Georg-August-Universität
Göttingen

Numerische Strömungsmechanik

Simulation mit FTCS- und BTCS-Verfahren

Author:
Philip Marszal
Matrikelnummer:
21212455

Prüfer:
Prof. Dr. Andreas
Tilgner

5. August 2016

Inhaltsverzeichnis

Zusammenfassung

Im Zuge dieses Projektes wird das Verhalten der BTCS und FTCS am Beispiel der Integration der Konvektions-Diffusions-Gleichung behandelt. Es wurden zwei C++ Anwendungen zu Integration der Gleichung erstellt, die benutzt wurden um Fehler, Stabilität und Laufzeit der beiden Verfahren zu untersuchen. Der Fehler des FTCS Verfahren nimmt mit feinerer Aufteilung des Gitters ab, führt aber bei gleichbleibender Zeitdiskretisierung bei hohen Péclet-Zahlen zu Instabilität. Die numerische Suche nach der Instabilitätsgrenze stößt bei begrenzten Integrationszeiträumen an ihre Grenze.

Für das in der BTCS-Methode verwendete SOR-Verfahren wurden optimale Werte des Relaxationsparameters bestimmt. Für einen Zeitschritt von 100 und einer Péclet-Zahl von 10, wurde dieser bei einem Wert von etwa 1.7 gefunden.

Der Vergleich zwischen BTCS und FTCS ergibt wenig Abweichung in Genauigkeit. Das BTCS-Verfahren ist jedoch mit einem größeren Programmieraufwand verbunden und braucht zur Integration des gleichen Zeitraums, bei kleinen Zeiten, ca. 60 mal so viel Zeit. Für große Zeiträume gleichen sich die Laufzeiten der Algorithmen an. Es wurde festgestellt, dass für Zeiten > 100 der BTCS-Algorithmus das FTCS-Verfahren an überholt.

1 Einleitung

Obwohl die Navier-Stokes-Gleichungen seit der ersten Hälfte des 19ten Jahrhunderts bekannt sind, gibt es heute noch immer keine allgemeine analytische Lösung dieser. Die Hartnäckigkeit dieses Problem hat das *Clay Mathematics Institute* sogar dazu gebracht ein 1 Millionen $ Ausschreiben auf die Lösung der Gleichungen auszusetzen.
Für die Untersuchung strömungsdynamischer Systeme, ist man daher nahezu vollkommen auf numerische Methoden angewiesen.
Ein Beispiel für die Integration der Konvektions-Diffusions-Gleichung wird hier mit den beiden finiten Differenzenmethoden FTCS und BTCS diskutiert.

2 Problemstellung

In diesem Projekt geht es um die Untersuchung der Konvektions-Diffusions-Gleichung(KDG) mit einem unterliegenden Geschwindigkeitsfeld. Interpretieren lässt sich diese zum Beispiel als ein Temperaturfeld in einem Fluid als Medium. Die zeitliche Veränderung dieses Feldes wird dann durch zwei verschiedene Wärmetransportarten verursacht. Zum einen breitet sich die Temperatur durch Diffusion (Brownsche-Teilchenbewegung) aus. In der KDG tritt diese Ausbreitungsart als zweite Ableitung im Ort auf.
Die zweite Transportart, die in einem Fluid auftritt, ist die Konvektion. Wärme wird die sich bewegende Flüssigkeit transportiert. Dies entspricht einem Skalarprodukt von Gradienten mit dem Geschwindigkeitsfeld in der KDG.
Gleichung (1) ist die entdimensionalisierte Form der KDG.

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

Das konkrete Problem wird durch die Randbedingungen und das Geschwindigkeitsfeld charakterisiert. Das System ist zunächst auf eine quadratische Box, $0 \leq x \leq 1$ und $0 \leq y \leq 1$, eingeschränkt. An den Rändern der Box werden folgende Randbedingungen angenommen:

$$\begin{aligned} T(t,x,y=0) = 0, & T(t,x,y=1) = 1, \\ \left.\frac{\partial T}{\partial x}\right|_{x=0} &= \left.\frac{\partial T}{\partial x}\right|_{x=1} = 0. \end{aligned} \tag{2}$$

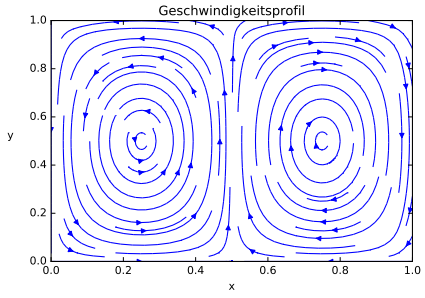
Das Geschwindigkeitsfeld ist durch Gleichung (3) gegeben.

$$\boldsymbol{u}(x,y) = \begin{pmatrix} u(x,y) \\ v(x,y) \end{pmatrix} = \begin{pmatrix} \pi \sin 2\pi x \cos \pi y \\ -2\pi \cos 2\pi x \sin \pi y \end{pmatrix} \tag{3}$$

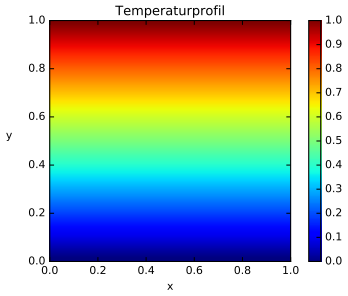
Eine kurze Rechnung ergibt, dass das Feld divergenzfrei ist. Abb. 1a zeigt das Geschwindigkeitsfeld in der Box.
Die Anfangsbedingung an das Temperaturfeld lautet

$$T(0,x,y) = y$$

und ist in Abb. 1b dargestellt.



(a) Geschwindigkeitsfeld in der Box.



(b) Temperaturverteilung zur Zeit $t = 0$.

