleichung (8) ergibt.

$$M = \begin{pmatrix} (1.+4D) & (v_{0,1}A-D) & 0 & \dots & -2D & 0 & \dots & \dots & 0 \\ -(v_{0,2}A+D) & (1.+4D) & (v_{0,2}A-D) & 0 & \dots & -2D & 0 & \dots & 0 \\ 0 & -(v_{0,3}A+D) & (1.+4D) & (v_{0,3}A-D) & 0 & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ \dots & \dots & -(u_{i,j}A+D) & \dots & -(v_{i,j}A+D) & (1.+4D) & (v_{i,j}A-D) & \dots \quad (u_{i,j}A-D) & \dots 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & & \dots & & & -2D & \dots -(v_{N_x+1,N_y}A+D) & (1.+4D) \end{pmatrix} \tag{8}$$

Hierbei wurde der Vektor $\boldsymbol{T}^n$ so gewählt, dass die Spalten des Gitters aneinander gehängt werden, sprich die Komponenten des Vektors laufen zunächst im j-Index durch : $\boldsymbol{T}^n = \left(T^n_{0,1}, T^n_{0,2}, ..., T^n_{0,N_y}, T^n_{1,1}, ..., T^n_{N_x+1,N_y}\right)$. Die Terme $A = \frac{Pe\ \Delta t}{\Delta x}$ und $D = \frac{\Delta t}{\Delta x^2}$ enstprechen den konstanten Vorfaktoren im Advektions- und Diffusionsterm.

3.4 Der SOR-Algorithmus

Die Matrix M hat für vernünftige Gittergrößen und Zeitschritte Δt die Eigenschaft, dass sie diagonal dominant ist. Also die Summe der Beträge der nicht-diagonalen Einträge in jeder Spalte nicht den Betrag des diagonalen Eintrags dieser Spalte übersteigt.
Für Matrizen mit dieser Eigenschaft liefert das sogenannte Gauß-Seidel-Verfahren eine approximative Lösung des Gleichungssystems $M\boldsymbol{x} = \boldsymbol{b}$, die garantiert gegen die tatsächliche Lösung konvergiert. Zunächst wird M in eine Summe aus untere und oberer Dreiecksmatrix, L und U, und der Diagonale D zerlegt. Die Iterationsvorschrift des Gauß-Seidel-Verfahrens sieht nun wie folgt aus:

$$\boldsymbol{x}_{n+1} = (L + D)^{-1}(\boldsymbol{b} - U\boldsymbol{x}_n) \tag{9}$$

Der Vorteil dieser Methode liegt darin, dass das Produkt von $(L + D)^{-1}$ mit einem Vektor einfach über Vorwärtssubstitution berechnet werden kann, dank der Dreiecksform von $L + D$.
Eine Weiterentwicklung des Gauß-Seidel-Algorithmus ist die *successive over-relaxation*-Methode (SOR). Dazu wird Gleichung (9) zunächst durch den Korrekturvektor $\boldsymbol{r}_n = M\boldsymbol{x}_n - \boldsymbol{b}$ ausgedrückt. Über den Relaxationsparameter ω kann nun die Konvergenz des Gauß-Seidel-Verfahrens erheblich beschleunigt werden. Die Iterationsvorschrift des SOR-Algorithmus lautet:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \omega(L + D)^{-1}(\boldsymbol{r}_n) \tag{10}$$

Für die Wärmetransportgleichung liegt der optimale Relaxationsparameter zwischen 0 und 2.

