Programmiermethoden des Wissenschaftlichen Rechnens

Winter semester 2018/2019
Prof. Dr. Marc Alexander Schweitzer
Clelia Albrecht and Albert Ziegenhagel

UNIVERSITÄT BONN

Exercise sheet 5.

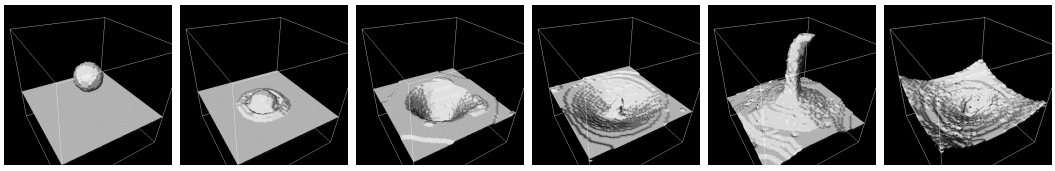
1 Einführung

In dem noch jungen Forschungsgebiet des wissenschaftlichen Rechnens werden der technische Sachverstand der Ingenieure, die numerischen Verfahren der Mathematiker und die modernen Methoden und Rechner der Informatiker gleichgewichtig interdisziplinär eingesetzt. Die mathematische Vorausberechnung technischer Prozesse, kurz numerische Simulation genannt, ist dabei ein wichtiges Ziel.

Durch die numerische Simulation kann beispielweise der Aufbau kostspieliger Versuchsaufbauten vermieden werden. Darüberhinaus lassen sich in vielen Fällen Aussagen treffen, bei denen dies auf Grund technischer Unzulänglichkeiten sonst nicht möglich wäre, oder ein praktisches Experiment sich von vornherein verbietet.

Ein wichtiges Beispiel für den Einsatz der numerischen Simulation ist die Berechnung von Strömungsvorgängen wie die Umströmung eines Flugzeugflügels, der Luftwiderstand eines Automobils, Verbrennungsvorgänge in Kraftwerken, sowie Schmelzprozesse und Kristallwachstum bei der Halbleiterfertigung. Aber auch bei der Wettervorhersage, in der Geologie, der Bioinformatik, der Mechanik (Elastizitätstheorie) und in vielen anderen Gebieten der Physik und der Chemie kommen Simulationsmethoden zum Einsatz.

Das folgende Vorgehen ist dabei charakteristisch: Aus der Beobachtung der Realität wird ein mathematisches Modell entwickelt, das nach geeigneter Diskretisierung näherungsweise gelöst wird. Mittels geeigneter Visualisierungsmethoden können die Ergebnisse interpretiert werden. Gegebenenfalls müssen dann die numerischen Lösungsverfahren oder das Modell nachgebessert werden (vergleiche hierzu auch Abbildung 1).



2 Die Problemstellung: Inkompressible viskose Strömungen

Im Praktikum wollen wir diese Schritte an einem konkreten Beispiel aus der Strömungsmechanik erlernen. Wir betrachten dazu die Behandlung viskoser, instationärer, laminarer Strömungen.
Dabei beschränken wir uns auf *inkompressible* Medien, wie beispielsweise Wasser, die, wie der Name sagt, nicht komprimierbar sind. Luft wäre hingegen ein Beispiel für ein kompressibles Medium, bei dem ein und dieselbe Masse unterschiedliche Volumina ausfüllen kann. Die *Viskosität* beschreibt die Zähigkeit des Mediums. So besitzt Luft eine sehr geringe, Wasser eine etwas größere und Honig eine hohe Viskosität. Bei nicht zu hohen Geschwindigkeiten und nicht zu geringen Viskositäten sind Strömungen relativ regelmäßig und wenig mischungsintensiv, d.h. *laminar* und nicht turbulent. *Instationär* bedeutet schließlich, dass sich die Strömung im Gegensatz zu stationären Strömungen mit der Zeit verändern kann.
Beispiele aus der Praxis sind in den Abbildungen 2 und 3 zu sehen.

3 Das mathematische Modell: Die Navier–Stokes–Gleichungen

Instationäre, inkompressible, viskose, laminare Strömungen lassen sich durch die *Navier–Stokes–Gleichungen* beschreiben. Der Einfachheit halber beschränken wir uns dabei auf den zweidimensionalen Fall und legen ein kartesisches Koordinatensystem zu Grunde.
Dann erhalten wir ein System partieller Differentialgleichungen, bestehend aus den zwei *Impulsgleichungen*

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

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

und der *Kontinuitätsgleichung*

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{3}$$

das zunächst auf dem rechteckigen Gebiet $\Omega := [0, a] \times [0, b] \subset \mathbb{R}^2$ in dem Zeitintervall $[0, t_{end}]$ gelöst werden soll. [1]
Dabei sind die drei gesuchten Größen

- $u : \Omega \times [0, t_{end}] \to \mathbb{R}$ die Geschwindigkeit des Fluids in x–Richtung,
- $v : \Omega \times [0, t_{end}] \to \mathbb{R}$ die Geschwindigkeit des Fluids in y–Richtung,
- $p : \Omega \times [0, t_{end}] \to \mathbb{R}$ der Druck.

Die reelle Zahl Re ist die *Reynoldszahl*, die die Zähigkeit des Fluids angibt. Je kleiner Re ist, umso zäher ist das Fluid.

[1] Neben der hier verwendeten Formulierung mit den Unbekannten (u, v, p) existiert auch eine Formulierung mit der Stromfunktion ψ und der Wirbelfunktion ω als Unbekannte (siehe auch Blatt 2).