Abbildung 1: Geschwindigkeitsfeld und Anfangsbedingung des Systems

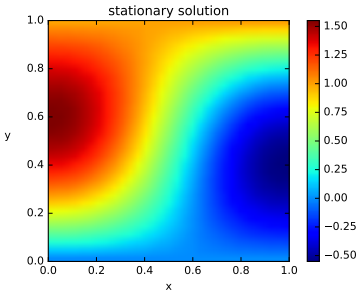
Zur Überprüfung der hier erstellten Verfahren ist es notwendig eine Referenzlösung zur Verfügung zu haben. Dazu gibt es die Möglichkeit eine inhomogene Temperaturverteilung zu wählen und die KDG mittels eines Quellterms Q so anzupassen, dass die gewählte Lösung eine stationäre Lösung der modifizierten Gleichung ist.
Als Wunschlösung wird das Temperaturfeld $T^* = \cos \pi x \sin \pi y + y$ gewählt. Die neue Differentialgleichung des Problems lautet nun:

$$\frac{\partial T(\boldsymbol{x}, t)}{\partial t} + \mathrm{Pe}\ \boldsymbol{u}_0 \cdot \nabla T(\boldsymbol{x}, t) = \nabla^2 T(\boldsymbol{x}, t) + Q(x, y). \tag{4}$$

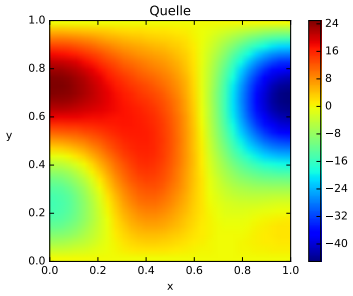
Durch Einsetzen von T^* in diese Gleichung lässt sich der Quellterm Q bestimmen als Gleichung (5)

$$\begin{aligned} Q = &- Pe\ \pi^2 \sin 2\pi x \cos \pi y \sin \pi x \sin \pi y \\ &- 2Pe\ \pi^2 \cos 2\pi x \cos \pi y \cos \pi x \sin \pi y \\ &- 2Pe\ \pi \cos 2\pi x \sin \pi y \\ &+ 2\pi^2 \cos \pi x \sin \pi y. \end{aligned} \tag{5}$$

Abb. 2 zeigt die stationäre Lösung T^* und die Quelle Q.



(a) Stationäre Lösung T^*



(b) Quellterm Q

Abbildung 2: Stationäre Lösung T^* und Quellterm Q aus Gleichung (4).

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.

$$T'(x) = \lim_{h \to 0} \frac{T(x+h) - T(x)}{h}. \tag{6}$$

Bei der Verwendung eines Gitters entspricht hier h dem Abstand der zwei benachbarten Gitterpunkte. Der Differenzenquotient in Gleichung (6) entspricht dem Vorwärtsdifferenzenquotienten erster Ordnung in h.
Eine Approximation höherer Ordnung erhält man z.B. durch den zentralen

Differenzenquotienten

$$T'(x) = \frac{T(x+h) - T(x-h)}{2h} + O(h^2), \tag{7}$$

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

$$T''(x) = \frac{T(x+h) - 2T(x) + T(x-h)}{h^2} + O(h^2). \tag{8}$$

Durch Kombinationen unterschiedlicher Differenzenquotienten können Näherungen höherer Ordnung erhalten werden [3].

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:

$$\begin{aligned} \frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t} + \mathrm{Pe}\ u_{i,j} \frac{T_{i+1,j}^{n} - T_{i-1,j}^{n}}{\Delta x} + \mathrm{Pe}\ v_{i,j} \frac{T_{i,j+1}^{n} - T_{i,j-1}^{n}}{\Delta y} \\ = \frac{T_{i+1,j}^{n} - 2T_{i,j}^{n} + T_{i-1,j}^{n}}{\Delta x^2} + \frac{T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}}{\Delta y^2} \end{aligned} \tag{9}$$

Hierbei entspricht Δt dem Zeitschritt, $\Delta x = 1./N_x$ und $\Delta y = 1./N_y$ entsprechen den jeweiligen Gitterabständen in x- und y-Richtung. Um den Rand des Systems zu betrachten werden $(N_x + 1) \cdot (N_y + 1)$ Gitterpunkte betrachtet. Gleichung (9) 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 (10) berechnen [3]:

$$\begin{aligned} T_{0,j}^{n+1} &= -\frac{1}{3}\left(T_{2,j}^{n+1} - 4T_{1,j}^{n+1}\right), \\ T_{N_x+2,j}^{n+1} &= -\frac{1}{3}\left(T_{N_x,j}^{n+1} - 4T_{N_x+1,j}^{n+1}\right). \end{aligned} \tag{10}$$

Mit Gleichung (9) und Gleichung (10) 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, garantieren im Gegenzug dafür aber Konvergenz.
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 $T_{i,j}^n$.

$$\begin{aligned} &\frac{T_{i,j}^n - T_{i,j}^{n-1}}{\Delta t} + \mathrm{Pe}\ u_{i,j}\frac{T_{i+1,j}^n - T_{i-1,j}^n}{\Delta x} + \mathrm{Pe}\ v_{i,j}\frac{T_{i,j+1}^n - T_{i,j-1}^n}{\Delta y} \\ &\qquad = \frac{T_{i+1,j}^n - 2T_{i,j}^n + T_{i-1,j}^n}{\Delta x^2} + \frac{T_{i,j+1}^n - 2T_{i,j}^n + T_{i,j-1}^n}{\Delta y^2} \end{aligned} \tag{11}$$

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 $T^n_{i,j}$ 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 (12) ergibt.

$$M = \begin{pmatrix} (1.+4D) & (v_{0,1}A-D) & 0 & \dots & -2D & 0 & \dots & \dots & 0 \\ -(v_{0,2}A+D) & (1.+4D) & (v_{0,2}A-D) & 0 & \dots & -2D & 0 & \dots & 0 \\ 0 & -(v_{0,3}A+D) & (1.+4D) & (v_{0,3}A-D) & 0 & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ \dots & \dots & -(u_{i,j}A+D) & \dots & -(v_{i,j}A+D) & (1.+4D) & (v_{i,j}A-D) & \dots & (u_{i,j}A-D) \quad \dots 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & & \dots & & & -2D & \dots & -(v_{N_x+1,N_y}A+D) \quad (1.+4D) \end{pmatrix} \tag{12}$$

