Institut für Geophysik
Universität Göttingen

Hausarbeit
Numerische Strömungsmechanik

Bearbeitet von:	Sebastian Klipp
Matrikelnummer:	20978802
Zeitraum:	24.02.2012 - 2.03.2012
E-Mail:	sebastian.klipp@stud.uni-goettingen.de
Modulnummer:	B.Phy.562
Prüfer:	Prof. Dr. Andreas Tilgner

Note:

Inhaltsverzeichnis

1 Einleitung **3**

2 Theorie **3**
2.1 Physikalische Grundlagen . . . 3
2.1.1 Die Diffusions-Advektions-Gleichung . . . 3
2.1.2 Thermodynamische Kennzahlen . . . 3
2.2 Numerische Grundlagen . . . 4
2.2.1 Das FTCS-Schema . . . 4
2.2.2 Die Trapez-Regel . . . 5

3 Der Programmcode **6**
3.1 Grundstruktur . . . 6
3.2 Spezifische Code-Erweiterungen . . . 7
3.2.1 Quellterm und analytische Lösung . . . 7
3.2.2 Nusselt-Zahl . . . 7
3.2.3 Maximaler Zeitschritt . . . 7

4 Auswertung **8**
4.1 Aufgabe 1 - Numerische Lösung . . . 8
4.2 Aufgabe 2 - Stationärer Fall . . . 10
4.3 Aufgabe 3 - Nusselt-Zahl I . . . 14
4.4 Aufgabe 4 - Nusselt-Zahl II . . . 17
4.5 Aufgabe 5 - Zeitschritte . . . 18

5 Diskussion der Ergebnisse **20**

6 Quellenangaben **21**

7 Anhang **21**
7.1 Programmcode Grundstruktur . . . 21
7.2 Farbplots . . . 25

1 Einleitung

Die Diffusions-Advektions-Gleichung beschreibt den Transport von Partikeln unter Einfluss von Diffusion und Advektion. Dieses physikalische Phänomen wird mit Hilfe des FTCS-Schemas numerisch simuliert und in Hinblick auf wichige Kennzahlen wie Peclet-Zahl und Nusselt-Zahl und auf numerische Exaktheit und Stabilität überprüft.
Der Programmcode zu den spezifischen numerischen Berechnungen und verdeutlichende Animationen sind im Quellordner unter */Code* bzw. */Animation* zu finden.

2 Theorie

2.1 Physikalische Grundlagen

2.1.1 Die Diffusions-Advektions-Gleichung

Die Diffusions-Advektions-Gleichung ist eine Kombination aus Diffusions- und Advektions-Gleichung. Sie beschreibt den Transport von Partikeln, welche in einem gegebenen System durch die beiden Prozesse der Diffusion und der Advektion bewegt werden. Die Bewegung solcher Partikel kann somit folgendermaßen beschrieben werden:

$$\frac{\partial T}{\partial t} + \underbrace{\vec{v_0}\,\nabla T}_{Advektion} = \underbrace{D\,\nabla^2 T}_{Diffusion} \tag{1}$$

Hierbei ist T die Temperatur an einem bestimmten Ort, D die Diffusionskonstante und $\vec{v_0}$ das Geschwindigkeitsfeld. Überführt man Gl.1 in eine dimensionslose Form, so erhält man die folgende Relation:

$$\frac{\partial T}{\partial t} + Pe\;\vec{v_0}\,\nabla T = \nabla^2 T$$

Hierbei ist Pe die Peclet-Zahl.
Betrachtet man nur den stationären Fall, so fällt der Term mit der Zeitableitung von T aus der Gleichung heraus.

2.1.2 Thermodynamische Kennzahlen

Die Peclet-Zahl Pe
Die Peclet-Zahl ist eine dimensionslose Kennzahl in der Thermodynamik. Sie gibt das Verhältnis von konvektiv transportierter zu geleiteter Wärmemenge an. Die Peclet-Zahl kann z.B. durch das Produkt von Reynolds-Zahl Re und Prandtl-Zahl Pr bestimmt werden:

$$Pe = Re \cdot Pr$$

Die Nusselt-Zahl Nu
Die Nusselt-Zahl ist eine dimensionslose Kennzahl in der Thermodynamik. Sie beschreibt den Wärmeübertrag eines Körpers an strömende Fluide. Alternativ kann sie als Gradient der Temperatur an der Oberfläche des Körpers aufgefasst werden.
Ist die Nusselt-Zahl zweier geometrisch ähnlicher Aufbauten gleich, so ist auch ihre Wärmeübertragung an ein Fluid gleich und unabhängig von den Ausdehnungen der Aufbauten. Die Nusselt-Zahl wird berechnet über:

$$Nu = \int dx \, \frac{\partial T}{\partial y}$$

wobei die y-Achse senkrecht zur Körperoberfläche steht, an der der Wärmeübertrag stattfindet.
Eine weitere alternative Berechnung der Nusselt-Zahl findet sich in Kap.4.4.

2.2 Numerische Grundlagen

2.2.1 Das FTCS-Schema

Das FTCS-Schema (**F**orward in **T**ime **C**entral in **S**pace) wird zur numerischen Lösung von Gleichungen wie z.B. der Diffusions-Advektions-Gleichung benutzt. Dabei wird der Wert eines Knotenpunktes x_j zu einem Zeitpunkt $t+\Delta t$ nur durch Werte seiner Nachbarknoten zum Zeitpunkt t bestimmt. Es findet eine getrennte Diskretisierung von Raum und Zeit statt. Die Zeitableitung wird über eine Vorwärtsdifferenz mit Fehlerordnung 1 bestimmt:

$$\left(\frac{\partial T}{\partial t}\right)_j^t \approx \frac{T_j^{t+1} - T_j^t}{\Delta t}$$

Die Ortsableitungen werden mit zentrierten Differenzen von Fehlerordnung 2 berechnet:

$$\left(\frac{\partial T}{\partial x}\right)_j^t \approx \frac{T_{j+1}^t - T_{j-1}^t}{2\Delta x}$$
$$\left(\frac{\partial^2 T}{\partial x^2}\right)_j^t \approx \frac{T_{j+1}^t - 2T_j^t + T_{j-1}^t}{\Delta x^2}$$

Für die Diffusions-Advektions-Gleichung ergibt sich somit folgende Näherung:

$$T_{x,y}^{t+1} \approx T_{x,y}^t + \Delta t \left(\frac{T_{x+1,y}^t - 2T_{x,y}^t + T_{x-1,y}^t}{\Delta x^2} + \frac{T_{x,y+1}^t - 2T_{x,y}^t + T_{x,y-1}^t}{\Delta y^2}\right)$$
$$-\Delta t \; Pe \; \left(v_{0,x}\frac{T_{x+1,y}^t - T_{x-1,y}^t}{2\Delta x} + v_{0,y}\frac{T_{x,y+1}^t - T_{x,y-1}^t}{2\Delta y}\right)$$

Zur Berechnung dieser Gleichung werden außerdem noch Anfangs- und Randbedingungen benötigt. Während Dirichlet-Bedingungen einfach zu implementieren sind, benötigt das Neumann-Kriterium eine Ortsableitung am Rand, welche folgendermaßen definiert ist:

$$\left(\frac{\partial T}{\partial x}\right)^{t}_{Rand} = \frac{-\frac{3}{2}T_0^{t+1} + 2T_1^{t+1} - \frac{1}{2}T_2^{t+1}}{\Delta t} \tag{2}$$

wobei T_1 und T_2 das Temperaturfeld für eine bzw. zwei Ortskoordinaten vor/hinter dem Rand bestimmen. Umgeformt ergibt sich somit für den Randwert:

$$T_0^{t+1} = \frac{2}{3}\left(-\Delta t\left(\frac{\partial T}{\partial x}\right)^{t}_{Rand} + 2T_1^{t+1} - \frac{1}{2}T_2^{t+1}\right)$$

Die Implementierung des FTCS-Schemas folgt in der Regel einem bestimmten Ablauf:

1. Setze Anfangsbedingungen
2. Schleife über alle Zeitschritte
   a) Schleife über innere Ortspunkte mit Bestimmung der neuen Temperaturen (z.B. mit dem Euler-Schritt)
   b) Schleife über die Randpunkte mit Anwendung von Randbedingungen für die Randtemperaturen
   c) Übertrage die neuen Temperaturwerte in die Variable für die alten Temperaturwerte
3. Programmende

Die Stabilität der Lösung setzt sich zusammen aus der Stabilität des Diffusionsterms und der Stabilität des Advektionsterms, auch als CFL-Kriterium bezeichnet:

$$\text{Diffusion: } \frac{\Delta t}{\Delta x^2} + \frac{\Delta t}{\Delta y^2} \leq \frac{1}{2} \tag{3}$$

$$\text{Advektion: } Pe\ v_{0x}\frac{\Delta t}{\Delta x} + Pe\ v_{0y}\frac{\Delta t}{\Delta y} \leq 1 \tag{4}$$

2.2.2 Die Trapez-Regel

Die Trapezregel ist ein Verfahren zur numerischen Integration einer Funktion $f(x)$ im Intervall $[a; b]$. Dabei ersetzt man die Fläche unter $f(x)$ durch mehrere Trapeze, deren Höhe der Intervalllänge $h = \frac{b-a}{n}$ entspricht (Bei n Intervallunterteilungen). Das Ergebnis $F(f)$ der Integration über $f(x)$ wird folgendermaßen angenähert:

$$F(f) = \int_a^b f(x)\ dx = Q(f) + R(f)$$

wobei $Q(f)$ die durch die Trapezformel angenäherte Fläche ist und $R(f)$ ein Restterm.

$$Q(f) = h\left(\frac{1}{2}f(a) + \frac{1}{2}f(b) + \sum_{i=1}^{n-1} f(a + i \cdot h)\right)$$

3 Der Programmcode

3.1 Grundstruktur

Der exakte kommentierte Programmcode für die Grundstruktur ist in Anhang 7.1 zu finden, oder aber im Quellordner als Datei *hausarbeit.cpp*.

