Nikolaus-Kopernikus-Gymnasium Weißenhorn

Abiturjahrgang 2014

Seminararbeit

im W-Seminar

Das Unendliche in der Mathematik

Leitfach:

Mathematik

Effiziente Simulation der Brownschen Bewegung

Verfasser: Max Bastian Mertens

Seminarleiter: Karl Bindl

Abgabetermin: 12. November 2013

abgegeben	

Bewertung:		Punkte		Punkte
	schriftliche Arbeit		x3	
	Abschlusspräsentation		x1	
			Summe:	
			Gesamtleistung nach § 61 (7) GSO = Summe : 2 (gerundet)	

Datum

Unterschrift des Seminarleiters

Inhaltsverzeichnis

1 Einleitung **3**

2 Physikalische und Mathematische Grundlagen **4**

2.1 Brownsche Molekularbewegung 4

2.2 Mathematische Prozesse 6

2.3 Brownsche Bewegung, Wiener-Prozess und Unendlichkeit 8

2.4 Differentialgleichungen und deren Lösung 9

2.4.1 Differentialgleichungen 9

2.4.2 Lösungsverfahren 10

2.5 Partikelsimulationen 13

3 Optimierte Algorithmen: Genauigkeit und Effizienz **16**

3.1 Integrationsverfahren 16

3.2 Barnes-Hut-Algorithmus 17

3.3 Kollisionserkennung 18

4 Simulationsprogramm **19**

4.1 Beschreibung 19

4.2 Details 19

4.3 Kurzanleitung 21

4.4 Durchgeführte Simulations-Experimente und Auswertung 23

5 Fazit **27**

6 Literaturverzeichnis **28**

7 Erklärung **30**

1 Einleitung

Wahl des Themas Die Brownsche Molekularbewegung und ihre mathematische und physikalische Beschreibung interessiert mich bereits seit längerem. Deshalb habe ich mich für das Thema *Effiziente Simulation der Brownschen Bewegung* entschieden. Dieses enthält sowohl Aspekte der Mathematik wie Differentialgleichungen und die Unendlichkeit, Berechnungen aus der Physik sowie Elemente aus der Informatik wie Algorithmen und graphische Darstellungen.

In dieser Arbeit sollen die Brownsche Molekularbewegung anhand eines realen Gases simuliert und die Ergebnisse der Simulation ausgewertet werden. Dabei wird besonderer Wert auf möglichst *physikalisch exakte und effiziente Berechnungen* des Modells gelegt.

Zusammenhang mit bisherigen Arbeiten Im Frühjahr 2013 wurde von mir ein Projekt mit dem Thema *Physikalische Simulation von Teilchenbewegungen* beim Wettbewerb Jugend forscht im Fachgebiet Mathematik/Informatik eingereicht. Dabei wurden bereits Erfahrungen mit der Simulation *makroskopischer* Objekte gesammelt. Abgesehen von diesen Erfahrungen wurde für diese Seminararbeit kein Inhalt des Programmcodes oder der Dokumentation des Jugend forscht-Projekts übernommen. Die beiden Arbeiten verliefen also zeitlich versetzt und in der Umsetzung unabhängig. Das hier vorgestellte Programm ist eine Neuauflage der Software und trägt daher den gleichen Namen *SphereSim*: Das alte Programm hatte jedoch den Versionszweig 1.x, die neue Software besitzt die Versionsnummern 2.x und liegt aktuell in Version 2.4.2 vor.[1]

Beigelegte CD Eine CD mit Daten liegt dieser Arbeit bei. Auf der CD sind die folgenden Ordner enthalten:

- *SphereSim*: enthält das für Windows kompilierte Programm mit allen Abhängigkeiten.
- *Quellcode*: enthält den vollständigen Quelltext des Programms.
- *Seminararbeit*: enthält diese Arbeit in digitaler Form als PDF-Dokument.
- *Dokumentation*: enthält die Dokumentation des Quelltextes. Die Datei *annotated.html* (im Browser öffnen) gibt einen Überblick über alle Klassen des Programms.

[1] Die Versionsnummer 2.4.2 setzt sich aus drei Ziffern zusammen: Die 2 zeigt die Neuauflage der Software an; die 4 deutet auf die vierte größere Änderung des Programms hin; die zweite 2 informiert schließlich über die zweite kleinere Änderung.

2 Physikalische und Mathematische Grundlagen

2.1 Brownsche Molekularbewegung

Entdeckungsgeschichte Die *Brownsche Molekularbewegung* oder *Brownsche Bewegung* wurde von dem schottischen Botaniker Robert Brown (1773-1858) entdeckt und 1827 von ihm als physikalisches Phänomen beschrieben.[1] Unter einem Mikroskop beobachtete er die thermische Bewegung als zufällige, zuckende Bewegungen von in Wasser gelösten Pollen. Christian Wiener konnte 1863 beweisen, dass die größeren Pollenteilchen durch Stöße von den kleineren Wassermolekülen bewegt werden.[2] Albert Einstein war nicht von der Existenz der Brownschen Bewegung überzeugt. Er vermutete in ihr aber einen möglichen Beweis für die endliche Größe von Atomen. Jean Baptiste Perrin konnte aufgrund dieser Überlegung experimentell die Anzahl der Moleküle pro Volumen bestimmen. 1926 erhielt Perrin für seine Arbeit zur Brownschen Bewegung den Nobelpreis für Physik. Norbert Wiener übertrug schließlich 1923 diese nicht differenzierbare Bewegung in die Mathematik und entwickelte den Wiener-Prozess (siehe Kapitel 2.2).[3]

Heute ist die Brownsche Bewegung wichtiger Bestandteil der Physik und Mathematik. Anwendungen sind zum Beispiel die Optimierung von Warteschlangen (z.B. im Supermarkt) und die Modellierung von Börsenkursen durch den sogenannten Wiener-Prozess.[4]

Eigenschaften Allgemein ist mit der physikalischen Brownschen Bewegung eine *Wärmebewegung von in Flüssigkeiten oder Gasen gelösten Teilchen* gemeint. Die Bewegung der Moleküle des Lösungsmittels sowie der gelösten Teilchen selbst folgt jeweils einer zufälligen Bahn und wird auch *Random Walk* genannt. Die Wahrscheinlichkeit für die Bewegung in eine bestimmte Richtung ist dabei für alle Richtungen gleich groß.[5]

Abbildung 2.1 zeigt den Pfad einer Brownschen Bewegung in 100 Schritte aufgelöst. Der Bewegungspfad ist zwar stetig, aber nicht differenzierbar. Es ist die statistische Selbstähnlichkeit der Bewegung sichtbar. Die Brownsche Bewegung ist daher ein Beispiel für ein *Zufallsfraktal*. Die Hausdorff-Dimension, also die fraktale Dimension, der Brownschen Bewegung beträgt

[1] [Sch94], S. 153
[2] [vS06], S. 757
[3] [Wik13b]
[4] [Wol10], S. 1
[5] [Wik13b]

$D_h = 2$. Das lässt vermuten, dass der Bewegungspfad im zweidimensionalen Raum flächendeckend ist. Dies ist jedoch nicht der Fall, denn es gibt viele Selbstüberschneidungen.[6]

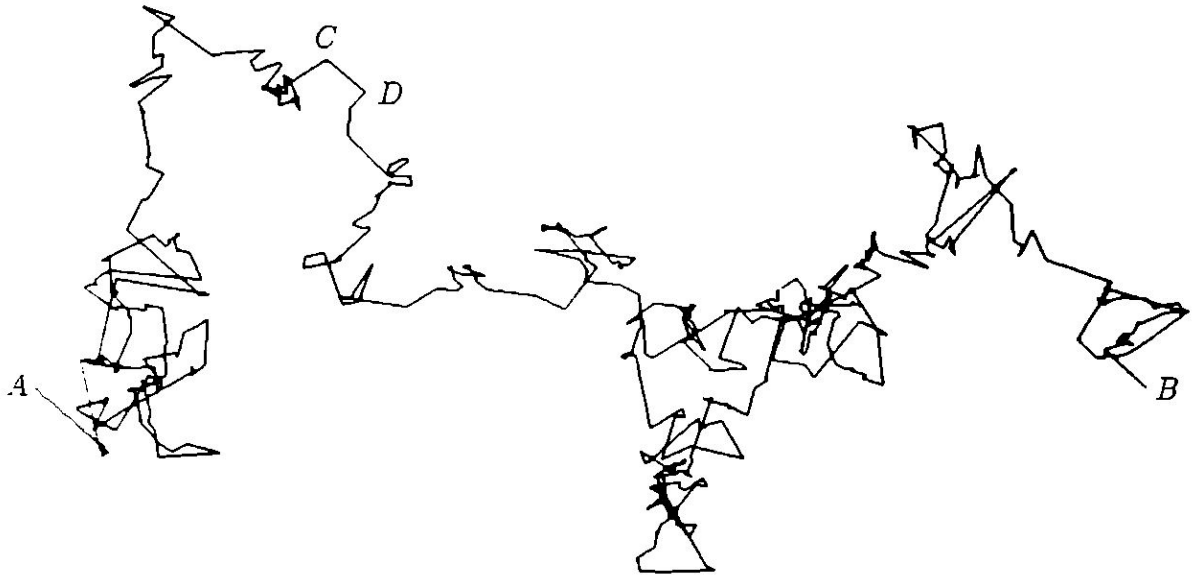

Abb. 2.1: Pfad einer Brownschen Bewegung in 100 Schritte aufgelöst.[7]

Man beobachtet Teilchen, die von ihrer Startposition aus einer Brownschen Bewegung folgen. P ist die Wahrscheinlichkeit für das Ereignis, dass ein Teilchen nach einer beliebigen Zeit an die Startposition zurückkehrt. Die Positionsänderung ist dann im Beobachtungszeitraum insgesamt null. Im zweidimensionalen Raum gilt für diese Rückkehrwahrscheinlichkeit $P = 1$, das Ereignis ist also *fast sicher*. Im dreidimensionalen Raum gilt dagegen $P < 1$, das Ereignis ist also nicht sicher.[8]

Geschwindigkeit Die Intensität der Brownschen Bewegung ist abhängig von Temperatur und Masse der Teilchen. Die Geschwindigkeiten der Teilchen folgen dabei einer *Maxwell-Boltzmann-Verteilung* (für die Gleichungen siehe Kapitel 4.4). In den Abbildungen 2.2 und 2.3 sind die Geschwindigkeiten der Brownschen Bewegung von Gasteilchen abhängig von der Masse bzw. der Temperatur der Teilchen als Diagramm dargestellt. Die Abbildungen zeigen, dass sich die meisten Teilchen mit Geschwindigkeiten bis etwa zur zehnfachen Schallgeschwindigkeit ($c_S \approx 300\ \frac{\mathrm{m}}{\mathrm{s}}$) bewegen.[9]