4 Programmstruktur

Zur Bearbeitung des Problems wurden die beiden Methoden, FTCS und BTCS, in zwei getrennten Programmen implementiert. Als interface mit den Programmen wurde eine Konfigurationsdatei erstellt, die es erlaubt Parameter der Algorithmen zu ändern, ohne das Programm neu komplilieren zu müssen. Die Interaktion mit der Konfigurationsdatei wurde in der Headerdatei *interface.h* implementiert, welche zudem sämtliche Funktionen zur Ausgabe der Ergebnisse beinhaltet.
Die Kernfunktionen der Integration befinden sich in der *integration.h*-Datei. Diese sind die Initialisierungen der Temperaturfelder, des Quellfeldes und der Geschwindigkeitsfelder. Außerdem sind einzelne Zeitschritte in den Funktionen *FTCS()* und *FTCS_with_Q()* zusammengefasst. Diese nehmen als Input das Temperaturfeld (call-by-reference), die Geschwindigkeitsfelder und das Quellfeld (call-by-value). Die Entscheidung, welche Gleichung integriert werden soll, wird in der Konfigurationsdatei getroffen.
Die BTCS-Methode benötigt weitere Funktionen, die von dem betrachteten Problem abhängen. Dies sind Funktionen zum transformieren des Feldes in einen Vektor und zurück (*reshape_vector()* und *shape_back()*), die Einführung der Dirichlet-Randbedingungen in den Vektor (*impose_dirichlet()*) und die Initialisierung der Matrix M (*BCTS_implicit_Matrix()*).
Im BTCS Programm werden zudem algebraische Operationen benötigt, die in der Datei *operations.h* definiert sind. Dabei handelt es sich um standard Vektor-Matrix Operationen, aber auch die Triangularisierung der M-Matrix (*triangularize()*) und das von der Problemstellung unabhängige SOR-Verfahren (*SOR()*).
Innerhalb der Schleife über die Zeit wurde zudem eine Abfrage eingeführt, die zu in der Konfigurationsdatei definierten Zeitpunkten das aktuelle Temperaturfeld ausgibt.
Die Darstellung der Ergebnisse wurde mittels *Python-Matplotlib* durchgeführt.

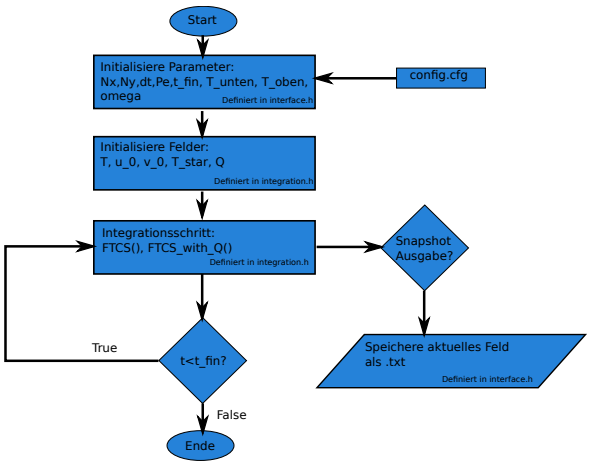

Abbildung 1: Flowchart der Implementierung der FTCS-Methode.

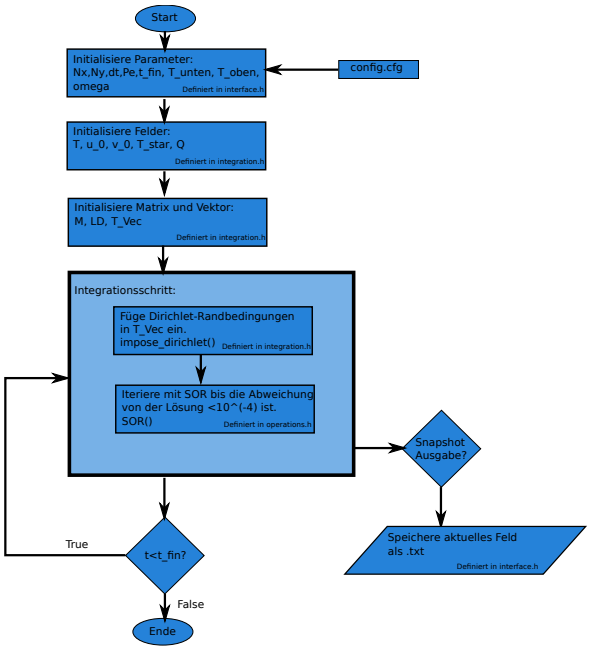

Abbildung 2: Flowchart der Implementierung der BTCS-Methode.