Kernelment ist hier die Funktion *main()*, deren Aufbau sich am FTCS-Schema orientiert und sich in folgende Bereiche gliedern lässt:

1. Variablen-Initialisierung
2. Festlegung der Anfangs- und Randbedingungen
3. Zeit- und Orts-Schleifen mit Ausführung des Euler-Schritts nach dem FTCS-Schema

Je nach Problemstellung können im Programmcode die in Tab.1 angegebenen Variablen angepasst werden. In der Grundform des Programmcodes wurden die Variablen auf Wer-

Pe	Die Peclet-Zahl
L	Seitenlänge des Kastens
tmax	Anzahl der Zeitschritte
dt	Länge des Zeitschrittes
Nx	Anzahl der diskreten Ortspunkte (ohne 0)
Ny	Anzahl der diskreten y-Ortspunkte (ohne 0)

Tabelle 1: Die physikalischen anpassbaren Größen

te gemäß der Aufgabenstellung (Aufgabe 1) gesetzt.
Nach Ausführung des Programms wird eine Datei *data.txt* erstellt, in denen die Ergebnisse des Algorithmus gespeichert sind. Die erste Spalte gibt hierbei den Zeitpunkt, die zweite die x-Koordinate, die dritte Spalte die y-Koordinate und die vierte Spalte den Temperaturwert an dem Ort (x,y) an.

3.2 Spezifische Code-Erweiterungen

3.2.1 Quellterm und analytische Lösung

Der exakte kommentierte Programmcode mit Quellterm und analytischer Lösung ist im Quellordner als Datei *hausarbeit2.cpp* zu finden. Die entsprechend der Aufgabe erweiterten Code-Zeilen sind in den Kommentaren durch den Zusatz *Code-Erweiterung* zu erkennen.

Gegenüber der Grundform wurde nun ein zusätzlicher Quellterm Q in die Diffusions-Advektions-Gleichung eingefügt. Um zusätzlich noch den Fehler der numerischen Lösung zur analytischen Lösung zu bestimmen, wird ein weiteres Array $Ta[x][y]$ für die analytische Temperaturlösung erstellt und ein Codeabschnitt, der den Fehler berechnet, angefügt.

3.2.2 Nusselt-Zahl

Der exakte kommentierte Programmcode mit der Berechnung der Nusseltzahl ist im Quellordner als Datei *hausarbeit3.cpp* zu finden. Die entsprechend der Aufgabe erweiterten Code-Zeilen sind in den Kommentaren durch den Zusatz *Code-Erweiterung* zu erkennen.

Gegenüber der Grundform wird eine weitere Datei *data_nu.txt* erstellt, welche für jeden Zeitschritt die Nusselt-Zahl enthält.
Außerdem wurde ein neuer Codeabschnitt hinzugefügt, der innerhalb der Zeitschleife mit Hilfe der Trapezregel die Nusselt-Zahl zu einem bestimmten Zeitpunkt t bestimmt und speichert.

3.2.3 Maximaler Zeitschritt

Der exakte kommentierte Programmcode mit der Bestimmung des maximalen Zeitschrittes ist im Quellordner als Datei *hausarbeit5.cpp* zu finden. Die entsprechend der Aufgabe erweiterten Code-Zeilen sind in den Kommentaren durch den Zusatz *Code-Erweiterung* zu erkennen.

Es soll nun mit Hilfe des Programmes bestimmt werden, welche Zeitschritte maximal zulässig sind für verschiedene Peclet-Zahlen Pe. Dafür wird der Programmcode so modifiziert, dass über das gesamte FTCS-Schema eine Schleife läuft, die mit jedem Durchlauf den Zeitschritt um $a = 10^{-6}$ erhöht. Nachdem das FTCS-Schema für einen bestimmten Zeitschritt durchlaufen wurde, wird der letzte stationäre Temperaturzustand über alle Orte gemittelt, so dass man eine mittlere Temperatur des Endzustandes $\bar{T}$ erhält. Da

jeder Ortspunkt maximal die Temperatur $T_{max}(x, y) = 1$ und minimal $T_{min}(x, y) = 0$ haben kann, muss für die mittlere Temperatur $0 \leq \bar{T} \leq 1$ gelten. Sobald nun bei einem bestimmten Zeitschritt Δt_{max} eine der Grenzen für $\bar{T}$ über- bzw. unterschritten wurde, erhält man keine stabile Lösung mehr, und das Programm gibt Δt_{max} aus.

4 Auswertung

4.1 Aufgabe 1 - Numerische Lösung

Die in diesem Abschnitt benötigten numerischen Berechnungen wurden mit dem Programmcode aus Kap.3.1 durchgeführt.

Führt man das Programm aus, so erhält man ein durch Diffusion und Advektion zeitlich veränderliches Temperaturfeld. In Abb.1 sieht man das Temperaturfeld T für alle Ortspunkte zur Zeit $t = 0$ farblich dargestellt, außerdem seinen stationären Zustand bei $t = 250$ in Abb.2 (Der Zustand kann als nahezu stationär angesehen werden, da sich die Temperaturwerte nur noch unbedeutend ändern). Für beide Plots wurden die in der Aufgabenstellung vorgeschlagenen Werte von $Pe = 2$, $\Delta t = 0.001$, $N_x = N_y = 10$ verwendet.