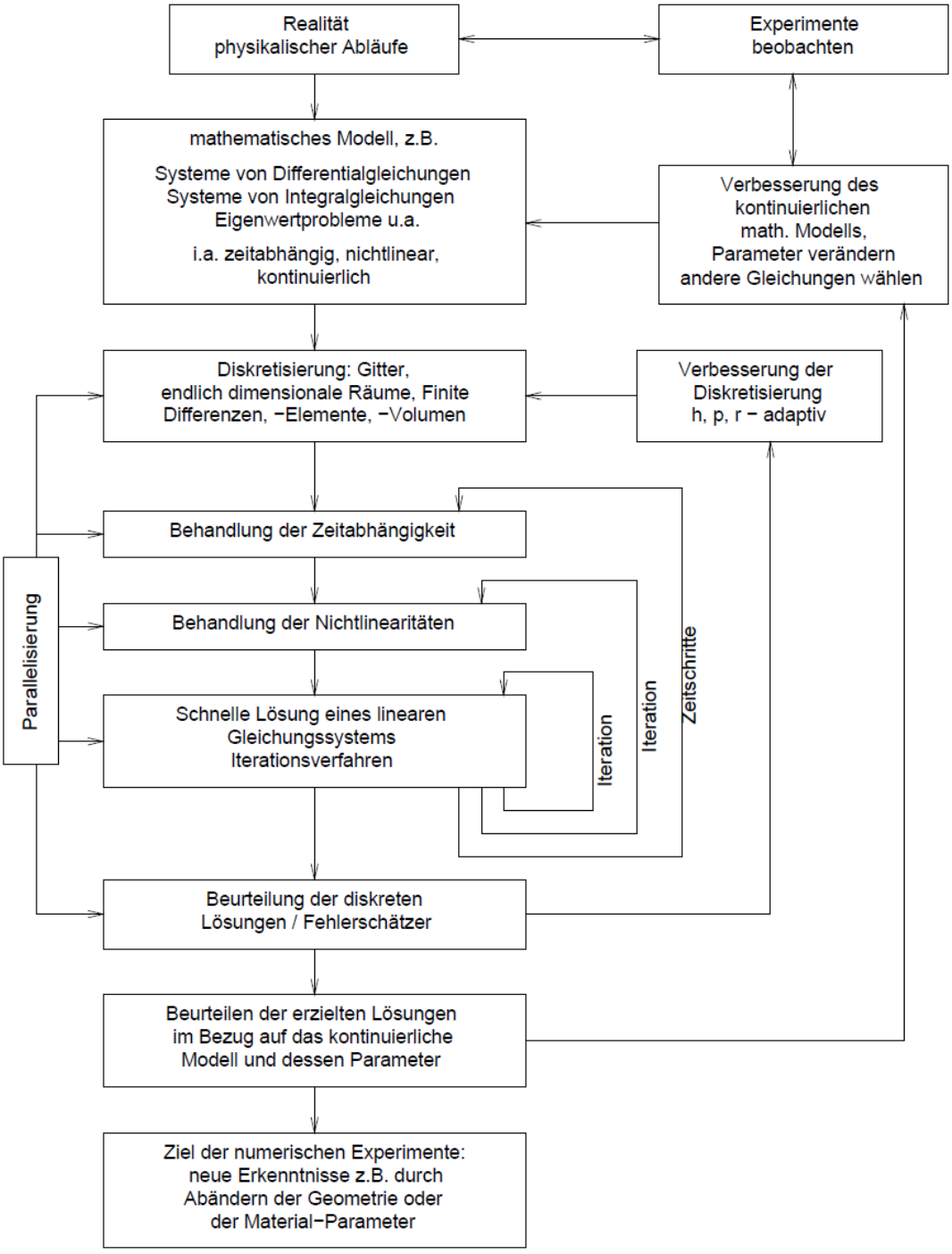


Abbildung 1: Typische Vorgehensweise in einem Teilgebiet des Wissenschaftlichen Rechnens

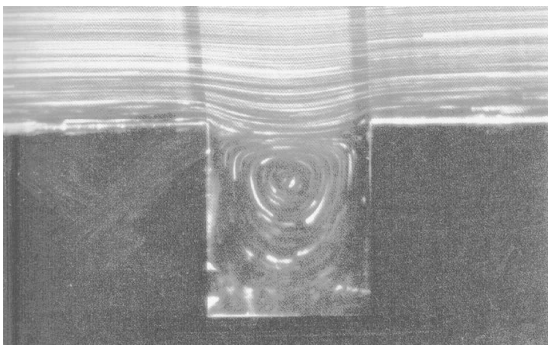


Abbildung 2: Strömung über einen Hohlraum

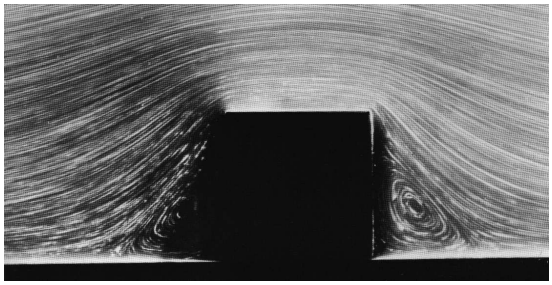


Abbildung 3: Strömung über eine Stufe

$g_x : \Omega \times [0, t_{end}] \to \mathbb{R}$ und $g_y : \Omega \times [0, t_{end}] \to \mathbb{R}$ sind äußere Kräfte, etwa die Erdanziehung oder andere Beschleunigungen, denen das System unterworfen ist. In der Literatur findet man häufig nicht die obige, komponentenweise Notation sondern die kompaktere Schreibweise

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) &= \mathbf{g} - \nabla p + \nu \Delta \mathbf{u} \ , \\ \nabla \cdot \mathbf{u} &= 0 \ . \end{aligned}$$

Dabei ist $\mathbf{u} = (u, v)^T$ bzw. $(u, v, w)^T$ in 3D, $\mathbf{g} = (g_x, g_y\{, g_z\})^T$ und $\mathbf{u} \otimes \mathbf{u}$ die 2×2 (3×3) Matrix $\mathbf{u} \cdot \mathbf{u}^T$. Die Divergenz $\nabla\cdot$ einer Matrix ist dabei wie üblich der Vektor der Divergenzen der Spalten.
Zu Beginn $(t = 0)$ seien Anfangsbedingungen $u = u_0(x, y)$ und $v = v_0(x, y)$ vorgegeben, die (3) erfüllen. Zusätzlich sind zu allen Zeitpunkten Bedingungen an den 4 Gebietsrändern erforderlich, so dass wir insgesamt ein *Anfangs–Randwertproblem* erhalten.

Für die Formulierung der Randbedingungen bezeichne w_1 die Geschwindigkeitskomponente senkrecht zum Rand (in Normalenrichtung), w_2 die Geschwindigkeitskomponente parallel zum Rand (in Tangentialrichtung) und $\partial w_1/\partial n$ bzw. $\partial w_2/\partial n$ die Ableitung in Normalenrichtung.

Am senkrechten Rand gilt also

$$w_1 = u, \qquad w_2 = v, \qquad \frac{\partial w_1}{\partial n} = \frac{\partial u}{\partial x}, \qquad \frac{\partial w_2}{\partial n} = \frac{\partial v}{\partial x},$$

und am waagrechten Rand gilt

$$w_1 = v, \qquad w_2 = u, \qquad \frac{\partial w_1}{\partial n} = \frac{\partial v}{\partial y}, \qquad \frac{\partial w_2}{\partial n} = \frac{\partial u}{\partial y}.$$

Für Punkte (x, y) des festen Randes $\Gamma := \partial\Omega$ sind folgende Randbedingungen möglich:

1. Haftbedingung (no–slip): $w_1(x, y) = 0$, $w_2(x, y) = 0$. (Fluid haftet am Rand)
2. Rutschbedingung (free–slip): $w_1(x, y) = 0$, $\partial w_2(x, y)/\partial n = 0$. (keine Reibung am Rand)
3. Einströmbedingung (inflow): $w_1(x, y) = w_1^0$, $w_2(x, y) = w_2^0$. w_1^0, w_2^0 gegeben
4. Ausströmbedingung (outflow): $\partial w_1(x, y)/\partial n = 0$, $\partial w_2(x, y)/\partial n = 0$.

