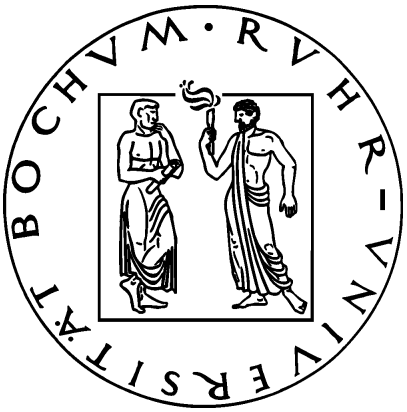
Strömungen in komplexen Geometrien

DIPLOMARBEIT

von

Felix Spanier



Institut für Theoretische Physik I

Plasma-, Laser- und Atomphysik

Ruhr-Universität Bochum

Bochum, Oktober 2002

Inhaltsverzeichnis

Abbildungsverzeichnis

Einleitung

Obschon die Hydrodynamik zu den ältesten Zweigen der modernen Physik gehört (die Euler-Gleichung ist seit 1755 bekannt), gibt es immer noch aktuelle Fragestellungen in diesem Gebiet. Neben den klassischen analytischen Ansätzen zur Lösung hydrodynamischer Probleme spielen die numerische Lösungen durch die ständig wachsenden Möglichkeiten der Nutzung von Computern und der zunehmden Komplexität der Probleme eine immer größere Rolle.

Die vorliegenden Arbeit beschäftigt sich mit hydrodynamischen Problemen, die unterschiedliche Geometrien umströmter Körper berücksichtigen. Dies ist ein Gebiet, das analytisch oftmals schwer oder gar nicht zu lösen ist. Numerische Lösungen sind zwar schon seit längerem möglich, allerdings waren sie bislang mit erheblichen Geschwindigkeits- und Stabilitätsproblemen behaftet.

Im Rahmen dieser Arbeit habe ich aus bekannten CWENO- und Projektionsverfahren ein neues numerisches Schema zur Lösung der Navier-Stokes-Gleichung entwickelt, das die Vorteile der beiden Methoden miteinander verbindet. Von besonderem Interesse war dabei die Verwendung des Schemas im Rahmen der Adaptiven Gitterverfeinerung, die mit den bisher bekannten Projektionsverfahren nicht ohne weiteres genutzt werden kann.

In der Regel wurde die Navier-Stokes-Gleichung bisher für komplexe Strömungsgeometrien auf der Basis der Finite-Elemente-Methode (FEM) gelöst. Deren Hauptvorteil liegt in Verwendung problemangepaßter Koordinaten, jedoch ergeben sich gerade dadurch auch große Nachteile bei dieser Methode. Die Koordinaten müssen zunächst einmal generiert werden, was mit erheblichem Zeitaufwand verbunden ist. Daher sind sie in Bezug auf die Prozessoroptimierung wenig geeignet. Ein weiteres Problem in der praktischen Anwendung ist der indirekte Speicherzugriff durch die neuen Gitter, die eine Cache-Optimierung

unmöglich machen. Dies führt zu erheblichen Rechenzeiten.

Um die Probleme der FEM zu umgehen, werden in dieser Arbeit Penalty-Methoden verwendet, die mit beliebigen Koordinaten arbeiten und somit wesentliche Geschwindigkeitsvorteile bieten. Mit den Penalty-Methoden wird das neu entwickelte numerische Schema für die Navier-Stokes-Gleichung gekoppelt, um einfache und komplexe Strömungsgeometrien zu simulieren. Von besonderem Interesse war es, diesen Schritt auch adaptiv zu behandeln, da dies eine Neuerung zu bekannten Verfahren darstellt.

Für die hier entwickelten Verfahren gibt es vielfältige Nutzanwendungen, von denen die klassischen Windkanalsimulationen sicher die bekanntesten sind. Aber auch die am Lehrstuhl betriebene Sonnenwindforschung kann einen Nutzen aus den Ergebnissen dieser Arbeit ziehen, da eine Verwendung für die Sonne als Geometrie möglich ist. Damit werden die bisherigen Randwertprobleme in der Sonnenwindsimulation durch einen einfachen Mechanismus lösbar. Die Anwendungen der hier entwickelten Verfahren sind nicht nur auf rein hydrodynamische Probleme beschränkt, sondern erlauben darüber hinaus auch die Lösung magneto-hydrodynamischer Fragestellungen und anderer numerischer Randwertprobleme.

1. Theoretische Grundlagen

1.1. Navier-Stokes-Gleichung

In der Hydrodynamik werden Flüssigkeiten als Kontinuum betrachtet. Es werden also Volumenelemente der Flüssigkeit betrachtet, die hinreichend viele Teilchen enthalten. Die erste wichtige Größe, die Geschwindigkeit $\mathbf{u}$, ist also die Geschwindigkeit eines Volumenelementes und somit die mittlere Geschwindigkeit aller enthaltenen Teilchen. Neben der Geschwindigkeit des Volumenelements ist die nächste wichtige Größe die Dichte ρ, die die Anzahl der Teilchen pro Volumen angibt.

1.1.1. Herleitung der Euler-Gleichung

Um nun die Bewegungsgleichung der Volumenelemente in einer idealen (reibungsfreien) Flüssigkeit zu erhalten, betrachten wir den von außen auf das Volumen wirkenden Druck p und erhalten als Gesamtkraft

$$\mathbf{F} = -\oint p \; d\mathbf{f} \tag{1.1}$$

In ein Volumenintegral umgeformt (Gauß'scher Integralsatz)

$$\mathbf{F} = -\int \nabla p \; dV \tag{1.2}$$

Einsetzen in die Newtonsche Bewegungsgleichung liefert dann für ein Volumenelement

$$m\frac{d\mathbf{u}}{dt} \;\; = \;\; \mathbf{F} \tag{1.3}$$

Wird die Masse durch das Volumenintegral der Dichte ersetzt und die Geschwindigkeitsableitung mit in das Integral gezogen, erhält man:

$$\int \rho \frac{d\mathbf{u}}{dt}\, dV = -\int \nabla p\, dV \tag{1.4}$$

$$\rho \frac{d\mathbf{u}}{dt} = -\nabla p \tag{1.5}$$

Diese Gleichung gibt das sogenannte *Lagrange-Bild* wieder. Hierbei wird die Geschwindigkeit nicht für Raumpunkte, sondern für die Volumenelemente angegeben, das Koordinatensystem wird also mit dem Volumenelement mitbewegt. Um die Änderung der Geschwindigkeit an den Raumpunkten zu bestimmen, benötigen wir die *substantielle Ableitung*, die aus der Definition der totalen Ableitung folgt

$$d\mathbf{u} = \frac{\partial \mathbf{u}}{\partial t} dt + (d\mathbf{r}\nabla)\mathbf{u} \tag{1.6}$$

$$\Rightarrow \frac{d\mathbf{u}}{dt} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\nabla)\mathbf{u} \tag{1.7}$$

Damit erhalten wir die *Euler-Gleichung*

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p \tag{1.8}$$

Im *Euler-Bild* ist die Geschwindigkeit eine Funktion des Ortes und der Zeit

$$\mathbf{u} = \mathbf{u}(\mathbf{r}, t) \tag{1.9}$$

Analog läßt sich die Gleichung für den Impulsstrom, die zeitliche Änderung der Impulsdichte $\rho\mathbf{u}$, in Tensorschreibweise darstellen:

$$\frac{\partial}{\partial t}(\rho u_i) = \rho \frac{\partial u_i}{\partial t} + u_i \frac{\partial \rho}{\partial t} \tag{1.10}$$

Dabei ist die partielle Ableitung von $\mathbf{u}$ nach t aus der Euler-Gleichung (s. Gl. 1.8)bekannt (hier in Tensorschreibweise)

$$\frac{\partial u_i}{\partial t} = -u_k \frac{\partial u_i}{\partial x_k} - \frac{1}{\rho}\frac{\partial p}{\partial x_i} \tag{1.11}$$

$$\Rightarrow \frac{\partial}{\partial t}(\rho u_i) = -\rho u_k \frac{\partial u_i}{\partial x_k} - \frac{\partial p}{\partial x_i} - u_i \frac{\partial \rho u_k}{\partial x_k} \tag{1.12}$$

$$= -\frac{\partial p}{\partial x_i} - \frac{\partial}{\partial x_k}(\rho u_i u_k) \tag{1.13}$$

$$= -\delta_{ik}\frac{\partial p}{\partial x_k} - \frac{\partial}{\partial x_k}(\rho u_i u_k) \tag{1.14}$$

$$\frac{\partial \rho u_i}{\partial t} = -\frac{\partial}{\partial x_k}\Pi_{ik} \tag{1.15}$$

$$\Pi_{ik} = -p\delta_{ik} - \rho u_i u_k \tag{1.16}$$

Die Größe Π_{ik} heißt Impulsstromdichtetensor und gibt den Fluß des Impulses an. Da der Impuls ein Vektor ist, muß der Fluß des Impuls ein Tensor sein. Man kann aus der Struktur des Tensors folgern, daß der Impulsfluß senkrecht zur Bewegungsrichtung allein durch den Druck p, parallel dazu aber durch $\rho u^2 + p$, also Druck und Staudruck, bestimmt ist.

1.1.2. Kontinuitätsgleichung

Neben der Erhaltung des Impulsstroms existiert in diesem Zusammenhang noch eine weitere Erhaltungsgröße, nämlich die Massen- oder Teilchenzahlerhaltung. Anschaulich bedeutet dies, daß der Ein- und Ausfluß eines Volumenelements gleich der Teilchenzahländerung in dem Element sein muß

$$\frac{\partial}{\partial t} \int \rho \, dV = - \oint \rho \mathbf{u} \, dV \tag{1.17}$$

$$\int \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) \right) \, dV = 0 \tag{1.18}$$

$$\tag{1.19}$$

Das Integral darf auf Grund der Beliebigkeit der Volumenelemente ausgelassen werden.

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{1.20}$$

Dies ist die Kontinuitätsgleichung, die sich weiter vereinfacht, wenn die Flüssigkeit als inkompressible (ρ=const) angenommen wird

$$\nabla \cdot \mathbf{u} = 0 \tag{1.21}$$

1.1.3. Zähe Flüssigkeiten

Zähigkeit in einer Flüssigkeit bedeutet, daß zwischen unterschiedlich schnellen Schichten (Newtonsche) Reibung auftritt. Um diese zu beschreiben, wird dem Impulsstromdichtetensor (s. Gl. 1.15) einen weiteren Term hinzugefügt

$$\Pi_{ik} = p\delta_{ik} + \rho u_i u_k - \sigma_{ik} \tag{1.22}$$

σ_{ik} ist der Spannungstensor, der von dem Unterschied der Geschwindigkeit von verschiedenen Flüssigkeitsschichten abhängen muß. Er enthält demnach die Ableitungen der Geschwindigkeiten nach den Koordinaten, da dies der Newtonschen Reibung entspricht. In

linearer Näherung wird davon ausgegangen, daß nur die ersten Ableitungen der Geschwindigkeiten eine Rolle spielen und der Spannungstensor auch nur linear von diesen Ableitungen abhängt. Damit erhalten wir folgende allgemeine Form (unter Beachtung der Einsteinschen Summenkonvention)

$$\sigma_{ik} = a\frac{\partial u_i}{\partial x_k} + b\frac{\partial u_k}{\partial x_i} + c\frac{\partial u_l}{\partial x_l}\delta_{ik} \tag{1.23}$$

Bei der genaueren Bestimmung der Koeffizienten müssen zwei weitere Bedingungen beachtet werden

- In einer gleichförmigen Strömung ($\mathbf{u}$=const) tritt keine Reibung auf
- In gleichmäßig rotierenden Strömungen tritt ebenfalls keine Reibung auf

Aus diesen Bedingungen folgt für Gl. 1.23 $a = b$, so daß zwei Konstanten übrig bleiben. Die übliche Schreibweise für den Spannungstensor ist

$$\sigma_{ik} = \eta\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i} - \frac{2}{3}\delta_{ik}\frac{\partial u_l}{\partial x_l}\right) + \zeta\delta_{ik}\frac{\partial u_l}{\partial x_l} \tag{1.24}$$

Der Vorteil dieser Formulierung ist, daß die Spur des Ausdrucks in den Klammern verschwindet. Die Größen η und ζ heißen Zähigkeitskoeffizienten, es sind materialabhängige Konstanten.

Der Ausdruck für den Spannungstensor vereinfacht sich weiter, wenn die Flüssigkeiten inkompressibel sind

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{1.25}$$

$$\Rightarrow \sigma_{ik} = \eta\left(\frac{\partial u_i}{\partial x_k} + \frac{\partial u_k}{\partial x_i}\right) \tag{1.26}$$

$$\frac{\partial \sigma_{ik}}{\partial x_k} = \eta\frac{\partial^2 u_i}{\partial x_k^2} \tag{1.27}$$

Mit dem daraus resultierendem Spannungstensor ergibt sich nun die *Navier-Stokes-Gleichung* :

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\nabla)\mathbf{u} + \nabla p = \nu\Delta\mathbf{u} \tag{1.28}$$

Hierbei ist ν die Viskosität, für die $\nu = \frac{\eta}{\rho}$ gilt. Um eine Gleichung für den Druck zu erhalten, muß die Divergenz der Gleichung gebildet und die Inkompressibilität noch einmal angesetzt werden

$$\Delta p = -(\mathbf{u}\nabla)\mathbf{u} + \nu\Delta\mathbf{u} \tag{1.29}$$

Unter Ausnutzung der Bedingung $\nabla \cdot \mathbf{u} = 0$ kann auch die Rotation der Gleichung gebildet werden, um eine DGL für die Vortizität $\omega = \nabla \times \mathbf{u}$ zu erhalten.

$$\frac{\partial}{\partial t}(\nabla \times \mathbf{u}) - \nabla \times (\mathbf{u} \times (\nabla \times \mathbf{u})) = -\nu \nabla \times (\nabla \times (\nabla \times \mathbf{u})) \tag{1.30}$$

$$\Rightarrow \frac{\partial \omega}{\partial t} = (\omega \nabla)\mathbf{u} + \nu \Delta \omega \tag{1.31}$$

Besonderer Vorteil dieser Gleichung ist, daß ω für den zweidimensionalen Fall ein Quasiskalar ist. In dieser Gleichung ist der Druck eliminiert, allerdings kann die Geschwindigkeit nur über die Stromfunktion Ψ bestimmt werden.

$$\omega = \nabla \times \mathbf{u} \tag{1.32}$$

$$\mathbf{u} = \nabla \times \mathbf{\Psi} \tag{1.33}$$

$$\Delta \mathbf{\Psi} = -\omega \tag{1.34}$$

1.1.4. Selbstähnlichkeit

Um die Zahl der Parameter in der Navier-Stokes-Gleichung zu verringern, wird der Ansatz der Normierung verfolgt. Die auftauchenden Größen werden dabei auf charakteristische Werte normiert, die sich dabei unter Umständen eliminieren. Im konkreten Fall werden charakteristische Geschwindigkeit, Zeit, Länge und Druck definiert (normierte Größe werden mit Dach gekennzeichnet, charakteristische Größen durch Großbuchstaben). Es ist dabei wichtig zu beachten, daß die Differentialoperatoren ebenfalls normiert werden.

$$\mathbf{u} = \hat{\mathbf{u}} U \tag{1.35}$$

$$l = \hat{l} L \tag{1.36}$$

$$t = \hat{t} T = \hat{t} \frac{L}{U} \tag{1.37}$$

$$p = \hat{p} P = \hat{p} \frac{1}{U^2} \tag{1.38}$$

$$\partial_t = \hat{\partial}_t \frac{1}{T} \tag{1.39}$$

$$\nabla = \hat{\nabla} \frac{1}{L} \tag{1.40}$$

Mit dieser Normierung läßt sich dann die Navier-Stokes-Gleichung umschreiben zu

$$\frac{U}{T}\hat{\partial}_t \hat{\mathbf{u}} + \frac{U^2}{L}\hat{\mathbf{u}}\hat{\nabla}\hat{\mathbf{u}} + \frac{1}{LP}\hat{\nabla}\hat{p} = \nu \frac{U}{L^2}\hat{\Delta}\hat{\mathbf{u}} \tag{1.41}$$

$$\hat{\partial}_t \hat{\mathbf{u}} + \hat{\mathbf{u}}\hat{\nabla}\hat{\mathbf{u}} + \hat{\nabla}\hat{p} = \frac{\nu}{UL}\hat{\Delta}\hat{\mathbf{u}} \tag{1.42}$$

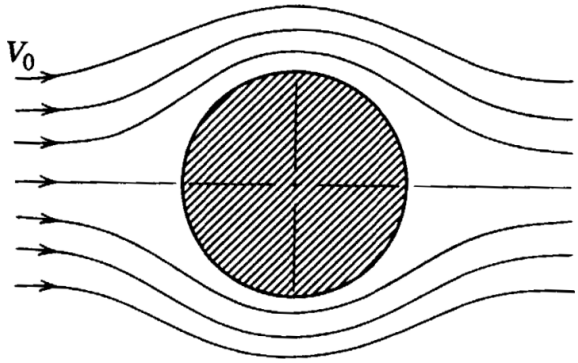


Abbildung 1.1.: *Umströmung eines Zylinders in nicht-viskoser Flüssigkeit (aus [Pao67])*

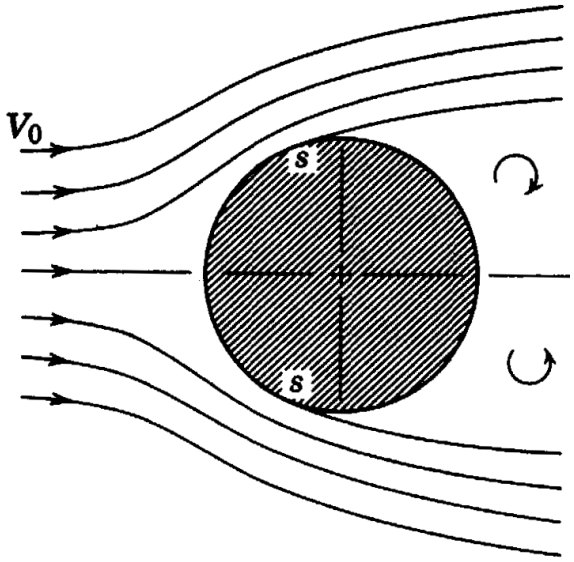


Abbildung 1.2.: *Laminare Umströmung mit Ablösepunkten (S)*

Auf der rechten Seite bleibt damit nur noch ein Parameter übrig, der als *Reynolds-Zahl* bezeichnet wird.

$$Re = \frac{UL}{\nu} \tag{1.43}$$

Die Tatsache, daß nur ein einziger, dimensionsloser Parameter für Strömungen existiert, bedeutet, daß alle Strömungen, mit derselben Reynolds-Zahl selbstähnlich sind.

1.2. Grenzschichten

Bei der Betrachtung umströmter Körper spielen Grenzschichten eine entscheidende Rolle. Sie stellen auch das Hauptproplem bei der numerischen Simualtion dar. Ich möchte an dieser Stelle kurz die beiden Grenzfälle für Grenzschichten bei der Umströmung eines Kreiszylinders vorstellen, die laminare und die turbulente Umströmung.

1.2.1. Laminare Grenzschicht

Am Rand eines umströmten Zylinders ist die Geschwindigkeit der Flüssigkeit Null (bzw. nimmt einen konstanten Wert für rotierende Zylinder an). Es muß also eine Schicht geben, in der die Geschwindigkeit von ihrem Mittelwert fern des Körpers auf Null abfällt, nämlich die Grenzschicht. Der Abfall der Geschwindigkeit wird von der Zähigkeit der Flüssigkeit bestimmt. Die Vorgänge in der Grenzschicht hängen also mit der Reynolds-Zahl zusammen.

Für kleine Reynolds-Zahlen liegt eine laminare Grenzschicht vor, deren Dicke Prandtl [Pra52] abgeschätzt hat

$$\delta \sim \frac{l}{\sqrt{Re}} \tag{1.44}$$

Für die numerische Simulation von Strömungen bedeutet dies erst einmal, daß Strömungen mit kleinen Reynolds-Zahlen keine großen Probleme darstellen, denn die Gradienten der Geschwindigkeit sind sehr klein.

In idealen Flüssigkeiten existiert keine Grenzschicht, sondern das Fluid schlüpft am Zylinder vorbei (Abb. 1.1) und bildet lediglich sogenannte Stagnationspunkte aus. Dagegen stellt in zähen Flüssigkeiten die Grenzschichtausbildung und -ablösung das wichtigste Phänomen dar. In den Ablösepunkten (s. Abb. 1.2) reißt die Grenzschicht ab und bildet hinter dem Zylinder ein Totwasser aus.

1.2.2. Turbulente Grenzschicht

Mit steigender Reynoldszahl wandern die Ablösepunkte nach hinten, ab einem gewissen Punkt lösen sich dann Wirbel ab (s. Abb. 1.3), aus denen sich auch eine Karmansche Wirbelstraße bildet (die abwechselnden Wirbel entstehen durch eine Hopf-Bifurkation). Erhöht man die Reynolds-Zahl weiter, bildet sich eine turbulente Strömung aus, die den universellen Gesetzen für die Turbulenz gehorcht (z.B. Energiekaskade und Skalenverhalten).

An der umströmten Fläche stellt sich eine sehr dünne laminare Grenzschicht ein, an die sich eine turbulente Grenzschicht anschließt. In der laminaren Grenzschicht ist der Gradient der Geschwindigkeit sehr groß, so daß bei Simulationen hierauf ein besonderes Augenmerk gelegt werden muß.

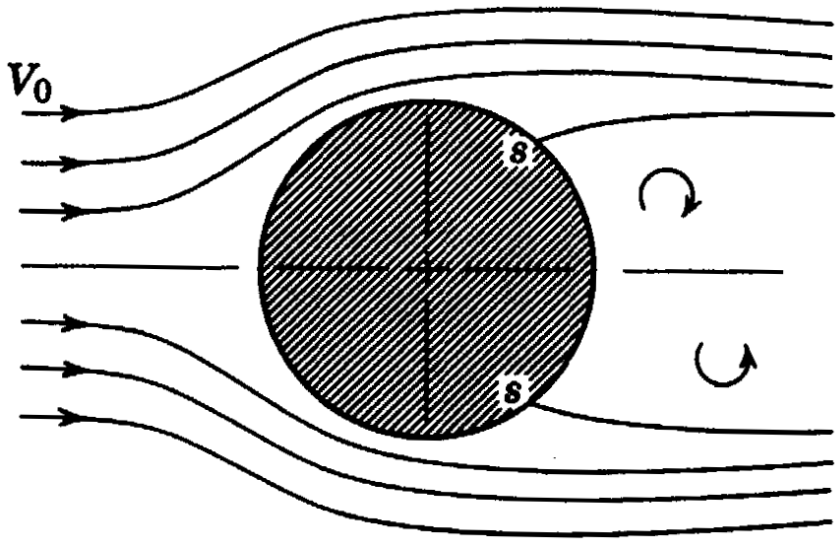


Abbildung 1.3.: *Turbulente Umströmung mit Ablösepunkten*

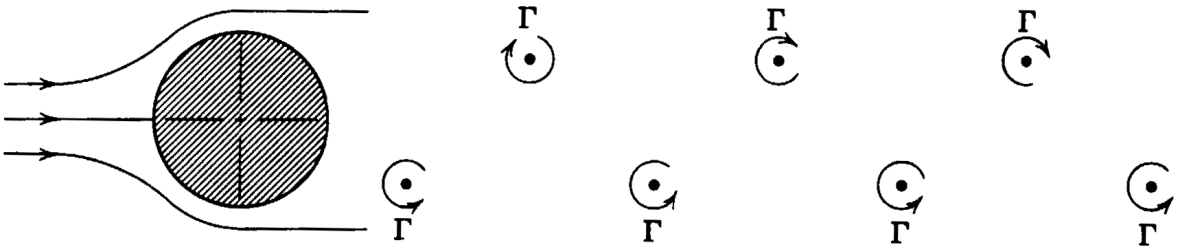


Abbildung 1.4.: *Ausbildung einer Karmanschen Wirbelstraße*

2. Numerische Methoden

2.1. Grundlagen der Numerik

Auch wenn numerische Simulationen in den letzten Jahrzehnten eine entscheidende Rolle in der Fluiddynamik übernommen haben, ermöglichen sie es trotzdem nicht, die Navier-Stokes-Gleichung exakt zu lösen. Alle benutzten Verfahren haben Vor- und Nachteile, die gegeneinander abgewogen werden müssen. Ich möchte hier die zentralen Probleme schildern, die häufig bei numerischen Verfahren auftreten.

