

# **Analyse der Druckberechnung mithilfe einer Zustandsgleichung im Vergleich zur Lösung eines Gleichungssystems in SPH- Flüssigkeitssimulationen**

Pascal Hunkler

May 2022

# Inhaltsverzeichnis

<b>1</b>	<b>Abstract</b>	<b>4</b>
<b>2</b>	<b>Einleitung</b>	<b>5</b>
<b>3</b>	<b>Grundlagen</b>	<b>6</b>
3.1	Navier-Stokes-Gleichung und Flüssigkeitssimulationen . . . . .	6
3.1.1	Partikelbasierte Simulation . . . . .	6
3.1.2	Gitterbasierte Simulation . . . . .	7
3.2	SPH . . . . .	7
3.2.1	Diskretisierung mit SPH . . . . .	7
3.2.2	SPH in partikelbasierten Simulationen . . . . .	9
3.3	Zeitintegration . . . . .	11
3.4	Grundlegender Aufbau einer partikelbasierten Flüssigkeitssimulation . . .	12
<b>4</b>	<b>Druckberechnung</b>	<b>13</b>
4.1	Druckberechnung mit einer Zustandsgleichung . . . . .	13
4.2	Druckberechnung mit IISPH . . . . .	14
4.2.1	Herleitung des Gleichungssystems . . . . .	14
4.2.2	Lösung des Gleichungssystems . . . . .	15
<b>5</b>	<b>Implementierung</b>	<b>18</b>
5.1	Programmierungsumgebung . . . . .	18
5.2	Grenzbehandlung . . . . .	18
5.3	Kernelfunktion, Kernelgradient . . . . .	19
5.4	Nachbarschaftssuche . . . . .	20
5.4.1	Uniformes Gitter . . . . .	20
5.4.2	Uniformes Gitter Aufbau . . . . .	21
5.4.3	Bestimmung der Nachbarn mithilfe des Uniformen Gitters . . . . .	21
5.5	Berechnung der Dichte . . . . .	22
5.6	Berechnung des Drucks . . . . .	22
5.7	Berechnung der Druckbeschleunigung . . . . .	26
5.8	Berechnung der restlichen Beschleunigungen . . . . .	26
5.9	Simulationsschritt . . . . .	27
5.10	Visualisierung . . . . .	29

<b>6</b>	<b>Analyse</b>	<b>30</b>
6.1	Szenarien . . . . .	30
6.2	Einfluss des Zeitschritts . . . . .	34
6.3	Entwicklung der Durchschnittsdichte . . . . .	36
6.4	Rechenaufwand . . . . .	39
<b>7</b>	<b>Fazit und Ausblick</b>	<b>40</b>

# **1 Abstract**

## **2 Einleitung**

## 3 Grundlagen

### 3.1 Navier-Stokes-Gleichung und Flüssigkeitssimulationen

Dieser Abschnitt behandelt die Navier-Stokes-Gleichung, welche die Grundlage dafür bildet, Flüssigkeitssimulationen zu realisieren. Sie beschreibt die Änderungsrate der Geschwindigkeit eines kleinen volumetrischen Flüssigkeitselements. Für die Navier-Stokes-Gleichung gibt es verschiedene Formen, die auf den eulerschen Ansatz oder den lagrangeschen Ansatz beruhen. Diese Arbeit beschäftigt sich im Wesentlichen auf lagrangesche, partikelbasierte Simulationen, der Vollständigkeit wegen wird jedoch auch noch kurz auf eulersche, gitterbasierte Ansätze eingegangen. Die Inhalte dieses Abschnitts beziehen sich auf Ihmsen et al. [IOS<sup>+</sup>14].

#### 3.1.1 Partikelbasierte Simulation

In einer lagrangeschen, partikelbasierten Flüssigkeitssimulation wird die Flüssigkeit in so genannte Partikel unterteilt, die sich mit der Flüssigkeit mitbewegen. Jedes Partikel nimmt ein Teilvolumen der Flüssigkeit ein und besitzt eine Masse. In einem partikelbasierten Ansatz kann die Navier-Stokes-Gleichung wie folgt beschrieben werden:

$$\frac{d\mathbf{v}_i}{dt} = -\frac{1}{\rho_i}\nabla p_i + \nu\nabla^2\mathbf{v}_i + \frac{\mathbf{F}_i^{\text{other}}}{m_i} \quad (3.1)$$

Sie gibt die Änderungsrate der Geschwindigkeit  $\frac{d\mathbf{v}_i}{dt}$  eines beweglichen Partikels mit Positionen  $\mathbf{x}_i$  nach Zeit an, welche durch verschiedene Beschleunigungen beeinflusst wird: Die Druckbeschleunigung  $-\frac{1}{\rho_i}\nabla p_i$ , die Viskositätsbeschleunigung  $\nu\nabla^2\mathbf{v}_i$ , die durch die Viskosität  $\nu$  beeinflusst wird, und der Einfluss anderer Kräfte  $\frac{\mathbf{F}_i^{\text{other}}}{m_i}$ , wie beispielsweise die Gravitation. Ziel einer partikelbasierten Simulation ist es, die verschiedenen Beschleunigungen zu berechnen, um die Geschwindigkeiten und Positionen der Partikel stets anzupassen. Eine Methode, um die Beschleunigungen an den Partikeln zu berechnen, ist Smoothed Particle Hydrodynamics (SPH), welches im nächsten Abschnitt vorgestellt wird. In partikelbasierten Simulationen muss regelmäßig die Nachbarschaft eines Partikels bestimmt werden, da diese frei beweglich sind und sich somit die Nachbarschaft ändert. Die Nachbarschaft eines Partikels wird benötigt, um die Beschleunigungen an einem Partikel mit SPH zu bestimmen. Im Gegensatz zu gitterbasierten Simulationen,

welche im nächsten Unterabschnitt vorgestellt werden, muss hier jedoch ein Term aus der Navier-Stokes-Gleichung weniger berechnet werden.

### 3.1.2 Gitterbasierte Simulation

In eulerschen, gitterbasierten Simulationen wird im Gegensatz zu partikelbasierten Simulationen mit Gittern statt Partikeln gearbeitet. Hier wird die Flüssigkeit nicht in frei bewegliche Partikel unterteilt, sondern in Teilvolumina, die fest an einem Ort sind und meist ein Gitter bilden. Die Navier-Stokes-Gleichung hat durch diesen anderen Ansatz folgende Form:

$$\frac{\partial \mathbf{v}_i}{\partial t} = -\frac{1}{\rho_i} \nabla p_i + \nu \nabla^2 \mathbf{v}_i + \frac{\mathbf{F}_i^{\text{other}}}{m_i} - \mathbf{v}_i \cdot \nabla \mathbf{v}_i \quad (3.2)$$

Betrachtet wird hier die Änderungsrate der Geschwindigkeit  $\frac{\partial \mathbf{v}_i}{\partial t}$  an einer festen Position  $\mathbf{x}_i$ . Dadurch, dass die Positionen fest sind, kommt in der Navier-Stokes-Gleichung neben den Beschleunigungen noch der Term  $-\mathbf{v}_i \cdot \nabla \mathbf{v}_i$  vor, welcher die Bewegung der Flüssigkeit ausgleicht. Im Gegensatz zu partikelbasierten Simulationen muss hier jedoch direkt erst einmal keine Nachbarschaftssuche stattfinden, da die Punkte nicht frei beweglich sind. Generell kann jedoch die Performance und Genauigkeit zwischen partikelbasierten und gitterbasierten nicht allgemein verglichen werden.

## 3.2 SPH

Das Konzept Smoothed Particle Hydrodynamics (SPH), ursprünglich formuliert von Lucy [Luc77] und unabhängig davon von Monaghan und Gingold [GM77], entstand ursprünglich aus dem Bereich der Astrophysik, wird aber heute auch in diversen anderen Bereichen, unter anderem auch der Computergrafik angewandt. Mithilfe von SPH kann durch Diskretisierung und Berechnung von Größen wie der Druckbeschleunigung oder der Viskositätsbeschleunigung die Navier-Stokes-Gleichung gelöst werden. Es eignet sich daher gut für partikelbasierte Simulationen.

### 3.2.1 Diskretisierung mit SPH

Die Inhalte dieses Abschnittes basieren hauptsächlich auf den Arbeiten von Monaghan [Mon05], von Price [Pri12] und von Koshier et al. [KBST20]. Für eine beliebige skalare Variable  $A$  gilt die Identität

$$A(\mathbf{x}) = \int A(\mathbf{x}') \delta(\mathbf{x} - \mathbf{x}') d\mathbf{x}' \quad (3.3)$$

$\delta$  ist hierbei die Dirac'sche Deltafunktion, die definiert ist als

$$\delta(\mathbf{x}) = \begin{cases} \infty, & \text{falls } \mathbf{x} = 0 \\ 0, & \text{sonst} \end{cases} \quad (3.4)$$

Die Dirac'sche Deltafunktion in Gleichung 3.3 kann mithilfe einer glättenden Kernelfunktion  $W$  mit endlicher Breite  $h$  approximiert werden.

$$A(\mathbf{x}) = \int A(\mathbf{x}') W(\mathbf{x} - \mathbf{x}', h) d\mathbf{x}' + O(h^2) \quad (3.5)$$

Damit die Approximation aus Gleichung 3.5 gültig ist, muss  $W$  folgende Eigenschaften besitzen:

$$\int_{\mathbb{R}^d} W(\mathbf{x}', h) d\mathbf{x}' = 1 \quad (\text{Normalisierung}) \quad (3.6)$$

$$\lim_{h \rightarrow 0} W(\mathbf{x}, h) = \delta(\mathbf{x}) \quad (\text{Dirac-}\delta) \quad (3.7)$$

Weitere wünschenswerte Eigenschaften der Kernelfunktion  $W$  sind:

$$W(\mathbf{x}, h) \geq 0 \quad (\text{Positivität}) \quad (3.8)$$

$$W(\mathbf{x}, h) = W(-\mathbf{x}, h) \quad (\text{Symmetrie}) \quad (3.9)$$

$$W(\mathbf{x}, h) = 0 \text{ für } \|\mathbf{x}\| \geq h \quad (\text{Kompakter Support}) \quad (3.10)$$

$\forall \mathbf{x} \in \mathbb{R}^d, h \in \mathbb{R}^+$ , während  $h$  der Kernelsupport ist. Eine beliebige Kernelfunktion ist der Cubic Spline Kernel [Mon92]. Er ist definiert als

$$W(\mathbf{x}, h) = \sigma_d \begin{cases} (2 - q)^3 - 4(1 - q)^3, & \text{für } 0 \leq q \leq 1 \\ (2 - q)^3, & \text{für } 1 \leq q \leq 2 \\ 0, & \text{sonst} \end{cases} \quad (3.11)$$

mit  $q = \frac{1}{h} \|\mathbf{x}\|$ . Der Kernelnormierungsfaktor  $\sigma_d$  ist abhängig von der Dimension  $d$  und beträgt für  $d = 1, 2, 3$   $\sigma_1 = \frac{1}{6h}, \sigma_2 = \frac{5}{14\pi h^2}, \sigma_3 = \frac{1}{4\pi h^3}$ . In der Literatur gibt es verschiedene Formulierungen für den Cubic Spline Kernel, die sich im Wesentlichen in der Parametrisierung unterscheiden. Der Vorteil dieser Kernelfunktion ist, dass die Eigenschaften zu Positivität, Symmetrie, und kompakten Support, erfüllt werden. Zudem erzielt Cubic Spline trotz seiner Einfachheit gute Ergebnisse.

Um die Interpolation aus Gleichung 3.5 bei einer Flüssigkeit zu diskretisieren, wird die Flüssigkeit in mehrere Partikel unterteilt. Jedes Partikel  $f$  besitzt eine Masse  $m_f$ , Dichte  $\rho_f$  und Position  $\mathbf{x}_f$ . Der Wert von  $A$  an einem Partikel  $f$  wird notiert als  $A_f$ , die Approximation davon als  $\langle A_f \rangle$ . Der Integral aus Gleichung 3.5 kann nun durch eine Summe, und die Masse  $\rho dV$  durch die Partikelmasse  $m_f$  ersetzt werden.

$$A(\mathbf{x}) = \int \frac{A(\mathbf{x}')}{\rho(\mathbf{x}')} W(\mathbf{x} - \mathbf{x}', h) \rho(\mathbf{x}') d\mathbf{x}' + O(h^2) \quad (3.12)$$

$$\approx \sum_{f_f} m_{f_f} \frac{A_{f_f}}{\rho_{f_f}} W(\mathbf{x} - \mathbf{x}_{f_f}, h) =: \langle A_f \rangle \quad (3.13)$$



$f_f$  sind hierbei alle Partikel der Flüssigkeit.  $W(\mathbf{x}_f - \mathbf{x}_{f_f}, h)$  wird ab nun durch  $W_{ff_f}$  abgekürzt. Da beispielsweise der Cubic Spline Kernel einen Kernsupport von  $2h$  besitzt, muss hier nur über Partikel  $f_f$  summiert werden, die in direkter Nachbarschaft sind, da die Kernfunktion für Partikel, die weiter als  $2h$  von dem Partikel  $f$  entfernt sind, null ist. Daher ist die Eigenschaft, dass der Support der Kernfunktion kompakt ist, wünschenswert.