[6] [Sch94], S. 153-155
[7] [Sch94], S. 155
[8] [Sch94], S. 155-156
[9] [Wik13c]

Größenordnungen Zwischen den Lösungsmittelmolekülen und den darin gelösten Teilchen finden etwa 10^{21} Stöße pro Sekunde statt.[10] Die mittlere freie Weglänge[11] der Teilchen kann dabei bis zu $10^{-9}\,\mathrm{m}$ klein sein, während sich die gesamte Bewegung auf die Gefäßgröße von beispielsweise $0{,}1\,\mathrm{m}$ ausdehnen kann. Für die Brownsche Bewegung skaliert also der Bereich, in dem statistische Selbstähnlichkeit auftritt, von 1 zu 10^8.[12]

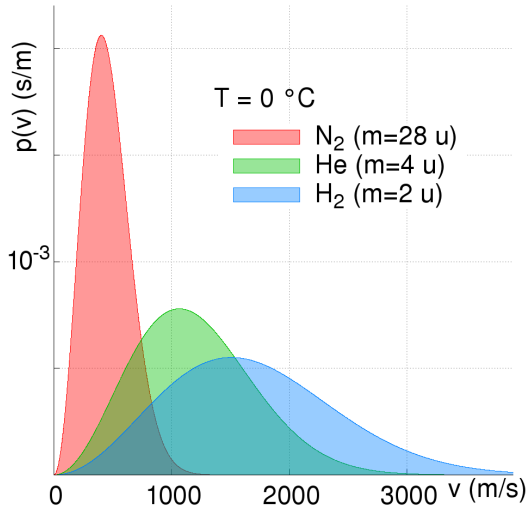

Abb. 2.2: Geschwindigkeiten von Gasteilchen bei den Molekülen N_2, He, H_2 bei Temperatur $0\,^{\circ}\mathrm{C}$.[13]

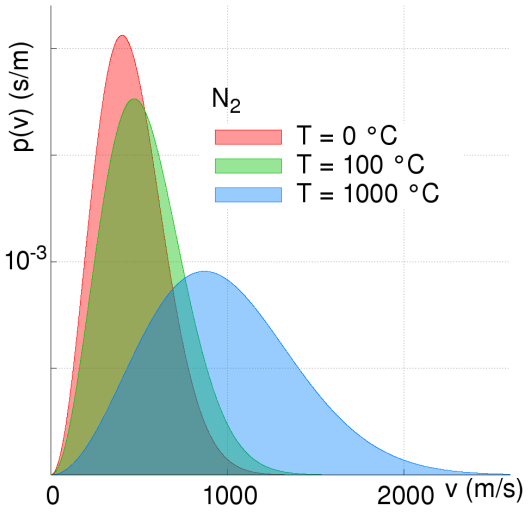

Abb. 2.3: Geschwindigkeiten von N_2-Molekülen bei den Temperaturen $0\,^{\circ}\mathrm{C}$, $100\,^{\circ}\mathrm{C}$, $1\,000\,^{\circ}\mathrm{C}$.[14]

2.2 Mathematische Prozesse

Prozesse allgemein Ein mathematischer *Prozess* $X(t)$ ist eine Funktion X der Zeit $t \in T$ mit Indexmenge T, die den momentanen Zustand eines Systems beschreibt (z.B. die vertikale Geschwindigkeit eines geworfenen Balles oder die Stromstärke in einem elektrischen Schaltkreis). Ein Prozess ist *univariat*, wenn $X(t)$ eine skalare bzw. eindimensionale Funktion ist.

Deterministische Prozesse Für *deterministische Prozesse* gilt: Die Werte $X(t')$ mit $t' \leq t$ legen den Wert $X(t + \mathrm{d}t)$ für ein infinitesimales positives $\mathrm{d}t$ fest.

Bei *gedächtnislosen* deterministischen Prozessen bestimmt allein $X(t)$ den Wert $X(t + \mathrm{d}t)$, d.h. es gibt eine Funktion f sodass gilt: $X(t + \mathrm{d}t) = X(t) + f(X(t), t)\mathrm{d}t$.[15]

[10] [Wik13b]
[11] Mittlere freie Weglänge: durchschnittlich zurückgelegter Weg zwischen zwei Kollisionen.
[12] [Sch94], S. 156
[13] [Wik13c]
[14] [Wik13c]
[15] [Gil91], S. xi

Stochastische Prozesse Bei *stochastischen Prozessen* legen die Werte $X(t')$ mit $t' \leq t$ nur die Wahrscheinlichkeit dafür fest, dass $X(t + \mathrm{d}t)$ für ein infinitesimales positives $\mathrm{d}t$ den Wert v annimmt. Deterministische Prozesse sind ein Spezialfall der stochastischen Prozesse, bei denen die Wahrscheinlichkeit nur für ein v den Wert 1 und sonst den Wert 0 annimmt.

Bei *gedächtnislosen* stochastischen Prozessen hängt die Wahrscheinlichkeit dafür, dass $X(t+\mathrm{d}t)$ einen Wert v annimmt, nur vom vorhergehenden $X(t)$ ab und nicht von mehreren bisherigen $X(t')$ mit $t' < t$.[16] Ein gedächtnisloser Prozess ist also in der unmittelbaren Zukunft allein durch die Gegenwart bestimmt, ungeachtet der Vergangenheit.[17]

Ein stochastischer Prozess heißt *zeitdiskret* bzw. *zufällige Folge*, wenn die Indexmenge T abzählbar ist (z.B. $T \subseteq \mathbb{N}_0$). Andernfalls, also bei überabzählbar großer Indexmenge (z.B. $T \subseteq \mathbb{R}_0$), heißt der Prozess *zeitstetig* bzw. *mit kontinuierlicher Zeit*.

Der *Zuwachs* eines Prozesses, z.B. $X(t_2) - X(t_1)$, beschreibt die Änderung des Zustands des Prozesses zwischen zwei Zeitpunkten t_1 und t_2.

Ein stochastischer Prozess besitzt *unabhängige Zuwächse*, wenn für alle $t_1, t_2, \ldots, t_n \in T$ gilt, dass alle Zuwächse von $X(t)$, also $X(t_2) - X(t_1), X(t_3) - X(t_2), \ldots, X(t_n) - X(t_{n-1})$, stochastisch unabhängig sind.

Ein stochastischer Prozess hat außerdem *stationäre Zuwächse*, wenn die Verteilungen der Zuwächse $X(t_n + h) - X(t_{n-1} + h)$ für alle $t_n, t_{n-1} \in T$ nur von t_n, t_{n-1} abhängen und unabhängig von h sind.[18]

Markov-Prozesse Ein *Markov-Prozess*[19], oder auch eine *Markov-Kette*, ist definiert als ein gedächtnisloser stochastischer Prozess.[20] Genauer versteht man darunter eine Folge von Zuständen $x_0, x_1, \ldots, x_n$ eines Systems. Die Wahrscheinlichkeit für das Eintreten des Zustands x_n ist aufgrund der Gedächtnislosigkeit nur von x_{n-1} abhängig. D.h. für beliebige n gilt[21]:

$$P(x_n|x_{n-1}, x_{n-2}, \ldots, x_0) = P(x_n|x_{n-1}). \tag{2.1}$$

Dabei drückt $P(a|b_1, b_2, \ldots, b_n)$ die bedingte Wahrscheinlichkeit für ein Ereignis a unter den Bedingungen $b_1, b_2, \ldots, b_n$ aus.

Ein Markov-Prozess ist *homogen*, wenn die Zuwächse $X(t_n) - X(t_{n-1})$ für alle $t_n, t_{n-1} \in T$ nur von $t_n - t_{n-1}$ abhängen und unabhängig von $X(t_n), X(t_{n-1})$ sind.[22]

[16] [Gil91], S. xii
[17] [Hee90], S. 52
[18] [Sch11], S. 11, [App09], S. xxi
[19] in deutscher Literatur auch *Markow*-Prozess, siehe [Sch11], S. 5 u.a.
[20] [Gil91], S. xii
[21] [Hee90], S. 52
[22] [Gil91], S. 99

Lévy-Prozesse *Lévy-Prozesse* sind Markov-Prozesse (also gedächtnislose, stochastische Prozesse) mit unabhängigen, stationären Zuwächsen.[23]

Wiener-Prozesse Ein *Wiener-Prozess*, die mathematische Bezeichnung für die Brownsche Bewegung, ist ein zeitstetiger, homogener Lévy-Prozess.[24] Vollständig lässt sich ein Wiener-Prozess $W(t)$ also charakterisieren:

- $W(t)$ ist eine Funktion der Zeit t und für alle Elemente $t_1, t_2, \ldots t_n \in T$ der Indexmenge T definiert.
- T ist überabzählbar groß, $W(t)$ ist also stetig.
- $W(t_n)$ ist nur von $W(t_{n-1})$ abhängig, mit $t_n, t_{n-1} \in T$.
- Für ein beliebiges $W(t)$ kann nur die Wahrscheinlichkeit bestimmt werden, dass $W(t + \mathrm{d}t)$ einen Wert v annimmt, nicht der Wert $W(t + \mathrm{d}t)$ selbst.
- Die Zuwächse $W(t_n) - W(t_{n-1})$ für alle $t_n, t_{n-1} \in T$ sind stochastisch unabhängig.
- Die Verteilung der Zuwächse $W(t_n + h) - W(t_{n-1} + h)$ für alle $t_n, t_{n-1} \in T$ ist nur von $t_n - t_{n-1}$ abhängig und unabhängig von $h, W(t_n + h), W(t_{n-1} + h)$.

2.3 Brownsche Bewegung, Wiener-Prozess und Unendlichkeit

Der Wiener-Prozess ist die mathematische Beschreibung der physikalischen Brownschen Bewegung. Das mathematische und das physikalische Modell stimmen jedoch nicht vollständig überein.

Der beschriebene Bewegungspfad lässt sich größer bzw. kleiner skalieren und genauer bzw. weniger genau auflösen. Der *mathematische Wiener-Prozess* kann *bis ins Unendliche* verkleinert oder vergrößert werden und beschreibt dabei beliebig kleine bzw. große selbstähnliche Bewegungen. Da die Bewegung zwischen zwei Punkten beliebig oft aufgelöst werden kann, wird auf dieser Bewegung eine unendlich große Weglänge zurückgelegt.

