# Einfache Modelle zur Strukturbildung im Universum

Masterarbeit im Fach

**Physik**

zur Erlangung des akademischen Grades:

**Master of Education (M.Ed.)**

eingereicht am

**01.10.2026**

von

**Martin Reitenbach**

geboren in Heilbronn 2001

Ruprecht-Karls-Universität Heidelberg
Fakultät für Physik und Astronomie
Haus der Astronomie

**Zusammenfassung**

Die vorliegende Abschlussarbeit befasst sich mit dem astronomischen Thema „Dunkle-Materie-Halos“ als Bestandteil von Strukturen im Universum auf Grundlage von einfachen Modellen. Im ersten Teil der Arbeit sollen mithilfe von N-Körper-Simulationen für eine verschiedene Anzahl von $N$ Teilchen ($N = 50; 100; 200$ und 2000) und jeweils drei unterschiedlich gewählten *softening*-Werten $\epsilon$, durchgeführt auf dem Hochleistungsrechner *bwUniCluster*, der Kollaps von Dunkle-Materie-Halos im Verlauf der Simulationsdauer studiert werden. Miteinhergehend sollen auch die während des Kollaps auftretende *Virialisierung* eines Halos, die durch das Virialtheorem aus der theoretischen Physik beschrieben werden kann, sowie die Meta- und Instabilität von Dunkle-Materie-Halos als selbst-gravitierende Systeme thematisiert werden. Im zweiten Teil der Arbeit wird in die Dunkle-Materie-Halos baryonische Materie in Form von Gas miteinbezogen, aus denen sich später weitere Strukturen, nämlich Sterne, gebildet haben. Dabei soll mithilfe des NFW-Dichteprofils für dunkle Materie und des $\beta$-Profils für Gase ein radiales Temperaturprofil hergeleitet werden, durch das sich das Gas im hydrostatischen Gleichgewicht mit der Gravitation durch die dunkle Materie befindet. Darauf aufbauend wird zum Abschluss die entsprechende Kühlzeit in Abhängigkeit verschiedener Halomassen und des radialen Dichteprofils hergeleitet und dargestellt.

# Inhaltsverzeichnis

**1 Einleitung** **1**

**2 Dunkle-Materie-Halos** **3**

2.1 Das Modell vom sphärischen Kollaps . . . . . 3

2.2 Virialisierung von Halos . . . . . 8

2.2.1 Über das Virialtheorem zum Virialradius . . . . . 9

2.2.2 Der Dichtekontrast nach der Virialisierung . . . . . 14

2.3 N-Körper-Simulationen zum Kollaps von Dunkle-Materie-Halos . . . . 15

2.3.1 Numerische Stabilität . . . . . 16

2.3.2 Numerische Integrationsmethoden . . . . . 17

2.3.3 Force-Softening und Zeitschritte . . . . . 19

2.3.4 Verteilung der Teilchen in einer Sphäre . . . . . 23

2.3.5 Skalierung der Bewegungsgleichungen . . . . . 25

2.4 Ergebnisse der N-Körper-Simulationen . . . . . 29

2.4.1 Zeitliche Entwicklung des Halokollapses . . . . . 29

2.4.2 Virialisierte Halos nach *violent relaxation* . . . . . 39

2.4.3 Meta- und Instabilität von Dunkle-Materie-Halos . . . . . 46

**3 Kosmisches Gas in Halos** **58**

3.1 Kollaps und Stabilität von kosmischen Gas . . . . . 59

3.2 Dichteprofile . . . . . 63

3.2.1 Dunkle Materie . . . . . 63

3.2.2 Gas . . . . . 68

3.3 Hydrostatisches Gleichgewicht . . . . . 70

3.4 Temperaturprofil . . . . . 73

3.5 Kühlung von kosmischen Gas . . . . . 76

3.5.1 Kühlungsmechanismen . . . . . 76

3.5.2 Kühlzeit . . . . . 79

**4 Zusammenfassung und Fazit** **84**

**Literaturverzeichnis** **87**

**A Promptverzeichnis** **92**

A.1 Prompts für die Python-Codes zu den Abbildungen, Simulationen und Ergebnissen in Kapitel 2 . . . . . 93

A.2 Prompts für die Python-Codes zu den Abbildungen und Ergebnissen in Kapitel 3 . . . . . 103

**B Python-Codes** **114**

B.1 Python-Codes zu den Abbildungen, Simulationen und Ergebnissen in Kapitel 2 . . . . . 114

B.2 Python-Codes zu den Abbildungen und Ergebnissen in Kapitel 3 . . . 136

**C Eigenständigkeitserklärung** **153**

# Abbildungsverzeichnis


2.1 Zeitlicher Verlauf in Einheit der Kollapszeit $t_c$ von der Expansion bis zum Kollaps einer überdichten Sphäre. . . . . 5

2.2 Die dynamische Reibung mit zeitlicher Entwicklung von oben nach unten qualitativ dargestellt für ein durchquerendes Teilchen (hier in rot), das sich mit einer Geschwindigkeit in horizontaler Richtung von links nach rechts bewegt, während es gleichzeitig aufgrund seiner Anziehungskraft die (gelben) Teilchen ober- und unterhalb anzieht. Die lokal erzeugte, höhere Dichte bremst die Beschleunigung des durchquerenden Teilchens ab. Bilder sind aus [22] entnommen. . . . . 9

2.3 Vergleich der beiden Integrationsverfahren im zeitlichen Verlauf am Beispiel des eindimensionalen harmonischen Oszillators; entnommen aus [10]. . . . . 19

2.4 Qualitativer Vergleich zwischen dem Newtonschen Gravitationsgesetz mit und ohne dem softening-Parameter (hier exemplarisch mit $\epsilon = 0.25$) 21

2.5 Schematische Darstellung eines 2D-Paramterfelds mit dem Polarwinkel $+1 \geq cos\vartheta \geq -1$ auf der y-Achse und dem Azimutwinkel $2\pi \geq \varphi \geq 0$ auf der x-Achse. Abbildung selbst erstellt. . . . . 24

2.6 Qualitativer Vergleich der Teilchenverteilung innerhalb einer Sphäre mit $R_{max} = 50$ für exemplarisch $N = 500$ Teilchen mit Farbkodierung: Im linken Bild ist die Teilchenanzahl in jeder Kugelschale konstant ($\rho_{Kugel}$ ist damit inhomogen) und im rechten Bild nicht konstant ($\rho_{Kugel}$ ist damit homogen). . . . . 25

2.7 Ergebnisse der durchgeführten Simulationen für $N = 50$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 11,0$, für das mittlere $\epsilon_{max}/2 = 5,5$ und für das untere $\epsilon_{max}/4 = 2,75$. . . . . 31

2.8 Ergebnisse der durchgeführten Simulationen für $N = 100$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 8,7$, für das mittlere $\epsilon_{max}/2 = 4,35$ und für das untere $\epsilon_{max}/4 = 2,18$. . . . . 32

2.9 Ergebnisse der durchgeführten Simulationen für $N = 200$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 7,0$, für das mittlere $\epsilon_{max}/2 = 3,5$ und für das untere $\epsilon_{max}/4 = 1,75$. . . . . 34
2.10 Ergebnisse der durchgeführten Simulationen für $N = 2000$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 3,2$, für das mittlere $\epsilon_{max}/2 = 1,6$ und für das untere $\epsilon_{max}/4 = 0,8$. . . . . 35
2.11 Darstellung der gegen die Zeit $t/t_c$ aufgetragene Virialsumme für $N = 50$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 11,0$, für das mittlere $\epsilon_{max}/2 = 5,5$ und für das untere $\epsilon_{max}/4 = 2,75$. . . . 41
2.12 Darstellung der gegen die Zeit $t/t_c$ aufgetragene Virialsumme für $N = 100$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 8,7$, für das mittlere $\epsilon_{max}/2 = 4,35$ und für das untere $\epsilon_{max}/4 = 2,18$. . . 43
2.13 Darstellung der gegen die Zeit $t/t_c$ aufgetragene Virialsumme für $N = 200$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 7,0$, für das mittlere $\epsilon_{max}/2 = 3,5$ und für das untere $\epsilon_{max}/4 = 1,75$. . . . 44
2.14 Darstellung der gegen die Zeit $t/t_c$ aufgetragene Virialsumme für $N = 2000$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 3,2$, für das mittlere $\epsilon_{max}/2 = 1,6$ und für das untere $\epsilon_{max}/4 = 0,8$. . . . . 45
2.15 Vier Momentaufnahmen (engl. *snapshots*) zu verschiedenen Zeitpunkten ($t/t_c = 0; 0.9; 1.0$ und $2.5$ von l. o. nach r. u.) eines mit $N = 4096$ Teilchen simulierten Halo-Kollapses; entnommen aus [21]. . . . . 48
2.16 Darstellung der Entfernungsverteilung für $N = 50$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 11,0$, für das mittlere $\epsilon_{max}/2 = 5,5$ und für das untere $\epsilon_{max}/4 = 2,75$. . . . . 51
2.17 Darstellung der Entfernungsverteilung für $N = 100$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 8,7$, für das mittlere $\epsilon_{max}/2 = 4,35$ und für das untere $\epsilon_{max}/4 = 2,18$. . . . . 53
2.18 Darstellung der Entfernungsverteilung für $N = 200$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 7,0$, für das mittlere $\epsilon_{max}/2 = 3,5$ und für das untere $\epsilon_{max}/4 = 1,75$. . . . . 54
2.19 Darstellung der Entfernungsverteilung für $N = 2000$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 3,2$, für das mittlere $\epsilon_{max}/2 = 1,6$ und für das untere $\epsilon_{max}/4 = 0,8$. . . . . 55

3.1 Vergleich zwischen der Dichteverteilung von dunkler Materie und Gas; entnommen von [20]. . . . . 58

3.2 Doppelt-logarithmische Darstellung des NFW-Profils: Auf der y-Achse ist die Dichte $\rho$ aufgetragen und auf die Skalendichte $\rho_s$ normiert. Auf der x-Achse ist der Radius $r$ aufgetragen und auf den Skalenradius $r_s$ normiert. Zur Orientierung ist der Wert $x_s = r_s/r_s = 1$ mit einer rot gestrichelten Linie markiert. . . . . 64
3.3 Konzentrationsparamater $c$ gegen die Halomasse [in Einheiten von $M_\odot$] in logarithmischer Darstellung für unterschiedliche Rotverschiebungen $z$ aufgetragen; entnommen von [14]. . . . . 67
3.4 Doppelt-logarithmische Darstellung des $\beta$-Profil mit $\beta = 2/3$. Auf der y-Achse ist die Dichte auf $\rho_0$ normiert sowie auf der x-Achse der Radius auf den Kernradius $r_c$ mit $x_c := r/r_c$. . . . . 69
3.5 Schematische Skizze eines würfelförmigen Gaspäckchens mit Kantenlänge $z$ mit den von oben und unten einwirkenden Druckkräften; entnommen von [31]. . . . . 71
3.6 Veranschaulichung der Gravitationskraft gegen die Druckkraft unter Annahme eines isothermen Gases in doppelt-logarithmischer Darstellung. . . . . 73
3.7 Dimensionsloses Temperaturprofil $\widetilde{T}(x)$ des Gases in doppelt-logarithmischer Darstellung, um hydrostatisches Gleichgewicht zu erreichen. . . . . 75
3.8 Veranschaulichung der Gravitationskraft gegen die Druckkraft mit dem in Abb. 3.7 ermittelten Temperaturprofil in doppelt-logarithmischer Darstellung. . . . . 76
3.9 Schematische Darstellung aller vier Kühlungsmechanismen, die sich auf atomarer Ebene abspielen; entnommen und leicht adaptiert von [35]. . . . . 77
3.10 Kühlfunktionen im Temperaturbereich von $T = 10^4$ bis $10^8$ für verschiedene Metallizitäten $Z$; entnommen von [35]. Die Daten selbst zur Erstellung dieser Kühlfunktionen sind bei Sutherland et al. zu finden [39]. . . . . 79
3.11 Kühlzeiten für Halos verschiedener Massen mit Farbkodierung sowohl für die Kühlzeit von $10^4$ bis $10^{14}$ Jahren als auch für das radiale Profil der Halos. . . . . 82
3.12 Kühlzeiten für Halos verschiedener Massen mit Farbkodierung für die Kühlzeit von $T = 10^4$ bis $10^{16}$ Jahre und unter Annahme einer konstanten, mittleren Dichte des Gases; entnommen und leicht adaptiert von [35]. . . . . 83

# Tabellenverzeichnis


2.1 Übersicht über die in den Simulationen verwendeten softening-Werte in Abhängigkeit der Teilchenanzahl. Werte wurden aufgerundet. In der Simulation ist $R_{vir} = 50/2 = 25$ Codeeinheiten gewählt. . . . . 22

2.2 Freifallzeiten $t_{ff}$ [in Gyr] für verschiedene (exemplarisch gewählte) Massen $M$ und Radien $R$ von Halos. . . . . 29

2.3 Übersicht zu den verschiedenen Zeiten $t_2$ in Abhängigkeit von unterschiedlichen Teilchenzahlen $N$. . . . . 49

2.4 Übersicht der berechneten Werte für die Relaxationszeit $t_2$ [in Gyr] für verschiedene Halomassen und -radien in Abhängigkeit der simulierten Teilchenzahl. . . . . 50

2.5 Übersicht über die räumliche Verteilung der Teilchen bei $N(< r)/N_{gesamt} = 0.4$ zu drei verschiedenen Zeitpunkten und unterschiedlicher Teilchenanzahl $N$ (bei $\epsilon_{max}/4$). . . . . 56

3.1 Übersicht über die berechneten Virialradien $r_{vir}$ [in kpc] in Abhängigkeit der Halomasse [in $M_\odot$] verschiedener Größenordnungen. . . . . 67

3.2 Übersicht über die berechneten Skalenradien $r_s$ über die Beziehung $c = r_{vir}/r_s$ in Abhängigkeit des für jede Halomasse $M$ bestimmten Konzentrationsparameters $c$ sowie die entsprechenden Skalendichten $\rho_s$ nach Gl. (3.36). . . . . 68

3.3 Übersicht über die berechneten Kernradien $r_c$ [in kpc]. . . . . 70

3.4 Übersicht die berechneten Temperaturwerte in Abhängigkeit der Halomasse. . . . . 75

3.5 Übersicht über die berechneten Werte der Dichte $\rho_0$ sowie der Anzahldichte $n_0$. . . . . 81

# Kapitel 1

# Einleitung

Wir schreiben das Jahr 1692: Der Universalgelehrte Sir Isaac Newton befindet sich in einem regem Briefwechsel mit dem Philologen und Theologen Richard Bentley. Sie diskutierten, ob Sterne, die in einem endlichen Universum eingebettet sind, nicht aufgrund ihrer gegenseitigen Anziehungskraft in einem Punkt zusammenfallen müssten, wohingegen Sterne in einem unendlichen Universum sich gar nicht bewegen und im Raum still stehen würden [19]. Für Newton ist die erste Annahme leicht mit einem Ja zu beantworten. Nähme man ein Universum von sphärischer Gestalt an und würde eine gewisse Anzahl von Sternen im Raum anordnen, würden die Sterne nahe der „Grenze“ der Sphäre eine zum Zentrum gerichtete Anziehungskraft spüren, verursacht durch die in der Nähe des Zentrums sitzenden Sternen. Außerhalb der Grenze gibt es keine Masse, die eine Anziehungskraft ausüben könnte, da die Annahme für einen endlichen Raum gilt. Folglich würden alle Sterne sich in Richtung Zentrum bewegen und dort kollabieren. Für Bentleys Annahme eines unendlichen Universums argumentierte Newton, dass sich alle Sterne untereinander in perfekt gleich großen Abständen zueinander befinden müssten, damit sich ihre gegenseitigen Anziehungskräfte kompensieren. Doch Newton war sich nicht sicher, ob diese perfekte Balance wirklich möglich ist und beschreibt es als einen instabilen Zustand [19]:

> (...) but that there should be a Central particle so accurately placed in ye middle as to be always equally attracted on all sides & thereby continue without motion, seems to me a suppositon fully as hard as to make ye sharpest needle stand upright on its point upon a looking glass. (...)

Der Briefwechsel zeigt, dass das Verständnis über das Universum und der ständig wirkenden Gravitation vor mehr als 300 Jahren noch nicht ganz verstanden waren. Die vorhin beschriebene Instabilität und geringe Wahrscheinlichkeit von regelmäßig

angeordneten Sternen, führt zur Annahme, dass die Verteilung der Sterne untereinander doch nicht gleichmäßig im Raum ist. Diese Ungleichmäßigkeit lässt sich auch auf das Inflatonfeld, dessen Inhomogenitäten durch Quantenfluktuationen verursacht wird, übertragen [35]. Dabei ist die Dichte dieses Feld nicht gleichmäßig, sodass es gewisse Regionen gibt, von denen ein Teil eine höhere Dichte („überdicht“) aufweist als solche mit geringerer Dichte („unterdicht“). Diese Inhomogenitäten wachsen mit der Zeit an, sodass die überdichten Regionen eine Gravitationskraft auf andere Regionen ausüben und damit weitere Materie anziehen können [35]. Eine der größten, bekannten Strukturen im Universum sind sogenannte *Halos*, die überwiegend aus dunkler Materie bestehen und Galaxien wie eine unsichtbare Blase umgeben, und daher im Folgenden als *Dunkle-Materie-Halos* betitelt werden. Halos entstehen nach einem Kollaps, wenn eine Region so dicht ist, dass sie sich von der geringeren Hintergrunddichte abkoppelt und aufgrund der Gravitation kollabiert. Halos finden sich in überdichten Knotenpunkten, an denen fadenförmige Filamente zusammenlaufen, in einem schwammartigen Netzwerk wieder [35].
Die genaue zeitliche Entwicklung von der Ausdehnung bis zum Kollaps solcher Halos ist das Hauptthema dieser Abschlussarbeit. Da die Forschung rund um Halos sehr kompliziert und vom Verständnis her nicht für jeden Leser zugänglich ist, greift diese Arbeit auf einfache Modelle zurück, wie beispielsweise auf das Modell vom sphärischen Kollaps im ersten Teil. Mithilfe von N-Körper-Simulationen soll der Kollaps eines einzelnen Halos simuliert werden und seine zeitliche Entwicklung näher untersucht werden. Dabei wird auch das Virialtheorem aufgegriffen, um den Prozess der *Virialisierung*, die Halos beim Kollaps durchlaufen, zu beschreiben, sowie die Meta- und Instabilität von Dunkle-Materie-Halos als selbst-gravitierende Systeme. Im zweiten Teil wird das Modell eines Dunkle-Materie-Halos erweitert, indem nun sichtbares Gas miteinbezogen wird, da sich aus diesem weitere sichtbare Strukturen, nämlich Galaxien und Sterne entwickelt haben. Dafür werden zuerst die Stabilität von Gaswolken durch das *Jeans-Kriterium* und die theoretische Beschreibung der Dichteprofile sowohl der dunklen Materie als auch des Gases näher behandelt. Darauf aufbauend wird unter der Annahme des hydrostatischen Gleichgewichts zwischen der Druckkraft des Gases und der Gravitationskraft ein radiales Temperaturprofil hergeleitet. Anschließend kann mit diesem Temperaturprofil die Kühlzeit des Gases in Abhängigkeit verschiedener Massen von Halos und ihres radialen Profils dargestellt werden.

# Kapitel 2

# Dunkle-Materie-Halos

## 2.1 Das Modell vom sphärischen Kollaps

Das Modell vom sphärischen Kollaps ist ein vereinfachtes Modell, die die Dynamik einer sphärischen Region, die überwiegend dunkle Materie beinhaltet und gegenüber einem homogenen, expandieren Hintergrund des Universums dichter ist [5]. Diese überdichten Regionen, die sogenannten Dunkle-Materie-Halos, werden bis zu einem maximalen Radius zuerst expandieren und dann kollabieren [5]. Für die folgende Beschreibung gehen wir von einem rein materie-dominierten (ohne Einfluss der dunklen Energie heute) und flachen Universum ($\Omega_{tot} = 1$) aus. Die Annahme eines Universums mit reiner Materie ist insofern gerechtfertigt, da sich die meisten Strukturen in dieser Phase zwischen der Rekombinationsphase bei $z \approx 140$ und $z = 5,5$ gebildet haben. Die Dunkle Energie hatte zu dieser Zeit einen verschwindend geringen Anteil von etwa 1/1300 an der gesamten Materiedichte. [35].
Wir betrachten nun ein Punktteilchen der Masse $m$ auf der Oberfläche einer überdichten Sphäre. Die einzige Kraft, die im System wirkt, ergibt sich durch die Masse $M$ der Sphäre über das Newton'sche Gravitationsgesetz. Das Punktteilchen erfährt demnach eine Beschleunigung, beschrieben durch:

$$\ddot{r} = -\frac{GM}{r^2} \tag{2.1}$$

Ausgehend von Gl. (2.1) lässt sich eine konstante, spezifische Energie $e = E/m$ herleiten, die unabhängig von der Masse $m$ des Punktteilchens ist [35].

$$e = \frac{1}{2}v_r^2 - \frac{GM}{r} \tag{2.2}$$

$v_r$ bezeichnet die radiale Geschwindigkeit des Punktteilchens. Wenn die Gesamtenergie $e \geq 0$ ist, dann kann ein Punktteilchen das gebundene System verlassen,

weil seine kinetische Energie die potentielle Energie (oder auch Bindungsenergie genannt) übersteigt. Für den Fall $e = 0$ lässt sich die Mindestgeschwindigkeit (oder auch Fluchtgeschwindigkeit genannt) ermitteln, die ein Punktteilchen braucht, um das System zu entfliehen. Unser Interesse gilt aber für den Fall, dass die Expansion der Sphäre zu einem bestimmten Zeitpunkt und einer bestimmten Ausdehnung zum Stillstand kommt ($E_{kin} = 0$) und in sich kollabiert. Das bedeutet eben auch, dass die Gesamtenergie $e < 0$ sein muss, damit es zum Kollaps kommt. Zur Lösung der Gl. (2.1) können wir eine Parametrisierung wählen, die den Radius $r$, aber auch die Zeit $t$ in Abhängigkeit vom Winkel $\eta$ beschreibt [35]:

$$r(\eta) = \frac{R}{2}(1 - \cos\eta) \tag{2.3}$$

$$t(\eta) = \frac{t_c}{2\pi}(\eta - \sin\eta) \tag{2.4}$$

$t_c$ beschreibt die Zeit, an dem der Kollaps stattfindet (=Kollapszeit). Die Vorfaktoren $R/2$ und $t_c/2\pi$ sind deshalb so gewählt, da bei $\eta = 0$ sowohl der Radius $r$ als auch die Zeit $t$ null ergeben sollen. Die Kugel ist dann nach der ersten Ausdehnung bis zum maximalen Radius $R$ der Sphäre, genau bei $\eta = 2\pi$ zur Zeit $t = t_c$ kollabiert. Bevor sie kollabiert, erreicht sie bei $\eta = \pi$ ihre größte Ausdehnung $r(\eta = \pi) = R$ nach der Zeit $t(\eta = \pi) = t_c/2$, also genau der Hälfte der Kollapszeit $t_c$. Dann würde rein theoretisch die Expansion wieder von vorne beginnen, erneut gefolgt vom Kollaps. Abbildung 2.1 zeigt schematisch den idealisierten Verlauf einer sphärischen, überdichten Region bis zur Kollapszeit $t_c$. Mit dieser Parameterwahl lässt sich nun Gl. (2.1) auf folgendem Weg lösen [35]: Dafür schreiben wir zuerst $\dot{r}$ in den Differentialoperator um und „erweitern“ mit $d\eta/d\eta$:

$$\dot{r} = \frac{dr}{dt} = \frac{dr}{d\eta}\frac{d\eta}{dt} = \frac{dr}{d\eta}\left(\frac{dt}{d\eta}\right)^{-1} \tag{2.5}$$

Sowohl $dr/d\eta$ als auch $(dt/d\eta)^{-1}$ lassen sich mithilfe der obigen Funktionen $r(\eta)$ und $t(\eta)$ ermitteln, sodass folgt:

$$\frac{dr}{d\eta} = \frac{R}{2}\sin\eta \quad \text{und} \quad \frac{dt}{d\eta} = \frac{t_c}{2\pi}(1 - \cos\eta) \tag{2.6}$$

Damit ergibt $\dot{r}$:

$$\dot{r} = \frac{dr}{d\eta}\left(\frac{dt}{d\eta}\right)^{-1} = \frac{\pi R}{t_c}\frac{\sin\eta}{(1 - \cos\eta)} \tag{2.7}$$

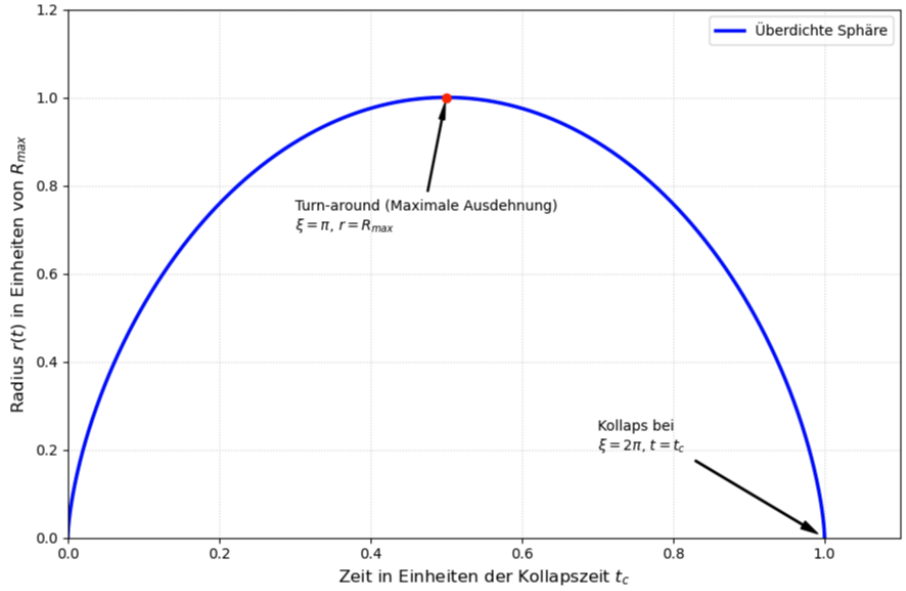



Abbildung 2.1: Zeitlicher Verlauf in Einheit der Kollapszeit $t_c$ von der Expansion bis zum Kollaps einer überdichten Sphäre.

Nun gehen wir einen Schritt weiter und schreiben $\ddot{r}$ um zu [35]:

$$\ddot{r} = \frac{d^2r}{dt^2} = \underbrace{\frac{d}{dt}}_{\text{mit } \frac{d\eta}{d\eta} \text{ erweitern}} \left(\frac{dr}{dt}\right) = \frac{d\eta}{dt}\frac{d}{d\eta}\left(\frac{dr}{dt}\right) = \left(\frac{dt}{d\eta}\right)^{-1}\frac{d}{d\eta}\underbrace{\left(\frac{dr}{dt}\right)}_{\rightarrow \text{Gl.(2.7)}} \tag{2.8}$$

Führt man die Ableitung nach $\eta$ aus, erhält man:

$$\frac{d}{d\eta}\left(\frac{dr}{dt}\right) = \frac{d}{d\eta}\left(\frac{\pi R}{t_c}\frac{\sin\eta}{(1-\cos\eta)}\right) = -\frac{\pi R}{t_c}\frac{1}{(1-\cos\eta)} \tag{2.9}$$

Multipliziert man nun $(dt/d\eta)^{-1}$ mit Gl. (2.9) ergibt sich damit:

$$\ddot{r} = \frac{2\pi}{t_c(1-\cos\eta)} \cdot \frac{-\pi R}{t_c}\frac{1}{(1-\cos\eta)} = -\frac{2\pi^2 R}{t_c^2}\frac{1}{(1-\cos\eta)^2} \tag{2.10}$$

Jetzt kann man Gl. (2.10) in Gl. (2.1) einsetzen und erhält:

$$-\frac{2\pi^2 R}{t_c^2}\frac{1}{(1-\cos\eta)^2} = -\frac{GM}{r^2} \underbrace{=}_{r(\eta) \text{ aus Gl.(2.3)}} -\frac{4GM}{R^2(1-\cos\eta)^2} \tag{2.11}$$

Sortieren und Kürzen der beiden Seiten der Gleichung liefert:

$$R^3 = \frac{2GM}{\pi^2}t_c^2 \tag{2.12}$$

Und da $R^3 \propto t_c^2$ gilt, ist Gl. (2.12) das dritte Kepler'sche Gesetz, wobei die Bahn des radial einfallenden Teilchens eine entartete Ellipse beschreibt.
Zum Zeitpunkt des Kollaps müsste die Dichte unendlich groß ($\rho \to \infty$) werden, wenn die ganze Masse im Zentrum der Sphäre konzentriert wäre. In der Realität wird dieser Fall aber nicht eintreten, allein schon aus dem unwahrscheinlichen Grund, dass solche überdichten Regionen perfekt sphärisch und symmetrisch aufgebaut sind. Da das Universum kontinuierlich expandiert und die Hintergrunddichte abnimmt, wollen wir nun die zeitliche Entwicklung der Dichte innerhalb der überdichten Sphäre im Vergleich zur abnehmenden Dichte des Universums genauer betrachten: Weil wir das Universum als gute Annäherung als flach ansehen können, lässt sich jede kosmische Zeit $t$ in Abhängigkeit vom Hubble-Paramter $H(t)$ folgendermaßen ausdrücken [35]:

$$t = \frac{2}{3H(t)} \tag{2.13}$$

Genauso lässt sich die kritische Dichte $\rho_c$ für ein flaches Universum mit Krümmungsfaktor $k = 0$ zu jeder kosmischen Zeit angeben [35]:

$$\rho_c(t) = \frac{3H(t)^2}{8\pi G} \tag{2.14}$$

$\rho_c = 1,9{\cdot}10^{-26}\ \mathrm{kg\,m^{-3}\,h^2}$ ist der heutige Wert [35], wobei $h = H_0/\,100\,\mathrm{km\,s^{-1}\,Mpc^{-1}}$ eine dimensionslose Zahl ist und als reduzierte Hubble-Konstante definiert wird, weil der Hubble-Parameter $H_0$ je nach Messmethode einen unterschiedlichen Wert aufweist, der zwischen 67 $\mathrm{km\,s^{-1}\,Mpc^{-1}}$ und 73 $\mathrm{km\,s^{-1}\,Mpc^{-1}}$ liegt [23]. Setzt man $H(t)$, ausgehend von Gl. (2.13), in Gl. (2.14), kann man die kritische Dichte auch, direkt von der Zeit abhängig, umschreiben [35]:

$$\rho_c(t) = \frac{1}{6\pi G t^2} \tag{2.15}$$

Da wir mit Gl. (2.3) und Gl. (2.4) den Radius und die Zeit in Abhängigkeit des Hilfsparameter $\eta$ darstellen, ergibt sich für $\rho_c(t)$

$$\rho(t) = \rho_c(t) = \frac{2\pi}{3Gt_c^2}\frac{1}{(\eta - \sin\eta)^2} \tag{2.16}$$

Die Dichte einer Sphäre $\rho_s$ ergibt:

$$\rho_s(t) = \frac{M}{\frac{4}{3}\pi r^3(t)} \tag{2.17}$$

bzw. durch $\eta$ ausgedrückt:

$$\rho_s(t) = \frac{6M}{\pi R^3} \frac{1}{(1-\cos\eta)^3} \tag{2.18}$$

Interessant ist aber nun der „Kontrast“ zwischen $\rho_c$ und $\rho_s$. Daher definiert man in der Kosmologie den sogenannten Dichtekontrast $\delta(t)$ [7]:

$$\rho_s(t) = \rho(t)(1+\delta(t)) \tag{2.19}$$

Teilt man $\rho_s(t)$ durch $\rho(t)$ und drückt Dichten durch Gl. (2.16) und Gl. (2.18) aus, ergibt sich folgendes Verhältnis:

$$\frac{\rho_s(t)}{\rho(t)} = \frac{\frac{6M}{\pi R^3}\frac{1}{(1-\cos\eta)^3}}{\frac{2\pi}{3Gt_c^2}\frac{1}{(\eta-\sin\eta)^2}} = \frac{9MGt_c^2}{\pi^2 R^3}\frac{(\eta-\sin\eta)^2}{(1-\cos\eta)^3} = 1+\delta(t) \tag{2.20}$$

Ersetzt man noch $R^3$ durch das dritte Kepler'sche Gesetz (2.12), kürzen sich sämtliche Konstanten heraus, wodurch nur noch ein Vorfaktor übrig bleibt:

$$\frac{\rho_s(t)}{\rho(t)} = \frac{9GMt_c^2}{\pi^2\frac{2GMt_c^2}{\pi^2}}\frac{(\eta-\sin\eta)^2}{(1-\cos\eta)^3} = \frac{9}{2}\frac{(\eta-\sin\eta)^2}{(1-\cos\eta)^3} = 1+\delta(t) \tag{2.21}$$

Das Ergebnis für den Dichtekontrast hängt erstaunlicherweise weder von der Masse noch von der Kollapszeit einer überdichten sphärischen Region ab, sondern nur von der Hilfsvariblen $\eta$. Für den Dichtekontrast zum Zeitpunkt der größten Ausdehnung, den wir als *turn-around*-Zeitpunkt $t_{ta}$ bezeichnen, ergibt sich nach $t_{ta} = t_c/2$, also bei $\eta = \pi$ [35]

$$\frac{9}{2}\frac{(\eta-\sin\eta)^2}{(1-\cos\eta)^3} - 1 = \frac{9\pi^2}{16} - 1 = \mathbf{4{,}55} = \delta(t) \tag{2.22}$$

Diese Herleitung des Dichtekontrasts beruht auf den Fakt, dass Halos im Inneren hochgradig nicht-lineare Dichten aufweisen [5]. Man kann aber auch im Rahmen der linearen Störungstheorie für $\eta \ll 1$ einen Dichtekonstrast $\delta_{lin}$ zum Kollapszeitpunkt $t_c$ herleiten, der bei [35]

$$\delta_{lin}(t) \approx 1,69 \tag{2.23}$$

liegt und als Schwellenwert für den Kollaps einer ideal sphärischen, überdichten Region gilt. Dieser Wert findet Anwendung im *Press-Schechter-Formalismus*, bei der es um die Verteilungsfunktion von Halos mit einer gegebenen Masse geht [5]. Für eine einfache und leicht nachvollziehbare Herleitung von $\delta_{lin}$ sei auf [35] oder auf eine sehr detaillierte Herleitung auf [5] verwiesen.