Zuvor werden kurz einige Begriffe geklärt, die in diesem Abschnitt verwendet werden.

u^n	num. Lösung der DGL zur Zeit $t = n\Delta t$
u_e^n	exakte Lösung der DGL zur Zeit $t = n\Delta t$
L	Operator des numerischen Verfahrens

Der Operator bezeichnet dabei das Verfahren, mit dem die DGL diskretisiert wird. Dabei gilt dann

$$u^{n+1} = Lu^n \tag{2.1}$$

2.1.1. Stabilität

Stabilität eines numerischen Verfahrens L bedeutet, daß $\|Lu_e^n - Lu^n\|$ durch $\|u_e^n - u^n\|$ abgeschätzt werden kann, die numerische Lösung also nicht "explodiert". Aus dieser Bedingung für die Stabilität ergibt sich dann das *CFL-Kriterium* (**C**ourant-**F**riedrich-**L**evy).

Dieses Kriterium muß für jedes Verfahren einzeln berechnet werden und hat für hyperbolische DGL [1] im allgemeinen die Form

$$u\frac{\Delta t}{\Delta x} \leq c \tag{2.2}$$

wobei c eine Konstante ist, die vom jeweiligen Verfahren abhängt. Das CFL-Kriterium gibt eine klare Beschränkung für die Wahl von Schrittweite und Zeitintervall vor. Werden diese Beschränkungen missachtet, wird die Lösung instabil und entfernt sich rasch von der physikalischen Lösung.

Im Fall parabolischer DGL, wie sie im Zusammenhang mit der Dissipation $\nu\Delta u$ auftreten, lautet das CFL-Kriterium

$$\nu\frac{\Delta t}{(\Delta x)^2} \leq c \tag{2.3}$$

Dabei bezeichnet ν ist Viskosität.

2.1.2. Genauigkeit

Neben der Stabilität spielt die Genauigkeit eine entscheidende Rolle für das Verfahren. Die Genauigkeit wird definiert durch

$$\begin{aligned} \|u^n - u_e^n\| &= \epsilon(\Delta x, \Delta t) && (2.4)\\ &= O((\Delta x)^p, (\Delta t)^q) && (2.5) \end{aligned}$$

Die Ordnung des Verfahrens wird durch die Exponenten p und q bestimmt.

Ungenauigkeiten führen zu Phasen- und Amplitudenfehlern. Gerade Potenzen von p entsprechen einem Dissipationsterm und führen somit zur Vernichtung von Energie, wohingen ungerade Potenzen zu Phasenverschiebungen führen (was aus der Darstellung in Fouriermoden folgt).

Neben der Genauigkeit wird auch noch der lokale Fehler (LTE) durch Gl. 2.6 definiert.

$$\begin{aligned} LTE &= u_e^{n+1} - Lu_e^n && (2.6)\\ &= O\left(\Delta t \sum \Delta t^p \Delta x^q\right) && (2.7) \end{aligned}$$

[1] DGL in der Form $\partial_t u = \partial_x f(u)$

Wenn der lokale Fehler eines Verfahrens sich wie in Gl. 2.7 schreiben läßt und somit für $\Delta x, \Delta t \to 0$ gegen Null geht, bezeichnet man das Verfahren als *konsistent*.

Nach dem Lax-Theorem folgt aus Stabilität und Konsistenz die Konvergenz des Verfahrens. Konvergenz bedeutet, daß bei unendlich kleinen Zeit- und Raumeinteilungen die Ergebnisse der Simulation gegen die exakte Lösung konvergiert. Dies ist nicht gleichbedeutend mit der Konvergenz des lokalen Fehlers. Das Lax-Theorem macht also eine Aussage über die globale Konvergenz eines Verfahrens.

2.1.3. Flußdarstellung

Die Euler-Gleichung ist ein Beispiel für hyperbolische Erhaltungssätze, die sich in der sogenannten Flußform schreiben lassen

$$\partial_t u = \partial_x f(u) \tag{2.8}$$

$$\Rightarrow \frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{F_{j+1/2}^n - F_{j-1/2}^n}{\Delta x} \tag{2.9}$$

F ist hier der numerische Fluß an den Zellenrändern, der erhalten wird. Daher muß das verwendete numerische Verfahren den Fluß ebenfalls erhalten.

Der Fluß für die Euler-Gleichung läßt sich bestimmen zu

$$F = (-\mathbf{uu} - p) \tag{2.10}$$

2.1.4. Schocks

Starke Gradienten oder Schocks in der Strömung stellen besondere Anforderungen an numerische Verfahren, da sich normalerweise an diesen Schocks Oszillationen ausbilden (Gibbs' Phänomen). Dieses Phänomen tritt deutlich bei der Verwendung spektraler Verfahren (also Verfahren, die auf Fourier-Moden basieren) hervor, bei denen die Moden nur sehr langsam gegen rechteckige Schocks konvergieren. Ein derartiges Verhalten ist leicht auf Differenzenschemata übertragbar.

Viele Verfahren, die diese Oszillationen unterdrücken, haben gleichzeitig eine sehr starke Dämpfung, die genauso unerwünscht ist wie die Oszillationen. Das zu verwendende Verfahren sollte also an Schocks keine Oszillationen ausbilden und trotzdem sehr wenig Dämpfung besitzen.

2.2. Adaptive Gitterverfeinerung

Bei der numerischen Lösung von DGL tritt häufig das Problem auf, daß Gebiete existieren, die eine feine Auflösung erforden, während das restliche Gebiet mit einer geringeren Auflösung behandelt werden kann. Das kann dazu führen, daß mit Festgitterverfahren die Auflösung des gesamten Gitters durch wenige Punkte bestimmt wird. Ein Ausweg aus dieser Schwierigkeit ist die Verwendung adaptiver Gitter.

Für diese Simulation wird in dieser Arbeit der Code von Grauer, Friedel und Marliani ([FGM97]) verwendet, der auf einer Idee von Berger und Colella ([BC89]) basiert. Diese Umsetzung der Adaptiven Gitterverfeinerung (**A**daptive **M**esh **R**efinement, AMR) benutzt feinere Gitter, die an beliebiger Stelle in beliebiger Größe, aber mit festem Verfeinerungsfaktor Punkte mit hohem Auflösungsbedarf überdecken. Ein anderes Konzept ist die Viertelung (im dreidimensionalen: Achtelung) von kritischen Gebieten, daß hier nicht verwendet wird, allerdings in neueren Codes wie *RACOON II* angewandt wird, da es Vorteile für die Parallelisierung mit sich bringt.

In diesem Abschnitt werden die grundlegend Konzepte des AMR-Verfahrens dargelegt, eine genauere Beschreibung des Programms findet sich in Anhang A, die Implementation von AMR ist in Kapitel 4 dargestellt.

2.2.1. Datenstrukturen

Die grundlegende Struktur des AMR-Codes ist eine Matrix, in der Daten gespeichert werden. Die Matrizen verschiedener Größen werden wiederum zusammen in einem Grid gespeichert. Das Grid ist das Gitter, auf dem alle Größen existieren. Die nächste Stufe in der Objekthierarchie ist das Level. Ein Level ist die Verfeinerungsstufe und enthält alle Grids, die in dieser Verfeinerungsstufe existieren. Dabei ist zu beachten, daß ein Gitter auf der höchsten Verfeinerungsstufe von Gittern aller niedrigeren Verfeinerungsstufe überdeckt werden muß.

2.2.2. Gitterverfeinerung

Der erste Schritt der Gitterverfeinerung ist die Suche nach Punkten, die “kritisch” sind. Hierfür müssen Kriterien bestimmt werden, die dem jeweiligen Problem entsprechen.

Wenn die kritischen Punkte bestimmt sind, werden als nächstes Rechtecke ermittelt, die alle kritischen Punkte umfassen, ohne dabei mehr unkritische Punkte zu überdecken als nötig.

2.2.3. Programmablauf

Der prinzipielle Ablauf des Programm gliedert sich wie folgt

- Initialisiere Gitter auf dem obersten Level
- Suche kritische Punkte
- Bilde Rechtecke, die alle kritischen Punkte umfassen
- Initialisiere die tieferliegenden Level
- Führe den Zeitschritt auf dem obersten Level aus
- Gehe zum nächsten Level
- Führe den Zeitschritt auf allen Gittern des Levels aus
- Führe die letzten beide Schritte solange aus, bis man das tiefste Level erreicht ist, dort n-mal den Zeitschritt ausführen und dann wieder die Level Schritt für Schritt hoch gehen

Das größte Problem stellt der Austausch von Daten zwischen den Leveln dar. Durch die schrittweise Integration gibt es unterschiedliche aktuelle Zeiten auf den Leveln. Um die Integration korrekt durchzuführen benötigt man deshalb auf den tieferen Leveln die Randwerte der höheren Level und nach Abschluß der Integration Daten der tieferen Level auf den höheren Leveln. Die Matrizen in den Grids enthalten für die Aktualisierung des Randes einen Bereich konstanter Breite, der nur durch die boundary-Routinen, also reine Randaktualisierungsroutinen beeinflußt wird.

Bei der Integration der tieferen Level werden die Randwerte in zwei Schritten [2] übergeben. Während auf dem groben Gitter der Zeitschritt $t \rightarrow t + \Delta t$ stattfindet, sind es auf

[2] unter Annahme des Verfeinerungsfaktors 2

dem feineren Gitter die Schritte $t \to t + \frac{1}{2}\Delta t \to t + \Delta t$. Um die korrekten Randbedingungen für das feinere Gitter zu ermitteln, wird nach der Integration des groben Gitters die Differenz $u^{n+1} - u^n$ gebildet. Anschließend wird nach jedem Teilschritt auf dem feinen Gitter ein Bruchteil dieser Differenz auf den Rand addiert, in diesem Fall die Hälfte.

Der Weg vom feinen zum groben Gitter ist erheblich einfacher: nach dem vollständigen Zeitschritt auf dem feinen Gitter werden die Daten in das grobe Gitter übertragen.

Ein weiterer Punkt der bei der Betrachtung der Ränder eine Rolle spielt, ist die Lösung von Poisson-Gleichungen auf tieferen Leveln. Während auf dem höchstem Level die Randbedingungen für die Poisson-Gleichung physikalisch vorgegeben sind, müssen auf tieferen Leveln Randbedingungen gegeben sein, die die Kontinuität der Lösung sicherstellen.

2.3. CWENO

Viele der im vorherigen Abschnitt angesprochenen Probleme lassen sich mit CWENO (**C**entral **w**eighted **e**ssential **n**on**o**sscillatory)-Schemata lösen, die zuerst von Kurganov und Levy beschrieben wurde ([KL00]). Das CWENO-Verfahren ist ein Lösungsverfahren für hyperbolische DGL mit Genauigkeit dritter Ordnung in glatten Gebieten und gesonderter Behandlung von Schocks.

2.3.1. Aufbau des Verfahrens

Ausgehend von einer hyperbolischen DGL

$$\frac{\partial}{\partial t}u(x,t) + \frac{\partial}{\partial x}f(u(x,t)) = 0 \tag{2.11}$$

werden Mittelwerte über die Zellen definiert und in Flußdarstellung gebracht. Daraus folgt

$$\bar{u}(x,t) = \frac{1}{\Delta x}\int_{x-1/2}^{x+1/2} u(\xi,t)\ d\xi \tag{2.12}$$

$$\bar{u}_{j+1/2}^{n+1} = \bar{u}_{j+1/2}^{n} + \int_{t}^{t+1} \left(f(u_{j+1},\tau) - f(u_j,\tau)\right)\ d\tau \tag{2.13}$$

Im nächsten Schritt wird mit Hilfe der Zellmittelwerte die Funktion u polynomisch in den Zellen rekonstruiert. Mit Hilfe der Polynomen lassen sich dann Werte an beliebigen Stellen in der Zelle bestimmen. Bei der Bestimmung der Polynome werden nicht nur die

Zellen selber beachtet, sondern auch die jeweilige linke und rechte Nachbarzelle

$$P_j(x) = w_L P_L(x) + w_C P_C(x) + w_R P_R(x) \tag{2.14}$$

P_C ist ein quadratisches Polynom, das den zentralen Zellmittelwert erhält, wohingegen die beiden anderen Polynome linear sind und den jeweils linken oder rechten Mittelwert erhalten.

$$\begin{aligned} P_L(x) &= \bar{u}_j^n + \frac{\bar{u}_{j+1}^n - \bar{u}_j^n}{\Delta x}(x_j - x) && (2.15)\\ P_R(x) &= \bar{u}_j^n + \frac{\bar{u}_j^n - \bar{u}_{j-1}^n}{\Delta x}(x_j - x) && (2.16)\\ P_C(x) &= \bar{u}_j^n + \frac{1}{12}(\bar{u}_{j+1}^n - 2\bar{u}_j^n + \bar{u}_{j-1}^n) \\ &\quad + \frac{\bar{u}_{j+1}^n - \bar{u}_{j-1}^n}{2\Delta x}(x_j - x) + \frac{\bar{u}_{j+1}^n - 2\bar{u}_j^n + \bar{u}_{j-1}^n}{(\Delta x)^2}(x_j - x)^2 && (2.17) \end{aligned}$$

Die Parabel im Zentrum ist eindeutig so gewählt, daß sie folgende Gleichung erfüllt

$$P_{\text{Exakt}}(x) = c_R P_R(x) + c_C P_C(x) + c_L P_L(x) \tag{2.18}$$

P_{Exakt} ist die eindeutige Parabel, die alle drei Zellmittelwerte erhält. Sie ist gegeben durch

$$P_{\text{Exakt}}(x) = u_j^n + u_j'(x - x_j) + \frac{1}{2}u_j''(x - x_j)^2 \tag{2.19}$$

Die Werte der Konstanten c wurden als $c_L = c_R = 1/4$ und $c_C = 1/2$ gewählt. Kurganov und Levy zeigen, daß ohne Verlust an Genauigkeit auch jede andere symmetrische Kombination von c_i mit $\sum_i c_i = 1$ gewählt werden kann. Aus c_i berechnen sich später die Gewichte w_i für das CWENO-Verfahren.

Bei der Rekonstruktion ist zu beachten, daß diese so nur im eindimensionalen Fall gilt. Im mehrdimensionalen Fall muß in P_C der mehrdimensionale Laplace-Operator eingesetzt werden.

Der nächste Schritt im CWENO-Verfahren ist nun die Bestimmung der Gewichtung w_i (mit $i \in \{L, C, R\}$). Dabei sollen diese so gewählt werden, daß in glatten Gebieten die maximale Genauigkeit erzielt wird. An Schocks allerdings soll die ideale einseitige Ableitung gewählt werden (um Oszillationen zu vermeiden). Die Gewichte sind dabei folgendermaßen definiert

$$\begin{aligned} w_i &= \frac{\alpha_i}{\sum_m \alpha_m} && (2.20)\\ \alpha_i &= \frac{c_i}{(IS_i + \epsilon)^p} && (2.21) \end{aligned}$$

Die Konstante ϵ verhindert, daß der Nenner Null wird und hat hier den Wert 10^{-6} (der Einfluß von ϵ ist zu vernachlässigen). Bei der Wahl von ϵ muß beachtet werden, daß die Konstante klein gegen Werte von IS sind. p wurde von Kurganov und Levy als 2 gewählt, mit diesem Parameter läßt sich zwischen hoher Genauigkeit und guter Schockauflösung abwägen. IS_i ist ein "Smoothness"-Indikator, der Schocks sucht und folgendermaßen definiert ist

$$IS_i = \sum_{l=1}^{2} \int_{x_j-1/2}^{x_j+1/2} (\Delta x)^{2l-1} (P_i^{(l)}(x))^2 dx \tag{2.22}$$

$$\Rightarrow IS_L = (\bar{u}_j^n - \bar{u}_{j-1}^n)^2 \tag{2.23}$$

$$IS_R = (\bar{u}_j^n - \bar{u}_{j+1}^n)^2 \tag{2.24}$$

$$IS_C = \frac{13}{3}(\bar{u}_{j+1}^n - 2\bar{u}_j^n + \bar{u}_{j-1}^n)^2 + \frac{1}{4}(u_{j+1}^n - u_{j-1}^n)^2 \tag{2.25}$$

Die Einführung folgender Abkürzungen bietet sich sinvollerweise an

$$P_j(x) = A_j + B_j(x - x_j) + \frac{1}{2}C_j(x - x_j)^2 \tag{2.26}$$

$$A_j = \bar{u}_j^n - \frac{w_c}{12}\left(\bar{u}_{j+1}^n - 2\bar{u}_j^n + \bar{u}_{j-1}^n\right) \tag{2.27}$$

$$B_j = \frac{1}{\Delta x}\left(w_r(\bar{u}_{j+1}^n - \bar{u}_j^n) + w_c\frac{\bar{u}_{j+1}^n - \bar{u}_{j-1}^n}{2} + w_l(\bar{u}_j^n - \bar{u}_{j-1}^n)\right) \tag{2.28}$$

$$C_j = 2w_c\frac{\bar{u}_{j+1}^n - 2\bar{u}_j^n + \bar{u}_{j-1}^n}{(\Delta x)^2} \tag{2.29}$$

Ausgehende von diesen Überlegungen zur Rekonstruktion in den Gitterzellen läßt sich jetzt der Zeitschritt entwickeln. Das grundsätzliche Vorgehen ist in Abb. 2.1 dargestellt.

An dieser Stelle wird zwischen dem voll-diskreten Schema, in dem die Zeitableitung diskretisiert wird und dem semi-diskreten Schema, in dem nur die Ortsableitung diskretisiert wird,unterschieden.

Im voll-diskreten Schema werden in jeder Zelle die Ausbreitungsgeschwindigkei-

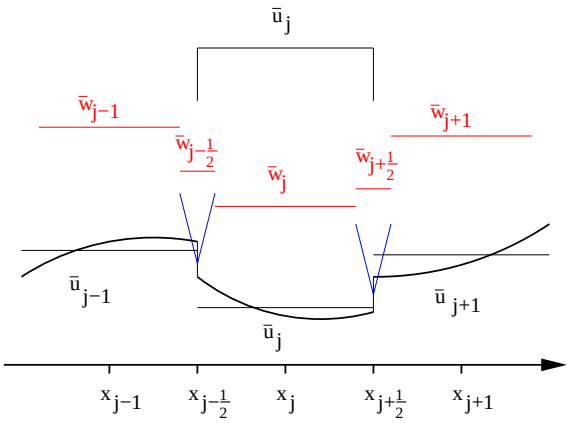


Abbildung 2.1.: *Schemazeichnung des CWENO-Verfahrens*

ten der Diskontinuitäten ermittelt

$$a^n_{j+1/2} = max\left(\frac{\partial f}{\partial u}(u^-_{j+1/2}), \frac{\partial f}{\partial u}(u^+_{j+1/2})\right) \tag{2.30}$$

Dementsprechend werden Gebiete bestimmt, in denen sich die Diskontinuität ausbreitet sowie Gebiete, die eine glatte Lösung besitzen. ("+" bezieht sich hier auf die rechte Seite der Diskontinuität, "−" auf die linke). Zur exakten Berechnung der w_j möchte ich hier auf Kurganov und Levy verweisen und stattdessen direkt auf das semi-diskrete Schema zu sprechen kommen, das hier tatsächlich zur Anwendung kommt.

Im semi-diskreten Grenzfall geht der Zeitschritt gegen Null, womit die Ausbreitungsradien der Diskontinuitäten ebenfalls gegen Null konvergieren. Mit diesem Limes lassen sich die Werte von u aus der Rekonstruktion links- und rechtsseitig angeben, so daß damit auch der Fluß bestimmt werden kann.

$$u^+_{j+1/2}(t) = A_{j+1} - \frac{\Delta x}{2}B_{j+1} + \frac{(\Delta x)^2}{8}C_{j+1} \tag{2.31}$$

$$u^-_{j+1/2}(t) = A_j + \frac{\Delta x}{2}B_j + \frac{(\Delta x)^2}{8}C_j \tag{2.32}$$

$$\begin{aligned}\Rightarrow \frac{d\bar{u}_j}{dt} &= -\frac{1}{2\Delta x}\left(f(u^+_{j+1/2}(t)) + f(u^-_{j+1/2}(t)) - f(u^+_{j-1/2}(t)) - f(u^-_{j-1/2}(t))\right) \\ &\quad + \frac{a^+_{j+1/2}}{2\Delta x}\left(u^+_{j+1/2}(t) - u^-_{j+1/2}(t)\right) - \frac{a^+_{j-1/2}}{2\Delta x}\left(u^+_{j-1/2}(t) - u^-_{j-1/2}(t)\right)\end{aligned} \tag{2.33}$$

Man kann leicht erkennen, daß der so gebildete Ausdruck in Flußform gebracht werden kann.

2.3.2. Vorteile des Verfahrens

Wie bereits oben beschrieben, besteht der zentrale Vorteil des CWENO-Verfahrens gegenüber klassischen Differenzschemata darin, daß CWENO an Schocks nicht zu Oszillationen neigt und trotzdem in glatten Gebieten eine Genauigkeit dritter Ordnung besitzt. Dadurch zeigt CWENO auch weniger numerische Dissipation als andere Verfahren, die an Schocks limitieren. Außerdem wird kein Riemann-Löser benötigt, der für jeden Punkt die Jacobi-Determinante berechnet und daraus die Eigenvektoren bestimmt. Dadurch wird das Verfahren wesentlich weniger aufwendig als andere Ansätze.

2.4. Runge-Kutta

Zur expliziten Berechnung des Zeitschritts hat sich schon lange die Runge-Kutta-Methode durchgesetzt, die sich aus dem Taylorschen Satz ergibt. Häufig wird das Verfahren 4. Ordnung eingesetzt. Hier wird aber die 3. Ordnung verwendet, da sie nur ein Hilfsfeld pro Variable benötigt und somit den Speicherplatzbedarf deutlich senkt und dabei immer noch eine hinreichende Genauigkeit liefert.

Da das Runge-Kutta-Verfahren mehrschrittig ist, muß bei der Verwendung der adaptiven Gitterverfeinerung berücksichtigt werden, daß es weitere Zwischenzeiten gibt, für die der Rand der feineren Gitter aktualisiert werden muß (näheres in Anhang B).

Runge-Kutta 3. Ordnung Das Verfahren 3. Ordnung ist in drei Teilschritte gegliedert, die jeweils auf vorher berechnete Zwischenwerte zurückgreifen. Die genaue Implementation wird im Anhang erläutert.

$$\begin{aligned} \partial_t u &= f(u) && (2.34)\\ u_1 &= u(t) + \Delta t \cdot f(u(t)) && (2.35)\\ u_2 &= \frac{3}{4}u(t) + \frac{1}{4}u_1 + \frac{1}{4}f(u_1) && (2.36)\\ u_3 &= \frac{1}{3}u(t) + \frac{2}{3}u_2 + \frac{2}{3}f(u_2) && (2.37)\\ u(t+\Delta t) &= u_3 && (2.38) \end{aligned}$$

2.5. Lösungsansätze für die Navier-Stokes-Gleichung

Bei den in dieser Arbeit simulierten Strömungen stellt die Inkompressibilität eines der wesentlichen Probleme dar. Es gibt grundsätzlich zwei Lösungsansätze, von dem sich der erste an Gleichung 1.31 orientiert und somit per se divergenzfrei durch Rotationsbildung ist (Methoden RK und Mix, s.u.). Der zweite Ansatz ist der Projektionsansatz, bei dem die Lösung der Navier-Stokes-Gleichung auf ihren divergenzfreien Anteil projeziert wird (Methoden PM II und PM III , s.u.).

Es gibt zwei grundsätzliche Anforderungen an die hier vorgestellten Verfahren:

1. Die Navier-Stokes-Gleichung muß in der Geschwindigkeit (nicht in der Vortizität)

gelöst werden, damit später die Randbedingungen für komplexe Geometrien eingesetzt werden können.

2. Das Verfahren muß mit der adaptiven Gitterverfeinerung kompatibel sein.

Aus dem erstem Grund kann der RK-Code nur als Vergleichsmethode verwendet werden. Da der vorgestelle PM II - und PM III -Code sich nicht in AMR implementieren ließ, mußte ein neues Verfahren entwickelt werden, das beide Anforderungen erfüllt.