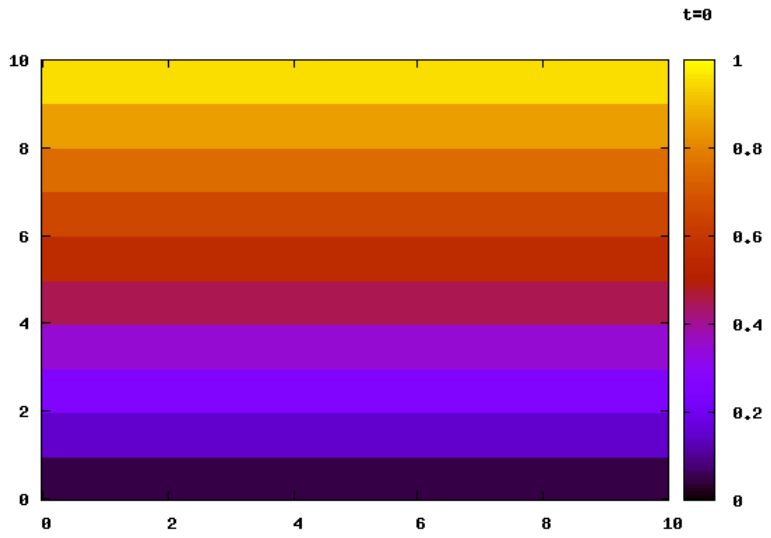

Abbildung 1: Temperaturfeld bei $t = 0$

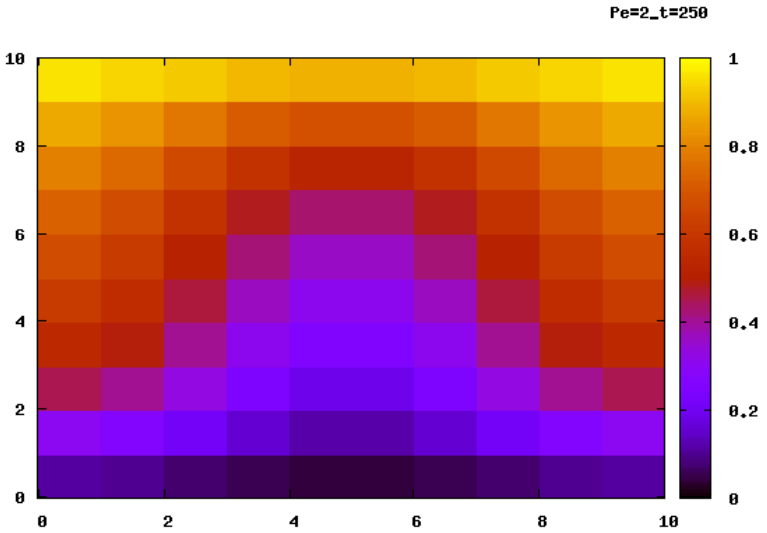

Abbildung 2: Stationäres Temperaturfeld bei $t = 250$, $Pe = 2$

Um sich zu vergewissern, dass das Geschwindigkeitsfeld die Partikel auch wie in der Aufgabenstellung gefordert transportiert, kann man zunächst das Geschwindigkeitsfeld an sich betrachten (siehe Abb.3). Man erkennt, dass es zwei Strömungen gibt, die ungefähr in Ellipsenform von der Unterseite des Systems durch die Mitte nach oben strömen, und sich an den Seitenrändern wieder nach unten bewegen.
An den beiden obigen Farbplots sieht man nun sehr gut, wie die Temperatur an der Unterseite des Systems hinausgetragen wird und somit kalte Partikel in der Mitte des Systems von dem Geschwindigkeitsfeld nach oben transportiert werden. An den Seitenrändern wiederum wird die hohe Temperatur, die von der Oberseite des Systems hineingebracht wurde, gemäß dem Geschwindigkeitsfeld nach unten transportiert. Schön zu erkennen sind auch die reibungsfreien Wände des Systems bei $x = 0$ und $x = 10$, hier wird keine Wärme aus dem System getragen, die Partikel gleiten reibungsfrei an den Wänden entlang. Es gibt also, wie in der Aufgabenstellung in den Randbedingungen gefordert, an diesen Stellen nur Gradienten des Temperaturfeldes parallel zu den beiden Seiten.
Um die spezifischen Eigenschaften unserer Problemstellung genauer zu untersuchen, wurden noch zwei weitere Messreihen aufgenommen für verschiedene Peclet-Zahlen Pe, so dass einmal für $Pe = 0.5$ die Diffusion und einmal für $Pe = 10$ die Advektion stark bevorzugt werden. Das stationäre Ergebnis für ersteren Fall ist im Anhang in Abb.12 angegeben, für den zweiten Fall wurde zusätzlich zur stationären Lösung (siehe Anhang Abb.14) noch ein Zwischenschritt bei $t = 10$ (siehe Anhang Abb.13) hinzugefügt,

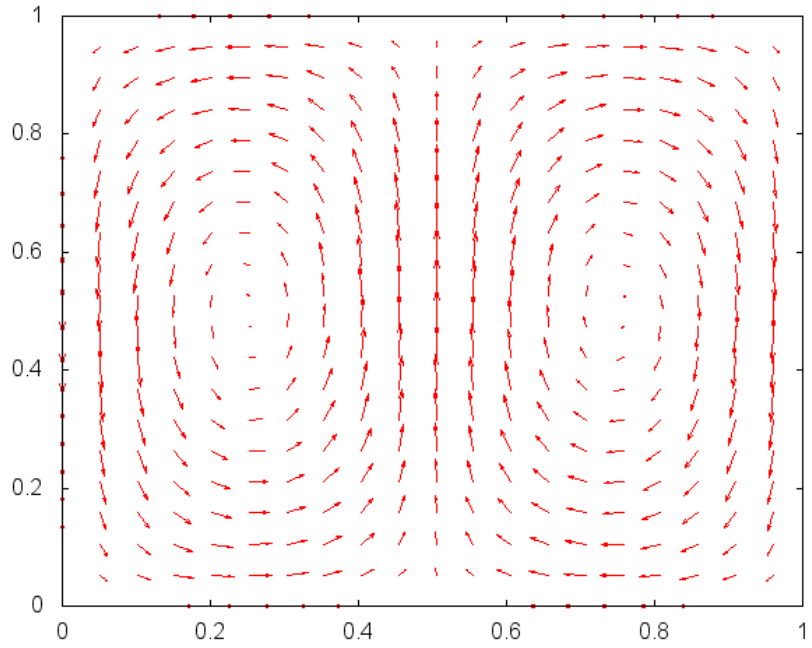

Abbildung 3: Geschwindigkeitsfeld des Systems

an welchem man sehr schön die Ausbreitung der Temperatur unter Beeinflussung des Geschwindigkeitsfeldes erkennen kann. Insgesamt sieht man gut, dass für kleine Peclet-Zahlen die Partikel kaum durch Advektion bewegt werden, da die Mitte des Systems nicht so stark von kalten Partikeln durchströmt wird und auch am Rand die hohen Temperaturpartikel sich viel weniger weit nach unten bewegen. Für hohe Pecletzahlen wird der Einfluss der Advektion sehr schön sichtbar.
Um den Temperaturverlauf mit der Zeit noch detaillierter zu beobachten, können zusätzlich die im Quellordner zu findenen Animationen *Pe=05.gif*, *Pe=2.gif* und *Pe=10.gif* betrachtet werden.

4.2 Aufgabe 2 - Stationärer Fall

Die in diesem Abschnitt benötigten numerischen Berechnungen wurden mit dem Programmcode aus Kap.3.2.1 durchgeführt.

Um die vorgegebene stationäre Lösung $T^* = \cos(\pi x)\sin(\pi y) + y$ zu erhalten, muss in die Diffusions-Advektions-Gleichung noch ein Quellterm Q eingefügt werden. Um diesen zu bestimmen, wird T^* in die Differentialgleichung eingesetzt, und diese einfach nach Q

umgeformt:

$$
\begin{aligned}
Q &= Pe\ \vec{v_0}\ \nabla T^* - \nabla^2 T^* \\
&= -Pe\ \pi^2 \sin\left(\frac{2\pi x}{L}\right) \sin(\pi x) \cos\left(\frac{\pi y}{L}\right) \sin(\pi y) \\
&\quad - Pe\ 2\pi^2 \cos\left(\frac{2\pi x}{L}\right) \cos(\pi x) \sin\left(\frac{\pi y}{L}\right) \cos(\pi y) \\
&\quad - Pe\ 2\pi \cos\left(\frac{2\pi x}{L}\right) \sin\left(\frac{\pi y}{L}\right) \\
&\quad + 2\ \pi^2 \cos(\pi x) \sin(\pi y)
\end{aligned}
$$

Dieses doch äußerst längliche Ergebnis könnte noch durch Additionstheoreme etwas zusammengefasst werden, jedoch wurde darauf verzichtet, da die obige Darstellung ohne komplizierte Sinus- und Kosinus-Argumente einfacher nachzuvollziehen ist.
In Abb.4 ist der Quellterm Q dargestellt. Man erkennt gut, dass er in der oberen rechten Ecke ein Temperaturminimum hat, während oben links ein Temperaturmaximum vorkommt. Die stationäre analytische Lösung T^* der Diffusions-Advektions-Gleichung ist in Abb.5 zu sehen. Hier bildet sich ein Minimum zwischen unterer Systemseite und Minimum des Quellterms aus, während das Maximux knapp unter dem des Quellterms liegt.

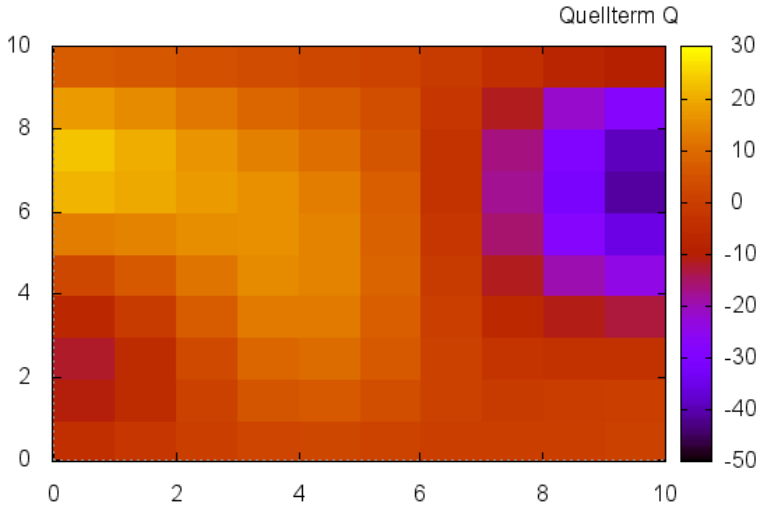

Abbildung 4: Quellterm Q

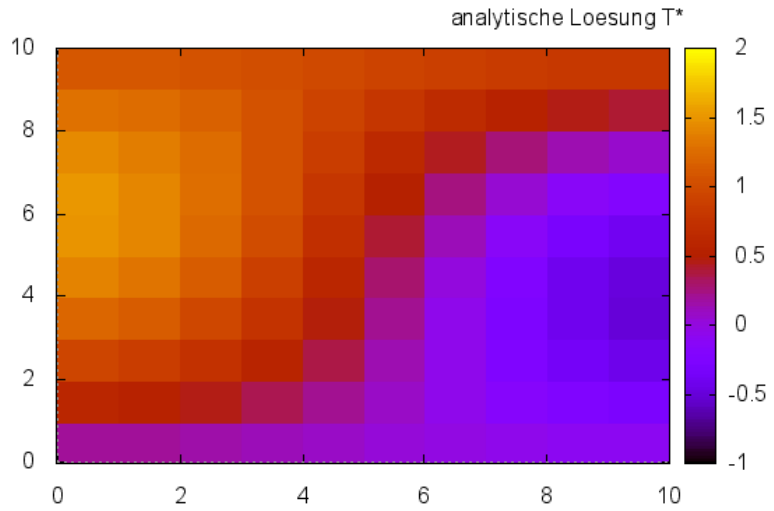

Abbildung 5: analytische Lösung T^*

Nun soll überprüft werden, ob die numerische Lösung T sich auch wirklich der analytischen Lösung T^* annähert. Dafür werden die Programm-Parameter etwas verändert. Es wird $\Delta t = 0.0001$ und $t_{max} = 4000$ gewählt ($Pe = 2$). Durch den kleineren Zeitschritt ist gewährleistet, dass die numerische Lösung stabil bleibt. Bei $t \approx 4000$ tritt ein annähernd stationärer Zustand auf, so dass wir das Temperaturfeld zu diesem Zeitpunkt mit dem der analytischen Lösung vergleichen können.
Zunächst einmal ist festzustellen, dass sich die numerische Lösung zeitlich so entwickelt, wie wir es vermutet haben. Es bildet sich ein Temperaturminimum auf der rechten Seite zwischen unterer Systemseite und Quellterm-Minimum aus, während ein Maximum kurz unter dem Quellterm-Maximum auf der linken Seite zu finden ist. In Abb.6 sieht man, wie sich das Temperatufeld nach 100 Zeitschritten ausgebreitet hat. Die numerische stationäre Lösung bei $t = 4000$ ist in Abb.7 zu finden. Außerdem kann der zeitliche Verlauf detaillierter in der Animation *Aufgabe2.gif* nachvollzogen werden.