Häufig ist es auch nötig, die Differentialoperatoren zu diskretisieren. Der Gradient einer Variable  $A$  kann durch folgende Approximation bestimmt werden:

$$\nabla A_f \approx \sum_{f_f} A_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \nabla W_{ff_f} \quad (3.14)$$

Diese Approximation hat eine Genauigkeit 0-ter Ordnung, wenn folgende Bedingung erfüllt wird:

$$\sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \nabla W_{ff_f} = \mathbf{0} \quad (3.15)$$

Wird zusätzlich die Bedingung

$$\sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} (\mathbf{x}_{f_f} - \mathbf{x}_f) \otimes \nabla W_{ff_f} = \mathbb{K} \quad (3.16)$$

erfüllt, hat sie eine Genauigkeit erster Ordnung. Ist  $\mathbf{A}$  höherdimensional, so können komplexere Differentialoperatoren wie folgt diskretisiert werden:

$$\nabla \mathbf{A}_f \approx \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \mathbf{A}_{f_f} \otimes \nabla W_{ff_f} \quad (3.17)$$

$$\nabla \cdot \mathbf{A}_f \approx \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \mathbf{A}_{f_f} \cdot \nabla W_{ff_f} \quad (3.18)$$

$$\nabla \times \mathbf{A}_f \approx - \sum_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \mathbf{A}_{f_f} \times \nabla W_{ff_f} \quad (3.19)$$

wobei  $a \otimes b = ab^T$  das dyadische Produkt ist. Leider führen diese „direkten“ Ableitungen zu einer schlechten Approximationsqualität und instabilen Simulationen. Daher gibt es mittlerweile viele alternative Formulierungen für diese Differentialoperatoren. Zwei davon werden im nächsten Unterabschnitt gezeigt.

### 3.2.2 SPH in partikelbasierten Simulationen

Dieser Unterabschnitt basiert auf die Arbeit von Koshier et al. [KBST20]. Ziel einer partikelbasierten Simulation ist es, die Beschleunigungen zu bestimmen, die in der Navier-

Stokes-Gleichung angegeben sind. Die Navier-Stokes-Gleichung für einen partikelbasierten Ansatz hat folgende Form:

$$\frac{d\mathbf{v}_f}{dt} = -\frac{1}{\rho_f} \nabla p_f + \nu \nabla^2 \mathbf{v}_f + \frac{\mathbf{F}_f^{\text{other}}}{m_f} \quad (3.20)$$

Es muss also in erster Linie die Druckbeschleunigung  $-\frac{1}{\rho_f} \nabla p_f$  und die Viskositätsbeschleunigung  $\nu \nabla^2 \mathbf{v}_f$  bestimmt werden. Die SPH Diskretisierung einer Variable  $A$  sieht wie folgt aus:

$$A_f \approx \sum_{f_f} m_{f_f} \frac{A_{f_f}}{\rho_{f_f}} W_{ff_f} \quad (3.21)$$

Hieraus ist ersichtlich, dass zur Berechnung von  $A$  die Dichte  $\rho_f$  aller Partikel benötigt wird. Diese kann jedoch leicht ermittelt werden, indem in Gleichung 3.21  $A$  durch  $\rho$  ersetzt wird. Der Faktor  $\frac{\rho_{f_f}}{\rho_{f_f}}$  kürzt sich weg und man erhält:

$$\rho_f \approx \sum_{f_f} m_{f_f} \frac{\rho_{f_f}}{\rho_{f_f}} W_{ff_f} = \sum_{f_f} m_{f_f} W_{ff_f} \quad (3.22)$$

Für die Berechnung der Druckbeschleunigung  $-\frac{1}{\rho_f} \nabla p_f$  wird die Diskretisierung des Druckgradienten benötigt. Wie der Druck berechnet werden kann, wird im nächsten Kapitel vorgestellt. Ist der Druck berechnet, ist es naheliegend, die normale SPH Diskretisierung für den Gradienten zu verwenden:

$$-\frac{1}{\rho_f} \nabla p_f \approx -\frac{1}{\rho_f} \sum_{f_f} A_{f_f} \frac{m_{f_f}}{\rho_{f_f}} \nabla W_{ff_f} \quad (3.23)$$

Das Problem bei dieser Näherung ist jedoch, dass die daraus resultierende Beschleunigung nicht den Impuls und Drehimpuls der Flüssigkeit erhält, was zu einer instabilen Simulation führt. Für die Druckberechnung wird häufig stattdessen eine alternative Approximation verwendet, die auf folgender Diskretisierung des Gradienten einer Variable  $A$  beruht:

$$\nabla A_f \approx \rho_f \sum_{f_f} m_{f_f} \left( \frac{A_f}{\rho_f^2} + \frac{A_{f_f}}{\rho_{f_f}^2} \right) \nabla W_{ff_f} \quad (3.24)$$

Der Vorteil an dieser Approximation ist, dass das eben genannte Problem, dass daraus resultierende Kräfte oder Beschleunigungen den Impuls und Drehimpuls der Flüssigkeit nicht erhalten, behoben wird. Dies ist essentiell für robuste, stabile Simulationen. Der Nachteil, der in Kauf genommen wird, ist, dass nun konstante oder lineare Gradienten nicht mehr exakt berechnet werden können. Berechnet man die Druckbeschleunigung anhand dieser Diskretisierung erhält man:

$$-\frac{1}{\rho_f} \nabla p_f \approx -\sum_{f_f} m_{f_f} \left( \frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W_{ff_f} \quad (3.25)$$

Nun muss noch die Viskositätsbeschleunigung  $\nu \nabla^2 \mathbf{v}_f$  berechnet werden. Die normale SPH Diskretisierung für den Laplace Operator ist bestimmt durch

$$\nabla^2 \mathbf{v}_f = \sum_{ff} \frac{m_{ff}}{\rho_{ff}} \mathbf{v}_{ff} \nabla^2 W_{ff} \quad (3.26)$$

Es hat sich jedoch gezeigt, dass die Verwendung der zweiten Ableitung der Kernelfunktion sensibel ist gegenüber Störungen der Partikelanordnung und zu schlechteren Ergebnissen führt als eine alternative Formulierung, die ohne die zweite Ableitung auskommt:

$$\nabla^2 \mathbf{v}_f = 2(d+2) \sum_{ff} \frac{m_{ff}}{\rho_{ff}} \frac{\mathbf{v}_{ff} \cdot \mathbf{x}_{ff}}{\|\mathbf{x}_{ff}\| + 0.01h^2} \nabla W_{ff} \quad (3.27)$$

$\mathbf{x}_{ff} = \mathbf{x}_f - \mathbf{x}_{ff}$ ,  $\mathbf{x}_{ff} = \mathbf{x}_f - \mathbf{x}_{ff}$  und  $d$  ist die Anzahl der räumlichen Dimensionen. Schließlich erhält man zur Berechnung der Viskositätsbeschleunigung [KBST20]:

$$\nu \nabla^2 \mathbf{v}_f = 2(d+2) \nu \sum_{ff} \frac{m_{ff}}{\rho_{ff}} \frac{\mathbf{v}_{ff} \cdot \mathbf{x}_{ff}}{\|\mathbf{x}_{ff}\| + 0.01h^2} \nabla W_{ff} \quad (3.28)$$

Diese Formulierung erfüllt einige wünschenswerte Eigenschaften: Sie ist galilei-invariant, was bedeutet, dass beispielsweise in einer sich gleichmäßig bewegenden Flüssigkeit die Viskositätsbeschleunigung nicht sich anders verhält als in einer äquivalenten ruhenden Flüssigkeit. Es entsteht unter anderem auch keine Reibung, wenn alle Partikel gleichmäßig rotieren. Zudem wird der Impuls und Drehimpuls erhalten.

### 3.3 Zeitintegration

Sind in einer partikelbasierten Simulation die Beschleunigungen für den Druck, die Viskosität und die restlichen Beschleunigungen, wie z.B. die Gravitation berechnet, muss eine numerische Zeitintegration durchgeführt werden, um die Geschwindigkeiten und Positionen der Partikel zu aktualisieren. Häufig wird dazu das Euler-Cromer Verfahren genutzt. [KBST20]

Die Position  $\mathbf{x}_f^{t+\Delta t}$  und Geschwindigkeit  $\mathbf{v}_f^{t+\Delta t}$  eines Partikels  $f$  zum Zeitpunkt  $t + \Delta t$  wird hier aus der berechneten Beschleunigung  $\mathbf{a}_f^t$  zum Zeitpunkt  $t$  wie folgt aktualisiert:

$$\mathbf{v}_f^{t+\Delta t} = \mathbf{v}_f^t + \Delta t \mathbf{a}_f^t \quad (3.29)$$

$$\mathbf{x}_f^{t+\Delta t} = \mathbf{x}_f^t + \Delta t \mathbf{v}_f^{t+\Delta t} \quad (3.30)$$

$\Delta t$  ist hierbei die vergangene Zeit zwischen zwei Simulationsschritten, und wird auch als Zeitschritt bezeichnet.

Die Wahl des Zeitschritts  $\Delta t$  ist entscheidend für die Performanz und Stabilität der Simulation. Generell ist es gewollt, möglichst wenige Simulationsschritte berechnen zu müssen,

damit die Simulation insgesamt schneller berechnet wird. Daher wird ein möglichst großer Zeitschritt  $\Delta t$  gewählt. Ist der Zeitschritt jedoch zu groß gewählt, wird die numerische Integration ungenauer, was zu instabilen Simulationen führt, die zusammenbrechen. Um einen möglichst großen Zeitschritt zu wählen, der noch zu einer stabilen Simulation führt, kann sich an der CFL-Bedingung orientiert werden. Die CFL-Bedingung begrenzt den Zeitschritt  $\Delta t$  abhängig eines Parameters  $\lambda$ , der Partikelgröße  $\hbar$ , und der Geschwindigkeit des schnellsten Partikels  $\mathbf{v}^{max}$ :

$$\Delta t \leq \lambda \frac{\hbar}{\|\mathbf{v}^{max}\|} \quad (3.31)$$

Die Idee dahinter ist, dass für z.B.  $\lambda = 1$  sich die Partikel nicht weiter als eine Partikelgröße bewegen. [KBST20]

### 3.4 Grundlegender Aufbau einer partikelbasierten Flüssigkeitssimulation

Dieser Abschnitt beschreibt, wie sich die Konzepte aus den vorigen Abschnitten und der Druckberechnung, die im nächsten Kapitel erklärt wird, kombinieren lassen, um daraus eine einfache Flüssigkeitssimulation umzusetzen. Zuerst wird für jedes Partikel  $f$  die Dichte berechnet mithilfe folgender Gleichung:

$$\rho_f = \sum_{ff} m_{ff} W_{ff} \quad (3.32)$$

Dann wird die Viskositätsbeschleunigung berechnet:

$$\mathbf{a}_f^v = 2(d+2)\nu \sum_{ff} \frac{m_{ff}}{\rho_{ff}} \frac{\mathbf{v}_{ff} \cdot \mathbf{x}_{ff}}{\|\mathbf{x}_{ff}\| + 0.01h^2} \nabla W_{ff} \quad (3.33)$$

Es wird dann eine vorläufige Geschwindigkeit  $\mathbf{v}_f^*$  der Partikel bestimmt, die unter dem Einfluss der Viskositätsbeschleunigung und anderen Beschleunigungen wie z.B. der Gravitation aus der ursprünglichen Geschwindigkeit  $\mathbf{v}_f^t$  durch eine Zeitintegration berechnet wird:

$$\mathbf{v}_f^* = \mathbf{v}_f^t + \Delta t (\mathbf{a}_f^v + \mathbf{a}_f^{ext}) \quad (3.34)$$

Nun muss noch der Druck der Partikel  $p_f$  berechnet werden. Dies wird im nächsten Kapitel beschrieben. Wenn der Druck berechnet ist, wird die Druckbeschleunigung an jedem Partikel berechnet:

$$\mathbf{a}_f^p = - \sum_{ff} m_{ff} \left( \frac{p_f}{\rho_f^2} + \frac{p_{ff}}{\rho_{ff}^2} \right) \nabla W_{ff} \quad (3.35)$$

Anschließend werden die Geschwindigkeit und Position jedes Partikels aktualisiert:

$$\mathbf{v}_f^{t+\Delta t} = \mathbf{v}_f^* + \Delta t \mathbf{a}_f^p \quad (3.36)$$

$$\mathbf{x}_f^{t+\Delta t} = \mathbf{x}_f^t + \Delta t \mathbf{v}_f^{t+\Delta t} \quad (3.37)$$

## 4 Druckberechnung

Dieses Kapitel behandelt die Druckberechnung in partikelbasierten Flüssigkeitssimulationen. Die Berechnung des Drucks ist Voraussetzung für die Bestimmung der Druckbeschleunigung, welche Teil der Navier-Stokes-Gleichung ist. Es werden zwei Methoden vorgestellt, um den Druck zu bestimmen: Die Druckberechnung mithilfe einer Zustandsgleichung und das globale Berechnen des Drucks, welches von einer inkompressiblen Flüssigkeit ausgeht, mithilfe von Implicit Incompressible SPH (IISPH) [ICS<sup>+</sup>14], welches auf Incompressible SPH (ISPH) [SL03] aufbaut.