2.5.1. Methode RK

Der *RK*-Code ist ein "klassischer" Wirbelstärke-Stromfunktions-Code, dessen Ergebnisse mit denen anderer Methoden verglichen werden. Aus Gleichung 1.31 lassen sich untenstehende Einzelschritte isolieren

$$\mathbf{u}^{n+1} = \nabla\times\Psi^n \tag{2.39}$$

$$\omega^{n+1} = \omega^n - \Delta t\left(\mathbf{u}^{n+1}\nabla\omega^{n+1} + \nu\Delta\omega^n\right) \tag{2.40}$$

$$\Delta\Psi^{n+1} = -\omega^{n+1} \tag{2.41}$$

Bei der Berechnung der rechten Seite der DGL kommt hier im Gegensatz zum klassischen Code das CWENO-Verfahren und für den Zeitschritt das Runge-Kutta-Verfahren zur Anwendung. Mit dem CWENO-Verfahren wird die Vortizität rekonstruiert und in der Nichtlinearität die aus der Stromfunktion berechnete Geschwindigkeit eingesetzt.

2.5.2. Methode PM II / PM III

In [BCM01] stellen Brown, Cortez und Minion ein Projektionsverfahren zweiter Ordnung vor (ähnlich [BCG89]), das hier PM II genannt wird. Die Herleitung dieses Verfahrens basiert auf dem Gedanken der Diskretisierung und Divergenzbildung der Navier-Stokes-Gleichung (Gl. 1.28), wobei ein Teilschritt mit einem Druck aus dem vergangenen Zeitschritt durchführt wird

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -\left((\mathbf{u}\nabla)\mathbf{u}\right)^{n+1/2} - \nabla p^{n-1/2} + \nu\Delta\mathbf{u}^n \tag{2.42}$$

$\mathbf{u}^*$ ist dabei eine nicht-divergenzfreie Lösung. In PM II wird zusätzlich ein Crank-Nicholson-Schritt (numerische Methode zur Stabilisierung, bei der zwischen alten und neuen Werten

gemittelt wird) für den Dissipationsterm eingebaut, um das numerische Verfahren zu stabilisieren. Dies ist notwendig, da die CFL-Bedingung für elliptische Gleichung restriktiver ist als für hyperbolische Gleichungen.

$$\nu\Delta\mathbf{u}^n \rightarrow \frac{1}{2}\nu\Delta\left(\mathbf{u}^n + \mathbf{u}^*\right) \tag{2.43}$$

Damit sind dann folgende Gleichungen zu lösen, die sich aus der Navier-Stokes-Gleichung ergeben, indem der Crank-Nicholson-Schritt eingesetzt wird

$$\begin{aligned} \mathbf{u}^{**} &= \mathbf{u}^n - \Delta t\left((\mathbf{u}\nabla)\mathbf{u} + \nabla p\right) && (2.44)\\ \left(1 - \frac{\nu\Delta t}{2}\Delta\right)\mathbf{u}^* &= \left(1 + \frac{\nu\Delta t}{2}\Delta\right)\mathbf{u}^{**} && (2.45) \end{aligned}$$

Um nun $\mathbf{u}^*$ auf seinen divergenzfreien Teil zu projezieren, wird ein Hilfsfeld ϕ berechnet, das folgende Bedingung erfüllen soll

$$\begin{aligned} \nabla\cdot\left(\mathbf{u}^* - \Delta t\nabla\phi\right) &= 0 && (2.46)\\ \Delta\phi &= \frac{\nabla\cdot\mathbf{u}^*}{\Delta t} && (2.47)\\ \mathbf{u}^{n+1} &= \mathbf{u}^* - \Delta t\nabla\phi^{n+1} && (2.48) \end{aligned}$$

Der letzte verbliebene Schritt ist nun noch der Zeitschritt für den Druck. Dazu wird ausgehend von der letzten Gleichung die Divergenz gebildet

$$p^{n+1/2} = p^{n-1/2} + \phi^{n+1} - \frac{\nu\Delta t}{2}\Delta\phi^{n+1} \tag{2.49}$$

Damit ergibt sich insgesamt folgendes Verfahren

$$\begin{aligned} \frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + \nabla p^{n+1/2} &= -[(u\cdot\nabla)\mathbf{u}]^{n+1/2} + \frac{\nu}{2}\nabla^2(\mathbf{u}^n + \mathbf{u}^*) && (2.50)\\ \Delta t\nabla^2\phi^{n+1} &= \nabla\cdot\mathbf{u}^* && (2.51)\\ \mathbf{u}^{n+1} &= \mathbf{u}^* - \Delta t\nabla\phi^{n+1} && (2.52)\\ \nabla p^{n+1/2} &= \nabla p^{n-1/2} + \nabla\phi^{n+1} - \frac{\nu\Delta t}{2}\nabla\nabla^2\phi^{n+1} && (2.53) \end{aligned}$$

In dem beschriebenen Verfahren wird ein Crank-Nicholson-Schritt für die Dissipation verwendet, der der Stabilisierung dient. Ob diese Stabilisierung notwendig ist, hängt von der jeweiligen Problemstellung ab. In den hier durchgeführten Simulationen erwies sich der implizite Schritt als sehr zeitaufwendig, weshalb er durch den expliziten Schritt ersetzt wurde. Es ergaben sich wegen der günstigen Parameter trotzdem keine Instabilitäten.

Außerdem läßt sich die additive Berechnung des Drucks schlecht mit der Randverarbeitung von AMR vereinbaren, so daß in dieser Arbeit ein neues Verfahren daraus weiter entwickelt wurde, das im folgenden *PM III* genannt wird.

$$\begin{aligned} \partial_t \mathbf{u}^n &= -((\mathbf{u}\nabla)\mathbf{u})^{n+1/2} + \nu\Delta\mathbf{u}^n && (2.54)\\ &= \mathbf{n}^{n+1} && (2.55)\\ \Delta p &= \nabla\cdot\mathbf{n} + \frac{\nabla\cdot\mathbf{u}^n}{\Delta t} && (2.56)\\ \mathbf{u}^{n+1} &= \mathbf{u}^n + \Delta t(\mathbf{n} - \nabla p) && (2.57) \end{aligned}$$

Dabei ist $\mathbf{n}$ als Abkürzung für die rechte Seite der Navier-Stokes-Gleichung definiert.

Kim und Moin benutzen ein ähnliches Verfahren (ebenfalls in [BCM01] vorgestellt), das allerdings einen impliziten Zeitschritt und ein anderes Druck-Update benutzt. Außerdem fehlt der Term $\nabla\cdot\mathbf{u}^n$, der hier notwendig ist, um die Divergenz zu stabilisieren.

In der endgültigen Implementation des oben beschriebenen Codes wird die Nichtlinearität $\mathbf{n}$ mit Hilfe von CWENO berechnet und aus deren Divergenz in einem Poisson-Schritt der Druck gelöst. Im letzten Teil wird aus Nichtlinearität und Druckgradienten die neue Geschwindigkeit bestimmt. Dieser Zeitschritt ist in ein Runge-Kutta-Verfahren 3. Ordnung eingebettet (s. a. Anhang A zur Programmstruktur). Die Struktur des ursprünglichen AMR-Codes mußte für die hier gewählte Implementation an einer Stelle aufgebrochen werden. Üblicherweise werden in einer Routine die rechte Seite der DGL berechnet und der Zeitschritt ausgeführt und in einer weiteren Routine eine Poisson-Gleichung gelöst. Hier muß aber die Poisson-Gleichung zwischen Bestimmung der rechten Seite und dem Zeitschritt gelöst werden.

Das PM II -Verfahren ist sowohl 2- als auch 3dimensional implementiert worden, da es das einzige von den hier vorgestellten Verfahren ist, daß sowohl in zwei als auch drei Dimensionen nur einen Poisson-Schritt benötigt. Ansonsten ergeben sich beim Übergang von zwei zu drei Dimensionen keine substantiellen Änderungen.

2.5.3. Methode Mix

Da PM II mit der Adaptiven Gitterverfeinerung nicht funktioniert, (s. Kapitel 4) wird ein weiteres Verfahren benötigt, das die o.g. Anforderungen an den Code erfüllt.

Das neue Verfahren ist eine Mischung aus den beiden vorher angeführten Verfahren: die

Lösung der Navier-Stokes-Gleichung stammt aus PM II , während die Divergenzfreiheit über die Rotationsbildung hergestellt wird, wie es in RK der Fall ist. Daher wird das Verfahren auch *Mix* genannt.

Damit ist der erste Schritt die Bestimmung der Geschwindigkeit aus der Stromfunktion

$$\mathbf{u}^n = \nabla\times \Psi^n \tag{2.58}$$

Anschließend wird die Navier-Stokes-Gleichung gelöst

$$\partial_t\mathbf{u} = -\left(\mathbf{u}\nabla\right)\mathbf{u} + \nu\Delta\mathbf{u} \tag{2.59}$$

Um wieder die Stromfunktion zu erhalten, wird die Vortizität beötigt, wobei hier nur die Vortizität der rechten Seite der Gleichung bestimmt wird und zur alten Vortizität addiert wird

$$\partial_t\omega = \nabla\times(\partial_t\mathbf{u}) \tag{2.60}$$

$$\omega^{n+1} = \omega^n + dt\cdot\Delta\omega \tag{2.61}$$

Der letzte Schritt ist dann wieder die Poisson-Gleichung, um die Stromfunktion zu bestimmen

$$\Delta\Psi^{n+1} = -\omega^{n+1} \tag{2.62}$$

Der Lösungsschritt für die Navier-Stokes-Gleichung ist aus PM III entnommen, allerdings muß vorher ein Problem gelöst werden: Bei dem Übergang $\Psi \rightarrow \mathbf{u} \rightarrow \omega$ kommt es zur numerischen Dissipation durch Informationsverlust. Durch die Abfolge der Bearbeitung der Felder ist der folgende Wert von ω nicht von den nächsten Nachbarn abhängig. Um dies zu verhindern wird ein staggered Grid, also ein gestaffeltes Gitter, für $\mathbf{u}$ benutzt, daß zwischen den Originalgitterpunkten liegt(s. Abb. 2.2). Dabei müssen die Werte von $\mathbf{u}$ interpoliert werden, damit die einzelnen Komponenten auf demselben Gitter liegen. Dasselbe gilt natürlich auch bei der Berechnung der Nichtlinearität der Wirbelstärke.

Es wird hier nur eine lineare Interpolation für die Geschwindigkeitsbildung verwendet,

die völlig ausreichend ist, um die numerische Dissipation deutlich zu verrringern.

$$\mathbf{u} = \nabla \times \Psi \tag{2.63}$$

$$(u_x)_{i,j+1/2} = \frac{\Psi_{i,j+1} - \Psi i,j}{dy} \tag{2.64}$$

$$(u_y)_{i+1/2,j} = \frac{\Psi_{i+1,j} - \Psi i,j}{dx} \tag{2.65}$$

$$\mathbf{u}_{i+1/2,j+1/2} = \begin{pmatrix} 1/2\left((u_x)_{i+1,j+1/2} + (u_x)_{i,j+1/2}\right) \\ 1/2\left((u_y)_{i+1/2,j+1} + (u_y)_{i+1/2,j}\right) \end{pmatrix} \tag{2.66}$$

Nebem dem staggered Grid, das auch im Zusammenhang mit Randbedingungen nur geringe Probleme bereitet, hat das Mix-Verfahren allerdings einen Nachteil gegenüber dem PM II -Verfahren: Beim Übergang von zweidimensionalen zu dreidimensionalen Rechnungen werden statt einem nun drei Poisson-Schritte (für jede Komponente von Ψ einen) benötigt, die für die Rechnung geschwindigkeitsbestimmend sind. Diesem Nachteil steht aber gegenüber, daß die Mix-Methode mit adaptiver Gitterverfeinerung funktioniert. Damit ist Mix mit AMR im 2dimensionalen klar PM II überlegen, im 3dimensionalen muß man Kosten für die zusätzlichen Poisson-Schritte gegen die Vorteile durch adaptive Gitter abwiegen.

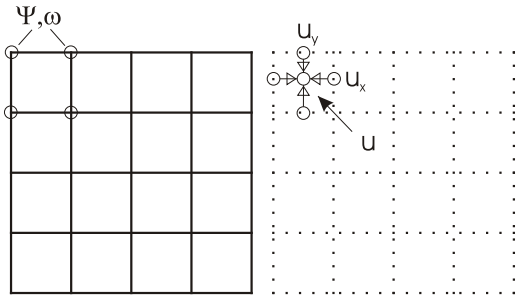


Abbildung 2.2.: *Gitter von Ψ, ω und* $\mathbf{u}$

2.6. Vergleich der Methoden

In diesem Abschnitt werden die vorgestellten numerischen Methoden anhand einiger einfacher Kriterien miteinander verglichen. Wichtige Kriterien sind die Erhaltung von Energie und Enstrophie

$$E = \int u^2 \, dV \tag{2.67}$$

$$\Omega = \int \omega^2 \, dV \tag{2.68}$$

sowie das Ausbilden von Strukturen und Filamenten.

Es werden Ergebnisse mit und ohne *limiting* gezeigt. Im Fall von *non-limiting* werden im CWENO-Verfahren die w_i, statt sie aus den Smoothness-Faktoren zu berechnen, gleich den c_i gesetzt (s. Gl. 2.20). Dies hat, wie sich zeigen wird, einen großen Einfluß auf die Dämpfung.

Als Anfangsbedingung für den Test werden zwei (gleichsinnige) Gauß-Vortizes benutzt

$$\omega(x,y) = A \cdot \left(e^{-\alpha((x-x_1)^2+(y-y_1)^2)} + e^{-\alpha((x-x_2)^2+(y-y_2)^2)} \right) \tag{2.69}$$

wobei $A = 1$ und $\alpha = 1.5625$ gesetzt wurde. Mit der Halbwertsbreite $R = \sqrt{\frac{\log 2}{\alpha}}$ und der maximalen Anfangsgeschwindigkeit v, kann nun die Reynolds-Zahl dieser Konfiguration bestimmt werden, die in diesem Fall in der Größenordnung von $Re \approx 9000$ liegt.

In Abb. 2.3 ist deutlich zu erkennen, daß die Verfahren Mix und PM II für die Energie identische und gute Ergebnisse liefert, während RK eine starke numerische Dissipation aufweist. Ein ähnliches, wenn auch weniger ausgeprägtes Verhalten sieht man bei der Enstrophie (s. Abb 2.4). Allein dies ist ein deutliches Signal für die Güte der verwendeten Verfahren Mix und PM II , insbesondere für die Lösung der Navier-Stokes-Gleichung in der Geschwindigkeit.

Auffällig ist der Unterschied zwischen *limiting* und *non-limiting* für RK, wie in Abb. 2.5 und 2.6 dargestellt. Die starke numerische Dissipation im *limiting*-Fall läßt sich durch die Dämpfung starker Gradienten erklären. Daß dieser Effekt nicht bei den anderen Verfahren auftritt, hängt wahrscheinlich mit der Tatsache zusammen, daß beim Übergang von Geschwindigkeit zur Vortizität die Gradienten stärker werden.

In Abb. 2.7 und 2.8 ist im Zentrum der beiden Wirbel zu erkennen, daß diese nicht miteinander verschmelzen, dies wird vom RK-Code nicht so deutlich reproduziert (s. Abb. 2.9 und 2.10), hier ist der Übergang zwischen den Wirbeln verschwommen . Die Bilder für den Mix-Code (Abb. 2.11 bis 2.14) zeigen zwar die klare Trennung der Wirbel, allerdings erscheinen hier kleine Artefakte in ihrer Spitze.

Wie die Daten für die Energie schon vermuten lassen, wird die Trennung der Wirbel im *non-limiting* Fall besser (s. Abb. 2.13 und 2.14). Für RK zeigen sich dann Bilder, wie sie Mix und PM II auch im *limiting* Fall zeigen (weshalb sie hier nicht gesondert dargestellt werden), bei den letztgenannten ändert sich in *non-limiting* Fall nichts.

In einem weiteren Test wird der Parameter A aus Gl. 2.69 auf den Wert 10 gesetzt und

somit eine zehnmal höhere Reynolds-Zahl ($Re = 90000$) simuliert. Für die Energie (Abb. 2.20) ergeben sich ähnliche Ergebnisse wie für niedrige Reynolds-Zahlen, die Enstrophie zeigt aber einen deutlich stärkeren Abfall (Abb. 2.21). Die Vortizität bildet erwartungsgemäß für PM II klare Strukturen, während bei RK die Wirbel sehr schnell verschmelzen (Abb. 2.15 und 2.16).

Ein Grund für den deutlich stärkeren Abfall der Enstrophie bei einer Auflösung von 256×256 liegt in der mangelnden Auflösung. Die Simulationen mit 512 bzw. 1024 Punkten Kantenlänge (bei gleichem CFL) zeigen einen geringeren Abfall, auf Grund der besseren Auflösung kleinskaliger Bewegungen. Auf groben Gittern werden eben diese kleinskaligen Bewegungen zu stark gedämpft.

Leider stehen für die Auflösung von 1024×1024 Punkten nur wenig Daten zur Verfügung, da sich die Fläche bei einer Verdoppelung der Auflösung vervierfacht und zusätzlich der Zeitschritt verdoppelt werden muß, um CFl konstant zu halten. Wird außerdem das Laufzeitverhalten des zeitbestimmenden Poisson-Schrittes von $N^2 \log N$ (mit N Anzahl der Punkte pro Kante), so wird deutlich daß bei einem Übergang von 256 auf 1024 Punkte die Laufzeit um den Faktor 10 zunimmt. Das heißt, daß eine Simulation mit $Re = 90,000$ bei einer Auflösung von 1024×1024 eine 100mal größere Rechenzeit benötigt wie eine $Re = 9,000$ Simulation mit Auflösung 256×256, da eine höhere Reynolds-Zahl hier mit einer höheren Geschwindigkeit einhergeht. Daher ist es unmöglich, mit den für diese Arbeit zur Verfügung stehenden Computern die Simulation weiter als bis $t = 5$ laufen zu lassen. Trotzdem läßt Abb. 2.18 die Tendenz erkennen, daß mit verbesserter Auflösung die Enstrophie immer besser erhalten wird. Abb. 2.21 zeigt zusätzlich, daß PM II eine ähnlich gute Enstrophie-Erhaltung aufweist wie eine doppelt so hoch aufgelöste RK Simulation.

Zu dem dreidimensionalen PM II -Code existieren einfache quasi-zweidimensionale Rechnungen, die für jede Kombination zweier Richtungen den hier angeführten zweidimensionalen Testfall durchführen. Die Ergebnisse stimmen mit den echt-zweidimensionalen Test überein.

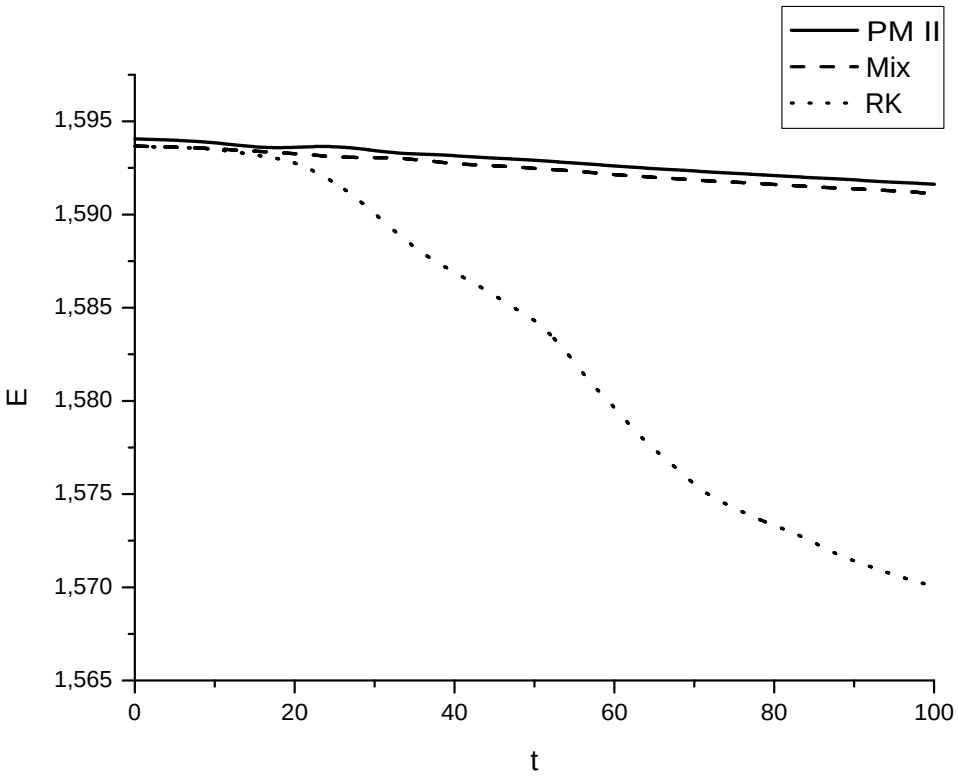


Abbildung 2.3.: *Vergleich von Energie für die drei Verfahren, 256x256 Gitterpunkte*

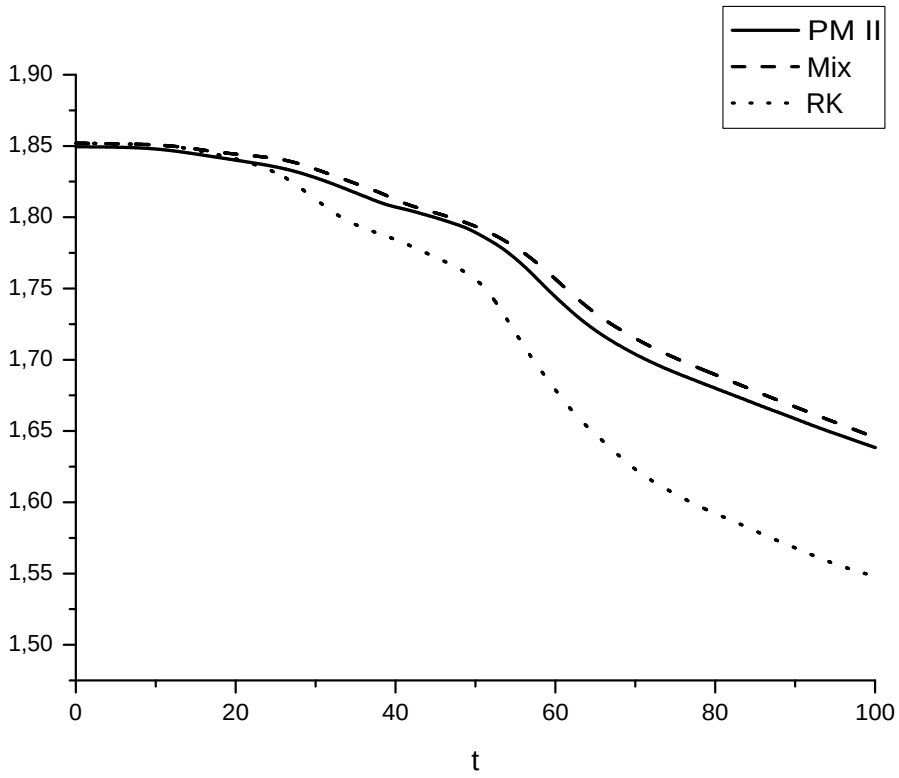


Abbildung 2.4.: *Vergleich von Enstrophie für die drei Verfahren, 256x256 Gitterpunkte*