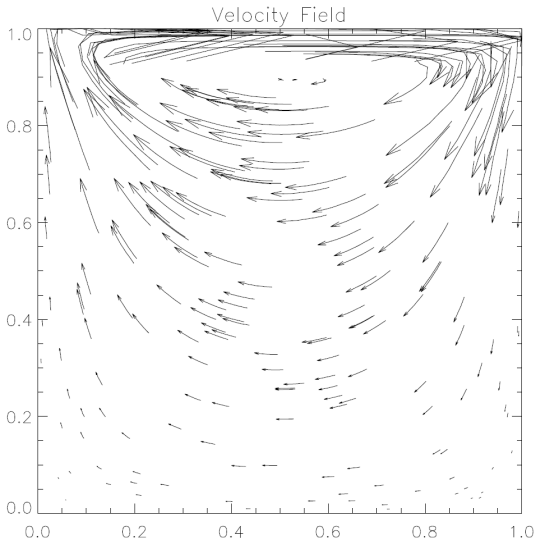


Abbildung 4: Driven Cavity (Geschwindigkeitsfeld): 4 Wände mit Haftbedingungen. Die obere Wand bewegt sich mit konstanter Geschwindigkeit nach rechts, d.h. $w_2 = w_2^0 = const.$ am oberen Rand, im Gegensatz zu $w_2 = 0$ bei gewöhnlichen Haftbedingungen (siehe auch Abschnitt 9.)

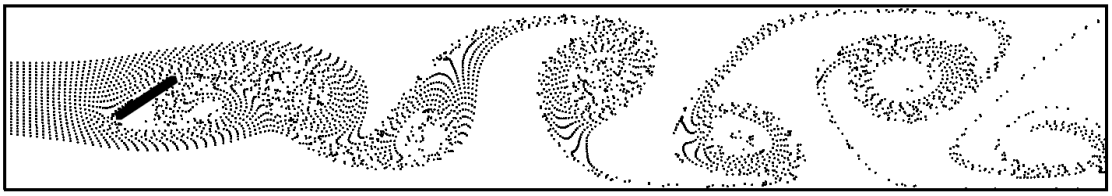


Abbildung 5: Strömung über einen schrägen Balken (Streichlinien): links: Einströmung (inflow), rechts: Ausströmung (outflow) oben und unten starre Wände (no-slip)

Sind an allen Rändern die Geschwindigkeiten selbst und nicht deren Normalenableitungen vorgegeben, so muss zusätzlich das Randintegral über die Geschwindigkeiten senkrecht zum Rand Null sein, d.h.

$$\int_\Gamma \begin{pmatrix} u \\ v \end{pmatrix} \cdot n \, ds = 0$$

Beispiele für Strömungen mit unterschiedlichen Randbedingungen sind etwa:

Mit physikalisch richtigen Randbedingungen erhalten wir insgesamt ein „gut–gestelltes“Problem, das eine eindeutige Lösung besitzt.
Diese Lösung ist jedoch im allgemeinen Fall nicht analytisch berechenbar, d.h. sie kann nicht in einer geschlossenen Formel angegeben werden. Sie ist meist nur *numerisch* approximierbar.

4 Die Diskretisierung

Mit dem Begriff *Diskretisierung* beschreibt man in der Numerik den übergang von einem kontinuierlichen Problem zu einem Problem, das nur in endlich vielen Punkten betrachtet wird. Im einfachsten Fall wird das Gebiet Ω durch ein Gitter überdeckt, und das zu lösende Problem wird statt auf dem ganzen Gebiet jetzt nur noch in den Kreuzungspunkten der Gitterlinen betrachtet. Das Gitter habe in x–Richtung $imax$ und in y–Richtung $jmax$ Zellen der gleichen Größe, so dass die Gitterlinien die Abstände $\delta x := a/imax$ und $\delta y := b/jmax$ haben. Die Zelle mit dem Index (i, j) entspricht dem Gebiet $[(i-1)\,\delta x, i\,\delta x] \times [(j-1)\,\delta y, j\,\delta y]$. Die Differentialoperatoren lassen sich dann durch Differenzensterne ersetzen. Gemäß der Definition der Ableitung

$$\frac{df}{dx} := \lim_{\delta x \to 0} \frac{f(x+\delta x) - f(x)}{\delta x}$$

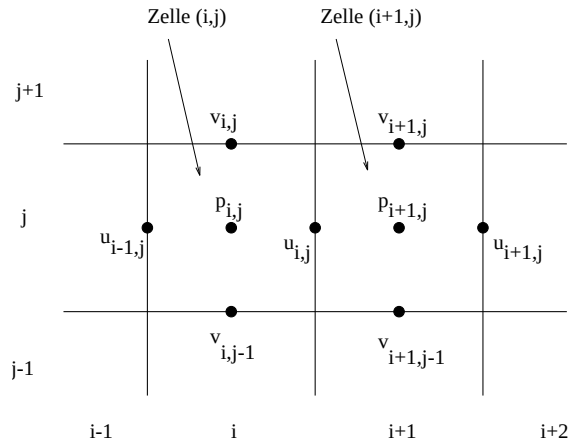


Abbildung 6: Staggered Grid

wird so im Eindimensionalen der kontinuierliche Differentialoperator df/dx durch den Differenzenoperator $\delta f/\delta x := (f(x+\delta x) - f(x))/\delta x$ zur Schrittweite δx approximiert, indem die Limesbildung weggelassen wird. x und $x + \delta x$ sind dabei Gitterpunkte. Wir erhalten so die Methode der *finiten Differenzen*[2]. Anschaulich ist klar, dass eine Verfeinerung des Gitters, d.h eine Verkleinerung der Schrittweite δx, zu einer besseren Approximation des Differentialquotienten führt.

Im Fall der Navier–Stokes–Gleichungen wird zur Diskretisierung des Gebietes Ω oftmals ein versetztes Gitter ("staggered grid") verwendet, bei dem die verschiedenen gesuchten Variablen nicht in den selben Gitterpunkten liegen. Wir verwenden ein Gitter, in dem der Druck p nur in den Mittelpunkten der Zellen, die Geschwindigkeit u nur in den Mittelpunkten der senkrechten Zellkanten und die Geschwindigkeit v nur in den Mittelpunkten der waagrechten Zellkanten betrachtet wird. Den Index (i, j) tragen der Druck im Mittelpunkt der Zelle (i, j), die u–Geschwindigkeit an der rechten Kante und die v–Geschwindigkeit an der oberen Kante der Zelle (i, j). Der Druck $p_{i,j}$ liegt also bei den Koordinaten $((i - 0.5)\,\delta x, (j - 0.5)\,\delta y)$, $u_{i,j}$ bei den Koordinaten $(i\,\delta x, (j - 0.5)\,\delta y)$ und $v_{i,j}$ bei den Koordinaten $((i - 0.5)\,\delta x, j\,\delta y)$ (vgl. Abbildung 6).