### 4.1 Druckberechnung mit einer Zustandsgleichung

Eine Methode, um den Druck zu berechnen, ist das Verwenden einer Zustandsgleichung. Hierbei wird der Druck direkt aus der Abweichung der berechneten Dichte  $\rho_f$  eines Partikels von der Ruhedichte  $\rho^0$  bestimmt. Ein Beispiel für eine solche Gleichung ist folgende:

$$p_f = \max \left( k \left( \frac{\rho_f}{\rho^0} - 1 \right), 0 \right) \quad (4.1)$$

Der Druck ist immer positiv, da es sonst zu Problemen kommt, wenn  $\rho_f < \rho^0$ , was vor allem an der Oberfläche der Flüssigkeit auftritt, da die Partikel an der Oberfläche weniger Nachbarn haben. Die Konstante  $k$  ist eine Steifigkeitskonstante. In [KBST20] ist beschrieben, dass die Wahl von  $k$  einen Einfluss auf die Dichteabweichung hat. Größere Werte für  $k$  führen zu einer geringeren Dichteabweichung, jedoch muss ein kleinerer Zeitschritt gewählt werden, da die Simulation sonst instabil wird. Kleinere Werte für  $k$  hingegen führen zu größeren Dichteabweichungen und damit auch zu weniger realistischen Simulationen. Anschaulich erklären lässt sich dies anhand eines Beispiels mit einer ruhenden Flüssigkeit, die unter dem Einfluss von Gravitation steht. Da die Flüssigkeit ruht, gilt

$$\mathbf{g} - \frac{1}{\rho_f} \nabla p_f = \mathbf{0} \quad (4.2)$$

Durch Diskretisierung des Druckgradienten erhält man

$$\mathbf{g} = \sum_{f_f} m_{f_f} \left( \frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W_{ff_f} \quad (4.3)$$

Verwendet man beispielsweise die Zustandsgleichung  $p_f = k(\rho_f - \rho^0)$  erhält man daraus

$$\mathbf{g} = k \sum_{ff} m_{ff} \left( \frac{\rho_f - \rho^0}{\rho_f^2} + \frac{\rho_{ff} - \rho^0}{\rho_{ff}^2} \right) \nabla W_{ff} \quad (4.4)$$

Die Gravitation ist konstant, also kann man aus dieser Gleichung schließen, dass sich insgesamt aus einer großen Steifigkeitskonstante  $k$  geringere Dichteabweichungen ergeben.

## 4.2 Druckberechnung mit IISPH

Dieser Abschnitt behandelt die Druckberechnung mit der IISPH-Methode. Implicit Incompressible SPH (IISPH) wurde erstmals von Ihmsen et al. entwickelt [ICS<sup>+</sup>14], worauf dieser Abschnitt auch basiert. Im Gegensatz zu Standard SPH (SSPH), also der Druckberechnung durch eine Zustandsgleichung, wird hier ein lineares Gleichungssystem gelöst, um den Druck der Partikel zu berechnen.

### 4.2.1 Herleitung des Gleichungssystems

Um das lineare Gleichungssystem herzuleiten, wird erst einmal die Kontinuitätsgleichung betrachtet:

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \mathbf{v} \quad (4.5)$$

Die Kontinuitätsgleichung stellt einen Zusammenhang her zwischen der Änderungsrate der Dichte und der Divergenz des Geschwindigkeitsfeldes. Diese Gleichung wird dann diskretisiert, indem die Änderungsrate der Dichte durch den Vorwärtsdifferenzquotienten  $\frac{\rho_f(t+\Delta t) - \rho_f(t)}{\Delta t}$  ersetzt und die Divergenz des Geschwindigkeitsfeldes durch das SPH-Konzept der Divergenz  $\nabla \cdot \mathbf{v}_f = -\frac{1}{\rho_f} \sum_{ff} m_{ff} \mathbf{v}_{ff} \nabla W_{ff}$ . Dadurch erhält man:

$$\frac{\rho_f(t + \Delta t) - \rho_f(t)}{\Delta t} = \sum_{ff} m_{ff} \mathbf{v}_{ff}(t + \Delta t) \nabla W_{ff}(t) \quad (4.6)$$

$\mathbf{v}_{ff}(t + \Delta t)$  ist hierbei  $\mathbf{v}_f(t + \Delta t) - \mathbf{v}_{ff}(t + \Delta t)$ . Diese Geschwindigkeiten basieren auf Druckbeschleunigungen, welche linear in den Werten für den Druck sind, da

$$\mathbf{a}_f^p = - \sum_{ff} m_{ff} \left( \frac{p_f}{\rho_f^2} + \frac{p_{ff}}{\rho_{ff}^2} \right) \nabla W_{ff} \quad (4.7)$$

Verwendet man das Euler-Cromer Verfahren, so kann die Geschwindigkeit eines Partikels zum Zeitpunkt  $t + \Delta t$  wie folgt geschrieben werden:

$$\mathbf{v}_f(t + \Delta t) = \mathbf{v}_f(t) + \Delta t \left( \mathbf{a}_f^{nonp} + \mathbf{a}_f^p \right) \quad (4.8)$$

$\mathbf{a}_f^{nonp}$  ist hierbei die Beschleunigung, die durch die Viskosität und andere Kräfte wie die Gravitation, entsteht. Wir betrachten ebenfalls die geschätzte Zwischen-Geschwindigkeit  $\mathbf{v}_f^*$ :

$$\mathbf{v}_f^* = \mathbf{v}_f(t) + \Delta t \mathbf{a}_f^{nonp} \quad (4.9)$$

Durch diese Zwischen-Geschwindigkeit entsteht auch eine Zwischen-Dichte

$$\rho_f^* = \rho_f(t) + \Delta t \sum_{ff} m_{ff} \mathbf{v}_{ff}^* \nabla W_{ff}(t) \quad (4.10)$$

Zudem gilt:

$$\mathbf{v}_f(t + \Delta t) = \mathbf{v}_f^* + \Delta t \mathbf{a}_f^p \quad (4.11)$$

Setzt man (4.11) in (4.6) ein, kann man Umformen:

$$\frac{\rho_f(t + \Delta t) - \rho_f(t)}{\Delta t} = \sum_{ff} m_{ff} \mathbf{v}_{ff}(t + \Delta t) \nabla W_{ff}(t) \quad (4.12)$$

$$= \sum_{ff} m_{ff} \mathbf{v}_{ff}^* \nabla W_{ff}(t) + \Delta t \sum_{ff} m_{ff} \mathbf{a}_{ff}^p \nabla W_{ff}(t) \quad (4.13)$$

$$= \frac{\rho_f^* - \rho_f(t)}{\Delta t} + \Delta t \sum_{ff} m_{ff} \mathbf{a}_{ff}^p \nabla W_{ff}(t) \quad (4.14)$$

Da wir erreichen wollen, dass die Dichte  $\rho_f(t + \Delta t)$  die Ruhedichte ist, damit die Partikel in einem unkomprimierten Zustand sind, wird  $\rho_f(t + \Delta t) = \rho^0$  angenommen. Mit Umformen erhält man dann:

$$\Delta t^2 \sum_{ff} m_{ff} \left( \mathbf{a}_f^p - \mathbf{a}_{ff}^p \right) \nabla W_{ff}(t) = \rho^0 - \rho_f^* \quad (4.15)$$

Setzt man nun (4.7) in (4.15) ein, ist der Druck linear und man erhält ein lineares Gleichungssystem

$$\mathbf{A}(t) \mathbf{p}(t) = \mathbf{b}(t) \quad (4.16)$$

Für jedes Partikel haben wir eine Gleichung der Form

$$\sum_{ff} A_{ff} p_{ff} = b_f = \rho_0 - \rho_f^* \quad (4.17)$$

### 4.2.2 Lösung des Gleichungssystems

Das Gleichungssystem, das gelöst werden muss, um den Druck zu berechnen, kann mit verschiedenen Methoden gelöst werden. Empfohlen von Ihmsen et al. [ICS<sup>+</sup>14] ist es, das relaxierte Jacobi-Verfahren anzuwenden, die Herangehensweise dazu ist in [ICS<sup>+</sup>14]

ebenfalls beschrieben, worauf dieser Unterabschnitt auch basiert. Hierzu werden die Werte für den Druck  $p_f$  wie folgt iterativ gelöst:

$$p_f^{l+1} = (1 - \omega)p_f^l + \omega \frac{\rho^0 - \rho_f^* - \sum_{f_f \neq f} A_{ff_f} p_{f_f}^l}{A_{ff}} \quad (4.18)$$

$l$  ist hierbei der Iterationsindex und  $\omega$  der Relaxierungsfaktor. Um dies zu berechnen, werden die Diagonalwerte  $A_{ff}$  der Matrix  $\mathbf{A}$  benötigt und  $\sum_{f_f \neq f} A_{ff_f} p_{f_f}^l$ . Diese können jedoch effektiv berechnet werden. Um die Koeffizienten zu extrahieren, betrachten wir den Term  $\Delta t^2 \mathbf{a}_f^p$ :

$$\Delta t^2 \mathbf{a}_f^p = -\Delta t^2 \sum_{f_f} m_{f_f} \left( \frac{p_f}{\rho_f^2} + \frac{p_{f_f}}{\rho_{f_f}^2} \right) \nabla W_{ff_f} \quad (4.19)$$

$$= \underbrace{\left( -\Delta t^2 \sum_{f_f} \frac{m_{f_f}}{\rho_f^2} \nabla W_{ff_f} \right)}_{\mathbf{d}_{ff}} p_f + \sum_{f_f} \underbrace{-\Delta t^2 \frac{m_{f_f}}{\rho_{f_f}^2} \nabla W_{ff_f}}_{\mathbf{d}_{ff_f}} p_{f_f} \quad (4.20)$$

Dies kann nun in (4.15) eingesetzt werden, daraus erhält man:

$$\rho^0 - \rho_f^* = \sum_{f_f} m_{f_f} \left( \mathbf{d}_{ff} p_f + \sum_{f_f} \mathbf{d}_{ff_f} p_{f_f} - \mathbf{d}_{ff_f f} p_{f_f} - \sum_{f_{ff}} \mathbf{d}_{ff_{ff}} p_{ff_{ff}} \right) \nabla W_{ff_f} \quad (4.21)$$

Beachtet werden muss, dass in  $\sum_{f_{ff}} \mathbf{d}_{ff_{ff}} p_{ff_{ff}}$  auch Druckwerte  $p_f$  vorkommen, da  $f$  und  $f_f$  Nachbarn sind. Um diese Werte zu separieren, können wir auch schreiben

$$\sum_{f_{ff}} \mathbf{d}_{ff_{ff}} p_{ff_{ff}} = \sum_{f_{ff} \neq f} \mathbf{d}_{ff_{ff}} p_{ff_{ff}} + \mathbf{d}_{ff f} p_f \quad (4.22)$$

Nun können wir die Teile mit Werten  $p_f$  von den Teilen mit Werten  $p_{f_f}$  und  $p_{ff_{ff}}$  in (4.21) trennen. Daraus erhält man:

$$\rho^0 - \rho_f^* = p_f \sum_{f_f} m_{f_f} (\mathbf{d}_{ff} - \mathbf{d}_{ff_f}) \nabla W_{ff_f} + \quad (4.23)$$

$$\sum_{f_f} m_{f_f} \left( \sum_{f_f} \mathbf{d}_{ff_f} p_{f_f} - \mathbf{d}_{ff_f f} p_{f_f} - \sum_{f_{ff} \neq f} \mathbf{d}_{ff_{ff}} p_{ff_{ff}} \right) \nabla W_{ff_f} \quad (4.24)$$

Jetzt können die Koeffizienten  $A_{ff}$  einfach berechnet werden durch

$$A_{ff} = \sum_{f_f} m_{f_f} (\mathbf{d}_{ff} - \mathbf{d}_{ff_f}) \nabla W_{ff_f} \quad (4.25)$$



Der iterative Schritt in (4.18) kann nun berechnet werden durch

$$p_f^{l+1} = (1 - \omega)p_f^l + \omega \frac{1}{A_{ff}} \left( \rho^0 - \rho_f^* - \right. \quad (4.26)$$

$$\left. \sum_{f_f} m_{f_f} \left( \sum_{f_f} \mathbf{d}_{ff_f} p_{f_f} - \mathbf{d}_{ff_f f_f} p_{f_f} - \sum_{f_{ff} \neq f} \mathbf{d}_{ff f_{ff}} p_{f_{ff}} \right) \nabla W_{ff_f} \right) \quad (4.27)$$

## 5 Implementierung

Dieses Kapitel beschäftigt sich damit, wie die Konzepte aus den vorherigen Kapiteln implementiert wurden.