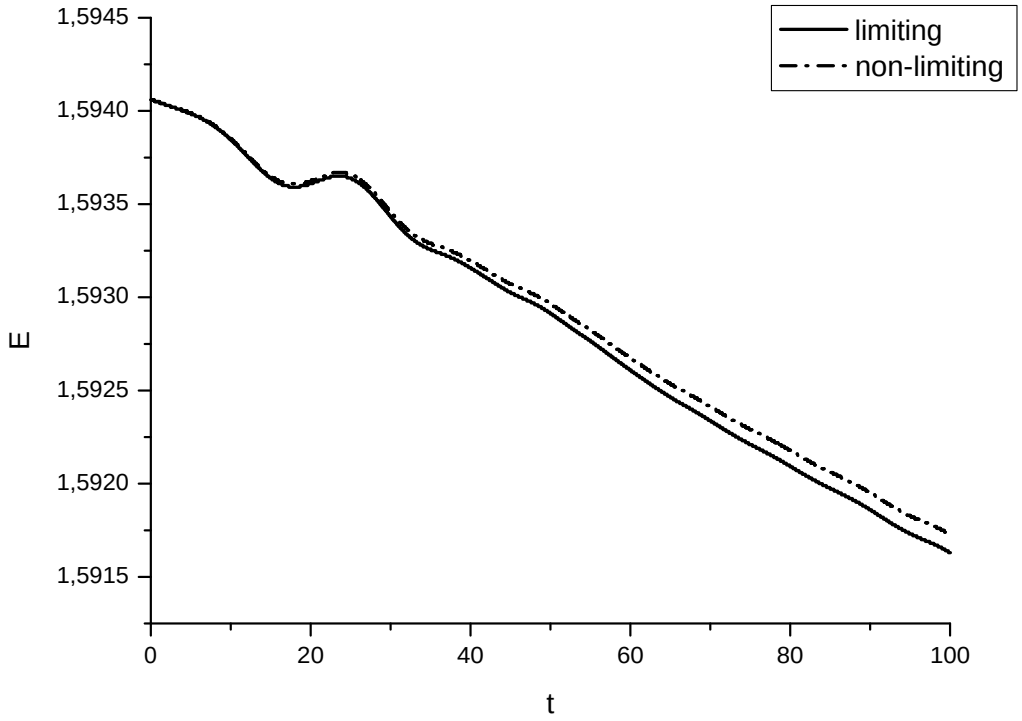


Abbildung 2.5.: *Vergleich limiting/non-limiting PM II*

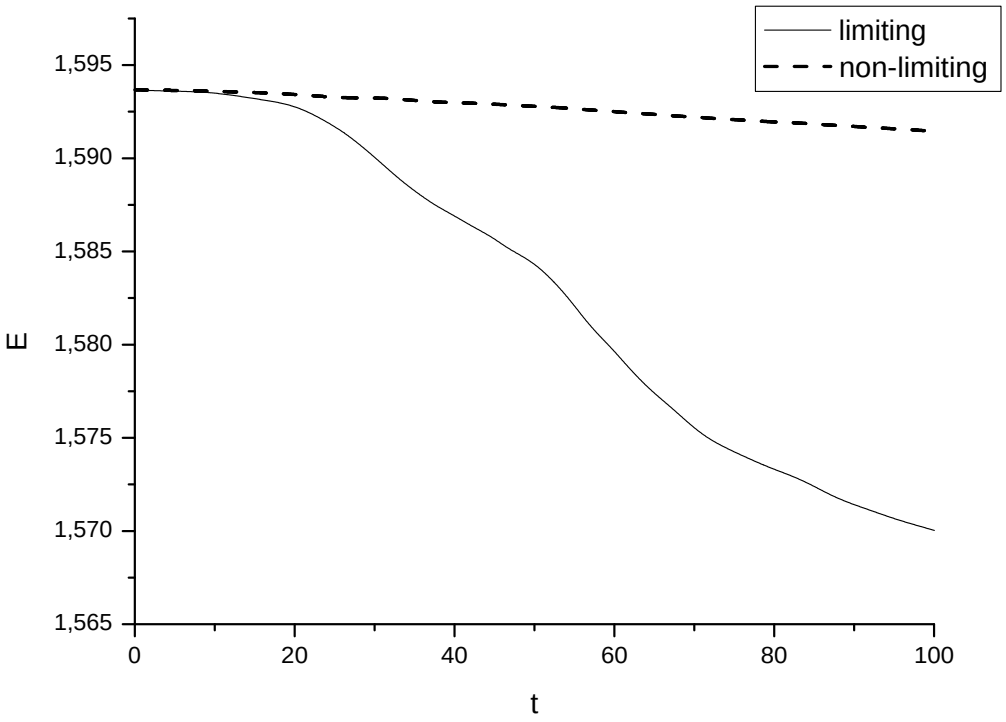


Abbildung 2.6.: *Vergleich limiting/non-limiting RK*

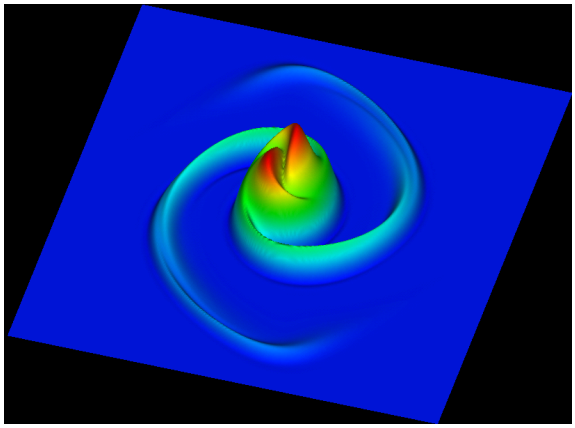


Abbildung 2.7.: *PM II zur Zeit* $t = 50$*, Limiting*

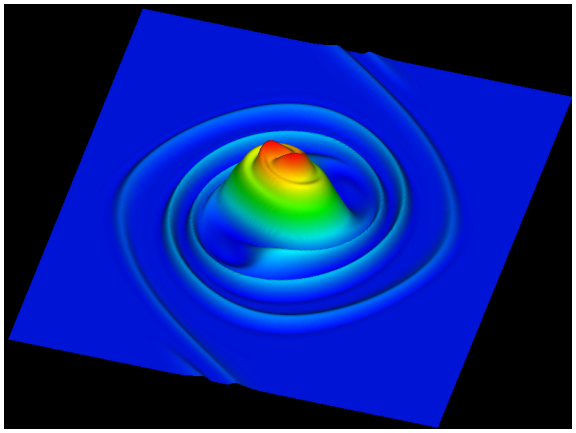


Abbildung 2.8.: *PM II zur Zeit* $t = 100$*, Limiting*

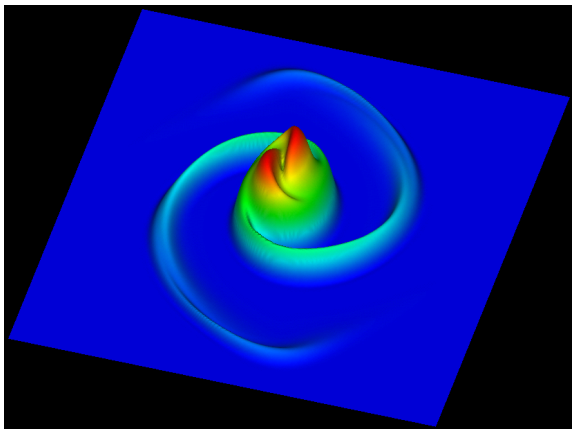


Abbildung 2.9.: *RK zur Zeit* $t = 50$*, Limiting*

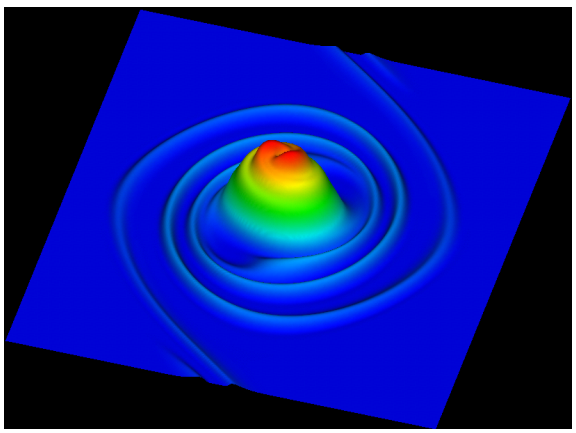


Abbildung 2.10.: *RK zur Zeit* $t = 100$*, Limiting*

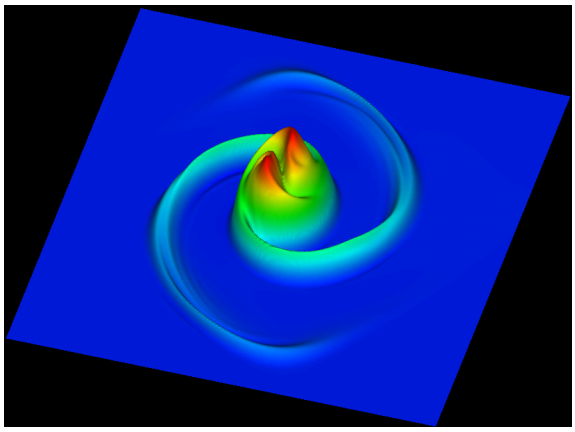


Abbildung 2.11.: *Mix zur Zeit* $t = 50$*, Limiting*

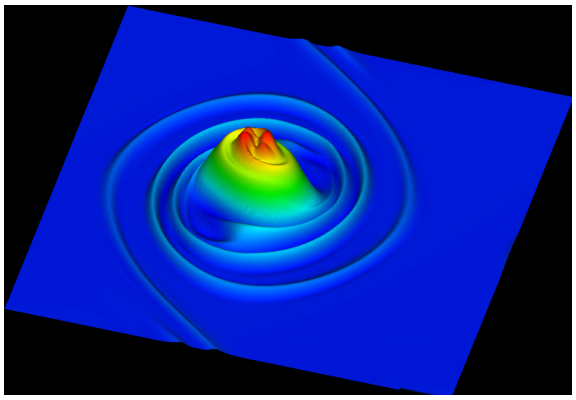


Abbildung 2.12.: *Mix zur Zeit* $t = 100$*, Limiting*

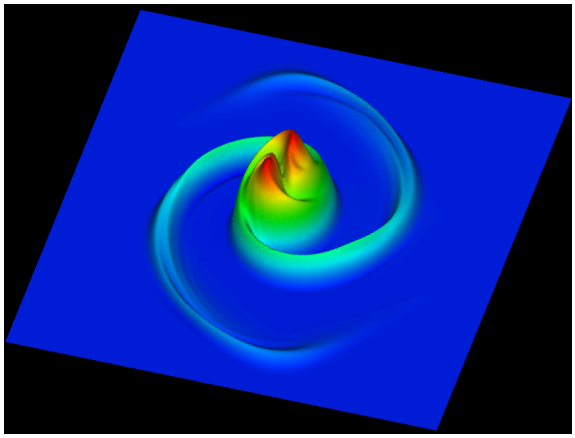


Abbildung 2.13.: *Mix zur Zeit* $t = 50$*, kein Limiting*

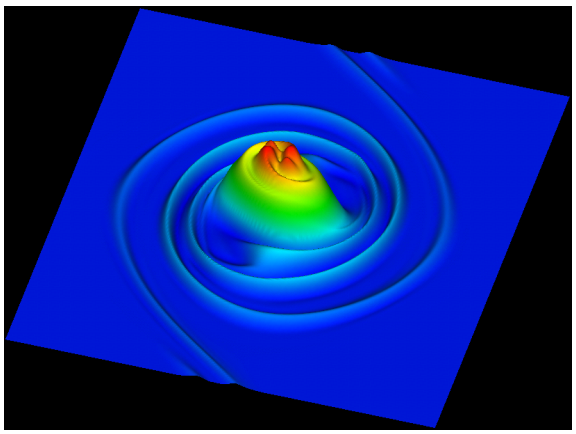


Abbildung 2.14.: *Mix zur Zeit* $t = 100$*, kein Limiting*

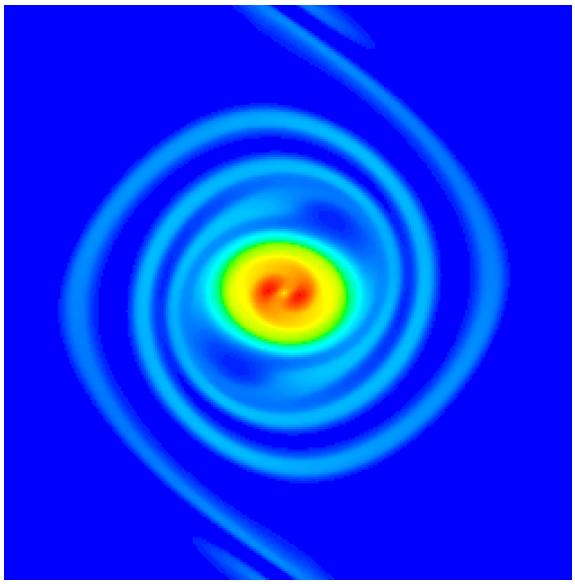


Abbildung 2.15.: *RK zur Zeit* $t = 10$, *Re 90,000*

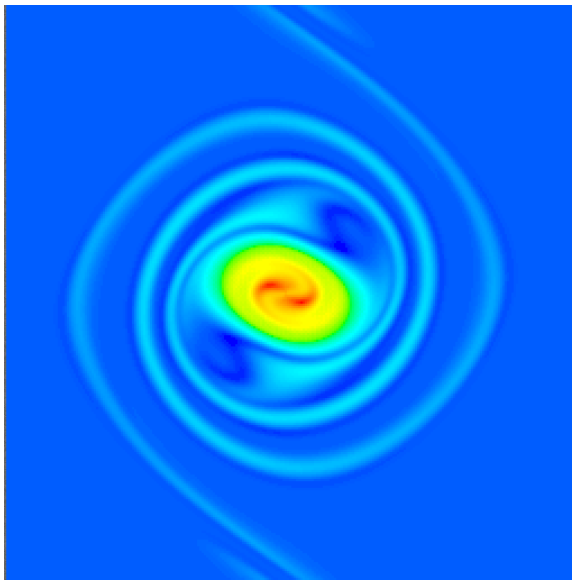


Abbildung 2.16.: *PM II zur Zeit* $t = 10$, *Re 90,000*

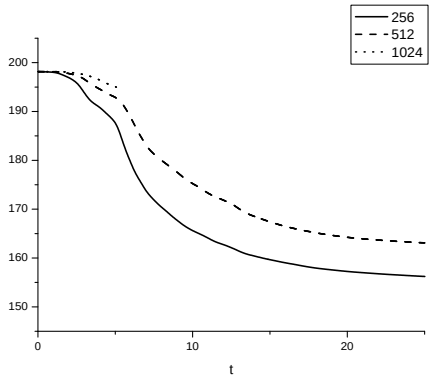


Abbildung 2.17.: *Re 90,000 Vefahren RK*

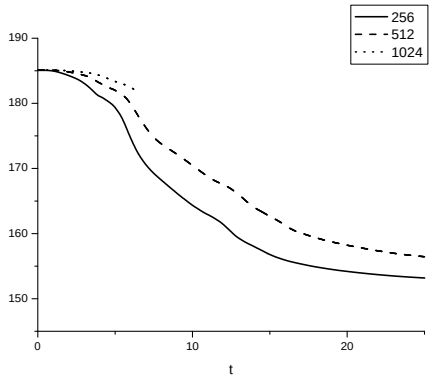


Abbildung 2.18.: *Re 90,000 Verfahren PM II*

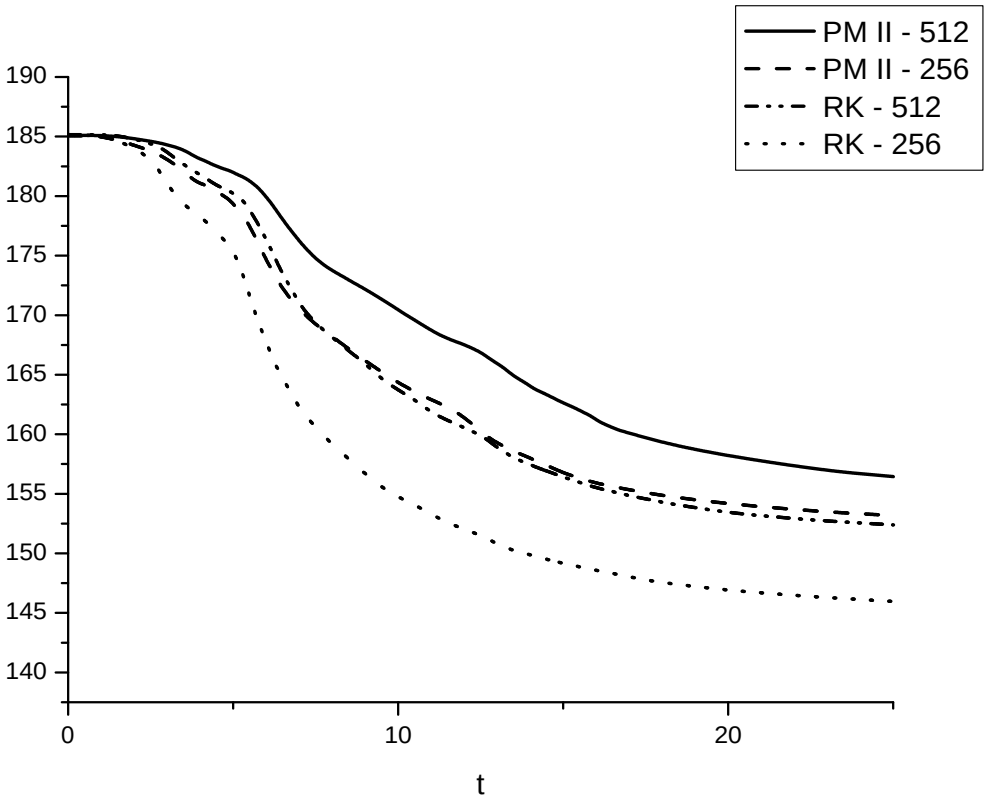


Abbildung 2.19.: *Vergleich der Enstrophie für PM II und RK bei 256 und 512 Punkten Auflösung*

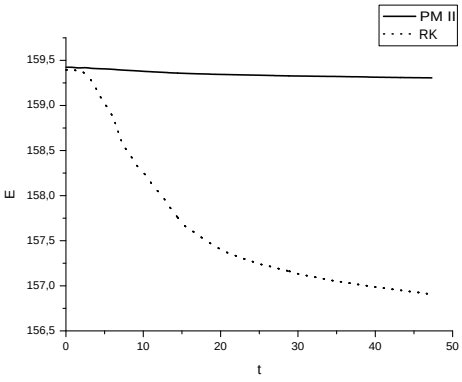


Abbildung 2.20.: *Vergleich von Energie bei Re 90,000*

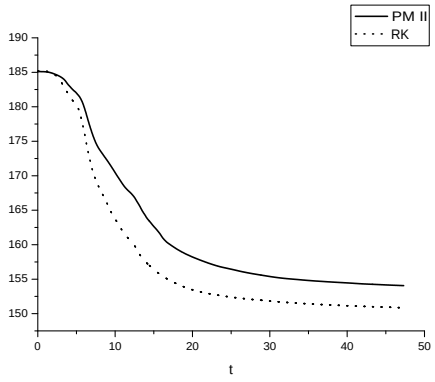


Abbildung 2.21.: *Vergleich von Enstrophie bei Re 90,000*

3. Geometrien

Der erste Teile dieses Kapitel behandelt die Implementation einfacher Geometrien, die benötigt werden, um Ein- und Ausfluß von Fluiden zu realisieren. Der zweite Teil beschäftigt sich mit der Implementation komplexer Geometrien mit Hilfe von Penalty-Methoden mit einigen Anwendungsbeispielen.

3.1. Einfache Geometrien

Unter dem Begriff einfache Geometrien lassen sich Systeme zusammenfassen, deren physikalische Ränder mit den Rändern des Integrationsgebietes zusammenfallen. Einfache Geometrien können durch eine geeignete Wahl der Randbedingungen der Integration abgebildet werden. Im Gegensatz zu den später beschriebenen Forcing-Verfahren werden die Randgeschwindigkeiten direkt vorgegeben und der Druck durch von Neumann-Bedingungen berechnet. Bei von Neumann-Randbedingungen wird die Ableitung der Lösung am Rand vorgegeben, Dirichlet-Randbedingungen dagegen geben die Lösung selber am Rand vor.

3.1.1. Randbedingungen für die Geschwindigkeitsformulierung

Johnston und Liu [JL00] haben ein Verfahren vorgestellt, um in einem geschlossen Kasten die Navier-Stokes-Gleichung zu integrieren. Die Problematik hierbei ist die Randbedingung für die Bestimmung des Drucks im Poisson-Schritt. Das Verfahren läßt sich in folgende Schritte gliedern:

1. Lösung der Navier-Stokes-Gleichung im Inneren

2. Bestimmung der von Neumann-Bedingung für den Druck

3. Lösen der Poisson-Gleichung für den Druck

Die Randbedingung hier ist $\mathbf{u} = 0$ auf allen Rändern, später sollen auch von Null verschiedene Geschwindigkeiten angenommen werden. Außerdem gilt auch auf dem Rand die Inkompressibilitätsbedingung $\nabla \cdot \mathbf{u} = 0$. Daraus ergibt sich (hier gezeigt am linken Rand)

$$u_x(0,j) = u_y(u,j) = 0 \tag{3.1}$$

$$\nabla \cdot \mathbf{u}(0,j) = 0 \tag{3.2}$$

$$\Rightarrow \frac{u_x(1,j) - u_x(-1,j)}{\Delta x} = 0 \tag{3.3}$$

Dies ist die sogenannte Spiegelbedingung für die Geschwindigkeit, die sich für diskretisiert Ableitung 2. Ordnung ergeben. Unter Verwendung dieser Formeln kann die Ableitung des Drucks entlang der Normalen bestimmt werden

$$\frac{\partial p}{\partial x} = -\partial_t u_x - u_x \partial_x u_x - u_x \partial_y u_y + \nu \Delta u_x \tag{3.4}$$

$$= \nu \Delta u_x \tag{3.5}$$

$$= \nu\Big(\frac{u_x(0,j+1) - 2u_x(0,j) + u_x(0,j-1)}{(\Delta y)^2} + \frac{u_x(1,j) - 2u_x(0,j) + u_x(-1,j)}{(\Delta x)^2}\Big) \tag{3.6}$$

$$= \nu \frac{u_x(1,j) + u_x(-1,j)}{(\Delta x)^2} \tag{3.7}$$

$$\Rightarrow \frac{\partial p}{\partial x} = \frac{2\nu}{(\Delta)^2} u_x(1,j) \tag{3.8}$$

In der ersten Zeile fallen alle Terme bis auf die Viskosität weg, da u_x auf dem Rand konstant Null ist.

So lassen sich die Randbedingung für die Geschwindigkeit und die Inkompressibilität zu einer von Neumann-Bedingung für den Druck kombinieren, bei deren Berechnung nur Punkte im Inneren benötigt werden. Im folgenden werden einige einfache Geometrien an Hand dieses Formalismus beschrieben.

Bei der Verwendung dieser Methode müssen die “boundary”-Routinen des AMR-Codes angepasst werden, um für alle Größen die korrekten Randwerte zu erhalten:

Geschwindigkeit Die Geschwindigkeitskomponente normal zum Rand (x-Richtung) wird an den Rändern gemäß der Annahme über die Divergenz gespiegelt.

$$u_x(-i,j) = u_x(i,j) \tag{3.9}$$

Die anderen Geschwindigkeitskomponenten werden Null gesetzt.