Die *physikalische Brownsche Bewegung* ist dagegen *begrenzt*: Teilchen wie beispielsweise Atome haben eine räumliche Ausdehnung und abhängig von der Temperatur eine bestimmte Anzahl an Teilchen pro Volumen. Daraus ergibt sich eine mittlere freie Weglänge, die endlich groß ist. Innerhalb dieser freien Weglänge bewegt sich ein Teilchen ohne Kollision mit anderen Teilchen, also annähernd geradlinig. Sofern der Stoff, beispielsweise das Gas, sich in einem endlich großen Gefäß befindet, ist die Brownsche Bewegung auch durch die Größe dieses Behälters begrenzt. Wenn die Brownsche Bewegung bis in die Größenordnung der mittleren freien

[23] [App09], S. xv
[24] [Gil91], S. 129

Weglänge oder der Gefäßgröße skaliert wird, entfällt also ihre Selbstähnlichkeit. Aus der mittleren freien Weglänge und der mittleren Zeit zwischen zwei Kollisionen kann die mittlere Geschwindigkeit abgeschätzt werden. Diese mittlere Geschwindigkeit der Brownschen Bewegung ist endlich groß. Beim Wiener-Prozess wäre dagegen eine unendlich große Geschwindigkeit nötig, um die unendlich große Weglänge zurückzulegen, was der Relativitätstheorie widerspricht.

Die *Simulation* von unendlich kleinen, unendlich schnellen Bewegungen ist nicht möglich, da PCs nur mit endlichen Zahlen rechnen können. Aus diesem Grund und um eine möglichst naturgetreue Berechnung durchzuführen, wird bei der hier vorgestellten Simulation auf das physikalische Modell zurückgegriffen.

2.4 Differentialgleichungen und deren Lösung

2.4.1 Differentialgleichungen

Allgemein Für Simulationen von bewegten Objekten werden die Bewegungsgleichungen der Klassischen Mechanik verwendet. Die simulierten Objekte bewegen sich mit Geschwindigkeiten, die weit unterhalb von $10\,\%$ der Lichtgeschwindigkeit liegen, es muss also nicht relativistisch gerechnet werden. Für die Kraft gilt deshalb: $\vec{F} = m \cdot \vec{a}$. Die Beschleunigung ist dabei die Ableitung der Geschwindigkeit nach der Zeit:

$$\vec{a} = \dot{\vec{v}} = \frac{\mathrm{d}\vec{v}}{\mathrm{d}t}. \tag{2.2}$$

Die Geschwindigkeit ist die Ableitung des Orts nach der Zeit:

$$\vec{v} = \dot{\vec{x}} = \frac{\mathrm{d}\vec{x}}{\mathrm{d}t}. \tag{2.3}$$

Für einfache physikalische Experimente wie dem freien Fall, bei denen die Beschleunigung konstant ist, gilt dabei:

$$\vec{\Delta x} = \vec{v} \cdot \Delta t + \frac{1}{2} \cdot \vec{a} \cdot \Delta t^2 \tag{2.4}$$

Bei Simulationen hängt jedoch die Kraft und damit die Beschleunigung auch vom Ort und der Geschwindigkeit der simulierten Objekte ab. Zur Berechnung der Bewegung werden deshalb Differentialgleichungen benötigt.

Definition Eine *Differentialgleichung* n-ter Ordnung ist eine Gleichung, die eine Variable, eine Funktion und deren Ableitung n-ter Ordnung sowie evtl. auch Ableitungen niedrigerer Ordnungen enthält. Ist die Funktion nur von *einer* reellen Variablen abhängig, spricht man von einer *gewöhnlichen Differentialgleichung* (DGL). Wenn die Gleichung nach der n-ten Ableitung

aufgelöst ist, spricht man von einer *expliziten*, ansonsten von einer *impliziten* DGL.[25]

Ein Beispiel für eine DGL ist die folgende Zerfallsgleichung:

$$N'(t) = -\frac{\ln 2}{T_{1/2}} \cdot N(t) \tag{2.5}$$

Die Funktion $N(t)$ bzw. ihre Ableitung $N'(t)$ geben die momentane Anzahl radioaktiver Atomkerne bzw. den momentanen Zerfall pro Zeit zum Zeitpunkt t an. Im Beispiel ist $T_{1/2} = 2\,\mathrm{h}$ die Halbwertszeit der Atomkerne. Die Gleichung enthält nur die erste Ableitung der Funktion $N(t)$ und ist nach dieser aufgelöst, es handelt sich also um eine explizite DGL 1. Ordnung.

Anfangswertprobleme Hat man eine DGL wie Gleichung (2.5) und einen Startwert gegeben und möchte den Wert der Funktion bei einem bestimmten Wert der Variablen wissen, so handelt es sich um ein *Anfangswertproblem* (AWP). Um ein AWP lösen zu können, müssen der Startwert der Variablen sowie alle Werte der Funktion und der in der DGL verwendeten Ableitungen bis zur Ordnung $p - 1$ für diesen Variablenwert bekannt sein.[26] Im Beispiel sind die DGL (2.5), die Startzeit $t_0 = 0$ und der Startwert von $N(t_0) = 10\,000$ Atomkernen gegeben und die Anzahl der Atomkerne nach 16 Stunden soll bestimmt werden.

2.4.2 Lösungsverfahren

Lösungsverfahren Damit für ein AWP eine Lösung gefunden werden kann, muss über die abgeleitete Funktion integriert werden. Eine DGL heißt explizit lösbar, wenn sie sich durch elementare Funktionen lösen lässt.[27] Gleichung (2.5) ist eine einfache DGL und daher explizit lösbar:

$$N(t) = N(t_0) \cdot 0{,}5^{\frac{t}{T_{1/2}}} \tag{2.6}$$

Komplexere DGLs sind selten explizit lösbar, sodass oft ein *numerisches Lösungsverfahren* benötigt wird. Im Beispiel wird angenommen, dass keine explizite Lösung vorhanden ist, um die numerische Lösung zu demonstrieren. Um ein AWP zu lösen, wird die abgeleitete Funktion $N'(t)$ mehrfach an den Stellen $t_0, t_0 + h, \ldots, t_0 + n \cdot h$ mit Schrittweite $h > 0$ ausgewertet und es wird über diese *diskreten* Werte integriert. Dabei wird grundsätzlich zwischen zwei Arten von Lösungsverfahren unterschieden: *Einschrittverfahren* verwenden lediglich einen bereits bekannten Funktionswert $N(t-h)$, während *Mehrschrittverfahren* auf mehrere bekannte Funktionswerte $N(t-h), N(t-2\cdot h), \ldots N(t-n\cdot h)$ zurückgreifen, um den nächsten Wert $N(t)$ zu berechnen.

[25] [Mey01], S. 1-2
[26] [Mey01], S. 4
[27] [Mey01], S. 4-5

Einschrittverfahren Einschrittverfahren haben jeweils eine bestimmte *Ordnung* p. Die Ordnung gibt an, wie sich eine Änderung der Schrittweite auf den Fehler auswirkt. Eine Skalierung der Schrittweite h um den Faktor a verursacht eine Änderung des Fehlers um etwa den Faktor a^p. Je höher die Ordnung eines Einschrittverfahrens ist, desto mehr lohnt sich also eine kleine Schrittweite. Einschrittverfahren mit höherer Ordnung werten die abgeleitete Funktion pro Schritt mehrmals aus, der Rechenaufwand ist also größer. Aufgrund der besseren Genauigkeit lohnen sich Verfahren höherer Ordnung jedoch: In Einzelfällen kann z.B. für ein Verfahren der Ordnung 1 die Berechnung der Lösung eine etwa 260 000 mal längere Zeit benötigen als bei einem Verfahren der Ordnung 4, wenn ein maximaler Gesamtfehler gefordert ist.[28]

Das bekannteste numerische Lösungsverfahren ist die *Euler-Polygonzugmethode*[29], auch bekannt unter dem Namen *Methode der kleinen Schritte*. Dieses ist ein Einschrittverfahren der Ordnung 1. Vom Zeitpunkt t_0 ausgehend wird angenommen, dass sich die Funktion $N'(t)$ in einem möglichst kleinen Zeitabschnitt $h > 0$ (Schrittweite) nur minimal ändert. Man nimmt für den gesamten Zeitraum $I = [t_0; t_0 + h[$ einen konstanten Funktionswert an, d.h. $N'(t_0 + \Delta t) \approx N'(t_0)$ für $0 \leq \Delta t < h$. Nun ist es möglich, über diesen Zeitabschnitt zu integrieren, ohne den genauen Verlauf der Funktion $N'(t)$ während des Zeitraums I zu kennen:

$$\int_{t_0}^{t_0+h} N'(t)\mathrm{d}t \approx \int_{t_0}^{t_0+h} N'(t_0)\mathrm{d}t = h \cdot N'(t_0) \tag{2.7}$$

Die Lösung des AWP erfolgt ähnlich:

$$N(t_0 + h) = N(t_0) + \int_{t_0}^{t_0+h} N'(t)\mathrm{d}t \approx N(t_0) + \int_{t_0}^{t_0+h} N'(t_0)\mathrm{d}t = N(t_0) + h \cdot N'(t_0) \tag{2.8}$$

Das Einsetzen der bekannten Funktionswerte $N(t_0)$ und $N'(t_0)$ und der Zeitdauer $h = T = 16\,\mathrm{h}$ in Gleichung (2.8) ergibt:

$$N(T) = N(t_0) + \int_{t_0}^{T} N'(t)\mathrm{d}t \approx N(0) + T \cdot N'(0) \approx -45452 \tag{2.9}$$

Diese Lösung ist nicht sinnvoll (eine negative Anzahl an Atomkernen) und zeigt die große Ungenauigkeit dieses Verfahrens. Ursache für den großen Fehler ist die zu große Schrittweite $h = 16\,\mathrm{h}$.