## 2.2 Virialisierung von Halos

Wie schon bereits angemerkt wurde, würde in einem idealisierten Modell die gesamte Masse in einem Punkt kollabieren, sodass die Dichte unendlich groß werden müsste. Danach würde wieder die Expansion folgen und erneut ein Kollaps. Die Realität zeigt aber einen anderen Ausgang. Da wir Galaxien oder Galaxienhaufen heute beobachten können, muss irgendein Prozess abgelaufen sein, der dafür gesorgt hat, dass der Kollaps zum Stillstand gekommen ist. Das System muss dann bei einer gewissen Ausdehnung eine Art Gleichgewichtszustand eingenommen haben. Dieser Prozess wird auch *Virialisierung* genannt, der mit dem Virialtheorem zusammenhängt [5]. In Realität besteht der Kollaps nicht aus einer kontinuierlichen, zusammenhängenden Masse, sondern eher auch einer Anordnung von vielen Teilchen mit einer bestimmten Masse und einer bestimmten räumlichen Anordnung. Alle Teilchen bilden zusammen ein annähernd kugelsymmetrisches System, in dem nur gravitative Anziehungskräfte untereinander herrschen [35]. Durchquert ein Teilchen in horizontaler Ebene eine Region, in der sich sowohl oberhalb als auch unterhalb andere Teilchen befinden, zieht das durchquerende Teilchen durch seine Anziehungskraft sowohl die oberen als auch die unteren Teilchen zu sich (s. Abb. 2.2). Die Flugbahn des Teilchen in horizontaler Richtung wird nicht beeinflusst, weil sich die Anziehungskräfte (von oben und unten) aufgrund der symmetrischen Anordnung gegenseitig kompensieren. Fliegt das Teilchen nun weiter, entsteht hinter ihm eine Ansammlung von Teilchen, dessen Region eine höhere, lokale Dichte aufweist als die Region vor dem Teilchen. Infolgedessen entsteht aufgrund der höheren Dichte eine kollektive Anziehungskraft auf das Teilchen, das seine Bewegung nun abbremst [22]. Diese Verringerung der Beschleunigung wird im Kontext von N-Körper-Bewegungen auch *dynamische Reibung* genannt [35]. Abbildung 2.2 zeigt mit zeitlichen Abstand voneinander qualitativ, wie man sich die dynamische Reibung besser vorstellen kann.

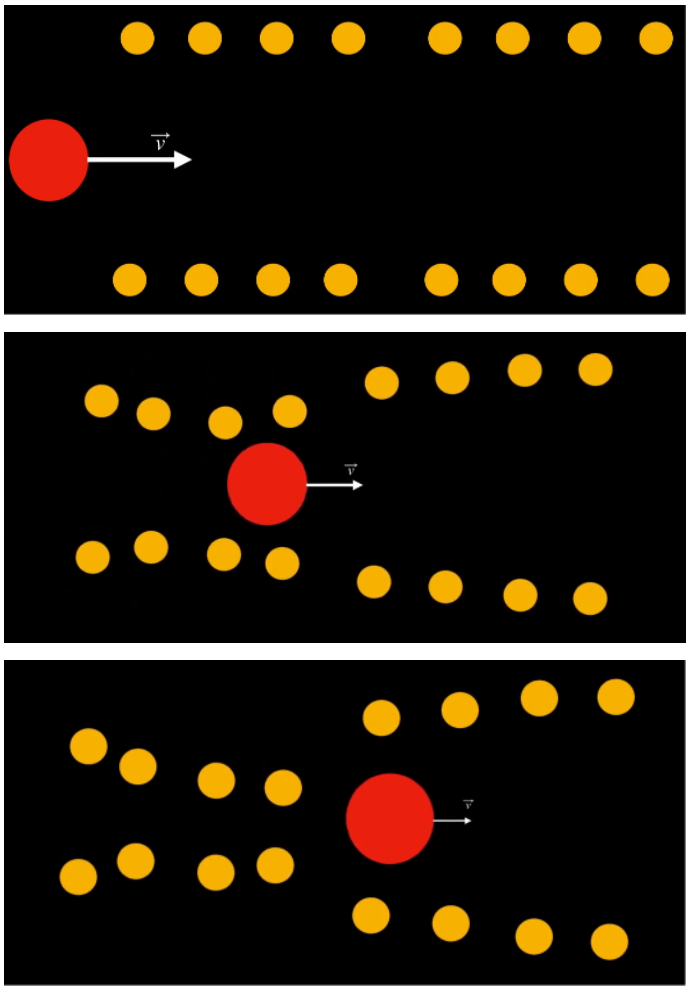


Abbildung 2.2: Die dynamische Reibung mit zeitlicher Entwicklung von oben nach unten qualitativ dargestellt für ein durchquerendes Teilchen (hier in rot), das sich mit einer Geschwindigkeit in horizontaler Richtung von links nach rechts bewegt, während es gleichzeitig aufgrund seiner Anziehungskraft die (gelben) Teilchen ober- und unterhalb anzieht. Die lokal erzeugte, höhere Dichte bremst die Beschleunigung des durchquerenden Teilchens ab. Bilder sind aus [22] entnommen.

### 2.2.1 Über das Virialtheorem zum Virialradius

Da der Prozess dieser dynamischen Reibung aber schnell unübersichtlich und mathematisch kompliziert werden kann, wollen wir im Zuge der Virialisierung anschauen, wie nun so ein chaotisches Teilchenensemble über den Kollaps hinweg in einen stabilen Zustand gelangen kann [35]. Dafür fordern wir, dass das System im zeitlichen Mittel betrachtet weder expandiert noch kollabiert. Da wir jedem Teilchen einen Ort $x$ und einen Impuls $p$ zuschreiben können, kann man das sogenannte Virial $G$ als Skalarprodukt des Ortes und des Impulses als Summe von $N$ Teilchen definieren

[35]:

$$G = \sum_{i}^{N} \vec{p_i} \cdot \vec{x_i} \tag{2.24}$$

Die zeitliche Änderung des Virials $G$ liefert:

$$\frac{dG}{dt} = \frac{d}{dt} \sum_{i}^{N} \vec{p_i} \cdot \vec{x_i} \tag{2.25}$$

Nach Anwendung der Produktregel folgt:

$$\frac{dG}{dt} = \sum_{i}^{N} \dot{\vec{p_i}} \cdot \vec{x_i} + \sum_{i}^{N} \vec{p_i} \cdot \dot{\vec{x_i}} \tag{2.26}$$

Die beiden Terme auf der rechten Seite lassen sich wie folgt umschreiben [17]:

$$\sum_{i}^{N} \dot{\vec{p_i}} \cdot \vec{x_i} = \sum_{i}^{N} \vec{F_i} \cdot \vec{x_i} \quad \text{und} \quad \sum_{i}^{N} \vec{p_i} \cdot \dot{\vec{x_i}} = \sum_{i}^{N} \vec{p_i} \cdot \vec{v_i} \tag{2.27}$$

Der Ausdruck $\sum_i^N \vec{p_i} \cdot \vec{v_i} = m \cdot |\vec{v_i}|^2 = 2T$ entspricht damit dem Doppelten der kinetischen Energie $T$ [35]. Gl. (2.26) ergibt demnach:

$$\frac{dG}{dt} = \sum_{i}^{N} \vec{F_i} \cdot \vec{x_i} + 2T \tag{2.28}$$

Jetzt berechnen wir einen zeitlichen Erwartungswert, indem wir zuerst auf beiden Seiten von Gl. (2.28) bis zu einer Zeit $\tau$ integrieren:

$$\int_0^\tau \frac{dG}{dt} = \int_0^\tau \left( \frac{d}{dt} \sum_{i}^{N} \vec{p_i} \cdot \vec{x_i} \right) dt = \int_0^\tau \left( \sum_{i} \vec{F_i} \cdot \vec{x_i} \; + 2T \right) dt \tag{2.29}$$

Den mittleren Ausdruck können wir vereinfachend umschreiben, da sich das Integral mit der Zeitableitung gegenseitig aufhebt, sodass die Virial-Funktion selbst, an den Integrationsgrenzen ausgewertet, übrig bleibt:

$$\sum_{i}^{N} \; [\vec{p_i}(\tau) \cdot \vec{x_i}(\tau)] - \sum_{i}^{N} \; [\vec{p_i}(0) \cdot \vec{x_i}(0)] = \int_0^\tau \left( \sum_{i} \vec{F_i} \cdot \vec{x_i} \; + 2T \right) dt \tag{2.30}$$

Wenn wir annehmen, dass die Teilchen zu Beginn der Simulation aus der Ruhe starten, sodass $p(0) = 0$ ist, fällt der zweite der Term auf der linken Seite von

Gl. (2.30) weg. Zum Schluss mitteln wir die beiden Seiten der Gleichung mit der Zeitdauer $\tau$ und erhalten,

$$\frac{1}{\tau}\left[p_i(\tau)\cdot x_i(\tau)\right] = \left\langle \sum_i \overrightarrow{F_i}\cdot\overrightarrow{x_i} \; + 2T \right\rangle \tag{2.31}$$

wobei ausgenutzt wurde, dass *per definitionem* für den Erwartungswert der Zeit einer beispielhaften, zeitabhängigen Größe $A(t)$ gilt [18]:

$$\langle A\rangle \equiv \lim_{t\to\infty}\frac{1}{t}\int_0^t A(t')\,dt' \tag{2.32}$$

Im Grenzfall $\lim\limits_{\tau\to\infty}$ wird die linke Seite von Gl. (2.31) null, sodass sich ergibt:

$$0 = \left\langle \sum_i \overrightarrow{F_i}\cdot\overrightarrow{x_i} \right\rangle + \langle 2T\rangle \tag{2.33}$$

Nun können wir noch die Kraft $\overrightarrow{F_i}$ näher spezifizieren [17]. Logischerweise wirken unter den Teilchen nur Anziehungskräfte gemäß dem Newton'schen Gravitationsgesetz. Die Kraft, die auf ein Teilchen $i$ von einem Teilchen $j$ wirkt, lautet

$$\overrightarrow{F_{ij}} = -\frac{Gm_im_j}{|\overrightarrow{x_i}-\overrightarrow{x_j}|^3}(\overrightarrow{x_i}-\overrightarrow{x_j}) \tag{2.34}$$

Summieren wir über alle Kräfte, die auf Teilchen $i$ wirken, auf, erhalten wir die Gesamtkraft $\overrightarrow{F_i}$:

$$\overrightarrow{F_i} = \sum_{j\neq i}\overrightarrow{F_{ij}} \tag{2.35}$$

Da wir letztendlich den Ausdruck $\sum_i \overrightarrow{F_i}\cdot\overrightarrow{x_i}$ auswerten wollen, müssen wir noch berücksichtigen, dass nach Newtons drittem Axiom gilt, dass sowohl Teilchen $i$ eine Kraft auf Teilchen $j$ ausübt und umgekehrt, also $\overrightarrow{F_{ij}} = -\overrightarrow{F_{ji}}$. Dadurch ergibt sich:

$$0 = \overrightarrow{F_{ij}}\cdot\overrightarrow{x_i} + \overrightarrow{F_{ji}}\cdot\overrightarrow{x_j} \tag{2.36}$$

$$\Rightarrow \overrightarrow{F_{ij}}\cdot\overrightarrow{x_i} - \overrightarrow{F_{ij}}\cdot\overrightarrow{x_j} = \overrightarrow{F_{ij}}\cdot(\overrightarrow{x_i}-\overrightarrow{x_j}) \tag{2.37}$$

Setzt man für $\overrightarrow{F_{ij}}$ noch Gl. (2.34) ein, erhält man:

$$\sum_i \overrightarrow{F_i}\cdot\overrightarrow{x_i} = -\sum_{i>j}\frac{Gm_im_j}{|\overrightarrow{x_i}-\overrightarrow{x_j}|^3}(\overrightarrow{x_i}-\overrightarrow{x_j})\cdot(\overrightarrow{x_i}-\overrightarrow{x_j}) \tag{2.38}$$

Unter Ausnutzung, dass $(\vec{x_i} - \vec{x_j}) \cdot (\vec{x_i} - \vec{x_j}) = |\vec{x_i} - \vec{x_j}|^2$ ergibt, folgt letztlich [17]:

$$-\frac{Gm_i m_j}{|\vec{x_i} - \vec{x_j}|} \equiv V_{ij} \tag{2.39}$$

Dieser Ausdruck ist die potentielle Energie $V_{ij}$ für ein Teilchenpaar. Summiert man noch über alle Teilchenpaare auf, erhält man die gesamte potentielle Energie $V$:

$$\sum_i \underbrace{\vec{F_i}}_{\sum_{j\neq i} \vec{F_{ij}}} \cdot \vec{x_i} = V = -\sum_{i<j} \frac{Gm_i m_j}{|\vec{x_i} - \vec{x_j}|} \tag{2.40}$$

Damit lautet Gl. (2.33) final [17]

$$\langle V \rangle + \langle 2T \rangle = 0 \tag{2.41}$$

und wodurch das Virialtheorem resultiert. Erfüllt so ein System das Virialtheorem, bedeutet das, dass es im zeitlichen Mittel stabil ist und somit weder weiter kollabiert oder noch sich weiter ausdehnt [17]. Das heißt aber nicht, dass das System statisch ist, denn die Teilchen bewegen sich weiterhin innerhalb des Gravitationspotentials [17]. Mithilfe des Virialtheorems lässt sich nun herleiten, welche Ausdehnung bzw. welchen Radius ein sphärischer Halo erreichen wird, wenn er die Virialisierung durchlaufen hat. Diesen Radius bezeichnen wir als Virialradius $R_{vir}$. Wir betrachten den Halo, wenn auch etwas unrealistisch im Vergleich zu den vorherigen Schritten, erst einmal mit dem Gravitationspotential einer Vollkugel mit einer homogenen, konstanten Dichte $\rho$ und einem Radius $R$ [35]. Die Masse hängt vom betrachteten Radius ab, da

$$M(r) = \frac{4}{3}\pi r^3 \cdot \rho \tag{2.42}$$

gilt. Das Gravitationspotential $\Phi(r)$ auf der Oberfläche der Sphäre beträgt

$$\Phi(r) = -\frac{GM(r)}{r} \tag{2.43}$$

Nun konstruiert man sich das System wie eine „Zwiebel“, die man Schale für Schale aufbaut, und immer ein Stück Masse, die aus dem Unendlichen kommt, der Kugel angefügt wird [35]. Jede Schale hat einen Oberflächeninhalt von $A(r) = 4\pi r^2$ und bei konstanter Dichte auch ein Massenelement $dm$, das sich über

$$dm = \rho\, dV_K = \rho A(r)\, dr = \rho\, 4\pi r^2\, dr \tag{2.44}$$

berechnen lässt, wobei $V_K$ das Volumen der Kugel meint. Die potentielle Energie $V$ der Kugel ergibt sich durch:

$$dV = \Phi\, dm = -\frac{GM(r)}{r}\rho\, 4\pi r^2\, dr \tag{2.45}$$

Setzt man für $M(r)$ noch Gl. (2.42) ein, erhält man:

$$dV = -\frac{16\pi^2 G}{3} r^4 \rho^2\, dr \tag{2.46}$$

Integration von $r = 0$ bis zum Gesamtradius $r = R$ ergibt [35]:

$$V = -\frac{16\pi^2 G}{3}\rho^2 \int_{r=0}^{r=R} r^4\, dr = -\frac{16\pi^2 G}{3}\rho^2 \frac{R^5}{5} \tag{2.47}$$

Nun kann man auf der rechten Seite die Größen umsortieren und ausnutzen, dass $M = \rho V_K$ ist:

$$V = -\frac{16\pi^2 G}{3}\rho^2 \frac{R^5}{5} = -\frac{3}{5}\frac{G}{R}\underbrace{\left(\frac{4\pi R^3}{3}\rho\right)^2}_{=M^2} = -\frac{3}{5}\frac{GM^2}{R} \tag{2.48}$$

Da alle Teilchen bzw. das gesamte System zum Zeitpunkt der größten Ausdehnung ruht, also $v = 0$, ist die Gesamtenergie $E_{tot}$ einfach $E_{tot} = V$. Um nun auf den Virialradius $R_{vir}$ zu kommen, schreiben wir Gl. (2.41) um zu

$$-\frac{\langle V\rangle}{2} = \langle T\rangle \tag{2.49}$$

und setzen diese Gleichung in die Gleichung für die Gesamtenergie ein:

$$E_{tot} = \langle V\rangle + \langle T\rangle = \langle V\rangle - \left\langle \frac{V}{2}\right\rangle = \left\langle \frac{V}{2}\right\rangle \tag{2.50}$$

Da wir weiterhin von der Erhaltung der Gesamtenergie vom *Turn-around*-Punkt bis zur Virialisierung ausgehen, können wir die Ausdrücke in Gl. (2.48) und (2.50) miteinander gleichsetzen, wobei wir auf der linken Seite den Virialradius mit $R_{vir}$ kennzeichnen:

$$\frac{1}{2}\left(-\frac{3}{5}\frac{GM^2}{R_{vir}}\right) = -\frac{3}{5}\frac{GM^2}{R} \tag{2.51}$$

Kürzt man beide Ausdrücke, ergibt sich das Endergebnis

$$\frac{1}{2R_{vir}} = \frac{1}{R} \longrightarrow R_{vir} = \frac{R}{2} \tag{2.52}$$

Das Ergebnis sagt demnach aus, dass die Ausdehnung eines virialisierten Halos einen Radius $R_{vir}$ einnimmt, der halb so groß ist wie der Radius $R$ bei der größten Ausdehnung. Die Realität ist deutlich komplexer. Denn die Herleitung des Virialradius' über das Virialtheorem betrachtet hier eine völlig isolierte sphärische Region, in der sowohl Masse als auch Energie erhalten bleiben. In der Realität dagegen können durch Zwei-Körper-Bewegungen Teilchen herausgeschleudert werden, wodurch es zum Massenverlust kommt (engl. *mass ejecction*) [21] oder es auch zum Verschmelzen kleinerer Halos kommt, um größere Halos zu bilden (engl. *merging*). Dies ist auch das Grundprinzip, wie sich hierarchische Strukturen bilden [24].

### 2.2.2 Der Dichtekontrast nach der Virialisierung

In der Kosmologie verwendet man für den Virialradius nicht einfach nur Gl. (2.52), sondern definiert diesen über den in Kapitel 2.1 bereits eingeführten Dichtekontrast $\delta$. Nun soll aber der Dichtekontrast eines Halos hergeleitet werden, der die Virialisierung durchlaufen hat (mehr dazu vgl. Kapitel 2.4.2). Zwei Dinge müssen dafür noch berücksichtigt werden: Zum einen gilt der Dichtekontrast $\delta(t = t_a) = \rho_s(t_{ta})/\rho(t_{ta}) = 4,55$ ($\rightarrow$ Gl. (2.22)) für eine überdichte, sphärische Region eben nur am *turn-around*-Punkt. Die Dichte des Halos nimmt aber bis zum Kollaps zu. Bildet man das Verhältnis der Dichten des Halos $\rho_s(t_{ta})$ zum *turn-around*-Zeitpunkt $t_{ta}$ und $\rho_s(t_{vir})$ zum Zeitpunkt der Virialisierung $t_{vir}$, ergibt sich [17]:

$$\frac{\rho_s(t_{vir})}{\rho_s(t_{ta})} = \frac{\frac{3M}{4\pi R_{vir}^3}}{\frac{3M}{4\pi R_{ta}^3}} = \left(\frac{R_{ta}}{R_{vir}}\right)^3 \tag{2.53}$$

Nutzt man Gl. (2.52) aus, vereinfacht sich der Ausdruck zu:

$$\frac{\rho_s(t_{vir})}{\rho_s(t_{ta})} = \left(\frac{2R_{vir}}{R_{vir}}\right)^3 = 2^3 = 8 \tag{2.54}$$

$$\rho_s(t_{vir}) = 8\rho_s(t_{ta}) \tag{2.55}$$

Die Dichte zum Zeitpunkt der Virialisierung hat sich gegenüber $\rho_s(t_{ta})$ verachtfacht. Zum anderen muss man auch noch die abnehmende Hintergrunddichte des Univer-

sums, während die Virialisierung abläuft, mitberücksichtigen. Da für die Kollapszeit $t_c = 2t_{ta}$ gilt und die Virialisierung in der Annahme nach etwa $t_{vir} \approx t_c = 2t_{ta}$ vollendet ist, lassen sich die Hintergrunddichten $\rho_c$ zum Zeitpunkt $t_{ta}$ und $t_{vir}$ als Verhältnis aufschreiben [17]:

$$\frac{\rho_c(t_{vir})}{\rho_c(t_{ta})} \underbrace{=}_{\rightarrow \text{Gl.}(2.15)} \frac{\frac{1}{6\pi G t_{vir}^2}}{\frac{1}{6\pi G t_{ta}^2}} = \left(\frac{t_a}{t_{vir}}\right)^2 \tag{2.56}$$

Nutzt man noch aus, dass $t_{vir} \approx t_c = 2t_{ta}$ gilt, dann vereinfacht sich der Ausdruck zu:

$$\frac{\rho_c(t_{vir})}{\rho_c(t_{ta})} = \left(\frac{t_a}{2t_a}\right)^2 = \frac{1}{4} \tag{2.57}$$

$$\rho_c(t_{vir}) = \frac{1}{4}\rho_c(t_{ta}) \tag{2.58}$$

Die Hintergrunddichte $\rho_c(t_{vir})$ ist demnach auf ein Viertel gegenüber $\rho_c(t_{ta})$ gesunken. Nun lässt sich final der Dichtekontrast zum Zeitpunkt der Virialisierung aufschreiben als [17]:

$$\delta_{vir} = \frac{\rho_s(t_{vir})}{\rho_c(t_{vir})} = \frac{8\rho_s(t_{ta})}{\frac{1}{4}\rho_c(t_{ta})} = 32 \cdot \underbrace{\frac{\rho_s(t_{ta})}{\rho_c(t_{ta})}}_{\rightarrow \text{Gl.}(2.22)} = 32 \cdot \frac{9\pi^2}{16} \approx \mathbf{178} \tag{2.59}$$

Das Ergebnis sagt also aus, dass ein im Einstein-de Sitter-Universum virialisierter Halo einen Dichtekontrast von etwa $\delta_{vir} = 178$ besitzt. In der Fachliteratur wird der Wert aufgrund von numerischen Simulationen oftmals auf 200 aufgerundet und als der Virialradius $R_{200} \equiv R_{vir}$ gekennzeichnet. Ein weiterer Grund für die Definition des Virialradius' ist zudem, dass Halos keine festen Grenzen zugewiesen werden können [15]. Die Dichte eines Halos ist letztendlich 200-mal größer als die Hintergrunddichte.

## 2.3 N-Körper-Simulationen zum Kollaps von Dunkle-Materie-Halos

Für die Etablierung der durchgeführten Simulationen wurde für diese Abschlussarbeit die Programmiersprache Python verwendet. Die Simulationen selbst wurden auf dem Hochleistungscomputersystem *bwUniCluster 3.0+KIT-GFA-HPC 3*, der sich im Wissenschaftlichen Rechenzentrum (SCC) am Karlsruher Institut für Technologie (KIT) befindet, durchgeführt.

### 2.3.1 Numerische Stabilität

N-Körper-Simulationen sind chaotische Systeme, deren Ergebnisse nicht mehr analytisch lösbar sind, sondern numerisch. Jede leichte Änderung der Anfangsbedingungen führt langfristig zu anderen Ergebnissen, die exponentiell anwachsen können [7]:

$$|\delta\vec{w}(t)| \approx |\delta\vec{w}(0)| \cdot e^{t/t_L} \tag{2.60}$$

$\delta\vec{w}$ bezeichnet eine kleine, anfängliche Änderung in der Phasenraum-Koordinate, bestehend aus den generalisierten Orten $\vec{q}$ und den Impulsen $\vec{p}$. $t_L$ ist hier die *Ljapunow*-Zeit und beschreibt die Zeit für das Anwachsen infinitesimaler Änderungen [7]. Dennoch können durch den Algorithmus bedingt numerische Fehler entstehen, die die tatsächlichen Ergebnisse verfälschen können. Diese „numerischen Artefakte" sollen so gut wie möglich kontrolliert und minimiert werden. Das Ziel solcher Simulationen ist es also, dass die simulierten Ergebnisse gegen die tatsächlich, wahren Ergebnisse aus der Natur konvergieren. Fehler, die bei der Simulation auftauchen können, sind zum einen Rundungsfehler des Computers, da dieser nur eine bestimmte Anzahl an Nachkommastellen berechnen kann. Das könnte die exponentielle Abweichung nach Gl. (2.60) während der Simulation verstärken. Eine weitere Fehlerquelle können Diskretisierungsfehler bei der Wahl der Zeitschrittlängen $\Delta t$ sein, mit der die Bewegungsgleichungen gelöst werden [8].
Dennoch besitzen Boekholt et al. zufolge solche N-Körper-Simulationen eine statistische Aussagekraft: Zwar können die simulierten Trajektorien einzelner Teilchen aufgrund der numerischen Fehler von den wahren Trajektorien abweichen [8]. Sie durchlaufen einen alternativen, aber dennoch physikalisch möglichen Pfad im Phasenraum. Damit bleiben die makroskopischen, statistischen Verteilungen des Gesamtsystems erhalten [8]. Damit die N-Körper-Simulation aussagekräftige Ergebnisse liefert, ist eine Berücksichtigung der numerischen Parameter unerlässlich, da diese auch den Ausgang der Simulation beeinflussen können, anstatt die korrekte physikalische Realität abzubilden. Konkret müssen folgende Parameter, die auch Power et al. in [34] aufführen, berücksichtigt werden:

- Der N-Körper-Code selbst
- Das Verfahren, um die Anfangsbedingung zu erzeugen
- Die Korrektheit der Kraftberechnung der Teilchen
- Das numerische Integrationsverfahren

- Die passende Wahl der Zeitschritte
- Der Softening-Parameter

Im Folgenden werden ein paar ausgewählte Methoden näher ausgeführt.

### 2.3.2 Numerische Integrationsmethoden

Da die Bewegungsgleichungen für N-Körper-Probleme (ab $N > 2$) im Vergleich zum Ein-Körper- und Zwei-Körper-Problem analytisch nicht mehr lösbar sind, greift man auf verschiedene numerische Methoden zurück. Zur Lösung der Bewegungsgleichungen verwendet man einen sogenannten *symplektischen Integrator*, der für die numerische Lösung von Hamilton'schen Gleichungen angewendet werden kann [7]: dieser hat allgemein die Aufgabe, die Gesamtenergie eines Systems zu wahren, damit es während der Simulation nicht künstlich zu Energieverlusten oder -zunahmen kommt (numerische Dissipation). Dafür muss das Volumen des Phasenraums zeitlich konstant bleiben (vgl. Liouvill'sches Theorem) [7].
Im Folgenden sollen zwei Integrationsmethoden, ein symplektischer und nicht-symplektischer Integrator, kurz vorgestellt und verglichen werden, um zu zeigen, welche Methode sich für die N-Körper-Simulation zum sphärischen Kollaps eines Dunkle-Materie-Halos besser eignet.