Die Punkte für u, v und p gehören also zu drei, hier nicht dargestellten Gittern, die jeweils um eine halbe Maschenweite versetzt sind.

In Folge dessen kommen nicht alle Werte auf dem Rand des Gebietes zu liegen. So gibt es etwa auf den senkrechten Rändern keine v–Werte und auf den waagrechten Rändern keine u–Werte. Daher wird noch eine Randschicht aus Gitterzellen mitgeführt (vgl. Abbildung 7), und die Randbedingungen werden durch eine Mittelung der nächstliegenden Werte erfüllt.

Weiterhin wird auch das Zeitintervall $[0, t_{end}]$ in gleiche Teilintervalle $[n\,\delta t, (n + 1)\,\delta t]$, $n = 0, \ldots, t_{end}/\delta t - 1$ zerlegt. Werte für u, v und p werden also nur zu den Zeitpunkten $n\,\delta t$ betrachtet.

Die Navier–Stokes–Gleichungen werden jetzt wie folgt diskretisiert:

Zunächst wollen wir die Ortsableitungen behandeln. Die Impulsgleichung (1) wird in den Mittelpunkten der senkrechten Kanten, die Impulsgleichung (2) wird in den Mittelpunkten der waagrechten Kanten, und die Kontinuitätsgleichung (3) wird in den Zellmittelpunkten ausgewertet.

Die Ausdrücke in Gleichung (1) ersetzen wir am Mittelpunkt der rechten Kante der Zelle (i, j), $i = 1, \ldots, imax - 1$, $j = 1 \ldots, jmax$, durch

[2]Daneben existieren eine Reihe anderer Diskretisierungsverfahren, wie *finite Elemente* oder *finite Volumen*.

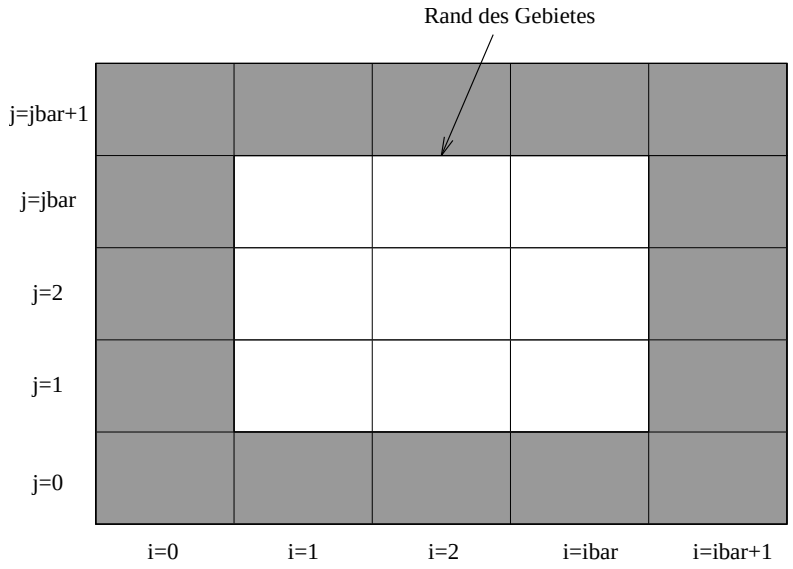


Abbildung 7: Gebiet mit Randzellen

$$
\begin{aligned}
\left[\frac{\partial(u^2)}{\partial x}\right]_{i,j} &:= \frac{1}{\delta x}\left(\left(\frac{u_{i,j}+u_{i+1,j}}{2}\right)^2-\left(\frac{u_{i-1,j}+u_{i,j}}{2}\right)^2\right)+\\
&\quad \alpha\frac{1}{\delta x}\left(\frac{|u_{i,j}+u_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i+1,j})}{2}-\frac{|u_{i-1,j}+u_{i,j}|}{2}\frac{(u_{i-1,j}-u_{i,j})}{2}\right),\\
\left[\frac{\partial(uv)}{\partial y}\right]_{i,j} &:= \frac{1}{\delta y}\left(\frac{(v_{i,j}+v_{i+1,j})}{2}\frac{(u_{i,j}+u_{i,j+1})}{2}-\frac{(v_{i,j-1}+v_{i+1,j-1})}{2}\frac{(u_{i,j-1}+u_{i,j})}{2}\right)+\\
&\quad \alpha\frac{1}{\delta y}\left(\frac{|v_{i,j}+v_{i+1,j}|}{2}\frac{(u_{i,j}-u_{i,j+1})}{2}-\frac{|v_{i,j-1}+v_{i+1,j-1}|}{2}\frac{(u_{i,j-1}-u_{i,j})}{2}\right) \qquad (4)\\
\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j} &:= \frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{(\delta x)^2},\\
\left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j} &:= \frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{(\delta y)^2}, \qquad \left[\frac{\partial p}{\partial x}\right]_{i,j} := \frac{p_{i+1,j}-p_{i,j}}{\delta x}.
\end{aligned}
$$

Die Ausdrücke in Gleichung (2) ersetzen wir am Mittelpunkt der oberen Kante der Zelle (i,j), $i=1,\ldots,imax$, $j=1\ldots,jmax-1$, durch

$$
\begin{aligned}
\left[\frac{\partial(uv)}{\partial x}\right]_{i,j} &:= \frac{1}{\delta x}\left(\frac{(u_{i,j}+u_{i,j+1})}{2}\frac{(v_{i,j}+v_{i+1,j})}{2}-\frac{(u_{i-1,j}+u_{i-1,j+1})}{2}\frac{(v_{i-1,j}+v_{i,j})}{2}\right)+\\
&\quad \alpha\frac{1}{\delta x}\left(\frac{|u_{i,j}+u_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i+1,j})}{2}-\frac{|u_{i-1,j}+u_{i-1,j+1}|}{2}\frac{(v_{i-1,j}-v_{i,j})}{2}\right),\\
\left[\frac{\partial(v^2)}{\partial y}\right]_{i,j} &:= \frac{1}{\delta y}\left(\left(\frac{v_{i,j}+v_{i,j+1}}{2}\right)^2-\left(\frac{v_{i,j-1}+v_{i,j}}{2}\right)^2\right)+\\
&\quad \alpha\frac{1}{\delta y}\left(\frac{|v_{i,j}+v_{i,j+1}|}{2}\frac{(v_{i,j}-v_{i,j+1})}{2}-\frac{|v_{i,j-1}+v_{i,j}|}{2}\frac{(v_{i,j-1}-v_{i,j})}{2}\right), \qquad (5)\\
\left[\frac{\partial^2 v}{\partial x^2}\right]_{i,j} &:= \frac{v_{i+1,j}-2v_{i,j}+v_{i-1,j}}{(\delta x)^2},\\
\left[\frac{\partial^2 v}{\partial y^2}\right]_{i,j} &:= \frac{v_{i,j+1}-2v_{i,j}+v_{i,j-1}}{(\delta y)^2}, \qquad \left[\frac{\partial p}{\partial y}\right]_{i,j} := \frac{p_{i,j+1}-p_{i,j}}{\delta y}
\end{aligned}
$$

und die Terme in der Kontinuitätsgleichung (3) ersetzen wir im Mittelpunkt der Zelle