Als Verbesserung wird eine *Diskretisierung* des AWP durchgeführt. D.h. man unterteilt die Zeit T in n kleinere Schrittweiten h, sodass gilt: $h = T/n$. Außerdem werden *Stützstellen* $N_0 = N(t_0), N_1 = N(t_0 + h), \ldots N_n = N(t_0 + n \cdot h)$ eingeführt, die jeweils nach der Euler-

[28] [Mey01], S. 61
[29] [Mey01], S. 7

Polygonzugmethode berechnet werden:

$$\begin{aligned} N_0 &= N(t_0), \\ N_1 &= N_0 + N'(t_0) \cdot h, \\ N_2 &= N_1 + N'(t_0 + h) \cdot h, \\ N_3 &= N_2 + N'(t_0 + 2 \cdot h) \cdot h, \\ &\dots \\ N_n &= N_{n-1} + N'(t_0 + (n-1) \cdot h) \cdot h \end{aligned} \tag{2.10}$$

Als Beispiel werden $n = 32$ Schritte mit einer Schrittweite von $h = 0{,}5\,\mathrm{h}$ verwendet. Im Vergleich zum Wert aus Gleichung (2.9) liegt der nun ermittelte Wert von $N_{32} \approx 23$ Atomkernen mit etwa $72\,\%$ Abweichung deutlich näher an der expliziten Lösung $N(16\,\mathrm{h}) \approx 39$, berechnet mit Gleichung (2.6).

Effiziente Einschrittverfahren Um die Lösung des AWP genauer zu berechnen, kann die Schrittweite verringert werden. Dadurch steigen die Genauigkeit, aber auch der Rechenaufwand in etwa proportional (Verfahren der Ordnung 1). Als Alternative können auch Einschrittverfahren höherer Ordnung verwendet werden.

Runge-Kutta-Verfahren sind eine Klasse von Einschrittverfahren mit höherer Ordnung. Ein Beispiel ist das *Heun-Verfahren* mit Ordnung 2.[30] Hierbei wird zur Berechnung eines Schritts von t_0 bis $t_0 + h$ die abgeleitete Funktion $N'(t)$ zweimal ausgewertet und die Werte als Stützstellen K_1, K_2 zur Berechnung verwendet:

$$\begin{aligned} K_1 &= N'(t_0), \\ K_2 &= N'(t_0 + h), \\ N'_{neu}(t_0) &= \frac{1}{2}(K_1 + K_2). \end{aligned} \tag{2.11}$$

$K_1 = N'(t_0)$ kann mit Gleichung (2.5) direkt über den bekannten Wert $N(t_0)$ berechnet werden. $N(t_0 + h)$ ist zu diesem Zeitpunkt unbekannt. Deshalb wird dieser Wert mit der Euler-Polygonzugmethode berechnet, um $K_2 = N'(t_0 + h)$ bestimmen zu können: $N(t_0 + h) = N(t_0) + N'(t_0) \cdot h$. Es folgt also:

$$N(t_0 + h) = N(t_0) + N'_{neu}(t_0) \cdot h. \tag{2.12}$$

Adaptive Schrittweite Je größer die Schrittweite ist, desto ungenauer wird das Ergebnis des Lösungsverfahrens. Je kleiner die Schrittweite jedoch gewählt wird, desto mehr Rechenauf-

[30] [Mey01], S. 58

wand ist zur Lösung notwendig. Es muss also über eine *Schrittweitenkontrolle* ein sinnvoller Wert für die Schrittweite gefunden werden. *Adaptive Schrittweite* bedeutet, dass die Schrittweite an die momentan zu lösende DGL angepasst wird. Dazu wird die Lösung jeweils mit zwei Runge-Kutta-Verfahren unterschiedlicher Ordnung berechnet. Der Fehler, also die Differenz zwischen den beiden Lösungen, wird als Kriterium gewählt, ob die aktuelle Schrittweite klein genug ist, oder ob die Schrittweite halbiert und der aktuelle Schritt erneut berechnet wird. Bei *eingebetteten Verfahren* teilen sich die zwei Runge-Kutta-Verfahren möglichst viele Stützstellen, um den Rechenaufwand zu reduzieren.[31] Bekannte eingebettete Verfahren mit adaptiver Schrittweite sind das Bogacki-Shampine-Verfahren (Runge-Kutta-Verfahren mit Ordnung 3 und 2), das Cash-Karp-Verfahren, das Runge-Kutta-Fehlberg-Verfahren und das Dormand-Prince-Verfahren (jeweils Ordnung 5 und 4).[32]

2.5 Partikelsimulationen

Allgemein Partikelsysteme werden bereits seit den 1960er Jahren simuliert.[33] Seitdem ist die Leistungsfähigkeit der PCs ständig gestiegen und optimierte Algorithmen wurden entwickelt. Am Max-Planck-Institut für Astrophysik wurden im Jahr 2010 auf Supercomputern etwa 300 Milliarden Teilchen simuliert, um beispielsweise die Entstehung des Universums zu untersuchen.[34] Aber auch im kleineren Maße werden beispielsweise in Bachelorarbeiten Simulationen der Gravitation von $10\,000$ bis $100\,000$ Teilchen durchgeführt[35].

In dieser Arbeit wird ausschließlich auf die Simulationen von kugelförmigen Objekten eingegangen, da andere Formen wie Würfel auf ein Gitter von Kugeln zurückgeführt werden können. Die Berechnungen werden anhand von Kugeln erläutert, für zweidimensionale Kreise gelten jedoch die gleichen Formeln, wenn nicht anders vermerkt.

Kollisionserkennung Zur Simulation makroskopischer Objekte müssen Kollisionen, also physikalische Stöße, erkannt und berechnet werden. *Elastisch* ist ein Stoß dabei, wenn keine Energie durch Reibung verloren geht, wie es bei einem *(teil-)plastischen* Stoß der Fall ist. Bei der Simulation eines Billiard-Spiels werden beispielsweise die Winkel berechnet, in denen die Kugeln voneinander und von den Wänden abprallen. Die Kollisionserkennung bei Kugeln ist einfach im Vergleich zu anderen Formen: Eine Kugel kollidiert genau dann mit einer Wand, wenn der Abstand Δx ihres Mittelpunktes zur Wand kleiner als ihr Radius R ist: $\Delta x < R$. Die

[31] [Mey01], S. 63
[32] [Wik13d]
[33] [Ree83], S. 92
[34] [Ang11]
[35] [Fra10], S. 47

Kraft, die dann auf die Kugel bzw. die Wand wirkt, beträgt nach der Kontaktmechanik[36]:

$$F = \frac{4}{3} \cdot E^* \cdot R^{1/2} \cdot d^{3/2} \tag{2.13}$$

Der Ersatz-Elastizitätsmodul beträgt:

$$\frac{1}{E^*} = \frac{1-\nu_1^2}{E_1} + \frac{1-\nu_2^2}{E_2}.$$

Dabei sind E_1, E_2 die Elastizitätsmodule, also ein Maß für die Härte des Materials, und ν_1, ν_2 die Poisson-Zahlen, ein Maß für die horizontale Ausdehnung bei vertikalem Druck, der Kugel und der Wand. Für die Eindrucktiefe d gilt: $d = R - \Delta x$.

Bei zwei Kugeln findet eine Kollision genau dann statt, wenn der Abstand zwischen ihren Mittelpunkten $\vec{x}_1, \vec{x}_2$ kleiner als die beiden Radien R_1, R_2 zusammen ist:

$$\Delta x = |\vec{x}_2 - \vec{x}_1| < R_1 + R_2.$$

Für die Kraft gelten ebenfalls Gleichungen (2.13) und (2.5). R ist dann als Ersatzradius für beide Kugeln definiert:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}.$$

Gravitation Für die Simulation größerer Objekte wie die Planeten unseres Sonnensystems sind die Kräfte der Gravitation von großer Bedeutung. In den meisten Simulationen und auch in dieser Arbeit werden diese Objekte als Punktmasse angenommen, da das bei kugelförmigen Objekten auch bei geringen Abständen eine gute Näherung ist. Die Anziehungskraft zwischen zwei Massen m_1, m_2 mit dem Abstand r beträgt nach dem newtonschen Gravitationsgesetz[37]:

$$F_G = G \cdot \frac{m_1 m_2}{r^2}. \tag{2.14}$$

Dabei ist G die Gravitationskonstante[38] und beträgt etwa $G \approx 6{,}6738 \cdot 10^{-11} \frac{\mathrm{m}^3}{\mathrm{kg \cdot s}}$.

Lennard-Jones-Potential Die Kräfte des physikalischen Stoßes gelten nur für makroskopische Objekte. Für mikroskopische Objekte gibt es dagegen eigene Kräfte, die nur für diese Größenordnungen gelten. Ein Beispiel ist das Lennard-Jones-Potential $V(r)$, das verschiedene Kräfte zwischen solchen Objekten zusammenfasst. Dieses Potential entspricht der potentiellen Energien E_{pot} bzw. dem Integral über die Beschleunigung a von ∞ nach dem Abstand r und

[36] [GP07], S. 036710-1
[37] [Mey13], S. 13
[38] [Mey13], S. 39

wird definiert als:

$$V(r) = E_{pot} = \int_{\infty}^{r} a \, \mathrm{d}r_0 = 4\epsilon \left(\left(\frac{\sigma}{r} \right)^{12} - \left(\frac{\sigma}{r} \right)^{6} \right).$$

Die Kraft wird aus der Ableitung des Potentials bestimmt[39]:

$$F_{LJ}(r) = \frac{\mathrm{d}V(r)}{\mathrm{d}r} = -48 \left(\frac{\epsilon}{\sigma} \right) \left(\left(\frac{\sigma}{r} \right)^{13} - \frac{1}{2} \left(\frac{\sigma}{r} \right)^{7} \right). \tag{2.15}$$

Verwendung der berechneten Kräfte Um die Bewegung der simulierten Objekte berechnen zu können, werden die bereits genannten, auf ein Objekt wirkenden Teilkräfte (siehe Gleichungen (2.13) bis (2.15)) aufsummiert: $F_{ges} = F_1 + F_2 + \cdots F_n$. Die Gesamtkraft wird dann zur Berechnung der Bewegung in die Differentialgleichungen eingesetzt (siehe Kapitel 2.4).

Bisher wurden nur Formeln genannt, mit denen die Beträge der Kräfte berechnet werden können. Alle diese Kräfte wirken jeweils vom Mittelpunkt des einen zum Mittelpunkt des anderen Objekts:

$$\vec{F} = F \cdot \frac{\Delta\vec{x}}{|\Delta\vec{x}|} \tag{2.16}$$

Mit diesen vektoriellen Kräften können nun die Bewegungen der simulierten Objekte in die verschiedenen Richtungen bestimmt werden.