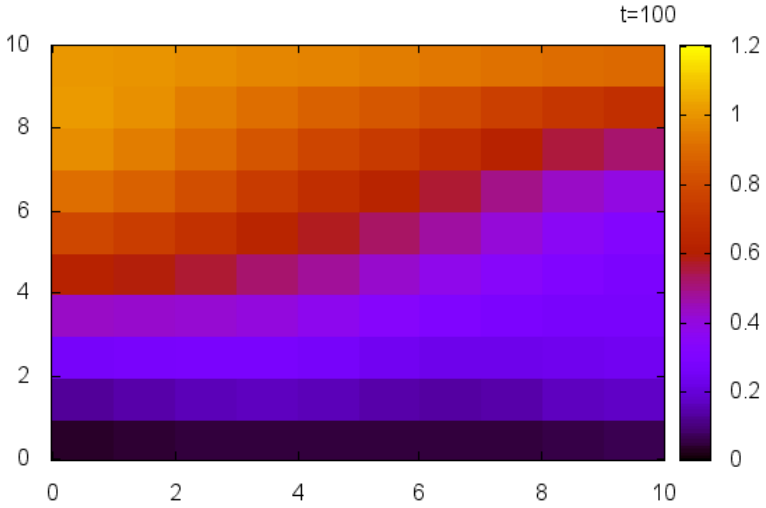

Abbildung 6: numerische Lösung T bei t=100

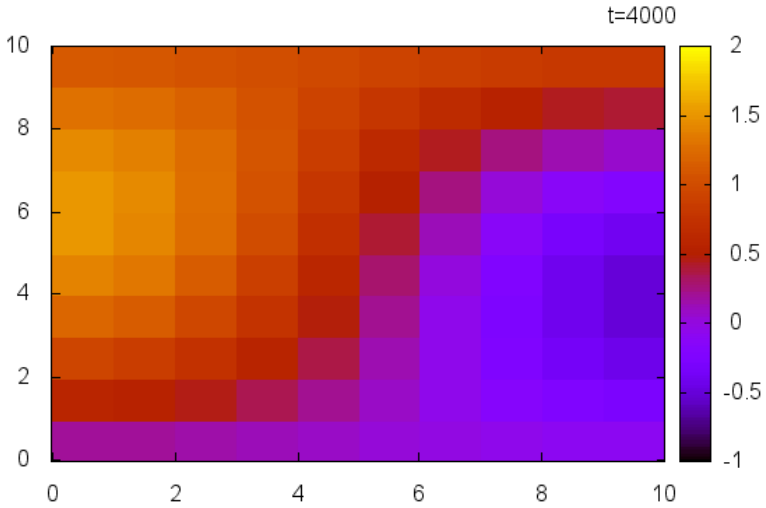

Abbildung 7: numerische Lösung T bei t=4000

Wie stark nun die Abweichung der stationären numerischen Lösung von der analytischen Lösung ist, lässt sich über den Fehler σ_T bestimmen:

$$\sigma_T = \sqrt{\sum_{ij}(T_{ij} - T^*_{ij})^2}$$

In Abb.8 ist der Fehler σ_T in Abhängigkeit der Ortsauflösung $N = N_x = N_y$ aufgetragen. Man erkennt sehr gut, dass mit zunehmender Auflösung der Fehler sinkt. Dies war zu

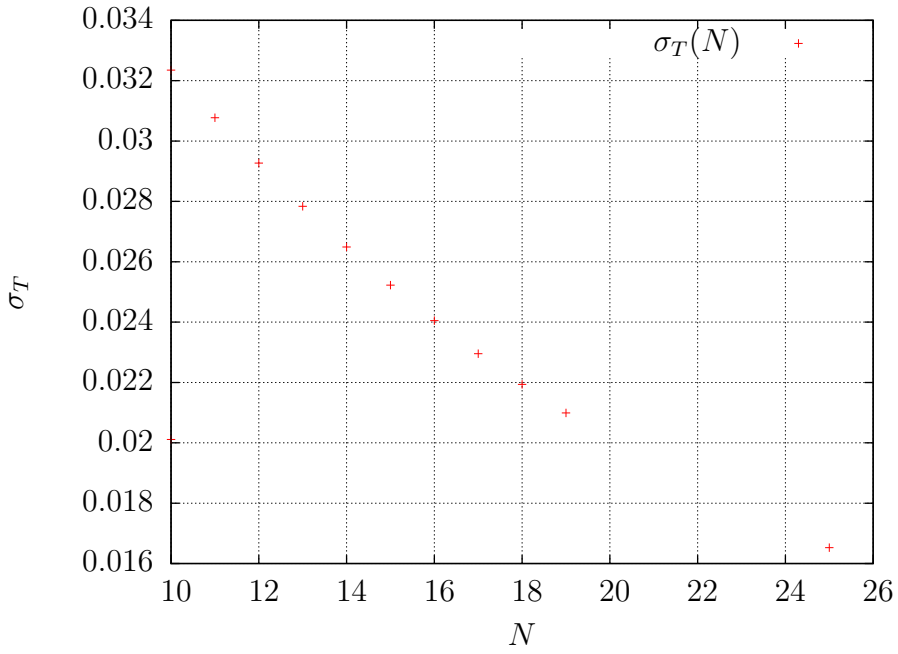

Abbildung 8: Fehler σ_T beider Lösungen in Abhängigkeit der Auflösung N

erwarten, da eine höhere Auflösung eine stärkere Nähe zur Realität bedeutet. Auch wenn es scheint, dass der Fehler linear sinkt, wird die Abnahme des Fehlers mit höheren Auflösungen doch immer kleiner. Es ist zu erwarten, dass σ_T sich mit N gegen ∞ an die x-Achse anschmiegt. Leider war es nicht möglich, noch höhere Auflösungen zu testen, da der Rechenaufwand für den verwendeten Computer zu immens wäre.

4.3 Aufgabe 3 - Nusselt-Zahl I

Die in diesem Abschnitt benötigten numerischen Berechnungen wurden mit dem Programmcode aus Kap.3.2.2 durchgeführt.

Mit Hilfe der Trapezregel (siehe Kap.2.2.2) kann das Integral

$$Nu = \int dx \frac{\partial T}{\partial y}(t, x, 0) \tag{5}$$

numerisch für jeden Zeitschritt gelöst werden. Die Ableitung nach y wird hierbei über die Formel angenähert, mit der auch die Neumann-Randbedingungen berechnet werden (siehe Gl.2). Es wurden die System-Parameter von Aufgabe 1 mit $\Delta t = 0.001$, $t_{max} = 250$ und $N_x = N_y = 10$ verwendet, so dass die Nusselt-Zahlen bis in den stationären Zustand bestimmt werden. In Abb.9 ist die Nusselt-Zahl Nu für die Parameterwerte $Pe = 2$ und $Pe = 10$ in Abhängigkeit des Zeitschritts aufgetragen.