### 5.1 Programmierungsumgebung

Die Software zur Flüssigkeitssimulation wurde überwiegend in der Programmiersprache C++ geschrieben. Daneben sind die Shader für die Visualisierung in OpenGL Shading Language geschrieben. Als Entwicklungsumgebung wurde Microsoft Visual Studio Community 2022 verwendet. Verwendete Bibliotheken sind GLFW und GLAD für die Visualisierung mit OpenGL, GoogleTest für UnitTests und OpenGL Mathematics (GLM) für mathematische Strukturen und Berechnungen.

### 5.2 Grenzbehandlung

Neben den Flüssigkeitspartikeln gibt es in der Implementierung auch Grenzpartikel. In der Notation werden die Grenzpartikel mit dem Index  $b$  und somit auch die Nachbarn eines Flüssigkeitspartikels  $f$ , welche zur Grenze gehören, mit dem Index  $f_b$  identifiziert. Für die Grenzbehandlung wird der Ansatz aus [KBST20] verwendet, wo es mehrere Schichten von Grenzpartikel mit identischer Größe und Masse gibt.

Die Grenzpartikel sind fest an einem Ort und verhindern, dass die Flüssigkeit auseinanderfließt. Die Grenze besteht aus drei Schichten von Grenzpartikel. Die mehreren Schichten sorgen dafür, dass wenn ein Partikel nahe an der Grenze ist, noch stets genügend Nachbarn hat.

Für die Grenzpartikel werden Größen wie die Dichte  $\rho_b$  und der Druck  $p_b$  nicht berechnet. Da diese jedoch für SPH-Berechnungen benötigt werden, müssen dennoch für diese Größen Werte gewählt werden. Für die Dichte  $\rho_b$  wird die Ruhedichte  $\rho^0$  gewählt. Wird der Druck  $p_{f_b}$  in einer SPH-Berechnung benötigt, wird stattdessen der Wert des Flüssigkeitspartikels gespiegelt und damit der Wert  $p_f$  verwendet.

### 5.3 Kernelfunktion, Kernelgradient

In der Implementierung wird der Cubic Spline Kernel verwendet. Da die Simulation zweidimensional ist, hat die Kernelfunktion damit folgende Form:

$$W(\mathbf{x}_f - \mathbf{x}_{f_f}, h) = \frac{5}{14\pi h^2} \begin{cases} (2-q)^3 - 4(1-q)^3, & \text{für } 0 \leq q \leq 1 \\ (2-q)^3, & \text{für } 1 \leq q \leq 2 \\ 0, & \text{sonst} \end{cases} \quad (5.1)$$

mit  $q = \frac{1}{h} \|\mathbf{x}_f - \mathbf{x}_{f_f}\|$  und  $h$  die Partikelgröße.

Diese Funktion lässt sich recht einfach mit Algorithmus 1 implementieren.

---

**Algorithm 1** Cubic Spline Kernelfunktion

---

```

1: function W( $\mathbf{x}_f - \mathbf{x}_{f_f}$ )
2:    $q \leftarrow \frac{\|\mathbf{x}_f - \mathbf{x}_{f_f}\|}{h}$ 
3:    $t_1 \leftarrow \max(1 - q, 0)$ 
4:    $t_2 \leftarrow \max(2 - q, 0)$ 
5:    $\sigma \leftarrow \frac{5}{14\pi h^2}$ 
6:   return  $\sigma \cdot (t_2^3 - 4t_1^3)$ 
7: end function

```

---

Nun, da die Kernelfunktion in Form des Cubic Spline Kernels implementiert ist, muss noch der Kernelgradient implementiert werden. Der Kernelgradient des Cubic Spline Kernels hat folgende Form:

$$\nabla W(\mathbf{x}_f - \mathbf{x}_{f_f}, h) = \frac{5}{14\pi h^2} \cdot \frac{\mathbf{x}_f - \mathbf{x}_{f_f}}{\|\mathbf{x}_f - \mathbf{x}_{f_f}\| h} \begin{cases} -3(2-q)^2 + 12(1-q)^2, & \text{für } 0 \leq q \leq 1 \\ -3(2-q)^2, & \text{für } 1 \leq q \leq 2 \\ 0, & \text{sonst} \end{cases} \quad (5.2)$$

ebenfalls mit  $q = \frac{1}{h} \|\mathbf{x}_f - \mathbf{x}_{f_f}\|$  und  $h$  die Partikelgröße.

Für den Kernelgradient wird Algorithmus 2 angewandt.

---

**Algorithm 2** Cubic Spline Kernelgradient

---

```
1: function  $\nabla W(\mathbf{x}_f - \mathbf{x}_{f_f})$ 
2:    $q \leftarrow \frac{\|\mathbf{x}_f - \mathbf{x}_{f_f}\|}{h}$ 
3:   if  $q == 0$  then
4:     return  $(0 \ 0)^\top$ 
5:   end if
6:    $t_1 \leftarrow \max(1 - q, 0)$ 
7:    $t_2 \leftarrow \max(2 - q, 0)$ 
8:    $\sigma \leftarrow \frac{5}{14\pi h^2}$ 
9:   return  $\sigma \frac{\mathbf{x}_f - \mathbf{x}_{f_f}}{\|\mathbf{x}_f - \mathbf{x}_{f_f}\| h} \cdot (-3t_2^2 + 12t_1^2)$ 
10: end function
```

---

## 5.4 Nachbarschaftssuche

Da für die SPH-Berechnungen die Nachbarn eines Partikels benötigt werden, muss eine Nachbarschaftssuche durchgeführt werden. Diese muss in jedem Simulationsschritt erfolgen, da sich die Nachbarn eines Partikels in jedem Simulationsschritt ändern können. Daher ist es sehr wichtig, dass die Nachbarschaftssuche effizient abläuft und nicht zu viel Zeit dafür aufgewandt wird.

Der naive Ansatz ist es, wenn man für jedes Partikel die Distanz zu den restlichen Partikeln misst und überprüft, ob die Distanz geringer ist als der Kernelsupport. Pro Simulationsschritt gibt es damit einen Zeitaufwand in  $O(n^2)$  für  $n$  Partikel in der Simulation. Dies ist nicht gerade optimal, denn schon bei etwas größeren Simulationen wird sehr viel Zeit für die Nachbarschaftssuche angewandt.

### 5.4.1 Uniformes Gitter

Um das Problem zu lösen, dass die Nachbarschaftssuche in quadratischer Zeit nach der Partikelanzahl abläuft, kann stattdessen ein Uniformes Gitter aufgebaut werden, was zur Folge hat, dass der Zeitaufwand für die Nachbarschaftssuche nun in  $O(n)$  ist. Das Uniforme Gitter unterteilt den Raum, in dem sich die Partikel bewegen, in mehrere gleich große quadratische Zellen, deren Größe dem Kernelsupport entspricht. Sinn dahinter ist, dass als Nachbarn eines Partikels nur die Partikel in der gleichen Zelle oder einer Nachbarzelle in Frage kommen. Da die Anzahl der Nachbarzellen und die Anzahl der Partikel in einer Zelle konstant ist, kommen für jedes Partikel nur konstant viele Nachbarpartikel in Frage. Daher ist die Nachbarschaftssuche durch das Uniforme Gitter in  $O(n)$  für jeden Simulationsschritt.

In der Implementierung wird ein Uniformes Gitter verwendet, das dem kompakten Gitter

entspricht, was in der Arbeit von Lagae und Dutré [LD08] beschrieben wird.

### 5.4.2 Uniformes Gitter Aufbau

Kern der Implementierung des Uniformen Gitters sind zwei Arrays  $C$  und  $L$ . Der Array  $L$  enthält die Partikelindizes, welche sortiert sind nach den Zellen, in denen diese sich befinden. Der Array  $C$  wird zum Aufbau von  $L$  benötigt und enthält für jeden Zellenindex den Offset, an dem in  $L$  ein Partikelindex eingefügt wird. Nach dem Aufbau von  $L$  gibt  $C$  jeweils anhand eines Zellenindex den Index für das erste Partikel in dieser Zelle in  $L$  an. Beispielsweise sind alle Partikel von  $L[C[j]]$  bis  $L[C[j + 1] - 1]$  in der Zelle mit Zellenindex  $j$ . Der Zellenindex eines Partikels wird anhand dessen Position bestimmt.

---

**Algorithm 3** Aufbau des Uniformen Gitters

---

```

1: for  $j \leftarrow 0$ , cells do
2:    $C[j] \leftarrow 0$ 
3: end for
4: for all particle  $i$  do
5:    $j \leftarrow \text{GETCELLINDEX}(\mathbf{x}_i)$ 
6:    $C[j] \leftarrow C[j] + 1$ 
7: end for
8: for  $i \leftarrow 1$ , cells do
9:    $C[j] \leftarrow C[j] + C[j - 1]$ 
10: end for
11: for all particle  $i$  do
12:    $j \leftarrow \text{GETCELLINDEX}(\mathbf{x}_i)$ 
13:    $C[j] \leftarrow C[j] - 1$ 
14:    $L[C[j]] \leftarrow i$ 
15: end for

```

---

### 5.4.3 Bestimmung der Nachbarn mithilfe des Uniformen Gitters

Ist das Uniforme Gitter aufgebaut, indem die Arrays  $L$  und  $C$  initialisiert wurden, so kann dieses für die Nachbarschaftssuche genutzt werden. Um die Nachbarn eines Partikels zu ermitteln, werden zuerst die Zelle des Partikels und dessen Nachbarzellen (bis zu 8 Nachbarzellen, je nachdem ob die Zelle am Rand ist oder nicht) überprüft. Alle Partikel in diesen Zellen kommen als Nachbar in Frage.

In einem zweiten Schritt wird dann überprüft, ob der Abstand zwischen diesen Partikeln und des Ursprungspartikels geringer als der Kernelsupport  $\hbar$  ist. Die Partikel, deren Abstand zum Ursprungspartikel geringer als der Kernelsupport ist, sind die Nachbarn des Ursprungspartikels.

Algorithmus 4 zeigt, wie die Ermittlung der Nachbarn eines Partikels  $f$  implementiert werden kann.

---

**Algorithm 4** Nachbarschaftssuche mithilfe des Uniformen Gitters

---

```

1: possibleNeighbors  $\leftarrow$  empty list
2: neighbors  $\leftarrow$  empty list
3: for all neighbor cell index  $j$  do
4:   for  $i \leftarrow C[j], C[j + 1] - 1$  do
5:     possibleNeighbors.ADD( $L[i]$ )
6:   end for
7: end for
8: for all  $i$  in possibleNeighbors do
9:   if  $\|\mathbf{x}_i - \mathbf{x}_f\| < h$  then
10:    neighbors.ADD( $i$ )
11:   end if
12: end for

```

---

## 5.5 Berechnung der Dichte

Zur Implementierung der Dichteberechnung wird die normale SPH Approximation der Dichte benutzt:

$$\rho_f = \sum_{f_i} m_{f_i} W_{ff_i} \quad (5.3)$$

In der Implementierung ist die Masse aller Partikel identisch, daher kann die Masse der Nachbarpartikel  $m_{f_i}$  durch  $m_f$  ersetzt werden und stattdessen die Dichte wie folgt berechnet werden:

$$\rho_f = m_f \sum_{f_i} W_{ff_i} \quad (5.4)$$

---

**Algorithm 5** Dichteberechnung

---

```

for all fluid particle  $f$  do
   $\rho_f \leftarrow m_f \sum_{f_i} W_{ff_i}$ 
end for

```

---

## 5.6 Berechnung des Drucks

Beim Start der Simulation kann ausgewählt werden, ob der Druck durch eine Zustandsgleichung oder durch IISPH berechnet wird. Daher gibt es für die Druckberechnung zwei verschiedene Implementierungen.

Für die Druckberechnung durch eine Zustandsgleichung wird folgende Zustandsgleichung verwendet:

$$p_f = \max \left( k \left( \frac{\rho_f}{\rho^0} - 1 \right), 0 \right) \quad (5.5)$$

Die Implementierung durch einen Algorithmus ist trivial.