(i,j), $i = 1,\ldots,imax$, $j = 1\ldots,jmax$, durch

$$\left[\frac{\partial u}{\partial x}\right]_{i,j} := \frac{u_{i,j} - u_{i-1,j}}{\delta x}, \qquad \left[\frac{\partial v}{\partial y}\right]_{i,j} := \frac{v_{i,j} - v_{i,j-1}}{\delta y}. \tag{6}$$

Dabei ist α ein Parameter zwischen 0 und 1. Mit $\alpha = 0$ erhält man für die Differentialoperatoren eine Approximation 2. Ordnung, d.h. der Approximationsfehler hat die Genauigkeit $O(\delta x^2)$ bzw. $O(\delta y^2)$. Diese Approximation kann jedoch für geringe Viskositätswerte, oder wenn die Ableitungsterme $\partial/\partial x$ bzw. $\partial/\partial y$ mit wesentlich größeren Faktoren in das Problem eingehen als die Ableitungsterme $\partial^2/\partial x^2$ und $\partial^2/\partial y^2$, zu Oszillationen in der Lösung führen. In solchen Fällen muss das sogenannte Donor–Cell–Schema ($\alpha = 1$) verwendet werden, das nur eine Approximation 1. Ordnung liefert. In der Praxis benutzt man eine Mischung aus beiden Verfahren mit $\alpha \in [0,1]$. Der Parameter α sollte dabei etwas größer als der maximale Wert von $|u\,\delta t/\delta x|$ und $|v\,\delta t/\delta y|$ im Gitter gewählt werden.

Bei der Diskretisierung der Impulsgleichungen (1) und (2) bezüglich der Zeit werden die Terme auf der linken Seite zum Zeitpunkt t_{n+1} und die Terme auf der rechten Seite zum Zeitpunkt t_n ausgewertet. Die Zeitableitung $\partial u/\partial t$ zur Zeit t_{n+1} wird durch den Differenzenquotienten 1. Ordnung $(u^{(n+1)} - u^{(n)})/\delta t$ ersetzt. Dies entspricht einem expliziten Zeitschritt–Verfahren mit Hilfe der Euler–Formel[3].

Die Kontinuitätsgleichung (3) wird zum Zeitpunkt t_{n+1} ausgewertet.

5 Der Algorithmus

Das gesamte Lösungsverfahren läßt sich nun mit den folgenden Einzelschritten beschreiben.

5.1 Die Zeitschleife

In einer äußeren Iterationsschleife wird beginnend beim Zeitpunkt $t = 0$ die Zeit solange um δt erhöht, bis die Endzeit t_{end} erreicht ist. Im Zeitschritt n ($n = 0,1,\ldots$) werden die Differentialgleichungen wie oben beschrieben diskretisiert. Dabei seien die Werte zum Zeitpunkt t_n bereits bekannt, und die Werte zum Zeitpunkt t_{n+1} sollen berechnet werden. In der kompakten Notation hat der gesamte Algorithmus die folgende Form

(Transport Schritt)

$$\frac{\mathbf{F}^{(n)} - \mathbf{u}^{(n)}}{\partial t} = \mathbf{g} - \nabla\cdot(\mathbf{u}^{(n)} \otimes \mathbf{u}^{(n)}) + \nu\Delta\mathbf{u}^{(n)} \tag{7}$$

(Projektions-Schritt) Löse das Sattelpunkt Problem

$$\frac{\mathbf{u}^{(n+1)} - \mathbf{F}^{(n)}}{\partial t} = -\nabla p^{(n+1)} \tag{8}$$

$$\nabla\cdot\mathbf{u}^{(n+1)} = 0 \quad . \tag{9}$$

Dies geschieht, indem man auf Gleichung (8) den Divergenzoperator anwendet und Gleichung (9) ausnutzt. Dann erhält man die sogenannte Druck-Poisson Gleichung

$$\Delta p^{(n+1)} = \frac{1}{\partial t}\nabla\cdot\mathbf{F}^{(n)} \ .$$

[3]Es gibt auch implizite Verfahren, bei denen auf der rechten Seite auch Werte zum Zeitpunkt t_{n+1} vorkommen. Diese erlauben wesentlich größere Zeitschritte. Die Lösung der entstehenden Gleichungssysteme ist aber meist wesentlich aufwendiger.

Verfahren der obigen Bauart werden in der Literatur als Splitting-Methoden bezeichnet. Der Grund liegt in der getrennten (gesplittet) Behandlung der Impulsgleichung bzw. der Kontinuitätsgleichung. Die *entkoppelte* Behandlung beider Gleichungen hat enorme algorithmische Vorteile. Splitting Methoden sind deshalb die wohl effizientesten Methoden für die instationären, inkompressiblen Navier-Stokes Gleichungen. Ihr Prototyp ist (1968) von A.J. Chorin (unabhänging davon auch durch R. Temam) angegeben worden, so daß man oft auch die Bezeichnung Chorinsche Projektionsmethode liest.

5.2 Die diskreten Impulsgleichungen

Die diskreten Impulsgleichungen werden so umgestellt, dass auf der linken Seite nur noch die Geschwindigkeit $u_{i,j}^{(n+1)}$ bzw. $v_{i,j}^{(n+1)}$ stehen. Dann erhalten die Gleichungen die Form

$$\begin{aligned} u_{i,j}^{(n+1)} &= F_{i,j}^{(n)} - \frac{\delta t}{\delta x}(p_{i+1,j}^{(n+1)} - p_{i,j}^{(n+1)}) \qquad (10)\\ &\qquad\qquad i = 1,\ldots,imax-1, \quad j = 1,\ldots,jmax;\\ v_{i,j}^{(n+1)} &= G_{i,j}^{(n)} - \frac{\delta t}{\delta y}(p_{i,j+1}^{(n+1)} - p_{i,j}^{(n+1)}) \qquad (11)\\ &\qquad\qquad i = 1,\ldots,imax, \quad j = 1,\ldots,jmax-1. \end{aligned}$$