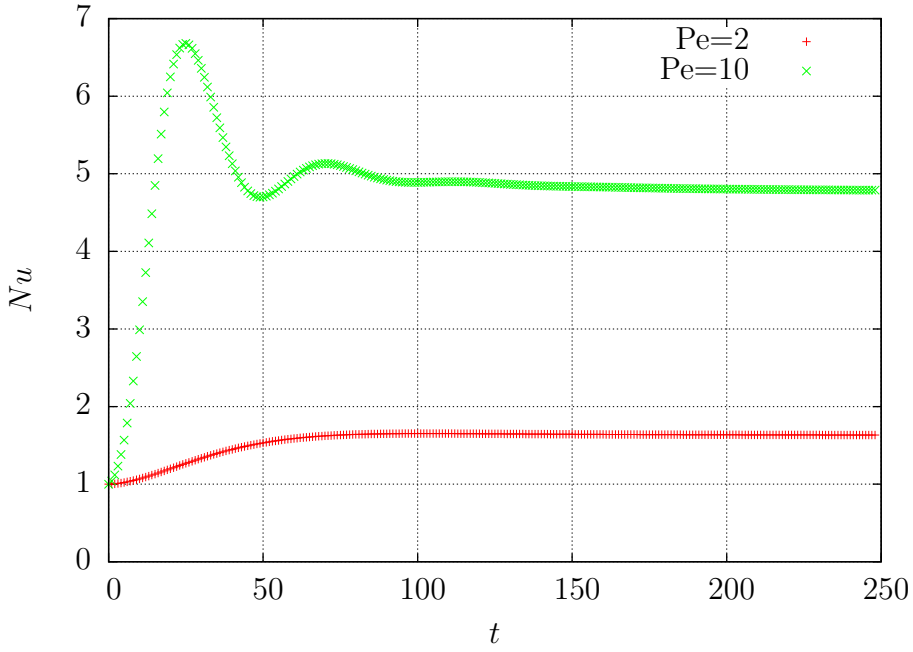

Abbildung 9: Nusselt-Zahl Nu in Abh. von t für verschiedene Peclet-Zahlen Pe

Man erkennt deutlich, dass für die kleinere Peclet-Zahl $Pe = 2$ ein schwaches Maximum bei $t = 102$ ausgebildet wird, danach bleibt Nu nahezu konstant. Für die größere Peclet-Zahl $Pe = 10$ werden deutlich zwei Maxima bei $t = 25$ und $t = 70$ und ein Minimum bei $t = 49$ sichtbar, danach bleibt Nu nahezu konstant, nur ein geringer Abfall ist erkennbar.
Da wir mit der Nusseltzahl den Wärmetransport der unteren Systemseite an das Fluid bei $y = 0$ beschreiben, wird bei den Maxima der Nusselt-Zahl besonders viel Wärme transportiert, ein großer Austausch findet zu solchen Zeitpunkten statt. Um dies genauer zu betrachten, wird in den sich im Anhang befindenden vier Abbildungen 15 bis 18 das Temperaturfeld zu den Zeitschrittpunkten der Maxima und Minima der Nusseltzahl

angegeben. Die Anfangs- bzw. stationären Zustände in beiden Fällen wurden bereits in Abb.1, Abb.2 und Abb.14 gezeigt.
Betrachtet man den Fall mit $Pe = 10$, so erkennt man in den Abbildungen zu den Zeitpunkten $t = 25$ und $t = 70$, dass besonders an den Rändern nahe zur unteren Systemseite hohe Temperaturen vorkommen, im Vergleich zu der Abbildung zum Zeitpunkt $t = 49$ sind die Farbwerte viel heller und damit die Temperatur höher. Hohe Temperaturen nahe der unteren Systemseite bedeuten, dass viel Wärme aus dem System genommen wird, denn es muss die Randbedingung $T(x, 0) = 0$ für alle Zeiten erfüllt werden. Viel entfernte Wärme bedeutet einen hohen Wärmetransport und somit eine hohe Nusseltzahl. Bei $t = 49$ finden wir in der Nähe von $y = 0$ geringere Temperaturen, somit muss auch weniger Wärme abtransportiert werden, und die Nusseltzahl ist kleiner. Das Maximum der Nusselt-Zahl für $Pe = 2$ lässt sich auf gleiche Weise erklären.
Untersucht man die Abhängigkeit der Nusselt-Zahl Nu von der Peclet-Zahl Pe (siehe Abb.10), so findet man, dass für von $Pe = 0$ ansteigende Peclet-Zahlen die Nusselt-Zahl konkav ansteigt, bis sie bei ca. $Nu(Pe = 5) = 3.25$ einen Wendepunkt erreicht und von da an nur noch konvex ansteigt. Bei einer Pecletzahl $Pe << 1$ nähert sich die Nusseltzahl Nu dem Wert 1 an. In diesem Zustand trägt die Advektion kaum noch zum Wärmetransport bei, und jeglicher Wärmeübertrag durch die untere Systemseite geschieht durch Diffusion. Das Steigen von Nu bei höheren Pecletzahlen ist durch die stärkere Advektion zu erklären, die durch die Strömung innerhalb des Systems Wärme von der oberen Systemseite zur unteren transportiert.

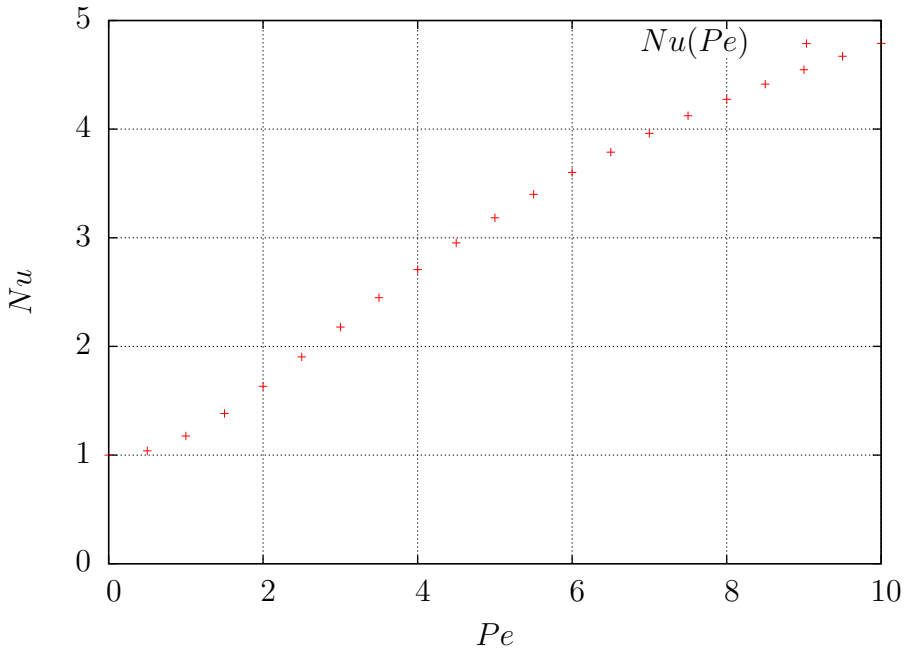

Abbildung 10: Abhängigkeit der Nusseltzahl Nu von der Peclet-Zahl Pe

4.4 Aufgabe 4 - Nusselt-Zahl II

Alternativ zur Relation 5 kann die Nusseltzahl auch über folgende Gleichung bestimmt werden:

$$Nu - 1 = -Pe \int_0^1 \int_0^1 v_{0y}(x,y)T(x,y)dx\,dy$$

Als Beweis verwenden wir die stationäre Diffusions-Advektions-Gleichung und integrieren diese über einen Kasten, dessen Seitenränder entlang der unteren Systemwand ($y = 0$), an den beiden Systemrändern bis zu einer Höhe y_0 und über eine Linie mit $y_0 = konst$ verlaufen. Außerdem darf man den Geschwindigkeitsvektor und den Gradienten des Advektionsterms vertauschen. Die Gleichung stellt sich nun folgendermaßen dar:

$$0 = \int_V \nabla(Pe\ \vec{v_0}T - \nabla T)dx\,dy$$

Nutzt man nun den Satz von Gauß für die Ebene, kann man den ersten Gradienten im Integral ersetzten, integriert wird nun über die Seitenränder des Kastens:

$$0 = \oint_{\partial V} (Pe\ \vec{v_0}T - \nabla T)d\vec{r}$$

Das Wegintegral lässt sich nun aufteilen in eine Integration im Intervall [0;1] über dx und in eine Integration im Intervall [0;y_0] über dy.

$$\begin{aligned} 0 = &\int_0^1 \left(Pe\ v_{0y}(x,y_0)T(x,y_0) - Pe\ v_{0y}(x,0)T(x,0) - \frac{\partial T}{\partial y}(x,y_0) + \frac{\partial T}{\partial y}(x,0) \right) dx \\ &+ \int_0^{y_0} \left(Pe\ v_{0x}(1,y)T(1,y) - Pe\ v_{0x}(0,y)T(0,y) - \frac{\partial T}{\partial x}(1,y) + \frac{\partial T}{\partial x}(0,y) \right) dy \end{aligned}$$

In diesem langen Term finden sich nun die Neumann-Randbedingungen wieder, wodurch diese Terme null werden. Außerdem verschwindet der Geschwindigkeitsanteil v_{0x} an den beiden Seitenwänden bei $x = 0$ und $x = 1$. Zusätzlich ist die Temperatur bei $y = 0$ stets $T(x,0) = 0$, wodurch ein weiterer Term entfällt. Der nächste notwendige Schritt ist eine weitere Integration über y, so dass schließlich folgende Gleichung entsteht:

$$\begin{aligned} 0 &= \int_0^1 \int_0^1 \left(Pe\ v_{0y}(x,y_0)T(x,y_0) - \frac{\partial T}{\partial y}(x,y_0) + \frac{\partial T}{\partial y}(x,0) \right) dx\,dy_0 \\ &= \int_0^1 \int_0^1 Pe\ v_{0y}(x,y)T(x,y)dx\,dy - \int_0^1 \int_0^1 \left(\frac{\partial T}{\partial y}(x,y_0) - \frac{\partial T}{\partial y}(x,0) \right) dx\,dy_0 \\ &= \int_0^1 \int_0^1 Pe\ v_{0y}(x,y)T(x,y)dx\,dy + \int_0^1 \frac{\partial T}{\partial y}T(x,0)dx - \int_0^1 (T(x,1) - T(x,0))dx \end{aligned}$$

Im letzten Umformungsschritt findet sich nun die Definition der Nusseltzahl Nu wieder. Das letzte Integral wird noch ausgeführt, und die gesamte Gleichung nach $Nu - 1$ umgestellt, so dass sich als Ergebnis die gesuchte Relation ergibt:

$$Nu - 1 = -Pe \int_0^1 \int_0^1 v_{0y}(x, y)T(x, y)dx\, dy \tag{6}$$

Für die numerische Berechnung der Nusselt-Zahl Nu gibt es nun für beide möglichen Varianten Vor- und Nachteile. Ein Vorteil von Relation 6 ist, dass keine Ableitungen benötigt werden. Ableitungen müssen numerisch immer genähert werden, sind fehlerbehaftet und erzeugen zusätzlichen Rechenaufwand. Allerdings werden nun statt nur einer Integration wie in Relation 5 zwei Integrationen benötigt, dies bedeutet wieder zusätzlichen Rechen- und Speicheraufwand. Je nach numerischer Methode für Integration und Differentiation und erwünschtem Rechenaufwand sollte man abwägen, welche Berechnungsvariante sinnvoll ist.
Ein schöner weiterer Vorteil der zweiten Variante für die qualitative Arbeit mit der Nusselt-Zahl ist, dass man hier direkt den Einfluss der Peclet-Zahl Pe und des Geschwindigkeitsfeldes v_{0y} ablesen kann.