Druck Aus der Ableitung des Drucks in der Richtung der Normalen (hier x) läßt sich der Druck linear approximieren

$$p(-i,j) = p(1,j) - 2(i+1)\frac{\nu}{(\Delta x)^2}u_x(1,j) \tag{3.10}$$

Als einfache Geometrie kann man nun einen Kasten annehmen und erhält folgenden Satz von Randbedingungen:

$$\begin{aligned}
\mathbf{u}(0,j) &= 0 && (3.11)\\
\mathbf{u}(i,0) &= 0 && (3.12)\\
\mathbf{u}(i_{max},j) &= 0 && (3.13)\\
\mathbf{u}(i,j_{max}) &= 0 && (3.14)\\
\frac{\partial p}{\partial x}|_{(0,j)} &= \frac{2\nu}{(\Delta x)^2}u_x(1,j) && (3.15)\\
\frac{\partial p}{\partial x}|_{(i_{max},j)} &= \frac{2\nu}{(\Delta x)^2}u_x(i_{max}-1,j) && (3.16)\\
\frac{\partial p}{\partial y}|_{(i,0)} &= \frac{2\nu}{(\Delta y)^2}u_y(i,1) && (3.17)\\
\frac{\partial p}{\partial y}|_{(i,j_{max})} &= \frac{2\nu}{(\Delta y)^2}u_y(i,j_{max}-1) && (3.18)
\end{aligned}$$

Für spätere Anwendungen ist noch eine Geometrie mit konstantem Ein- und Ausfluß erforderlich (auch Inflow-Geometrie genannt). Die Herleitung der Randbedingung folgt Gleichung 3.4, allerdings mit konstantem u_x auf zwei gegenüberliegenden Rändern. Für die verbleibenden zwei Ränder werden periodische Randbedingungen angenommen.

$$\begin{aligned}
\mathbf{u}(0,j) &= c\mathbf{e}_x && (3.19)\\
\mathbf{u}(i_{max},j) &= 0 && (3.20)\\
\frac{\partial p}{\partial x}|_{(0,j)} &= \frac{2\nu}{(\Delta x)^2}(u_x(1,j)-c) && (3.21)\\
\frac{\partial p}{\partial x}|_{(i_{max},j)} &= \frac{2\nu}{(\Delta x)^2}u_x(i_{max}-1,j) && (3.22)\\
& && (3.23)
\end{aligned}$$

3.1.2. Randbedingungen für die Wirbelstärkeformulierung

Ein ähnlicher Formalismus wie der vorherige existiert auch für die Wirbelstärke und ist bekannt als Thoms Formel. Mit ähnlicher Überlegung wie oben folgt

$$\begin{aligned} \frac{\partial \Psi}{\partial \mathbf{n}} &= 0 && (3.24)\\ \Psi &= 0 && (3.25)\\ \Rightarrow \omega_{i,0} &= 2\frac{\Psi_{i,1}}{(\Delta x)^2} && (3.26) \end{aligned}$$

Im Unterschied zur Geschwindigkeitsformulierung wird im Poisson-Schritt (diesmal für die Stromfunktion) eine Dirichlet-Randbedingung verwendet, da hier am Rand die Werte von Ψ bekannt sind.

Analog zur Geschwindigkeitsformulierung wird auch hier ein konstanter Durchfluß durch eine Geometrie benötigt. Ausgehend von den Randbedingungen für die Geschwindigkeit

$$\begin{aligned} u_x &= c && (3.27)\\ u_y &= 0 && (3.28) \end{aligned}$$

können mit der Definition $\nabla \times \Psi = \mathbf{u}$ die Randbedingungen ausgerechnnet werden(Exemplarisch wird dies hier für den linken und oberen Rand durchgeführt)

linker Rand

$$\begin{aligned} \frac{\partial \Psi}{\partial y}|_{(0,j)} &= c && (3.29)\\ \Rightarrow \Psi(0,j) &= j\Delta x c && (3.30)\\ \frac{\partial \Psi}{\partial x}|_{(0,j)} &= 0 && (3.31)\\ \Rightarrow \Psi(-1,j) &= \Psi(1,j) && (3.32)\\ \Delta \Psi &= \omega && (3.33)\\ \Rightarrow \omega(0,j) &= \frac{2\Psi(1,j) - 2j\Delta y c}{(\Delta x)^2} && (3.34) \end{aligned}$$

oberer Rand

$$\frac{\partial \Psi}{\partial y}|_{(i,0)} = c \tag{3.35}$$

$$\Rightarrow \Psi(i,-1) = \Psi(i,1) - 2\Delta y c \tag{3.36}$$

$$\frac{\partial \Psi}{\partial x}|_{(i,0)} = 0 \tag{3.37}$$

$$\Rightarrow \Psi(i-1,0) = \Psi(i+1,0) \tag{3.38}$$

$$\Delta \Psi = \omega \tag{3.39}$$

$$\Rightarrow \omega(i,0) = \frac{2\Psi(i,1) - 2\Delta y c - 2\Psi(i,0)}{(\Delta y)^2} \tag{3.40}$$

$$\tag{3.41}$$

Auch hier müssen die boundary-Routinen angepaßt werden, insbesondere für die Stromfunktion müssen die Randwerte nach der Spiegelmethode gesetzt werden.

Wegen des linearen Anstiegs von Ψ in y-Richtung an den Ränder ist eine Implementation von periodischen Randbedingungen am oberen und unteren Rand nur schwer zu realisieren. Daher wird in dieser Arbeit nur eine Kanalgeometrie benutzt, die einen Ein- und Ausfluß, sowie zwei feste Wände besitzt.

3.2. Penalty-Methoden

Komplexe Geometrien erfordern in der Simulation einen erheblich größeren Aufwand als die im letzten Abschnitt beschriebenen einfachen Geometrien. Häufig bauen Simulationen auf der Finite-Elemente-Methode (FEM) auf, die zwar in der Rechnung einfach zu verwenden ist, allerdings für die Erzeugung der Gitter einen hohen Rechenaufwand besitzt. Außerdem sind diese Verfahren nur wenig in Richtung der Cache-Nutzung optimiert.

Fadlun et. al. ([FVOMY00]) nutzen hingegen statt der problemangepaßten Koordinaten der FEM orthogonale Gitter. Sie verwenden virtuelle Kräfte auf die Strömung, um die Randbedingungen der Geometrie zu erfüllen. Die größte Schwierigkeit bei dieser Methode ist die mangelnde Übereinstimmung von Geometrie und Gitter, die allerdings durch geeignete Interpolation gelöst wird. In [FVOMY00] werden zwei verschiedene Ansätze diskutiert, um die Randbedingungen für umströmte Körper einzuhalten. Beide wirken sich unterschiedlich auf die Steifigkeit der DGL aus.

3.2.1. Kraft

Wie schon oben erwähnt gibt es zwei Methoden, um eine Geschwindigkeitsverteilung $\mathbf{V}$ durch Kräfte zu erzielen. Die erste soll nur der Vollständigkeit halber hier kurz erläutert werden, da sie weder eine hohe Stabilität besitzt noch sich leicht implementieren läßt. Sie basiert auf der Gleichung

$$\mathbf{f}(\mathbf{x},t) = \alpha_f \int_0^t (\mathbf{u}(\mathbf{x},t') - \mathbf{V}(\mathbf{x},t'))\, dt' + \beta_f \left(\mathbf{u}(\mathbf{x},t') - \mathbf{V}(\mathbf{x},t')\right) \tag{3.42}$$

die das numerische Analogon zu einem PI-Regler ist, da sie sowohl auf den Proportional- als auch den Integralanteil des Unterschieds zwischen Soll- und Istgeschwindigkeit reagiert. α_f und β_f sind Konstanten, deren Werte von den simulierten Strömungen abhängen. Hohe Werte der Konstanten führen zu steifen Gleichungen, die die Bedingungen für den Zeitschritt deutlich verschärfen (Fadlun et al. erwähnen CFL= $O(10^{-2})$).

In einem anderen Ansatz wird das *direct Forcing* eingesetzt. Hier kommen virtuelle Kräfte auf die Flüssigkeitselemente zum Einsatz, die nicht durch Einsatz eines Reglers stabilisiert werden, sondern durch passende Interpolation der Randbedingungen.

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = \mathrm{RHS}^{n+1/2} + \mathbf{f}^{n+1/2} \tag{3.43}$$

$$\Rightarrow \mathbf{f}^{n+1/2} = -\mathrm{RHS}^{n+1/2} + \frac{\mathbf{V}^{n+1} - \mathbf{u}^n}{\Delta t} \tag{3.44}$$

RHS bezeichnet die rechte Seite der Navier-Stokes-Gleichung . Aus dieser Formel folgt, daß nun auf der Geometrie jeder Punkt auf die Geschwindigkeit $\mathbf{V}$ gezogen wird.

Dieses Verfahren ist erheblich stabiler und erfordert keinerlei Zusatzterme, die aufwendig berechnet werden müssten.

3.2.2. Interpolation

Um die Diskrepanz zwischen krummliniger Geometrie und orthogonalem Gitter zu überbrücken, ist ein Konzept zur Interpolation der Randbedingungen auf das vorhandene Gitter erforderlich. Nach Fadlun et al. gibt es drei verschiedene Verfahren, denen gemeinsam ist, daß die Kraft auf einen Punkt wirkt, der dem umströmten Körper am nächsten ist (Forcing-Punkt). An diesem Forcing-Punkt muß dann eine Sollgeschwindigkeit bestimmt werden, die sich aus der Sollgeschwindigkeit auf dem Körper und der Strömungsgeschwindigkeit eines weiteren Punktes (Interpolationspunkt) zusammensetzt. Während

Forcing-Punkte die dem Körper nächsten Punkte sind (s.o.), lassen sich Interpolationspunkte über ein Normalenalgorithmus finden, der später detaillierter beschrieben wird (s. Kapitel 3.2.3). Der Interpolationsfaktor zwischen Körper und Interpolationspunkt wird durch das jeweilige Verfahren bestimmt.

No Interpolation ist sicher das einfachste denkbare Verfahren, das die Sollgeschwindigkeit des Forcing-Punktes wird einfach gleich der Sollgeschwindigkeit des Körpers gesetzt.

Volume Fraction berechnet den Interpolationsfaktor aus dem Anteil der Körpers an der Zelle, an deren Ecke der Forcing-Punkt liegt.

Linear Interpolation berechnet die Sollgeschwindigkeit als lineare Interpolation zwischen Körper und Interpolationspunkt. Der Interpolationsfaktor hängt vom Abstand zwischen Körper, Forcing- und Interpolationspunkt ab.

Die Verfahren sind schematisch in Abb. 3.1 dargestellt.

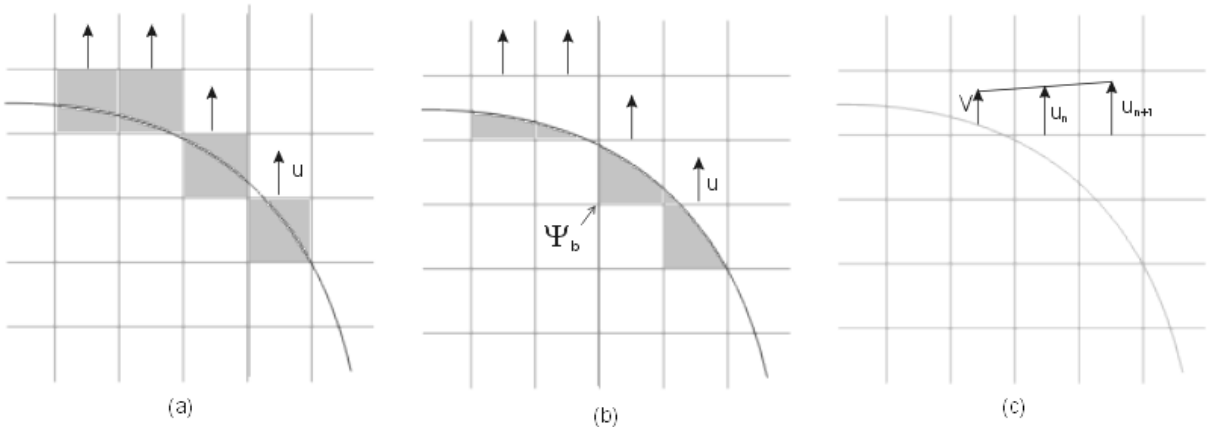


Abbildung 3.1.: *Interpolationsverfahren für Penalty-Methoden: (a) No Interpolation (b) Volume Fraction (c) Linear Interpolation*

Die beiden letztgenannten Verfahren sind von etwa gleicher Genauigkeit (zwischen erster und zweiter Ordnung), einzig die Genauigkeit des ersten Verfahrens ist schlechter als erste Ordnung. In dieser Arbeit fiel die Wahl auf das lineare Interpolationsschema, da es einfacher zu implementieren und geringfügig genauer ist. Die genaue Implementation wird in Kapitel 3.2.3

3.2.3. Listenschema

Um das direct Forcing zu implementieren, werden die Punkte benötigt, an denen die Kraft wirkt, sowie die dazugehörigen Interpolationsfaktoren und Sollgeschwindigkeiten. Das daraus entwickelte generische Verfahren kann beliebige Geometrien zerlegen, sofern es einen Algorithmus gibt, mit dem der Abstand zwischen dem umströmten Körper und einem Punkt berechnet werden kann. Hierbei wird in einem ersten Schritt eine Liste der Forcing-Punkte angelegt. Im zweiten Schritt werden die Interpolationspunkte bestimmt und aus der Abstandsfunktion die Interpolationsfaktoren berechnet.

Schema der Punktsuche Im 2dimensionalen gibt es ein einfaches Schema, um alle Punkte zu finden, die in der Nähe des Körpers liegen.

1. Zerlege die Geometrie in einfache Teilgeometrien, für die die Abstandsfunktion bekannt ist.

2. Wähle einen geeigneten (außenliegenden) Anfangspunkt, der sicher ein nächster und kein übernächster Punkt ist (eine gute Wahl sind Punkte, die *auf* der Geometrie liegen.)

3. Prüfe die umliegenden 8 Punkte. Wähle den Punkt, der außen liegt, den geringsten Abstand zur Geometrie hat und noch nicht vorher besucht wurde und füge ihn zur Liste hinzu.

4. Wiederhole Punkt 3, bis das Abbruchkriterium erfüllt ist. Beim Abbruchkriterium muß zwischen offenen und geschlossenen Geometrien unterschieden werden:

   geschlossene Geometrie: Der Endpunkt ist gleich dem Anfangspunkt.

   offene Geometrie: Der Abstand zur Geometrie ist größer als die Zelldiagonale des Gitters.

Im 3dimensionalen ist mehr Aufwand erforderlich, da hier keine eindeutige Kurve existiert, auf der alle Punkte liegen. Deswegen wird hier ein Würfel angenommen, der den Körper vollständig umschließt. In diesem Würfel wird jeder Punkt darauf geprüft, ob er ein möglicher Forcing-Punkt ist. Dazu wird untersucht, ob

- er außerhalb des Körpers liegt.
- er einen Abstand vom Körper von weniger als einer Zelldiagonale hat.
- mindestens 2 seiner 26 Nachbarn innerhalb des Körpers liegen.

Der letzte Punkt ist nicht eindeutig, hier können auch andere Kriterien gewählt werden. Eine Alternative ist die Beschränkung auf die sechs direkten Nachbarn, von denen nur einer im Inneren liegen müßte.

Wahl der Interpolationspunkte Zur Demonstration der Wahl der Interpolationspunkte wird ein Kreissegment gewählt (Abb. 3.2). Hier ist die Normale des Kreises zu sehen, die durch den Forcing-Punkt verläuft. Als Interpolationspunkte bieten sich die drei Punkte in Richtung der Normalen an (Nord **N**, Nordwest **NW**, West **W**). Nach der Berechnung des Schnitts der Normalen mit der Zelle, in der Forcing- und Interpolationspunkt liegen, wird geprüft in welcher Hälfte einer Zellkante die Normale diese schneidet. Da der obere Schnittpunkt in der linken Hälfte liegt, kann der N-Punkt ausgeschlossen werden. Mit derselben Argumentation läßt sich auch der W-Punkt ausschließen. Der Interpolationsfaktor berechnet sich nun folgendermaßen ($\mathbf{X}_0$: Kreismittelpunkt, $\mathbf{x}$ Forcingpunkt, $\mathbf{x}_i$ Interpolationspunkt)

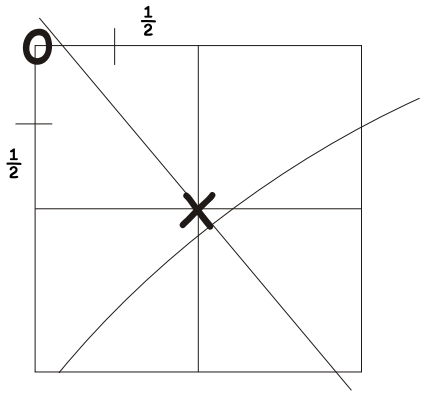


Abbildung 3.2.: *Forcing-Punkt(x), Interpolationspunkt(o), Kreissegment und Normale*

$$IF = \frac{|\mathbf{x} - \mathbf{X}_0| - R}{|\mathbf{x} - \mathbf{X}_0| + |\mathbf{x}_i - \mathbf{X}_0| - 2R} \tag{3.45}$$

Schema des Forcing Sind Forcing- und Interpolationspunkte sowie Interpolationsfaktoren bekannt, kann die Kraft in die Berechnungen aufgenommen werden. Bei der Auswertung der rechten Seite der Navier-Stokes-Gleichung werden dann folgende Schritte eingefügt:

1. An jedem Punkt der Geometrie wird die lokale Sollgeschwindigkeit bestimmt

$$\mathbf{V}(\mathbf{x}) = (1 - IF)\mathbf{V} + IF\mathbf{u}(\mathbf{x}_i) \tag{3.46}$$

2. Die wirkende Kraft wird direkt aus Differenz zwischen Ist- und Soll-Geschwindigkeit bestimmt

$$\mathbf{F}(\mathbf{x}) = \frac{\mathbf{V}(\mathbf{x}) - \mathbf{u}(\mathbf{x})}{\Delta t} \tag{3.47}$$

Es spielt dabei keine, ob das Innere des Körpers explizit behandelt wird oder die Geschwindigkeit im Inneren vollständig auf Null gesetzt wird.

3.2.4. Implementation

Das beschriebene Forcing-Verfahren wurde sowohl für den PM II -Code als auch für den Mix-Code implementiert. In ersterem ergeben sich keine nennenswerten Probleme, da er als direkt geschwindigkeitsbasierter Code alle Voraussetzungen für dieses Verfahren erfüllt. Der Mix-Code stellt allerdings eine gewisse Hürde dar. Auf Grund seines staggered Grid ist es von großer Bedeutung, auf welchem Gitter welche Größen existieren. Nach dieser Überlegung wird deutlich, daß die Forcing-Punkte um $(\frac{1}{2}, \frac{1}{2})$ gegenüber dem Standardgitter verschoben werden müssen.

In der Implementation der komplexen Geometrie muß nun auch auf die einfachen Geometrien zurückgegriffen werden, die im letzten Abschnitt eingeführt wurden. Dabei ist noch einmal darauf hinzuweisen, daß bei Verwendung des Mix-Codes Inflow-Geometrien nur als Kanal-Geometrie, nicht aber als periodische Geometrie ausgeführt werden können, da die Lösung der Poisson-Gleichung hier keine periodischen Randbedingungen zuläßt.

3.3. Beispiele

Um das Penalty-Verfahren zu testen, wurde die Implementation mehrerer Geometrien durchgeführt. Das einfachste 2dimensionale Beispiel ist der in der Theorie sehr gut beschriebene umströmte Zylinder. Hier galt es den Übergang von laminarer Strömung zur Karmanschen Wirbelstraße zu beobachten. Im dreidimensionalen Fall wurde eine umströmte Kugel bei zwei verschiedenen Reynolds-Zahlen simuliert, die einer laminaren und einer turbulenten Strömung entsprechen. Weitere Simulationen waren nicht möglich, da

bis zur Entwicklung charakteristischer Strukturen in der Strömung bis zu 10 Tage Rechenzeit [1] vergehen. Das Verhalten des Codes an scharfen Kanten wurde mit einem umströmten Quadrat getestet und durch beliebige umströmte Polygone simuliert. Als Anwendungsbeispiel diente die Simulation eines umströmten VW Polo II (Abb. 3.7 und 3.8).

Name	$\mathbf{u}_0$	ν	R	Re
lowvisc	$0,1$	$1 \cdot 10^{-3}$	$0,15$	30
evenmore	$0,1$	$3 \cdot 10^{-4}$	$0,15$	100
medium	$0,1$	$5 \cdot 10^{-5}$	$0,15$	600

Tabelle 3.1.: Parameter für die Testreihe umströmter Zylinder

Wie erwartet zeigt sich der Übergang von laminarer Strömung zur Karmanschen Wirbelstraße in einem Bereich der Reynolds-Zahl zwischen 30 (Abb. 3.3) und 100 (Abb. 3.4), was mit dem von Whitaker [Whi68] angegebenen Werte von 60 korrespondiert. Auch ein Vergleich der Daten für den Druck zeigt sich eine gute Übereinstimmung mit theoretischen Überlegungen.

Bei der Simulation höherer Reynolds-Zahlen treten immer dann Probleme auf, wenn die Viskosität unter einen kritischen Wert fällt, nämlich genau dann, wenn die numerische Viskosität größer wird als die physikalische. Die Reynolds-Zahl wird dann ausschließlich von der numerischen Dissipation bestimmt. Um diese höheren Reynolds-Zahlen zu simulieren, müssen die Geschwindigkeit und die charakteristische Länge verändert werden, was wiederum einen Einfluß auf das CFL hat.

Von besonderem Interesse ist ferner die Simulation der rotierenden Walzen (Abb. 3.6), da hier die Geschwindigkeit am Rand der Walze nicht Null ist. Erwartungsgemäß lösen sich in der Simulation Wirbel ab, so daß auch dieser Funktionstest als bestanden betrachtet werden kann.

Neben den runden Testobjekten wurde auch ein Quadrat (Abb. 3.9 und 3.10) umströmt, sowie die oben erwähnte Simulation eines umströmten Autos durchgeführt.

Auch wenn es für diese Simulationen keine theoretischen Überlegungen gibt, so zeigt sich dennoch das ein Verhalten, das intuitiv zu erwarten ist. An den Kanten der Objekte lösen sich (bei hinreichend hoher Reynolds-Zahl) Wirbel ab. Außerdem ist zu erkennen,

[1] auf einer Compaq Alpha 800 MHz

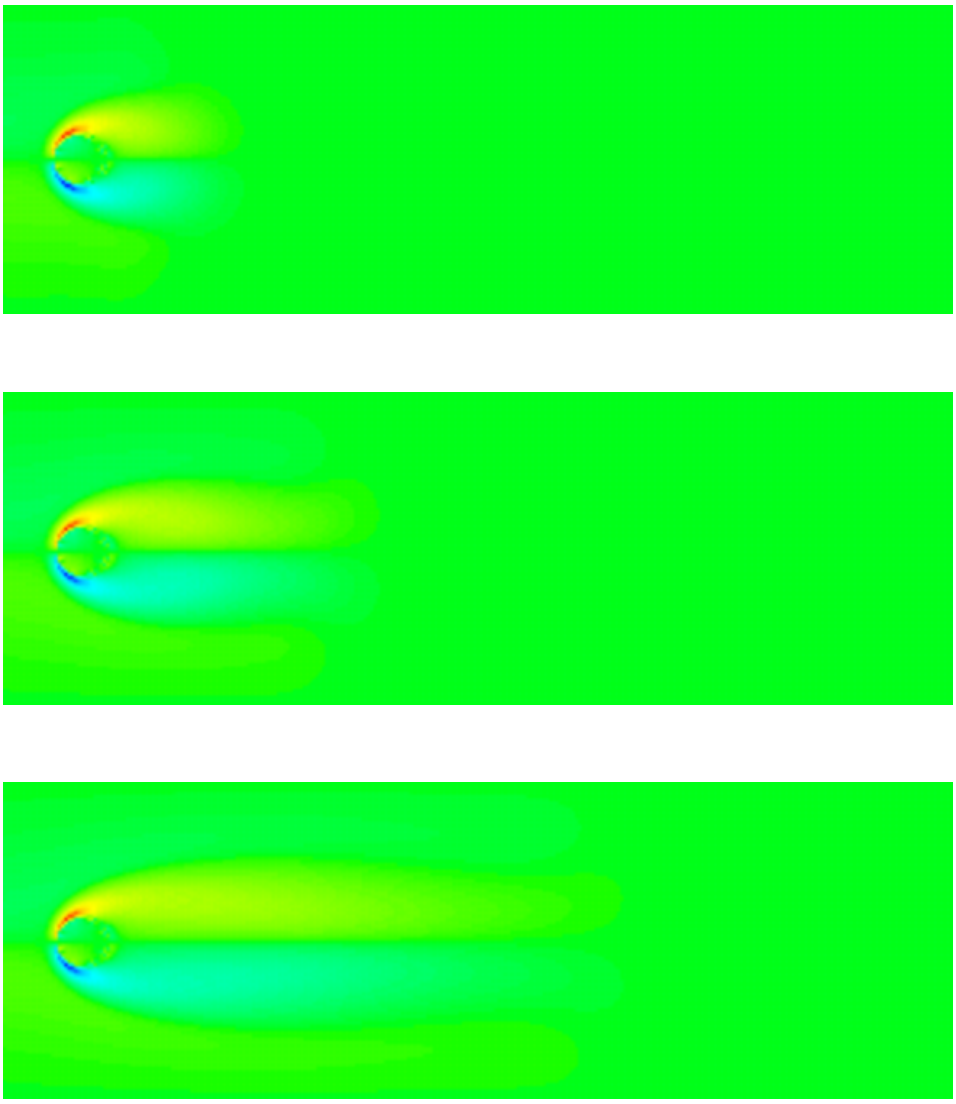


Abbildung 3.3.: *Vortizität für den Parameter "lowvisc", bei* $t = 10$, $t = 20$, $t = 40$

daß auch die Behandlung von Ecken keine Probleme mit sich bringt, da es hier keinerlei Instabilitäten auftreten.