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 [3]:

$$\boldsymbol{x}_{n+1} = (L + D)^{-1}(\boldsymbol{b} - U\boldsymbol{x}_n) \tag{13}$$

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 (13) 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 [1]:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \omega(L + D)^{-1}(\boldsymbol{r}_n) \tag{14}$$

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

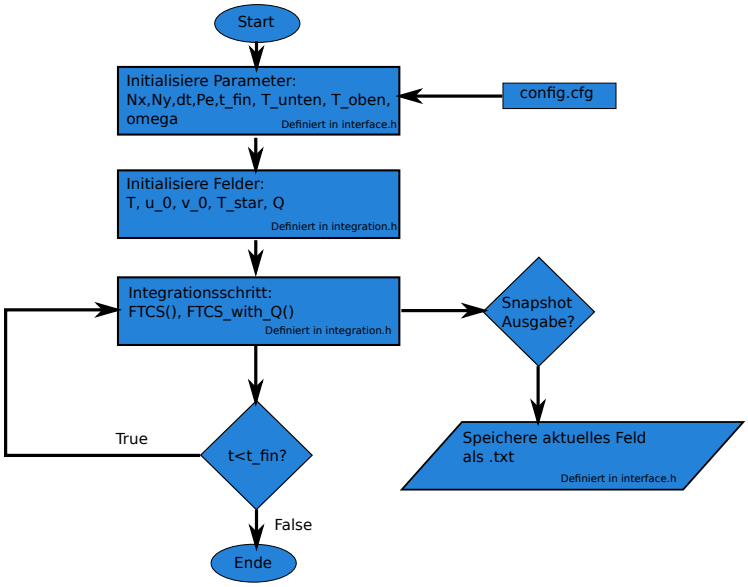


Abbildung 3: Flowchart der Implementierung der FTCS-Methode.

4 Programmstruktur

Zur Bearbeitung des Problems wurden die beiden Methoden, FTCS und BTCS, in zwei getrennten Programmen implementiert. Als Schnittstelle mit den Programmen wurde eine Konfigurationsdatei erstellt, die es erlaubt Parameter der Algorithmen zu ändern, ohne das Programm neu kompilieren 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.

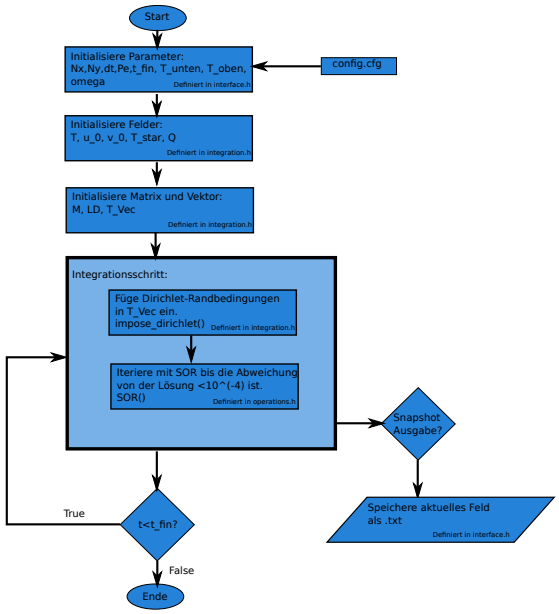


Abbildung 4: Flowchart der Implementierung der BTCS-Methode.

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.

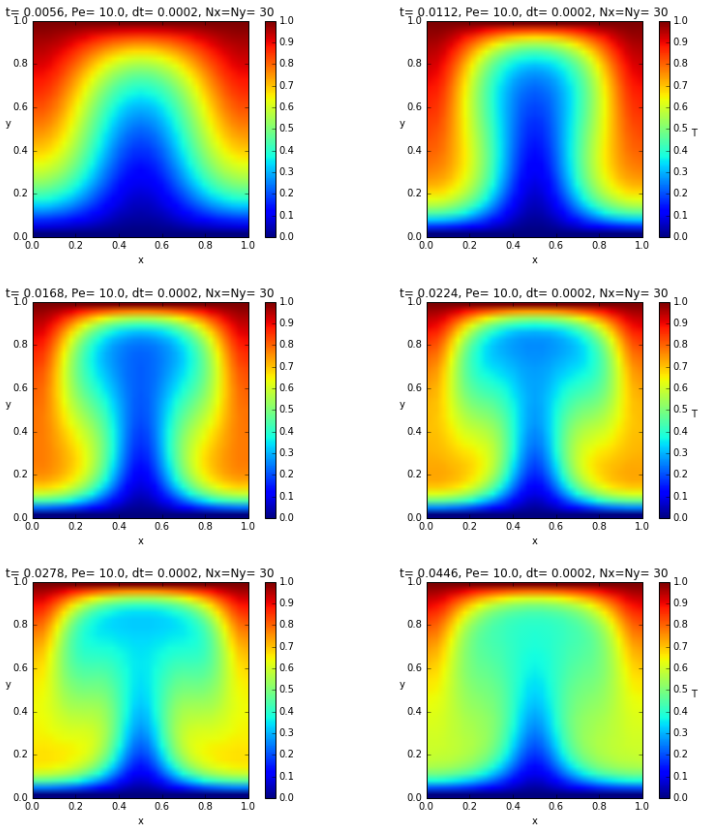


Abbildung 5: Temperaturfeld zu verschiedenen Zeiten. Berechnet mit dem FTCS-Verfahren.