**Euler-Verfahren** Dieses Verfahren ist ein Beispiel für einen nicht-symplektischen Integrator und zeichnet sich dadurch aus, dass alle Änderungsraten wie beispielsweise die Geschwindigkeit oder Ort linear angenähert werden. Möchte man die Geschwindigkeit $\vec{v_j} = \dot{\vec{x_j}}$ eines Teilchens $j$ unter dem Einfluss einer Beschleunigung $\vec{a_j} = \dot{\vec{v_j}} = \ddot{\vec{x_j}}$ zu einem Zeitpunkt $t$ berechnen, lautet die Bewegungsgleichung [10]:

$$\dot{\vec{x_j}}(t_{n+1}) = \dot{\vec{x_j}}(t_n) + \Delta t \cdot \ddot{\vec{x_j}}(t_n) \tag{2.61}$$

und für den Ort:

$$\vec{x_j}(t_{n+1}) = \vec{x_j}(t_n) + \Delta t \cdot \dot{\vec{x_j}}(t_n) \tag{2.62}$$

$\Delta t$ bezeichnet das Zeitintervall und $n$ die Anzahl der Wiederholungsschritte (Iteration). Das Procedere der Rechnungen läuft immer nach folgenden Schema ab: Man startet mit einem Anfangsort $\vec{x_j}(t_n)$ und einer Anfangsgeschwindigkeit $\dot{\vec{x_j}}(t_n)$ zum Zeitpunkt $t_n$ nach einem Zeitschritt $n$ [10]. Mit dem Anfangsort berechnet man die Beschleunigung $\ddot{\vec{x_j}}(t_n)$, die beispielsweise über das Newtonsche Gravitationsgesetz definiert werden kann, wobei das Gesetz auch Grundlage für die Simulation des

Dunkle-Materie-Halos in dieser Abschlussarbeit ist. Die Beschleunigung ist gegeben durch [10]

$$\ddot{\vec{x}}_j(t) = -G \sum_{i \neq j} m_i \frac{\vec{x}_j(t) - \vec{x}_i(t)}{|\vec{x}_j(t) - \vec{x}_i(t)|^3} \tag{2.63}$$

und definiert die Summe der einzelnen Beschleunigungsvektoren, die jedes Teilchen $i$ auf Teilchen $j$ ausübt. Mit der berechneten Beschleunigung wird dann zusammen mit der Anfangsgeschwindigkeit die „neu aktualisierte“ Geschwindigkeit $\dot{\vec{x}_j}(t_{n+1})$ gemäß Gl. (2.61) zum Zeitpunkt $t_{n+1}$ berechnet und mit jener Geschwindigkeit der „neu aktualisierte“ Ort $\vec{x_j}(t_{n+1})$ nach Gl. (2.62) [10]. Dann beginnt das Procedere von vorne. Doch hier zeigt sich die Ungenauigkeit des Eulerverfahrens, denn die Beschleunigung eines Objekts wird für das betrachtete Zeitintervall $\Delta t$ als konstant angenommen, während in Realität die Beschleunigung maßgeblich vom Ort abhängt und sich innerhalb des Zeitintervalls schon wieder geändert haben könnte. Infolgedessen wachsen die Fehler mit Zunahme der Iterationen in der Ordnung $\mathcal{O}(\Delta t)$ an [30]. Es kommt zur Abweichung der numerischen Ergebnisse mit realen Ergebnissen, da die Energieerhaltung langfristig verletzt wird.

**Velocity-Verlet-Verfahren** Dieses Verfahren ist ein Beispiel für einen symplektischen Integrator und basiert auf dem Prinzip des *Leap-Frog-Verfahrens* (dt.: Bocksprung-Verfahren), bei dem man mit halben Zeitschritten zwischen einem Zeitpunkt $t_n$ und $t_{n+1}$, also $t_{n+1/2}$ rechnet. Beim klassischen *Leap-Frog-Verfahren* werden Ort und Geschwindigkeit abwechselnd bestimmt, weil beide Größen nicht gleichzeitig zum selben Zeitpunkt bekannt sind [38]. Die Geschwindigkeit zum Zeitpunkt $t_{n+1/2}$ ergibt dann [10]:

$$\dot{\vec{x}}_j(t_{n+1/2}) = \dot{\vec{x}}_j(t_n) + \frac{\Delta t}{2} \ddot{\vec{x}}_j(t_n) \tag{2.64}$$

Mit dieser berechneten „Halbzeit“-Geschwindigkeit wird nun der Ort zum vollen Zeitschritt $t_{n+1}$ berechnet [10]:

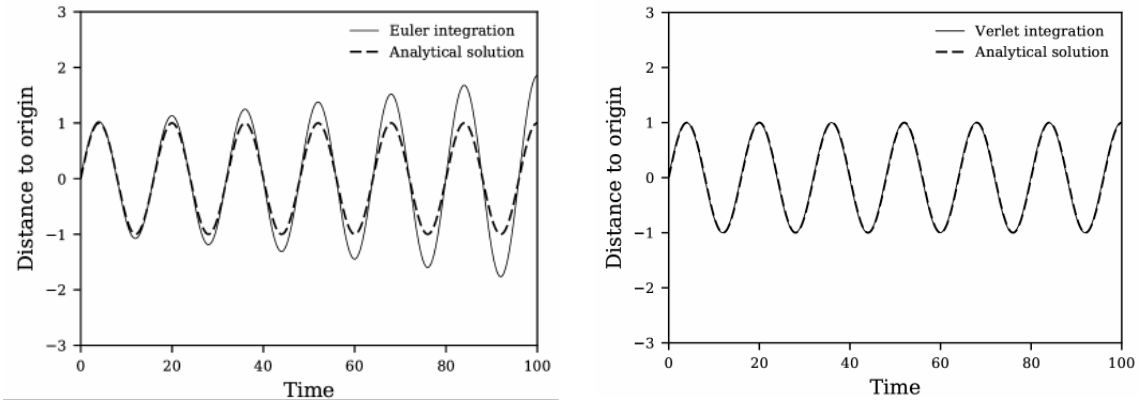
$$\vec{x}_j(t_n) = \vec{x}_j(t_n) + \Delta t \cdot \dot{\vec{x}}_j(t_{n+1/2}) \tag{2.65}$$

Anschließend wird mit dem neu bestimmten Ort $\vec{x}_j(t_n)$ die Beschleunigung für $t_{n+1}$ ermittelt und dann im Abschluss die Geschwindigkeit im vollen Zeitschritt berechnet [10]:

$$\dot{\vec{x}}_j(t_{n+1}) = \dot{\vec{x}}_j(t_{n+1/2}) + \frac{\Delta t}{2} \ddot{\vec{x}}_j(t_{n+1}) \tag{2.66}$$

Der Vorteil des Velocity-Verlet-Verfahrens liegt darin, dass sie eine deutlich bessere numerische Stabilität im Vergleich zum Euler-Verfahren mit sich bringt und

deshalb auch für die N-Körper-Simulation verwendet wird. Abbildung 2.3 zeigt die




(a) Euler-Verfahren (b) Velocity-Verlet-Verfahren

Abbildung 2.3: Vergleich der beiden Integrationsverfahren im zeitlichen Verlauf am Beispiel des eindimensionalen harmonischen Oszillators; entnommen aus [10].

Unterschiede der beiden Integrationsverfahren am Beispiel des harmonischen Oszillators. Beim Euler-Verfahren nimmt die Amplitude der Schwingung kontinuierlich zu, wodurch dem System künstlich Energie zugefügt wird und die Energieerhaltung verletzt. Beim Verlet-Verfahren zeigt sich eine Übereinstimmung zwischen den analytischen berechneten und den simulierten Ergebnissen. Symplektische Integratoren können die Energieerhaltung besser als nicht-symplektische bewahren [8].

### 2.3.3 Force-Softening und Zeitschritte

**Force-Softening** Das *softening* $\epsilon$, das in der Abschlussarbeit weiterhin als englisches Wort verwendet wird, heißt wörtlich übersetzt *weich machen* oder *abschwächen* und ist ein Parameter, der bei Berechnung von Kräften in N-Körper-Simulation verwendet wird. Der Zweck dieses softening wird einem schnell bewusst, wenn man sich überlegt, wie sich die Anziehungskraft zweier Körper verhält, wenn ihr Abstandsquadrat immer kleiner wird oder gar gegen Null geht ($r \to 0$). Gemäß dem Newton'schen Gravitationsgesetz gilt für die Kraft in Abhängigkeit des Abstands:

$$F \propto \frac{1}{r^2} \tag{2.67}$$

Strebt der Abstand $r \to 0$, wird demnach die Kraft $F \to \infty$ gehen. Da ein Computer aber keine "unendlichen" Werte für die Kraft berechnen kann, führt man das softening $\epsilon$ ein, um das Kraftgesetz zu modifizieren und eine obere Schranke für die Kraft einzuführen. Die Teilchen, aus denen der Halo in den Simulationen besteht,

sind hier nicht als Atome oder winzig kleine Punktteilchen aufzufassen. Denn ein echter Halo (ohne sichtbare Materie) besteht aus einer unvorstellbar großen Zahl an „Dunkle-Materie-Teilchen“. Der Computer müsste für jedes Teilchen $N$ zu jedem anderen Teilchen $N-1$ die Wechselwirkungskraft berechnen, was sehr schnell den Rechenaufwand massiv übersteigen würde. Denn für die Gesamtzahl der Kraftberechnung $F_{tot}$ gilt [27]:

$$F_{tot} = N \cdot (N-1) \approx N^2 \tag{2.68}$$

Wenn wir beispielsweise annehmen, dass ein Halo aus $N = 10^{60}$ Dunkle-Materie-Teilchen besteht, dann wäre $F_{tot} \approx (10^{60})^2 = 10^{120}$, was einer unvorstellbaren großen Summe entspricht, für die ein Computer zur Berechnung ewig Zeit bräuchte. Bei kollisionsfreien N-Körper-Simulationen werden daher häufig viel weniger Teilchen $N_{Sim}$ verwendet als das System aus realen Teilchen $N_{real}$ besteht ($N_{Sim} \ll N_{real}$) [7]. Daraus folgt dann auch, dass die simulierten Teilchen mit der Masse $M_{Sim}$ so aufgefasst werden, dass sie die Masse von mehreren Dunkle-Materie-Teilchen $M_{DM}$ – quasi wie „Makro-Teilchen“ – repräsentieren und entsprechend eine Masse von mehreren Sonnenmassen haben können ($M_{DM} \ll M_{Sim}$). In Simulationen von Kugelsternhaufen repräsentiert ein Teilchen beispielsweise die Masse eines jeden Sterns [27].
Intuitiv könnte man sich denken, das Kraftgesetz dann folgendermaßen zu modifizieren:

$$F \propto \frac{1}{r^2 + \epsilon^2} \tag{2.69}$$

Diese Formel wäre zwar konsistent mit den Einheiten, wäre aber dennoch nicht ganz richtig. Denn wenn man das Gravitationspotential, das man über die Beziehung $-m \cdot \vec{\nabla}\Phi(r) = \vec{F}$ herleiten kann, berechnet, wäre $\Phi(r) \propto \ln(r^2 + \epsilon^2)$. Das entspricht eben nicht dem Newton'schen Gravitationspotential

$$\Phi(r) \propto -\frac{1}{r} \tag{2.70}$$

Stattdessen verwendet man das sogenannte *Plummer-Potential* [1],

$$\Phi(r) = -\frac{GM}{\sqrt{r^2 + \epsilon^2}} \tag{2.71}$$

dessen Bezeichnung auf den Kosmologen H.C. Plummer zurückgeht. 1911 entdeckte Plummer eine zufriedenstellende Dichteverteilung $\rho(r)$ von offenen Kugelsternhaufen [33]. Wendet man die Poisson-Gleichung ($\Delta\Phi = 4\pi G\rho$) auf die Dichteverteilung an, erhält man das Potential $\Phi(r)$ [9]. Wendet man aber nun $-m \cdot \vec{\nabla}\Phi(r) = \vec{F}$ auf das

kugelsymmetrische Plummer-Potential in Gl. (2.71) an, so folgt:

$$\vec{F} = -\frac{GMm}{(r^2 + \epsilon^2)^{3/2}} \cdot \vec{r} \tag{2.72}$$

Abb. 2.4 zeigt exemplarisch den Unterschied zwischen dem klassichen Newton'schen

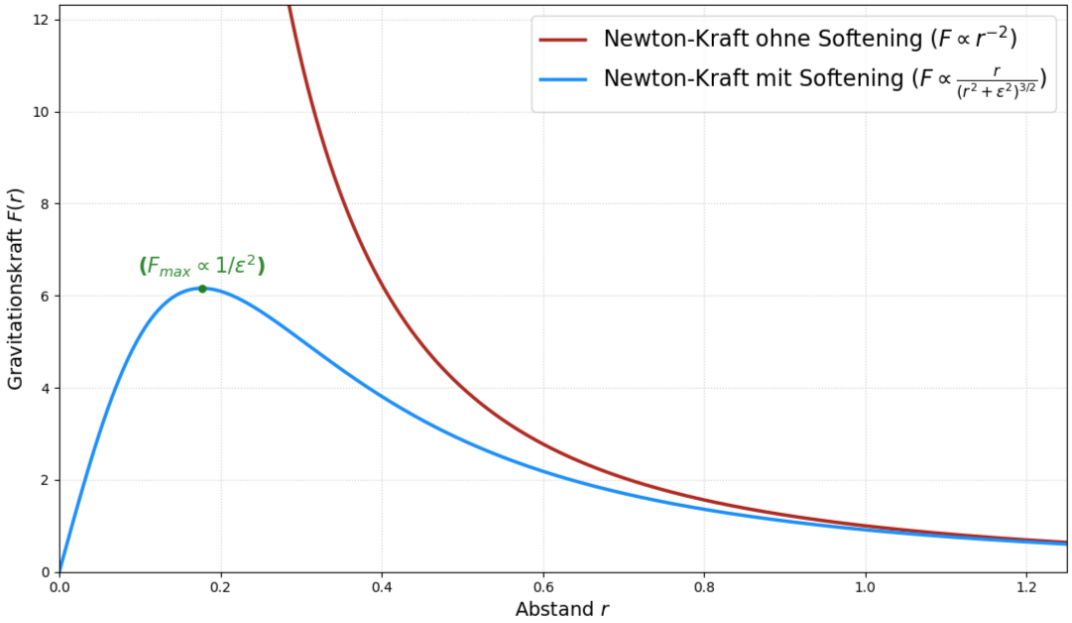



Abbildung 2.4: Qualitativer Vergleich zwischen dem Newtonschen Gravitationsgesetz mit und ohne dem softening-Parameter (hier exemplarisch mit $\epsilon = 0.25$)

Gravitationsgesetz und dem mit einem softening $\epsilon$ modifizierten Kraftgesetz. Interessant ist hier, dass die Kraft für $r = 0$ auch null ergibt. Das erkennt man nicht nur an Gl. (2.72), sondern lässt sich auch physikalisch so erklären: Die Teilchen erhalten mit dem Parameter $\epsilon$ eine Ausdehnung, sodass sie sich kollisionsfrei, wie es auch bei N-Körper-Simulationen zu beispielsweise Galaxienhaufen angenommen wird, durchdringen können. Durchdringt nun ein Teilchen $A$ ein Teilchen $B$, sodass sich $A$ vollständig in $B$ befindet, heben sich die Anziehungskräfte von $B$ auf $A$ vollständig auf. Die Kraft ist für einen kurzen Moment tatsächlich null. Für $r = \epsilon$ ergibt sich die maximale, gedeckelte Kraft mit $F_{max} \propto 1/\epsilon^2$.
Die Frage ist aber nun, wie groß ein passender Wert für den softening-Parameter gewählt werden sollte, ohne die Kräfte der Teilchen künstlich zu verzerren. Eine vorgeschlagene Faustregel in [27] lautet, den mittleren Abstand der Teilchen, die in einer Kugel mit einem Volumen $V$ verteilt sind, zu nehmen:

$$\epsilon \approx V_{Teilchen}^{1/3} \tag{2.73}$$

Das Volumen $V_{Teilchen}$ lässt sich über das Kugelvolumen pro Gesamtzahl der Teilchen wie folgt berechnen:

$$V_{Teilchen} = \frac{V_{Kugel}}{N} = \frac{\frac{4}{3}\pi R_{vir}}{N} \tag{2.74}$$

Setzt man Gl. (2.74) in Gl. (2.73) ergibt sich für den softening-Parameter, wobei $R_{vir}$ den Virialradius bezeichnet [27]:

$$\epsilon = \left(\frac{\frac{4}{3}\pi R_{\text{vir}}^3}{N}\right)^{1/3} = \left(\frac{4}{3}\pi\right)^{1/3} \cdot R_{vir} \cdot N^{-1/3} \tag{2.75}$$

Für die in der Abschlussarbeit unterschiedlich verwendete Teilchenzahl $N$ wurden die Simulationen mit jeweils drei verschiedenen Werten für den softening-Parameter durchgeführt. Der mit Gl. (2.75) berechnete Wert wird hier als eine Art maximales softening ($\epsilon_{max}$) aufgefasst, sodass dann einmal die Hälfte sowie ein Viertel von $\epsilon_{max}$ als Werte herangezogen wurden. Tabelle 2.1 fasst die softening-Werte für die verschiedenen $N$ Teilchen zusammen:

Tabelle 2.1: Übersicht über die in den Simulationen verwendeten softening-Werte in Abhängigkeit der Teilchenanzahl. Werte wurden aufgerundet. In der Simulation ist $R_{vir} = 50/2 = 25$ Codeeinheiten gewählt.

| N Teilchen | 50 | 100 | 200 | 2000 |
|---|---|---|---|---|
| $\epsilon_{max}$ | 11,0 | 8,7 | 7,0 | 3,2 |
| $\epsilon_{max}/2$ | 5,5 | 4,35 | 3,5 | 1,6 |
| $\epsilon_{max}/4$ | 2,75 | 2,18 | 1,75 | 0,8 |

**Zeitschritte** Die Festlegung eines geeigneten Werts für die Zeitschritte spielt eine wesentliche Rolle für die numerischen Stabilität einer N-Körper-Simulation. Denn während der Zeit ändern sich die Abstände der Teilchen untereinander und damit auch ihre Kräfte. Man stelle sich zwei Teilchen vor, die sich aufeinander zu bewegen: Legt das Teilchen bei einer zu großen Zeitschrittlänge $\Delta t$ eine große Strecke zurück und landet sehr nah beim anderen Teilchen, berechnet der Computer eine große Beschleunigung $a$ zwischen diesen Teilchen. Wendet der Computer in diesem Moment die Beschleunigung auf die gesamte Zeitschrittlänge $\Delta t$ an, dann führt dies fälschlicherweise zu einer überhöhten Geschwindigkeit $\Delta v$ gemäß der Formel $\Delta v = a \cdot \Delta t$. Wenn sich die Teilchen in der Realität aber wieder voneinander entfernen, fällt auch die Beschleunigung wieder ab. Ist der Zeitschritt aber zu groß gewählt, führen die Berechnungen zu Abweichungen von den realen Geschwindigkeiten unter den Teil-

chen. Infolgedessen kann es vorkommen, dass Teilchen künstlich zu viel Energie erhalten und wie durch einen Steinschleudereffekt (engl. *slingshot effect*) das gebundene System des Teilchenensembles verlassen können. Natürlich könnte man einen sehr kleinen, konstanten Wert für die Zeitschritte festlegen, müsste aber eine höhere Berechnungsdauer des Computers in Kauf nehmen, um den Diskretisierungsfehler zu minimieren. Jedoch führt dies dazu, dass dann die Rundungsfehler eines Computers zu nehmen, weil eben mehr Rechenschritte ausgeführt werden müssen [8]. Im Allgemeinen bietet es sich bei N-Körper-Simulationen an, adaptive Zeitschritte, die sich während der Simulation ändern, zu verwenden, um eine höhere Genauigkeit bei der wechselseitigen Kraftberechnung der Teilchen zu erzielen [34]. Power et al. schlagen vor, die Zeitschritte $\Delta t$ in Abhängigkeit der Beschleunigung $a$ und des Softening-Parameters $\epsilon$ ausdrücken. Konkret soll gelten [34]:

$$\Delta t = \eta \cdot \sqrt{\frac{\epsilon}{a}} \tag{2.76}$$

$\eta$ ist als eine dimensionslose Konstante definiert, die die Größe der Zeitschritte kontrolliert. Power et al. halten aufgrund ihrer Ergebnisse den Wert $\eta \approx 0,2$ für geeignet [34]. Je kleiner der Wert gewählt wird, desto kleiner werden auch die Zeitschritte und umgekehrt. Man erkennt auch an Gl. (2.76), dass $\Delta t$ umso kleiner wird, je größer die Beschleunigung zwischen den Teilchen wird und umgekehrt. Daher bietet diese Formel einen guten Kompromiss zwischen einer erhöhten Genauigkeit bei der Berechnung der Beschleunigung für nahe Begegnungen und die Vermeidung einer unnötig hohen Berechnungsdauer, wenn die Distanzen der Teilchen größer werden, sodass auch größere Zeitschritte ausreichen.

### 2.3.4 Verteilung der Teilchen in einer Sphäre

Für die N-Körper-Simulation werden eine unterschiedliche Anzahl von $N$ Teilchen ($N = 50; 100; 200; 2000$) gewählt und in einer Kugel mit einem Radius $R$ zufällig verteilt. Für die zufällige Generierung der Winkelanteile wird für jeden Simulationsdurchlauf ein zufälliger Punkt in einem 2D-Parameterfeld ausgewählt [6]. Abbildung 2.5 zeigt schematisch so ein Parameterfeld.

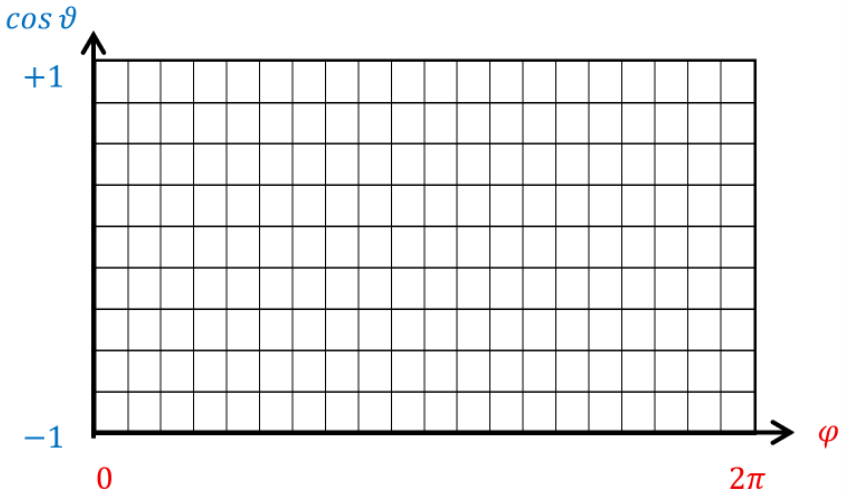



Abbildung 2.5: Schematische Darstellung eines 2D-Paramterfelds mit dem Polarwinkel $+1 \geq cos\vartheta \geq -1$ auf der y-Achse und dem Azimutwinkel $2\pi \geq \varphi \geq 0$ auf der x-Achse. Abbildung selbst erstellt.

Um eine isotrope Verteilung des Polarwinkel $\vartheta$ zu erhalten, ist es nicht möglich, einfach eine Zufallszahl zwischen 0 und $\pi$ für den Winkel zu wählen. Denn weil das Flächenelement $dA$ einer Kugel, beschrieben wird durch

$$dA = r^2 \sin\vartheta \, d\vartheta \, d\varphi \tag{2.77}$$

geht $\sin\vartheta \to 0$ an den Polen einer Kugel, also bei $\vartheta = 0$ und $\vartheta = \pi$, und somit geht auch das Flächenelement $dA \to 0$. Würde man außerdem die Teilchen über den Winkel $\vartheta$ gleich verteilen, hätte man an den Polen der Kugel eine erhöhte Dichte an Teilchen als am Äquator. Denn die Teilchen können nahe des Äquators auf eine größere Fläche verteilt werden als an den Polen. Um das zu vermeiden, wählt man eine Zufallszahl $u$ im Intervall $[-1, +1]$ für $\cos\vartheta$ [37], sodass sich nahe des Äquators mehr Teilchen aufhalten als an den Polen.
Der Radius, also der Abstand eines jeden Teilchens zum Zentrum der Sphäre wird ebenso zufällig für jeden Simulationsdurchlauf gewählt. Die Verteilung der Teilchen innerhalb der Kugel muss aber so erfolgen, dass die Dichte der Kugel über ihr gesamtes Volumen hinweg gleichmäßig ist. Nun könnte man sich vorstellen, die Kugel in Schalen mit gleicher Schalendicke $dr$ zu unterteilen und jede Schale mit derselben Teilchenanzahl aufzufüllen. Dann hätte man aber das Problem, dass die Dichte der gesamten Kugel inhomogen wäre, da die Kugelschalen, die weiter weg vom Zentrum sind, ein größeres Raumvolumen einnehmen als näher am Zentrum befindliche Kugelschalen. Denn es gilt:

$$dV = 4\pi r^2 dr \tag{2.78}$$

Demnach müssen die Kugelschalen mit einer jeweils unterschiedlich großen Anzahl an Teilchen aufgefüllt werden, um eine gleichmäßige, homogene Dichte in der Kugel zu erhalten. Das wird dadurch erreicht, indem eine sogenannte Inversion durchgeführt wird [37]. Man generiert eine Zufallszahl $u$ im Intervall $[0, 1]$, die sich als Verhältnis

$$u = \frac{\frac{4\pi}{3}r^3}{\frac{4\pi}{3}R^3} = \left(\frac{r}{R}\right)^3 \tag{2.79}$$

auffassen lässt. Wenn ein Teilchen sich bei $r = R$ befindet, dann ist $u = 1$ und bei $r = 0$ folgt $u = 0$. Nun formt man den Ausdruck nach dem Radius $r$ um und zieht die dritte Wurzel:

$$r = u^{1/3} \cdot R \tag{2.80}$$

Abbildung 2.6 zeigt den qualitativen Vergleich zweier Möglichkeiten, wie die Teilchen mit exemplarisch je $N = 500$ innerhalb einer Sphäre verteilt werden können.

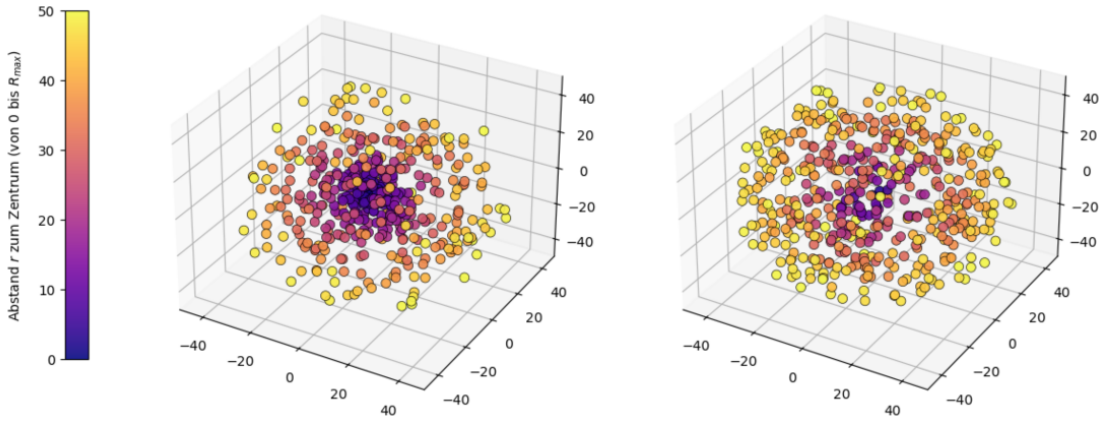



Abbildung 2.6: Qualitativer Vergleich der Teilchenverteilung innerhalb einer Sphäre mit $R_{max} = 50$ für exemplarisch $N = 500$ Teilchen mit Farbkodierung: Im linken Bild ist die Teilchenanzahl in jeder Kugelschale konstant ($\rho_{Kugel}$ ist damit inhomogen) und im rechten Bild nicht konstant ($\rho_{Kugel}$ ist damit homogen).

Die linke Abbildung zeigt, dass die Dichte im Zentrum deutlich höher ist, weil sich sehr viele Teilchen in einem kleinen Volumen befinden, und nach außen hin die Teilchenzahl abnimmt. Das rechte Bild zeigt eine homogene Dichte innerhalb der Sphäre.

