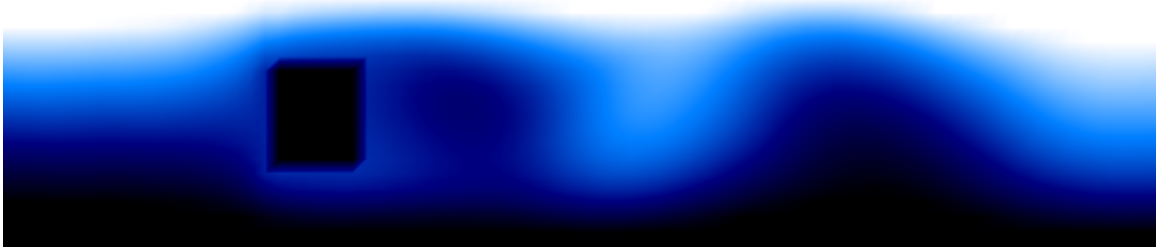
F-Praktikum - Computational-Physics-Versuch

NSG - Numerische Behandlung der Navier-Stokes-Gleichungen



Basierend auf [10]. Ausgearbeitet von Andreas Weyhausen.

Betreuer: Hannes Rüter
hannes.rueter@uni-jena.de
Theoretisch-physikalisches Institut

---

1 Aufgabenstellung

Schreiben Sie ein Programm, welches die Navier-Stokes-Gleichungen für eine inkompressible Flüssigkeit in zwei Dimensionen löst. Beschäftigen Sie sich hierfür zunächst mit der Theorie der Navier-Stokes-Gleichung und der Finiten-Differenzen-Methode. Implementieren Sie anschließend Ihr Programm unter Berücksichtigung dieser Anleitung. Simulieren und visualisieren Sie mit Ihrem Programm eine Nischenströmung und vergleichen Sie ihre Resultate mit aus der Literatur bekannten Werten.

1.1 Themen

- Computational-Fluid-Dynamics
- inkompressible Navier-Stokes-Gleichungen
- Finite-Differenzen
- Donor-Cell-Schema
- Verschobene Gitter
- SOR Verfahren
- Simulation einer Nischenströmung
- Visualisierung

2 Einleitung

Die Computational Fluid Dynamik (CFD) ist eine wichtige Methode der Strömungsmechanik. Ziel dieser Methode ist das numerische Lösen strömungsmechanischer Probleme. Viele dieser Probleme sind nicht analytisch sondern nur numerisch lösbar. Die Simulation strömungmechanischer Vorgänge ersetzt die oft aufwendigen und kostspieligen Experminete beispielsweise im Windkanal. Innerhalb der Computational Physics kommt der CFD eine wichtige Bedeutung zu. Die Entwicklung der verwendeten Techniken erfolgte nicht zuletzt aufgrund ihrer Notwendigkeit zum Lösen strömungsmechanischer Probleme. Diese Probleme resultieren im Allgemeinen in nichtlinearen, partiellen Differentialgleichungen. Um diese zu lösen, werden spezielle numerische Techniken benötigt. Außerdem sind die Simulationen häufig sehr rechenaufwendig, was den Einsatz moderner Rechentechnik, beispielsweise Computercluster, und moderner Methoden, beispielsweise paralleles Rechnen, erfordert.

In diesem Versuch werden wir die laminare Nischenströmung einer inkompressiblen Flüssigkeit simulieren. Dazu werden wir die Navier-Stokes-Gleichungen in zwei Dimensionen numerisch mit Hilfe der Finiten-Differenzen-Technik lösen.

3 Navier-Stokes-Gleichungen

Die Navier-Stokes-Gleichungen für inkompressible Flüssigkeiten bilden ein System partieller Differentialgleichungen welche die Zeitevolution des Geschwindigkeitfeldes $\vec{u}$ und des Drucks p des Fluids beschreiben. Die Dichte ist für inkompressible Flüssigkeiten konstant. Die Navier-Stokes-Gleichungen für den hier betrachteten Fall lauten

$$\frac{\partial \vec{u}}{\partial t} + (\vec{u} \cdot \nabla)\vec{u} + \nabla p = \frac{1}{Re}\Delta \vec{u} + \vec{g} \qquad \text{(Impulsgleichung)}, \tag{1}$$

$$\nabla \cdot \vec{u} = 0 \qquad \text{(Kontinuitätsgleichung)}, \tag{2}$$

wobei Re die Reynoldszahl und $\vec{g}$ Gravitationskräfte bezeichnet. Im zweidimensionalen Fall ergeben sich aus Gleichung (1) die folgenden Zeitevolutionsgleichungen für die Geschwindigkeitskomponenten $\vec{u} = (u, v)$

$$\frac{\partial u}{\partial t} + \frac{\partial p}{\partial x} = \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - \frac{\partial (u^2)}{\partial x} - \frac{\partial (uv)}{\partial y} + g_x, \tag{3}$$

$$\frac{\partial v}{\partial t} + \frac{\partial p}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) - \frac{\partial (uv)}{\partial x} - \frac{\partial (v^2)}{\partial y} + g_y. \tag{4}$$