5 Ergebnisse

Die anfängliche Betrachtung des Temperaturfeldes im Verlauf der Zeit, wie es durch die FTCS-Methode berechnet wird, liefert die Ergebnisse in Abb. 5. Hiefür wurde eine Péclet-Zahl von $Pe = 10$ und ein Gitter von $N_x = N_y = 30$ angenommen. Die Schrittweite in der Zeit beträgt $\Delta t = 0.0002$.

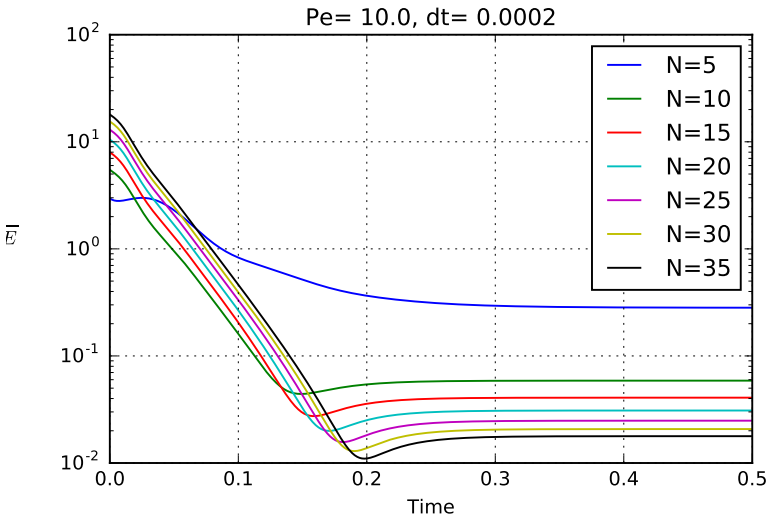


Abbildung 6: Sum of squared errors (SSE) der numerischen Integration im Vergleich zu dem Referenzfeld T^*.

5.1 Fehler der FTCS-Lösung

Allerdings muss noch überprüft werden, wie genau das Ergebnis ist, und, wie die Gittergröße sich auf die Konvergenz auswirkt. Dazu wird Gleichung (4) benutzt. Da die stabile stationäre Lösung dieser Gleichung bekannt ist verfügt man über ein Referenzfeld, gegen das die FTCS-Näherung konvergieren müsste. In Abb. 6 ist die quadratische Abweichung von T^* als Funktion der Zeit für verschiedene Gittergrößen dargestellt. Es ist sichtbar, dass die numerische Lösung gegen einen konstanten Fehler konvergiert. Eine Verfeinerung des Gitters von $N = 5$ auf 10 führt zu einer drastischen Verbesserung der Genauigkeit der Lösung, da der Fehler um fast eine Größenordnung sinkt. Bei weiterer Verfeinerung des Gitters kann ein weiterer Zuwachs an Genauigkeit festgestellt werden.
Abb. 7 zeigt die Fehler zur Zeit $t = 0.5$. Sichtbar wird hier, dass der Zuwachs an Genauigkeit mit steigender Gittergröße abnimmt. Für die hier am häufigsten verwendete Gittergröße $N = 30$ beträgt die SSE nach einer Zeit von 0.5 nur ca. 0.02. Eine SSE von nur 0.02 ist durchaus akzeptabel, vor allem da dies einer noch geringeren Abweichung der einzelnen Gitterpunkte von T^* entspricht, da die SSE nicht auf die Anzahl der Gitterpunkte normiert ist. Damit lässt sich sagen, dass das FTCS-Verfahren eine physikalisch

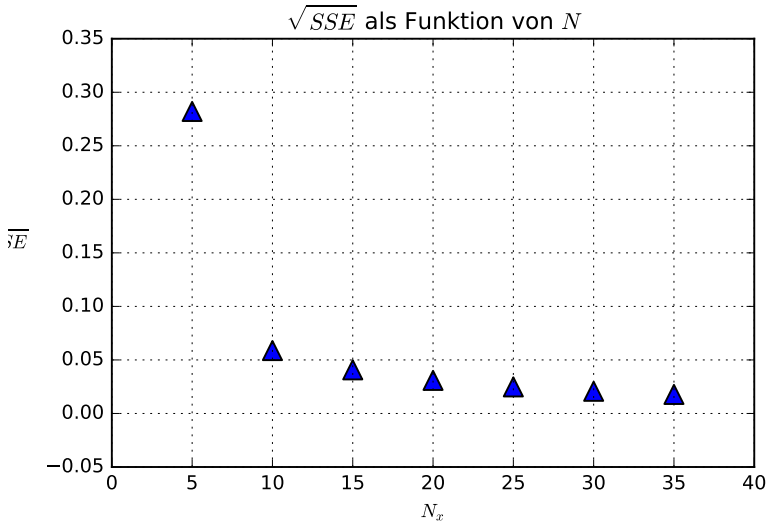


Abbildung 7: Sum of squared errors (SSE) der numerischen Integration im Vergleich zu dem Referenzfeld T^*.

sinnvolle Lösung des Problems liefert.

5.2 Vergleich unterschiedlicher Péclet-Zahlen

Nun da bekannt ist, dass die numerische Lösung des Verfahrens eine korrekte Lösung des Problems liefert, kann das ursprüngliche Problem aus Gleichung (1) genauer betrachtet werden. In Abb. 8 sind die Temperaturfelder für die Péclet-Zahlen $Pe = 2$ und $Pe = 10$ zu den Zeiten $t = 0.005, 0.05$ und 0.5 visualisiert. Die Gittergröße beträgt wieder $N = 30$ und die Zeitschrittweite $\Delta t = 0.0002$.
Aus dem Vergleich wird deutlich welchen Einfluss die Péclet-Zahl auf die Dynamik des Systems hat. Eine größere Péclet-Zahl führt zu einer schnelleren advektiven Veränderung des Temperaturfeldes.