### 2.3.5 Skalierung der Bewegungsgleichungen

Da ein Computer keine Einheiten rechnen kann, müssen wir die Bewegungsgleichung der Teilchen für die Simulationen in eine dimensionslose Form überführen. Dazu

betrachten wir die allgemeine Bewegungsgleichung gemäß der Ursprungsgleichung Gl. (2.1) für ein Teilchen $i$, auf das andere Teilchen $j$ eine Anziehungskraft ausüben:

$$m_i \frac{d^2 r_i}{dt^2} = -\sum_{i \neq j}^{N} \frac{G m_i m_j}{|r_i - r_j|^3}(r_i - r_j) \tag{2.81}$$

Kürzen liefert:

$$\frac{d^2 r_i}{dt^2} = -\sum_{i \neq j}^{N} \frac{G m_j}{|r_i - r_j|^3}(r_i - r_j) \tag{2.82}$$

Nun führen wir dimensionslose Größen für die Masse $m_j$ eines Teilchens, für den Radius $r_i$ und die Zeit $t$ ein,

$$\widetilde{m}_j = \frac{m_j}{M_0} \quad | \quad \widetilde{r}_i = \frac{r_i}{R_0} \quad | \quad \widetilde{t} = \frac{t}{T_0} \tag{2.83}$$

wobei die Größen $M_0, R_0$ und $T_0$ die jeweiligen Bezugsgrößen sind. $M_0$ bezeichnet die Bezugsmasse, für die wir die Gesamtmasse eines Halos nehmen ($M_0 = M_{ges}$). Dadurch ergibt sich, dass wir eine dimensionslose Form der Gesamtmasse $\widetilde{M}_{ges}$ einführen können:

$$\widetilde{M}_{ges} = \frac{M_{ges}}{M_0} = \frac{M_{ges}}{M_{ges}} = 1 \tag{2.84}$$

Die Gesamtmasse $M_{ges}$ ergibt sich durch das Produkt der Masse $m$ eines jeden Teilchens und der Teilchenzahl $N$, also $M_{ges} = N \cdot m$, mit der die N-Körper-Simulation durchgeführt wird. Hier wird auch die Annahme getroffen, dass die Teilchen alle dieselbe, reale Masse haben sollen. Da für die Simulationen die Teilchenzahl $N$ variiert wird, verändert sich auch die Masse $\widetilde{m}_j$ in Abhängigkeit der Teilchenzahl [3]:

$$\widetilde{m}_j = \frac{m_j}{M_0} = \frac{\frac{M_{ges}}{N}}{M_{ges}} = \frac{1}{N} \tag{2.85}$$

Das bedeutet, dass die Einzelmassen $\widetilde{m}_j$ umso kleiner werden, je weiter die Teilchenzahl erhöht wird und umgekehrt. Insgesamt heißt das, dass wir einen Halo mit einer festgelegten Masse simulieren und die Anzahl der Teilchen, aus denen er besteht, vergrößern, also schlichtweg die „Auflösung“ des Halos mit variierender Teilchenzahl. Als Letztes betrachten wir die Zeitskala und ersetzen nun die Größen $r_i$, $m_j$ und $t$ in Gl. (2.82) durch die dimensionslosen Skalierungen aus Gl. (2.83), womit wir erhalten:

$$\frac{d^2(R_0 \widetilde{r}_i)}{d\left(T_0 \widetilde{t}\right)^2} = -\sum_{i \neq j}^{N} \frac{G\ \widetilde{m}_j M_0}{|\widetilde{r}_i R_0 - \widetilde{r}_j R_0|^3}(\widetilde{r}_i R_0 - \widetilde{r}_j R_0) \tag{2.86}$$

Zieht man die Bezugsgrößen sowie die Konstante $G$ heraus, ergibt sich:

$$\frac{R_0}{T_0^2}\frac{d^2\widetilde{r}_i}{d\tilde{t}^2} = -\left(\frac{GM_0}{R_0^2}\right)\sum_{i\neq j}^{N}\frac{\widetilde{m}_j}{|\widetilde{r}_i-\widetilde{r}_j|^3}(\widetilde{r}_i-\widetilde{r}_j) \tag{2.87}$$

Sortieren der Größen liefert:

$$\frac{d^2\widetilde{r}_i}{d\tilde{t}^2} = -\left(\frac{GM_0T_0^2}{R_0^3}\right)\sum_{i\neq j}^{N}\frac{\widetilde{m}_j}{|\widetilde{r}_i-\widetilde{r}_j|^3}(\widetilde{r}_i-\widetilde{r}_j) \tag{2.88}$$

Die Überprüfung der Einheiten im eingeklammerten Ausdruck zeigt,

$$\frac{[G][M_0][T_0^2]}{[R_0^3]} = \frac{\frac{\mathrm{m}^3}{\mathrm{kg}\cdot\mathrm{s}^2}\cdot\mathrm{kg}\cdot\mathrm{s}^2}{\mathrm{m}^3} = \frac{\cancel{m^3}\,\cancel{kg}\,\cancel{s^2}}{\cancel{kg}\,\cancel{s^2}\,\cancel{m^3}} = 1 \tag{2.89}$$

dass Gl. (2.88) dimensionslos ist. Der eingeklammerte Ausdruck erinnert wieder an das dritte Kepler'sche Gesetz mit $R^3 \propto t_c^2$, wie wir es schon in der Herleitung für den spährischen Kollaps in Gl. (2.12) bemerkt haben. Für $T_0$ erhält man den Ausdruck:

$$T_0 = \sqrt{\frac{R_0^3}{GM_0}} \tag{2.90}$$

Vergleicht man $T$ (ohne Index 0) und $t_c$ ganz allgemein miteinander, fällt auf, dass

$$t_c \underbrace{=}_{\rightarrow \text{Gl. (2.12)}} \sqrt{\frac{\pi^2R^3}{2GM_{ges}}} = \frac{\pi}{\sqrt{2}}\underbrace{\sqrt{\frac{R^3}{GM_{ges}}}}_{=T} \approx 2,22\,T \tag{2.91}$$

gilt und $t_c$ und $T$ sich damit um einen Faktor $2,22$ unterscheiden.
Genauso können wir die sogenannte *Freifallzeit* $t_{ff}$ angeben, die die Fallzeit der Teilchen zwischen dem *Turn-around*-Punkt bei der größten Ausdehnung bis zum Kollaps beschreibt. Dies entspricht genau der Hälfte der Kollapszeit:

$$t_{ff} = \frac{1}{2}\sqrt{\frac{\pi^2R^3}{2GM_{ges}}} = \frac{\pi}{2\sqrt{2}}\underbrace{\sqrt{\frac{R^3}{GM_{ges}}}}_{=T} \approx 1,11\,T \tag{2.92}$$

An der Stelle sei auch angemerkt, dass die Freifallzeit, die allgemein durch den Ausdruck

$$t_{ff} = \sqrt{\frac{3\pi}{32G\bar{\rho}}} \tag{2.93}$$

definiert ist, aus Gl. (2.2) für die spezifische Energie hergeleitet werden kann, wenn man ein Punktteilen in Ruhe ($v = 0 \rightarrow E_{tot} = E_{pot}$) im Abstand $r = R$ auf der Oberfläche einer Kugel (mit Radius $R$ und konstanter, mittlerer Dichte $\bar{\rho}$) annimmt und die Gleichung nach der Zeit integriert [35].
Für das simulierte System können wir die Kollapszeit nach $\tilde{t_c} = 2,22\,\widetilde{T}_0$ berechnen. Die reale Zeit $t_c$ eines echten Halos können wir wieder durch Einführung von dimensionslosen Bezugsgrößen mit $\widetilde{R} = R/R_0$, $\widetilde{G} = G'/G$ und $\widetilde{M}_{ges} = M/M_{ges}$ in Gl. (2.91) umschreiben zu:

$$t_c = \frac{\pi}{\sqrt{2}}\sqrt{\frac{(\widetilde{R}\cdot R_0)^3}{(\widetilde{G}\cdot G)(\widetilde{M}_{ges}M_0)}} = \underbrace{\frac{\pi}{\sqrt{2}}\underbrace{\sqrt{\frac{\widetilde{R}^3}{\widetilde{G}\widetilde{M}_{ges}}}}_{\widetilde{T}_0}}_{\tilde{t_c}}\underbrace{\sqrt{\frac{R_0^3}{GM_0}}}_{T_0} \tag{2.94}$$

Weil üblicherweise $\widetilde{G} = 1$ gewählt wird [3], $\widetilde{M}_{ges} = 1$ nach Gl. (2.84) ist und der Radius im Simulationscode nach freier Wahl auf $\widetilde{R} = 50$ Code-Längeneinheiten gesetzt wurde, ergibt sich für $\tilde{t_c}$:

$$\widetilde{t_c} = 2,22\,\widetilde{T}_0 = 2,22\sqrt{\frac{50^3}{1\cdot 1}} \approx 785,4\,\text{Codeeinheiten} \tag{2.95}$$

Die Rechnung für $t_{ff}$ erfolgt analog wie Gl. (2.94) nur mit dem Vorfaktor 1/2 vor dem Ausdruck. Damit lässt sich auch $\widetilde{t}_{ff}$ in Codeeinheiten bestimmen.

$$\widetilde{t}_{ff} = 1,11\,\widetilde{T}_0 \approx 392,4\,\text{Codeeinheiten} \tag{2.96}$$

Die Kollapszeit $\tilde{t_c}$ beträgt laut den Ergebnissen der Simulationen etwa $\tilde{t_c} \approx 900-915$, also etwa das 1,44- bis 1,16-fache der theoretischen Kollapszeit $\tilde{t_c}$. Das könnte darauf zurückzuführen sein, dass aufgrund der dynamischen Reibung, die sich durch die asymmetrische Anordnung ergibt, sich Teilchen untereinander teilweise abbremsen können und das Zentrum der Kugel später erreichen.
Damit man eine Vorstellung über die Dauer der Freifallzeit bis zum Kollaps für echte Halos erhält, kann man nun für verschiedene Massen und Größen von Halos die entsprechende Freifallzeit $t_{ff}$ (oder auch $t_c$) berechnen. Die Tabelle 2.2 zeigt eine Übersicht der gemäß Gl. (2.92) berechneten Freifallzeiten in Gyr ($= 1\cdot 10^9\,a$) an. Die folgenden Werte sind hier nur als Beispielwerte aufzufassen, da ein Halo bei einer gegebenen Masse je nach der kritischen Dichte des Universums zur Rotverschiebung $z$ mit einem bestimmten Radius korreliert.

Tabelle 2.2: Freifallzeiten $t_{ff}$ [in Gyr] für verschiedene (exemplarisch gewählte) Massen $M$ und Radien $R$ von Halos.

| M \ R | 50 kpc | 100 kpc | 200 kpc |
|---|---|---|---|
| $10^{10}\,M_\odot$ | 1,85 | 5,24 | 14,81 |
| $10^{11}\,M_\odot$ | 0,59 | 1,66 | 4,68 |
| $10^{12}\,M_\odot$ | 0,19 | 0,52 | 1,48 |
| $10^{13}\,M_\odot$ | 0,06 | 0,17 | 0,47 |

Der Trend der Tabelle zeigt, dass die Freifallzeit umso länger dauert, je größer der Radius des Halos wird. Und je größer die Masse des Halos ist, umso schneller kollabiert er. Theoretisch würde ein – im Vergleich zu massereicheren Halos – leichter Halo der Masse $M_{Halo} = 10^{10} M_\odot$ und einer Ausdehnung von $R = 200\,\mathrm{kpc}$ in einem statischen Hintergrund mehr als das jetzige Alter des Universums an Zeit benötigen, um zu kollabieren. Selbst wenn der Kollaps gleichzeitig mit der Entstehung des Universum begonnen hätte, wäre der Kollaps heute immer noch nicht vollendet. Dass solch ein Halo in der heutigen Zeit überhaupt nicht kollabieren könnte, liegt auch daran, dass sein Dichtekontrast kleiner ist als der Wert, der in Gl. (2.22) nach dem Modell des sphärischen Kollapses berechnet wurde. Betrachtet man den Halo als sphärisch mit konstanter, mittlerer Dichte $\bar{\rho}$ und nimmt für die kritische Dichte des Universums (für $z = 0$) $\rho_c = 9,2 \cdot 10^{-27}\,\mathrm{m^3\,kg^{-1}}\,(\mathrm{h_{70}})^2$, dann ergibt der Dichtekontrast:

$$\delta = \frac{\bar{\rho}}{\rho_c} \approx 2,2 \tag{2.97}$$

Ein Halo derselben Masse, aber mit kleinerem Radius von $R = 50\,\mathrm{kpc}$ hätte eine Freifallzeit von etwa 1,85 Milliarden Jahre und hätte somit genug Zeit gehabt zu kollabieren. Auch Halos größerer Massen und Radien würden in einer viel kürzeren Zeit kollabieren als das Alter des Universums und könnten damit als heutige Strukturen beobachtet werden.

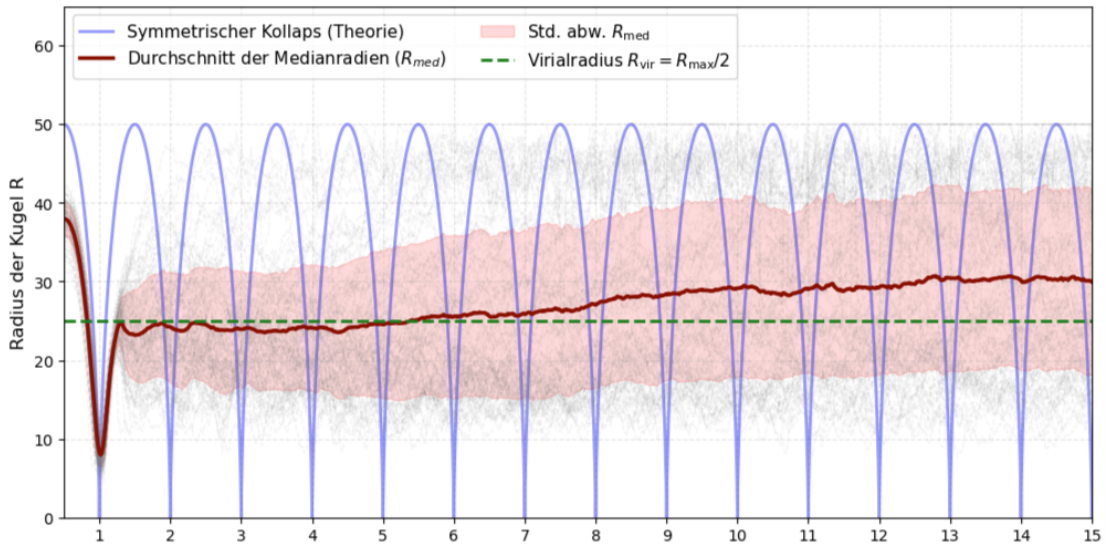
## 2.4 Ergebnisse der N-Körper-Simulationen

### 2.4.1 Zeitliche Entwicklung des Halokollapses

Nachdem in den vorherigen Unterkapiteln die Voraussetzungen für einen korrekten N-Körper-Simulationscode sowie die Parameter dargestellt wurden, sollen im

Folgenden hier die Ergebnisse der durchgeführten Simulationen aufgeführt und diskutiert werden. Es wurde jede Anzahl an $N$ Teilchen mit den drei vorher bestimmten softening-Werten jeweils 100 mal simuliert. Das bedeutet, dass mit jedem Simulationsdurchlauf eine leicht unterschiedliche Anordnung der Teilchen innerhalb der Sphäre erzeugt wird. Jedes Teilchen selbst startet aus der Ruhe, also $v = 0$. Außerdem wird die Simulation sehr oft wiederholt, um das statistische Verhalten zu zeigen.
Zur Darstellung der Ergebnisse sei angemerkt, dass der Medianradius aller Abstände der Teilchen zum Zentrum der Kugel gegen die Zeit $t$ in Einheiten der Kollapszeit $t_c$ aufgetragen wurde. Der Median (anstelle des arithmetischen Mittelwerts) wird deshalb gewählt, weil er gegenüber sehr großen oder kleinen Werten robuster ist. Jede Kurve selbst ist als Medianradius gegen die Zeit aufgetragen. In den Schaubildern ist dann immer der Mittelwert (rote durchgezogene Kurve) aller Mediankurven mit einer Standardabweichung von $\pm 1\sigma$ abgebildet.

Die folgenden Schaubilder zeigen den Simulationsverlauf für $N = 50$ Teilchen bis zu einer Zeit von $t/t_c = 15$, also das 15-fache der Kollapszeit.



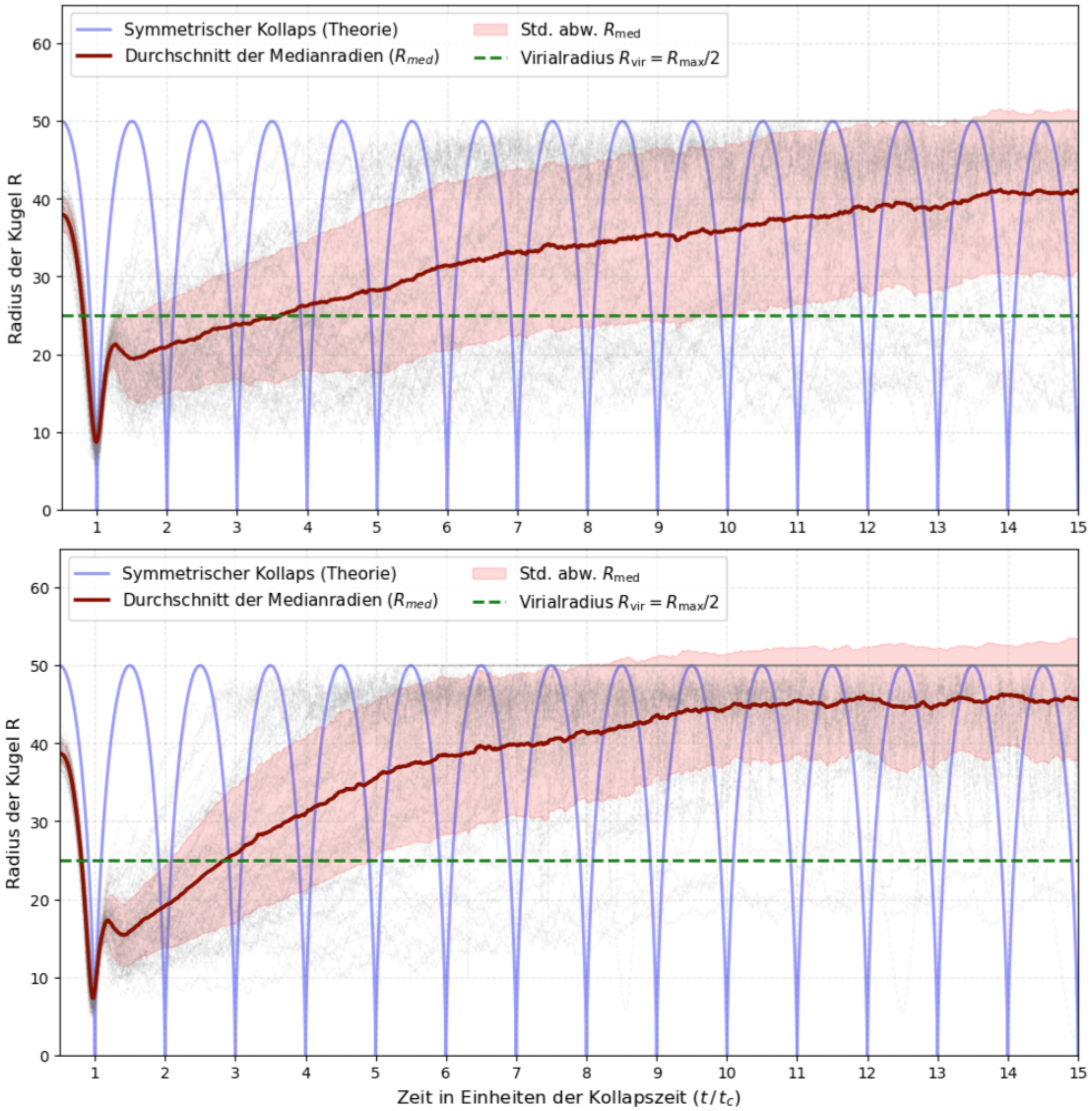



Abbildung 2.7: Ergebnisse der durchgeführten Simulationen für $N = 50$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 11,0$, für das mittlere $\epsilon_{max}/2 = 5,5$ und für das untere $\epsilon_{max}/4 = 2,75$.

Die folgenden Schaubilder zeigen den Simulationsverlauf für $N = 100$ Teilchen bis zu einer Zeit von $t/t_c = 15$, also das 15-fache der Kollapszeit.

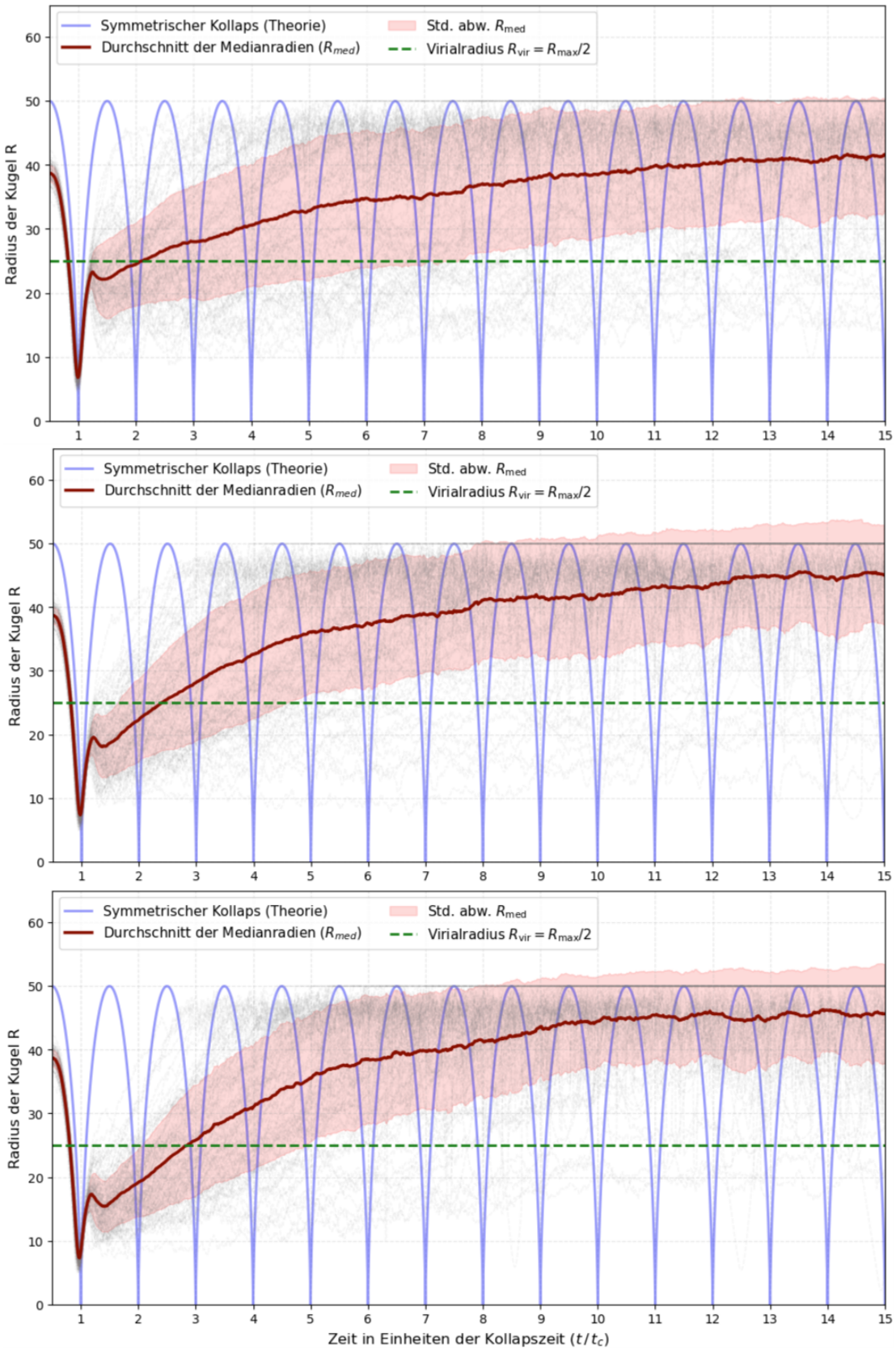
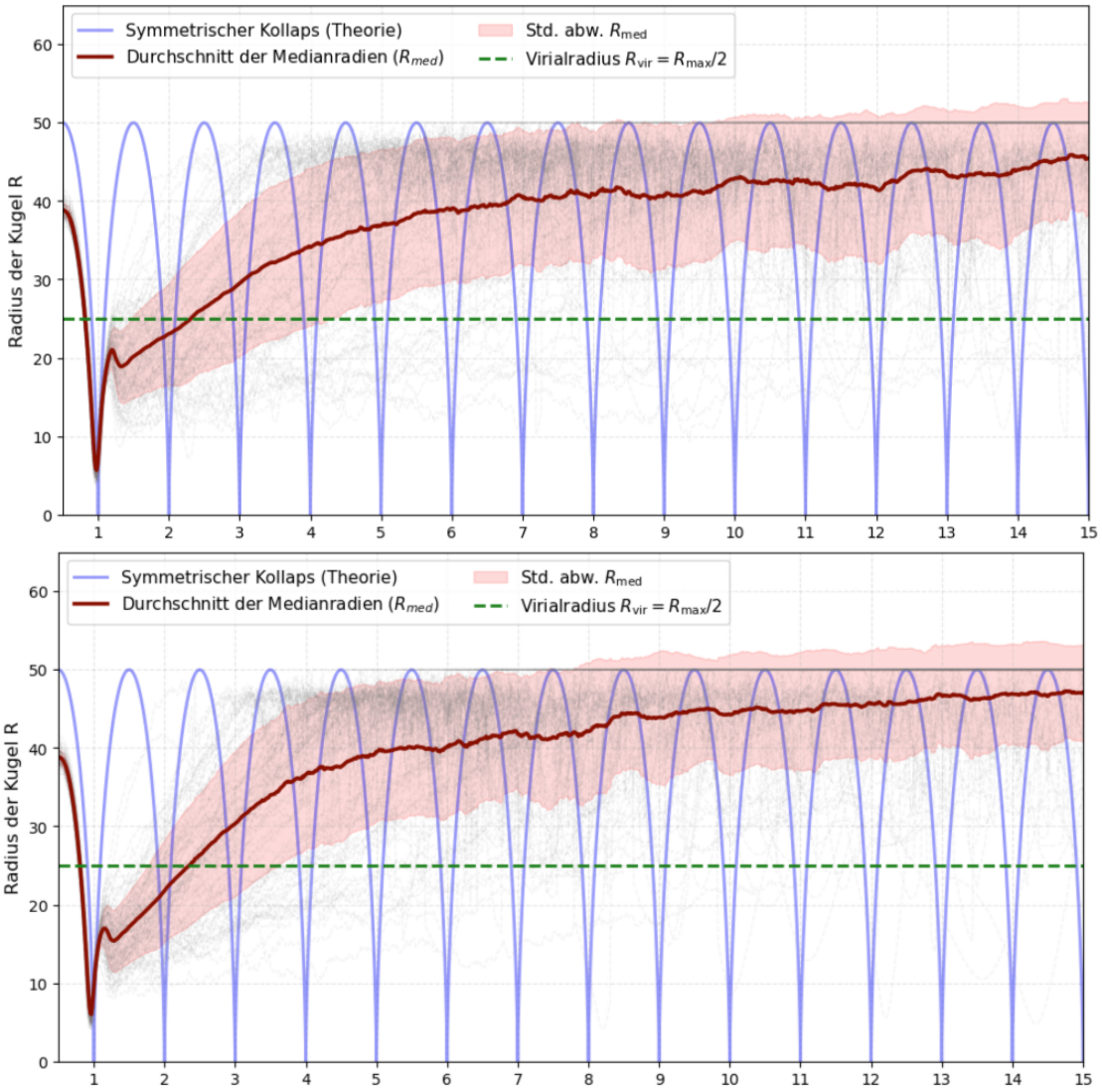



Abbildung 2.8: Ergebnisse der durchgeführten Simulationen für $N = 100$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 8,7$, für das mittlere $\epsilon_{max}/2 = 4,35$ und für das untere $\epsilon_{max}/4 = 2,18$.

Die folgenden Schaubilder zeigen den Simulationsverlauf für $N = 200$ Teilchen bis zu einer Zeit von $t/t_c = 15$, also das 15-fache der Kollapszeit.



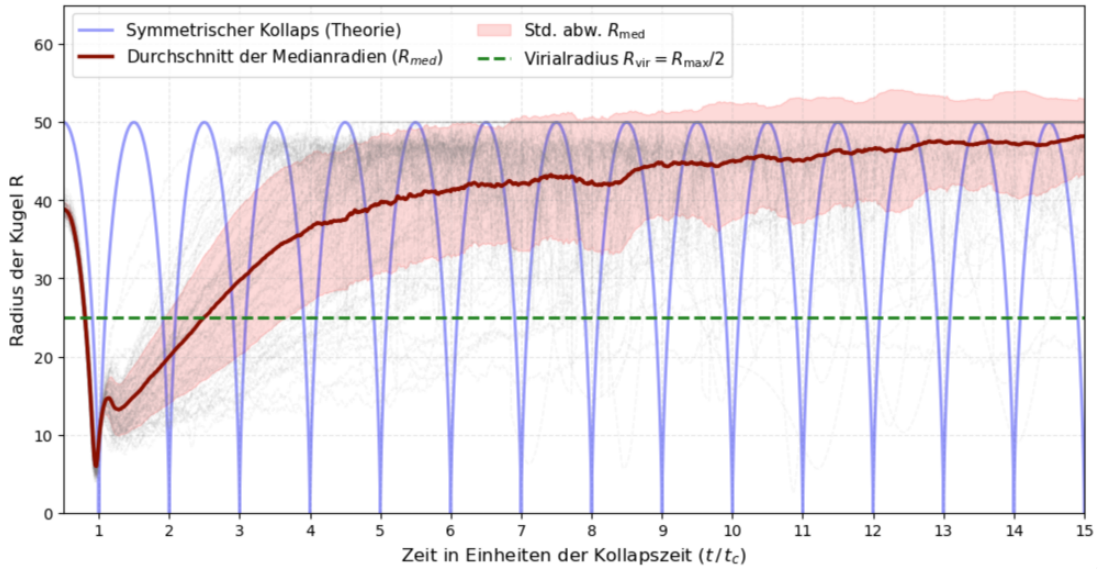
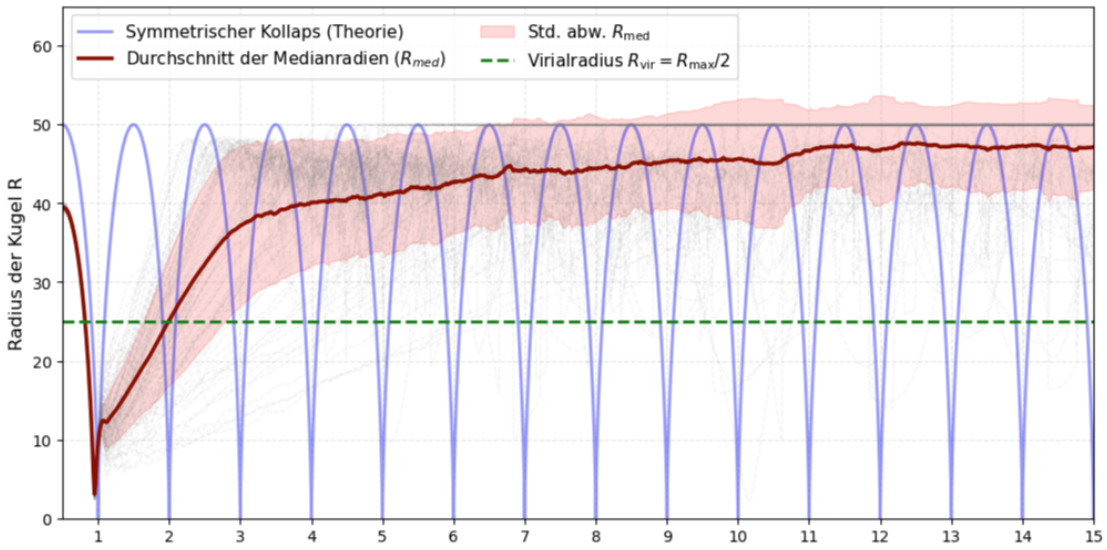



Abbildung 2.9: Ergebnisse der durchgeführten Simulationen für $N = 200$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 7,0$, für das mittlere $\epsilon_{max}/2 = 3,5$ und für das untere $\epsilon_{max}/4 = 1,75$.