4.5 Aufgabe 5 - Zeitschritte

Die in diesem Abschnitt benötigten numerischen Berechnungen wurden mit dem Programmcode aus Kap.3.2.3 durchgeführt.

In Abb.11 sind die Verläufe der im stationären Zustand über alle Orte gemittelten Temperatur $\bar{T}$ in Abhängigkeit von Δt für verschiedene Peclet-Zahlen Pe angezeigt, wobei jeweils nur der für die Stabilität relevante Abschnitt geplottet wurde. Die maximalen Zeitschritte Δt_{max} sind in Tab.2 angegeben.

Pe	0.1	1	10
Δt_{max}	0.002708	0.002685	0.002204

Tabelle 2: Numerisch ermittelte Zeitschritte Δt_{max} für verschiedene Peclet-Zahlen Pe

Man erkennt, dass mit größer werdender Peclet-Zahl Pe der Zeitschritt immer kleiner werden muss, um stabile Lösungen zu erhalten. Da mit hoher Pecletzahl die Advektion an Bedeutung gewinnt, scheint eine starke Advektion kleinere Zeitschritte zu fordern. Dies werden wir im Folgenden genauer untersuchen. Die Größe des Zeitschritts ergibt sich theoretisch aus dem CFL-Kriterium für Diffusion und Advektion (siehe Kap.3), wobei für die Geschwindigkeitsanteile maximale Werte von $v_{0x} = \pi$ und $v_{0y} = 2\pi$ angenommen werden.

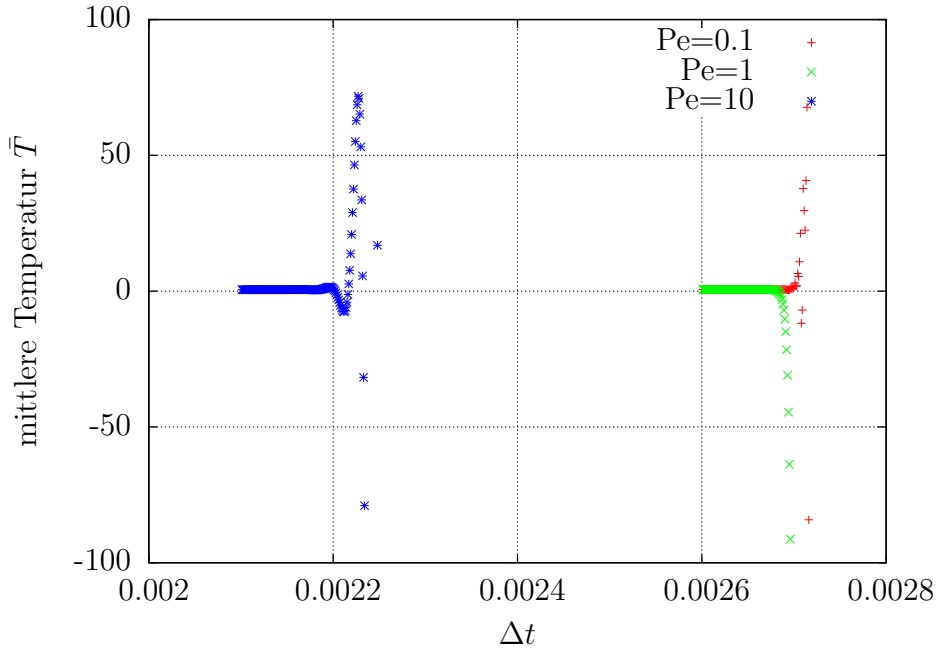

Abbildung 11: Mittlere Temperatur im stat. Zustand $\bar{T}$ in Abh. vom Zeitschritt Δt

Zusammen mit $\frac{1}{\Delta x} = N_x$ und $\frac{1}{\Delta y} = N_y$ erhält man somit die maximal zulässigen Zeitschritte:

$$\text{Diffusion: } \Delta t_{max,D} \leq \frac{1}{2}\frac{1}{N_x^2 + N_y^2}$$

$$\text{Advektion: } \Delta t_{max,A} \leq \frac{1}{Pe(v_{0x}N_x + v_{0y}N_y)}$$

Betrachtet man nun die drei Fälle für die verschiedenen Peclet-Zahlen, so erkennt man, dass für $Pe = 0.1$ und für $Pe = 1$ die Advektion stark unterdrückt wird. Laut Advektions-Stabilitätskriterium wäre das Ergebnis nämlich bei einem großen Zeitschritt von $\Delta t_{max} = 0.106103$ bzw. $\Delta t_{max} = 0.010610$ noch stabil, allerdings zwingt das Diffusions-Stabilitätskriterium mit einem maximalen Zeitschritt von $\Delta t_{max} = 0.002500$ den tatsächlichen Zeitschrittwert nah in seinen Größenordnungsbereich, der Einfluss der Diffusion dominiert.
Während diese Dominanz bei $Pe = 1$ schon schwächer wird, und der Zeitschritt Δt_{max} etwas größer ausfallen darf, sind schließlich bei $Pe = 10$ beide Prozesse bei Betrachtung der Größenordnung etwa gleichberechtigt. Das Advektionskriterium benötigt einen Zeitschritt von $\Delta t_{max} = 0.001061$, dies ist nun sogar geringer als die Einschränkung durch das Diffusionskriterium. Der experimentell ermittelte Zeitschritt findet sich zwischen diesen beiden Werten wieder.
In Tab.3 sind die Ergebnisse für Δt_{max} für beide Stabilitätskriterien bei verschiedenen

Peclet-Zahlen angegeben.

Pe	0.1	1	10
Diffusion: $\Delta t_{max,D}$	0.002500	0.002500	0.002500
Advektion: $\Delta t_{max,A}$	0.106103	0.010610	0.001061

Tabelle 3: Theoretisch bestimmte Zeitschritte Δt_{max} für verschiedene Peclet-Zahlen Pe

5 Diskussion der Ergebnisse

Insgesamt ist die numerische Simulation der Diffusions-Advektions-Gleichung gut gelungen. Die Temperatur breitet sich selbst bei geringer Auflösung gemäß Diffusion und Advektion aus. Es konnte sehr schön an Beispielen nachvollzogen werden, wie die Peclet-Zahl Pe das Verhältnis von Advektion und Diffusion bestimmt.
Auch das Hinzufügen eines Quellterms, um eine vorgegebene stationäre Lösung zu erhalten, führt zu erfreulichen Ergebnissen, denn die Erhöhung der Auflösung zeigt, dass der Fehler der numerischen Lösung stetig absinkt, was für ein gutes numerisches Verfahren spricht.
Die Nusseltzahl verhält sich wie erwartet. Treten hohe Temperaturen in der Nähe der unteren Systemgrenze auf, steigt auch der Wärmetransport und somit die Nusseltzahl an.
Weiterhin passen die numerisch und theoretisch bestimmten maximalen Zeitschritte größenordnungtechnisch gut zueinander, jedoch behindert die Überlagerung des Stabilitätskriteriums für Diffusion und Advektion ein genaues Vorhersagen des realen maximalen Zeitschritts.

Insgesamt war es schade, dass es kaum möglich war, die Aufgaben bei großer Auflösung zu untersuchen, da die Rechenleistung des verwendeten Computers leider nicht ausreichte, um die Simulation in angemessener Zeitspanne durchzuführen. Daher kann z.B. die Entwicklung des Fehlers der numerischen Lösung für große Auflösungen nur abgeschätzt werden.
Weiterhin war es problematisch, mit einem numerisch stationären Zustand zu arbeiten, denn auch wenn bei den verwendeten Zuständen bei Zeitänderung kaum Schwankungen auftraten, so waren sie dennoch gering vorhanden. Eine höhere Anzahl von Zeitschritten hätte hier den Fehler verringern können, jedoch gelangt auch hier wieder der verwendete Computer an die Grenzen seiner Rechenleistung, so dass leider eine geringere Anzahl von Zeitschritten verwendet werden musste, als anfangs gewünscht.