Für 3dimensionale Strömungen ist hier der turbulente Fall einer umströmten Kugel gezeigt (Abb. 3.11). Der turbulente Charakter tritt dabei sehr deutlich hervor, insbesondere die Ausbildung von Strukturen in der Vortizität ist gut zu erkennen.

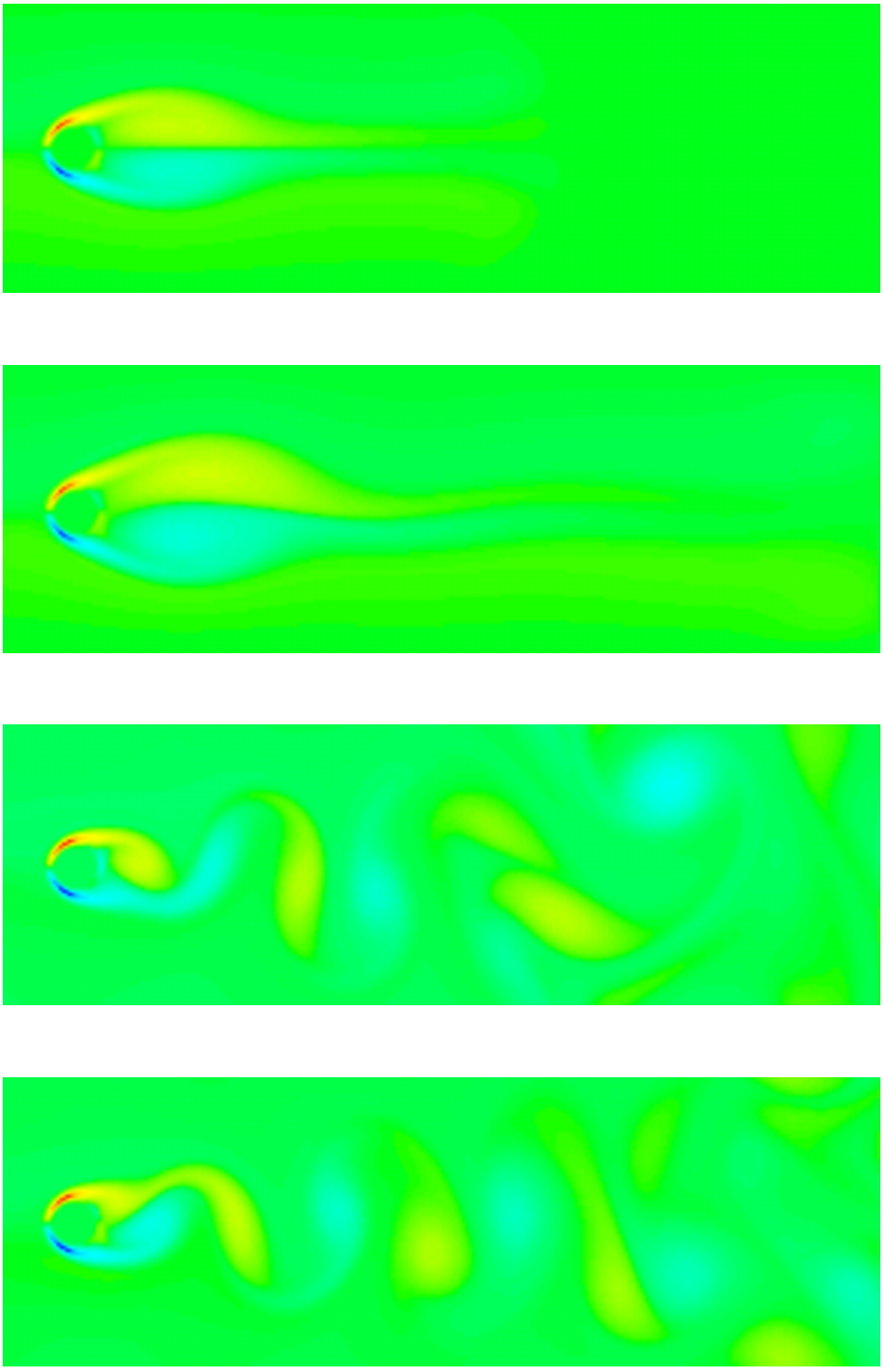


Abbildung 3.4.: *Vortizität für Parameter "evenmore", bei* $t = 40$, $t = 80$, $t = 140$, $t = 180$

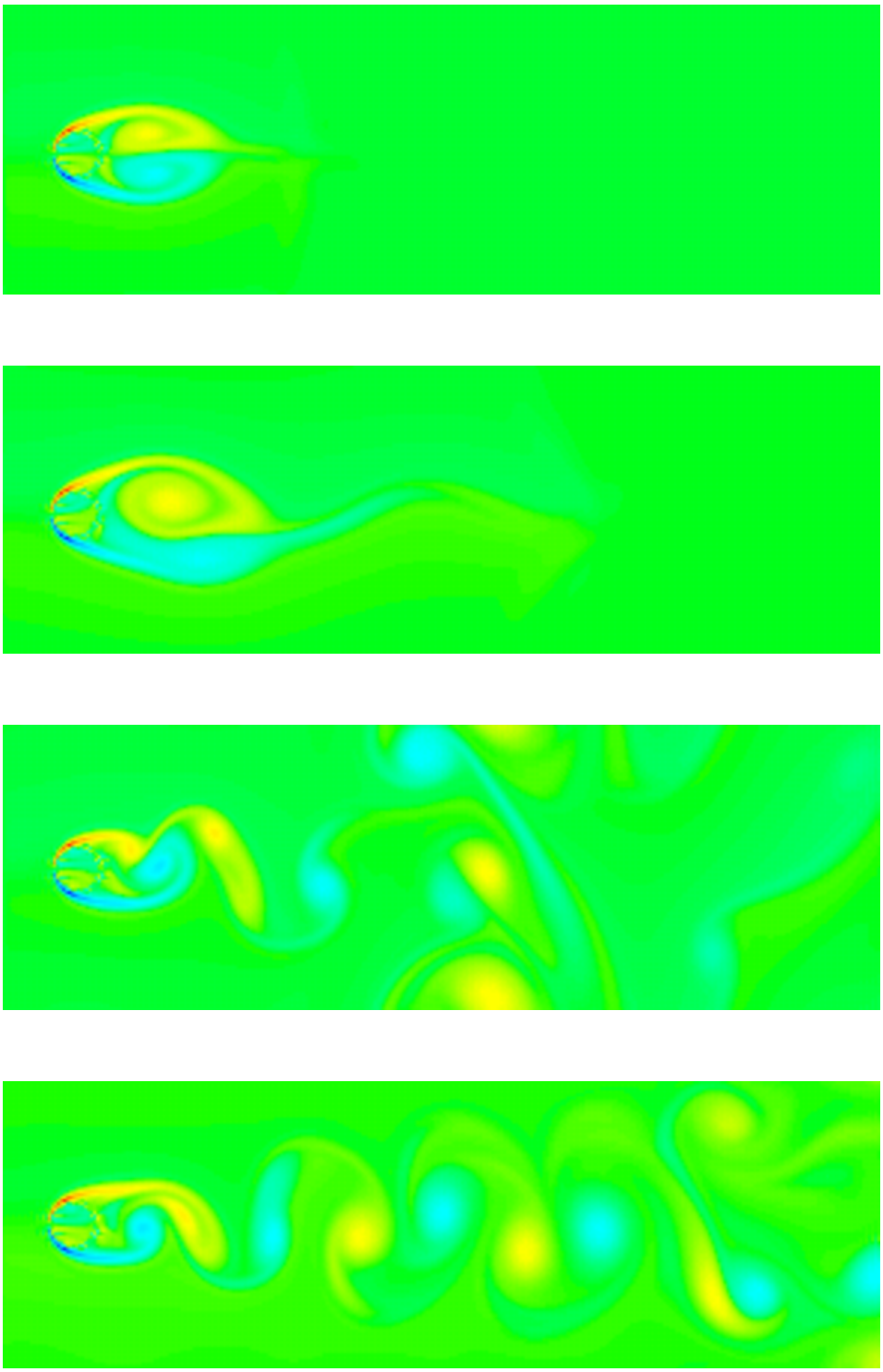


Abbildung 3.5.: *Vortizität für Parameter "medium", bei* $t = 20$, $t = 40$, $t = 80$, $t = 150$

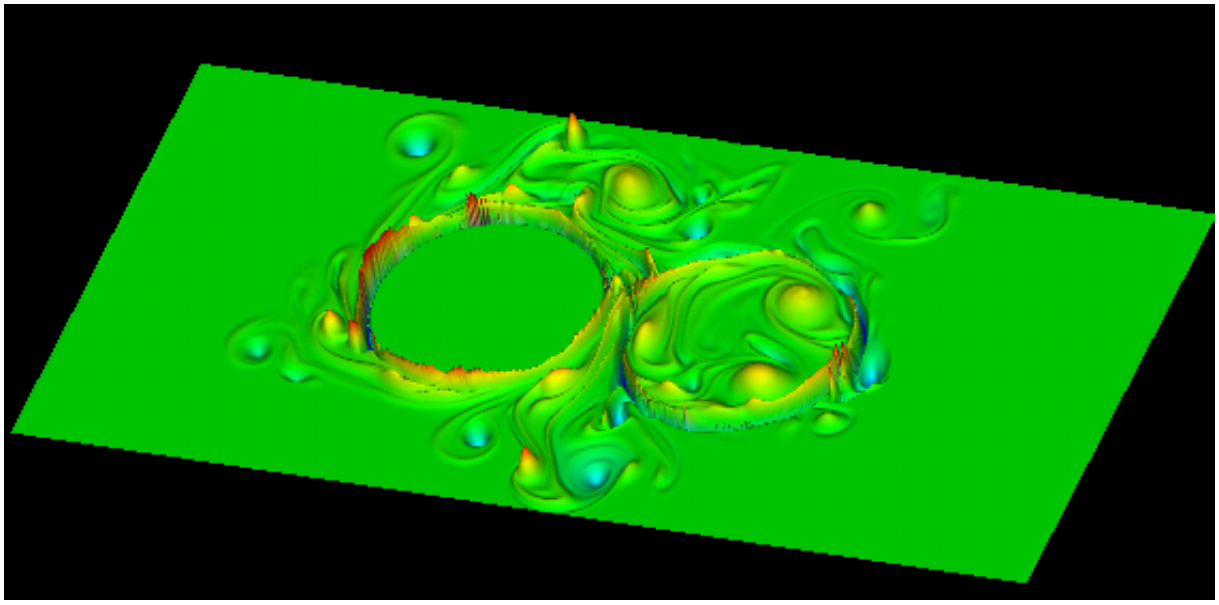


Abbildung 3.6.: *Vortizität gegensinnig rotierende Walzen*

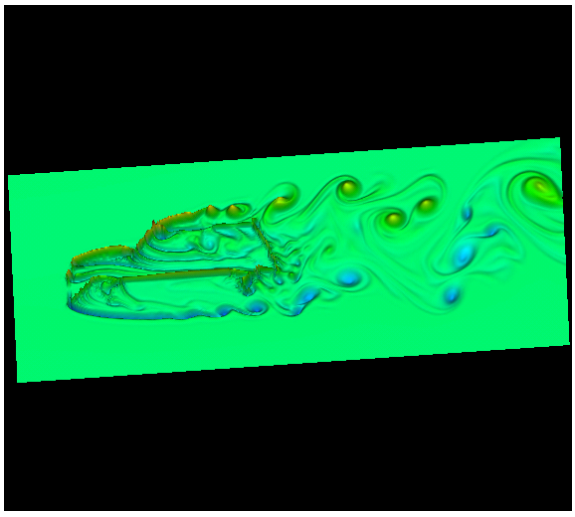


Abbildung 3.7.: *Vortizität der Umströmung eines Autos, nach 5s bei* $Re = 6.8 \cdot 10^6$

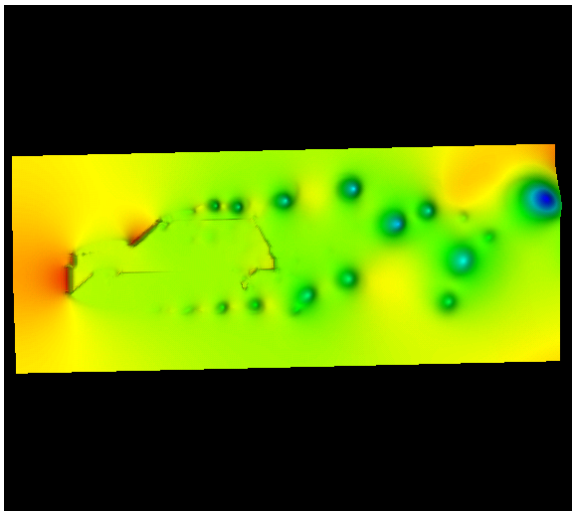


Abbildung 3.8.: *wie links, allerdings Darstellung des Druck*

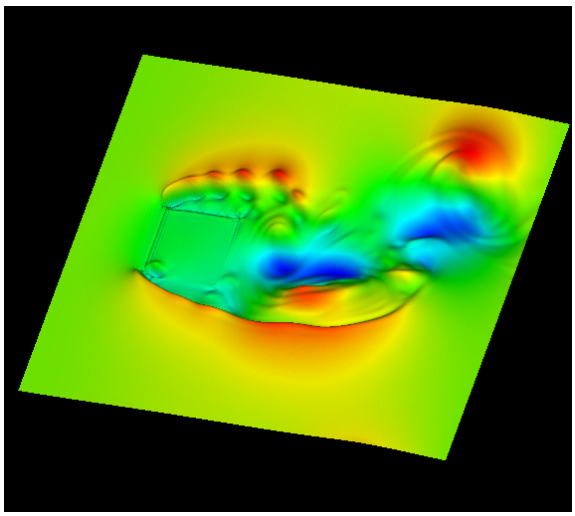


Abbildung 3.9.: u_x *eines umströmten Quadrates,* $Re = 100.000$

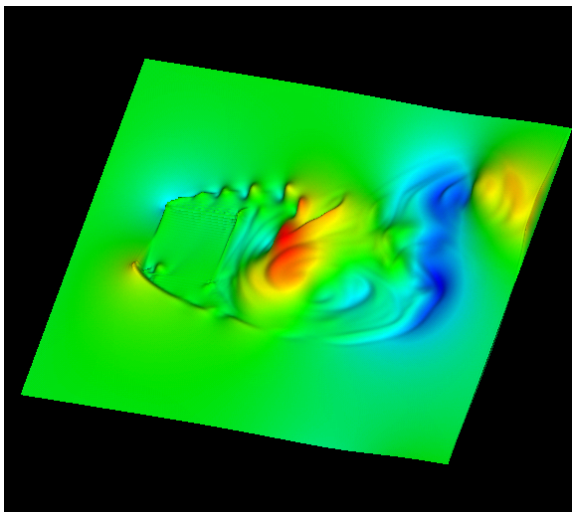


Abbildung 3.10.: u_y *eines umströmten Quadrates,* $Re = 100.000$

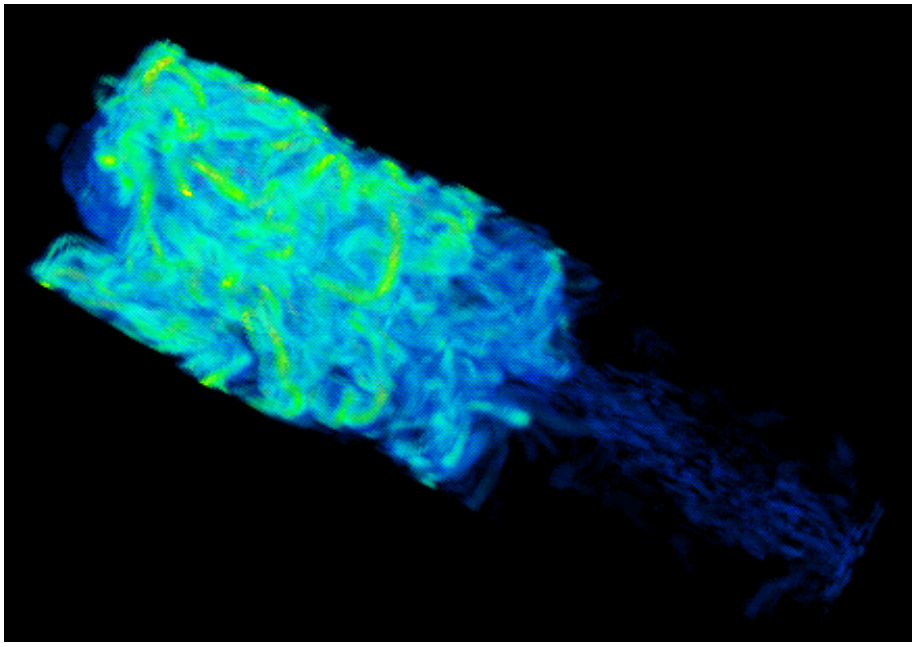


Abbildung 3.11.: *Vortizität einer umströmten Kugel* $Re = 25.000$*, die Kugel ist links oben an der Spitze*

4. Gitterverfeinerung

4.1. Grundlagen der adaptiven Gitterverfeinerung

Die Bilder aus den Testfällen für die Lösungsverfahren zeigen sehr deutlich, daß bei der numerischen Lösung von DGL Gebiete mit niedrigen und hohen zeitlichen Änderungen vorhanden sind. Bei den Festgitterverfahren bestimmen die Gebiete hoher Geschwindigkeit und hoher Gradienten die Auflösung auf dem gesamten Gebiet. Ein Weg aus dem Dilemma, daß die laufzeitbestimmenden Schrittweiten als globale Größen durch lokale Maxima bestimmt wird, ist die *Adaptive Gitterverfeinerung* (abgekürzt: AMR), wie sie von Berger und Colella ([BC89]) vorgeschlagen wurde.

Dieses Kapitel widmet sich in erster Linie der Implementation der beschriebenen Verfahren in den AMR-Code und den dabei auftretenden Problemen. Die Konzepte von AMR wurden schon im Kapitel 2.2 eingeführt, eine genauere Beschreibung der Programmstruktur findet sich in Anhang A.

Schwierigkeiten können sich an verschiedenen Stellen ergeben. Zu den Problemen, die sich auch ohne Nutzung der Gitterverfeinerung einstellen, zählen die Strukturinkompatibilitäten zwischen Programm und Rechnung (wie es bei der PM II -Implementation der Fall ist). Bei der Nutzung von AMR treten die meisten Probleme an den Rändern und durch unterschiedliche Zeitschritte auf verschiedenen Leveln auf.

4.2. Implementation

Häufig funktionieren Verfahren, die für das höchste Level (d.h. keine Gitterverfeinerung) in AMR implementiert wurden, ohne weiteres auch auf tieferen Leveln (mit Gitterverfei-

nerung). Bezüglich der Programmstruktur und der Randbedingungen sind jedoch einige fundamentale Überlegungen zu den benutzten Feldern erforderlich. Diese werden im folgenden für die verschiedenen Verfahren dargestellt.

4.2.1. RK

Für die Methode RK existiert bereits sowohl im Zweidimensionalen als auch im Dreidimensionalen [GMG98] eine Implementation für AMR. Da die AMR-Programmstruktur genau für diese Erfordernisse ausgelegt ist, treten hier auch keine strukturellen Probleme auf.

Als Kriterien für die Gitterverfeinerung wurden gewählt

$$10 \cdot \omega \cdot \Delta x \quad > \quad \epsilon \tag{4.1}$$

$$\left(\partial_x^2 + \partial_y^2\right) \omega \cdot (\Delta x)^3 \quad > \quad \epsilon \tag{4.2}$$

Neben dem Kriterium "große Wirbelstärke" (Gl. 4.1) tritt hier das sogenannte Deformationsmatrix-Kriterium (Gl. 4.2) auf, welches auf deutliche Änderungen in der Lösung reagiert.

4.2.2. PM II

Schon bei der Einführung des PM II -Verfahrens wurde festgestellt, daß die Programmstruktur geändert werden muß, um einen zweigeteilten Zeitschritt nutzen zu können. Zusätzlich hat sich aber auch der Versuch, PM II adaptiv zu realisieren, ohne die Struktur des Verfahrens zu ändern, als naiv erwiesen, da im Poisson-Schritt folgender Term steht.

$$\Delta p = \nabla \cdot \mathbf{n} + \frac{\nabla \cdot \mathbf{u}^n}{\Delta t} \tag{4.3}$$

Ursprünglich wurde die rechte Seite direkt in der Routine `poi`, in der auch die Poisson-Gleichung gelöst wird, berechnet. Da aber der Druck im gesamten Gitter inklusive Rand gelöst werden soll, würden durch die Divergenzbildung auch Punkte von $\mathbf{n}$ und $\mathbf{u}$ benötigt, die außerhalb des verfeinerten Gitters liegen. Daher muß die rechte Seite in ein eigenes Feld gesetzt werden, das seine Ränder aus übergeordneten Leveln erhält. Dabei ergibt sich aber ein weiteres Problem: Auf Grund der unterschiedlichen Zeitschritte Δt auf verschiedenen

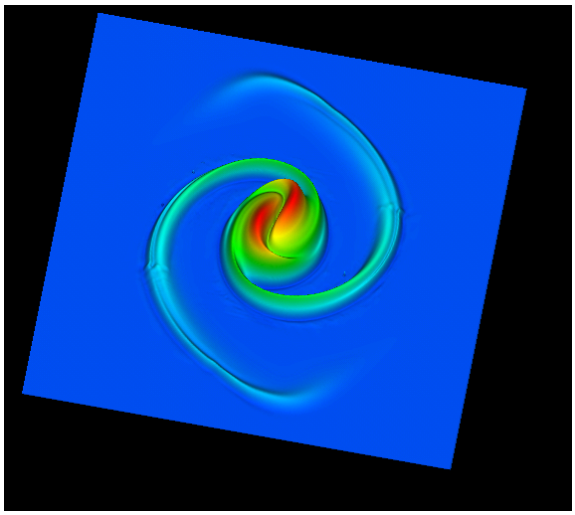


Abbildung 4.1.: *Direkte Implementation des PM II -Codes, ω zur Zeit $t = 50$*

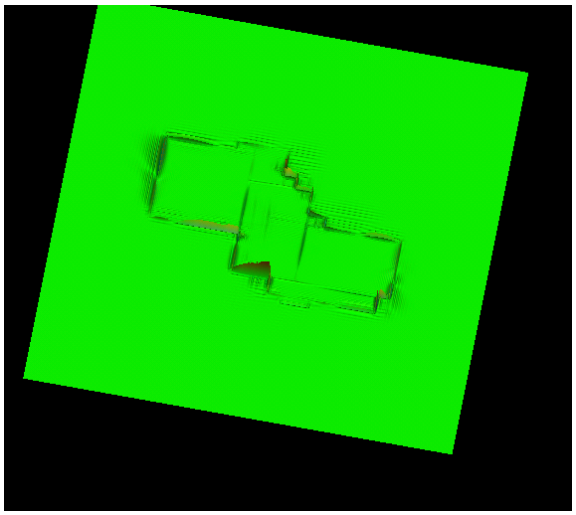


Abbildung 4.2.: *Direkte Implementation des PM II -Codes, div2 zur Zeit $t = 10$*

Leveln muß die rechte Seite in zwei Felder aufgeteilt werden, da sonst Diskontinuitäten auftreten würden, die eine korrekte Randbetrachtung unmöglich machen.

$$\Delta p = \underbrace{\nabla \cdot \mathbf{n}}_{=\text{div1}} + \frac{1}{\Delta t} \underbrace{\nabla \cdot \mathbf{u}^n}_{=\text{div2}} \tag{4.4}$$

An dieser Stelle muß bestimmt werden, welche Randroutinen auf tieferen Leveln aufgerufen werden müssen, welche Ränder also nach einem Zeitschritt mit Daten aus höheren Leveln aktualisiert werden müssen. Ganz wesentlich ist es, die normalen Randroutinen für die aktuelle Geschwindigkeit (analog `omboundary`) und den Druck (analog `psiboundary`) aufzurufen. Der obigen Überlegung zu Folge müssen auch div1 und div2 aktualisiert werden. Nicht aktualisiert werden dagegen die zwischengespeicherten Geschwindigkeiten für den Runge-Kutta-Schritt. Das Kriterium zur Gitterverfeinerung wurde anhand des RK-Codes und des mitberechneten ω bestimmt.