Aus Gleichung (2) erhalten wir eine Laplace-Gleichung als Zwangsbedingung

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{5}$$

Da wir in der numerischen Simulation nur ein endliches Gebiet betrachten können, benötigen wir Randbedingungen. Geben wir zusätzlich Anfangswerte vor, welche die Zwangsbedingung erfüllen, so erhalten wir ein Anfangs-Randwert-Problem.

Um die Randbedingungen zu formulieren, zerlegen wir die Geschwindigkeitsvektoren an den Rändern in ihre tangentialen und normalen Komponten zum Rand. Wir bezeichnen diese mit φ_t und φ_n. Die Randbedingungen definieren wir dann mit Hilfe dieser Komponenten und deren Normalenableitungen. In diesem Versuch betrachten wir nur den einfachen Fall von senkrechten und waagerechten Randstücken. Somit ergibt sich

- für senkrechte Randstücke:

$$\varphi_n = u, \qquad \varphi_t = v, \qquad \frac{\partial \varphi_n}{\partial n} = \frac{\partial u}{\partial x}, \qquad \frac{\partial \varphi_t}{\partial n} = \frac{\partial v}{\partial x}. \tag{6}$$

- für waagrechte Randstücke:

$$\varphi_n = v, \qquad \varphi_t = u, \qquad \frac{\partial \varphi_n}{\partial n} = \frac{\partial v}{\partial y}, \qquad \frac{\partial \varphi_t}{\partial n} = \frac{\partial u}{\partial y}. \tag{7}$$

Nun können wir folgende Bedingungen formulieren:

- **Haftbedingung:** Die Flüssigkeit verlässt das Gebiet nicht, es gibt eine Wechselwirkung zwischen Flüssigkeit und Rand

$$\varphi_n(x, y) = 0, \qquad \varphi_t(x, y) = 0. \tag{8}$$

- **Ausströmbedingung:** Die Flüssigkeit verlässt das Gebiet.

$$\frac{\partial \varphi_n(x,y)}{\partial n} = 0, \qquad \frac{\partial \varphi_t(x,y)}{\partial n} = 0 \tag{9}$$

- **Einströmbedingung:** Die Flüssigkeit tritt mit der Geschwindigkeit $\vec{u} = (\varphi_n^0, \varphi_t^0)$ in das Gebiet ein.

$$\varphi_n(x,y) = \varphi_n^0, \qquad \varphi_t(x,y) = \varphi_t^0 \tag{10}$$

4 Numerik

Für die numerische Simulation der Navier-Stokes-Gleichungen diskretisieren wir die Evolutionsvariablen auf Gittern und approximieren die räumlichen Ableitungen durch Finite-Differenzen.

4.1 Diskretisierung der Navier-Stokes-Gleichungen

Als Domäne für unsere Simulation wählen wir ein rechteckiges Gebiet

$$\Omega := [0,a] \times [0,b] \subset \mathbb{R}^2 \tag{11}$$

und zerlegen es in x-Richtung in $i_{\max}$ gleiche Teile und in y-Richtung in $j_{\max}$ gleiche Teile. Wir füllen also das Gebiet Ω mit $i_{\max} \cdot j_{\max}$ Zellen der Fläche $\delta x \cdot \delta y$ aus, wobei

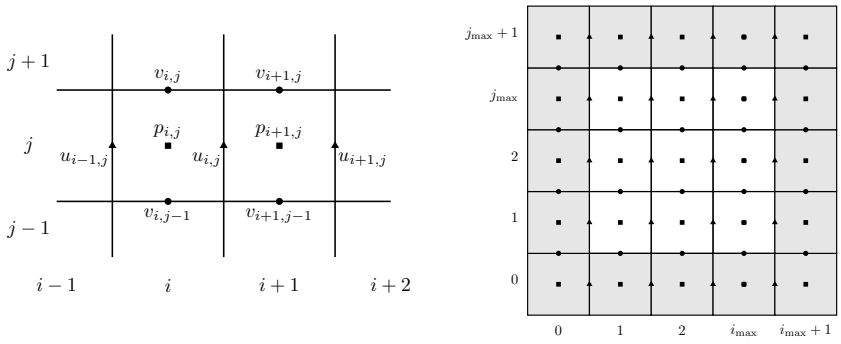
$$\delta x := \frac{a}{i_{\max}} \qquad \text{und} \qquad \delta y := \frac{b}{j_{\max}}. \tag{12}$$