6 Quellenangaben

- Wassim Abu Abed: Masterarbeit Finite Volumen Methode zur Lösung Advektions-Diffusionsgleichungen
- Prandtl, Oswatitsch, Wieghardt: Führer durch die Strömungslehre. 9.Auflage. vieweg Verlag.
- Ferziger, Peric: Computational Methods for Fluid Dynamics. 3.Auflage. Springer Verlag.
- Lehrportal: https://lp.uni-goettingen.de
- www.Wikipedia.org

7 Anhang

7.1 Programmcode Grundstruktur

```
////
// Programmcode zu Aufgabe 1 - Grundstruktur des Programmcodes
//  Numerische Loesung der Diffusions-Advektions-Gleichung und Ausgabe der Ergebnisse in der Datei "data.txt"
////

// Paket-Einbindung
#include <fstream>
#include <cmath>

using namespace std;

//// Beginn der Main-Funktion
int main()
{

    //// Initialisierung der Programm-relevanten Variablen
    ofstream data("data.txt");           // Erstellt Ausgabedatei "data.txt"

    double Pe = 2;                       // Peclet-Zahl
    double L = 1;                        // Laenge der Kasten-Seiten (in der nicht-dimensionslosen-Form)

    int tmax = 300;                      // Anzahl der Zeitschritte
    double dt = 0.001;                   // Laenge eines Zeitschrittes
    int Nx = 10;                         // (Nx+1) ist die Anzahl der diskreten Raumpunkte in x-Richtung
    double dx = (double)1./Nx;          // Laenge eines Ortsschrittes in x-Richtung
                                         //   (in dimensionsloser Form -> Seitenlaenge des Kastens ist 1)
    double x2;                           // dimensionlose x-Ortskoordinate, wichtig fuer Berechnungen: x2 = (double)x*dx ,
                                         //    wobei x die nicht-dimensionslose Koordinate ist
    int Ny = 10;                         // (Ny+1) ist die Anzahl der diskreten Raumpunkte in y-Richtung
    double dy = (double)1./Ny;          // Laenge eines Ortsschrittes in y-Richtung
                                         //   (in dimensionsloser Form -> Seitenlaenge des Kastens ist 1)
    double y2;                           // dimensionlose y-Ortskoordinate, wichtig fuer Berechnungen: y2 = (double)y*dy ,
                                         //    wobei y die nicht-dimensionslose Koordinate ist

    double Ti[Nx+1][Ny+1];               // Temperaturfeld-Array fuer den alten Zeitschritt t;
                                         //   Ti[x][y] gibt die Temperatur am Ort (x,y) zur Zeit t an
    double Tf[Nx+1][Ny+1];               // Temperaturfeld-Array fuer den neuen Zeitschritt t+1;
                                         //   Tf[x][y] gibt die Temperatur am Ort (x,y) zur Zeit t+1 an


    //// Erstelle Anfangs- und Randbedingungen fuer Ti[x][y] und Tf[x][y]
    for (int x=0; x <=Nx; x++)
    {
        Tf[x][0]  = 0;          // y-Randbedingung: Bei y=0 ist Tf fuer alle y-Werte gleich 0
        Tf[x][Ny] = 1;          // y-Randbedingung: Bei y=Ny ist Tf fuer alle x-Werte gleich 1

        for (int y=0; y <=Ny; y++)
        {
            y2 = (double)y*dy;
            Ti[x][y] = y2;      // Anfangsbedingung: Bei t=0 ist Ti[x][y] gleich y
        }
    }


    //// Speichere die Anfangswerte Ti[x][y] in "data.txt"
    for (int y=0; y<=Ny; y++)
    {
        for (int x=0; x<=Nx; x++)
        {
            data << 0 << "\t" << x << "\t" << y << "\t" << Ti[x][y] << endl;
        }
        data << endl;   // Nach jedem Block durchlaufener x-Werte gibt es eine Leerzeile
    }
```

```
64
65
66      //// Beginne mit der Zeit-Schleife fuer den Euler-Schritt
67      for (int t = 0; t<(tmax-1); t++)
68      {
69          //// Speichere die y-Randwerte unten bei y = 0 in der Datei "data.txt"
70          for (int x=0; x<=Nx; x++)
71          {
72             data << t+1 << "\t" << x << "\t" << 0 << "\t" << Tf[x][0] << endl;
73          }
74          data << endl;       // Nach jedem Block durchlaufener x-Werte gibt es eine Leerzeile
75
76
77          //// Durchlaufe nun alle Orts-Koordinaten, um innerhalb der Schleifen mit der Euler-Methode die Temperatur zu berechnen
78          for (int y = 1; y<Ny; y++) // Ortsschleife y; y beginnt bei 1 und laeuft nur bis (Ny-1); Randbedingungen werden separat behandelt.
79          {
80              y2 = (double)y*dy;
81
82              for (int x = 1; x<Nx; x++) // Ortsschleife x; x beginnt bei 1 und laeuft nur bis (Nx-1); Randbed. werden separat behandelt.
83              {
84                  x2 = (double)x*dx;
85
86                  //// Berechne nun mit dem Eulerschritt und der zentralisierten Naeherung die Tf[x][y] fuer den naechsten Zeitschritt
87                  Tf[x][y] = Ti[x][y] + dt* (           // Euler-Schritt
88                                          ((Ti[x+1][y] - 2.*Ti[x][y] + Ti[x-1][y])/(dx*dx))
89                                                // Diffusionsterm I;  zentrierte Naeherung der 2.Ableitung (x)
90                                          + ((Ti[x][y+1] - 2.*Ti[x][y] + Ti[x][y-1])/(dy*dy))
91                                                // Diffusionsterm II; zentrierte Naeherung der 2.Ableitung (y)
92                                          - Pe * (M_PI*sin(2.*M_PI*x2/L)*cos(M_PI*y2/L)) * (Ti[x+1][y] - Ti[x-1][y])/(2.*dx)
93                                                // Advektionsterm I;  zentrierte Naeherung der 1.Ableitung (x)
94                                          - Pe * (-2.*M_PI*cos(2.*M_PI*x2/L)*sin(M_PI*y2/L)) * (Ti[x][y+1] - Ti[x][y-1])/(2.*dy)
95                                                // Advektionsterm II; zentrierte Naeherung der 1.Ableitung (y)
96                                         );
97              }
98
99
100             //// Setze x-Randbedingung links (Neumann-Bedingung, die Ableitung ergibt null)
101             Tf[0][y] = 2./3. * ( -dt* 0. + 2.*Tf[1][y] - 1./2. * Tf[2][y] );
102
103             //// Setze x-Randbedingung rechts (Neumann-Bedingung, die Ableitung ergibt null)
104             Tf[Nx][y] = 2./3. * ( -dt* 0. + 2.*Tf[Nx-1][y] - 1./2. * Tf[Nx-2][y] );
105
106
107             //// Speichere die neuen Temperaturwerte in der Datei "data.txt"
108             for (int x=0; x<=Nx; x++)
109             {
110                 data << t+1 << "\t" << x << "\t" << y << "\t" << Tf[x][y] << endl;
111             }
112             data << endl;   // Nach jedem Block durchlaufener x-Werte gibt es eine Leerzeile
113
114         }
115
116
117         //// Speichere die y-Randwerte oben bei y = 1 in der Datei "data.txt"
118         for (int x=0; x<=Nx; x++)
119         {
120            data << t+1 << "\t" << x << "\t" << Ny << "\t" << Tf[x][Ny] << endl;
121         }
122         data << endl;       // Nach jedem Block durchlaufener x-Werte gibt es eine Leerzeile
123
124
125         ////   bertrage   nun die neuen Werte Tf nach Ti, wo sie im naechsten Zeitschritt als alte Zeitwerte dienen,
126           //   so dass neue Tf berechnet werden koennen
```

```
127             for (int x=0; x<=Nx; x++)
128             {
129                 for (int y=0;y<=Ny;y++)
130                 {
131                     Ti[x][y] = Tf[x][y];
132                 }
133
134             }
135
136         }   // Ende der Zeit-Schleife
137
138     }   // Ende von main()
```