Die Terme $F_{i,j}^{(n)}$ und $G_{i,j}^{(n)}$ beinhalten die in den diskreten Impulsgleichungen vorkommenden Geschwindigkeiten zum Zeitpunkt n. Mit den diskretisierten Ausdrücken aus (4) und (5) sind dies:

$$\begin{aligned} F_{i,j} &:= u_{i,j} + \delta t\left(\frac{1}{Re}\left(\left[\frac{\partial^2 u}{\partial x^2}\right]_{i,j} + \left[\frac{\partial^2 u}{\partial y^2}\right]_{i,j}\right) - \left[\frac{\partial (u^2)}{\partial x}\right]_{i,j} - \left[\frac{\partial (uv)}{\partial y}\right]_{i,j} + g_x\right) \qquad (12)\\ &\qquad\qquad i = 1,\ldots,imax-1, \quad j = 1,\ldots,jmax;\\ G_{i,j} &:= v_{i,j} + \delta t\left(\frac{1}{Re}\left(\left[\frac{\partial^2 v}{\partial x^2}\right]_{i,j} + \left[\frac{\partial^2 v}{\partial y^2}\right]_{i,j}\right) - \left[\frac{\partial (uv)}{\partial x}\right]_{i,j} - \left[\frac{\partial (v^2)}{\partial y}\right]_{i,j} + g_y\right) \qquad (13)\\ &\qquad\qquad i = 1,\ldots,imax, \quad j = 1,\ldots,jmax-1. \end{aligned}$$

5.3 Die Druckgleichung

Jetzt können wir die durch die rechten Seiten von (10) und (11) gegebenen Ausdrücke für die Geschwindigkeiten $u_{i-1,j}^{(n+1)}, u_{i,j}^{(n+1)}, v_{i,j-1}^{(n+1)}$ und $v_{i,j}^{(n+1)}$ in die diskrete Kontinuitätsgleichung der Zelle (i,j) einsetzen und erhalten eine Gleichung, die nur noch Druckwerte zum Zeitpunkt t_{n+1} und (bereits bekannte) Geschwindigkeiten zum Zeitpunkt t_n enthält:

$$\begin{aligned} &\frac{p_{i+1,j}^{(n+1)} - 2p_{i,j}^{(n+1)} + p_{i-1,j}^{(n+1)}}{(\delta x)^2} + \frac{p_{i,j+1}^{(n+1)} - 2p_{i,j}^{(n+1)} + p_{i,j-1}^{(n+1)}}{(\delta y)^2} =\\ &\qquad\qquad \frac{1}{\delta t}\left(\frac{F_{i,j}^{(n)} - F_{i-1,j}^{(n)}}{\delta x} + \frac{G_{i,j}^{(n)} - G_{i,j-1}^{(n)}}{\delta y}\right), \qquad (14)\\ &\qquad\qquad i = 1,\ldots,imax, \quad j = 1,\ldots,jmax. \end{aligned}$$

Dies ist die bekannte Form einer diskretisierten Poisson–Gleichung für die Größe $p^{(n+1)}$

$$\frac{\partial^2 p^{(n+1)}}{\partial x^2} + \frac{\partial^2 p^{(n+1)}}{\partial y^2} = rhs$$

auf dem Gebiet Ω mit einer beliebigen rechten Seite rhs. Zur eindeutigen Lösbarkeit benötigen wir aber noch die Randwerte $p_{i,j}$ $(i = 0,\ imax+1,\ j = 0,\ jmax+1)$, $F_{i,j}$

$(i = 0, imax)$ und $G_{i,j}$ $(j = 0, jmax)$, die wir aus den Impulsgleichungen am Rand gewinnen können (siehe unten).
Jetzt ist es möglich, die Druckgleichung (14) mit einem beliebigen Verfahren zur Lösung linearer Gleichungssysteme zu lösen. Da direkte Verfahren wie die Gaußelimination bei sehr großen Problemen (hier $imax \cdot jmax$ Unbekannte) einen zu großen Rechenaufwand besitzen, kommen bei der numerischen Lösung partieller Differentialgleichungen gewöhnlich iterative Gleichungslöser zum Einsatz. Bei unserem Problem haben wir jeweils für die Lösung der Gleichungssysteme eine recht gute Näherung: den Druck $p^{(n)}$ aus dem alten Zeitschritt. Nur iterative Verfahren können das effizient ausnutzen.
Ein klassisches iteratives Verfahren ist das Gauß–Seidel–Verfahren, bei dem, ausgehend von einem Startwert, in jedem Zyklus sukzessive alle Zellen abgelaufen werden und in der Zelle (i, j) der Druck in dieser Zelle so geändert wird, dass die zugehörige Gleichung exakt erfüllt wird.
Eine verbesserte Variante ist das SOR–Verfahren:

$$
\begin{array}{l}
it = 1, \ldots, itmax \\
\quad i = 1, \ldots, imax \\
\quad\quad j = 1, \ldots, jmax \\
\quad\quad\quad p_{i,j}^{it+1} := (1-\omega)\, p_{i,j}^{it} + \\
\quad\quad\quad\quad \dfrac{\omega}{2(\frac{1}{(\delta x)^2} + \frac{1}{(\delta y)^2})} \left(\dfrac{p_{i+1,j}^{it} + p_{i-1,j}^{it+1}}{(\delta x)^2} + \dfrac{p_{i,j+1}^{it} + p_{i,j-1}^{it+1}}{(\delta y)^2} - rhs_{i,j} \right)
\end{array}
\tag{15}
$$

$rhs_{i,j}$ ist die rechte Seite der Druckgleichung (14) für die Zelle (i, j) und ω ist ein Parameter (Relaxationsfaktor), der aus dem Intervall $]0, 2]$ gewählt werden muss (häufig $\omega = 1.7$). Für $\omega = 1$ ergibt sich das Gauß–Seidel–Verfahren. Die Iteration wird abgebrochen, wenn die maximale Anzahl von Iterationsschritten $itmax$ erreicht ist oder wenn das Residuum

$$
res := \left(\sum_{i=1}^{imax} \sum_{j=1}^{jmax} \left(\frac{p_{i+1,j} - 2p_{i,j} + p_{i-1,j}}{(\delta x)^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{(\delta y)^2} - rhs_{i,j} \right)^2 / (imax \cdot jmax) \right)^{1/2} \tag{16}
$$

eine vom Benutzer zu wählende Toleranzgrenze ε unterschritten hat.
Als Startwert der Iteration zur Berechnung der Druckwerte $p^{(n+1)}$ dienen jeweils die Druckwerte zum Zeitpunkt n.
Mit den so berechneten Druckwerten zum Zeitpunkt t_{n+1} können dann auch die Geschwindigkeitswerte u und v zum Zeitpunkt t_{n+1} gemäß (10) und (11) berechnet werden.