Randbedingungen Simulierte Objekte befinden sich immer in einem quaderförmigen Simulationsbereich. Im Makroskopischen können Abstoßungskräfte an den Wänden dieses Quaders berechnet werden. Bei der Simulation von Atomen werden dagegen häufig periodische Randbedingungen gewählt: Während der Simulation werden Atome, die durch eine Wand den Quader verlassen haben, auf der anderen Seite mit unveränderter Geschwindigkeit eingefügt. Der gedachte Simulationsbereich ist also in jede Raumrichtung unendlich groß, es existieren praktisch unendlich viele Kopien des gleichen Quaders nebeneinander. Es werden nur die jeweils am nächsten liegenden Kopien der anderen Atome zur Berechnung der Kräfte auf ein Atom berücksichtigt. Die anderen Kopien üben nur verschwindend geringe, sich gegenseitig aufhebende Kräfte auf das eine Atom aus.

[39] [Hee90], S. 30

3 Optimierte Algorithmen: Genauigkeit und Effizienz

In dem vorgestellten Programm wird besonderer Wert auf eine große Effizienz gelegt. Je weniger Rechenzeit notwendig ist, desto genauer kann eine Simulation bei gleicher oder geringerer Rechenzeit durchgeführt werden. Eine Optimierung in der Effizienz bedeutet also auch eine Optimierung in der Genauigkeit.

3.1 Integrationsverfahren

Da eine Schrittweitenkontrolle für ein Simulationsprogramm notwendig ist, werden für diese Arbeit nur eingebettete Verfahren mit adaptiver Schrittweite in Betracht gezogen (siehe Kapitel 2.4.2). Damit die Berechnungen möglichst genau und effizient durchgeführt werden können, wurden verschiedene Integrationsverfahren getestet und verglichen. Dazu wurde ein hüpfender Gummiball mit Erdanziehungskraft und Abstoßung vom Boden für die Zeit von einer Sekunde simuliert. Es wurde gemessen, wie viele Auswertungen N der Beschleunigung (also wie viel Rechenaufwand) zur Simulation notwendig waren und wie groß der relative Fehler f der Gesamtenergie nach der Simulation ist. Für die Anfangsenergie E_A, die Endenergie E_E und den relativen Fehler f gilt dabei:

$$f = \left| 1 - \frac{E_A}{E_E} \right| \tag{3.1}$$

Der Fehler liegt also im Intervall $[0; \infty[$. Diese beiden Werte liefern keine direkte Vergleichbarkeit der Verfahren, daher werden sie mit der folgenden Formel zu einem Wert W zusammengefasst:

$$W(f, N) = 20 - 0{,}1 \cdot \ln(f) - \ln(N) \tag{3.2}$$

Die Formel basiert auf keinen mathematischen Gesetzmäßigkeiten und wurde ausschließlich zum Vergleich verschiedener Verfahren für diese Arbeit entwickelt. Durch die natürlichen Logarithmen beeinflussen große Werte von f und N das Ergebnis in sehr geringem Maß. Falls Anfangs- und Endenergie *exakt gleich* sind, also $E_E = E_A$, dann ist $f = 0$ und deshalb $W(f, N) = +\infty$. Dieser Fall tritt aber aufgrund von Ungenauigkeiten bei den Berechnungen durch PCs praktisch nie auf[1]. Der Faktor $0{,}1$ gewichtet den Fehler geringer als den Rechenaufwand. Der Summand 20 und die negativen Vorzeichen verschieben die Werte W so, dass

[1] Da die Berechnungen mit 64-bit-Gleitkommazahlen durchgeführt werden, treten nach knapp 20 Nachkommastellen Rundungsfehler auf, die aufgrund der heutigen Prozessorarchitekturen unumgänglich sind. Dass die *berechnete* Gesamtenergie des Systems exakt gleich bleibt, ist also sehr unwahrscheinlich.

meistens kleine positive Zahlen entstehen und dass größere Werte ein „besseres“ (effizienteres und genaueres) Integrationsverfahren bedeuten.

Für verschiedene Verfahren wurden die folgenden Ergebnisse festgestellt:

Verfahren:	Bogacki-Shampine	Runge-Kutta-Fehlberg	Dormand-Prince	Cash-Karp
N:	4236	702	721	582
f:	$2{,}1 \cdot 10^{-5}$	$2{,}3 \cdot 10^{-4}$	$1{,}4 \cdot 10^{-4}$	$1{,}5 \cdot 10^{-4}$
W:	12,72	14,28	14,30	14,51

Da beim Cash-Karp-Verfahren ein akzeptabler Fehler entsteht und nur sehr wenig Rechenschritte benötigt werden, ist der Wert W hierfür am größten. Das Bogacki-Shampine-Verfahren liefert zwar einen deutlich geringeren Fehler; zur Simulation sind jedoch weitaus mehr Rechenschritte nötig. Das Cash-Karp-Verfahren ist also für die Simulation am besten geeignet und wird für die vorgestellten Simulationen verwendet.

3.2 Barnes-Hut-Algorithmus

Bei der Berechnung der Gravitation und des Lennard-Jones-Potentials müssen zwischen jeweils zwei Teilchen der Abstand bestimmt und die Kraft berechnet werden. Bei einer Anzahl von n Teilchen müssen also $\frac{1}{2}n(n-1)$ Kräfte berechnet werden, der Rechenaufwand entspricht also $\mathcal{O}(n^2)$.[2] Für eine doppelte Anzahl von Teilchen ist also eine etwa viermal so lange Rechenzeit oder ein viermal so leistungsfähiger PC nötig. Für die möglichst realistische Simulation eines Gases sollten jedoch möglichst viele Teilchen verwendet werden. Es wird also ein Algorithmus gesucht, der bei einer großen Menge an Teilchen eine langsamer ansteigende Rechenzeit benötigt.

Bei der Berechnung der Anziehungskraft der Sonne auf die Planeten des Sonnensystems werden die einzelnen Teilchen der Sonne zu einer Gesamtmasse mit einem Massenschwerpunkt zusammengefasst. Die Ausdehnung der Sonne ist im Vergleich zum Abstand zu den Planeten vernachlässigbar. Dieses Prinzip verwendet ebenfalls der *Barnes-Hut-Algorithmus*[3]: Es werden *weit* entfernte Teilchengruppen zu Massenschwerpunkten zusammengefasst und diese zur Berechnung der Gravitation und des Lennard-Jones-Potentials verwendet. Die Kräfte nah beieinander liegender Teilchen werden jedoch wieder paarweise berechnet. Als Bedingung für eine *weit* entfernte Teilchengruppe wird $d/r < \theta_{max}$ mit Abstand r, Länge der Diagonale der Teilchengruppe d und $\theta_{max} = 0{,}5$ verwendet. Dadurch verhält sich die Rechenzeit insgesamt wie $\mathcal{O}(n \cdot \ln n)$, sie steigt also deutlich langsamer an.

[2] $\mathcal{O}$ ist ein Landau-Symbol. D.h. es wird asymptotisch abgeschätzt, wie schnell der Rechenaufwand für große n ansteigt. Es gilt: $\mathcal{O}(r \cdot f(n)) = \mathcal{O}(f(n))$ und $\mathcal{O}(f(n) + r) = \mathcal{O}(f(n))$ für alle $r \in \mathbb{R}$.

[3] [Wik13a]

[4] Eigene Quelle

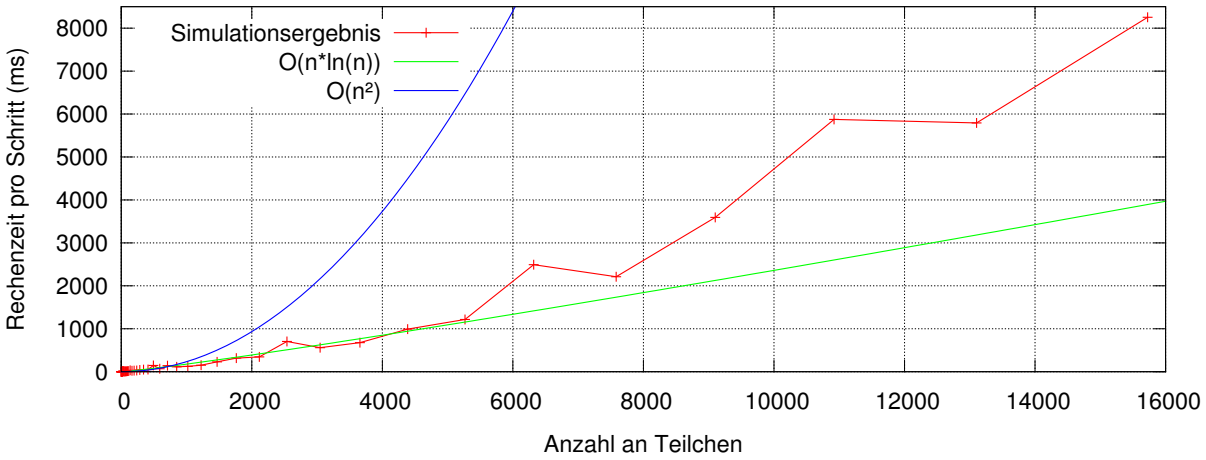

Abb. 3.1: Rechenzeit (rot) und angenäherte Funktionen mit $\mathcal{O}(n \cdot \ln n)$ (grün) und $\mathcal{O}(n^2)$ (blau).[4]

In Abbildung 3.1 sind die Rechenzeiten bei der Simulation von Gravitation zwischen den Teilchen dargestellt. Die Kurve der Simulationsergebnisse liegt deutlich näher an $\mathcal{O}(n \cdot \ln n)$ als an $\mathcal{O}(n^2)$, der implementierte Algorithmus ist also sehr effizient. Bei einer Bachelorarbeit, in der die Gravitation zwischen $10\,000$ Teilchen mit einem vergleichbaren Algorithmus simuliert wird, dauert ein Rechenschritt mindestens 26 Sekunden.[5] Wie die Abbildung 3.1 zeigt, dauert die Berechnung der Gravitation bei $10\,000$ Teilchen mit der hier vorgestellten Software weniger als 5 Sekunden. Die Software ist also wesentlich effizienter.

Durch die Zusammenfassung von Massenschwerpunkten entstehen geringe Ungenauigkeiten. Die größere Effizienz ermöglicht jedoch mehr simulierte Rechenschritte pro Sekunde. Insgesamt ist also eine deutlich genauere Simulation möglich.