Alternativ kann der Druck durch IISPH [ICS<sup>+</sup>14] berechnet werden. Dazu muss ein Gleichungssystem der Form  $\mathbf{A}\mathbf{p} = \mathbf{s}$  gelöst werden. Bevor das Gleichungssystem gelöst wird, werden die Diagonalwerte  $A_{ff}$  der Matrix  $A$  und die Komponenten  $s_f$  des Lösungsvektors  $\mathbf{s}$  berechnet. Im Kapitel „Druckberechnung“ wurde zur Bestimmung der Diagonalwerte folgende Gleichung aus [ICS<sup>+</sup>14] verwendet, die aus Notationsgründen modifiziert ist:

$$A_{ff} = \sum_{f_f} m_{f_f} (\mathbf{d}_{ff} - \mathbf{d}_{f_ff}) \nabla W_{ff_f} \quad (5.6)$$

$\mathbf{d}_{ff}$  ist definiert als:

$$\mathbf{d}_{ff} = -\Delta t^2 \sum_{f_f} \frac{m_f}{\rho_f^2} \nabla W_{ff_f} \quad (5.7)$$

$\mathbf{d}_{f_ff}$  ist definiert als:

$$\mathbf{d}_{f_ff} = -\Delta t^2 \frac{m_f}{\rho_f^2} \nabla W_{f_ff} \quad (5.8)$$

Da in der Implementierung die Massen der Partikel identisch sind, vereinfacht sich  $A_{ff}$  daher zu:

$$A_{ff} = \sum_{f_f} m_{f_f} (\mathbf{d}_{ff} - \mathbf{d}_{f_ff}) \nabla W_{ff_f} \quad (5.9)$$

$$= \sum_{f_f} m_f (\mathbf{d}_{ff} - \mathbf{d}_{f_ff}) \nabla W_{ff_f} \quad (5.10)$$

$$= \sum_{f_f} m_f \left( \left( -\Delta t^2 \sum_{f_f} \frac{m_f}{\rho_f^2} \nabla W_{ff_f} \right) - \Delta t^2 \frac{m_f}{\rho_f^2} \nabla W_{ff_f} \right) \nabla W_{ff_f} \quad (5.11)$$

$$= -\Delta t^2 \frac{m_f^2}{\rho_f^2} \sum_{f_f} \left( \left( \sum_{f_f} \nabla W_{ff_f} \right) + \nabla W_{ff_f} \right) \nabla W_{ff_f} \quad (5.12)$$

Bezieht man die Partikel der Grenze mit ein, erhält man:

$$A_{ff} = -\Delta t^2 \frac{m_f^2}{\rho_f^2} \left( \sum_{f_f} \left( \left( \sum_{f_f} \nabla W_{ff_f} \right) + \nabla W_{ff_f} \right) \nabla W_{ff_f} + \right. \quad (5.13)$$

$$\left. \sum_{f_b} \left( \left( \sum_{f_b} \nabla W_{ff_b} \right) + \nabla W_{ff_b} \right) \nabla W_{ff_b} \right) \quad (5.14)$$

$s_f$  hat folgende Form:

$$s_f = \rho_f^0 - \rho_f^* \quad (5.15)$$

Die Zwischen-Dichte  $\rho_f^*$  wird aus den Zwischen-Geschwindigkeiten wie folgt berechnet unter Berücksichtigung der Grenzpartikel und den identischen Massen:

$$\rho_f^* = \rho_f + m_f \Delta t \left( \sum_{f_f} (\mathbf{v}_f^* - \mathbf{v}_{f_f}^*) \nabla W_{ff_f} + \sum_{f_b} \mathbf{v}_f^* \nabla W_{ff_b} \right) \quad (5.16)$$

Die Komponenten des Lösungsvektors werden also durch folgende Gleichung implementiert:

$$s_f = \rho_f^0 - \rho_f - m_f \Delta t \left( \sum_{f_f} (\mathbf{v}_f^* - \mathbf{v}_{f_f}^*) \nabla W_{ff_f} + \sum_{f_b} \mathbf{v}_f^* \nabla W_{ff_b} \right) \quad (5.17)$$

Ebenfalls wird der Druck  $\mathbf{p}^0$  auf  $\mathbf{0}$  initialisiert.

Nun kann iterativ der Druck berechnet werden. Der Index  $l$  bezeichnet die Anzahl der Iterationen. Dazu wird zuerst jeweils die Druckbeschleunigung  $\mathbf{a}_f^p$  durch Algorithmus 7 berechnet. Danach wird  $(\mathbf{A}\mathbf{p})_f^l$  berechnet, was definiert ist durch:

$$(\mathbf{A}\mathbf{p})_f^l = \Delta t^2 \sum_{f_f} m_{f_f} (\mathbf{a}_f^p - \mathbf{a}_{f_f}^p) \nabla W_{ff_f} \quad (5.18)$$

Unter Einbeziehung der Grenzpartikel und den identischen Massen, wird  $(\mathbf{A}\mathbf{p})_f^l$  wie folgt berechnet:

$$(\mathbf{A}\mathbf{p})_f^l = m_f \Delta t^2 \left( \sum_{f_f} (\mathbf{a}_f^p - \mathbf{a}_{f_f}^p) \nabla W_{ff_f} + \sum_{f_b} \mathbf{a}_f^p \nabla W_{ff_b} \right) \quad (5.19)$$

Jetzt kann schließlich der Druck aktualisiert werden durch:

$$p_f^{l+1} = \max(p_f^l + \omega \frac{s_f - (\mathbf{A}\mathbf{p})_f^l}{\mathbf{A}_{ff}}, 0) \quad (5.20)$$

Der Relaxierungsfaktor  $\omega$  ist in der Implementierung auf den Wert 0.5 festgelegt, da dieser Wert nahezu optimale Ergebnisse liefern soll. [ICS<sup>+</sup>14]

Es wird ebenfalls der durchschnittliche Dichtefehler  $\frac{1}{n_f} \sum_f \frac{(\mathbf{A}\mathbf{p})_f - s_f}{\rho_f^0}$  berechnet. Ist dieser Fehler unter dem Wert von 0.001, so wird der letzte berechnete Wert des Druckes genommen, andernfalls wird weiter iteriert und ein neuer Druck berechnet. Ihmsen et al. [ICS<sup>+</sup>14] empfiehlt, als Konvergenzkriterium für den durchschnittlichen Dichtefehler Werte von 0.001 oder geringer zu verwenden.

Algorithmus 6 zeigt die Implementierung der Druckberechnung mit IISPH.



---

**Algorithm 6** Berechnung des Drucks der Partikel mit IISPH

---

```
1: for all fluid particle  $f$  do
2:    $A_{ff} \leftarrow -\Delta t^2 \frac{m_f^2}{\rho_f^2} \cdot \left( \sum_{ff} \nabla W_{ff} \nabla W_{ff} + \sum_{fb} \nabla W_{fb} \nabla W_{fb} \right)$ 
3:    $s_f \leftarrow \rho_f^0 - \rho_f - m_f \Delta t \left( \sum_{ff} (\mathbf{v}_f^* - \mathbf{v}_{ff}^*) \nabla W_{ff} + \sum_{fb} \mathbf{v}_f^* \nabla W_{fb} \right)$ 
4:    $p_f \leftarrow 0$ 
5: end for
6:  $e \leftarrow \infty$ 
7: while  $e \geq 0.001$  do
8:    $e \leftarrow 0$ 
9:   Compute pressure accelerations  $\mathbf{a}_f^p$  using algorithm 7
10:  for all fluid particle  $f$  do
11:     $(\mathbf{A}p)_f \leftarrow m_f \Delta t^2 \left( \sum_{ff} (\mathbf{a}_f^p - \mathbf{a}_{ff}^p) \nabla W_{ff} + \sum_{fb} \mathbf{a}_f^p \nabla W_{fb} \right)$ 
12:    if  $f$  has no neighbors then
13:       $(\mathbf{A}p)_f \leftarrow 0$ 
14:    end if
15:     $p_f \leftarrow \max(p_f + \omega \frac{s_f - (\mathbf{A}p)_f}{\mathbf{A}_{ff}}, 0)$ 
16:     $e \leftarrow e + \frac{(\mathbf{A}p)_f - s_f}{\rho_f^0}$ 
17:  end for
18:   $e \leftarrow \frac{e}{n_f}$ 
19: end while
```

---

## 5.7 Berechnung der Druckbeschleunigung

Zur Berechnung der Druckbeschleunigung wird die alternative Form der SPH-Approximation verwendet, die den Impuls und Drehimpuls der Flüssigkeit erhält:

$$\mathbf{a}_f^p = -\frac{1}{\rho_f} \nabla p_f = -\sum_{ff} m_{ff} \left( \frac{p_f}{\rho_f^2} + \frac{p_{ff}}{\rho_{ff}^2} \right) \nabla W_{ff} \quad (5.21)$$

Die Implementierung dazu findet sich in Algorithmus 7.

---

### Algorithm 7 Berechnung der Druckbeschleunigungen

---

```

for all particle i do
  if particle i belongs to the boundary then
     $\mathbf{a}_i^p \leftarrow (0 \ 0)^\top$ 
    continue
  end if
   $\mathbf{acc}_p \leftarrow (0 \ 0)^\top$ 

  // Druckbeschleunigung an Partikel i
  for all neighbor j of particle i do
    if particle j belongs to the boundary then
       $\mathbf{acc}_p \leftarrow \mathbf{acc}_p - \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{(\rho_f^0)^2} \right) \cdot \nabla W_{ij}$ 
    else
       $\mathbf{acc}_p \leftarrow \mathbf{acc}_p - \left( \frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \cdot \nabla W_{ij}$ 
    end if
  end for
   $\mathbf{acc}_p \leftarrow \mathbf{acc}_p \cdot m_f$ 

   $\mathbf{a}_i^p \leftarrow \mathbf{acc}_p$ 
end for

```

---

## 5.8 Berechnung der restlichen Beschleunigungen

Neben den Druckbeschleunigungen müssen auch andere Beschleunigungen berechnet werden. In der Implementierung wird die Viskositätsbeschleunigung berechnet und die Gravitationsbeschleunigung, welche konstant ist. Die Viskositätsbeschleunigung  $\nu \nabla^2 \mathbf{v}_f$  wird ebenfalls nicht durch die normale SPH-Approximation berechnet, sondern durch

folgende alternative SPH-Approximation:

$$\nu \nabla^2 \mathbf{v}_f = 2(d+2)\nu \sum_{ff} \frac{m_{ff}}{\rho_{ff}} \frac{\mathbf{v}_{ff} \cdot \mathbf{x}_{ff}}{\|\mathbf{x}_{ff}\| + 0.01h^2} \nabla W_{ff} \quad (5.22)$$

Da der Parameter  $\nu$  frei gewählt ist, kann der Vorfaktor  $2(d+2)$  weggelassen werden. Ebenfalls sind in der Implementierung die Massen der Partikel identisch. Zusätzlich muss noch die Grenzbehandlung mit einbezogen werden. Daher wird in Algorithmus 8 die Viskositätsbeschleunigung durch folgende Gleichung berechnet:

$$\nu \nabla^2 \mathbf{v}_f = \nu m_f \left( \sum_{ff} \frac{1}{\rho_{ff}} \frac{\mathbf{v}_{ff} \cdot \mathbf{x}_{ff}}{\|\mathbf{x}_{ff}\| + 0.01h^2} \nabla W_{ff} \right) \quad (5.23)$$

$$+ \sum_{fb} \frac{1}{\rho_f} \frac{\mathbf{v}_f \cdot \mathbf{x}_{fb}}{\|\mathbf{x}_{fb}\| + 0.01h^2} \nabla W_{fb} \quad (5.24)$$

---

**Algorithm 8** Berechnung der restlichen Beschleunigungen

---

```

for all particle i do
  if particle i belongs to the boundary then
     $\mathbf{a}_i^n \leftarrow (0 \ 0)^\top$ 
    continue
  end if
   $\mathbf{acc}_g \leftarrow (0 \ -9.81)^\top$ 
   $\mathbf{acc}_v \leftarrow (0 \ 0)^\top$ 

  // Viskositätsbeschleunigung an Partikel i
  for all neighbor j of particle i do
    if particle j belongs to the boundary then
       $\mathbf{acc}_v \leftarrow \mathbf{acc}_v + \frac{1}{\rho_i} \frac{(\mathbf{v}_i - \mathbf{v}_j)(\mathbf{x}_i - \mathbf{x}_j)}{(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j) + 0.01h^2} \cdot \nabla W_{ij}$ 
    else
       $\mathbf{acc}_v \leftarrow \mathbf{acc}_v + \frac{1}{\rho_j} \frac{(\mathbf{v}_i - \mathbf{v}_j)(\mathbf{x}_i - \mathbf{x}_j)}{(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j) + 0.01h^2} \cdot \nabla W_{ij}$ 
    end if
  end for
   $\mathbf{acc}_v \leftarrow 2\nu m_f \cdot \mathbf{acc}_v$ 

   $\mathbf{a}_i^n \leftarrow \mathbf{acc}_g + \mathbf{acc}_v$ 
end for

```

---

## 5.9 Simulationsschritt

Der Simulationsschritt ist der Kern der Simulation. In jedem Simulationsschritt werden die Beschleunigungen berechnet, die auf die Partikel wirken, und anschließend anhand diesen Beschleunigungen die Geschwindigkeiten und Positionen der Partikel aktualisiert.

Zu Beginn des Simulationsschrittes findet die Nachbarschaftssuche statt, damit die SPH-Approximationen berechnet werden können. Die Nachbarschaftssuche geschieht mithilfe des Uniformen Gitters, welches zuvor beschrieben wurde.

Nachdem die Nachbarn bestimmt sind, wird mithilfe von Algorithmus 5 die Dichte an jedem Flüssigkeitspartikel berechnet.