Unsere Variablen werden wir an folgenden Stellen des so entstandenen Gitters speichern:

- Der Druck p_{ij} wird im Mittelpunkt der Zelle $((i-0.5)\delta x, (j-0.5)\delta y)$ gespeichert.
- Die Geschwindigkeitskomponente in x-Richtung u_{ij} werden in den Mittelpunkten der senkrechten Zellkante $(i\delta x, (j-0.5)\delta y)$ gespeichert.
- Die Geschwindigkeitskomponente in y-Richtung v_{ij} werden in den Mittelpunkten der waagrechten Zellkante bei $((i-0.5)\delta x, j\delta y)$ gespeichert.

Die Verwendung dieser drei verschobenen Gitter verhindert mögliche Oszillationen und Uneindeutigkeiten des Drucks. Für eine Diskussion dieses Stabilität-Problems lesen Sie bitte Kapitel 3.1.1 in [10].

Die obigen Gitterdefinitionen liefern nicht für alle Variablen vollständige Randwerte; auf den senkrechten Rändern gibt es keine Werte für v, auf den waagrechten keine Werte für u. Um die Randwerte zu vervollständigen, führen wir eine Schicht von Randzellen ein, mit deren Hilfe wir die Randbedingungen (8) - (9) wie folgt implementieren:



- **Haftbedingungen:** Die Haftbedingung besagt, dass die Geschwindigkeitskomponenten am Rand gleich Null sein sollen. Die Werte, die direkt auf dem Rand liegen, setzen wir auf

$$u_{0,j} = 0, \quad u_{i_{\max},j} = 0 \qquad j = 1, \ldots, j_{\max}, \tag{13}$$
$$v_{i,0} = 0, \quad v_{i,j_{\max}} = 0 \qquad i = 1, \ldots, i_{\max}. \tag{14}$$

  An den Rändern auf welche keine Werte liegen, erhalten wir einen Randwert durch Mittlung über die beiden benachbarten Werte. Den so ermittelten Mittelwert auf dem Rand setzen wir gleich Null und erhalten die Bedingungen

$$v_{0,j} = -v_{1,j}, \quad v_{i_{\max+1},j} = -v_{i_{\max},j} \qquad j = 1, \ldots, j_{\max}, \tag{15}$$
$$u_{i,0} = -u_{i,1}, \quad u_{i,j_{\max+1}} = -u_{i,j_{\max}} \qquad i = 1, \ldots, i_{\max}. \tag{16}$$

- **Ausströmbedingung:** Die Ausströmbedinung besagt, dass sich die Geschwindigkeit senkrecht zum Rand nicht ändert. Wir erreichen dies, indem wir Geschwindigkeiten am Rand gleich der benachbarten Geschwindigkeiten setzen.

$$u_{0,j} = u_{1,j}, \quad u_{i_{\max},j} = u_{i_{\max}-1,j} \tag{17}$$
$$v_{0,j} = v_{1,j}, \quad v_{i_{\max}+1,j} = v_{i_{\max},j} \qquad j = 1, \ldots j_{\max} \tag{18}$$
$$u_{i,0} = u_{i,1}, \quad u_{i,j_{\max}+1} = u_{i,j_{\max}} \tag{19}$$
$$v_{i,0} = v_{i,1}, \quad v_{i,j_{\max}} = v_{i,j_{\max}-1} \qquad i = 1, \ldots i_{\max} \tag{20}$$

- **Einströmbedingung:** Für die Einströmbedingung setzen wir die Werte für die Geschwindigkeit auf einen vorgegebenen Wert. Sollten keine Werte auf dem Rand vorhanden sein, setzen wir den Wert durch geeignete Mittelung.

4.2 Lösen der Impulsgleichungen

Wir betrachten das System zum diskreten Zeitpunkt n und evolvieren es durch einen Euler-Schritt der Impulsgleichungen (3) und (4) und zum neuen Zeitpunkt $n+1$.

$$u^{(n+1)} = F^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial x}, \tag{21}$$

$$v^{(n+1)} = G^{(n)} - \delta t \frac{\partial p^{(n+1)}}{\partial y}, \tag{22}$$

wobei F und G wie folgt definiert sind

$$F = u + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial u^2}{\partial x} - \frac{\partial uv}{\partial y} + g_x \right], \tag{23}$$

$$G = v + \delta t \left[\frac{1}{Re} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial uv}{\partial x} - \frac{\partial v^2}{\partial y} + g_y \right]. \tag{24}$$

Offensichtlich werden die Geschwindigkeitskomponenten u und v zum neuen Zeitpunkt $n+1$ aus den Geschwindigkeitskomponenten zum Zeitpunkt n, sowie der Druckkomponente zum Zeitpunkt $n+1$, berechnet. Das Verfahren ist also explizit in der Geschwindigkeit und implizit im Druck. Wir müssen also zuerst den neuen Druck berechnen, bevor wir die neuen Geschwindigkeitskomponenten ermitteln können. Eine Bestimmungsgleichung für den Druck werden wir im nächsten Abschnitt herleiten.