3.3 Kollisionserkennung

Zur Kollisionserkennung müssen wie bei der Gravitation die Abstände zwischen jeweils zwei Kugeln berechnet und mit der Summe der Radien verglichen werden. Der Rechenaufwand entspricht hier ebenfalls $\mathcal{O}(n^2)$. Als Optimierung wird nun das simulierte System in viele Zellen, also quaderförmige Bereiche, aufgeteilt. Während der Simulation werden die Zellen gespeichert, die eine Kugel momentan belegt. Anschließend werden nur die Kollisionen von jeweils zwei Kugeln überprüft, die die gleiche(n) Zelle(n) belegen. Als Ergebnis kann eine Rechenzeit von $\mathcal{O}(n \cdot \ln n)$ festgestellt werden. Es können also bei gleichem Rechenaufwand deutlich mehr Kugeln simuliert werden.

Durch die Einteilung in Zellen und die anschließende Kollisionserkennung entstehen keine Ungenauigkeiten. Es wird ausschließlich die Effizienz des Programmes erhöht.

[5] [Fra10], S. 47; Software in der Programmiersprache C entwickelt.

4 Simulationsprogramm

4.1 Beschreibung

Das Programm *SphereSim 2.4.2* führt Simulationen von *mikroskopischen* Objekten durch und berechnet beispielsweise das Lennard-Jones-Potential bei den Atomen eines Gases. Simulationen von *makroskopischen* Objekten (Kollision von Billiardkugeln, Gravitation der Planeten im Sonnensystem) wären ebenfalls ohne Weiteres möglich und sind für spätere Softwareversionen geplant. Das Thema dieser Arbeit beinhaltet jedoch aufgrund des mathematischen Ursprungs nur die Simulation von (mikroskopischen) Gasen. Für die Simulationen werden Algorithmen verwendet, die die Effizienz und Genauigkeit des Programms steigern (siehe Kapitel 3).

4.2 Details

Wahl der Programmiersprache Die Software wurde vollständig in der *Programmiersprache C++* programmiert. Im Gegensatz zu interpretierten Programmiersprachen wie beispielsweise Java oder Python wird C++-Code in Maschinensprache übersetzt und direkt vom Prozessor ausgeführt. Dadurch können die Simulationen mit der vollen Geschwindigkeit des PCs durchgeführt werden. Die Sprache C++ ermöglicht eine objektorientierte Programmierung und erlaubt dadurch eine hohe Plattformkompatibilität: Das Programm läuft unter den Betriebssystemen Windows und Linux; die Ausführung unter Mac OS X ist theoretisch auch möglich.

Das folgende Codebeispiel zeigt die Berechnung der Kräfte zwischen zwei Teilchen:

```
dVec = sphere2.pos - sphere.pos; d = dVec.norm();
if(collisionDetection)
{
        bothRadii = sphere2.radius + sphere.radius;
        if(d < bothRadii)
        {
                dNormalized = dVec/d;
                dOverlapping = bothRadii - d;
                R = 1/((1/sphere.radius)+(1/sphere2.radius));
                force = 4.0f/3.0f*simulatedSystem.E_sphere_sphere*sqrt(R*
                    dOverlapping*dOverlapping*dOverlapping);
                forceVec -= force*dNormalized;
        }
}
```

```
if(gravity)
        force += simulatedSystem.gravitationalConstant * sphere.mass *
            sphere2.mass * dVec / d / d / d;
if(lennardJonesPotential)
{
        force -= 48*simulatedSystem.lenJonPotEpsilon/(simulatedSystem.
            lenJonPotSigma*simulatedSystem.lenJonPotSigma)
                *dVec*(pow(simulatedSystem.lenJonPotSigma/d, 14)-0.5*pow(
                    simulatedSystem.lenJonPotSigma/d, 8));
}
```

Das zweite Codebeispiel zeigt die Berechnung der Kraft, die auf ein Teilchen aufgrund der Erdanziehung und der Kollision mit den Wänden wirkt:

```
force = simulatedSystem.earthGravity*sphere.mass;
for(quint8 dim = 0; dim<3; dim++)
{
        if((d = (sphere.radius - sphere.pos(dim))) > 0)
        {
                forceNormalized = 4.0/3.0*simulatedSystem.E_sphere_wall*
                    sqrt(sphere.radius*d*d*d);
                force(dim) += forceNormalized;
        }
        if((d = (sphere.radius + sphere.pos(dim) - simulatedSystem.
            boxSize(dim))) > 0)
        {
                forceNormalized = 4.0/3.0*simulatedSystem.E_sphere_wall*
                    sqrt(sphere.radius*d*d*d);
                force(dim) -= forceNormalized;
        }
}
```

Interner Programmaufbau Das Programm besteht aus über 7 000 Zeilen Quellcode in 38 Klassen, die die Simulation und Anzeige der Teilchen umsetzen. Dabei ist das Programm in Server und Clients aufgeteilt: Der *Server* verwaltet alle Simulationsaufgaben und hat keine graphische Oberfläche. Die *Clients* verbinden sich über eine Netzwerkverbindung mit dem Server und starten die Simulationen.[1] Der Client *Viewer* zeigt beispielsweise das simulierte System in einer dreidimensionalen animierten Ansicht an. Der Client *Grapher* führt dagegen Simulationen durch, um anschließend Ergebnisse in Diagrammen darzustellen (siehe Kapitel 4.4).

Diese Aufteilung des Programms hat mehrere Vorteile: Zum Beispiel kann der Server auf einem leistungsfähigen PC ohne Bildschirm gestartet und der Client auf einem anderen PC mit Bildschirm ausgeführt werden. Die Netzwerkkommunikation verbindet beide Programmteile

[1] Eine Internetverbindung wird nicht benötigt. Die Netzwerkkommunikation läuft ausschließlich auf dem PC ab.

miteinander. Außerdem können beliebige Clients auf den Server zugreifen. So könnten mehrere Clients unterschiedliche (wie mikroskopische und makroskopische) Simulationen verwalten, während die Bedienung der einzelnen Programme übersichtlich bleibt. Beispielsweise wurde auch ein Client *Tester* programmiert, um die volle Funktionalität des Servers zu überprüfen.

Bibliotheken Für das Programm wird die Bibliothek *Qt* in Version 5.1.1 benutzt. Diese stellt Funktionen zur Anzeige von Programmoberflächen und zur Netzwerkkommunikation bereit.

Für die dreidimensionale Anzeige wird die *OpenGL*-Bibliothek verwendet. Mit dieser lassen sich die simulierten Teilchen anschaulich darstellen und animieren.

Die Bibliothek *Eigen* stellt Funktionen bereit, die für die Berechnungen mit Vektoren benutzt werden. Sie wird in ihrer Version 3.2.0 verwendet.

Die Verwendung mehrerer Prozessorkernen beschleunigt die Berechnungen der Simulation. Deshalb wird die *OpenMP*-Bibliothek eingesetzt, die eine Ausführung des Programms auf allen verfügbaren Prozessorkernen ermöglicht.

Programmanforderungen Zur Ausführung des Programms wird ein PC mit Microsoft Windows[2] (ab Version XP) sowie eine Grafikkarte mit OpenGL ES 2.0-Unterstützung benötigt. Server und Client verwenden insgesamt weniger als 100 MB Arbeitsspeicher.

4.3 Kurzanleitung

Abb. 4.1: Startfenster des vorgestellten Programms SphereSim 2.4.2.[3]

[2]Das Programm wurde unter Linux entwickelt und eine Ausführung unter diesem Betriebssystem ist ohne Weiteres möglich. Wegen der geringen Verbreitung von Linux wird auf der CD jedoch nur die Windows-Version bereitgestellt.

Im Ordner *SphereSim* auf der beigelegten CD ist das Programm mit allen Abhängigkeiten enthalten. Der Ordner kann auf die Festplatte des PCs kopiert oder das Programm direkt von CD gestartet werden. Es muss zuerst die Datei *SphereSim_Server.exe* ausgeführt werden, um den Server zu starten. Anschließend muss *SphereSim_Viewer.exe* ausgeführt werden, um den Client mit der graphischen Oberfläche anzuzeigen. Der Client verbindet sich automatisch mit dem Server. Sollte eine Firewall nach Netzwerkzugriff für das Programm fragen, muss dieser akzeptiert werden, damit die programminterne Kommunikation funktioniert. Im Startfenster (siehe Abbildung 4.1) wird die Anzahl der Teilchen[4] eingegeben und bestätigt. Anschließend startet die Oberfläche des Hauptprogramms.

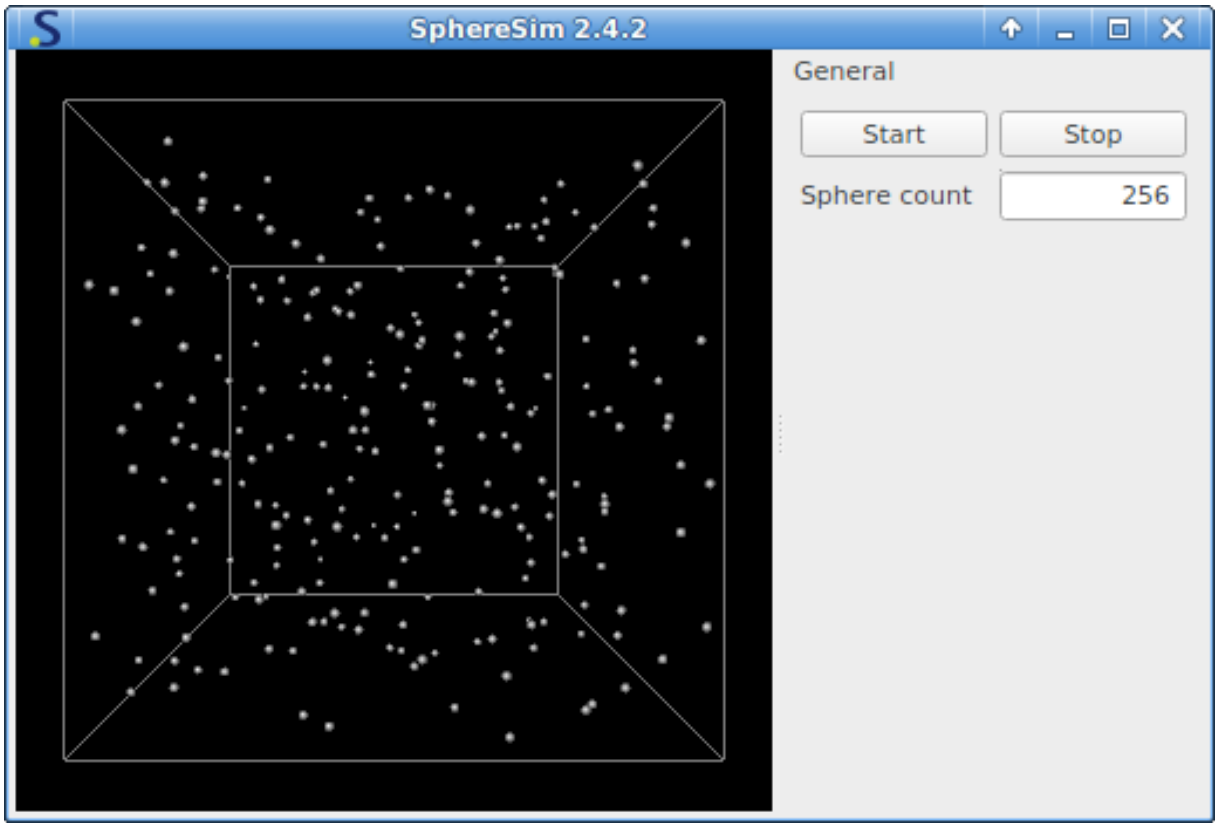

Abb. 4.2: Screenshot des vorgestellten Programms SphereSim 2.4.2.[5]

Oberfläche Die Oberfläche des Programms ist bewusst sehr übersichtlich gehalten, um nicht von der eigentlichen Simulation abzulenken (siehe Abbildung 4.2). Auf der linken Seite ist die aktuelle Simulation mit den Teilchen und dem simulierten Bereich als Quader dargestellt. Auf der rechten Seite sind Buttons zum Starten und Pausieren der Simulation vorhanden. Außerdem wird die Anzahl der Teilchen angezeigt.