Dann werden die restlichen Beschleunigungen mit Algorithmus 8 berechnet, die nicht durch den Druck ausgelöst sind.

Anhand diesen Beschleunigungen wird eine Zwischen-Geschwindigkeit  $\mathbf{v}_f^*$  an jedem Flüssigkeitspartikel  $f$  berechnet.

Nun wird der Druck berechnet. Je nachdem, welche Druckberechnung gewünscht ist, wird der Druck durch eine Zustandsgleichung oder durch IISPH mit Algorithmus 6 berechnet.

Anschließend werden die Druckbeschleunigungen berechnet mit Algorithmus 7.

Zum Schluss werden anhand des Euler-Cromer Verfahrens die Geschwindigkeiten  $\mathbf{v}_f$  und Positionen  $\mathbf{x}_f$  der Flüssigkeitspartikel aktualisiert.

Der Simulationsschritt ist in Algorithmus 9 dargestellt.

---

**Algorithm 9** Simulationsschritt

---

- 1: Determine neighbors of each particle
  - 2: Compute density  $\rho_f$  of each fluid particle using algorithm 5
  - 3: Compute non-pressure accelerations  $\mathbf{a}_f^n$  using algorithm 8
  - 4: **for all** fluid particle  $f$  **do**
  - 5:      $\mathbf{v}_f^* \leftarrow \mathbf{v}_f + \Delta t \mathbf{a}_f^n$
  - 6: **end for**
  - 7: Compute pressure  $p_f$  of each fluid particle using algorithm 6 or state equation
  - 8: Compute pressure accelerations  $\mathbf{a}_f^p$  using algorithm 7
  - 9: **for all** fluid particle  $f$  **do**
  - 10:      $\mathbf{v}_f \leftarrow \mathbf{v}_f^* + \Delta t \mathbf{a}_f^p$
  - 11: **end for**
  - 12: **for all** fluid particle  $f$  **do**
  - 13:      $\mathbf{x}_f \leftarrow \mathbf{x}_f + \Delta t \mathbf{v}_f$
  - 14: **end for**
  - 15:
-

## 5.10 Visualisierung

Visualisiert wird die Simulation durch die Programmierschnittstelle *OpenGL* in einem Fenster der Größe 1200x750.

Jedes Partikel wird durch einen Kreis dargestellt. Beim Starten der Simulation kann die Partikelgröße  $h$  gewählt werden. Die Partikelgröße entspricht gleichzeitig dem Durchmesser des zugehörigen Kreises in Pixel. Wird also beispielsweise eine Partikelgröße  $h = 10$  gewählt, so ist in der Visualisierung jeder Kreis auch vom Durchmesser 10 Pixel.

Die globalen Koordinaten  $\mathbf{x}_f$  eines Partikels  $f$ , die in der Simulation verwendet werden, sind zu dessen Fensterkoordinaten  $\mathbf{x}'_f$  gespiegelt an einer Parallelen der x-Achse, die sich in der Mitte des Fensters mit y-Koordinaten  $y = 375$  befindet.

Seien  $\mathbf{x}'_f$  also die Fensterkoordinaten des Partikels  $f$  mit globalen Koordinaten  $\mathbf{x}_f = (x \ y)^\top$ . Dann gilt

$$\mathbf{x}'_f = (x \ 750 - y)^\top \quad (5.25)$$

Die Grenzpartikel werden alle in grauer Farbe dargestellt. Die Farbe eines Flüssigkeitspartikels  $f$  hängt von dem Betrag dessen Geschwindigkeit  $\|\mathbf{v}_f\|$  ab. Dabei erscheinen ruhende Partikel komplett in blau, schnellere Partikel in grün und schließlich ab  $\|\mathbf{v}_f\| > \frac{h}{\Delta t}$  komplett in rot. Das heißt, aus der Visualisierung lässt sich erkennen, ob die CFL-Bedingung

$$\Delta t \leq \lambda \frac{\hbar}{\|\mathbf{v}^{max}\|} \quad (5.26)$$

für  $\lambda = 1$  erfüllt wird. Die Übergänge zwischen den Farben sind fließend.

Sei  $\mathbf{c}_f$  der Wert der Farbe des Partikels  $f$  im RGB-Farbraum. Dann wird  $\mathbf{c}_f$  wie folgt berechnet:

$$\mathbf{c}_f = \begin{cases} \begin{pmatrix} 0 \\ \sin^2\left(\frac{\pi\|\mathbf{v}_f\|}{\frac{h}{\Delta t}}\right) \\ \cos^2\left(\frac{\pi\|\mathbf{v}_f\|}{\frac{h}{\Delta t}}\right) \end{pmatrix} & \text{falls } \|\mathbf{v}_f\| < \frac{h}{2\Delta t} \\ \begin{pmatrix} \cos^2\left(\frac{\pi\|\mathbf{v}_f\|}{\frac{h}{\Delta t}}\right) \\ \sin^2\left(\frac{\pi\|\mathbf{v}_f\|}{\frac{h}{\Delta t}}\right) \\ 0 \end{pmatrix} & \text{falls } \frac{h}{2\Delta t} \leq \|\mathbf{v}_f\| < \frac{h}{\Delta t} \\ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} & \text{sonst} \end{cases} \quad (5.27)$$

# 6 Analyse

## 6.1 Szenarien

In der Simulationssoftware können fünf verschiedene Szenarien simuliert werden. Die verschiedenen Szenarien können beim Start der Software gewählt werden. Die Szenarien unterscheiden sich im Wesentlichen in den Anfangspositionen der Partikel. In diesem Abschnitt werden diese Szenarien vorgestellt.

Das erste Szenario ist eine klassische Dammbruchsimulation, die in vielerlei Anwendungen von Flüssigkeitssimulationen vorkommt. Der Aufbau ist einfach und die Eigenschaften der Flüssigkeit kann gut beobachtet werden, so dass sich dieses Szenario gut zum Testen einer Simulation eignet. Sie stellt eine Situation dar, in der ein Damm, der die Flüssigkeit aufhält, plötzlich verschwunden ist. Der Aufbau des Dammbruchszenarios ist in Abbildung 6.1 dargestellt.

Das zweite Szenario ist sehr ähnlich zum Dammbruchszenario. Hier ist der Damm noch vorhanden. Dieser besitzt jedoch ein Loch, wodurch die Flüssigkeit durchfließt. In diesem Szenario kann beobachtet werden, wie die Flüssigkeit durch das Loch durchströmt und sich die Pegelstände vor und hinter dem Damm angleichen. Der Aufbau dieses Szenarios ist in Abbildung 6.2 dargestellt.

Das dritte Szenario zeigt zu Beginn die Flüssigkeit auf einer Erhöhung. Hier kann beobachtet werden, wie die Flüssigkeit auf beiden Seiten herunterfällt und sich zwei voneinander getrennte Flüssigkeiten bilden. Der Aufbau dieses Szenarios ist in Abbildung 6.3 dargestellt.

Im vierten Szenario kann beobachtet werden, wie die Flüssigkeit von einer schiefen Ebene in Richtung einer horizontalen Ebene herunterfließt. Dargestellt ist der Aufbau dieses Szenarios in Abbildung 6.4.

Das fünfte Szenario zeigt eine ruhende Flüssigkeit. Beobachtet werden kann hier wie sich die Partikel anordnen. Dieses Szenario eignet sich auch gut, um den Verlauf der Durchschnittsdichte der Flüssigkeitspartikel zu beobachten. Dargestellt ist der Aufbau dieses Szenarios in Abbildung 6.5.