5.3 Zulässige Zeitschritte

Bis jetzt wurde immer der Zeitschritt $\Delta t = 0.0002$ verwendet. Aus den obigen Simulationen ging eindeutig hervor, dass für die betrachteten Gittergrößen und Péclet-Zahlen dieser Zeitschritt noch ein stabiles Verfahren liefert. Jetzt

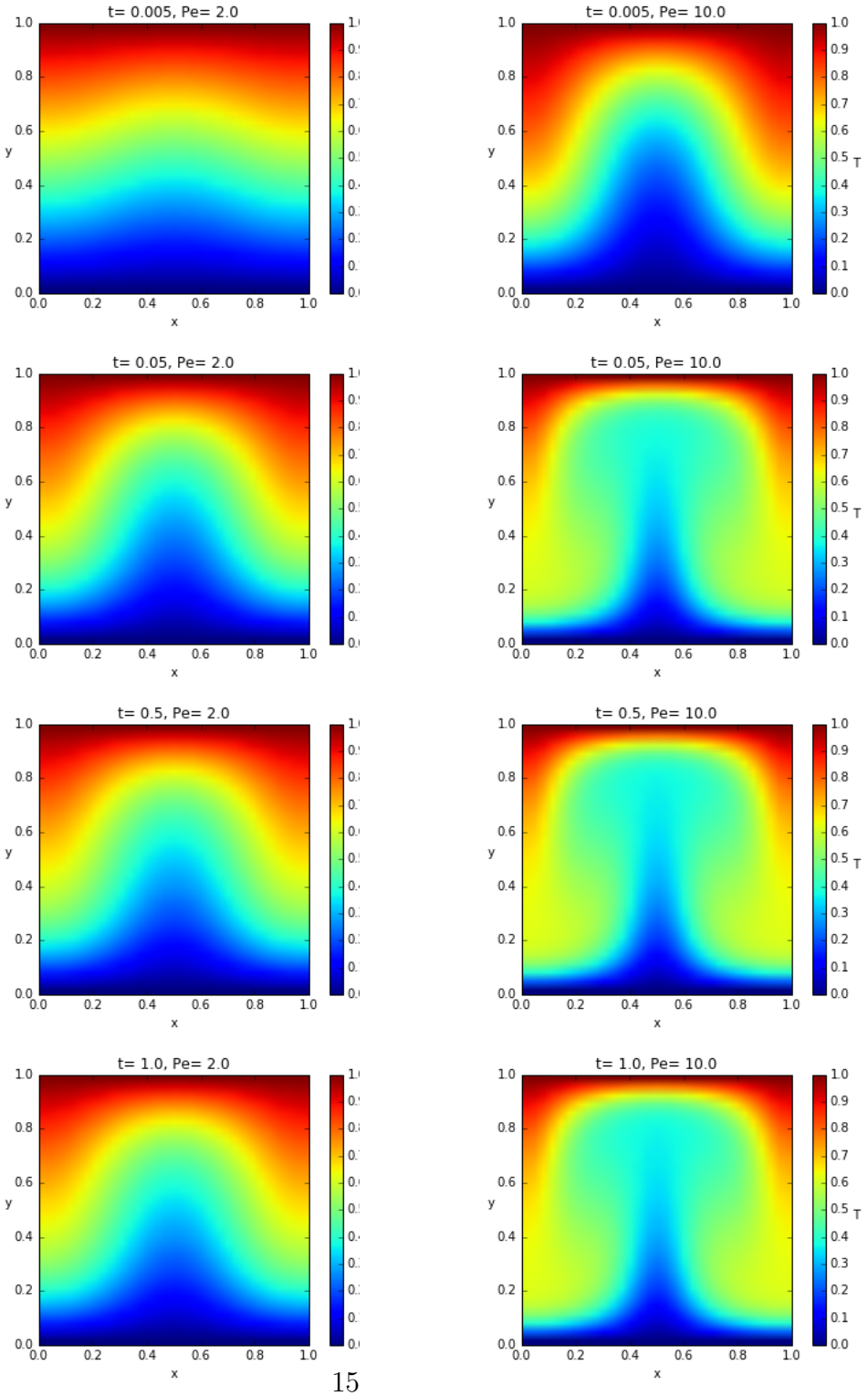


Abbildung 8: Temperaturfeld zu verschiedenen Zeiten und verschiedenen Pe. Berechnet mit dem FTCS-Verfahren. Links $Pe = 2$, rechts $Pe = 10$.

wird untersucht ab welcher Größe der Zeitschritt zu einer Divergenz des Verfahrens führt.
Eine theoretische Grenze für die Stabilität des Verfahrens wird durch das CFL-Kriterium für Diffusions- und Advektionsgleichungen vorgegeben. Für die Stabilität des Diffusionsterms muss folgende Ungleichung gelten:

$$\frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2} < 0.5. \tag{15}$$

Intuitiv lässt sich diese Gleichung verstehen als die Bedingung, dass die mittlere quadratische Verschiebung durch die Diffusion in einem Zeitschritt den Abstand zweier Gitterpunkte nicht übersteigen darf. Für eine Diffusionskonstante von 1 ergibt sich damit die CFL-Zahl 0.5.
Die zweite Stabilitätsbedingung des FTCS-Verfahrens rührt von dem Advektionsterm her und ist ähnlich zu verstehen wie Gleichung (15). In einem Zeitschritt darf die Verschiebung der Temperatur nur maximal den nächsten Gitterpunkt erreichen. Die Ungleichung für den zwei-dimensionalen Fall lautet [2]:

$$\frac{Pe\ u_{max}\Delta t}{\Delta x} + \frac{Pe\ v_{max}\Delta t}{\Delta y} < CFL = 1. \tag{16}$$