Die Größe des Zeitschritts δt kann nicht beliebig gewählt werden. Laut [13] sollte er folgende drei Bedingungen erfüllen um eine stabile Simulation zu gewährleisten

$$\frac{2\delta t}{Re} < \frac{1}{(\delta x^{-2} + \delta y^{-2})}, \qquad |u_{\max}|\delta t < \delta x, \qquad |v_{\max}|\delta t < \delta y, \qquad . \tag{25}$$

Wir verwenden diese Bedingungen um eine adaptive Schrittweitensteuerung zu implementieren.

$$\delta t := \tau \min \left(\frac{Re}{2} \frac{1}{(\delta x^{-2} + \delta y^{-2})}, \frac{\delta x}{|u_{\max}|}, \frac{\delta y}{|v_{\max}|} \right). \tag{26}$$

Hierbei ist τ ein Sicherheitsfaktor für welchen gilt $\tau \in]0, 1]$.

Nach der Diskretisierung in der Zeit, werden wir nun den räumlichen Anteil der Impulsgleichungen diskretisieren. Die Impulsgleichung für u wird in den Mittelpunkten der senkrechten Zellkante, die Impulsgleichung für v in den Mittelpunkten der waagrechten

Kanten ausgewertet.

$$u_{i,j}^{(n+1)} = F_{i,j}^{(n)} - \delta t \left[\frac{\partial p^{(n+1)}}{\partial x}\right]_{i,j} \qquad \text{für } i = 1,\ldots,i_{\max}-1, \quad j = 1,\ldots,j_{\max}, \tag{27}$$

$$v_{i,j}^{(n+1)} = G_{i,j}^{(n)} - \delta t \left[\frac{\partial p^{(n+1)}}{\partial y}\right]_{i,j} \qquad \text{für } i = 1,\ldots,i_{\max}, \quad j = 1,\ldots,j_{\max}-1, \tag{28}$$

$$F_{i,j} = u_{i,j} + \delta t \left[\frac{1}{Re}\left(\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j} + \left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j}\right) - \left[\frac{\partial u^2}{\partial x}\right]_{i,j} - \left[\frac{\partial (uv)}{\partial y}\right]_{i,j} + g_x\right], \tag{29}$$

$$G_{i,j} = v_{i,j} + \delta t \left[\frac{1}{Re}\left(\left[\frac{\partial^2 v}{\partial x^2}\right]_{i,j} + \left[\frac{\partial^2 v}{\partial y^2}\right]_{i,j}\right) - \left[\frac{\partial (uv)}{\partial x}\right]_{i,j} - \left[\frac{\partial v^2}{\partial y}\right]_{i,j} + g_y\right]. \tag{30}$$

Im Folgenden werden wir alle vorkommenden räumlichen Ableitungen durch Finite Differenzen auf den Gittern approximieren. Dazu verwenden wir für die Ableitungen der linearen Terme zentrierte Differenzen. Für die Ableitung der nichtlinearen Terme bilden wir eine Linearkombination aus zentrierten Differenzen und sogenannten Donor-Cell-Stencils. Dies garantiert die numerische Stabilität. Weiterführende Information hierzu finden Sie in [10].

Die erste Ableitung des Drucks approximieren wir durch

$$\left[\frac{\partial p}{\partial x}\right]_{i,j} = \frac{p_{i+1,j} - p_{i,j}}{\delta x}, \qquad \left[\frac{\partial p}{\partial y}\right]_{i,j} = \frac{p_{i,j+1} - p_{i,j}}{\delta y}. \tag{31}$$

Für die zweiten Ableitungen der lineare Terme verwenden wir standardmäßig zentrierte Differenzen.

$$\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\delta x)^2}, \qquad \left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\delta y)^2}. \tag{32}$$

$$\left[\frac{\partial^2 v}{\partial x^2}\right]_{i,j} = \frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{(\delta x)^2}, \qquad \left[\frac{\partial^2 v}{\partial y^2}\right]_{i,j} = \frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{(\delta y)^2}. \tag{33}$$