Die folgenden Schaubilder zeigen den Simulationsverlauf für $N = 2000$ Teilchen bis zu einer Zeit von $t/t_c = 15$, also das 15-fache der Kollapszeit.



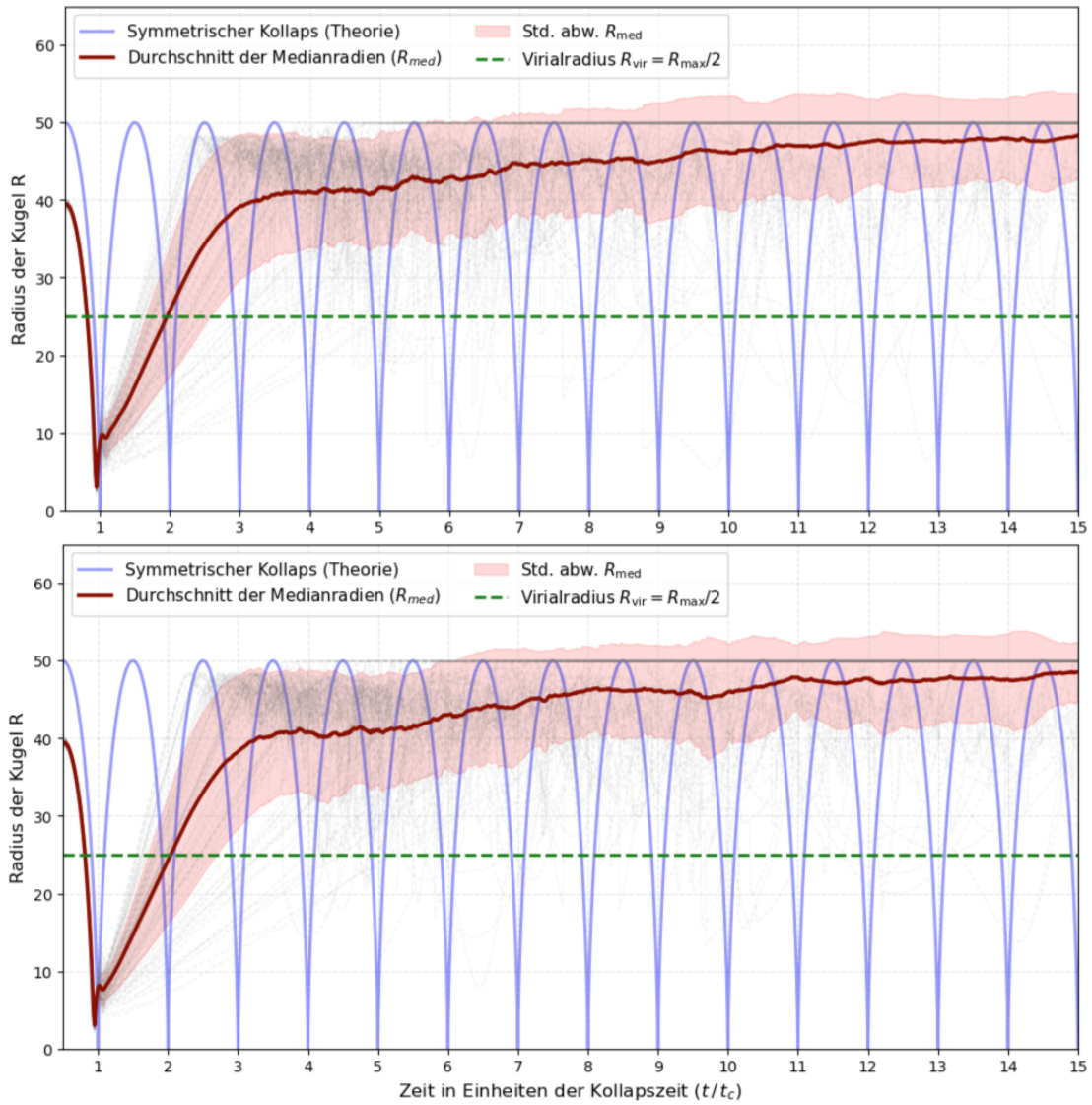



Abbildung 2.10: Ergebnisse der durchgeführten Simulationen für $N = 2000$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 3,2$, für das mittlere $\epsilon_{max}/2 = 1,6$ und für das untere $\epsilon_{max}/4 = 0,8$.

Die grün gestrichelte Linie zeigt den Virialradius an, den das System der Theorie zufolge nach Gl. (2.52) erreichen soll. Außerdem ist in jedem Schaubild das Verhalten eines idealisierten Systems (in Form der Zykloidenkurve), das in einem Punkt kollabiert und sich wieder auf seinen ursprünglichen Radius bei der größten Ausdehnung expandiert, abgebildet. An dieser Stelle sei auch angemerkt, dass wir über die gesamte Simulationszeit nur die Teilchen des jeweiligen Ensembles betrachten, die zum gebundenen System gehören. Denn ein Teil der Teilchen bekommt mit der Zeit aufgrund von gravitativen Nahbegegnungen untereinander genügend kinetische Energie von anderen Teilchen übertragen, sodass deren Gesamtenergie positiv wird

und aus den gebundenen System geschleudert werden [21]. Es kann aber auch vorkommen, dass Teilchen zwar im System gravitativ gebunden sind, aber dennoch auf langgestreckten Bahnen fliegen können. Laut Joyce et al. nimmt auch der Anteil der entweichenden Teilchen in den Simulationen mit der steigender Teilchenzahl $N$ zu [21]. Um die „Ausreißer“ für die Simulation auszusortieren, wurde im Simulationscode ein entsprechender „Filter“ eingebaut, der dafür Sorge trägt und nur das gebundene Teilchenensemble weiterberücksichtigt.
In allen Schaubildern, unabhängig von der Teilchenzahl, lässt sich immer dasselbe Muster erkennen. Alle Teilchenensembles kollabieren zum Zeitpunkt $t_c$ und dehnen sich für einen kurzen Moment zur Zeit $1,2\,t/t_c$ bis knapp zum Virialradius $R_{vir}$ aus. Mit steigender Teilchenzahl wird dieser Peak nach dem ersten Kollaps immer kleiner (vgl. Abb. 2.10 für $N = 2000$). An der Stelle zeigen sich die Unterschiede, je nach gewähltem softening $\epsilon$. Je größer das softening gewählt wird, umso näher kommt das System nach dem ersten Kollaps dem theoretischen Virialradius. Das lässt sich damit erklären dass ein höheres softening eine geringere, abgeschwächte Anziehungskraft im gesamten System ausübt, wodurch sich das Teilchenensemble nach dem ersten Kollaps weiter ausdehnen kann als bei einem kleineren softening, bei dem die gesamte Anziehungskraft unter allen Teilchen größer ist und das System kompakter hält. Nach dem ersten Kollaps zieht sich das Teilchenensemble in allen Schaubildern bis $1,5\,t/t_c$ minimal etwas zusammen und dehnt sich dann wieder aus. Die Abstände zum Zentrum der übrigen die Teilchen, die noch zum gebundenen System gehören, nehmen ab $1,5\,t/t_c$ kontinuierlich zu. Die Kurvenverläufe nähern sich nahezu asymptotisch zwischen $R = 40$ und $R = 50$ gegen Ende der Simulationsdauer. Nur im Schaubild für $N = 50$ mit $\epsilon_{max} = 11,0$ liegt die Mittelwertkurve nach $t/t_c = 15$ bei etwa $R \approx 35$. Der mäßige Anstieg im Vergleich zu den Ergebnissen mit einem softening von $\epsilon_{max}/2$ und $\epsilon_{max}/4$ lässt darauf schließen, dass die geringe Teilchenanzahl und das sehr große softening die Nahbegegnungen unter den Teilchen deutlich abschwächt, wodurch weniger Teilchen herausgeschleudert werden können und das gesamte System eine Ausdehnung nahe dem theoretischen Virialradius behält. Auffällig ist auch, dass das Einpendeln der Kurven auf ein annähernd konstantes Niveau nach einer kürzeren Zeit erfolgt, je kleiner das softening gewählt wurde.
Alle Ergebnisse zeigen eindeutig, dass sich das System nach seinem ersten Kollaps scheinbar mit der Zeit wieder ausdehnt. Das steht wohl scheinbar im Widerspruch mit der Idee des Virialtheorems, das ja besagt, dass ein virialisierter Halo weder kollabiert noch weiter expandiert und sich im Gleichgewicht befindet. Dafür betrachten wir in detaillierter Ausführung die genauen Prozesse, wie ein Halo nach dem Kol-

laps einen Gleichgewichtszustand erreichen kann. In den dargestellten Ergebnissen wurde ein Halo simuliert, der nur aus Dunkle-Materie-Teilchen besteht und keine baryonische Materie (Bestandteil von kosmischen Gas) enthält. Würde ein Halo Gas im Inneren beinhalten, dann wäre es verständlich, dass das Gas einen Gegendruck zur kontinuierlich wirkenden Gravitationskraft aufbauen kann, um es in Gleichgewichtszustand zu bringen oder zumindest annähernd.
Wie kann also trotzdem ein Dunkle-Materie-Halo durch den Prozess der Virialisierung einen Gleichgewichtszustand erreichen? Jedes Teilchen besitzt eine geordnete, (radiale) kinetische Energie, die aber dann während der Kollapszeit in ungeordnete kinetische Energie übergeht und wie ein „Druck“ wirken kann [5]. Damit aber die Energien der Teilchen untereinander „umverteilt“ werden können, braucht es eine Art Streuprozess, der das bewerkstelligt. Die Teilchen werden an Schwankungen des gesamten Gravitationspotentials gestreut, da sich das Potential $\Phi(r,t)$ mit der Zeit verändert und eben nicht konstant bleibt. Der Kosmologe Donald Lynden-Bell hat für diesen Prozess den Begriff *violent relaxation* [26] geprägt. Auf Deutsch würde man es wörtlich mit „gewaltsame Entspannung“ übersetzen. Das mit Absicht verwendete Oxymoron soll betonen, dass der Prozess der Relaxation nicht wie bei manchen Galaxienhaufen länger als das Alter des Universums andauern soll, sondern auf der Zeitskala des freien Falls des Systems. In einem idealisierten System würde die gesamte Masse der Kugel in einem Punkt zusammenfallen, wenn alle Teilchen so angeordnet wären, dass das Gesamtsystem perfekt symmetrisch mit homogener Dichte ist. Alle Teilchen würden zur gleichen Zeit das Zentrum erreichen und wieder ihren ursprünglichen Ort, von dem sie gestartet sind. Hat man dagegen eine leichte asymmetrische Anordnung der Teilchen, wie es in den durchgeführten Simulationen und in der Realität der Fall ist, erzeugt man damit lokale Dichte-Inhomogenitäten, die dann dafür sogen, dass Teilchen von ihren radialen Orbits abweichen und am sich ändernden Gravitationspotential gestreut werden [36]. In [9] wird auch Begriff *shell crossing* verwendet, der beschreiben soll, dass sich die Schalen, in die eine Kugel unterteilt werden kann, aufgrund der Asymmetrie während des Kollaps gegenseitig kreuzen werden. Infolgedessen werden eben auf manche Teilchen ein der Teil der kinetischen Energie übertragen, die aus dem gebundenen System geschleudert werden können, und andere Teilchen werden dementsprechend Energie verlieren. Wenn die Energie eines einzelnen Teilchens in einem zeitlich schwankenden Potential $\Phi(\vec{r}(t),t)$ laut [7] nicht erhalten bleibt, können wir schreiben, dass

$$\frac{dE(t)}{dt} = \frac{1}{2}m\frac{d|\vec{v}(t)|^2}{dt} + m\frac{d}{dt}\Phi(\vec{r}(t),t) \tag{2.98}$$

gilt, wobei die Energie $E(t)$ eines Teilchens lautet:

$$E(t) = \frac{1}{2} m |\vec{v}(t)|^2 + m\Phi(\vec{r}(t), t) \tag{2.99}$$

Den ersten Ausdruck der rechten Seite in Gl. (2.98) können wir umschreiben zu:

$$= \frac{1}{2} \cdot 2\vec{v} \cdot \left( m \frac{d\vec{v}}{dt} \right) = \vec{v} \cdot m \cdot \vec{a} = \vec{v} \cdot \underbrace{\vec{F}}_{=-m\vec{\nabla}\Phi} \tag{2.100}$$

Den zweiten Ausdruck schreiben wir als totale Zeitableitung auf und nutzen dabei die Kettenregel:

$$m \frac{d}{dt} \Phi(\vec{r}(t), t) = m \sum_i^3 \frac{\partial \Phi_i}{\partial x_i} \frac{\partial x_i}{\partial t} + m \frac{\partial \Phi}{\partial t} = m\vec{\nabla}\Phi \cdot \vec{v} + m \frac{\partial \Phi}{\partial t} \tag{2.101}$$

Setzt man beide Ausdrücke in Gl. (2.98) ein, kommt man zu:

$$\frac{dE}{dt} = -m\vec{v} \cdot \vec{\nabla}\Phi + m\vec{v} \cdot \vec{\nabla}\Phi + m \frac{\partial \Phi}{\partial t} \tag{2.102}$$

$$\frac{dE}{dt} = m \frac{\partial \Phi}{\partial t} \tag{2.103}$$

Oder umgeschrieben mit der spezifischen Energie $e = E/m$ nach [26]:

$$\frac{de}{dt} = \frac{\partial \Phi}{\partial t} \tag{2.104}$$

Damit wäre bewiesen, dass die Energie eines jeden Teilchens nicht erhalten bleibt, wenn das Gravitationspotential während der *violent relaxation* nicht konstant bleibt und sich zeitlich ändert, obwohl die Gesamtenergie aller betrachteten Teilchen erhalten bleibt [36]. Labini et al. erwähnen in ihrer Publikation, dass der Ausgang der *violent relaxation* „weniger gewaltsam“ ablaufen würde, wenn die Teilchen zu Beginn eine Anfangsgeschwindigkeit hätten, was auch als Kollaps mit „warmen Anfangsbedinungen“ bezeichnet wird [41]. Die Anfangsbedingung der Teilchen, die hier mit $v = 0$ simuliert wird, kann dementsprechend als Kollaps mit „kalten Anfangsbedinungen“ angesehen werden. Mit der anfänglichen kinetischen Energie könnten die Teilchen eben einen „Druck“ ausüben könnten, um ein stabileres Systems auszubilden (vgl. Kap. 2.4.3).

Lynden-Bell argumentiert, dass die *violent relaxation* auf einer dynamischen Zeitskala $t_{dyn}$ abläuft und damit viel schneller als normale kollisionsfreie Relaxationspro-

zesse, die viel länger andauern (je nach Größe des betrachteten Systems) [9]. Die dynamische Zeitskala entspricht in diesem Kontext in etwa der Zeit des freien Falls $t_{ff}$, sodass

$$\tau_{vio} \approx t_{dyn} \approx t_{ff} \tag{2.105}$$

gilt. Die Zeitdauer entspricht gleichzeitig in etwa der Zeit, in der das Gravitationspotential heftig schwankt.

### 2.4.2 Virialisierte Halos nach *violent relaxation*

Wenn der Prozess der *violent relaxation* in etwa die Zeit für den freien Fall benötigt, müsste sich das System direkt nach dem Kollaps gemäß dem Virialtheorem nach Gl. (2.41) virialisiert haben. Dies können wir darstellen, indem die Viralsumme wieder gegen die Kollapszeit $t/t_c$ aufgetragen wird. Wie schon bei den vorherigen Ergebnissen zeigt die hier in den Schaubildern dunkelgrüne, durchgezogene Kurve den Mittelwert der 100 simulierten Durchläufe mit jeweils einer Standardabweichung $\pm 1\sigma$ an. Zusätzlich wurde auch noch mit einer rot gestrichelten Linie der Endwert nach Gl. (2.31) abgebildet, da das klassische Virialtheorem mit $2\langle T\rangle + \langle V\rangle = 0$ erst nach sehr langen Zeitdauern $\tau$, also $\lim_{\tau\to\infty} = 0$ ergibt. Selbst nach einer Simulationsdauer von $t/t_c = 15$ hat das Skalarprodukt von Impuls und Ort aller Teilchen $[\vec{p}_i(\tau) \cdot \vec{x}_i(\tau)]/\tau$ einen endlichen Wert.
Da in allen Simulationen ja auch ein softening verwendet wird, bietet es sich an dieser Stelle an, das Virialtheorem hinsichtlich des softenings zu modifizieren [1]. Die Herleitung erfolgt analog bis Gl. (2.33). Aber nun wird die Kraft $\overrightarrow{F_{ij}}$ mit dem softening $\epsilon$ analog zur Kraftformel in Gl. (2.72) umgeschrieben zu:

$$\overrightarrow{F_{ij}} = -\frac{Gm_i m_j}{\left(|\vec{x_i} - \vec{x_j}|^2 + \epsilon^2\right)^{3/2}}(\vec{x_i} - \vec{x_j}) \tag{2.106}$$

Summiert man wiederum über alle Kräfte auf, die auf ein Teilchen $i$ wirken, erhält man die Gesamtkraft $\vec{F_i} = \sum_{i\neq j} \overrightarrow{F_{ij}}$ und nutzt im zweiten Schritt wieder das dritte Newton'sche Axiom aus, wodurch analog zu Gl. (2.38) der folgende Ausdruck resultiert [1]:

$$\sum_i \vec{F_i} \cdot \vec{x_i} = -\sum_{i<j} \frac{Gm_i m_j}{\left(|\vec{x_i} - \vec{x_j}|^2 + \epsilon^2\right)^{3/2}} |\vec{x_i} - \vec{x_j}|^2 \tag{2.107}$$

Der besseren Übersicht halber führen wir die Notation $|\vec{x_i} - \vec{x_j}|^2 \equiv |\overrightarrow{x_{ij}}|^2 \equiv x_{ij}^2$ ein und addieren $x_{ij}^2$ mit $+\epsilon^2 - \epsilon^2$:

$$\sum_i \overrightarrow{F_i} \cdot \vec{x_i} = -\sum_{i<j} \frac{G m_i m_j (x_{ij}^2 + \epsilon^2 - \epsilon^2)}{\left(x_{ij}^2 + \epsilon^2\right)^{3/2}} \tag{2.108}$$

Den Bruch in Gl. (2.108) teilen wir nun auf und erhalten:

$$\sum_i \overrightarrow{F_i} \cdot \vec{x_i} = \underbrace{-\sum_{i<j} \frac{G m_i m_j \cdot (x_{ij}^2 + \epsilon^2)}{\left(x_{ij}^2 + \epsilon^2\right)^{3/2}}}_{=V_{soft}} \underbrace{-\sum_{i<j} \frac{G m_i m_j \cdot (-\epsilon^2)}{\left(x_{ij}^2 + \epsilon^2\right)^{3/2}}}_{\equiv C_{soft}} \tag{2.109}$$

Der rechte Term, der mit $V_{soft}$ beschriftet wurde, entspricht nach Kürzen von $x_{ij}^2 + \epsilon^2$ analog zu Gl. (2.71) dem Plummer-Potential, also das Potential $V$ unter Berücksichtigung des softenings $\epsilon$. Der zweite Term wurde mit $C_{soft}$ beschriftet und bezeichnet einen weiteren Endwert, der bei der Darstellung der Virialtheorems beachtet werden muss.
In Kurzform lautet die Gleichung für das „modifizierte“ Virialtheorem:

$$\sum_i \overrightarrow{F_i} \cdot \vec{x_i} = V_{soft} - C_{soft} \tag{2.110}$$

Setzt man den Ausdruck wieder in Gl. (2.31) ein und berechnet das zeitliche Mittel, erhält man:

$$\frac{p_i(\tau) \cdot x_i(\tau)}{\tau} = \langle 2T + V_{soft} - C_{soft} \rangle \tag{2.111}$$

bzw.

$$\langle 2T + V_{soft} \rangle = \frac{p_i(\tau) \cdot x_i(\tau)}{\tau} + \langle C_{soft} \rangle \tag{2.112}$$

Für $\tau \to \infty$ wird der erste Ausdruck auf der rechten Seite null ergeben, sodass nur noch $\langle C_{soft} \rangle$ übrigbleibt.
Die folgenden Schaubilder zeigen die Ergebnisse der gegen die Zeit aufgetragene Virialsumme für $N = 50$ Teilchen bis zu einer Zeit von $t/t_c = 15$, wieder für drei verschiedene softening-Werte.

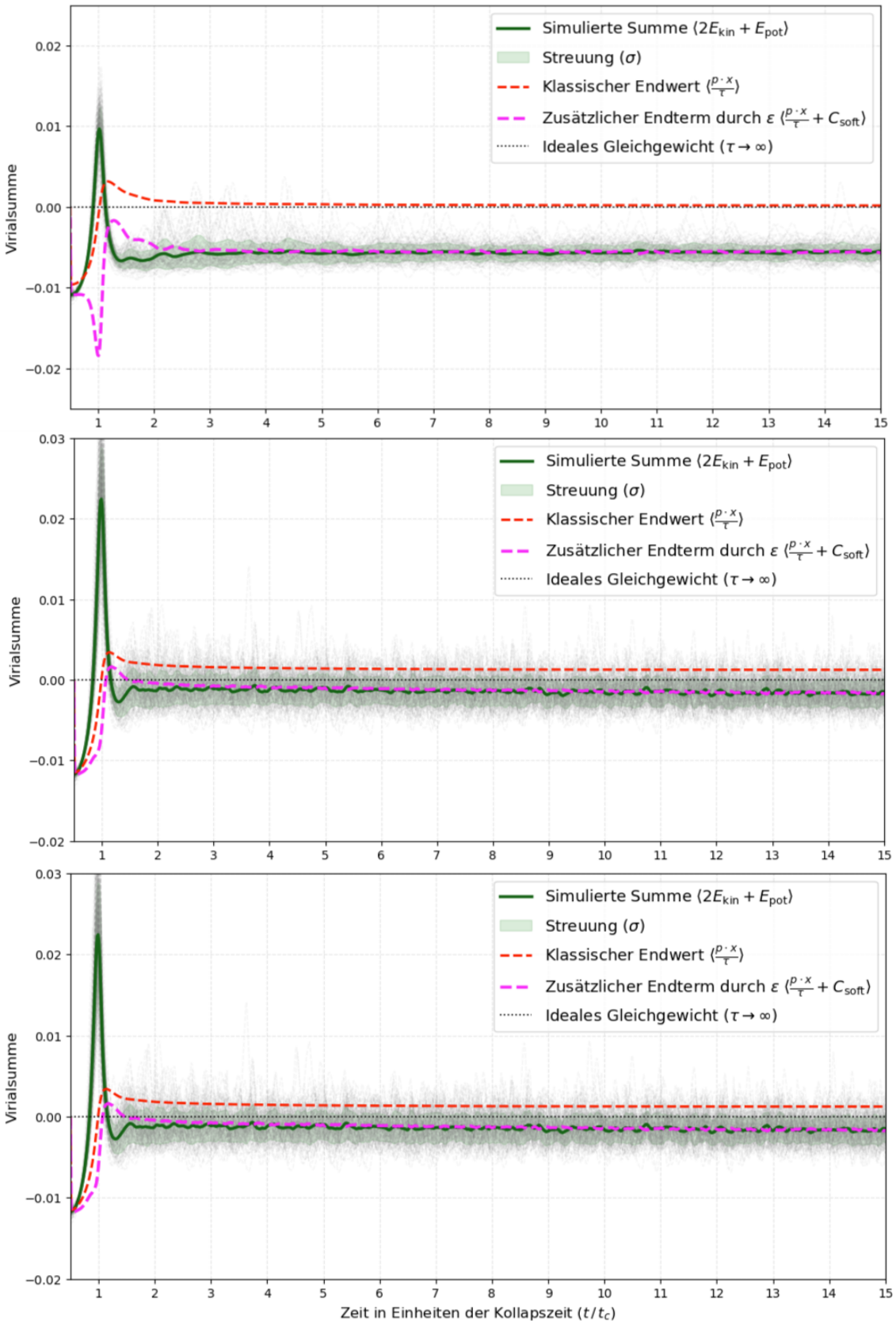
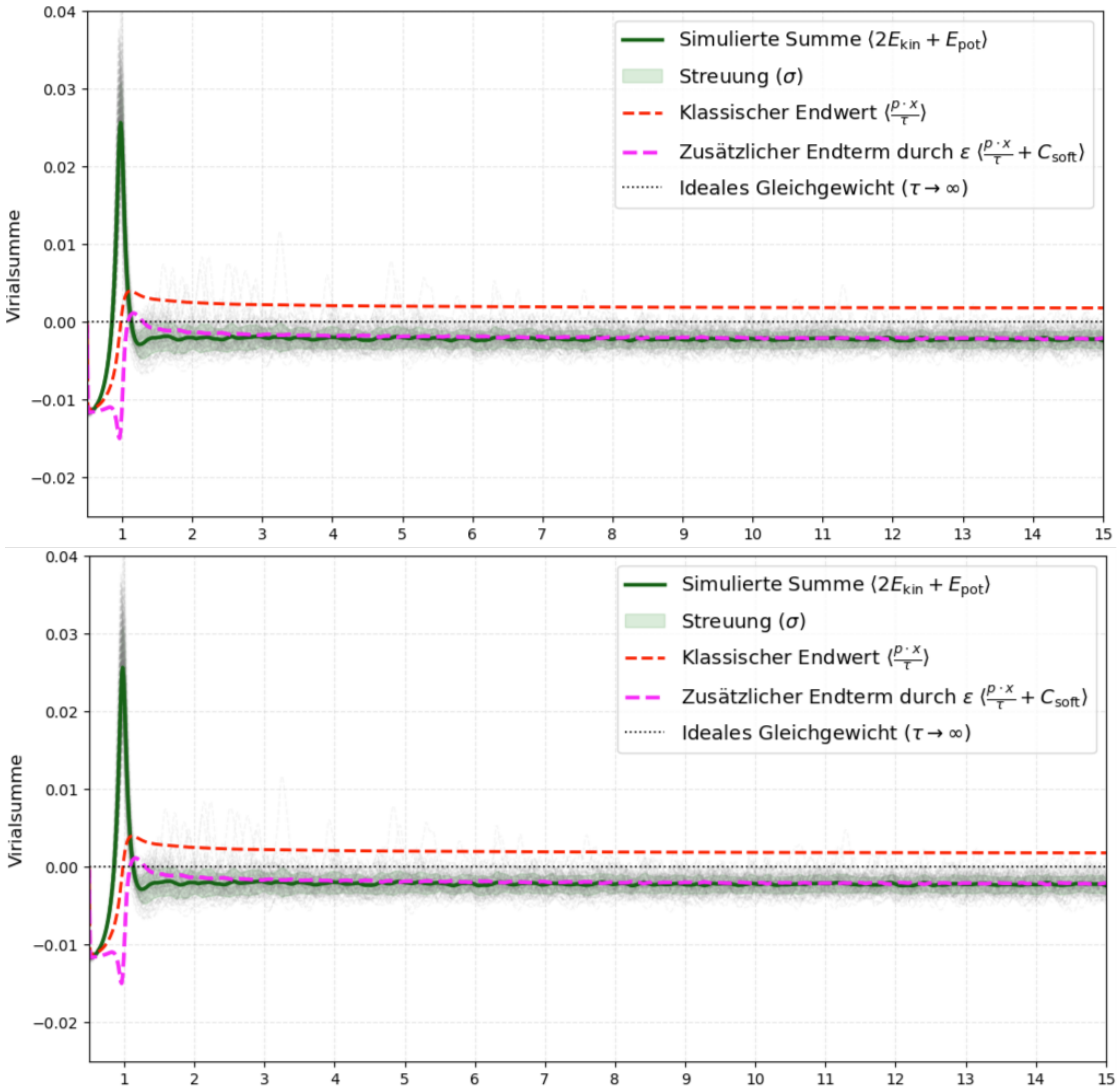



Abbildung 2.11: Darstellung der gegen die Zeit $t/t_c$ aufgetragene Virialsumme für $N = 50$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 11,0$, für das mittlere $\epsilon_{max}/2 = 5,5$ und für das untere $\epsilon_{max}/4 = 2,75$.

Die folgenden Schaubilder zeigen die Ergebnisse der gegen die Zeit aufgetragene Virialsumme für $N = 100$ Teilchen bis zu einer Zeit von $t/t_c = 15$, wieder für drei verschiedene softening-Werte.