Für unser gegebenes Geschwindigkeitsfeld ergibt sich $u_{max} = \pi$ und $v_{max} = 2\pi$. In Abb. 9 sind die Stabilitätsgrenzen als Funktionen der Péclet-Zahl dargestellt. Der Umschlag des jeweils geltenden Kriteriums befindet sich bei $Pe = 12.56$ für eine konstant gehaltene Gittergröße mit $N = 30$.
Für die Péclet-Zahlen $Pe = (0.1, 1, 10, 20, 100)$ wurde das System mit unterschiedlichen Zeitschritten integriert. Abb. 10 zeigt die Entwicklung des Mittelwertes der Beträge der Gitterpunkte. Die jeweils größten stabilen Zeitschritte sind in Abb. 9 als Punkte eingetragen.
Aus der Analyse von Abb. 10 geht hervor, dass für $Pe = (0.1, 1, 10, 20)$ selbst Zeitschritte, die leicht über der Stabilitätsschranke liegen, noch stabil erscheinen.
Allerdings geht aus den Simulationen hervor, dass die der Divergenz für kleinere Zeitschritte später einsetzt, bzw. erkennbar wird. Daraus kann man schließen, dass man bessere Näherungen für den maximalen stabilen Zeitschritt erhält wenn man den Integrationszeitraum erhöht.
Ein anderes Problem tut sich jedoch auf wenn man $Pe = 100$ betrachtet. Diese wurde hier lediglich betrachtet, weil der Umschlag der Stabilitätsgrenze

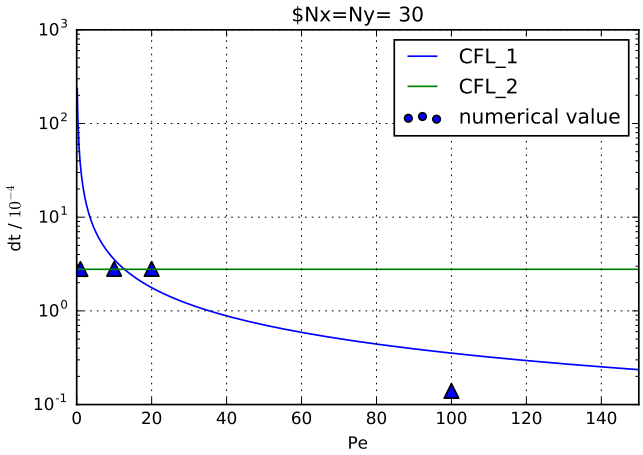


Abbildung 9: Theoretisch erwartete Grenzen für die Stabilität in Abhängigkeit von der Péclet-Zahl. Die Punkte markieren die in Abb. 10 gefundenen maximal stabilen Werte.

bei $Pe = 20$ nicht erkennbar war, wahrscheinlich wegen des oben beschriebenen Phänomens.
Im Gegensatz zu den kleineren Péclet-Zahlen, liefert die numerische Untersuchung von $Pe = 100$ Zeitschritte die nach Gleichung (16) stabil sein sollten, aber in der Praxis zu Divergenz führen.
Diese Tatsache verdeutlicht, dass es sich bei den CFL-Kriterien nur um notwendige Kriterien handelt und nicht um hinreichende. Eine Neumann'sche Stabilitätsanalyse könnte weitere Kriterien für die Stabilität des Verfahren für größere Péclet-Zahlen aufzeigen.

5.4 Feineinstellung des SOR-Algorithmus

Da in jedem Zeitschritt des BTCS-Verfahrens ein lineares Gleichungssystem in $N \cdot (N-2)$ Dimensionen gelöst werden muss, und dies Rechenzeit in Anspruch nimmt, ist es sinnvoll das SOR-Verfahren zu optimieren. Gesucht ist nun der Relaxationsparameter ω, für welchen die geringste Anzahl an Schritten nötig ist um die Gleichung bis auf eine bestimmte Fehlergrenze zu lösen.
Als Testgleichung wird $M\boldsymbol{x} = \boldsymbol{T}^0$ verwendet. Wobei $\boldsymbol{T}^0$ die Anfangsbedingung unseres Problems darstellt.

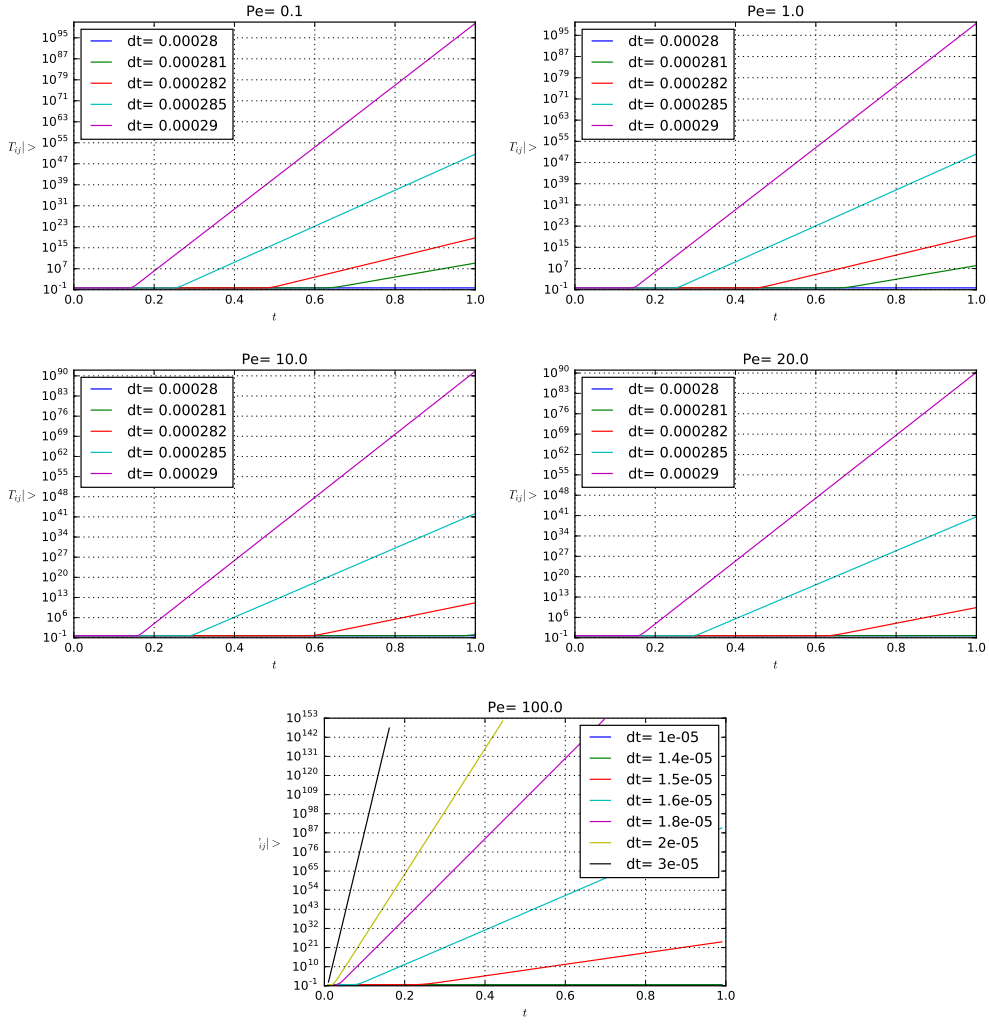
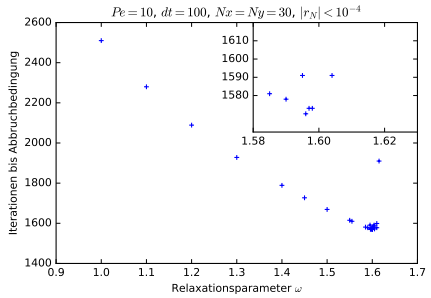
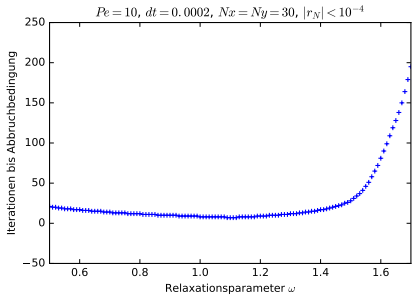