Erste Testläufe dieser Implementation entsprachen den Erwartungen, da für kleine Zeiten die Ergebnisse (Abb. 4.1) weitgehend mit den schon vorgestellten, nicht verfeinerten Testläufen übereinstimmten und auch das Regridding (die Neuerzeugung von Gittern) unproblematisch verlief. Für spätere Zeiten zeigten sich aber Artefakte in der Lösung, die zu noch späteren Zeite eine deutlich falsche Lösung ergaben. Der Grund dafür fand sich im div2-Feld (Abb. 4.2), das vom Beginn der Simulation an Sprünge an den Rändern der

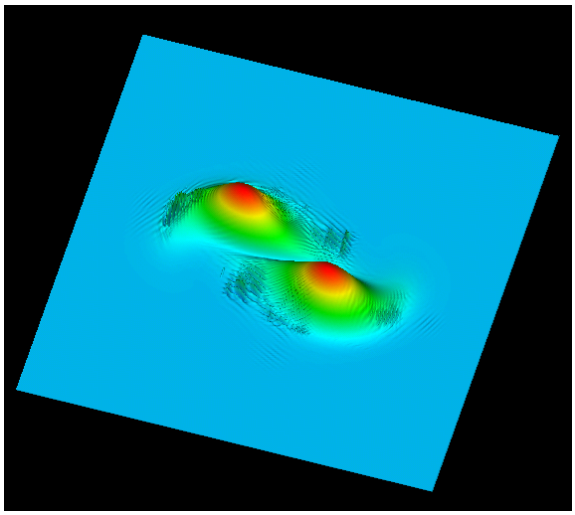


Abbildung 4.3.: *PM II -Codes, kein div2-Feld, ω zur Zeit $t = 50$*

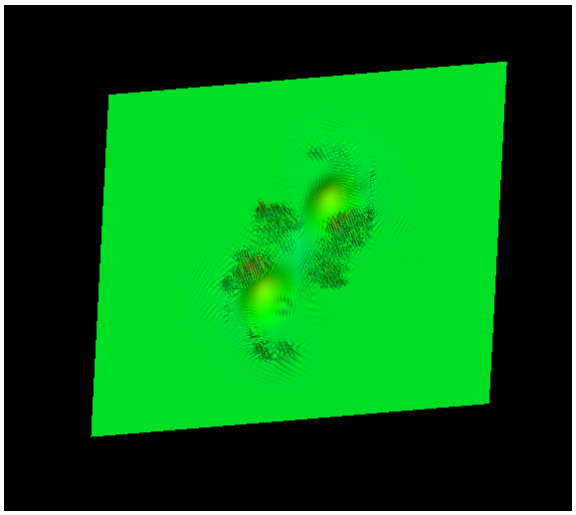


Abbildung 4.4.: *PM II -Codes, kein div2-Feld, div1 zur Zeit $t = 10$*

tieferen Level zeigt, die dann zu Oszillationen führen.

Ein Vergleich dieser Sprünge bei verschiedenen Auflösungen zeigt, daß die Stufen mit zunehmender Auflösung kleiner werden. Dies deutet daraufhin, daß kleine Stufen in der Geschwindigkeit durch Divergenzbildung zu großen Stufen im div2-Feld führen, also die Genauigkeit unzureichend ist.

Die ersten Versuche, diesen Fehler zu beheben, basierten auf der Vermutung einer fehlerhaften Randaktualisierung. Daher wurde wahlweise die Aktualisierung des Randes abgeschaltet beziehungsweise der Rand auf tieferen Leveln gelöscht. Anschließend wurden die Randwerte neu berechnet, wobei entweder für die äußeren Punkte einseitige Ableitungen verwendet wurden oder der Druck nur im physikalischen Gebiet gelöst wurde. Da jedoch keiner der Lösungsansätze eine Verbesserung brachte, liegt das Problem offenbar einzig darin, daß die Werte von div2 zu klein gegenüber der numerischen Auflösung sind. Dabei ist zu berücksichtigen, daß div2, also $\nabla \cdot \mathbf{u}^n$, nahezu Null ist und die Größenordnung ausschließlich von der Genauigkeit des Projektionsverfahrens abhängt. Trotz seines kleinen Wertes spielt div2 eine entscheidende Rolle für die Funktion von PM II .

Der nächste, prinzipiell einfachste Ansatz bestand darin, das div2-Feld völlig wegzulassen, wobei natürlich kleinere Ungenauigkeiten in Kauf genommen werden müssen. Die damit gewonnenen Ergebnisse (Abb. 4.3 und 4.4) waren jedoch unbrauchbar, denn

die Divergenz schnellte in die Höhe und es wurde somit eine völlig andere physikalische Fragestellung gelöst. Dieses Verhalten war allerdings auch zu erwarten, da das div2-Feld zusätzlich in PM II eingebaut wurde, um die Divergenz zu kontrollieren. Hier stieg allerdings die Divergenz noch deutlicher an als in nicht verfeinerten Gittern.

Um dieses Problem der Divergenz zu lösen, bestand der nächste Schritt darin, nach dem Regridding wieder eine divergenzfreie Lösung vorzugeben. Das Konzept ähnelte demjenigen, das beim Initialisieren der Daten verwendet wird: Aus ω wird Ψ berechnet und daraus wiederum die Geschwindigkeit, diesmal allerdings ausschließlich auf den tieferen Leveln und nur beim Regridden. Da Ψ aus Rechenzeitgründen nicht auf allen Leveln zur Verfügung steht, werden von Neumann-Randbedingungen benötigt, die sich aus der Geschwindigkeit vom höheren Level ergeben.

Trotz des vielversprechenden Konzepts traten auch hier massive Schwierigkeiten auf. An den Rändern zwischen verschiedenen Leveln entstanden in der Geschwindigkeit lokale Extrema (Abb. 4.5), die nicht im eigentlichen Randgebiet, sondern im physikalischen Gebiet lagen. Auf Grund ihrer Position kann ausgeschlossen werden, daß sie durch die Randaktualisierung entstehen. Es zeigte sich, daß die Ursache in einem Genauigkeitsfehler lag: ω war vollkommen glatt, ebenso das daraus resultierende Ψ-Feld ebenso. Für die zweite Ableitung von $\mathbf{u}$ galt dies allerdings nicht mehr. Daher wurde die Genauigkeit für alle Schritte auf 4. Ordnung erhöht. Jedoch führte auch dieser Maßnahme nicht zum Erfolg, denn die Teilschritte in den Routinen `poi` und `init` sind zwar 4. Ordnung, nicht aber das eigentliche Lösungsverfahren CWENO. Dies hat zur Folge, daß nach hinreichend langer Zeit wieder Extrema erscheinen.

Nach diesen Ergebnissen wird deutlich, daß sich PM II nicht so intuitiv implementieren läßt wie anfangs angenommen. Für die aufgezeigten Probleme des Verfahrens ergeben sich im Moment auch keine Lösungen, weswegen es sich empfiehlt, ein anderes Lösungsverfahren für das Problem zu finden.

4.2.3. Mix

Aus den oben genannten Gründen hat sich das PM II -Verfahren als ungeeignet für Gitterverfeinerung gezeigt, weshalb ein neues geschwindigkeitsbasiertes Verfahren notwendig wurde. Es handelt sich dabei um das sogenannte Mix-Verfahren, das bereits oben (Kapitel 2.5.3) beschrieben worden ist. Für dieses Verfahren wird die Geschwindigkeit jeweils

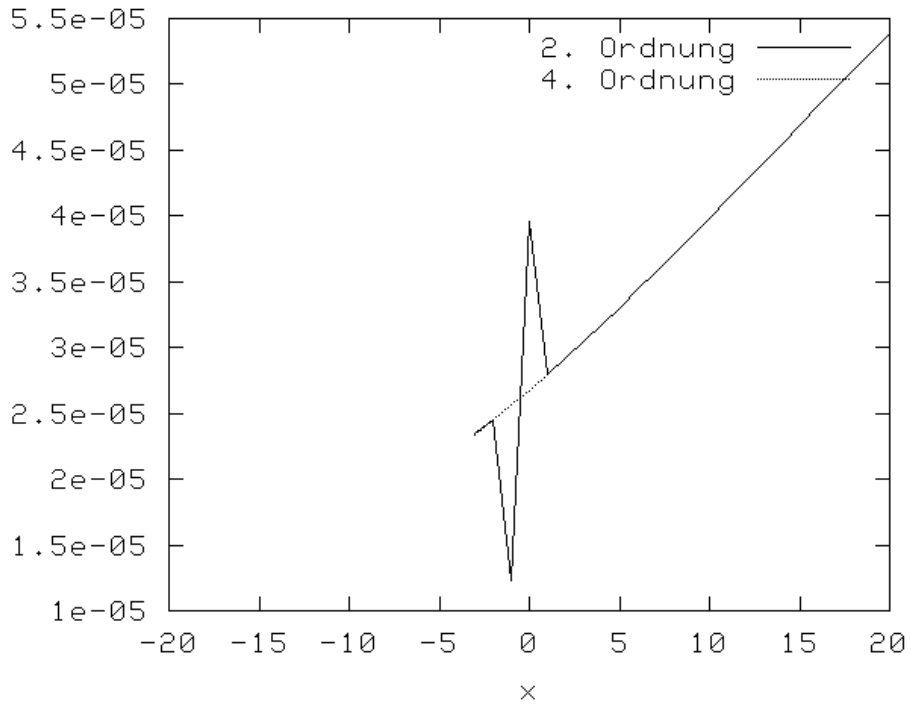


Abbildung 4.5.: *Vergleich zwischen 2. und 4. Ordnung Genauigkeit bei PM II mit* Ψ*-Zwischenschritt*

divergenzfrei neu berechnet.

Die Implementation des Mix-Verfahrens verläuft erst einmal bezüglich ω und Ψ zunächst analog zu der des RK-Verfahrens. Durch die Verwendung eines staggered Grids existiert nicht für jeden Punkt des Gitters ein Wert für die Geschwindigkeit. Da allerdings die Geschwindigkeit in jedem Zeitschritt neu berechnet wird, kann die Randbetrachtung völlig außer Acht gelassen und die Geschwindigkeit als temporäres Feld betrachtet werden.

Trotz des geschwindigkeitsbasierten Zeitschritts ist Mix dennoch ein Wirbelstärke-Verfahren, so daß viele Probleme, wie zum Beispiel das Verfeinerungskriterium analog zu RK gelöst werden können.

Bei der Betrachtung der Daten fällt auf (Abb. 4.6 bis 4.11), daß **u** Kanten aufweist, obschon ω und Ψ glatt sind. Dies läßt sich aber durch das staggered Grid erklären, da hier nicht für alle Randpunkte Werte existieren. Die Daten für die Enstrophie (Abb. 4.11) zeigen, daß ein Mix-Code mit einer Vefeinerung ähnlich gute Ergebnisse erzielt wie ein Code mit doppelter Auflösung. Dabei wird allerdings nicht das gesamte Gebiet in hoher

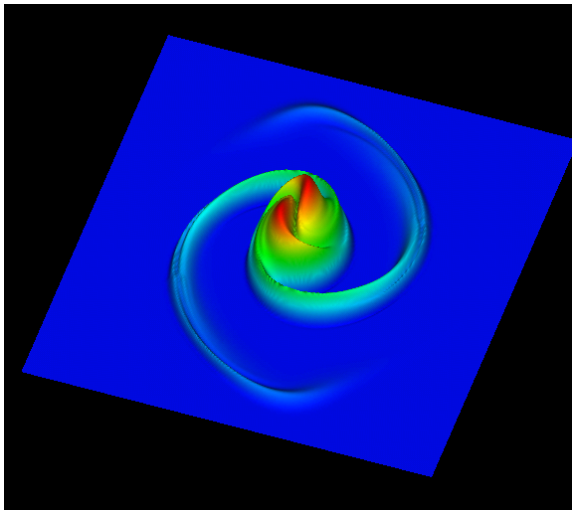


Abbildung 4.6.: *Mix mit einem Level, Vortizität zur Zeit* $t = 50$

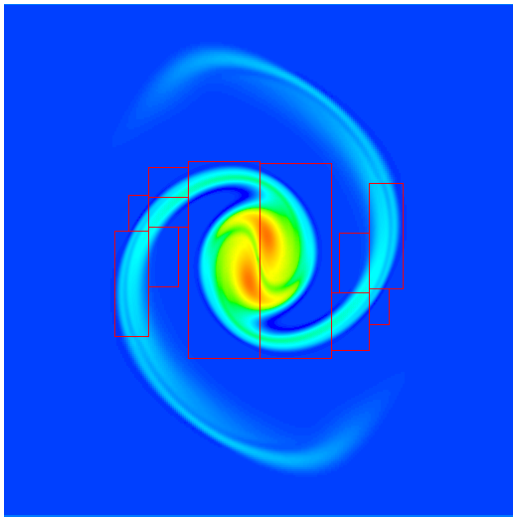


Abbildung 4.7.: *wie links, aber mit Grenzen der Gitter*

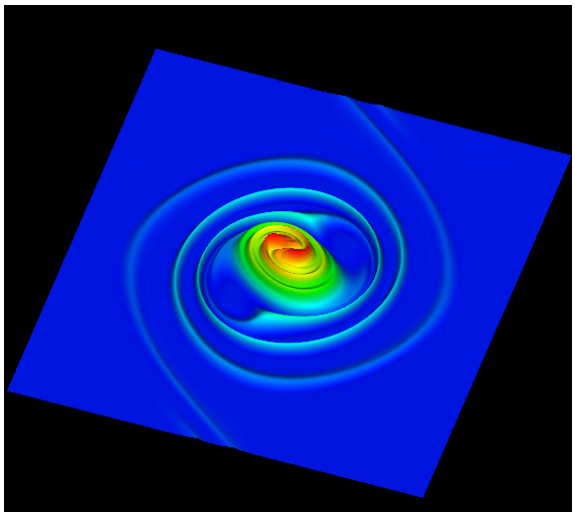


Abbildung 4.8.: *Mix mit einem Level, Vortizität zur Zeit* $t = 100$

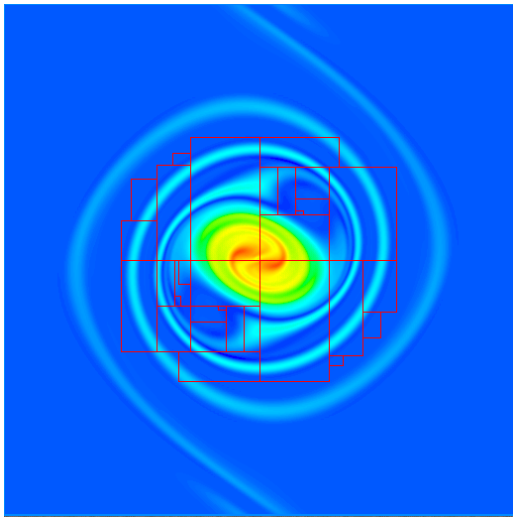


Abbildung 4.9.: *wie links, aber mit Grenzen der Gitter*

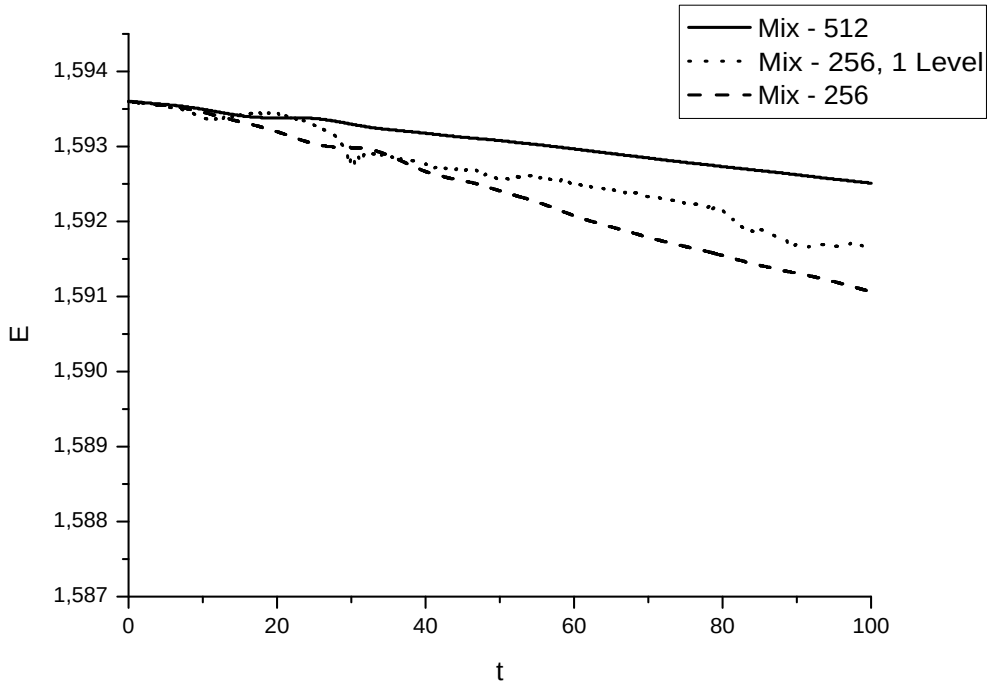


Abbildung 4.10.: *Energie für Mix, Auflösung* 256×256 *(mit und ohne AMR),* 512×512

Auflösung gerechnnet, weshalb sich ein deutlicher Rechengeschwindigkeitszuwachs ergibt. Die Kurve des zeitlichen Verlaufs der Energie ist nicht glatt, da für die Berechnung die Geschwindigkeit benutzt wird, die auf einem staggered Grid liegt. Bei dieser Berechnung ist der Interpolationsalgorithmus für die Randpunkte nicht korrekt, daher wird der Fehler für die Energie mit der Zahl tieferer Level größer.

4.2.4. Penalty-Methoden

Mit dem funktionierenden Mix-Verfahren auf adaptiver Basis lassen sich nun auch beliebige Geometrien adaptiv implementieren. Der erste Schritte dazu ist die Verbindung von Mix-Verfahren und Penalty-Methoden.

Die Problematik bei der Berechnung von Kräften für das Mix-Verfahren liegt im staggered Grid. Um Abstände, Forcing- und Interpolationspunkte zu berechnen, muß unbedingt die Verschiebung um eine halbe Gitterzelle entlang der Diagonalen zwischen

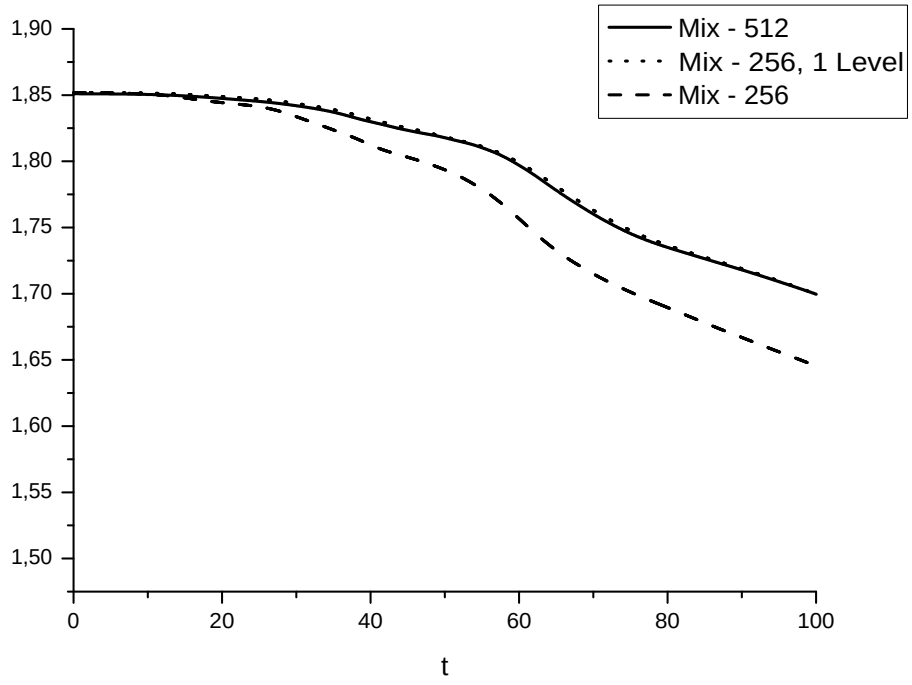


Abbildung 4.11.: *Enstrophie für Mix, Auflösung* 256×256 *(mit und ohne AMR),* 512×512

Geschwindigkeits- und Ortsraum beachtet werden. Die Implementation verlief problemlos und lieferte dieselben Ergebnisse wie die nicht verfeinerten Interpolationsroutinen.

Der nächste Schritt in der Implementation ist die Teilung der Liste aller Punkte, an denen die Penalty-Methoden einsetzen. Für Grids auf tiefen Leveln werden nur Punkte beachtet, die physikalisch innerhalb des Gitters liegen. Bei der Berechnung der Punkte, also bei der Neuinitialisierung eines Gitters, wird zwar die Prozedur für alle Punkte durchlaufen, diejenigen außerhalb des Gitters werden aber verworfen.

Die Bestimmung des Kriteriums für die Gitterverfeinerung verläuft prinzipiell so wie bei den RK-Codes, allerdings mit einigen Einschränkungen. Zuerst werden für den Verfeinerungsschritt alle Punkte in der Nähe des umströmten Körpers als kritische Punkte markiert. Somit sind beim Start der Simulation die Gebiete um den Körper gut aufgelöst, obwohl sich zu diesem Zeitpunkt noch keine Strukturen in der Strömung ausgebildet haben, sondern nur die laminare Anfangsbedingung existiert. Die zweite Einschränkung betrifft die Verfeinerung am Rande des physikalischen Gebiets. Da hier andere Rand-

bedingungen für die Poisson-Gleichung als im Inneren angenommen werden (konstanter Einstrom statt Kontinuität der Lösung), wird der Einfachheit halber eine Verfeinerung direkt am Rand verhindert. Dadurch wird eine Vermischung innerer und äußerer Randbedingungen für die Poisson-Gleichung unterdrückt, indem eventuelle kritische Punkte in der Nähe des physikalischen Randes als nicht kritisch betrachtet werden. Im allgemeinen wird das Simulationsgebiet ohnehin so gewählt, daß das Hauptgeschehen im Zentrum abläuft und am Rand möglichst laminare Strömungen auftreten.

In den Abb. 4.12 bis 4.14 sind 2dimensionale Simulationen umströmter Zylinder zu sehen. Ein wichtiger Unterschied zu den in Kapitel 3.3 gezeigten Simulationen muß hier beachtet werden, daß hier eine Kanalgeometrie angewendet wird. Der Einfluß der Ränder auf die Simulation wird besonders deutlich beim Vergleich von Abb. 4.12 und 4.14. Diese Simulationen liefen mit identischen Parametern, bis auf die Breite des Kanals. Daher sind die Ergebnisse der ersten Simulation turbulenter. Um den Einfluß der Ränder völlig auszuschalten müsste die Kanalbreite noch erheblich größer sein als der Radius. Bei den hier gewählten Parametern ist das Verhältnis von Zylinderdurchmesser und Kanalbreite 5 bzw. 15. Gegen größere Kanalbreiten spricht hier allerdings die Rechenzeit.