Für die nichtlineare Ableitungen erhalten wir:

$$\left[\frac{\partial(u^2)}{\partial x}\right]_{i,j} = \frac{1}{\delta x}\left(\left(\frac{u_{i,j}+u_{i+1,j}}{2}\right)^2 - \left(\frac{u_{i-1,j}+u_{i,j}}{2}\right)^2\right) + \gamma\frac{1}{\delta x}\left(\frac{|u_{i,j}+u_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i,j}|}{2}\frac{(u_{i-1,j}-u_{i,j})}{2}\right), \quad (34)$$

$$\left[\frac{\partial(uv)}{\partial y}\right]_{i,j} = \frac{1}{\delta y}\left(\frac{(v_{i,j}+v_{i+1,j})}{2}\frac{(u_{i,j}+u_{i,j+1})}{2} - \frac{(v_{i,j-1}+v_{i+1,j-1})}{2}\frac{(u_{i,j-1}+u_{i,j})}{2}\right) + \gamma\frac{1}{\delta y}\left(\frac{|v_{i,j}+v_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i,j+1})}{2} - \frac{|v_{i,j-1}+v_{i+1,j-1}|}{2}\frac{(u_{i,j-1}-u_{i,j})}{2}\right). \quad (35)$$

$$\left[\frac{\partial(v^2)}{\partial y}\right]_{i,j} = \frac{1}{\delta y}\left(\left(\frac{v_{i,j}+v_{i,j+1}}{2}\right)^2 - \left(\frac{v_{i,j-1}+v_{i,j}}{2}\right)^2\right) + \gamma\frac{1}{\delta y}\left(\frac{|v_{i,j}+v_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i,j+1})}{2} - \frac{|v_{i,j-1}+v_{i,j}|}{2}\frac{(v_{i,j-1}-v_{i,j})}{2}\right), \quad (36)$$

$$\left[\frac{\partial(uv)}{\partial x}\right]_{i,j} = \frac{1}{\delta x}\left(\frac{(u_{i,j}+u_{i,j+1})}{2}\frac{(v_{i,j}+v_{i+1,j})}{2} - \frac{(u_{i-1,j}+u_{i-1,j+1})}{2}\frac{(v_{i-1,j}+v_{i,j})}{2}\right) + \gamma\frac{1}{\delta x}\left(\frac{|u_{i,j}+u_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i+1,j})}{2} - \frac{|u_{i-1,j}+u_{i-1,j+1}|}{2}\frac{(v_{i-1,j}-v_{i,j})}{2}\right). \quad (37)$$

Der Faktor γ bestimmt, wie die beiden Anteile gewichtet werden. Nach Hirt [11] sollte er so gewählt werden, dass die Bedingung

$$\gamma \geq \max_{ij}\left(\frac{|u_{ij}|\,\delta t}{\delta x}, \frac{|v_{ij}|\,\delta t}{\delta y}\right) \quad (38)$$

erfüllt ist.

4.3 Bestimmungsgleichung für den Druck

Wir leiten eine Bestimmungsgleichung für den Druck aus der Kontinuitätsgleichung (5) her. Zum Zeitschritt $n+1$ gilt

$$0 = \frac{\partial u^{(n+1)}}{\partial x} + \frac{\partial v^{(n+1)}}{\partial y} \quad (39)$$

Benutzen wir die Gleichungen (21) und (22), so erhalten wir folgende Poisson-Gleichung für den Druck

$$\frac{\partial^2 p^{(n+1)}}{\partial x^2} + \frac{\partial^2 p^{(n+1)}}{\partial y^2} = \frac{1}{\delta t}\left(\frac{\partial F^{(n)}}{\partial x} + \frac{\partial G^{(n)}}{\partial y}\right). \quad (40)$$

Um diese Gleichung zu lösen, diskretisieren wir sie in den Zellmittelpunkten

$$\frac{p_{i+1,j}^{(n+1)} - 2p_{i,j}^{(n+1)} + p_{i-1,j}^{(n+1)}}{(\delta x)^2} + \frac{p_{i,j+1}^{(n+1)} - 2p_{i,j}^{(n+1)} + p_{i,j-1}^{(n+1)}}{(\delta y)^2} = \frac{1}{\delta t}\left(\frac{F_{i,j}^{(n)} - F_{i-1,j}^{(n)}}{\delta x} + \frac{G_{i,j}^{(n)} - G_{i,j-1}^{(n)}}{\delta y}\right). \tag{41}$$