[3] Eigene Quelle
[4] Damit die Teilchen gleichmäßig im Simulationsbereich platziert werden können, werden nur Quadratzahlen als Teilchenzahl simuliert.
[5] Eigene Quelle

4.4 Durchgeführte Simulations-Experimente und Auswertung

Simulation eines realen Gases In dieser Arbeit sollen anhand der effizienten Simulation eines Gases Eigenschaften der Brownsche Molekularbewegung gezeigt werden. Bei *idealen Gasen* werden nur die direkten Kollisionen von Atomen untereinander berechnet. Ein *reales Gas* bedeutet dagegen, dass auch weiter reichende Kräfte zwischen jeweils zwei Atomen (wie Gravitation) betrachtet werden. Für möglichst gut mit der Realität übereinstimmende Ergebnisse wird ein reales Gas simuliert.

Zur Simulation eignen sich Edelgase am besten, da sie keine Ladung besitzen und keine chemischen Bindungen eingehen. Besondere Kräfte wie die Coulomb-Kräfte müssen also nicht berücksichtigt werden. Als Beispiel für ein Edelgas wird in den Simulationen Argon verwendet. Hierfür sind die Masse $m = 39{,}95\,\mathrm{u} = 6{,}634 \cdot 10^{-26}\,\mathrm{kg}$ und die Werte $\epsilon = 119{,}8\,\mathrm{K} \cdot k_B = 1{,}654 \cdot 10^{-21}\,\mathrm{J}$ und $\sigma = 0{,}3405\,\mathrm{nm}$ für das Lennard-Jones-Potential bekannt[6,7]. k_B bezeichnet dabei die Boltzmann-Konstante[8] mit dem Wert $k_B = 1{,}381 \cdot 10^{-23}\,\frac{\mathrm{J}}{\mathrm{K}}$.

Andere simulierte Experimente können ohne Weiteres ebenfalls mit der programmierten Software umgesetzt werden. Das Thema dieser Arbeit umfasst jedoch nur die Brownschen Bewegung, daher wird hier exemplarisch die Simulation eines realen Gases vorgestellt.

Bei einem Atomabstand von beispielsweise $r = 10^{-8}\,\mathrm{m}$ beträgt die Kraft des Lennard-Jones-Potentials etwa $F_{LJ} = 6{,}187 \cdot 10^{-28}\,\mathrm{N}$ und die Gravitationskraft $F_G = 2{,}937 \cdot 10^{-47}\,\mathrm{N}$. Damit gilt: $F_G \approx 5 \cdot 10^{-20} \cdot F_{LJ}$. Die Gravitationskraft hat offensichtlich im Vergleich zum Lennard-Jones-Potential keinen großen Einfluss auf die Bewegung der Atome und wird deshalb in der Simulation nicht betrachtet.[9]

Aufbau des simulierten Systems Im vorgestellten Programm werden $N = 256$ Argonatome in einem würfelförmigen Raum mit periodischen Randbedingungen simuliert. Der Würfel hat eine Kantenlänge von $5\,\mathrm{nm}$. Die Atome werden mit einer zufälligen Position im Würfel und einer zufälligen Geschwindigkeit nach einer Maxwell-Boltzmann-Verteilung bei einer Temperatur $T_0 = 200\,^{\circ}\mathrm{C} = 473{,}15\,\mathrm{K}$ initialisiert. Als Schrittweite wird $h = 2 \cdot 10^{-14}\,\mathrm{s}$ gewählt, da die Atome aufgrund ihrer geringen Masse große Beschleunigungen erfahren. In den ersten 25 und dann nach jeweils 25 Simulationsschritten wird die kinetische Energie der Teilchen so verändert, dass das System ein thermodynamisches Gleichgewicht, also überall gleichen konstanten Druck und Temperatur, erreicht.[10] Die Temperatur des Systems wird berechnet mit:

$$T = \frac{E_{kin}}{\frac{1}{2} \cdot 3N \cdot k_B}. \tag{4.1}$$

[6] [Mey13], hinterer Bucheinband
[7] [Hee90], S. 50
[8] [Mey13], S. 39
[9] Eigene Berechnungen
[10] [Hee90], S. 31

Die Geschwindigkeiten der Atome werden also jeweils mit dem folgenden Faktor β skaliert, um die Temperatur T_0 zu erhalten[11]:

$$\beta = \sqrt{\frac{3N \cdot k_B \cdot T_0}{2 \cdot E_{kin}}} \tag{4.2}$$

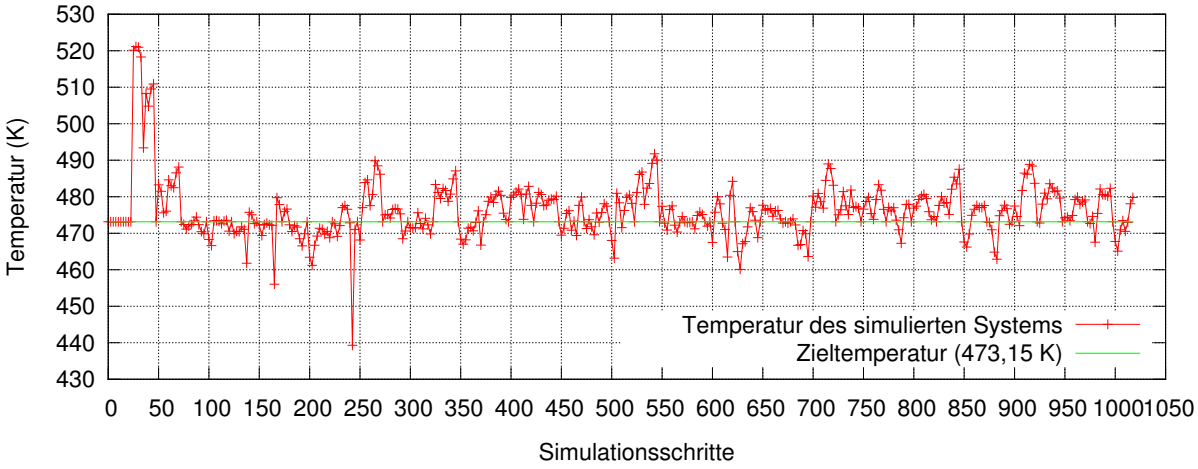

Abb. 4.3: Temperatur (in K) im Zeitverlauf der Simulation.[12]

Temperaturverlauf In Abbildung 4.3 ist die Temperatur im Verlauf der Simulation zu sehen. Die Temperaturanpassung nach jeweils 25 Simulationsschritten ist hier erkennbar. Die Schwankungen der Temperatur sind normal, da sich im simulierten System ständig Energieformen umwandeln und die kinetische Energie variiert. Gegen Ende der Simulation, nach 995 Simulationsschritten, ist mit $482\,\mathrm{K}$ die letzte größere Abweichung der Temperatur T von T_0 von etwa $1{,}8\,\%$ zu sehen. Es stellt sich also ein relativ stabiles thermodynamisches Gleichgewicht ein.

Geschwindigkeitsverteilung Bei idealen und auch bei realen Gasen im thermodynamischen Gleichgewicht entsprechen die Geschwindigkeiten der Atome einer Maxwell-Boltzmann-Verteilung[14]. Dies soll im simulierten Experiment gezeigt werden.
In Abbildung 4.4 sind die gemessene Geschwindigkeitsverteilung am Ende der Simulationszeit und eine daran angenäherte Maxwell-Boltzmann-Verteilung zu sehen. Die gemessenen Werte unterscheiden sich kaum von der genäherten Funktion, die Geschwindigkeiten der Atome liegen

[11] [Hee90], S. 37
[12] Eigene Quelle
[13] Eigene Quelle
[14] [MLM02], S. 8

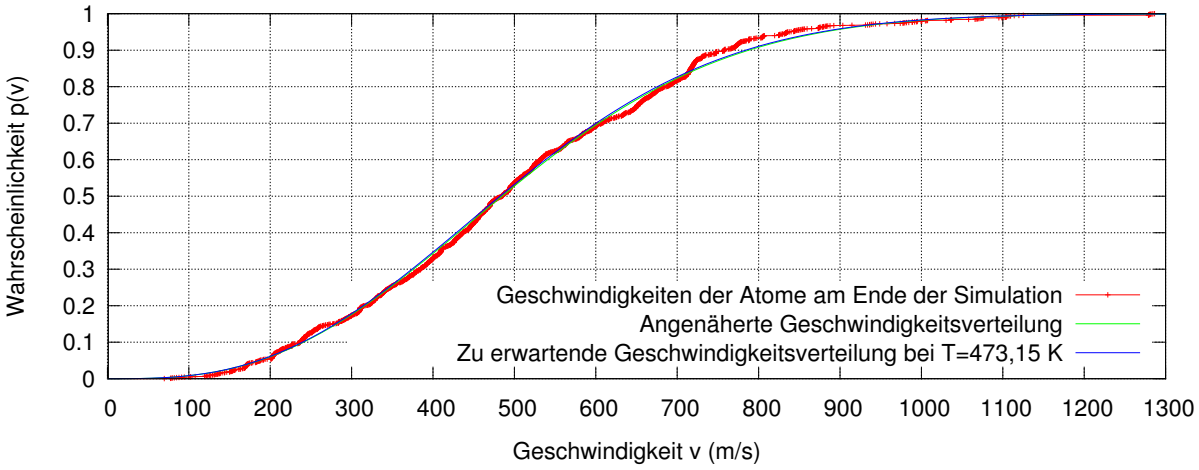

Abb. 4.4: Theoretische Verteilung $p(v)$ (blau) der Geschwindigkeiten und Näherung (grün) der Simulation (rot).[13]

also in einer Maxwell-Boltzmann-Verteilung vor. Die Gleichung für die Verteilung lautet[15]:

$$p(v) = \operatorname{erf}\left(\frac{v}{\sqrt{2}a}\right) - \sqrt{\frac{2}{\pi}}\frac{v e^{-v^2/(2a^2)}}{a}$$