5 Ergebnisse

Die anfängliche Betrachtung des Temperaturfeldes im Verlauf der Zeit, wie es durch die FTCS-Methode berechnet wird, liefert die Ergebnisse in Abb. 3. Hiefür wurde eine Peclet-Zahl von $Pe = 10$ und ein Gitter von $N_x = N_y = 30$ angenommen. Die Schrittweite in der Zeit beträgt $dt = 0.0002$.
Allerdings muss noch überprüft werden, wie genau das Ergebnis ist, und, wie die Schritt

6 Vergleich FTCS- und BTCS

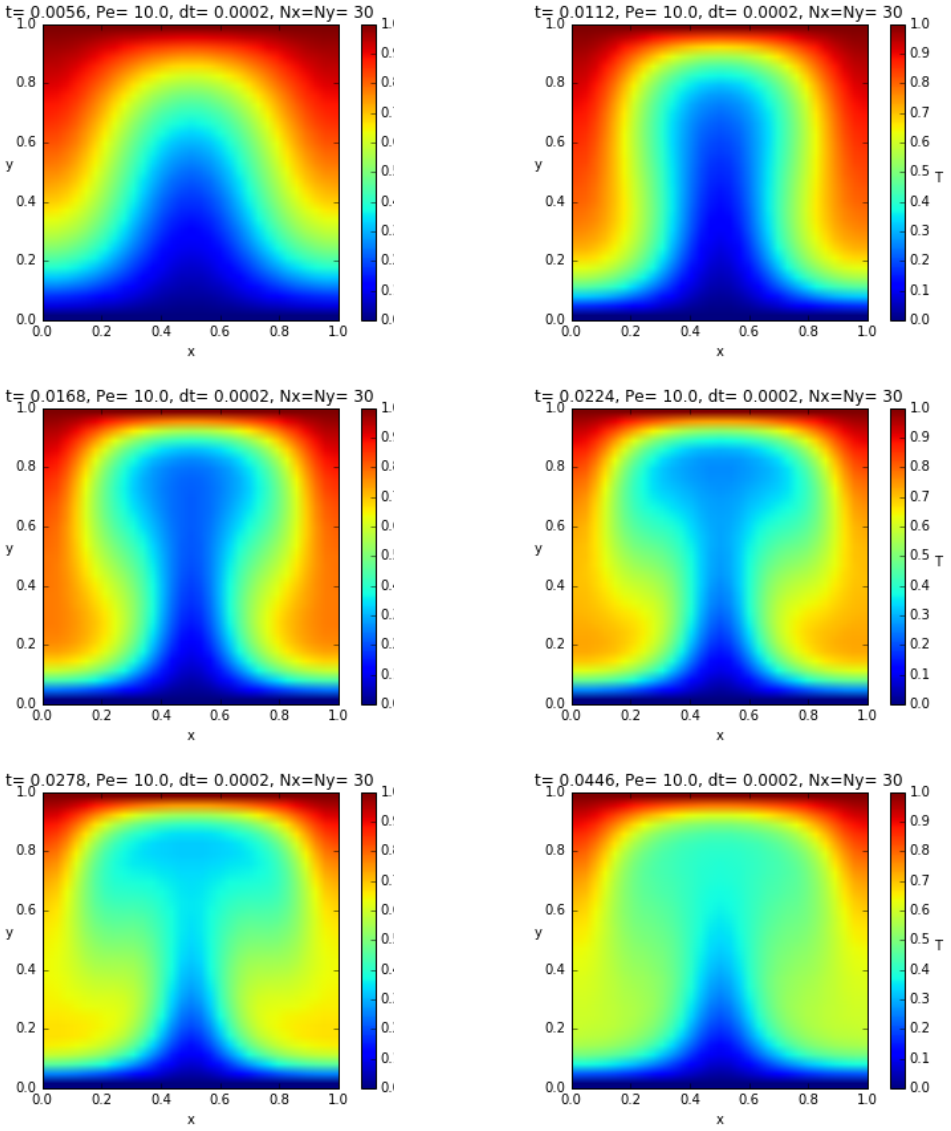

Abbildung 3: Temperaturfeld zu verschiedenen Zeiten. Berechnet mit dem FTCS-Verfahren.