Wir lösen diese diskrete Poisson Gleichung mit einem Relaxationsverfahren, dem sogenannten Successive-Over-Relaxation-Verfahren (SOR). Mehr zu diesem und anderen Relaxationsverfahren finden sie sich in der Literatur, zum Beispiel in[12]. Nun berechnen wir im n-ten Zeitschritt iterativ den neuen Druckwert $p^{(n+1)}$. Als Startwert wählen wir $p_{ij}^{\mathrm{it}=0} := p_{ij}^{(n)}$ und iterieren dann

$$p_{i,j}^{\mathrm{it}+1} := (1-\omega)p_{i,j}^{\mathrm{it}} + \frac{\omega}{2\left((\delta x)^{-2} + (\delta y)^{-2}\right)}\left(\frac{p_{i+1,j}^{\mathrm{it}} + p_{i-1,j}^{\mathrm{it}+1}}{(\delta x)^2} + \frac{p_{i,j+1}^{\mathrm{it}} + p_{i-1,j-1}^{\mathrm{it}+1}}{(\delta y)^2} - \mathrm{RHS}_{i,j}\right), \tag{42}$$

wobei $\mathrm{RHS}_{i,j}$ die rechte Seite von Gleichung (40) bezeichnet und ω den Relaxationsfaktor.

Bei jedem Iterationsschritt wird das Residuum

$$r_{i,j}^{\mathrm{it}+1} := \frac{p_{i+1,j}^{\mathrm{it}+1} - 2p_{i,j}^{\mathrm{it}+1} + p_{i-1,j}^{\mathrm{it}+1}}{(\delta x)^2} + \frac{p_{i,j+1}^{\mathrm{it}+1} - 2p_{i,j}^{\mathrm{it}+1} + p_{i,j-1}^{\mathrm{it}+1}}{(\delta y)^2} - \mathrm{RHS}_{i,j} \tag{43}$$

berechnet. Die Iteration wird abgebrochen, wenn die Norm des Residuums eine vorgegebenen relative Toleranzgrenze unterschreitet

$$\|r^{\mathrm{it}+1}\| < \varepsilon \|p^{\mathrm{it}=0}\|. \tag{44}$$

Wir setzen dann $p^{(n+1)} = p^{\mathrm{it}+1}$. Als Norm kann zum Beispiel die diskrete L_2-Norm

$$\|r^{\mathrm{it}}\|_2 := \left(\frac{1}{i_{\max} j_{\max}} \sum_{i=1}^{i_{\max}} \sum_{j=1}^{j_{\max}} (r_{i,j}^{\mathrm{it}})^2\right)^{1/2} \tag{45}$$

oder die Maximumnorm L_∞

$$\|r^{\mathrm{it}}\|_\infty := \max\{|r_{i,j}^{\mathrm{it}}|, i = 1 \ldots i_{\max}, j = 1, \ldots, j_{\max}\} \tag{46}$$

verwendet werden.

Für jeden Iterationsschritt müssen wir auch die Randzellen füllen. Hierzu übertragen wir einfach die alten Druck-Werte der Nachbarzellen in die Geisterzellen.

$$p_{0,j}^{it+1} = p_{1,j}^{it}, \qquad p_{i_{\max}+1,j}^{it+1} = p_{i_{\max},j}^{it}, \tag{47}$$

$$p_{i,0}^{it+1} = p_{i,1}^{it}, \qquad p_{i,j_{\max}+1}^{it+1} = p_{i,j_{\max}}^{it}, \qquad . \tag{48}$$

5 Algorithmus zur Lösung der Navier-Stokes-Gleichungen

Im folgen sind nun nochmals alle Schritte zur Lösen der Navier- Stokes-Gleichungen in richtiger Reihenfolge beschrieben:

- Parameter einlesen und alle Variablen initialisieren
- Speicher für Felder allokieren und Felder initialisieren
- Beginn der Zeitschleife
  - Zeitschritt δt gemäß der Stabilitiätsbedinung (Gl. 26) wählen
  - Geisterzellen gemäss der Randbedingungen (Gl. 8-10) füllen
  - $F^{(n)}$ und $G^{(n)}$ gemäss Gleichung (29) und (30) berechnen
  - Die rechte Seite der Druckgleichung (40) berechnen
  - Beginn SOR Iterationsschleife
    - Druck für alle Zellen gemäss Gleichung (42) berechnen
    - Residuum (Gl. 43) und dessen Norm berechnen
    - Wenn Norm des Residuums klein genug, Schleife abbrechen
  - Neue Geschwindigkeitskomponenten $u^{(n+1)}$ und $v^{(n+1)}$ (Gl. 21 und 22) berechnen
  - Ausgabe der Felder

6 Implementierung

Implementieren Sie den Algorithmus in folgenden Schritten:

1. Schreiben Sie zuerst das Grundgerüst des Programms, welches alle benötigten Variablen und Felder anlegt und initialisieren Sie die Felder mit Null.