Abbildung 10: Suche nach dem maximalen stabilen Zeitschritt für verschiedene Péclet-Zahlen.



(a) Parameterscan in ω für die Parameter $Pe = 10$, $\Delta t = 100$ und $N = 30$. Der optimale ω-Wert liegt bei etwa 1.599.



(b) Parameterscan in ω für die Parameter $Pe = 10$, $\Delta t = 0.0002$ und $N = 30$. Der optimale ω-Wert liegt bei etwa 1.1.

Ein Parameterscan in ω wurde für die Matrizen M mit Parametern $Pe = 10$, $\Delta t = 100$, $N = 30$ und $Pe = 10$, $\Delta t = 0.0002$, $N = 30$ durchgeführt. Als Fehlergrenze für $|\boldsymbol{r}_n|$ wird 10^{-4} gewählt. Die Ergebnisse sind in Abb. 11a und Abb. 11b dargestellt.
Es zeigt sich dass der optimale Wert für ω stark von der Matrix M abhängt. Nicht nur da Minimum der Funktion liegt an unterschiedlichen Positionen ($\omega_{min} = 1.599$ für $\Delta t = 100$ und $\omega_{min} = 1.1$ für $\Delta t = 0.0002$), auch das Profil der Iterationen für jedes ω verändert sich stark. Für Abb. 11a explodiert die Anzahl der benötigten Iterationen kurz nach erreichen des optimalen Relaxationsparameters, während das Minimum bei einem Zeitschritt von $\Delta t = 0.0002$ wesentlich flacher ist.
Daraus lässt sich schließen, dass die Genauigkeit der Wahl des Relaxationsparameters bei bestimmten Matrizen einen bedeutenden Einfluss auf die Geschwindigkeit des Algorithmus nehmen kann.

5.5 Vergleich FTCS und BTCS

Mit dem optmimierten SOR-Verfahren kann nun sinnvoll ein Vergleich zwischen dem FTCS- und dem BTCS-Verfahren durchgeführt werden. Für die Parameter $Pe = 10$, $\Delta t = 0.0002$ und $N = 30$ is in Abb. 12 das Temperaturfeld für verschieden Zeitpunkte, mit dem FTCS- und dem BTCS-Verfahren integriert, aufgezeigt.
Es sind keine starken Unterschiede zu erkennen. Man kann unter Umständen eine Abweichung zur Zeit $t = 0.0054$ erkennen, diese ist aber nur sehr schwach

ausgeprägt.
Einen deutlicheren Unterschied bildet die Laufzeit der beiden Verfahren. Kleine Integrationszeiträume von $t_{end} = 0.5$ werden vom FTCS-Verfahren mit dem maximal möglichen Zeitschritt von 0.000277 in weniger als 1 s gelöst. Das BTCS-Verfahren benötigt dafür mit einem Zeitschritt von 0.5 ca. 55 s.
Bei längeren Integrationszeiträumen jedoch stößt die FTCS-Methode an ihre Grenzen, da sie nur für sehr kleine Zeitschritte stabil ist. Dadurch werden wesentlich mehr Iterationen benötigt, wodurch die Laufzeit anwächst. Die BTCS-Methode kann selbst bei großen Zeitschritten ein gutes Ergebnis liefern, wie in Abb. 13 zu sehen ist.
Für einem Integrationszeitraum von $t_{end} = 100$ liefert das FTCS-Verfahren eine Laufzeit von 120 s und das BTCS-Verfahren eine Laufzeit von 93 s, bei einem Zeitschritt von $\Delta t = 100$. Damit bestätigt sich die Vermutung, dass implizite Verfahren bei größeren Integrationszeiträumen, und wenn keine hohe Zeitauflösung benötigt wird, die effizientere Alternative zu den expliziten Methoden bilden.
Die genaue Performance der beiden Methoden, lässt sich jedoch nur relativ bestimmen, da bei der Implementierung der Algorithmen nicht auf Geschwindigkeit gesetzt wurde, sondern auf Übersichtlichkeit und Lesbarkeit des Quellcodes.