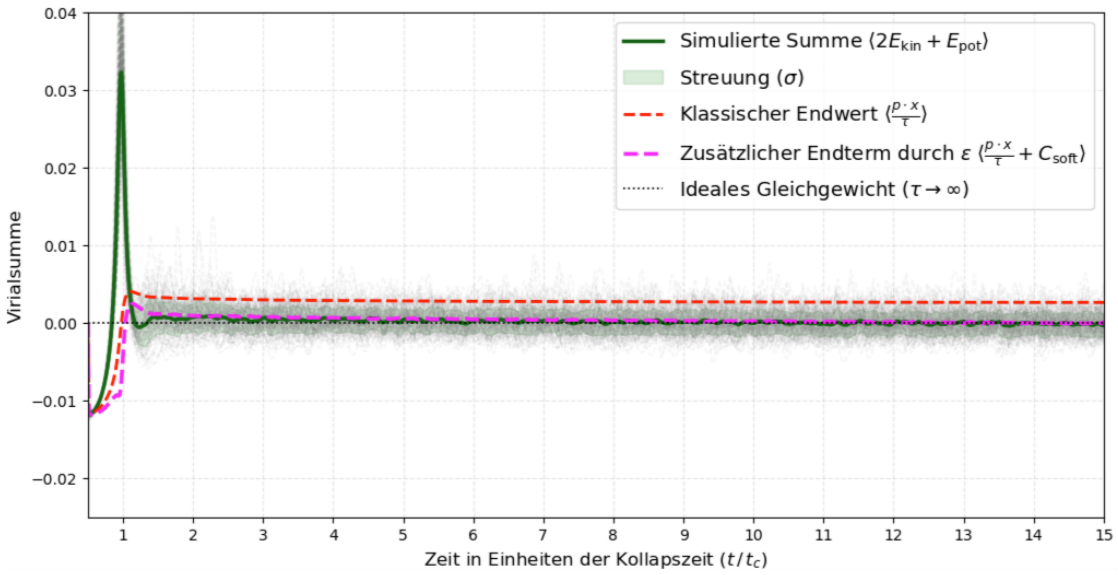
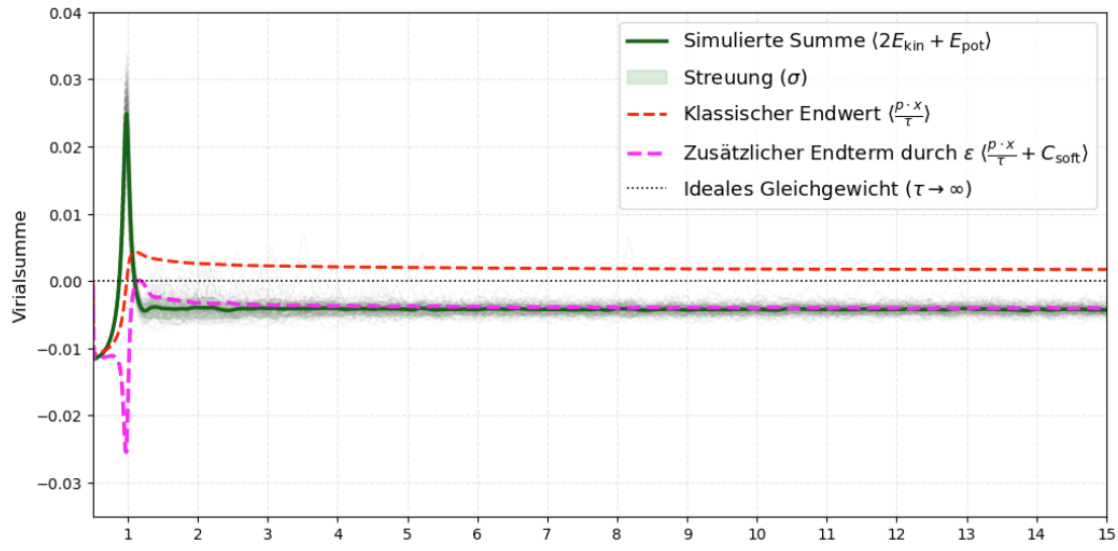



Abbildung 2.12: Darstellung der gegen die Zeit $t/t_c$ aufgetragene Virialsumme für $N = 100$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 8,7$, für das mittlere $\epsilon_{max}/2 = 4,35$ und für das untere $\epsilon_{max}/4 = 2,18$.

Die folgenden Schaubilder zeigen die Ergebnisse der gegen die Zeit aufgetragene Virialsumme für $N = 200$ Teilchen bis zu einer Zeit von $t/t_c = 15$, wieder für drei verschiedene softening-Werte.



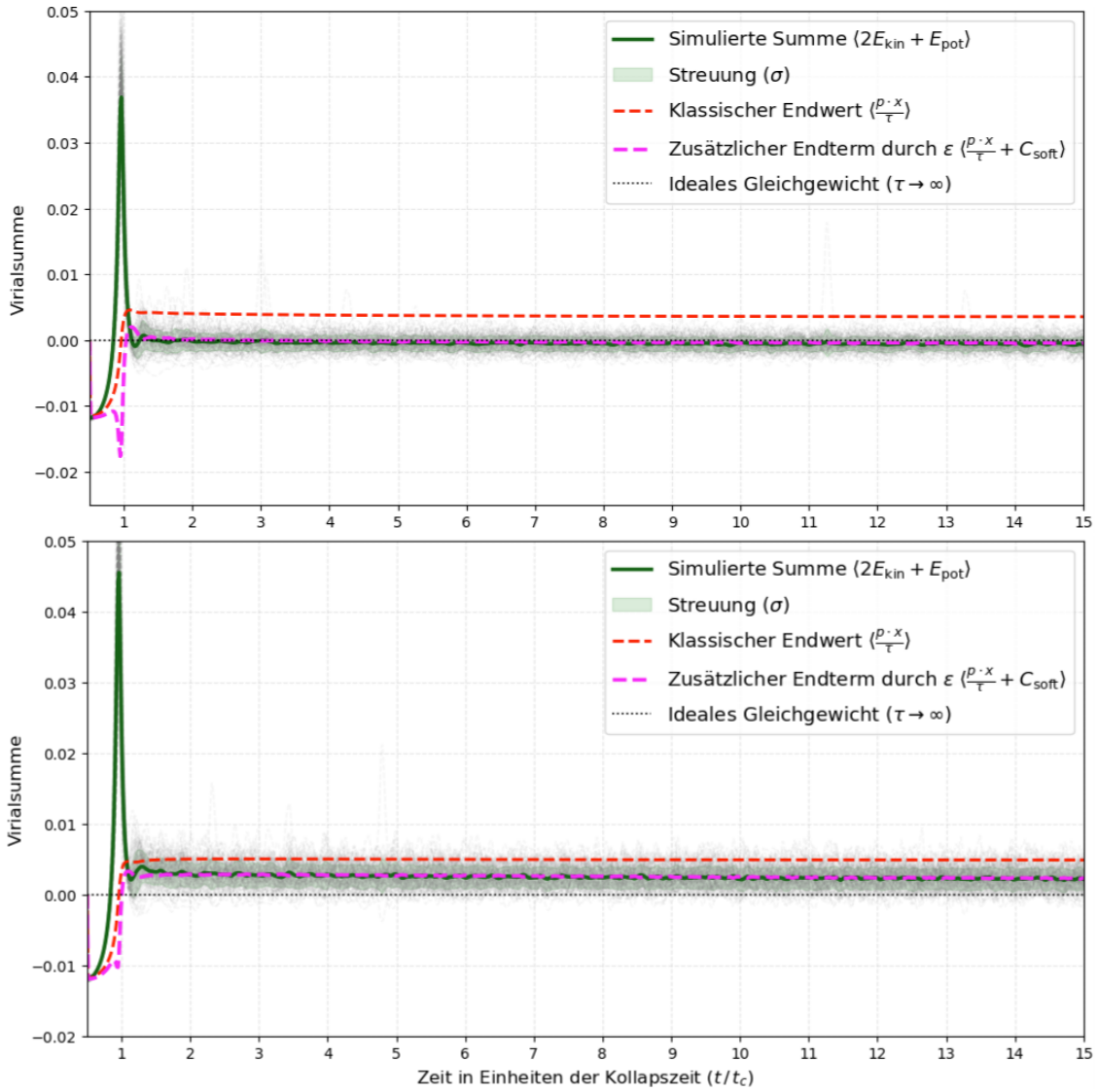



Abbildung 2.13: Darstellung der gegen die Zeit $t/t_c$ aufgetragene Virialsumme für $N = 200$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 7,0$, für das mittlere $\epsilon_{max}/2 = 3,5$ und für das untere $\epsilon_{max}/4 = 1,75$.

Die folgenden Schaubilder zeigen die Ergebnisse der gegen die Zeit aufgetragene Virialsumme für $N = 2000$ Teilchen bis zu einer Zeit von $t/t_c = 15$, wieder für drei verschiedene softening-Werte.

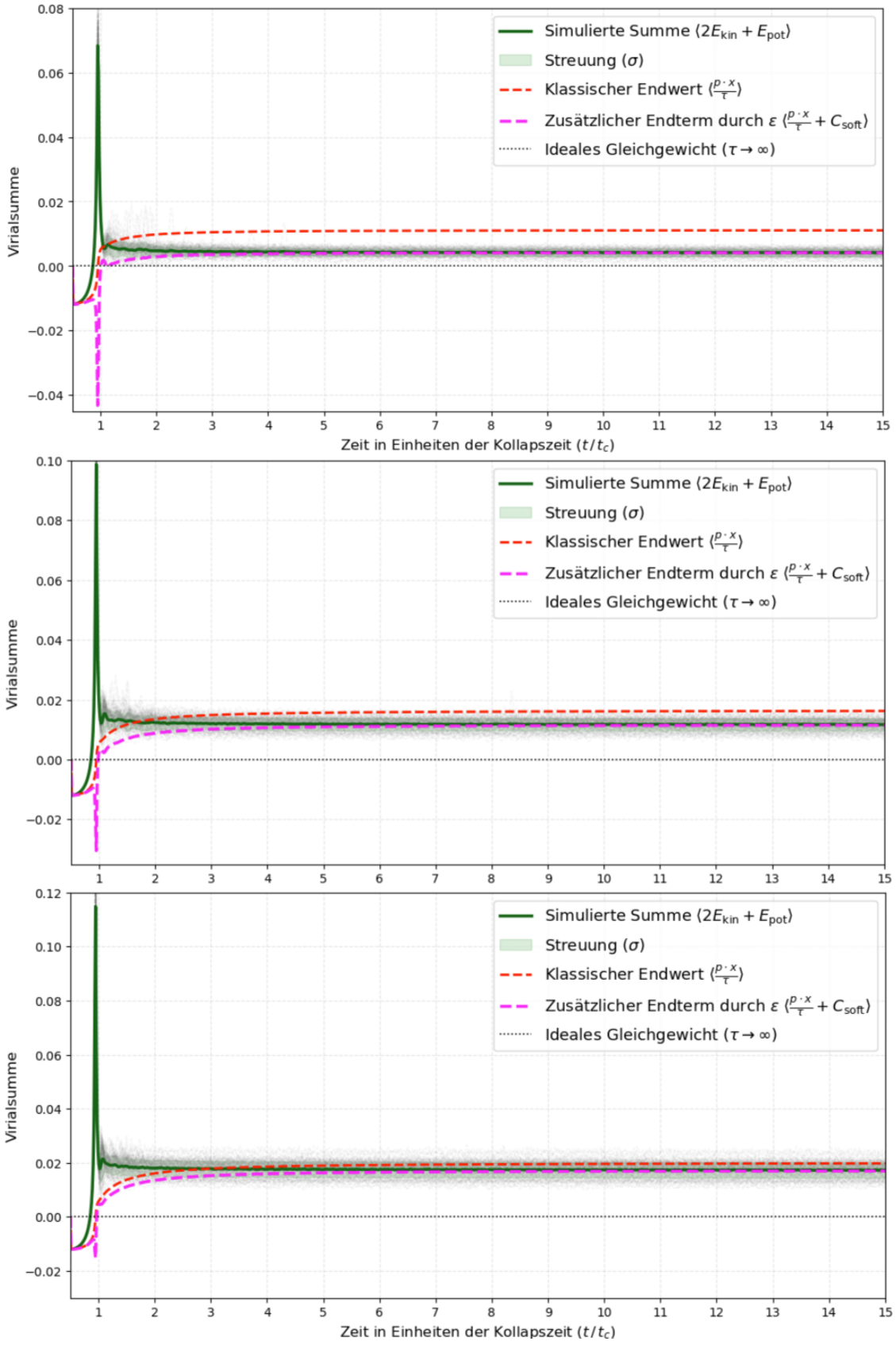



Abbildung 2.14: Darstellung der gegen die Zeit $t/t_c$ aufgetragene Virialsumme für $N = 2000$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 3,2$, für das mittlere $\epsilon_{max}/2 = 1,6$ und für das untere $\epsilon_{max}/4 = 0,8$.

In allen Schaubildern ist beim ersten Kollaps $t/t_c = 1,0$ ein sehr großer Peak zwischen knapp $0,010$ bei $N = 50$ und etwa $0,115$ bei $N = 2000$ zu sehen, der mit abnehmenden softening größere Werte erreicht. Der Ausschlag hierbei ist darauf zurückzuführen, dass die gesamte potentielle Energie der Teilchen in kinetische Energie umwandelt wird und auch der Vorfaktor 2 gemäß dem Virialtheorem mit einhergeht. Genauso ist bei allen Ergebnissen deutlich zu sehen, dass die grüne Kurve der Virialsumme sich nach etwa $1,1 - 1,2\, t/t_c$ auf einen konstanten Wert einpendelt. Der eingependelte Werte verschiebt sich in positive y-Richtung, je kleiner das softening gewählt wurde, womit auch die potentielle Energie einen weniger negativen Gesamtwert annehmen kann. Damit bestätigen die Ergebnisse, dass die *violent relaxation* beim Kollaps tatsächlich auf etwa der Zeitskala des freien Falls stattfindet und somit die Virialisierung des Halos früh vollendet ist - vgl. [9, 26, 35]. Auch die nahezu perfekte Überlappung der pink gestrichelten Kurve mit der Virialsumme durch die Berücksichtigung des hergeleiteten Endwerts $C_{soft}$ zeigt, dass die Modifizierung des Virialtheorems funktioniert hat.

### 2.4.3 Meta- und Instabilität von Dunkle-Materie-Halos

Wenn also die Virialisierung auf der Zeitskala des freien Falls durchlaufen ist, ist aber immer noch nicht geklärt, warum sich das System in den Ergebnissen nach $1,5\ t/t_c$ scheinbar wieder ausdehnt (Anstieg des Mittelwertskurve). In sehr vielen wissenschaftlichen Publikationen und Büchern über die Halokollaps-Forschung beispielsweise in [5, 9, 21, 40] findet man häufig den Begriff *quasi-steady-state* oder auf deutsch „Quasistationäres (oder metastabiles) Gleichgewicht“. Der Begriff soll darauf hindeuten, dass Dunkle-Materie-Halos (unter Vernachlässigung von kosmischen Gas im Innern) eigentlich keinen Gleichgewichtszustand erreichen können. Das ergibt sich deshalb, weil Dunkle-Materie-Halos selbst-gravitierende Systeme darstellen und im thermodynamischen Sinne niemals ein Gleichgewichtszustand einnehmen werden [5]. Das lässt mit dem Virialtheorem zeigen, dem das System genügt. Die ungeordnete kinetische Energie kann als Maß für die Temperatur herangezogen werden. Für die isochore Wärmekapazität $C_V$, die man über die partielle Zeitableitung der inneren Energie $U$ umschreiben kann, ergibt sich nach [5]

$$C_V = \left(\frac{\partial U}{\partial T}\right)_V \propto \frac{\partial \langle E_{tot}\rangle}{\partial \langle E_{kin}\rangle} = \frac{\partial}{\partial \langle E_{kin}\rangle}\left(\langle E_{kin} + E_{pot}\rangle\right) \tag{2.113}$$

$$C_V = \frac{\partial}{\partial \langle E_{kin}\rangle}(\langle E_{kin} - 2E_{kin}\rangle) \tag{2.114}$$

wobei kurzzeitig die Notation ($T \longleftrightarrow E_{kin}$ und $V \longleftrightarrow E_{pot}$) gewechselt wurde und nach Gl. (2.49) aus dem Virialtheorem die Beziehung $E_{pot} = -2E_{kin}$ ausgenutzt wurde. Final ergibt die Wärmekapazität damit

$$C_V = \frac{\partial}{\partial \langle E_{kin} \rangle}(-\langle E_{kin} \rangle) = -1 \tag{2.115}$$

Das Ergebnis zeigt, dass selbst-gravitierende Systeme eine negative Wärmekapazität besitzen. Die Vorstellung einer negativen Wärmekapazität ist gemäß den Alltagserfahrungen eher kontraintuitiv. Denn wenn ein Körper mit einer bestimmten Temperatur Energie in Form von Wärme abstrahlt, kühlt es dadurch ab. Bei selbstgravitierenden Systemen passiert das Gegenteil: Verlieren sie Energie, weil sie Teilchen nach außen abgeben, heizen sie sich auf. Oder zugespitzt ausgedrückt: Selbstgravitierende Systeme kühlen, indem sie heißer werden. Der zunehmende Energieverlust führt dazu, dass das System kontrahiert, also der Radius eines sphärischen Systems sinkt. Dadurch wird die potentielle Energie noch negativer wird, da $E_{pot} \propto 1/r$ skaliert. Und weil das System weiterhin dem Virialtheorem ($2E_{kin} = -E_{pot}$) genügt, muss die kinetische Energie ansteigen. Steigt aber die kinetische Energie, bewegen sich die Teilchen schneller und können im Laufe der Zeit leichter aus dem System entweichen. Dieser Prozess läuft solange weiter, bis auf langfristiger Zeitskala betrachtet das gesamte System „verdampft“ [6]. Das Ende dieses Prozesses wird in der Fachliteratur [25, 41] auch als *gravothermal collapse/catastrophe* (zu deutsch: *gravothermaler Kollaps/Katastrophe*) bezeichnet.
In der Fachliteratur [2, 21, 40, 41] wird unter anderem erwähnt, dass sich die räumliche Struktur der Teilchen während des Quasi-Gleichgewichtszustands als eine Art Kern-Halo-Struktur (engl. *core-halo-structure*) beschreiben lässt. Konkret meinen sie, dass es einen Bereich mit hoher Dichte an Teilchen gibt, der als Kern gesehen wird, und einen vom Kern weiter entfernten Bereich, der als diffuser Halo betrachtet wird, in dem nur noch ein Bruchteil der Teilchen um den Kern herum verstreut sind. Die Abbildung 2.15 zeigt von Joyce et al. vier Momentaufnahmen (zu den Zeitpunkten $t/t_c = 0; 0.9; 1.0 \text{ und } 2.5$) eines Systems mit $N = 4096$ Teilchen von Anfang des Kollaps bis zur Ausbildung einer Kern-Halo-Struktur.

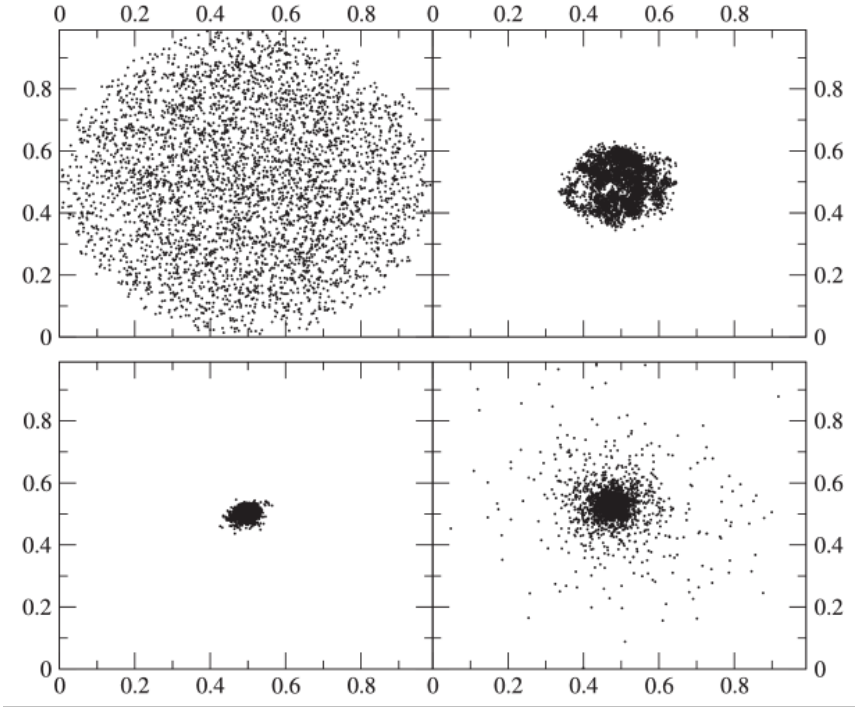



Abbildung 2.15: Vier Momentaufnahmen (engl. *snapshots*) zu verschiedenen Zeitpunkten ($t/t_c = 0; 0.9; 1.0$ und $2.5$ von l. o. nach r. u.) eines mit $N = 4096$ Teilchen simulierten Halo-Kollapses; entnommen aus [21].

Die letzte Momentaufnahme bei $t/t_c = 2,5$ zeigt eindrücklich den dichten Kern mit sehr vielen, nahen Teilchen und den diffusen Bereich, in der die Dichte der Teilchen vom Zentrum weg abfällt. Wenn Halos als ein System in einem Quasi-Gleichgewichtszustand nach Abschluss der Virialisierung angesehen werden können, kann man laut [40, 41] eine Zeitspanne definieren, in der sie auch in der Kern-Halo-Struktur verharren. Diese beträgt:

$$t_{ff} < t < t_2 \tag{2.116}$$

$t_2$ wird in diesem Kontext auch als Relaxationszeit für Zwei-Körper-Kollisionen bezeichnet, die auf langfristiger Skala geschehen. Konkret bedeutet das, dass für die Zeit $t > t_2$ hier die Eigendynamik des selbst-gravitierenden Systems samt der Evaporation der Teilchen folgt. Die Zeit $t_2$ erfolgt nach [7] gemäß der Gleichung:

$$t_2 \approx \frac{N}{10\, ln(N)}\, t_{cross} \tag{2.117}$$

$t_{cross}$ bezeichnet hier die Zeit, die ein Teilchen (oder ein Stern im Kontext von Bin-

ney) braucht, um vom äußeren Rand eines Systems (z.B. Galaxie) bis zum Zentrum zu gelangen. In unserem Kontext ist $t_{cross}$ gleichbedeutend mit dem bereits eingeführten Begriff der Freifallzeit $t_{ff}$. Erstaunlicherweise hängt das Ergebnis in Gl. (2.117) weder von der Masse noch der Größe eines Halos ab, sondern nur von der Teilchenzahl, aus dem ein Halo (oder eine Galaxie) besteht. Für eine Galaxie mit typischerweise $N \approx 10^{11}$ Sternen und einer Freifallzeit $t_{ff} \approx 1 \cdot 10^8\, a = 100$ Myr beträgt $t_2 \approx 1 \cdot 10^{11}\, a = 10$ Gyr, sodass das System praktisch als kollisionslos betrachtet werden kann [7]. Bei Systemen mit kleinerer Teilchenzahl, z.B. (offenen) Kugelsternhaufen (engl. *globular clusters* und *open clusters*) mit $N \approx 10^2 - 10^5$ Sternen spielen die Nahbegegnungen der Sterne für die Struktur der Systeme eine wesentliche Rolle [7].
Für unsere Modellsimulation können wir anhand der verschiedenen Teilchenzahlen ebenso die Zeit $t_2$ bestimmen. Tabelle 2.3 zeigt eine Übersicht zu den verschiedenen Zeiten in Abhängigkeit der Teilchenzahl $N$:

Tabelle 2.3: Übersicht zu den verschiedenen Zeiten $t_2$ in Abhängigkeit von unterschiedlichen Teilchenzahlen $N$.

| N | $\frac{N}{10\,ln(N)}$ | $t_2$ in Einheiten von $t_{ff}$ | $t_2$ in Einheiten von $t_c$ |
|---|---|---|---|
| 50 | 1,27 | 1,27 | 1,14 |
| 100 | 2,17 | 2,17 | 1,58 |
| 200 | 3,77 | 3,77 | 2,39 |
| 2000 | 26,31 | 26,31 | 13,65 |
| 10000 | 108,57 | 108,57 | 54,79 |

$t_2$ wurde hier auch in Einheiten der Kollapszeit $t_c$ angegeben, da die Ergebnisse auf der Zeitachse in Einheiten von $t/t_c$ angegeben worden sind. Die Umrechnung von $t_{ff}$ von $t_c$ erfolgt durch

$$t_2 \,\text{in Einheiten von}\, t_c = \frac{t_2 \,\text{in Einheiten von}\, t_{ff}}{2} + 0,5 \tag{2.118}$$

Die Umrechungsformel folgt aus dem Grund, weil die Achsenskalierung in unserem Fall so gewählt wurde, dass bei $t = 0 \rightarrow t/t_c = 0,5$ bzw. $t/t_{ff} = 0$ beim *turn-around*-Punkt gilt, und die Freifallzeit der Hälfte der Kollapszeit entspricht ($t_{ff} = t_c/2$). Wie anhand der Tabelle zu sehen ist, steigen die Zeiten für $t_2$ mit wachsender Teilchenzahl. Theoretisch betrachtet würde für den Fall, dass $N \rightarrow \infty$ strebt, die Zeit $t_2$ auch unendlich lange andauern.

Gemäß den Werten für die Freifallzeit $t_{ff}$ aus der Tabelle 2.2 lassen sich auch die realen Werte für $t_2$ ermitteln und damit auch die Zeitspanne des Halos im Quasi-Gleichgewichtszustand. Tabelle 2.4 fasst die Werte $t_2$ [in Gyr] für verschiedene Halomassen und -radien in Abhängigkeit der simulierten Teilchenzahlen $N$ zusammen.

Tabelle 2.4: Übersicht der berechneten Werte für die Relaxationszeit $t_2$ [in Gyr] für verschiedene Halomassen und -radien in Abhängigkeit der simulierten Teilchenzahl.

| $N$ | $R$ [kpc] | $10^{10}\,M_\odot$ | $10^{11}\,M_\odot$ | $10^{12}\,M_\odot$ | $10^{13}\,M_\odot$ |
|---|---|---|---|---|---|
| 50 | 50 | 2,35 | 0,75 | 0,24 | 0,07 |
| | 100 | 6,65 | 2,11 | 0,66 | 0,22 |
| | 200 | 18,81 | 6,17 | 1,88 | 0,60 |
| 100 | 50 | 4,01 | 1,28 | 0,41 | 0,13 |
| | 100 | 11,34 | 3,60 | 1,13 | 0,37 |
| | 200 | 32,14 | 10,12 | 3,21 | 1,02 |
| 200 | 50 | 6,97 | 2,22 | 0,72 | 0,23 |
| | 100 | 19,75 | 6,26 | 1,96 | 0,64 |
| | 200 | 55,83 | 17,64 | 5,58 | 1,77 |
| 2000 | 50 | 48,67 | 15,52 | 4,99 | 1,58 |
| | 100 | 137,86 | 43,67 | 13,68 | 4,47 |
| | 200 | 389,65 | 127,86 | 38,93 | 12,37 |

Wir können die Tatsache, dass im Laufe der Simulationszeit Teilchen aus dem System entweichen, mit einer kumulativen Entfernungsverteilung darstellen. Konkret bedeutet es, dass man die Verteilung der Teilchen innerhalb der Kugel von 0 bis zum maximalen Radius $R$ zu verschiedenen Zeitpunkten der Simulation darstellt und die Veränderungen dieser räumlichen Verteilung beobachtet. Die folgenden Abbildungen zeigen für alle Teilchenzahlen jeweils die Ergebnisse der Entfernungsverteilung der Teilchen, wieder in Abhängigkeit des gewählten softenings. Zur Darstellung der Entfernungsverteilung wurden verschiedene Zeitpunkte ($t/t_c = 0.5, 1.0, 2.0, 3.0, 5.0, 7.0, 10.0$) gewählt. Die $y-$Achse gibt die Anzahl der Teilchen $N(< r)$ an, die sich zu einem Zeitpunkt $t$ bei einem Abstand $r$ innerhalb der Kugel befinden, normiert auf die simulierte Gesamtteilchenzahl $N_{gesamt}$ vor dem ersten Kollaps.
Die folgenden Schaubilder zeigen die Ergebnisse für $N = 50$ Teilchen:

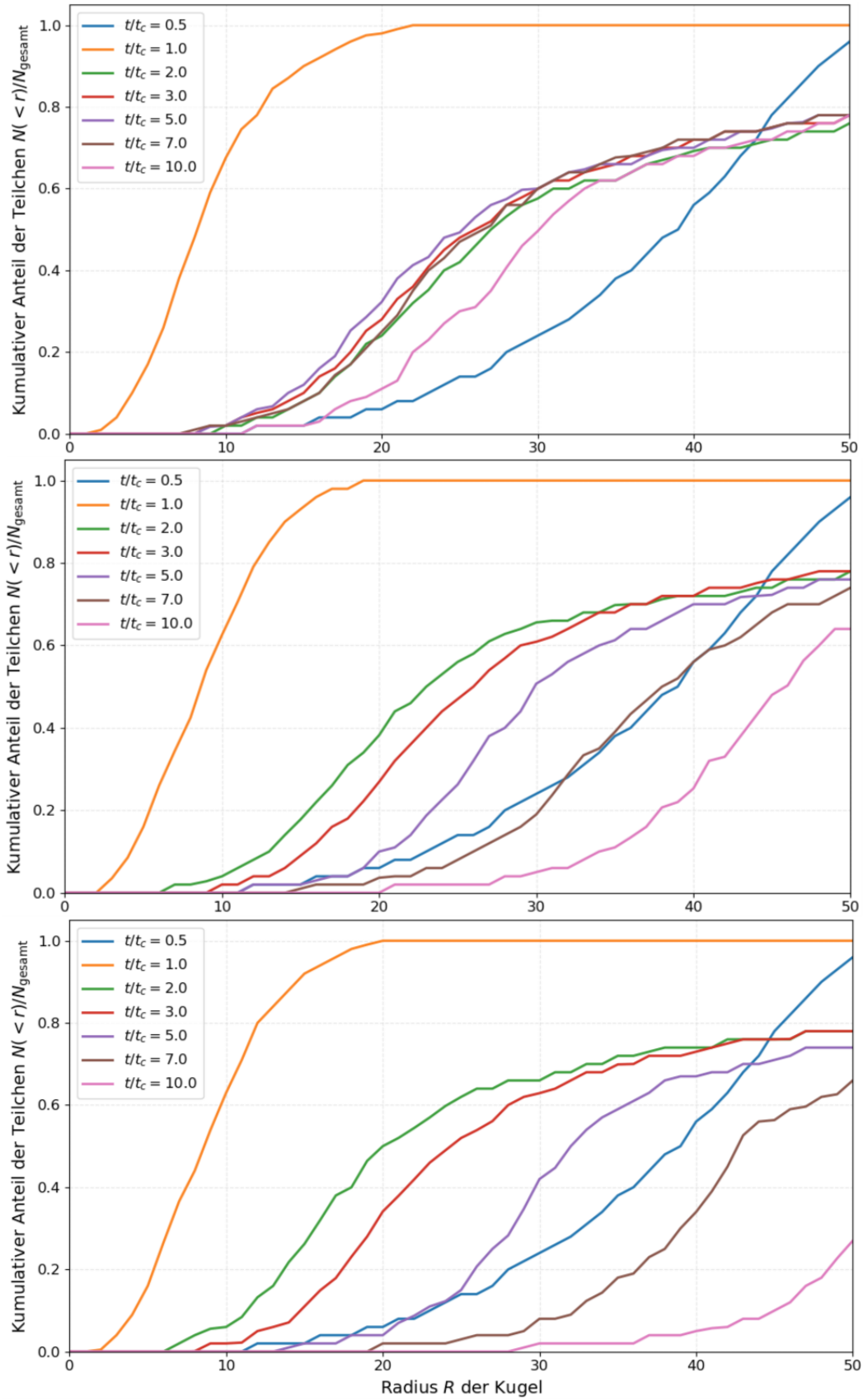
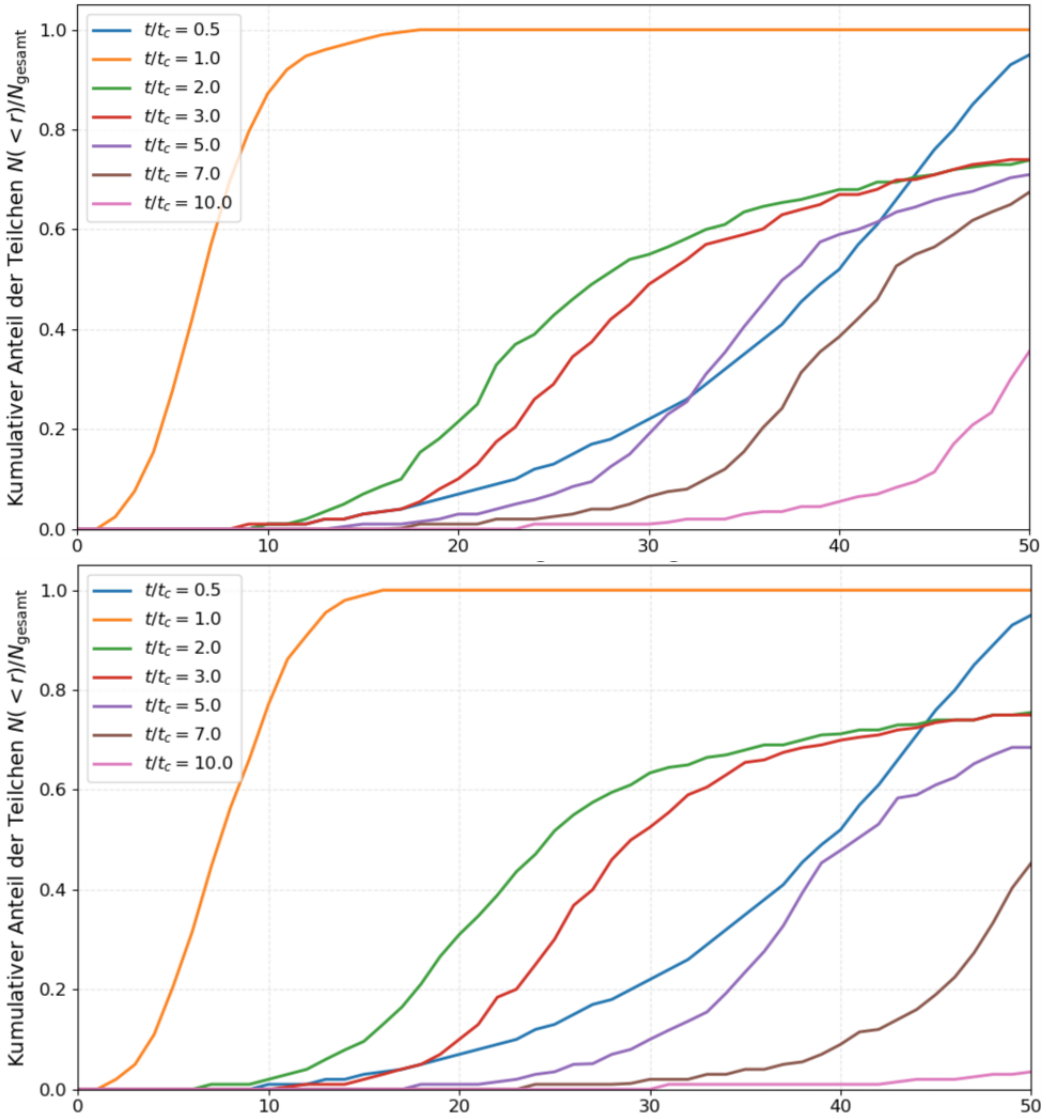



Abbildung 2.16: Darstellung der Entfernungsverteilung für $N = 50$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 11,0$, für das mittlere $\epsilon_{max}/2 = 5,5$ und für das untere $\epsilon_{max}/4 = 2,75$.