2. Implementieren Sie im zweiten Schritt eine Ausgaberoutine für die Felder. Initialisieren Sie u und v mit einer bekannten Funktion (z.B. $u = \cos(x)\cos(y)$) und testen Sie die Ausgaberoutine, indem Sie sich die Felder ausgeben und plotten lassen.

3. Programmieren Sie nun die Funktion, welche F und G berechnet. Hierfür müssen die Ableitungs-Stencils (32)-(37), sowie die Gleichungen (29)-(30) implementiert werden. Testen Sie die Ableitungs-Stencils indem Sie sich damit die Ableitung einer bekannten Funktion ausrechnen lassen.

4. Realisieren Sie die Druckberechnung mit Hilfe des SOR-Verfahrens. Lösen Sie als Test die elliptische Gleichung $\Delta p(x, y) = -2p(x, y)$. Geben Sie die Lösung aus und vergleichen Sie Ihre numerische Lösung mit der analytischen Lösung. Testen Sie das Verfahren, seine Konvergenz und Konvergenzgeschwindigkeit für unterschiedlich Gitterauflösungen und Relaxationsparameter ω.

5. Implementieren Sie nach dem Testproblem für die Druckiteration eine Funktion, welche die rechte Seite von Gleichung (41) berechnet.

6. Programmieren Sie zuletzt die Zeitschleife. Hierfür wird eine Funktion benötigt, welche den Zeitschritt laut der Stabilitätsbedingung setzt. Beachten Sie hierbei, dass Sie eine mögliche Division durch Null abfangen. Weiterhin muss die Funktion implementiert werden, die aus den berechneten Werten von F und G, sowie den neuen Druckkomponenten die neuen Geschwindigkeitskomponenten berechnet.

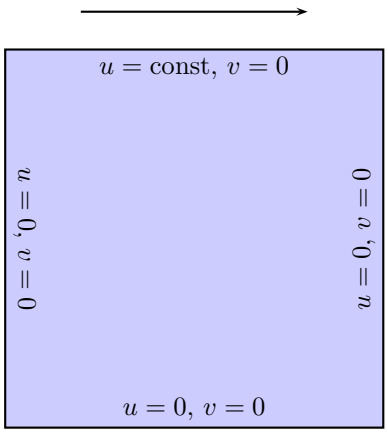
7 Zusatzhinweise und -aufgaben

- Überlegen Sie sich im ersten Schritt die Struktur Ihres Programmes. Es ist sinnvoll, den Quellcode in logische Einheiten zu unterteilen. Eine mögliche Einteilung ist:
  - Ein- und Ausgabe
  - Initialisierung
  - Speicherverwaltung
  - Hauptprogramm
  - Zeitschritt
  - Druckberechnung
  - Randbedingungen
- Überlegen Sie sich, welche Parameter Sie für Ihr Programm benötigen werden. Diese sollten nicht im Quellcode gesetzt werden, sondern von einer Parameterdatei eingelesen werden. Dies ermöglicht ein Ändern der Parameter, ohne neu kompilieren zu müssen. Hierfür können Sie die Routine zum Einlesen der Parameter selbst programmieren oder auf eine Bibliothek zurückgreifen [3].
- Überlegen Sie sich, welche Funktionen Sie benötigen, um den Algorithmus in Ihr Programm zu implementieren. Welche Übergabe- und Rückgabeparameter haben diese Funktionen? Eine mögliche Liste von Funktionen finden Sie in [10].
- Wenn Ihr Programm aus mehreren Teilen besteht, empfiehlt es sich, Software zu benutzen, welche das Kompilieren automatisiert. Mögliche Programme sind hierfür `gnu make` [2, 14, 5] oder `eclipse`.
- Kommentieren Sie Ihren Code! Für eine übersichtliche Dokumentation können Sie `Doxygen` [1, 4] verwenden.
- Machen Sie sich Gedanken über die Ausgabe ihres Programmes. Für die Visualisierung des Geschwindigkeitsfeldes wird ein Programm benötigt, welches zweidimensionale Vektordaten darstellen kann. Mögliche Programme sind `Matlab` oder `Visit` [8, 6, 7].

8 Simulation einer Nischenströmung

8.1 Problemstellung

Simulieren Sie mit Ihrem Programm eine Nischenströmung (Lid-Driven-Cavity). Dieses Modell ist eines der am einfachsten zu simulierenden Systeme und stellt einen Standardtest für Fluid-Dynamik-Codes dar. Hierzu betrachten Sie eine Flüssigkeit in einem quadratischen Gebiet. Alle Ränder, bis auf den oberen Rand, werden mittels der Haftbedingung gesetzt. Am oberen Rand wird die Geschwindigkeitkomponente in x-Richtung auf einen konstanten Wert gesetzt, in y-Richtung gilt die Haftbedingung.