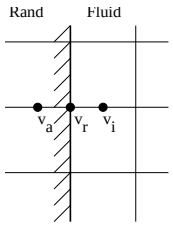
5.4 Die Randwerte für die diskreten Gleichungen bei Haftbedingungen

Bei der Berechnung von F und G nach (12) und (13) wird für $i = 1, imax$ und $j = 1, jmax$ auf u– und v–Werte zugegriffen, die auf dem Gebietsrand oder sogar außerhalb liegen können. Wir wollen uns hier zunächst auf Haftbedingungen beschränken, bei denen die kontinuierlichen Geschwindigkeiten am Rand jeweils 0 sein sollen. Also setzen wir für die Werte, die direkt auf dem Rand liegen

$$
u_{0,j} = 0,\ u_{imax,j} = 0,\ j = 1, \ldots, jmax \qquad v_{i,0} = 0,\ v_{i,jmax} = 0,\ i = 1, \ldots, imax. \tag{17}
$$

An den 4 Rändern erhalten wir somit die weiteren Bedingungen:

$$
\begin{array}{ll}
v_{0,j} = -v_{1,j}, & v_{imax+1,j} = -v_{imax,j}, \quad j = 1, \ldots, jmax, \\
u_{i,0} = -u_{i,1}, & u_{i,jmax+1} = -u_{i,jmax}, \quad i = 1, \ldots, imax.
\end{array}
\tag{18}
$$



Da weiterhin auf den senkrechten Rändern keine v–Werte und auf den waagrechten Rändern keine u–Werte liegen, wird hier der Randwert 0 durch eine Mittelung der Werte zu beiden Seiten des Randes erzielt:

$$v_r := \frac{v_a + v_i}{2} = 0 \quad \Rightarrow \quad v_a = -v_i.$$

Wie schon erwähnt, wird der Druck am Rand aus den diskreten Impulsgleichungen hergeleitet, und zwar durch Multiplikation mit dem jeweiligen Normalenvektor. Es ergibt sich insgesamt:

$$\begin{array}{llll} p_{0,j} = p_{1,j}, & p_{imax+1,j} = p_{imax,j}, & j = 1, \ldots, jmax; \\ p_{i,0} = p_{i,1}, & p_{i,jmax+1} = p_{i,jmax}, & i = 1, \ldots, imax \end{array} \tag{19}$$

und

$$\begin{array}{llll} F_{0,j} = u_{0,j}, & F_{imax,j} = u_{imax,j}, & j = 1, \ldots, jmax, \\ G_{i,0} = v_{i,0}, & G_{i,jmax} = v_{i,jmax}, & i = 1, \ldots, imax. \end{array} \tag{20}$$

5.5 Die Stabilitätsbedingung

Um die Stabilität des numerischen Algorithmus zu gewährleisten und keine Oszillationen zu erzeugen, müssen schließlich die folgenden drei Stabilitätsbedingungen an die Schrittweiten δx, δy und δt gestellt werden:

$$\frac{2}{Re}\,\delta t < \frac{(\delta x)^2(\delta y)^2}{(\delta x)^2 + (\delta y)^2}, \qquad |u_{max}|\,\delta t < \delta x, \qquad |v_{max}|\,\delta t < \delta y.^4 \tag{21}$$

$|u_{max}|$ und $|v_{max}|$ sind die maximalen Absolutwerte der vorkommenden Geschwindigkeiten.

Basierend auf diesen Stabilitätsbedingungen kann eine adaptive Zeitschrittweitensteuerung verwendet werden. Dabei wird δt für den nächsten Zeitschritt so gewählt, dass jede der drei Bedingungen erfüllt ist:

$$\delta t := \tau \, \min\left(\frac{Re}{2} \left(\frac{1}{\delta x^2} + \frac{1}{\delta y^2} \right)^{-1}, \frac{\delta x}{|u_{max}|}, \frac{\delta y}{|v_{max}|} \right). \tag{22}$$

Der Faktor $\tau \in]0, 1]$ ist ein Sicherheitsfaktor.

5.6 Zusammenfassung

Abschließend sei der Algorithmus nochmals übersichtlich dargestellt:

[4] Die letzten beiden Bedingungen sind die sogenannten Courant–Friedrichs–Levi–(CFL)–Bedingungen.

```
Einlesen der Problemparameter
Setze t := 0, n := 0
Belege u, v, p mit Anfangswerten
Solange t < t_end
    Wähle δt nach (22)
    Setze die Randwerte für u und v gemäß (17),(18)
    Berechne F^(n) und G^(n) gemäß (12),(13),(20)
    Berechne die rechte Seite rhs der Druckgleichung (14)
    Setze it := 0
    Solange it < itmax und res > ε
       Setze die Randwerte für den Druck gemäß (19)
       Führe einen SOR-Zyklus gemäß (15) durch
       Berechne das Residuum res der Druckgleichung gemäß (16)
       it := it + 1
    Berechne u^(n+1) und v^(n+1) gemäß (10),(11)
    Gegebenenfalls Ausgabe der Werte u, v, p zur Visualisierung
    t := t + δt
    n := n + 1
Ausgabe der Werte u, v, p zur Visualisierung
```

6 Die Problemgrößen und Datenstrukturen

Der oben beschriebene Algorithmus soll im Praktikum in der Programmiersprache Python und C++ realisiert werden. Er benötigt die folgenden Größen, die, bis auf die Variablen `t`, `it`, `delx` und `dely`, am besten in einer Eingabedatei zur Verfügung gestellt werden. `REAL` steht dabei jeweils für `float` in Python bzw. einen template parameter in C++, der wahlweise `float` oder `double` sein kann.

- Geometriegrößen:

`REAL xlength`	Gebietslänge in x–Richtung
`REAL ylength`	Gebietslänge in y–Richtung
	Das Berechnungsgebiet ist somit $\Omega = [0, \texttt{xlength}] \times [0, \texttt{ylength}]$.
`int imax`	Anzahl der inneren Zellen in x–Richtung
`int jmax`	Anzahl der inneren Zellen in y–Richtung
`REAL delx`	Länge einer Zelle in x–Richtung δx
`REAL dely`	Länge einer Zelle in y–Richtung δy

- Größen für die Zeititeration:

`REAL t`	aktuelle Zeit
`REAL t_end`	Endzeit t_{end}
`REAL delt`	Zeitschrittweite δt
`REAL tau`	Sicherheitsfaktor für die Zeitschrittweitensteuerung τ
`REAL del_vec`	Zeitabstand, in dem Daten zur Visualisierung in eine Datei geschrieben werden sollen

- Parameter für die Druckiteration:

`int itermax`	Maximale Anzahl von Druck-Iterationen pro Zeitschritt
`int it`	Zähler für die SOR–Iterationen
`REAL res`	Norm des Residuums der Druckgleichung
`REAL eps`	Genauigkeitskriterium ε für die Druckiteration (res < eps)
`REAL omg`	Relaxationsfaktor ω für die SOR-Iteration
`REAL alpha`	Upwind-Differencing-Faktor α

- Problemabhängige Größen:

`REAL Re`	Reynoldszahl Re
`REAL GX,GY`	äußere Kräfte g_x, g_y, z.B. Gravitation
`REAL UI,VI,PI`	Anfangsbelegung der Geschwindigkeiten und des Drucks

Weiterhin sollen die folgenden Felder der Dimension `[0,imax+1]x[0,jmax+1]` als Datenstrukturen verwendet werden:

- Datenfelder

`Matrix<REAL> U`	Geschwindigkeit in x–Richtung
`Matrix<REAL> V`	Geschwindigkeit in y–Richtung
`Matrix<REAL> P`	Druck
`Matrix<REAL> RHS`	Rechte Seite für die Druckiteration
`Matrix<REAL> F,G`	F, G

Exercise 32. (Driven Cavity)

a) Schreiben Sie unter Verwendung obiger Größen und Datenstrukturen die Prozeduren, welche die folgigen Funktionalitäten bereitsstellen:

- Einlesen einer Parameterdatei zu welcher der Dateipfad übergeben wird. Zusätzlich werden die Werte `delx`, `dely` aus `imax` und `xlength` bzw. `jmax` und `ylength` bestimmt. Eine Beispiel Eingabdedatei mit dem Namen `driven_cavity_input.txt` ist auf der Praktikumswebsite zu finden.
- Initialisieren der Felder `U,V,P` mit den konstanten Werten `UI,VI` und `PI` als Anfangswerte.
- Berechnen der Schrittweite `delt` für den nächsten Zeitschritt nach (22). War der aus der Parameterdatei eingelesene Wert für `tau` negativ, so soll anstatt der berechneten Zeitschrittweite direkt der aus der Parameterdatei eingelesene Wert für `delt` verwendet werden.
- Setzen der Randwerte für die Felder `U` und `V` gemäß der Formeln (17) und (18).
- Berechnen von `F` und `G` gemäß (12) und (13). Dabei müssen am Rand die Formeln (20) angewendet werden.
- Berechnung der rechten Seite der Druckgleichung (14).
- SOR–Iteration für die Druck–(Poisson–)Gleichung gemäß (15). In jeder Iteration müssen zunächst die Randwerte für `P` gemäß (19) gesetzt werden. Die Iteration wird abgebrochen, wenn die Norm des Residuums `res` die Toleranzgrenze `eps` unterschreitet oder die maximale Iterationsanzahl `itermax` erreicht ist. Nach dem Abbruch soll die Anzahl der benötigten Iterationen zurückgegeben werden und in `res` die aktuelle Größe der Norm des Residuums stehen.
- Berechnen der neuen Geschwindigkeiten gemäß (10) und (11).
- Ausgabe der berechneten Daten `U,V,P` in eine Datei. Die Ausgabe soll nicht nach jedem Zeitschritt erfolgen, sondern nur in den Zeitabständen `del_vec`, wobei `del_vec` aus der Eingabedatei eingelesen wird.[5]
  Bei jeder Ausgabe werden nacheinander die aktuellen Werte der Felder `U,V` und `P` jeweils zeilenweise geschrieben. Die Werte sollen jeweils im Mittelpunkt der Zellen liegen. Dazu müssen die Werte von `U` und `V` nach der Formel $(a_1 + a_2)/2$ noch aus geeigneten Kantenwerten gemittelt werden.
  Die Form der Ausgabedatei ist trivial. Man schreibt einfach die Werte folgender Variablen untereinander in eine Datei:

```
xlength
ylength
imax
jmax
Feld U(x,y)
Feld V(x,y)
Feld P(x,y)
```

b) Als erstes Beispiel soll ein typisches Modellproblem der numerischen Strömungsmechanik, das sogenannte "Driven–Cavity–Problem" (zu deutsch "Nischenströmung") simuliert werden. Es handelt sich dabei um einen mit Flüssigkeit gefüllten Topf, über den ein Band mit konstanter Geschwindigkeit gezogen wird. Dabei werden an allen 4 Rändern Haftbedingungen verwendet. Lediglich am oberen Rand wird die Geschwindigkeit in x–Richtung u nicht auf 0 sondern auf 1 gesetzt, um das gezogene Band zu

[5] Da der jeweilige Zeitschritt `delt` aus einer Zeitschrittweitensteuerung ermittelt wird, können die Werte für `del_vec` nicht a priori so bestimmt werden, dass die jeweils zugehörige „Aktionszeit" in der Zeitschleife genau erreicht wird. In der Regel ist jedoch `delt` klein gegen `del_vec`, so dass die jeweilige Aktion beim ersten überschreiten der zugehörigen Aktionszeit ausgeführt werden kann.

simulieren. Im Programm wird dies realisiert, indem die Randwerte am oberen Rand auf

$$u_{i,jmax+1} = 2.0 - u_{i,jmax}, \quad i = 1, \ldots, imax$$

gesetzt werden.

Als weitere Parameter sollen verwendet werden:

```
imax = 50     jmax = 50     xlength = 10   ylength = 10
delt = 0.02   t_end = 2.0   tau = 0.5      del_vec = 2.0
eps = 0.001   omg = 1.7     alpha = 0.5    itermax = 100
GX = 0.0      GY = 0.0      Re = 10
UI = 0.0      VI = 0.0      PI = 0.0
```

c) Implementiere obige Funktionalität in Python. Der Aufruf des Programms soll wie folgt aussehen:

```
python3 driven_cavity.py --input <parametersfile> --output <outputfile>
```

wobei <`parametersfile`> der Dateipfad zu einer Eingabedatei und <`outputfile`> der Basis-Dateiname der Ausgabedateien sein soll. Die Ausgabedateien sollen einen Suffix erhalten, welcher sie durchnummeriert. Beispielsweise würde ein Aufruf

```
python3 driven_cavity.py --input param.in --output dc_output
```

dateien mit den den Namen `dc_output_001`, `dc_output_002`, `dc_output_002`, ... erzeugen.
Zum Parsen der Kommandozeilen Parameter kann das Packet `argparse` verwendet werden.

d) Implementiere ein Python programm welches die generierten Ausgabedateien einließt und mittels `matplotlib` das Vektorfeld der Geschwindigkeiten aus `U` und `V` als auch das Skalarfeld `P` visualisiert. Der Aufruf soll wie folgt aussehen

```
python3 visualize.py --input <inputfile>
```

wobei <`inputfile`> der Pfad zu einer der während der Simulation generierten Ausgabedateien ist.

e) Implementiere die gesamte obige Funktionalität in C++. Der Aufruf des Programms soll wie folgt aussehen:

```
driven_cavity.exe <parametersfile> <outputfile>
```

Achte auf eine Vernünftige Struktur deines Programms in Klassen und Dateien. Verwende alle bisher gelernten Paradigmen, wo dies sinnvoll ist. Das Kompilieren des Programms soll mittels einer CMake Datei gesteuert werden.