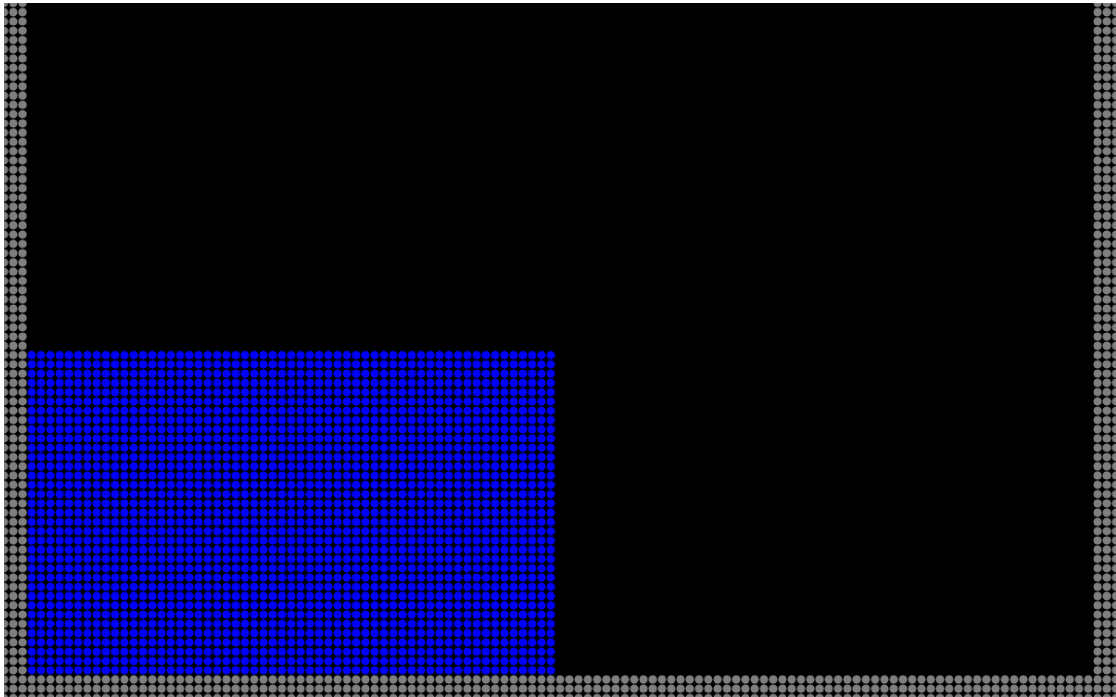


Abbildung 6.1: Brechender Damm

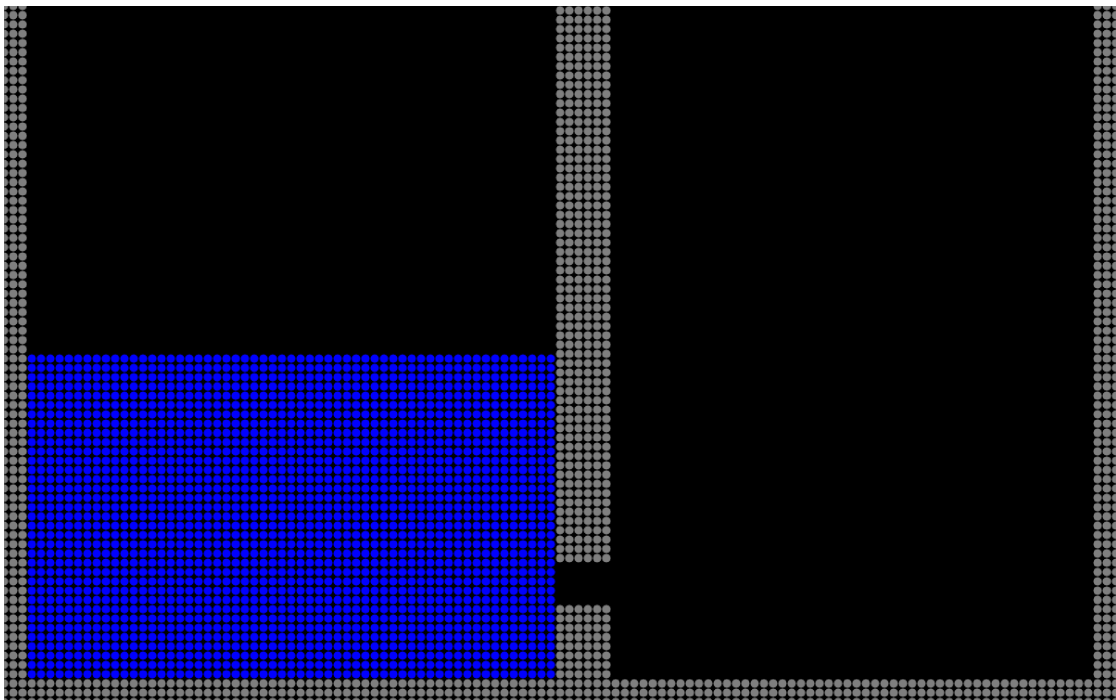


Abbildung 6.2: Damm mit Leck

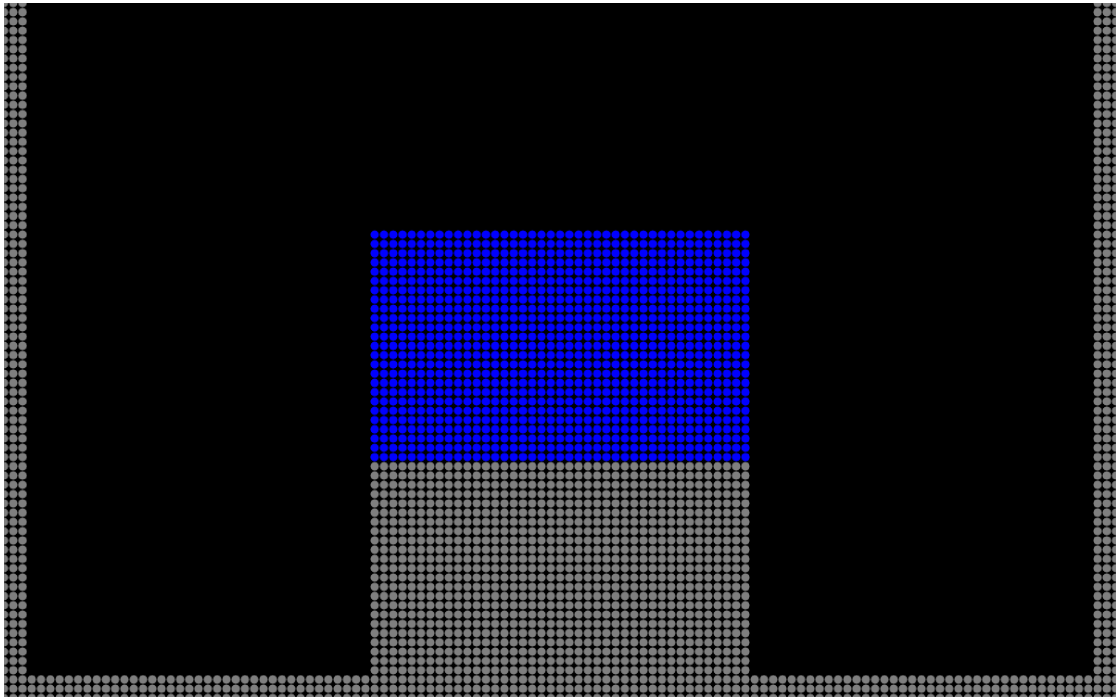


Abbildung 6.3: Von einer Plattform herunterfallende Flüssigkeit

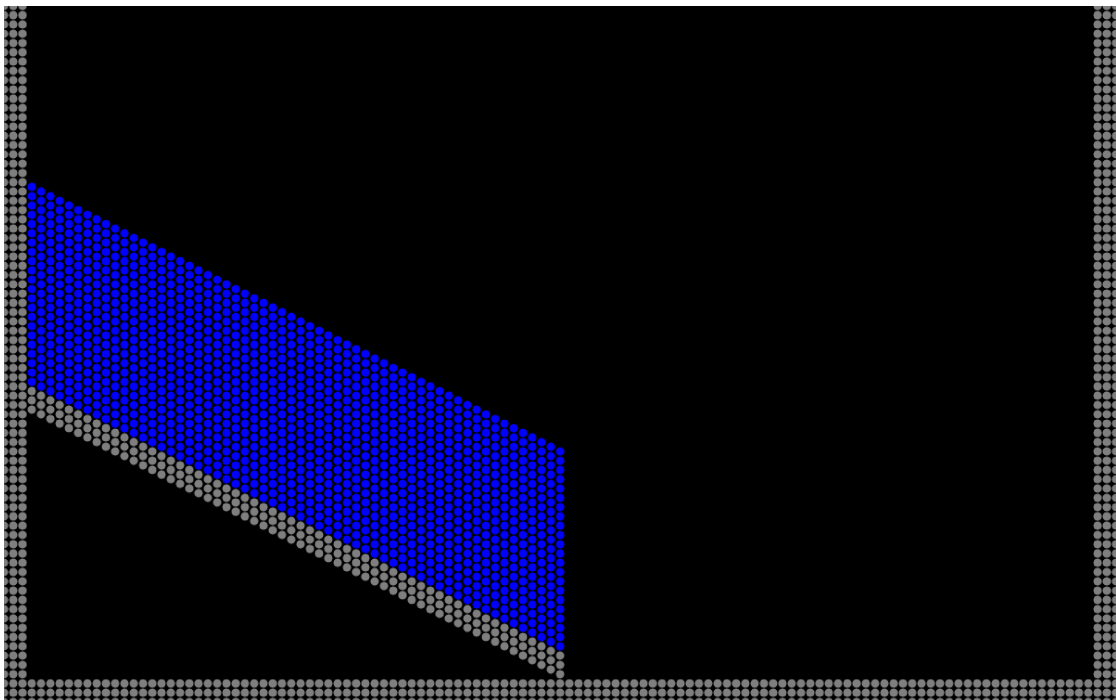


Abbildung 6.4: Von einer schiefen Ebene herunterfließende Flüssigkeit



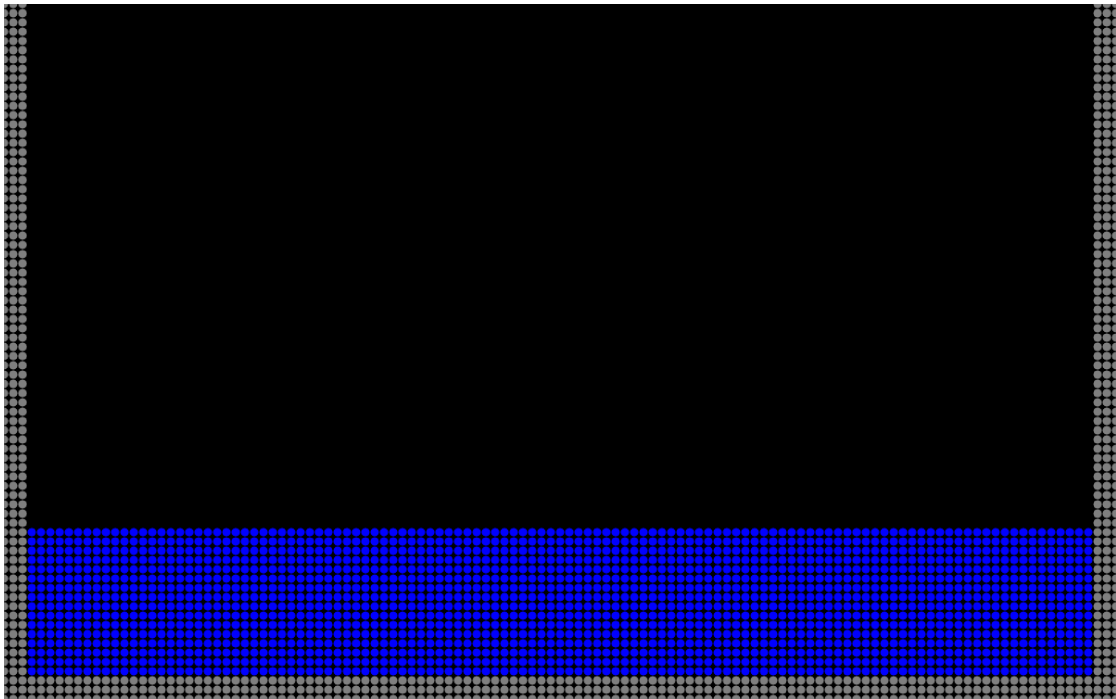


Abbildung 6.5: Ruhende Flüssigkeit

## 6.2 Einfluss des Zeitschritts

$\nu = 200$  SSPH:  $k = 10^5$

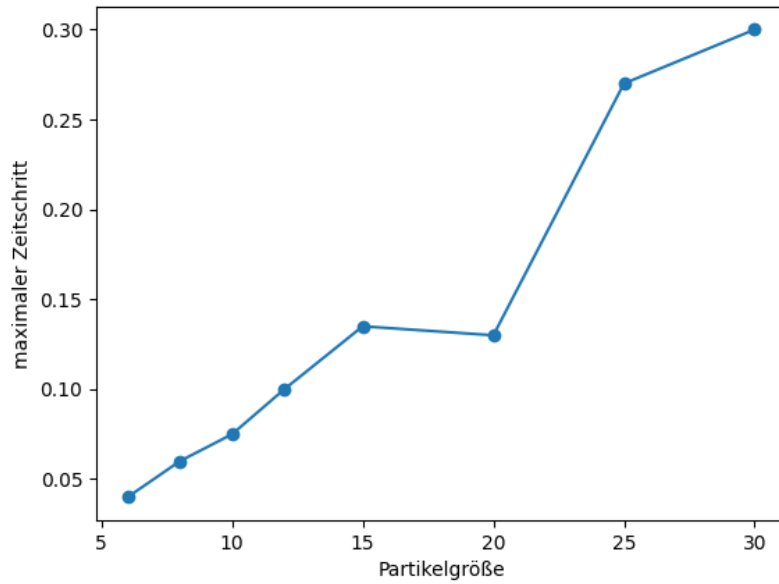


Abbildung 6.6: IISPH maximaler Zeitschritt nach Partikelgröße

$h = 10$   $\nu = 200$  IISPH:  $\Delta t \leq 0.075$

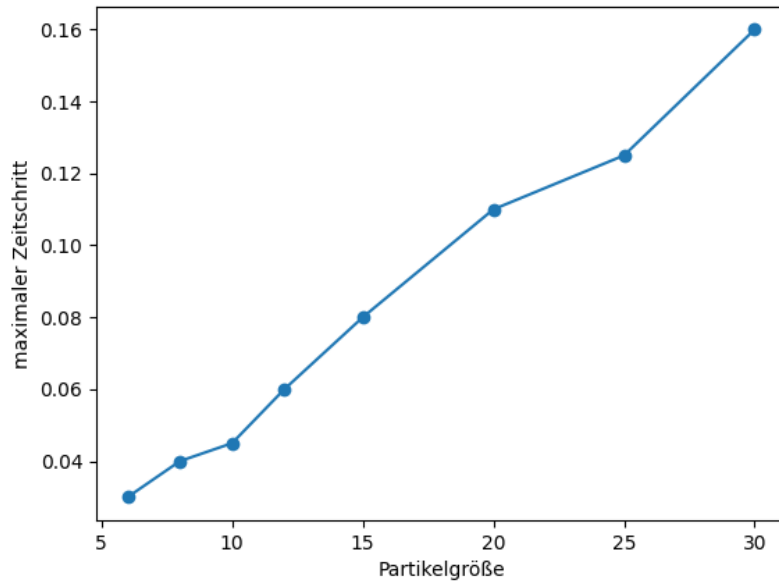


Abbildung 6.7: SSPH maximaler Zeitschritt nach Partikelgröße

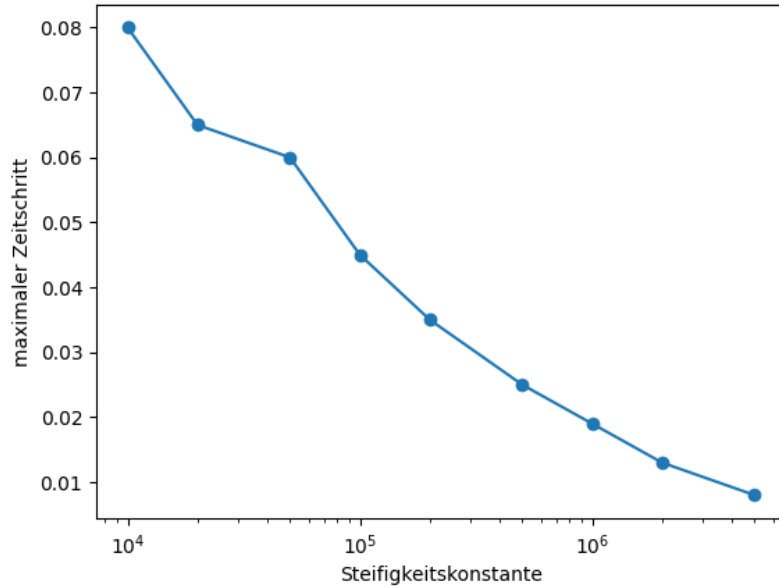


Abbildung 6.8: SSPH maximaler Zeitschritt nach Steifigkeitskonstante

## 6.3 Entwicklung der Durchschnittsdichte

Interessant ist es auch, sich die Entwicklung der Durchschnittsdichte der Flüssigkeitspartikel unterschiedlicher Simulationen anzuschauen. Wünschenswert sind geringe Oszillationen der Durchschnittsdichte und ein Wert nahe  $\rho^0$ .

Dazu wurden Daten aus drei SSPH-Simulationen und einer IISPH-Simulation des Szenarios, in der die Flüssigkeit ruht, gesammelt. Dieses Szenario wurde gewählt, weil sich hier die Partikel lediglich neu anordnen und keine großen Dichteschwankungen durch die Bewegung der Flüssigkeit zu erwarten sind. Die Simulationen wurden durchgeführt an Partikel mit Größe  $h = 10$  und Viskosität  $\nu = 200$ . Für die IISPH-Simulation wurde ein Zeitschritt  $\Delta t = 0.1$  gewählt. Damit wurden 1000 Simulationsschritte ausgewertet.