Als Modell dient eine mit Wasser gefüllte quadratische Box, über die ein Band mit konstanter Geschwindigkeit gezogen wird.

8.2 Simulation

Starten Sie Ihre Simulation mit einer ruhenden Flüssigkeit und verschwindendem Druck

$$u(t=0)_{i,j} = v(t=0)_{i,j} = p(t=0)_{i,j} = 0. \tag{49}$$

Setzen Sie die x Geschwindigkeitskomponente am oberen Rand auf einen konstanten Wert und simulieren Sie zuerst die inkompressbile Flüssigkeit in einer quadratischen Box und bearbeiten Sie dabei folgende Punkte:

- Analysieren Sie die Zeitevolution der Flüssigkeit. Achten Sie hierbei auf die Ausbildung von Wirbeln und auf den sich einstellenden stationären Endzustand.
- Studieren Sie das Verhalten unterschiedlich zäher Flüssigkeiten, indem Sie die Reynoldszahl variieren ($Re \in (10, 10000)$). Wie wirkt sich dies auf die Wirbelbildung und den stationären Endzustand aus?

- Zur Validierung Ihres Codes vergleichen Sie Ihre Ergebnisse mit denen aus Ghia et al. [9]. In dieser Arbeit werden die Geschwindigkeitskompontenten entlang der Linie $x = 0$ für verschiedene Reynoldszahlen gegeben. Untersuchen Sie, wie abhängig von der räumlichen Auflösung $\delta x, \delta y$, ihr Ergebnis gegen das von Ghia et al. gegebenen konvergiert.
- Nachdem Sie sich von der Richtigkeit überzeugt haben, experimentieren Sie mit unterschiedlichen Boxgeometrien. Wie verhält sich die Flüssigkeit für rechteckige Geometrien?
- Gehen Sie am oberen Rand von der konstanten Geschwindigkeitskompontente in eine sich zeitlich periodisch änderende über.

9 Fragen

- Was besagt die Reynoldszahl?
- Welche zusätzlichen Terme gibt es bei den Navier-Stokes-Gleichungen für kompressible Flüssigkeiten?
- Was beschreiben die Euler-Gleichungen?
- Warum kann das Programm keine Flüssigkeiten mit beliebig großer Reynoldszahl simulieren? Welches Phänomen tritt in diesem Fall auf?
- Warum werden in diesem Versuch verschobene Gitter verwendet?
- Welche anderen Methoden gibt es, um stabile FD Simulationen zu erhalten?
- Was sind Donor-Cell-Stencils?
- Welche Ordnung haben die im Versuch verwendeten Finite-Differenzen-Stencils? Was besagt die Ordnung?
- Wodurch unterscheiden sich explizite von impliziten Zeitschritt-Verfahren. Nennen Sie Beispiele für solche Verfahren?
- Was ist der Unterschied zwischen Gauß-Seidel, SOR und Jacobi-Verfahren?
- Was sind elliptische, parabolische und hyperbolische Differentialgleichungen?
- Was besagt das Courant-Limit und warum ist es wichtig für eine stabile Zeitevolution?

Literatur

[1] Doxygen - generate documentation from source code. http://www.stack.nl/~dimitri/doxygen/.

[2] Gnu make. http://www.gnu.org/software/make.

[3] Libconfig - c/c++ configuration file library. http://www.hyperrealm.com/libconfig.

[4] Selflinux - doxygen howto. http://selflinux.org/selflinux/html/doxygen01.html.

[5] A simple makefile tutorial. http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/.

[6] Visit - visualization tool. https://wci.llnl.gov/codes/visits/.

[7] Visit wiki. https://visualization.hpc.mil/wiki/VisIt.

[8] Visualization toolkit file formats. http://www.vtk.org/VTK/img/file-formats.pdf.

[9] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. Journal of Computational Physics, 48:387–411, December 1982.

[10] Michael Griebel, Thomas Dornseifer, and Tilman Neunhoeffer. Numerische Simulationen in der Stroemungsmechanik. Scientific Computing. Vieweg Verlag, 1995.

[11] C. W. Hirt, B. D. Nichols, and N. C. Romero. SOLA: A numerical solution algorithm for transient fluid flows. NASA STI/Recon Technical Report N, 753:32418–+, January 1975.

[12] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. Numerical Recipes 3rd Edition: The Art of Scientific Computing. Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[13] Murilo F. Tome and Sean McKee. Gensmac: A computational marker and cell method for free surface flows in general domains. Journal of Computational Physics, 110(1):171 – 186, 1994.

[14] Ben Yoshino. Make - a tutorial. http://www.eng.hawaii.edu/Tutor/Make.