Die folgenden Schaubilder zeigen die Ergebnisse für $N = 100$ Teilchen:



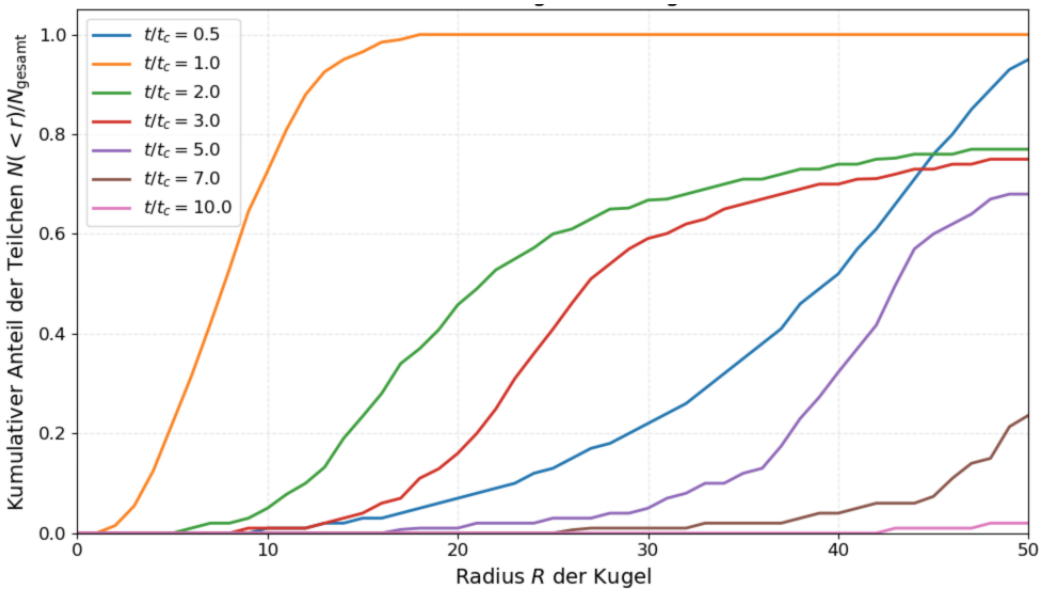
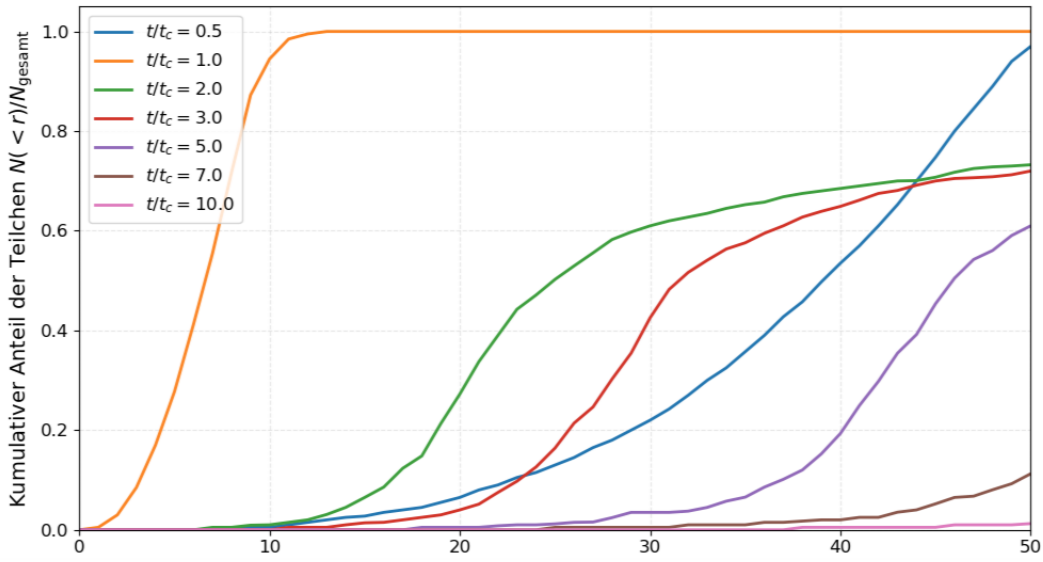



Abbildung 2.17: Darstellung der Entfernungsverteilung für $N = 100$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 8,7$, für das mittlere $\epsilon_{max}/2 = 4,35$ und für das untere $\epsilon_{max}/4 = 2,18$.

Die folgenden Schaubilder zeigen die Ergebnisse für $N = 200$ Teilchen:



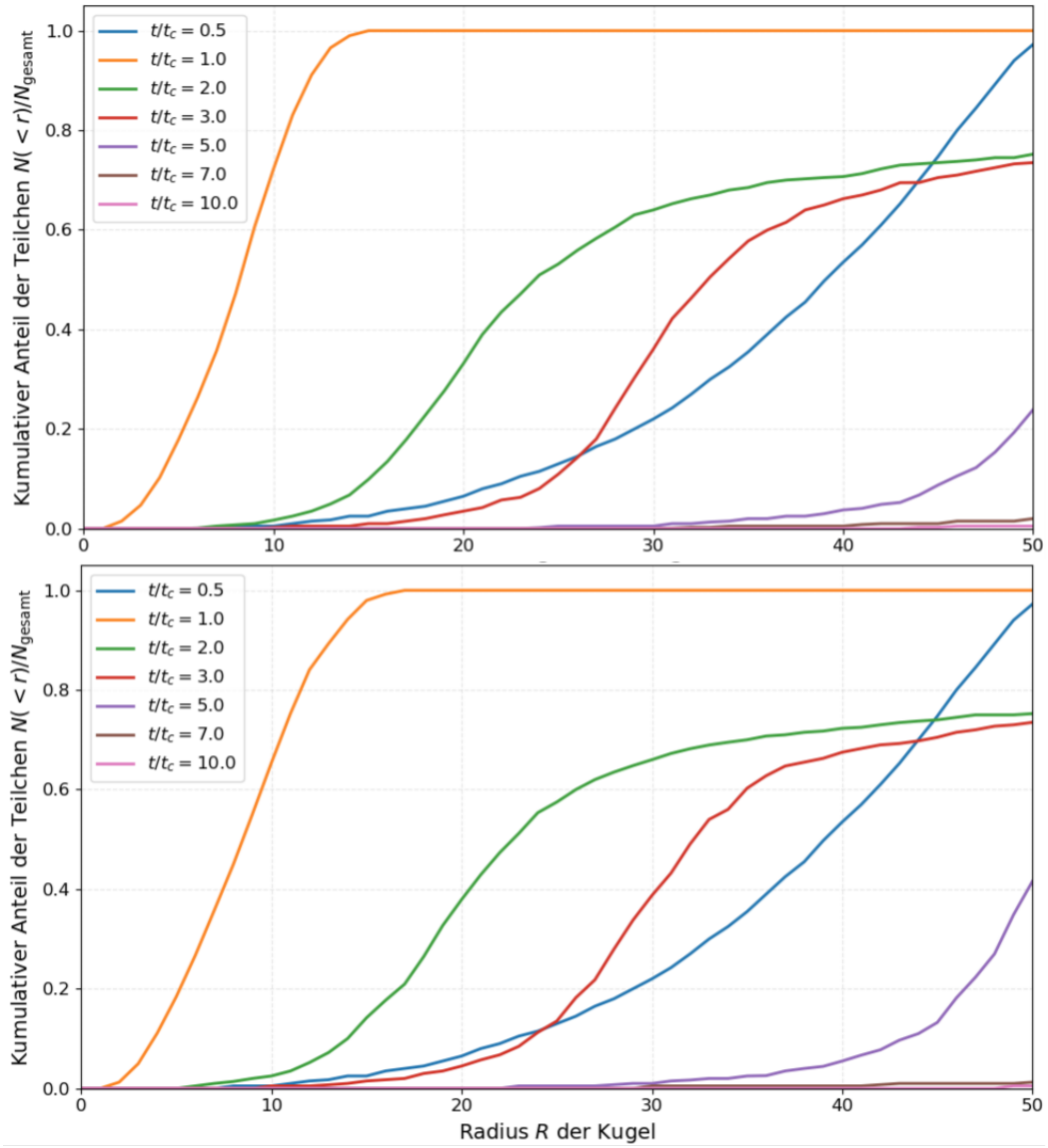



Abbildung 2.18: Darstellung der Entfernungsverteilung für $N = 200$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 7,0$, für das mittlere $\epsilon_{max}/2 = 3,5$ und für das untere $\epsilon_{max}/4 = 1,75$.

Die folgenden Schaubilder zeigen die Ergebnisse für $N = 2000$ Teilchen:

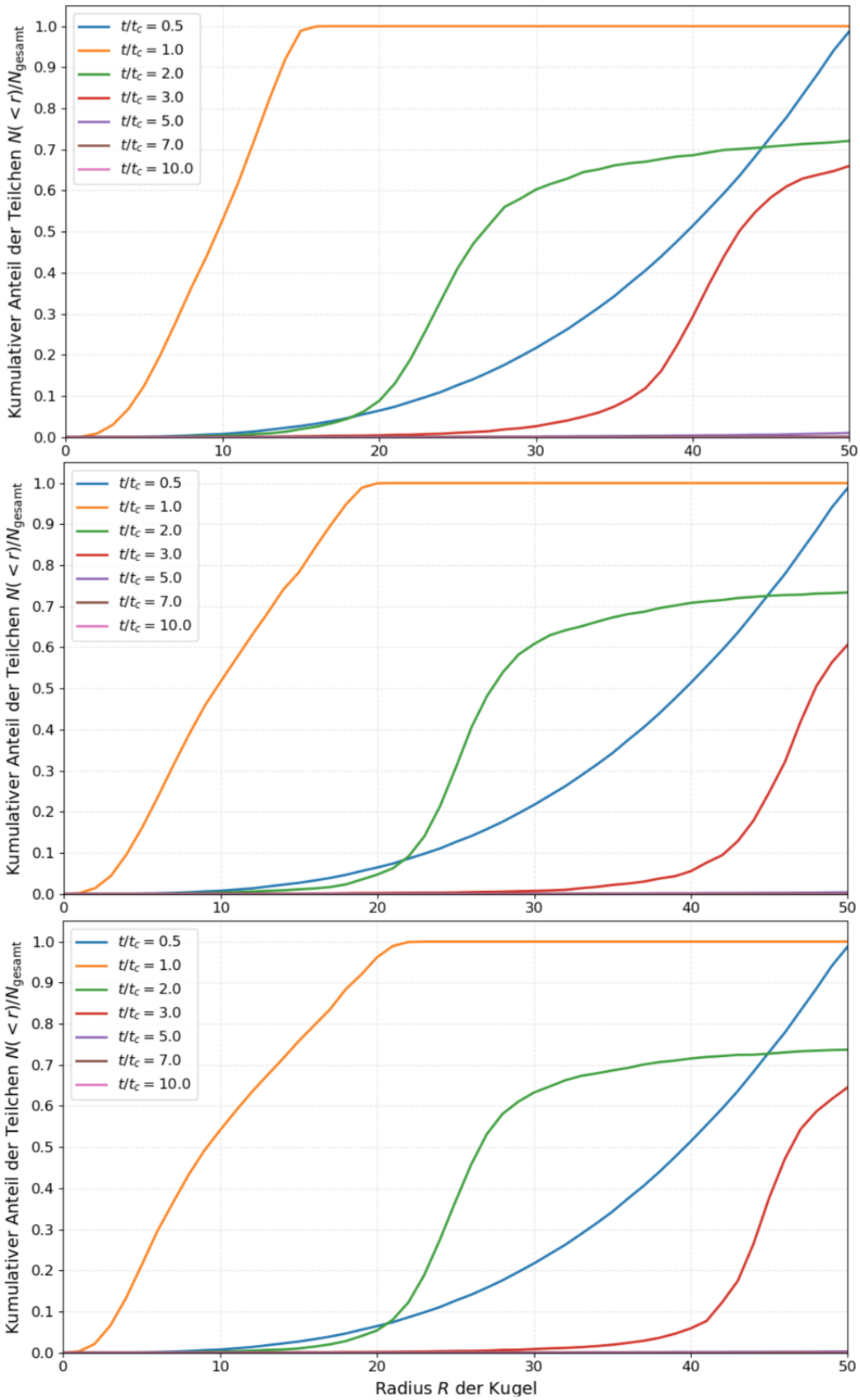



Abbildung 2.19: Darstellung der Entfernungsverteilung für $N = 2000$ Teilchen: Für das obere Bild beträgt das softening $\epsilon_{max} = 3,2$, für das mittlere $\epsilon_{max}/2 = 1,6$ und für das untere $\epsilon_{max}/4 = 0,8$.

Es zeigt sich, dass in allen Ergebnissen unabhängig von der Teilchenzahl als auch vom softening alle Halos nach dem ersten Kollaps ($t/t_c = 1,0$) mindestens 20% bis fast zu 30% der Gesamtteilchen aus dem System entweichen. Das ist natürlich auf den Kollaps zurückzuführen, bei dem ein Teil der Teilchen über die zuvor beschriebenen Prozesse im Zuge der Virialisierung an kinetischer Energie gewinnen, um aus dem System zu entweichen. Bei der Abbildung für $N = 50$ und $\epsilon_{max} = 11$ ist zu erkennen, dass bis zum Zeitpunkt $t/t_c = 10,0$ die Teilchenanzahl sich kaum verändert und bei etwa 75% der Gesamtteilchenzahl verharrt. Da aber ein System mit gerade 50 Teilchen eine sehr geringe Zeitspanne von $1,0\,t_{ff} < t < 1,27\,t_{ff}$ hat, in der es sich im Quasi-Gleichgewichtszustand befindet, und danach eigentlich der Prozess der Evaporation über längere Zeitskala hinweg dominiert, ist der Ausgang des Ergebnisses wohl auf ein zu hohes softening zurückzuführen. Womöglich spielt aber auch die geringe Anzahl an Teilchenbegegnungen mit gerade nur 50 Teilchen eine Rolle, weshalb bis zum Ende der Simulationsdauer weniger Teilchen aus dem System entweichen als bei höherer Teilchenzahl. Die Ergebnisse für $N = 50$ mit $\epsilon_{max}/2$ und $\epsilon_{max}/4$ zeigen, dass mit abnehmenden softening auch mehr Teilchen im Verlauf der Zeit aus dem System entweichen. Für $\epsilon_{max}/4$ liegt der Anteil der übriggebliebenen Teilchen zur Zeit $t/t_c = 10,0$ bei knapp unter 30%. In fast allen Ergebnissen ist zu erkennen, dass sich mit fortschreitender Zeit die Anzahl der übrigen Teilchen verringert. Bis $t/t_c = 10,0$ sind noch nur etwa 30% der Teilchen mit anfänglichen $N = 50$ übrig, während bei größeren Teilchenzahlen der Anteil noch niedriger liegt mit etwa $1 - 2\%$ bei beispielsweise $N = 200$ und $\epsilon_{max}/4$. Genauso wie die Anzahl verändert sich die räumliche Verteilung der übrigen Teilchen vom Zentrum der Kugel weg und verlagert sich hin zu den äußeren Bereichen der Sphäre. Das Zentrum der Kugel enthält gegen Ende der Simulationszeit kaum noch Teilchen. Tabelle 2.5 fasst für alle Teichenzahlen $N$ bei einem softening von $\epsilon_{max}/4$ zusammen, wo sich exemplarisch 40% der Teilchen zu drei verschiedenen Zeitpunkten zusammen.

Tabelle 2.5: Übersicht über die räumliche Verteilung der Teilchen bei $N(< r)/N_{gesamt} = 0.4$ zu drei verschiedenen Zeitpunkten und unterschiedlicher Teilchenanzahl $N$ (bei $\epsilon_{max}/4$).

| $N$ | $t/t_c = 1,0$ | $t/t_c = 2,0$ | $t/t_c = 3,0$ |
|---|---|---|---|
| 50 | $r = 7,0$ | $r = 18,0$ | $r = 21,2$ |
| 100 | $r = 6,5$ | $r = 18,5$ | $r = 24,5$ |
| 200 | $r = 7,0$ | $r = 20,2$ | $r = 30,0$ |
| 2000 | $r = 7,2$ | $r = 26,0$ | $r = 45,0$ |

Auffällig ist auch, dass in fast allen Ergebnissen (außer bei $N = 2000$) bei einer Zeit von $t/t_c = 2,0$ und $t/t_c = 3,0$ die Teilchenanzahl sich kaum oder nur minimal verändert und zwischen etwa $77 - 79\%$ für $N = 50$, $76 - 78\%$ für $N = 100$ und $77 - 78\%$ für $N = 200$ konstant hält. Nur die räumliche Verteilung der Teilchen innerhalb $0 < r < R$ verändert sich zwischen den zwei Zeitpunkten ein wenig. Das lässt sich vermutlich dadurch erklären, dass die simulierten Halos bis zum Zeitpunkt $t_2$, der für die Teilchenzahlen $N = 50, 100$ und 200 mit $t_2 \approx 1,27\, t_{ff} - 3,77\, t_{ff}$ sehr nah beieinander liegt, den kurzen Quasi-Gleichgewichtszustand widerspiegelt.
Als Aussicht für eine weitere, aufbauende Untersuchung wäre es interessant zu wissen, wie sich die Ergebnisse der Simulationen ändern würden, wenn man den Teilchen vor Beginn des Kollaps eine Anfangsgeschwindigkeit zuteilen würde, z.B. in Form einer Tangentialgeschwindigkeit, wenn man rotierende Halos betrachten möchte, oder mit einer Geschwindigkeitsdispersion unter den Teilchen. Wie schon auf S. 38 kurz angemerkt, gehen Labini et al. bei einem Kollaps mit „warmen Anfangsbedinungen“, wie es in so einem Fall genannt wird, davon aus, dass die *violent relaxation* „weniger gewaltsam“ ablaufen würde, wobei es letztendlich auch zur Ausbildung einer Kern-Halo-Struktur kommen würde [41]. Zu beachten sei auch, dass bei der Herleitung des Virialtheorems dann ein weiterer Term berücksichtigt werden müsste, weil die Teilchen nun einen Anfangsimpuls hätten (vgl. Gl. (2.30)).

# Kapitel 3

# Kosmisches Gas in Halos

Während wir uns in Kapitel 2 mit Halos aus reiner dunkler Materie auseinander gesetzt haben, wollen wir im Folgenden die Modelle und Betrachtungen ein wenig realistischer gestalten und nun baryonische Materie, darunter kosmisches Gas, miteinbeziehen, aus der sich Sterne und Galaxien gebildet haben. Baryonisch meint im kosmologischen Kontext, entgegen der eigentlichen Definition in der Teilchenphysik, nicht nur die Betrachtung von schweren Teilchen wie Protonen oder Neutronen, sondern auch die im Vergleich dazu leichten Elektronen (Leptonen) [35]. Der Anteil an (kalter) dunkler Materie beträgt etwa $\Omega_{cdm} = 0,22$ und der von sichtbarer, baryonischer Materie etwa $\Omega_{gas} = 0,05$ an der gesamten Dichte im Universum. Damit beträgt der Anteil der dunklen Materie an der Gesamtmaterie rund 80% und für die sichtbare Materie nur 20% [35]. Obwohl dieser Anteil sehr gering ist, haben sich aus dem kosmischen Gas, das sich innerhalb von Halos befindet, später weitere beobachtbare Strukturen, beispielsweise Sterne oder Galaxien, gebildet. Da die dunkle Materie mit ihrem höheren Materieanteil ein größeres Gravitationspotential erzeugt, findet sich kosmisches Gas auch in dunkle-Materie-Halos wieder.

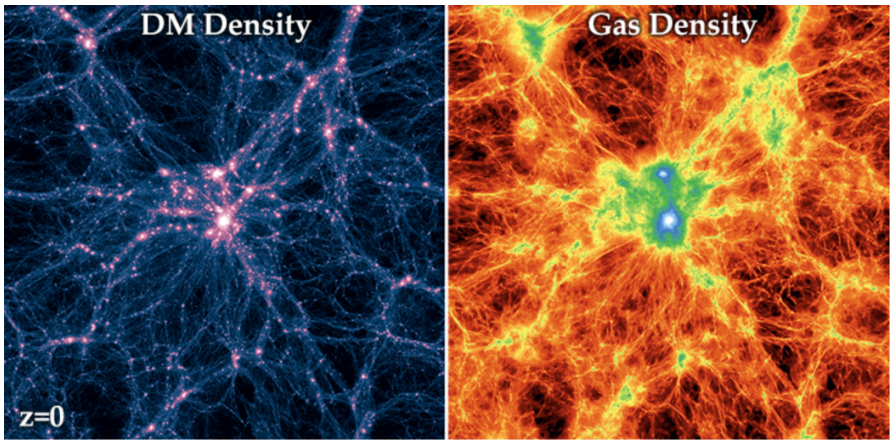



Abbildung 3.1: Vergleich zwischen der Dichteverteilung von dunkler Materie und Gas; entnommen von [20].

Abbildung 3.1 zeigt einen Vergleich zwischen der Dichteverteilung der dunklen Materie als auch des Gases, die farblich unterschiedlich kontrastiert sind. Interessant ist hier zu sehen, dass an den Knotenpunkten der Filamente der dunklen Materie gleichzeitig auch eine hohe Dichte von Gas aufzufinden ist, wenn man beide Bilder miteinander vergleicht.

## 3.1 Kollaps und Stabilität von kosmischen Gas

Während bei Dunkle-Materie-Halos im Modell des sphärischen Kollapses (vgl. Kapitel 2) der Dichtekontrast ausschlaggebend war, ab wann Halos sich von der Hintergrunddichte abkoppeln und kollabieren, müssen wir beim kosmischen Gas den Druck, den das Gas aufbaut, berücksichtigen. Da die permanent wirkende Gravitationskraft seinerseits auf das Gas Druck ausübt, wird dieses komprimiert. Normalerweise würden wir gemäß unserer Alltagserfahrung erwarten, dass komprimiertes Gas heißer wird. Zum Beispiel führt die Kompression von Luft in einem Dieselmotor zur Entzündung des eingespritzten Diesels. Auf kosmologischer Ebene erfolgt genau das Gegenteil. Denn das Gas kühlt aufgrund der hohen Dichte durch die Kompression schneller ab, was sehr kontraintuitiv erst einmal ist [35]. Doch genau dieser Prozess sorgt am Ende dafür, dass eine Gaswolke weiter komprimiert wird, weil sie keinen entsprechenden Gegendruck zur Gravitation der dunklen Materie aufbauen kann, bis irgendwann die Kernfusion zündet und ein Stern entsteht [35]. Damit wir ein besseres Verständnis davon erhalten, ab wann eine Gaswolke unter der Gravitation kollabiert, muss es dafür Bedingungen geben und diese können wir mit dem sogenannten *Jeans-Kriterium* aufstellen.
Zur Herleitung betrachten wir wieder der Einfachheit halber eine ideal symmetrische Sphäre mit einer Masse $M$, einem Radius $R$ und einer konstanten, mittleren Dichte $\bar{\rho}$. Nun befindet sich ein Punktteilen in einem Abstand $r = R$ auf der Oberfläche der Sphäre. Jetzt stellt sich die Frage, die wir uns schon beim Modell des sphärischen Kollaps (vgl. Kapitel 2.1) gestellt haben, welche Zeit das Punktteilchen aus der Ruhe ($v = 0$) heraus braucht, um das Zentrum zu erreichen. Mithilfe der Energieerhaltung ($\rightarrow$ Gl. (2.2)) lässt sich diese Fallzeit ermitteln [35]

$$e = \frac{1}{2}\left(\frac{dr}{dt}\right)^2 - \frac{GM}{r} \tag{3.1}$$

wobei die radiale Geschwindigkeit $v_r$ durch $(dr/dt)^2$ ersetzt wurde. Für das Punktteilchen auf der Oberfläche der Kugel mit $v = 0$ ergibt seine spezifische Gesamtener-

gie:

$$e = -\frac{GM}{R} \tag{3.2}$$

Setzt man dies in Gl. (3.1) ein, erhält man:

$$\frac{1}{2}\left(\frac{dr}{dt}\right)^2 = GM\left(\frac{1}{r} - \frac{1}{R}\right) \tag{3.3}$$

Um den Term auf der linken Seite zu lösen, führen wir eine Variablensubstitution mit $x = r/R$ durch, woraus auch $dx = dr/R$:

$$\frac{1}{2}\left(\frac{R\,dx}{dt}\right)^2 = GM\left(\frac{1}{xR} - \frac{1}{R}\right) = \frac{GM}{R}\left(\frac{1}{x} - 1\right) \tag{3.4}$$

Unter Ausnutzung, dass $M = \frac{4}{3}\pi R^3\bar{\rho}$ ist, folgt:

$$\frac{1}{2}\left(\frac{dx}{dt}\right)^2 = \frac{G}{R^3}\,\bar{\rho}\frac{4}{3}\pi R^3\left(\frac{1}{x} - 1\right) \tag{3.5}$$

Kürzen des Radius' $R$ und Wurzel ziehen vereinfacht den Ausdruck zu:

$$\frac{dx}{dt} = \sqrt{\frac{8}{3}G\pi\bar{\rho}}\sqrt{\left(\frac{1}{x} - 1\right)} \tag{3.6}$$

Trennung der Variablen und Substitution der Integrationsgrenzen von $r = 0 \rightarrow x = 0$ nach $r = R \rightarrow x = 1$ ergibt [35]:

$$\int_0^1 \frac{1}{\sqrt{\left(\frac{1}{x} - 1\right)}}\,dx = \sqrt{\frac{8}{3}G\pi\bar{\rho}}\int_0^{t_{ff}} dt \tag{3.7}$$