In den SSPH-Simulationen wurden drei verschiedene Steifigkeitskonstanten gewählt. Einmal ein niedriger Wert  $k = 10^4$ , ein mittlerer Wert  $k = 10^5$  und ein hoher Wert  $k = 10^6$ . Da die beiden höheren Werte für die Steifigkeitskonstante keinen Zeitschritt  $\Delta t = 0.1$  erlauben, wurden hier niedrigere Zeitschritte von  $\Delta t = 0.05$  und  $\Delta t = 0.01$  gewählt. Um jedoch die Entwicklung der Durchschnittsdichte mit der IISPH-Simulation zu vergleichen und die gleiche Zeitspanne zu beobachten, wurde hier eine größere Anzahl an Iterationen von 2000 Iterationen bzw. 10000 Iterationen festgelegt.

Das Diagramm in Abbildung 6.9 zeigt die Entwicklung der Durchschnitte in der IISPH-Simulation nach Iteration.

Die Diagramme aus den Abbildungen 6.10, 6.11, 6.12 zeigen die Entwicklung der Durchschnitte in den SSPH-Simulationen nach Iteration.

In den Diagrammen ist zu erkennen, dass sich bei SSPH mit zunehmender Steifigkeitskonstante die Oszillationen abschwächen. Mit einer Steifigkeitskonstante von  $k = 10^4$  dominieren die Oszillationen die Entwicklung der Durchschnittsdichte, während bei einer Steifigkeitskonstante von  $k = 10^6$  die Durchschnittsdichte deutlich weniger oszilliert. In allen Simulationen ist zu sehen, dass mit der Zeit die Amplituden der Oszillationen geringer werden. In IISPH sind die Oszillationen kaum zu erkennen. Dadurch, dass für die gleiche Zeitspanne zehn mal weniger Iterationen berechnet werden müssen und zudem die Oszillationen sehr schwach sind, ist IISPH hier im Vorteil.

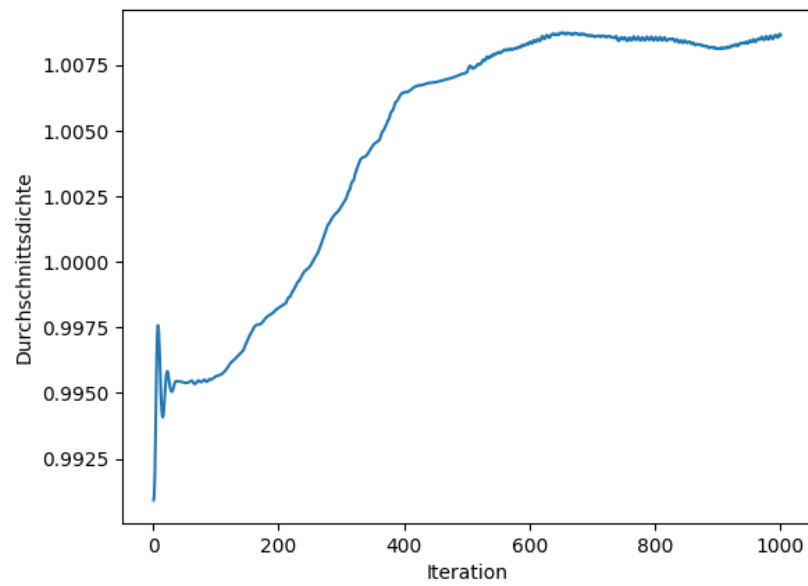


Abbildung 6.9: IISPH Durchschnittsdichte

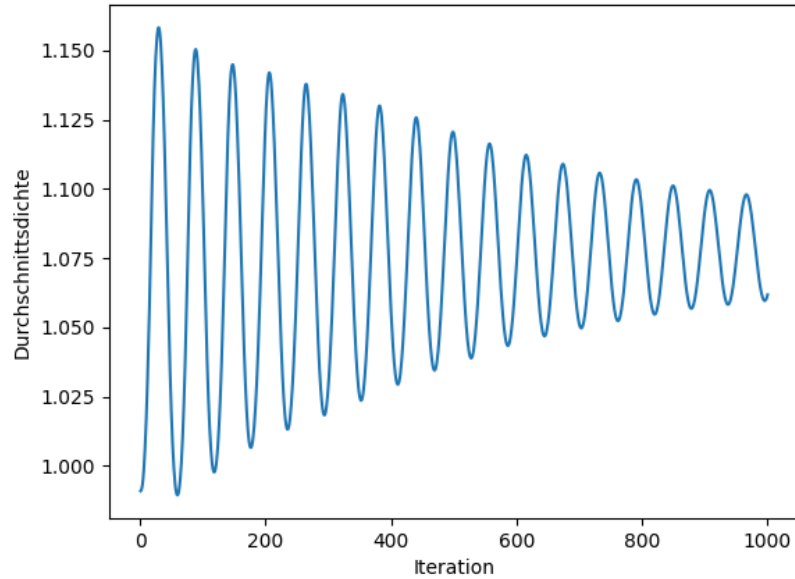


Abbildung 6.10: SSPH Durchschnittsdichte mit niedriger Steifigkeitskonstante

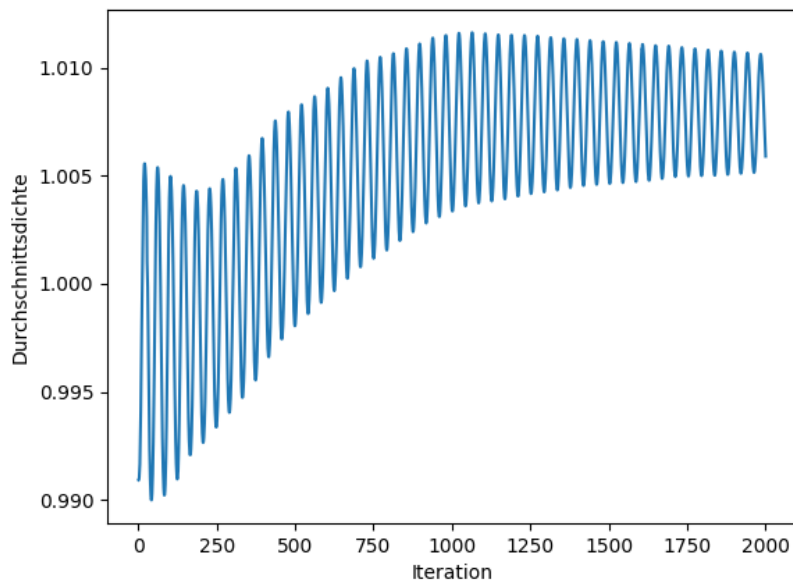


Abbildung 6.11: SSPH Durchschnittsdichte mit mittlerer Steifigkeitskonstante

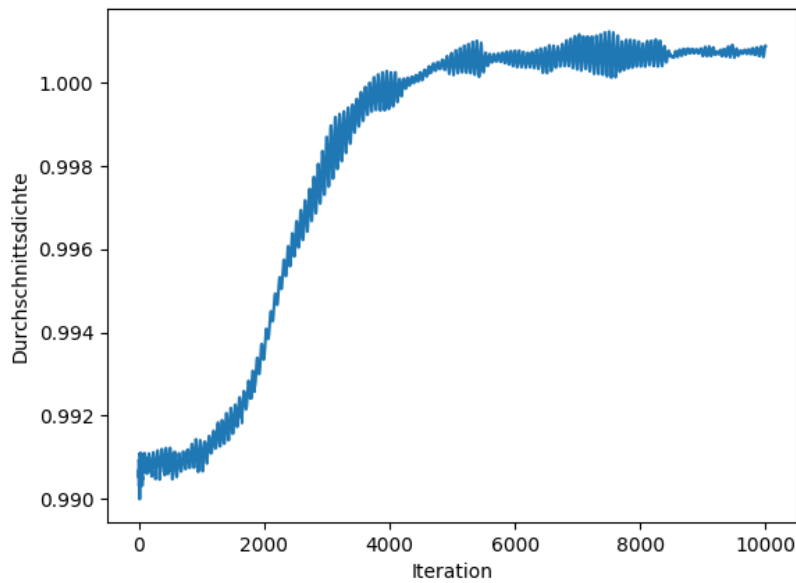


Abbildung 6.12: SSPH Durchschnittsdichte mit hoher Steifigkeitskonstante

## 6.4 Rechenaufwand

Wichtig zu berücksichtigen ist ebenfalls, den Rechenaufwand der unterschiedlichen Druckberechnungsmethoden anzuschauen. Dazu wurden Daten aus acht Dammbruchsimulationen gesammelt, davon sechs mit IISPH und zwei mit SSPH. Die Partikelgröße  $h$  beträgt dabei jeweils 10 und die Viskosität  $\nu$  ist jeweils auf 200 festgelegt. Der Zeitschritt  $\Delta t$  und die Steifigkeitskonstante  $k$  in SSPH variieren nach Simulation.

Ermittelt wurde die benötigte Zeit für 100 Simulationsschritte. Daraus wurde die durchschnittliche Zeit  $\bar{t}_{step}$  für einen Simulationsschritt berechnet. Setzt man diesen Wert in Verhältnis zum Zeitschritt  $\Delta t$ , so erhält man einen Anhaltspunkt, wie schnell sich äquivalente Sequenzen der Simulation berechnen lassen. Sollte die Simulation möglichst schnell berechnet werden, so ist ein möglichst geringer Wert für das Verhältnis  $\bar{t}_{step}/\Delta t$  wünschenswert.

Die Simulationen wurden auf dem Prozessor *Intel® Core™ i5-6600k* berechnet, welcher 4 Kerne besitzt und auf eine Taktfrequenz von 4 Ghz übertaktet ist.

			$\bar{t}_{step}$	$\bar{t}_{step}/\Delta t$
SSPH	$\Delta t = 0.01$	$k = 10^6$	53 ms	5300
SSPH	$\Delta t = 0.05$	$k = 10^5$	53 ms	1060
IISPH	$\Delta t = 0.01$		128 ms	12800
IISPH	$\Delta t = 0.02$		127 ms	6350
IISPH	$\Delta t = 0.03$		145 ms	4833
IISPH	$\Delta t = 0.05$		164 ms	3280
IISPH	$\Delta t = 0.08$		191 ms	2388
IISPH	$\Delta t = 0.1$		210 ms	2100

Aus diesen Daten lässt sich erkennen, dass in IISPH für einen größeren Zeitschritt  $\Delta t$  die durchschnittliche Berechnungszeit  $\bar{t}_{step}$  pro Simulationsschritt zunimmt. Jedoch sinkt das Verhältnis  $\bar{t}_{step}/\Delta t$  gleichzeitig, weswegen die Berechnung durch größere Zeitschritte insgesamt effizienter ist.

In SSPH ist  $\bar{t}_{step}$  relativ konstant, weil bei der Druckberechnung kein Gleichungssystem iterativ gelöst werden muss, wo durch größere Zeitschritte auch größere Fehler entstehen, was die Druckberechnung verlängert. In SSPH kann jedoch für die Steifigkeitskonstante  $k = 10^6$  nur maximal ein Zeitschritt von  $\Delta t = 0.01$  gewählt werden, da die Simulation für größere Zeitschritte sonst instabil wird. Um in SSPH größere Zeitschritte wählen zu können, muss eine geringere Steifigkeitskonstante gewählt werden, was jedoch zu einer weniger realistischen Simulation führt.

## **7 Fazit und Ausblick**



# Literaturverzeichnis

- [GM77] Robert A. Gingold and Joseph J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly notices of the royal astronomical society*, 181(3):375–389, 1977. ISBN: 1365-2966 Publisher: Oxford University Press Oxford, UK.
- [ICS<sup>+</sup>14] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit Incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):426–435, March 2014.
- [IOS<sup>+</sup>14] Markus Ihmsen, Jens Orthmann, Barbara Solenthaler, Andreas Kolb, and Matthias Teschner. SPH fluids in computer graphics. In *EUROGRAPHICS 2014/S. LEFEBVRE AND M. SPAGNUOLO*. Citeseer, 2014.
- [KBST20] Dan Koschier, Jan Bender, Barbara Solenthaler, and Matthias Teschner. Smoothed particle hydrodynamics techniques for the physics based simulation of fluids and solids. *arXiv preprint arXiv:2009.06944*, 2020.
- [LD08] Ares Lagae and Philip Dutré. Compact, fast and robust grids for ray tracing. In *Computer Graphics Forum*, volume 27, pages 1235–1244. Wiley Online Library, 2008. Issue: 4.
- [Luc77] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013, December 1977.
- [Mon92] Joe J. Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30:543–574, 1992. ISBN: 0066-4146.
- [Mon05] Joe J. Monaghan. Smoothed particle hydrodynamics. *Reports on progress in physics*, 68(8):1703, 2005. ISBN: 0034-4885 Publisher: IOP Publishing.
- [Pri12] Daniel J. Price. Smoothed particle hydrodynamics and magnetohydrodynamics. *Journal of Computational Physics*, 231(3):759–794, February 2012.
- [SL03] Songdong Shao and Edmond Y.M. Lo. Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources*, 26(7):787–800, July 2003.