7.2 Farbplots

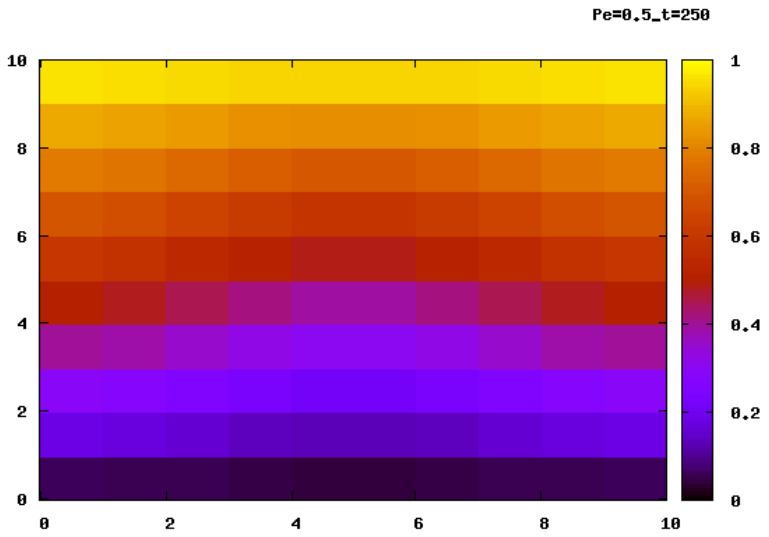

Abbildung 12: Stationäres Temperaturfeld bei $t = 250$, $Pe = 0.5$

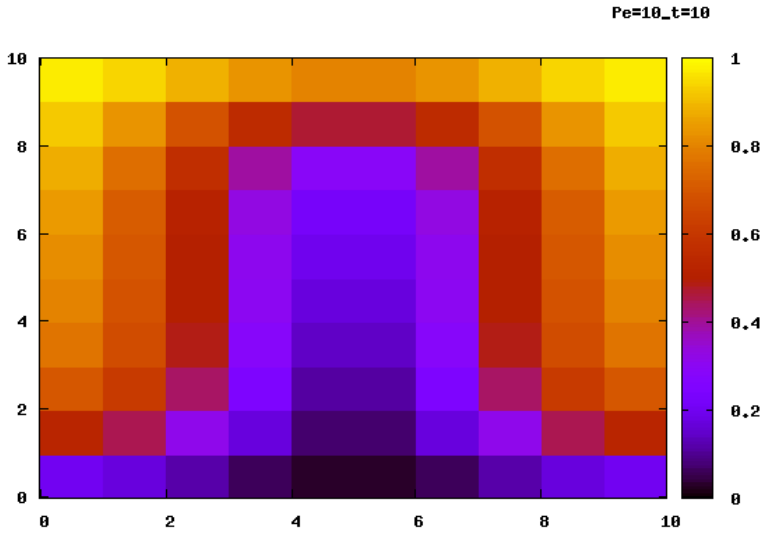

Abbildung 13: Temperaturfeld bei $t = 10$, $Pe = 10$

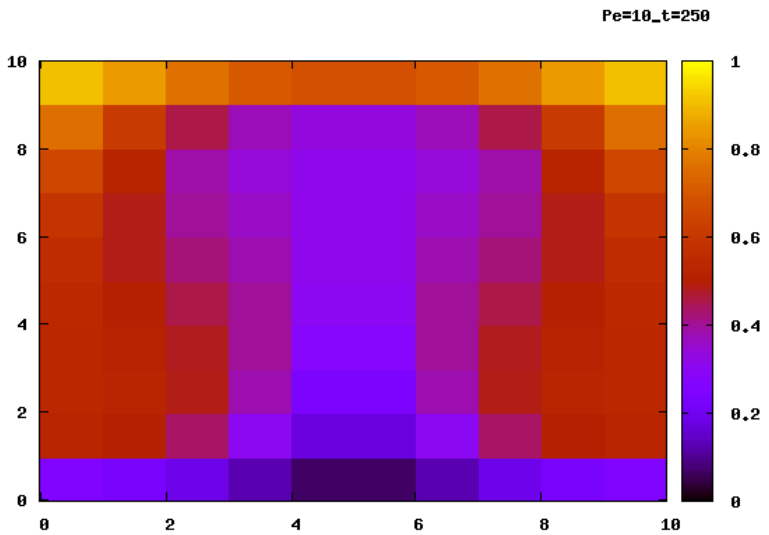

Abbildung 14: Stationäres Temperaturfeld bei $t = 250$, $Pe = 10$

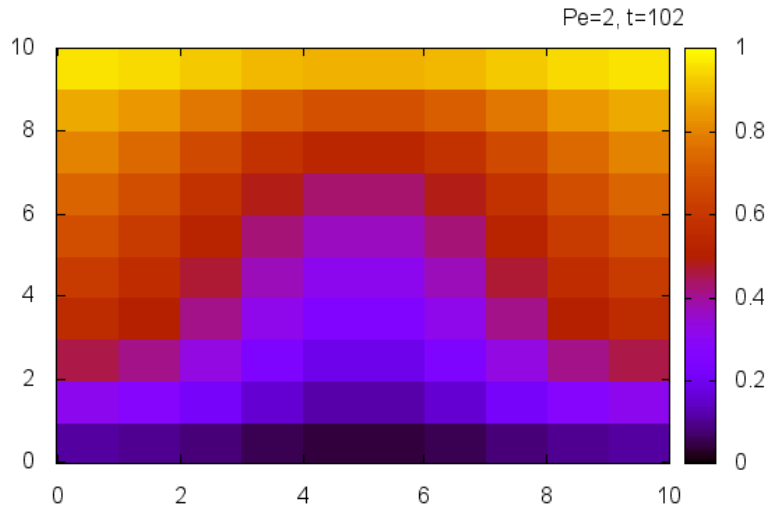

Abbildung 15: Temperaturfeld $T(x, y)$ zur Zeit $t = 102$ (Maximum von Nu), $Pe = 2$

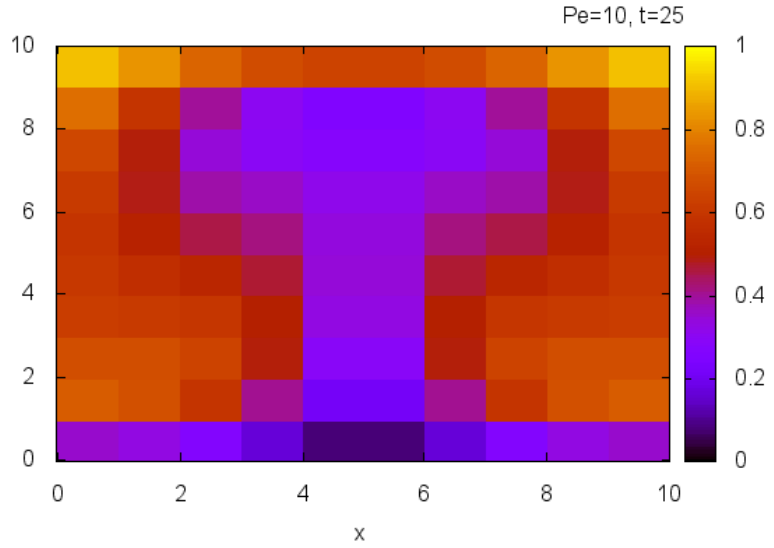

Abbildung 16: Temperaturfeld $T(x, y)$ zur Zeit $t = 25$ (Maximum von Nu), $Pe = 10$

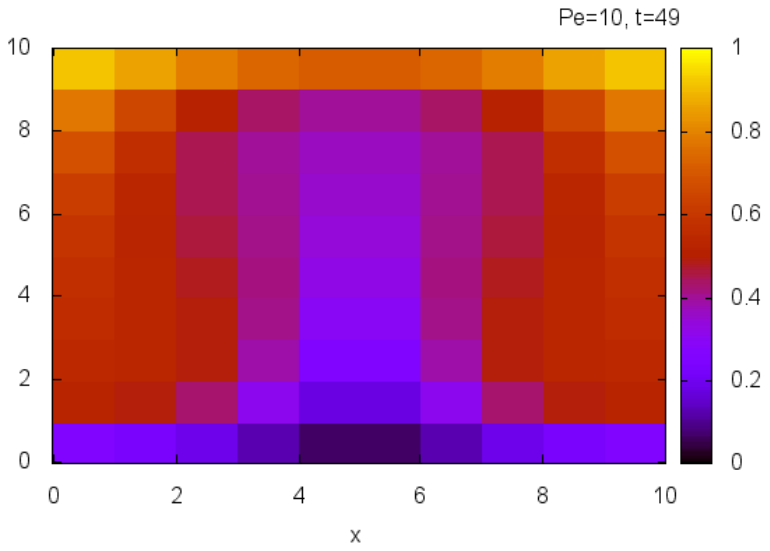

Abbildung 17: Temperaturfeld $T(x, y)$ zur Zeit $t = 49$ (Minimum von Nu), $Pe = 10$

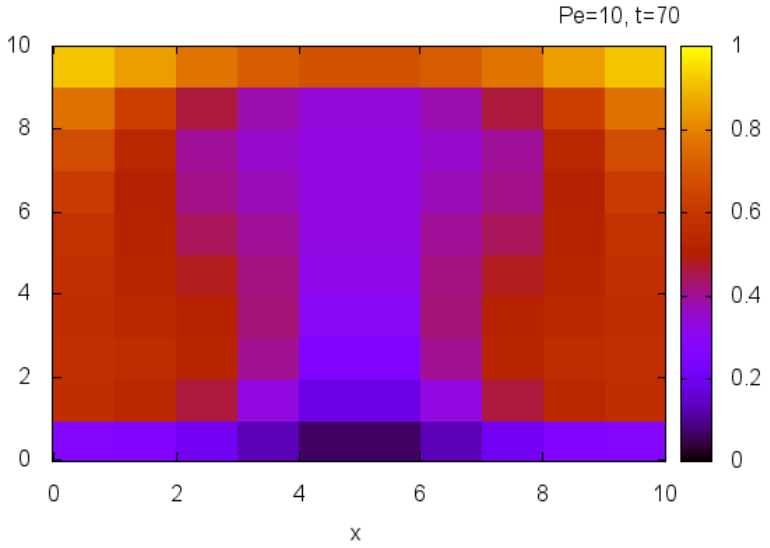

Abbildung 18: Temperaturfeld $T(x, y)$ zur Zeit $t = 70$ (Maximum von Nu), $Pe = 10$