4.2.5. Beurteilung der Methode

Mit der Mix-Methode steht ein Verfahren zur Verfügung, das sowohl die Simulation ohne Geometrien durch die Möglichkeit Adaptiver Gitterverfeinerung verbessert, als auch

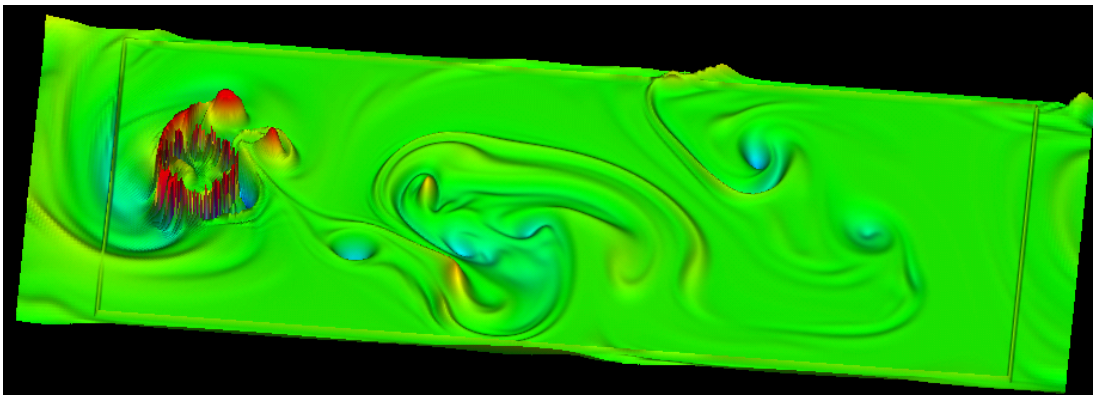


Abbildung 4.12.: *Vortizität eines umströmten Zylinders,* $Re = 4000$*, Zylinderradius* $R = 0,2$*, Kanalbreite* $b = 2$*, eine Verfeinerungsstufe*

schnelle, hochaufgelöste Simulationen umströmter Geometrien ermöglicht. Damit ist es durch die in dieser Arbeit entwickelte Methode problemlos möglich beliebige Geometrien im Rahmen von AMR zu simulierten, was der Zielsetzung entspricht.

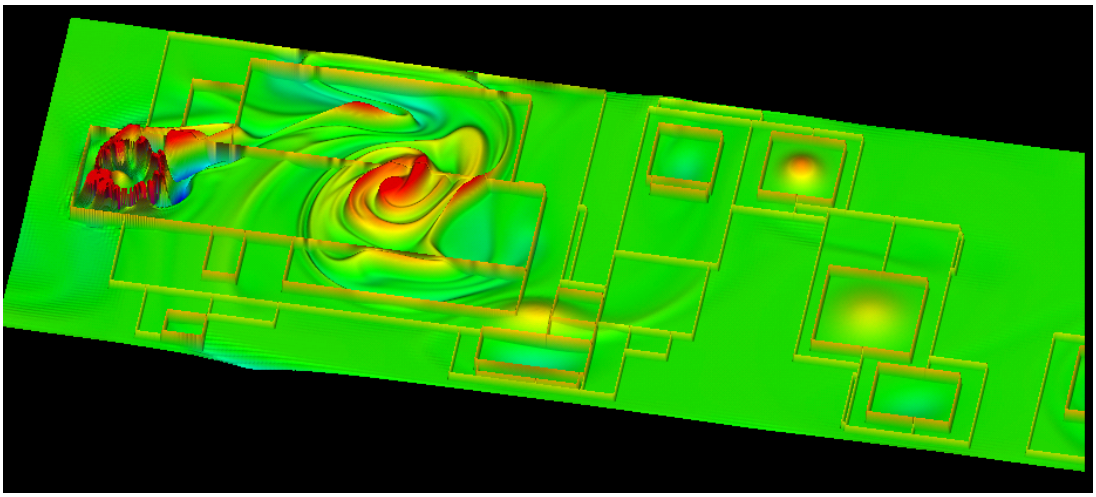


Abbildung 4.13.: *Vortizität eines umströmten Zylinders, $Re = 600$, Zylinderradius $R = 0,15$, Kanalbreite $b = 2$, zwei Verfeinerungsstufen*

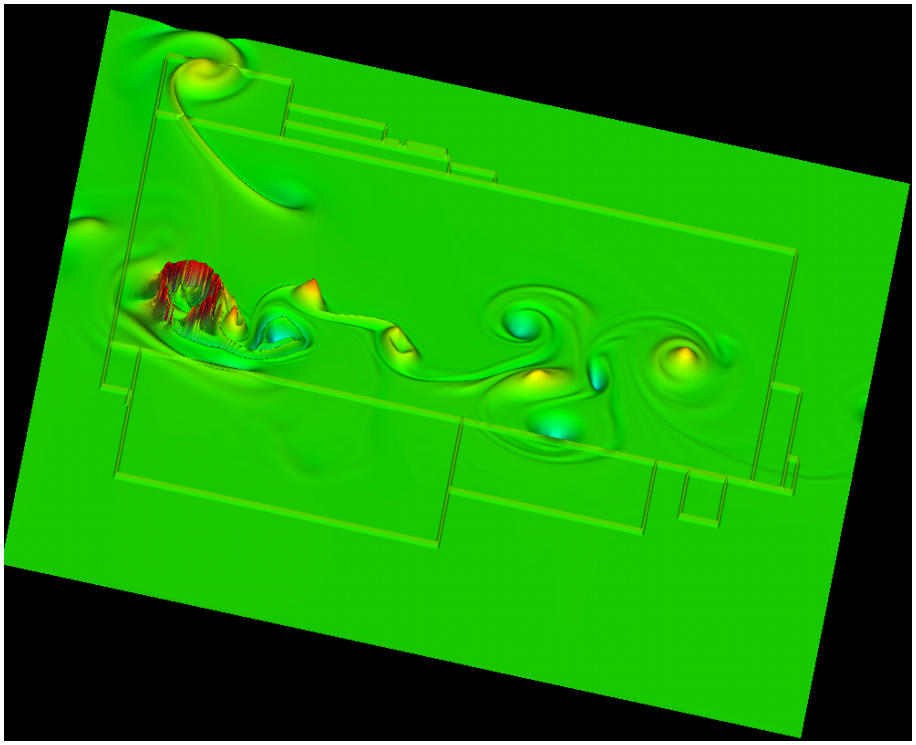


Abbildung 4.14.: *Vortizität eines umströmten Zylinders,* $Re = 4000$*, Zylinderradius* $R = 0,2$*, Kanalbreite* $b = 6$*, eine Verfeinerungsstufen*

Zusammenfassung

Die Ergebnisse dieser Arbeit zeigen, daß bei der Simulation der Navier-Stokes-Gleichung geschwindigkeitsbasierte Rechnungen mit CWENO-Verfahren deutliche Verbesserungen gegenüber Formulierungen mit Wirbelstärke und Stromfunktion bringen. Dabei ist die numerische Dissipation geringer als bei bekannten Wirbelstärke-Stromfunktionssimulationen. Auch die Auflösung starker Gradienten wird durch die Verwendung von CWENO optimiert. Das Mix-Verfahren beweist, daß der entscheidende Schritt hierbei in der Lösung der Navier-Stokes-Gleichung in der Geschwindigkeit liegt.

Aus den gewonnen Erkenntnissen läßt sich ein adaptives Verfahren entwickeln, das auf dem Mix-Code basiert. Es wird gezeigt, daß der intuitive Ansatz für Projektionsverfahren (PM II) sich nicht mit dem adaptiven Ansatz kombinieren läßt. Mit dem adaptiven Mix-Ansatz werden durch eine Verfeinerung Ergebnisse erzielt, die mit denen aus Simulationen mit doppelter Auflösung nahezu übereinstimmen, jedoch bis zu 50% Rechenzeit einsparen.

Bei der Simulation von Geometrien lösen die Penalty-Methoden die gestellten Probleme, wobei sich erst sehr spät Instabilitäten entwickeln. Die Wahl des Interpolationsverfahrens spielt bei den Ergebnissen keine nennenswerte Rolle, ebenso ist es irrelevant, ob das Innere von umströmten Körpern explizit behandelt wird. Bei der Simulation hoher Reynolds-Zahlen treten Schwierigkeiten auf, wenn die Viskosität kleiner gewählt wird als die "numerische Viskosität", die sich aus der numerischen Dissipation des Lösungsverfahrens ergibt. Hier wird die Strömung durch eine numerische und nicht mehr durch eine physikalische Reynolds-Zahl bestimmt.

Die Fusion von Penalty-Methoden, Mix-Verfahren und AMR scheint ein aussichtsreicher Ansatz für die Lösung komplexer Strömungsprobleme zu sein. Mit dieser Kombination sollten sich beliebige Umströmungsprobleme schnell und mit hoher Genauigkeit lösen

lassen. Mit dem bestehenden Code sind in Simulationen mit einer effektiven Auflösung von 1024×1024 Punkten realistisch durch Nutzung von AMR, wohingegen ohne AMR eine Auflösung von 512×512 Punkten auf normalen PCs schon einen enormen darstellt.

Die zukünftigen Entwicklungsschritte betreffen die Parallelisierung des Codes. Durch geeignete Multigrid-Methoden lassen sich die Poisson-Schritte vollständig parallelisieren und erlauben somit erhebliche Rechenzeitvorteile gerade bei 3dimensionalen Simulationen.

A. Struktur des AMR-Codes

Dieser Anhang dient einer detaillierten Beschreibung der Struktur des AMR-Rahmenprogramms.

A.1. Datenstruktur

Die grundlegende Datenstrukturen sind schon in Kap. 2.2 eingeführt worden, daher lege ich an dieser Stelle das Gewicht auf den Zusammenhang der Strukturen. Alle Strukturen sind in *C++* als Objekte ausgeführt.

Die *Matrix*, die die Daten enthält, ist ein als Template programmiertes Array. Um die Matrix mehrdimensional zu nutzen, muß sie vorher für jede Dimennsionalität instantiiert werden.

Das *Grid* enthält Arrays von Matrizen sowie alle Routinen, die mit der Berechnung des Zeitschrittes zusammenhängen.

Oberhalb des Grid ist das *Level* angesiedelt. Es enthält eine verkettete Liste der Gitter. Diese Liste wird mit Hilfe eines Iterators durchlaufen, um den Zeitschritt auf den Gittern durchzuführen.

A.2. Methoden

Jedes Grid enthält die notwendigen Methoden zur Durchführung eines Zeitschritt:

singlestep enthält die Berechnung der Nichtlinearität einer hyperbolische DGL und geht von $\omega^n \rightarrow \omega^{n+1}$.

poi löst elliptische Gleichungen $\Delta\Psi = \omega$.

omboundary, psiboundary verarbeiten die Randbedingungen für die entsprechenden Felder.

Zusätzlich wurde die Methode *Init* implementiert, die die Startwerte für die Matrizen vorgibt.

In den Level sind die Routinen definiert, die neue Gitter erzeugen können. Außerdem stellen sie die Routinen, um den Zeitschritt in den Grids aufzurufen und dabei elliptische Gleichungen korrekt zu behandlen.

Im Hauptprogramm befindet sich die hier abgedruckte Routine *integrate*.

```
for (int n = 0; n < RK_STEPS; n++) {
  //Iteration für Runge-Kutta
  cout << "RKSTEP␣=␣" << n+1 << endl;
  lev->singlestep(n); //Zeitschritt
  lev->better_om_boundary();
  //tauscht Werte benachbarter Gitter
  lev->schwarz_iteration(n); //löst Poisson-Gleichung
#ifdef RK_PM2
  lev->singlestep(n+RK_STEPS);
  //2. Teil des Zeitschritt für PM II
#endif
}
if (lev.next() != NULL) {
  lev++;
  lev->calc_default_boundary();
  //berechnet Randwerte für tiefere Level
  while (behind(lev())) integrate(lev,levels);
  //integriert tiefere Level
  lev->update();
  lev--;
}
lev->check_it(levels);
```

Diese Routine geht durch alle Level und ruft in ihnen die singlesteps auf. Daraufhin wird die Schwarz-Iteration durchgeführt, die eine richtige Lösung der Poisson-Gleichung zwischen benachbarten Gittern sicherstellt, indem sie die Lösungen in den Randgebieten solange austauscht, bis der Fehler unter eine definierte Schwelle sinkt.

A.3. Nutzung der Gitterverfeinerung

Bisher wurden nur die Methoden ohne Gitterverfeinerung dargestellt. Interessant ist jedoch die Möglichkeit, an beliebigen Stellen beliebig große feinere Gitter einzufügen (im Gegensatz zu Quadtree/Octtree-Verfahren, die nur symmetrische Unterteilungen erlauben). Das Ablaufschema läßt sich folgendermaßen skizzieren:

- Initialisiere Gitter auf dem obersten Level
- Suche kritische Punkte
- Bilde Rechtecke, die alle kritischen Punkte umfassen
- Initialisiere die tieferliegenden Level
- Führe den Zeitschritt auf dem obersten Level aus
- Führe n-mal folgende Schritte aus
  - Führe den Zeitschritt auf dem nächstem Level aus
  - Aktualisiere die Ränder des tieferen Level
- Gib Werte vom tieferen an das höhere Level weiter
- Prüfe nach einer bestimmten Zahl von Iterationen erneut, ob kritische Punkte vorliegen

In Abb. A.3 ist das Verfahren zu sehen, nach dem die Rechtecke minimaler Fläche, die alle kritischen Punkte umfassen, gebildet werden. Die Integration des feineren Gitters in das grobe Gitter ist in Abb. A.3 abgebildet.

Die Implementation eines Problems erfordert eine Routine, in der geprüft wird, ob ein bestimmter Punkt ein kritischen Punkt ist (`criterion`). Diese Routine ist ein Teil von `Grid`. Als Kriterium wird in der Regel ein starker Gradient gewählt, da hier die größten Änderungen in der Lösung der DGL auftreten.

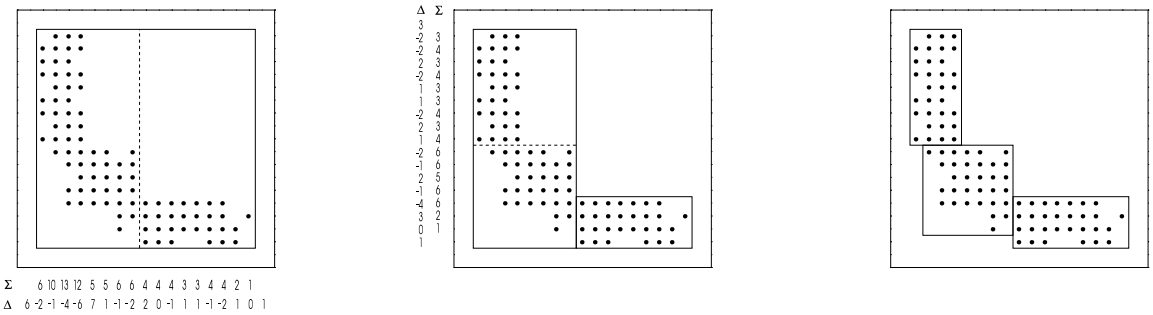


Abbildung A.1.: Bestimmung der Rechtecke aus den kritischen Punkten

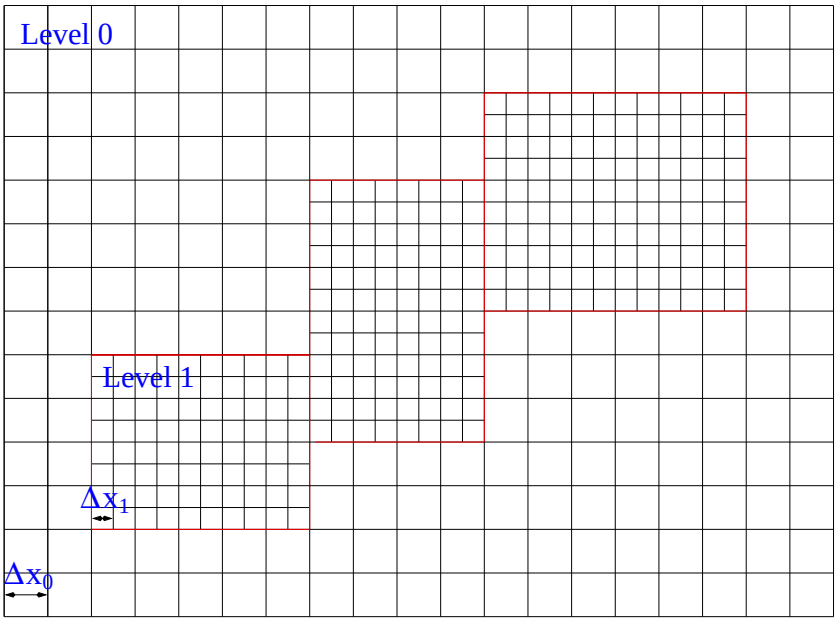


Abbildung A.2.: Integration der Gitter

B. Runge-Kutta-Verfahren

Wie schon in Kapitel 2.4 beschrieben, erfordert die Verwendung von Runge-Kutta-Verfahren im Zusammenhang mit AMR eine zusätzliche Überlegung. Statt der Halbierung des Zeitschritts kommen hier kleinere Einteilungen zur Anwendung. Im Programm wird dazu aus der zur Verfügung stehenden Aktualisierung des Randes ein Bruchteil $frac$ bestimmt, der mit der aktuellen Zeit im Runge-Kutta-Schritt zusammenhängt.

Um dies genauer zu analysieren, müssen die Zeitschritte auf den verschiedenen Leveln zerlegt werden. Auf dem obersten Level wird aus dem Zeitschritt

$$t^n \rightarrow t^n + \Delta t \tag{B.1}$$

dann in 3. Ordnung

$$t^n \rightarrow t^{(1)} \rightarrow t^{(2)} \rightarrow t^n + \Delta t \tag{B.2}$$

Auf dem nächsttieferen Level wird (ohne Runge-Kutta) dann folgender Zeitschritt ausgeführt

$$t^n \rightarrow t^n + \frac{1}{2}\Delta t \rightarrow t^n + \Delta t \tag{B.3}$$

Wenn sich auf dem höchstem Level eine Änderung von $u^{n+1} = u^n + \Delta u$ ergibt, muß die Randanpassung auf dem tieferen Level derart erfolgen

$$u^n \rightarrow u^n + \frac{1}{2}\Delta u \rightarrow u^n + \Delta u = u^{n+1} \tag{B.4}$$

Wird das Runge-Kutta-Verfahren eingesetzt, müssen nun weitere Zwischenschritte eingefügt werden.

$$u^n \rightarrow u^n + frac \cdot \frac{1}{2}\Delta u \tag{B.5}$$

$frac$ ist die Korrektur für die aktuelle Zeit im Runge-Kutta-Schritt, $\frac{1}{2}$ kommt aus der Verfeinerung.

Für das schon beschrieben Verfahren 3. Ordnung ergibt sich im Programm nun folgender Ablauf

1. $u^{n+1} = u^n + \Delta t \cdot n$, $t = t^n + \Delta t$, $frac = 1$, d.h. wir gehen einen kompletten Zeitschritt
2. $u^{n+1} = \frac{3}{4}u^n + \frac{1}{4}u^{n+1} + \frac{1}{4}\Delta t \cdot n$, $t = t^n + \frac{1}{2}\Delta t$, $frac = -\frac{1}{2}$
3. $u^{n+1} = \frac{1}{3}u^n + \frac{2}{3}u^{n+1} + \frac{2}{3}\Delta t \cdot n$, $t = t^n + \Delta t$, $frac = 0.5$

Man benötigt also nur zwei Felder u^n für die alten Werte und u^{n+1} für die aktuellen Werte.

Runge-Kutta 4. Ordnung Das Runge-Kutta-Verfahren 4. Ordnung ist folgermaßen definiert

$$\begin{aligned}
\partial_t u^n &= f(u^n, t^n) && \text{(B.6)}\\
k_1 &= \Delta t f(u^n, t^n) && \text{(B.7)}\\
k_2 &= \Delta t f(u^n + \frac{1}{2}k_1, t^n + \frac{1}{2}\Delta t) && \text{(B.8)}\\
k_3 &= \Delta t f(u^n + \frac{1}{2}k_2, t^n + \frac{1}{2}\Delta t) && \text{(B.9)}\\
k_4 &= \Delta t f(u^n + \frac{1}{2}k_3, t^n + \Delta t) && \text{(B.10)}\\
u^{n+1} &= u^n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) && \text{(B.11)}
\end{aligned}$$

Dieses Verfahren benötigt nun im Gegensatz zur 3. Ordnung ein Hilfsfeld zur Datenspeicherung nebem dem Feld zur Speicherung der alten Werte. Die Implementation ist daher geringfügig verschieden von der vorher beschriebenen

1. $u^{n+1} = u^n + \frac{1}{2}\Delta t \cdot n$, $k = \frac{1}{3}\left(u^{n+1} - u^n\right)$, $frac = \frac{1}{2}$
2. $u^{n+1} = u^n + \frac{1}{2}\Delta t \cdot n$, $k+ = \frac{2}{3}u^{n+1}$, $frac = 0$
3. $u^{n+1} = u^n + \Delta t \cdot n$, $k+ = \frac{1}{3}u^{n+1}$, $frac = \frac{1}{2}$
4. $u^{n+1} = k + \frac{1}{6}\Delta t \cdot n$, $frac = 0$

Literaturverzeichnis

[BC89] Berger, M. J. ; Colella, P.: Local Adaptive Mesh Refinement for Shock Hydodynamics. In: *J. Comput. Phys.* 82 (1989), S. 64–84

[BCG89] Bell, J. B. ; Colella, P. ; Glaz, H. M.: A second-order projection method for the incompressible Navier–Stokes equation. In: *J. Comput. Phys.* 85 (1989), S. 257–283

[BCM01] Brown, D. L. ; Cortez, R. ; Minion, M. L.: Accurate Projection Methods for the Incompressible Navier-Stokes Equation. In: *J. of Computional Physics* 168 (2001), S. 464 – 469

[FGM97] Friedel, H. ; Grauer, R. ; Marliani, C.: Adaptive Mesh Refinement for Singular Current Sheets in Incompressible Magnetohydrodynamic Flows. In: *J. Comput. Phys.* 134 (1997), S. 190–198

[FVOMY00] Fadlun, E. A. ; Verzicco, R. ; Orlandi, P. ; Mohd-Yusof, J.: Combined Immersed-Boundary Finite-Difference Methods for Three-Dimensional Complex Flow Simulations. In: *J. of Computional Physics* 161 (2000), S. 35 – 60

[GMG98] Grauer, R. ; Marliani, C. ; Germaschewski, K.: Adaptive mesh refinement for singular solutions of the incompressible Euler equations. In: *Phys. Rev. Lett.* 80 (1998), S. 4177–4180

[JL00] Johnston, H.E. ; Liu, J.-G.: Local pressure boundary conditions. In: *Journal of Computational Physics* (2000)

[KL00] Kurganov, A. ; Levy, D.: A Third-Order Semidiscrete Central Scheme for Conservation Laws and Convection-Diffusion Equations. In: *SIAM J. Sci. Computing* 22 (2000), Nr. 4, S. 1461 – 1488

[LL78] Landau, L. D. ; Lifshitz, E. M.: *Lehrbuch der Theoretischen Physik*. Bd. 6: *Hydrodynamik*. Akademie-Verlag, 1978

[Pao67] Pao, R.H.: *Fluid Mechanic*. Wiley and Sons, 1967

[Pra52] Prandtl, L.: *Führer durch die Strömungslehre*. Braunschweig : Vieweg-Verlag, 1952

[Whi68] Whitaker, S.: *Introduction to Fluid Mechanics*. Prentice Hall, 1968

Danksagung

Während meiner Diplomarbeit haben mir viele Menschen mit Rat und Tat zur Seite gestanden, daher möchte ich ihnen auch danken. Namentlich möchte ich diejenigen erwähnen, ohne die diese Arbeit nicht das wäre, was sie jetzt ist.

- Herrn Prof. Dr. Rainer Grauer danke ich für seine Hilfe und Ratschläge
- Herrn Dr. Holger Schmitz danke ich für die interessanten Diskussionen und kritischen Fragen zu dieser Arbeit
- Herrn cand. phys. David Riemenschneider kann ich gar nicht oft genug dafür danken, daß er es mit mir 12 Monate ausgehalten hat
- Herrn Dr. Guido Piaszenski danke ich für die Akribie, mit der er auch kleinste sprachliche Schwächen dieser Arbeit aufgespürt hat
- Frau Gisela Buhr und Frau Heidi Brüggenthies danke ich für das gute Arbeitsklima

Meiner Mutter danke ich für die Unterstützung meines Studiums. Widmen möchte ich diese Arbeit meinem verstorbenen Vater, der auf meinem Lebensweg eine große Hilfe war und meine Entscheidungen immer mitgetragen hat.

Bochum, Oktober 2002

Felix Spanier