Dabei ist v die Geschwindigkeit und $\operatorname{erf}(x)$ die Fehlerfunktion. Die Konstante a beträgt für Gase:

$$a = \sqrt{\frac{k_B \cdot T}{m}}$$

Für die genäherte Funktion wurde ein Wert von $a \approx 315{,}1\,\frac{\mathrm{m}}{\mathrm{s}}$ bestimmt. Die Temperatur für die genäherte Funktion beträgt also:

$$T = \frac{a^2 \cdot m}{k_B} \approx 477\,\mathrm{K}$$

Die zu den Geschwindigkeiten der Atome passende Maxwell-Boltzmann-Verteilung entspricht also einer Temperatur, die etwa $4\,\%$ höher (Abweichung geringer als $1\,\%$) als deren Starttemperatur $T_0 = 473{,}15\,\mathrm{K}$ ist. Das zeigt, dass das Gas sehr realitätsnah simuliert wurde.

In Abbildung 4.5 sind die Dichtefunktionen (Ableitungen der Verteilungsfunktion) von der gemessenen und der zu erwartenden Verteilung der Geschwindigkeiten dargestellt.

Die Dichtefunktion der Maxwell-Boltzmann-Verteilung lautet[17]:

$$f(v) = p'(v) = \sqrt{\frac{2}{\pi}}\frac{v^2 e^{-v^2/(2a^2)}}{a^3}$$

[15] [Wik13c]
[16] Eigene Quelle
[17] [Wik13c]

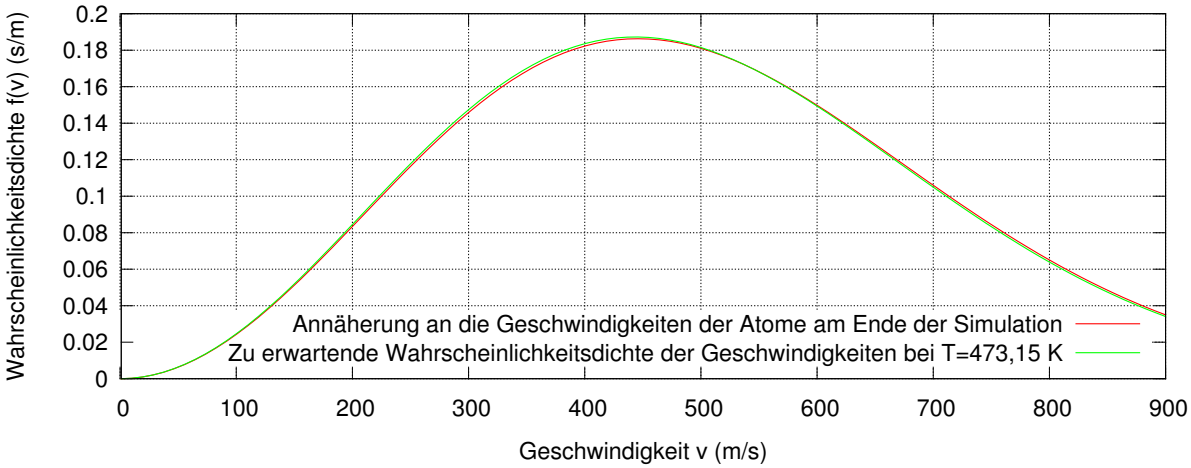

Abb. 4.5: Erwartete Dichtefunktion (grün) der Geschwindigkeiten und Näherung der Simulationsergebnisse (rot).[16]

Für die Graphen werden die bereits bestimmte Konstante a und die zu erwartende Konstante a_0 verwendet:

$$a_0 = \sqrt{\frac{k_B \cdot T_0}{m}} \approx 313{,}8\,\frac{\mathrm{m}}{\mathrm{s}}$$

Die simulierten Atome sind nur unwesentlich schneller als erwartet. Das ist auf die bereits berechnete, etwas höhere Temperatur für die genäherte Verteilung zurückzuführen.

5 Fazit

Im Laufe dieser Arbeit wurden die physikalische Brownschen Bewegung und mathematische Beschreibung des Wiener-Prozesses erarbeitet. Dabei wurde die unendliche Teilbarkeit des Wiener-Prozesses als Unterschied zwischen den beiden Modellen festgestellt. Anschließend wurden die physikalischen und mathematischen Grundlagen für Partikelsimulationen wie die Berechnung verschiedener Kräfte sowie die numerische Lösung von Differentialgleichungen in einem Programm umgesetzt. Dabei wurde besonderer Wert auf möglichst effiziente und realitätsnahe Simulationen gelegt. Die Ergebnisse der durchgeführten Simulationen wurden abschließend mit bekannten physikalischen Theorien verglichen.

Das in dieser Arbeit vorgestellte Programm kann in verschiedenen Bereichen Anwendung finden. In der Schule könnten mathematische und physikalische Experimente wie der freie Fall anschaulich dargestellt werden. In der Wissenschaft könnten simulierte theoretische Modelle mit realen Messungen und Beobachtungen verglichen werden. In der Informatik kann das Programm als Beispiel für objektorientierte Programmierung und für eine Client-Server-basierte Software mit Netzwerkkommunikation dienen. Letztendlich könnte das Programm aber auch als Physics-Engine in einem Spiel, beispielsweise in einer Billard-Simulation, verwendet werden, um physikalisch korrekte Animationen zu erzeugen. Besonders aufgrund der Aufteilung in Server und Clients ist die Software dahingehend beliebig erweiterbar.

6 Literaturverzeichnis

[Ang11] Simon Angulo, Raul und White. The Millennium-XXL Project: Simulating the Galaxy Population in Dark Energy Universes. `http://www.mpa-garching.mpg.de/mpa/research/current_research/hl2011-9/hl2011-9-en.html`, 2011. [Online; Stand 31. Oktober 2013].

[App09] David Applebaum. *Lévy Processes and Stochastic Calculus (Cambridge Studies in Advanced Mathematics)*. Cambridge University Press, 2009.

[Fra10] Margaretha Franck. Effiziente numerische Simulation von Teilchensystemen (Bachelorarbeit). `http://www.informatik.uni-kiel.de/fileadmin/arbeitsgruppen/scientific_comput/Absolventen/Bachelor_Franck_Margaretha.pdf`, 2010. [Online; Stand 6. November 2013].

[Gil91] Daniel T. Gillespie. *Markov Processes: An Introduction for Physical Scientists*. Academic Press, 1991.

[GP07] Thomas Geike and Valentin L. Popov. Mapping of three-dimensional contact problems into one dimension. *Phys. Rev. E*, 76:036710, Sep 2007.

[Hee90] Dieter W. Heermann. *Computer Simulation Methods in Theoretical Physics*. Springer, 1990.

[Mey01] Peter Meyberg, Kurt und Vachenauer. *Höhere Mathematik 2: Differentialgleichungen, Funktionentheorie, Fourier-Analysis, Variationsrechnung (Springer-Lehrbuch) (German Edition)*. Springer, 2001.

[Mey13] Prof. Dr. habil. Lothar Meyer. *Naturwissenschaftliche Formelsammlung für die bayerischen Gymnasien, 2. Fassung*. Bayerisches Staatsministerium für Unterricht und Kultus, Cornelsen Schulverlage, 2013.

[MLM02] Pirooz Mohazzabi, Shannon L. Helvey, and Jeremy McCumber. Maxwellian distribution in non-classical regime. *Physica A: Statistical Mechanics and its Applications*, 316(1):314–322, 2002.

[Ree83] W. T. Reeves. Particle Systems—a Technique for Modeling a Class of Fuzzy Objects. *ACM Transactions on Graphics*, 2(2):91–108, April 1983.

[Sch94] Schroeder, Manfred. Aus dem Amerikan. übers. von Jürgen Brau. *Fraktale, Chaos und Selbstähnlichkeit: Notizen aus dem Paradies der Unendlichkeit*. Verständliche Wissenschaft. Spektrum, Akad. Verl., Heidelberg [u.a.], 1994.

[Sch11] Prof. Dr. Volker Schmidt. *Stochastik II. Vorlesungsskript Wintersemester 2011/12*. Universität Ulm, Institut für Stochastik, Ulm, November 2011.

[vS06] M. von Smoluchowski. Zur kinetischen Theorie der Brownschen Molekularbewegung und der Suspensionen. *Annalen der Physik*, 326(14):756–780, 1906.

[Wik13a] Wikipedia. Barnes-Hut-Algorithmus — Wikipedia, Die freie Enzyklopädie, 2013. [Online; Stand 9. November 2013].

[Wik13b] Wikipedia. Brownsche Bewegung — Wikipedia, Die freie Enzyklopädie. `http://de.wikipedia.org/w/index.php?title=Brownsche_Bewegung&oldid=118814767`, 2013. [Online; Stand 29. Juni 2013].

[Wik13c] Wikipedia. Maxwell-Boltzmann-Verteilung — Wikipedia, Die freie Enzyklopädie. `http://de.wikipedia.org/w/index.php?title=Maxwell-Boltzmann-Verteilung&oldid=119855900`, 2013. [Online; Stand 1. Juli 2013].

[Wik13d] Wikipedia. Runge–Kutta methods — Wikipedia, Die freie Enzyklopädie, 2013. [Online; Stand 9. November 2013].

[Wol10] Philip Wolff. Mathematik des Schlangestehens - „Beim Warten sind wir wie Moleküle“ - Wissen - Süddeutsche.de. `http://sz.de/1.579604`, 2010. [Online; Stand 9. November 2013].

7 Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Ort, Datum	Max Bastian Mertens