(a) BTCS (b) FTCS

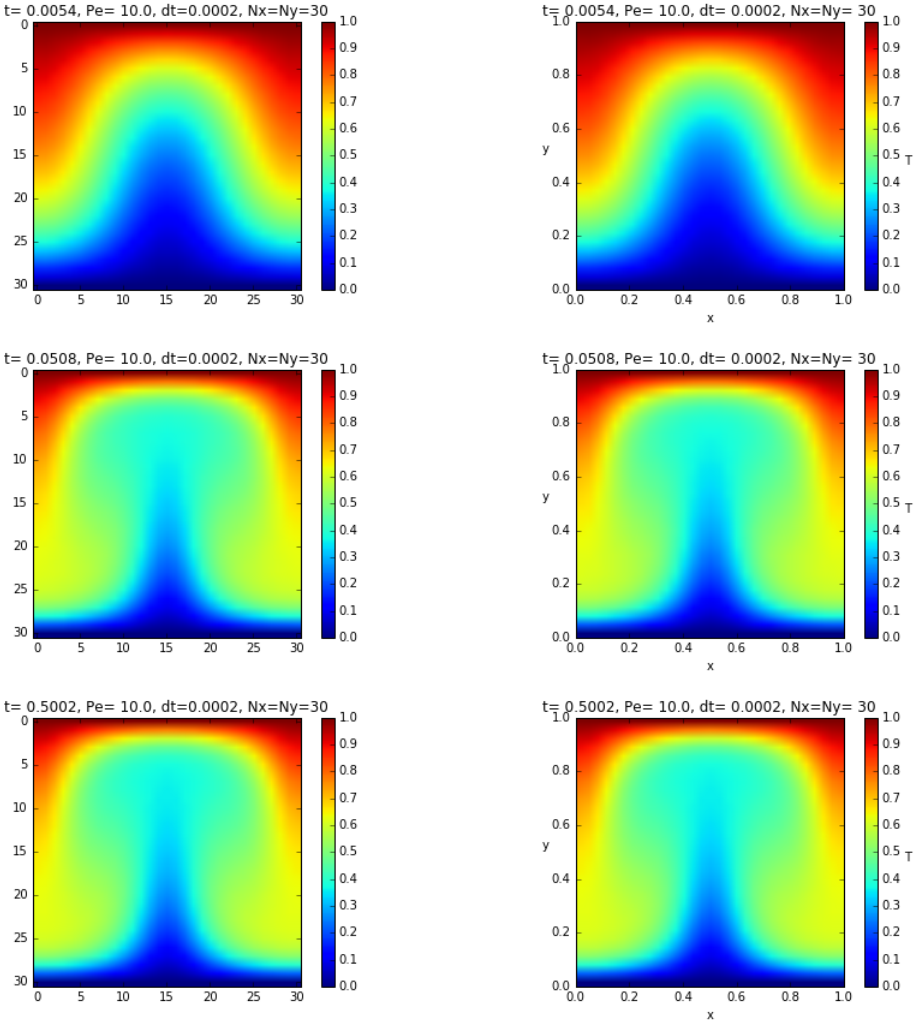
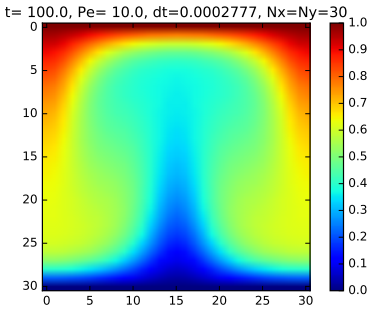


Abbildung 12: Vergleich zwischen den Temperaturfeldern, wie sie durch die BTCS- und FTCS-Integration zustande kommen.

(a) FTCS



(b) BTCS

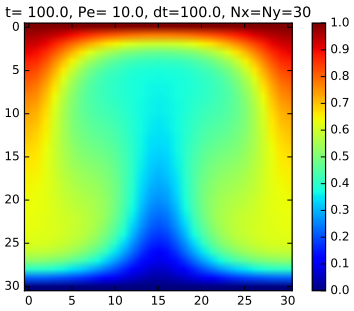


Abbildung 13: Vergleich zwischen den Temperaturfeldern, wie sie durch die BTCS- und FTCS-Integration zustande kommen, nach einem Zeitraum von $t = 100$.

6 Fazit

Die Vor- und Nachteile der BTCS-und FTCS-Verfahren wurden im Zuge dieser Ausarbeitung deutlich aufgezeigt. Während das FTCS-Verfahren sehr einfach zu implementieren ist, und pro Zeitschritt sehr effizient rechnet, sind die Möglichkeiten des FTCS-Verfahrens beschränkt. Eine obere Schranke an den Zeitschritt, um die Stabilität des Verfahrens zu garantieren, führt zu überaus langen Laufzeiten für Simulationen großer Zeiträume.
Interessiert man sich für die Simulation langer Zeiträume, bei relativ grober Gittereinteilung, so greift man besser zu dem BTCS-Schema. Zwar büßt man an Zeitauflösung ein, da man die Anzahl der Zeitschritte möglichst gering halten möchte, gewinnt aber an Stabilität, und ab einer bestimmten Größe des simulierten Zeitraums spart man ebenfalls Laufzeit gegenüber des FTCS-Schemas.
Implementiert man das BTCS-Verfahren, so muss der Relaxationsparameter sorgfältig gewählt werden, da er die Anzahl der Iterationen pro Zeitschritt deutlich reduzieren kann.

Literatur

[1] Louis A. Hageman and David M. Young (Auth.). *Applied Iterative Methods*. Computer Science Applied Mathematics. Elsevier Inc, Academic Press, 1981.

[2] Friedrichs K. Courant R. Lewy, H. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100:32–74, 1928.

[3] Prof. Dr. Milovan Perić (auth.) Prof. Dr. Joel H. Ferziger. *Numerische Strömungsmechanik*. Springer-Verlag Berlin Heidelberg, 1 edition, 2008.