Die Zeit, nach der wir integrieren, ist nichts anderes als die Freifallzeit $t_{ff}$, die in Kapitel 2.3.4 als die Zeit vom *turn-around*-Punkt bis zum Kollaps eingeführt wurde. Die Lösung der linken und rechten Seite von Gl. (3.7) führt zu [35]:

$$\frac{\pi}{2} = \sqrt{\frac{8}{3}G\pi\bar{\rho}}\;t_{ff} \tag{3.8}$$

und damit final die Freifallzeit:

$$t_{ff} = \sqrt{\frac{3\pi}{32G\bar{\rho}}} \tag{3.9}$$

Erstaunlicherweise hängt dieses Ergebnis nur von der mittleren Dichte des betrachteten Systems ab, aber nicht allein von seiner Größe oder der Masse.
Gehen wir einen Schritt weiter und betrachten eine Sphäre mit Gas in ihrem Inneren, um einen Gegendruck zur Gravitationskraft aufzubauen und damit dem Kollaps entgegenzuwirken. Die Zeit, die das System braucht, um den Gegendruck aufzubauen, muss also kleiner sein als die nötige Freifallzeit $t_{ff}$. Diese „Gegendruck"-Zeit bezeichnen wir als $t_{Ggd}$. Es gilt daher [35]

$$t_{ff} > t_{Ggd} \tag{3.10}$$

Ist $t_{ff} < t_{Ggd}$ hat das System zu wenig Zeit, um einen Gegendruck aufzubauen und kollabiert infolgedessen. Eine weitere Größe, die mit der Zeit einhergeht, ist die Geschwindigkeit $c_s$ (analog zur Schall- oder Lichtgeschwindigkeit), mit der sich der Druck im Gas ausbreitet, und lautet nach [35]

$$c_s = \sqrt{\frac{\partial p}{\partial \rho}} \tag{3.11}$$

Damit erhält man für $t_{Ggd}$

$$t_{Ggd} = \frac{R}{c_s} \tag{3.12}$$

Es folgen nun zwei Möglichkeiten das Jeans-Kriterium aufzustellen: Einmal in Abhängigkeit der Größe oder der Masse des Systems. Bei einer Sphäre bezeichnet $R$ den Radius und es gilt für die Stabilität [35]:

$$R < R_J \equiv c_s \sqrt{\frac{3\pi}{32 G \bar{\rho}}} \tag{3.13}$$

wobei $R_J$ als die sogenannte *Jeans-Länge* bezeichnet wird und als Schwellenwert gilt, ab welcher Größe das System zu kollabieren beginnt. Alternativ lässt sich auch die *Jeans-Länge* durch die *Jeans-Masse* $M_J$ ersetzen, indem man $M = \frac{4}{3}\pi R^3 \bar{\rho}$ verwendet. Formt man nach dem Radius $R$ um

$$R = \left(\frac{3M}{4\pi\bar{\rho}}\right)^{1/3} \tag{3.14}$$

und setzt es in Gl. (3.13) ein, ergibt sich:

$$R_J = \left(\frac{3M_J}{4\pi\bar{\rho}}\right)^{1/3} = c_s \sqrt{\frac{3\pi}{32 G \bar{\rho}}} \tag{3.15}$$

Nun wird der Ausdruck nach $M_J$ umgeformt und vereinfacht.

$$M < M_J = \underbrace{\frac{4}{3}\left(\frac{3}{32}\right)^{3/2}\pi^{5/2}}_{\approx 0{,}67}\, c_s^3\, \bar{\rho}^{-1/2}\, G^{-3/2} \tag{3.16}$$

An dieser Stelle sei daraufhin gewiesen, dass der berechnete Vorfaktor je nach Art der Herleitung variieren kann [35]. Anhand der Jeans-Masse wird deutlich, ab welchen Schwellenwert für die Masse $M$ das System zu kollabieren beginnt.
Die Dichte $\bar{\rho}$ geht zwar mit dem Jeans-Kriterium einher, aber auch die Ausbreitungsgeschwindigkeit des Drucks $c_s$ muss genauer konkretisiert werden. Wenn man annimmt, dass der Großteil von Gaswolken aus einem Gemisch von Wasserstoff und Helium besteht, dann können wir mithilfe der *idealen Gasgleichung* schreiben, bei der wir der Einfachheit halber annehmen, dass die Teilchen kein Eigenvolumen besitzen und keine Wechselwirkungskräfte untereinander ausüben [35]

$$pV = Nk_BT \longrightarrow p = \underbrace{\frac{N}{V}}_{=n} k_BT \tag{3.17}$$

wobei $n$ die Anzahldichte der Teilchen meint – nicht zu verwechseln mit der Stoffmenge $[n] = \text{mol}$ aus der Chemie. Des Weiteren führen wir die dimensionslose, mittlere Teilchenmasse $\mu$ in Einheiten der Protonenmasse $m_p$ ein [35]. Dadurch ergibt sich, dass $n$ umgeschrieben werden kann:

$$n = \frac{\rho}{\mu m_p} \quad | \quad \text{Einheitenüberprüfung: } \frac{1}{\text{m}^3} = \frac{\frac{\text{kg}}{\text{m}^3}}{\text{kg}} \tag{3.18}$$

Eingesetzt in die ideale Gasgleichung resultiert zu:

$$p = \frac{\rho}{\mu m_p} k_BT \tag{3.19}$$

Wenn man Gl. (3.11) hinzunimmt, werden nun die vorherigen Schritte verständlicher, da zur Berechnung der Ausbreitungsgeschwindigkeit $c_s$ Gl. (3.19) nach der Dichte abgeleitet werden kann:

$$c_s = \sqrt{\frac{\partial}{\partial\rho}\left(\frac{\rho}{\mu m_p}k_BT\right)} = \sqrt{\frac{k_BT}{\mu m_p}} \tag{3.20}$$

Somit ist $c_s$, wie man erkennen kann, proportional zu $T^{1/2}$. Bei steigender Temperatur einer Gaswolke nimmt auch entsprechend die Jeans-Länge bzw. die Jeans-Masse ($M_J \propto T^{3/2}$) zu.

## 3.2 Dichteprofile

Nachdem wir uns in Kapitel 3.1 mit der Stabilität von kosmischem Gas mit dem einhergehenden Jeans-Kriterium auseinandergesetzt haben, wollen wir uns im Folgenden nun mit den Dichteprofilen der dunklen Materie aber auch des Gases in Halos beschäftigen. Laut der Formel für die Jeans-Länge oder -Masse hängt die Stabilität eines Objekts von seiner mittleren Dichte $\bar{\rho}$ ab. Aber das wäre nur eine Vereinfachung, da die Dichten sowohl des Gases als auch der dunklen Materie über den Raum hinweg eben nicht konstant sind. Infolgedessen können wir auf Grundlage der Fachliteratur zur Halo-Forschung die lokale Dichteverteilung der dunklen Materie und des Gases genauer beschreiben.

### 3.2.1 Dunkle Materie

Im Jahr 1995 haben die Kosmologen Julio F. Navarro, Carlos S. Frenk und Simon D.M. White N-Körper-Simulationen zu Halos im Modell der kalten dunklen Materie mit verschiedenen Größenordnungen von Halomassen (von Zwerghalos mit $M \approx 10^{10}\,M_\odot$ bis zu galaxienreichen Halos $M \approx 10^{15}\,M_\odot$) durchgeführt und eine radiale Dichteverteilung erstellt. Interessanterweise haben sie festgestellt, dass die Form der Dichteverteilung für verschiedene Halomassen ähnlich ist und gemäß der Formel

$$\frac{\rho}{\rho_c} = \frac{\delta_c}{\left(\frac{r}{r_s}\right)\left(1+\frac{r}{r_s}\right)^2} \tag{3.21}$$

dargestellt werden kann [28, 29]. Diese Formel wird auch als NFW-Dichteprofil bezeichnet. $\rho_c$ bezeichnet wieder die kritische Dichte ($\rightarrow$ Gl. (2.14)), $r_s$ ist der Skalenradius und $\delta_c$ ist die charakteristische, dimensionslose Dichte, die mit der Masse des Halos zusammenhängt und beschrieben werden kann durch [28]:

$$\delta_c = \frac{200}{3}\frac{c^3}{\ln(1+c) - \frac{c}{1+c}} \tag{3.22}$$

$c$ ist der sogenannte Konzentrationsparameter und wird definiert durch:

$$c = \frac{r_{vir}}{r_s} \tag{3.23}$$

$r_{vir}$ ist der bereits bekannte Virialradius eines Halos, der auch als $r_{200}$ angegeben werden kann [35]. Im Allgemeinen ist der Konzentrationsparameter von der Halomasse, der Rotverschiebung sowie von kosmologischen Parametern abhängig [5].
Der Einfachheit halber lässt sich Gl. (3.21) laut [5] etwas umschreiben, wobei die Substitution $x_s := r/r_s$ eingeführt wird:

$$\frac{\rho}{\rho_s} = \frac{1}{x_s \left(1 + x_s\right)^2} \tag{3.24}$$

$\rho_s$ bezeichnet hier die Skalendichte analog zum Skalenradius $r_s$ und ergibt sich aus dem Produkt $\rho_c \delta_c$. Den Skalenradius $r_s$ kann man sich wie einen „Knickpunkt“ vorstellen, an dem die Steigung nahe dem Zentrum bis zum Skalenradius $r_s$ und vom Skalenradius bis in den äußeren Bereich des Halos wechselt. Abbildung 3.2 zeigt den Verlauf des NFW-Dichteprofils in doppelt-logarithmischer Darstellung.

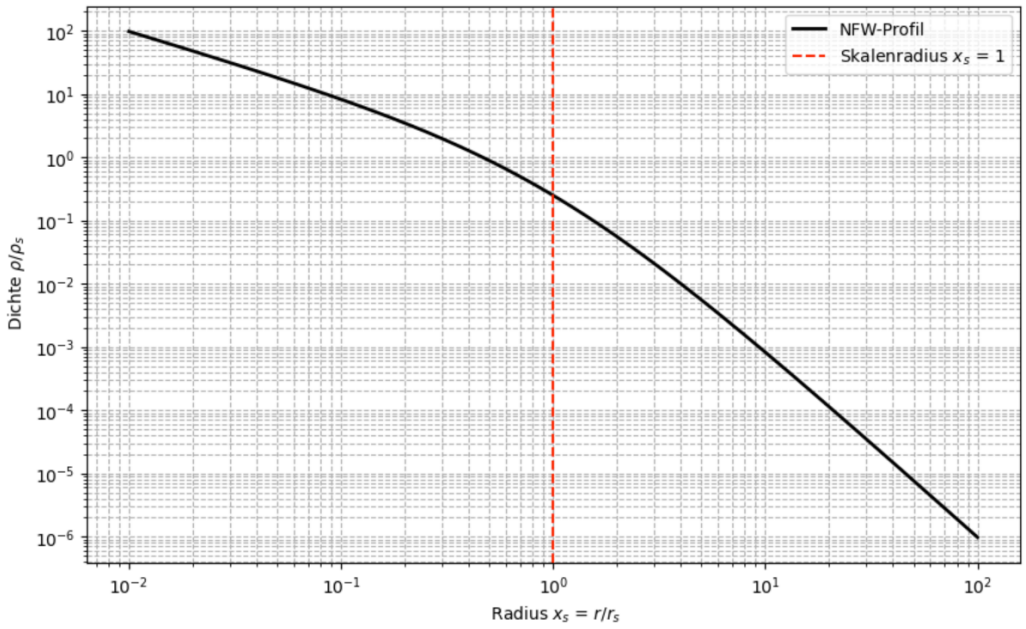



Abbildung 3.2: Doppelt-logarithmische Darstellung des NFW-Profils: Auf der y-Achse ist die Dichte $\rho$ aufgetragen und auf die Skalendichte $\rho_s$ normiert. Auf der x-Achse ist der Radius $r$ aufgetragen und auf den Skalenradius $r_s$ normiert. Zur Orientierung ist der Wert $x_s = r_s/r_s = 1$ mit einer rot gestrichelten Linie markiert.

Die y-Achse ist auf die Skalendichte sowie die x-Achse auf den Skalenradius normiert.

Zur Orientierung, ab welchem Punkt es zum Wechsel der Steigung im Dichteverlauf kommt, ist eine rot gestrichelte Linie bei $r = r_s \rightarrow x_s = 1$ eingezeichnet. Für $x_s \ll 1$ bzw. $r \ll r_s$ ergibt die Dichte:

$$\rho(x_s \ll 1)/\rho_s \approx x_s \longrightarrow \rho \propto r^{-1} \tag{3.25}$$

und für $x_s \gg 1$ bzw. $r \gg r_s$ ergibt die Dichte:

$$\rho(x_s \ll 1)/\rho_s \approx x_s^3 \longrightarrow \rho \propto r^{-3} \tag{3.26}$$

Anlass zur Diskussion bringt die Steigung nahe dem Zentrum, bei der die Dichte theoretisch $\rho \rightarrow \infty$ gehen würde. Vielmehr deuten Beobachtungen in inneren Regionen von Galaxien auf eine konstante Dichteverteilung der dunklen Materie hin. Dieses bisher ungelöste Problem in der Kosmologie wird als *core-cusp*-Problem bezeichnet [16].
Mithilfe des NFW-Profils lässt sich auch die eingeschlossene Masse $M(r)$ unter der Annahme eines sphärischen Halos über das Kugelvolumen berechnen. Allgemein gilt [5]:

$$M(r) = \int_0^r 4\pi r'^2 \rho(r')\, dr' \tag{3.27}$$

Für $\rho(r')$ setzen wir das NFW-Dichteprofil $\rho_{NFW}$ ein:

$$M(r) = 4\pi \int_0^r r'^2 \frac{\rho_s}{\left(\frac{r'}{r_s}\right)\left(1+\frac{r'}{r_s}\right)^2}\, dr' \tag{3.28}$$

Wie schon zuvor, führen wir die Substitution $x_s := r/r_s$ durch, aus dem auch $dx_s = dr/r_s$ folgt [5]:

$$M(r) = 4\pi \int_0^{x_s} x_s'^2 r_s^2 \frac{\rho_s}{x_s'(1+x_s')^2}\, r_s\, dx_s' = 4\pi\rho_s r_s^3 \int_0^x \frac{x_s'}{(1+x_s')^2}\, dx' \tag{3.29}$$

Unter Ausnutzung der Identität [5]

$$\frac{x}{(1+x)^2} = \frac{1}{1+x} - \frac{1}{(1+x)^2} \tag{3.30}$$

lässt sich Gl. (3.29) elementar integrieren. Für unsere Zwecke setzen wir die Integrationsgrenzen auf $r = 0 \rightarrow x = 0$ und den Virialradius $r = r_{vir} \rightarrow x = r_{vir}/r_s \stackrel{!}{=} c$,

wodurch die obere Integrationsgrenze dem vorhin eingeführten Konzentrationsparameter entspricht, um letztendlich die sogenannte *Virialmasse* $M(r_{vir}) \equiv M_{vir}$ – in der Literatur auch manchmal mit $M_{200}$ gekennzeichnet [35]– herzuleiten.

$$M_{vir} = 4\pi\rho_s r_s^3 \int_0^{x_s=c} \left( \frac{1}{1+x'_s} - \frac{1}{(1+x'_s)^2} \right) dx' \tag{3.31}$$

$$= 4\pi\rho_s r_s^3 \left[ \ln(1+x'_s) + \frac{1}{1+x_s} \right]_0^c \tag{3.32}$$

Auswertung des Integrals an den eingesetzten Grenzen führt zu:

$$M_{vir} = 4\pi\rho_s r_s^3 \left( \ln(1+c) + \frac{1}{1+c} \right) - \left( \underbrace{\ln(1+0)}_{=0} + \frac{1}{1+0} \right) \tag{3.33}$$

$$= 4\pi\rho_s r_s^3 \left( \ln(1+c) + \frac{1}{1+c} \right) - 1 \tag{3.34}$$

Den Term $-1$ erweitern wir mit $1+c/1+c$ und erhalten final:

$$M_{vir} = 4\pi\rho_s r_s^3 \left( \ln(1+c) - \frac{c}{1+c} \right) \tag{3.35}$$

Mithilfe der Gl. (3.35) lässt sich nun in unserem Fall die Skalendichte $\rho_s$ bestimmen:

$$\rho_s = \frac{M_{vir}}{4\pi r_s^3 \left( \ln(1+c) - \frac{c}{1+c} \right)} \tag{3.36}$$

Um nun diese Formel für unsere Zwecke nutzen zu können, fehlen uns aber noch die Werte für den Konzentrationsparameter $c$ und den Skalenradius $r_s$. Da wir aber bereits wissen, dass die Beziehung $c = r_{vir}/r_s$ gilt, können wir $r_s$ bestimmen, indem wir für eine gegebene Halomasse $M$, die gleichzeitig hier auch die Virialmasse $M_{vir}$ ist, den entsprechenden Virialradius $r_{vir}$ bestimmen. Dafür nutzen wir aus, dass

$$M_{vir} = \frac{4}{3}\pi r_{vir}^3 \cdot 200\rho_c \tag{3.37}$$

gilt. An der Stelle sei noch einmal erinnert, dass der Virialradius $r_{vir}$ den Radius $r$ eines virialisierten Halos bezeichnet, dessen Dichte 200-mal (*eigentlich*: 178-mal vgl. Kap. 2.2.2) größer ist als die der Hintergrunddichte $\rho_c$ bei einer gegebenen Rot-

verschiebung $z$ (in unserem Fall: $z = 0$). Damit folgt aus Gl. (3.37):

$$r_{vir} = \left( \frac{3}{4\pi} \cdot \frac{M_{vir}}{200\rho_c} \right)^{1/3} \tag{3.38}$$

Für verschiedene Halomassen unterschiedlicher Größenordnungen lässt sich nun jeweils der Virialradius berechnen, deren Werte in Tabelle 3.1 zusammengefasst sind.

Tabelle 3.1: Übersicht über die berechneten Virialradien $r_{vir}$ [in kpc] in Abhängigkeit der Halomasse [in $M_\odot$] verschiedener Größenordnungen.

| $M_{Halo}$ | $10^{10}\,M_\odot$ | $10^{11}\,M_\odot$ | $10^{12}\,M_\odot$ | $10^{13}\,M_\odot$ | $10^{14}\,M_\odot$ | $10^{15}\,M_\odot$ |
|---|---|---|---|---|---|---|
| $r_{vir}$ [in kpc] | 44,45 | 95,76 | 206,31 | 444,47 | 957,59 | 2063,07 |

Da der Konzentrationsparameter in der Regel von der Halomasse abhängt, können wir mithilfe der Abbildung 3.3 von [14] entnehmen, bei welcher Halomasse (für $z = 0$) sich welcher Wert für $c$ ergibt:

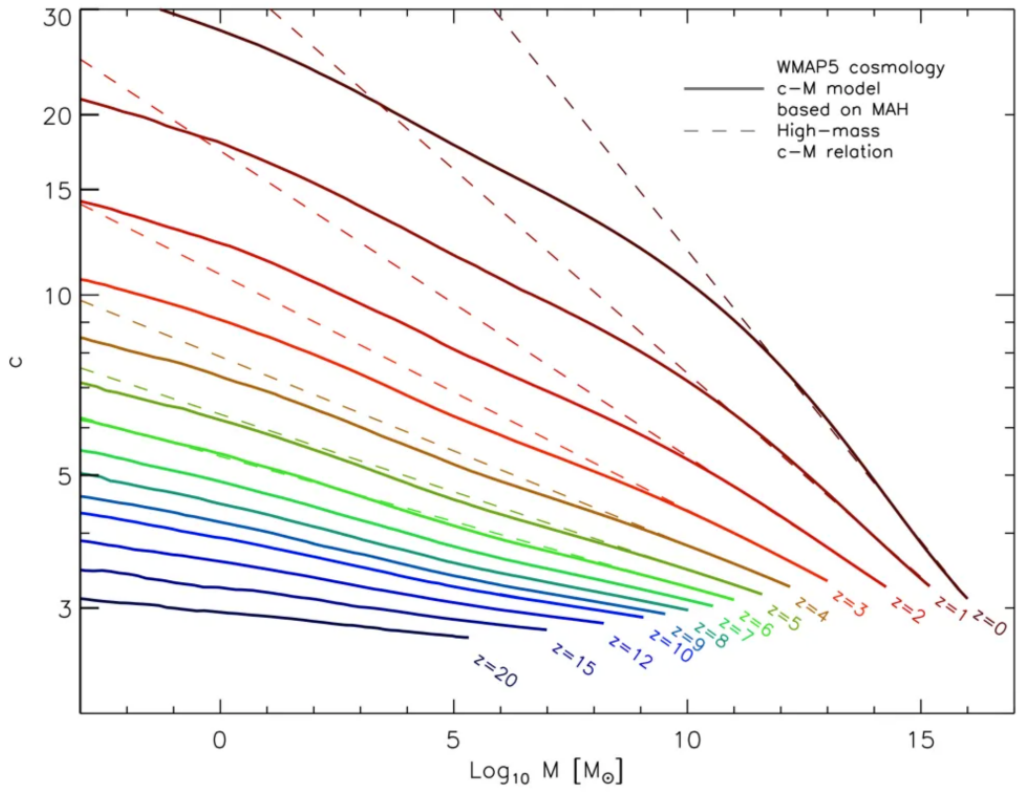



Abbildung 3.3: Konzentrationsparamater $c$ gegen die Halomasse [in Einheiten von $M_\odot$] in logarithmischer Darstellung für unterschiedliche Rotverschiebungen $z$ aufgetragen; entnommen von [14].

Aus der Abb. 3.3 werden die Werte (für $z = 0$) des Konzentrationsparameters aus dem Modell *High-Mass c-M relation*, das von Correa et al. genutzt wurde, entnommen. Die Werte des Modells *WMAP5-Cosmology* werden deshalb nicht gewählt, weil das Modell die dunkle Energie mit $\Omega_\Lambda = 0,74$ im Universum berücksichtigt [14]. Auch im zweiten Teil dieser Arbeit berufen wir uns weiterhin auf dem Modell des Einstein-de-Sitter-Universums ohne dunkle Energie ($\Omega_\Lambda = 0$). Anhand der Abbildung ist zu erkennen, dass der Konzentrationsparameter für kleinere Halomassen größer ist als für größere Halomassen. Allgemein ist der Wert für $c$ umso größer, je früher sich Halos bilden. Dadurch zeigt sich die hierarchische Entstehung von Halos, und impliziert, dass kleine Halos sich früher gebildet haben, während die Hintergrunddichte des Universums noch größer war als im Vergleich zu heute [5]. Mit den Virialradien lassen sich nun auch die Skalenradien berechnen. Und darauffolgend können mit den berechneten Skalienradien dann auch die entsprechenden Skalendichten $\rho_s$ gemäß Gl. (3.36) berechnet werden, wobei für die Dichte die Radien von kpc in m und die Massen in kg umgerechnet worden sind. Tabelle 3.2 fasst die Ergebnisse zusammen:

Tabelle 3.2: Übersicht über die berechneten Skalenradien $r_s$ über die Beziehung $c = r_{vir}/r_s$ in Abhängigkeit des für jede Halomasse $M$ bestimmten Konzentrationsparameters $c$ sowie die entsprechenden Skalendichten $\rho_s$ nach Gl. (3.36).

| $M_{Halo}$ | $10^{10}\,M_\odot$ | $10^{11}\,M_\odot$ | $10^{12}\,M_\odot$ | $10^{13}\,M_\odot$ | $10^{14}\,M_\odot$ | $10^{15}\,M_\odot$ |
|---|---|---|---|---|---|---|
| $c$ | 12,5 | 9,5 | 7,8 | 6,2 | 4,9 | 3,9 |
| $r_s$ [kpc] | 3,56 | 10,08 | 26,45 | 71,69 | 193,45 | 528,99 |
| $\rho_s$ [$10^{-22}\,\mathrm{m}^3/\mathrm{kg}$] | 7,14 | 3,64 | 2,26 | 1,31 | 0,78 | 0,46 |

### 3.2.2 Gas

Auch Gase können anhand einer Dichteverteilung beschrieben: Im Jahr 1976 haben A. Cavaliere und R. Fusco-Femiano auf Grundlage der Daten zur Röntgenleuchtkraftverteilung von heißem Plasma in Galaxienhaufen eine gute Übereinstimmung mit dem *King*-Modell gefunden [11]. Die Dichteverteilung des Gases in Galaxien kann gemäß den Beobachtungsdaten daher an das sogenannte *β-Modell*

$$\rho_{Gas} = \rho_0 \left(1 + \left(\frac{r}{r_c}\right)^2\right)^{-\frac{3}{2}\beta} \tag{3.39}$$

angepasst werden. $\rho_0$ bezeichnet hier die Zentral- oder Kerndichte und stellt die maximale Dichte bei $r = 0$ dar. $r_c$ bezeichnet den Kernradius. Das $\beta$-Modell kann als eine Art modifiziertes King-Modell angesehen werden, der um den Faktor $\beta$ erweitert wurde. Die Konstante $\beta$ kann über das hydrostatische Gleichgewicht unter der Annahme eines isothermen Gasverteilung hergeleitet werden und ist laut [36] definiert als

$$\beta := \frac{\mu m_p \sigma_v^2}{k_B T_{Gas}} \tag{3.40}$$

$\sigma_v^2$ beschreibt die Geschwindigkeitsdispersion des Gases. Im Folgenden verwenden wir für unsere Zwecke einen Wert von etwa $\beta \approx 2/3$, den man als ungefähren Wert aus den Ergebnissen der Publikation von Chen et al [13] entnehmen kann. Wie schon beim NFW-Profil bietet sich wiederum die Substitution mit dem normierten Kernradius $x_c := r/r_c$ an. Abbildung 3.4 zeigt den Verlauf des Dichteprofils.

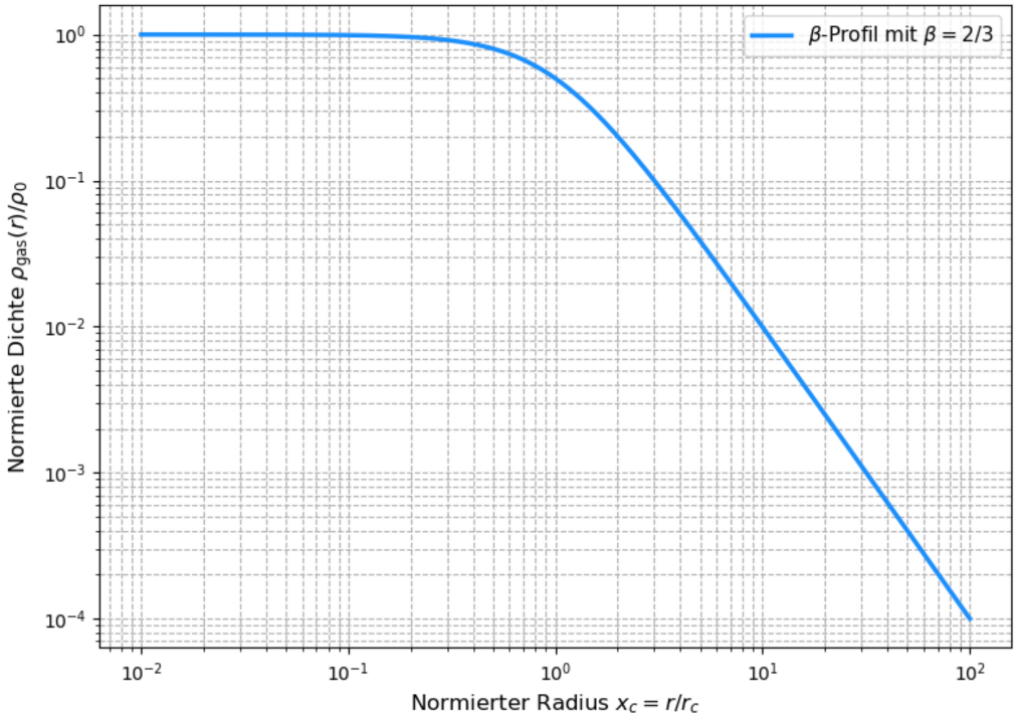



Abbildung 3.4: Doppelt-logarithmische Darstellung des $\beta$-Profil mit $\beta = 2/3$. Auf der y-Achse ist die Dichte auf $\rho_0$ normiert sowie auf der x-Achse der Radius auf den Kernradius $r_c$ mit $x_c := r/r_c$.

Für $x \ll x_c$ erkennt man die Abflachung des Profils, das auf eine konstante Dichte des Gases nahe dem Zentrum hindeutet. Für $x \gg x_c$ fällt die Dichte mit $\rho \propto x^{-2}$ ab. Wir möchten nun analog zum Skalenradius beim NFW-Profil mithilfe der bereits berechneten Werte auch Werte für den Kernradius $r_c$ ermittlen. Aus den Ergebnissen der Publikation von Chan besteht zwischen dem Skalen- und dem Kernradius die

hergeleitete Beziehung [12]

$$r_s \approx \sqrt{3}\, r_c \tag{3.41}$$

Dabei wurden über das hydrostatische Gleichgewicht eine Beziehung zwischen dem Modell der kalten dunklen Materie mittels des NFW-Profils und dem Modell des heißen Gases mittels des $\beta$-Profils hergeleitet. Für eine detaillierte Betrachtung der Herleitung sei auf [12] verwiesen. Mithilfe der Gl. (3.41) können wir mit den Skalenradien die Kernradien ermitteln. Tabelle 3.3 fasst die Ergebnisse zusammen.

Tabelle 3.3: Übersicht über die berechneten Kernradien $r_c$ [in kpc].

| $M_{Halo}$ | $10^{10}\,M_\odot$ | $10^{11}\,M_\odot$ | $10^{12}\,M_\odot$ | $10^{13}\,M_\odot$ | $10^{14}\,M_\odot$ | $10^{15}\,M_\odot$ |
|---|---|---|---|---|---|---|
| $r_c$ [in kpc] | 2,05 | 5,82 | 15,27 | 41,39 | 111,96 | 305,41 |

## 3.3 Hydrostatisches Gleichgewicht

Auf den vorherigen beiden Unterkapiteln aufbauend sollen die beiden Aspekte Stabilität des Gases ($\rightarrow$ Jeans-Kriterium) gegenüber der Gravitation sowie die unterschiedlichen Dichteprofile der dunklen Materie ($\rightarrow$ NFW-Profil) und des Gases ($\rightarrow$ $\beta$-Profil) zusammengeführt werden. Im Folgenden betrachten wir ein Gas innerhalb eines von dunkler Materie dominierten Halos und stellen die Frage, wie groß der Druck des Gases auf die Gravitation der dunklen Materie sein muss, um ein hydrostatisches Gleichgewicht zu erzielen. Allgemein ist aus den Vorlesung zur klassischen Mechanik bekannt, dass ein Gleichgewicht im Gesamtsystem nur dann bestehen kann, wenn die auftretenden Kräfte gleich groß sind und entgegengerichtet zueinander wirken. In unserem Fall muss für ein Gleichgewicht gelten, dass die Gravitationskraft $F_{Grav}$ gleich groß wie die Druckkraft $F_{Druck}$ sein muss:

$$-F_{Grav} = F_{Druck} \tag{3.42}$$

Oder allgemein muss $\sum_i F_i = 0$ für ein Kräftegleichgewicht gelten. Die Druckkraft, die auf eine bestimmte Flächeneinheit $A$ wirkt, ist nichts anderes als das Produkt des Drucks und der Fläche, also $F = pA$. Zur Beschreibung des Gleichgewichts betrachten wir ein würfelförmiges Gaspäckchen der Kantenlänge $z$ (s. Abb. 3.5).

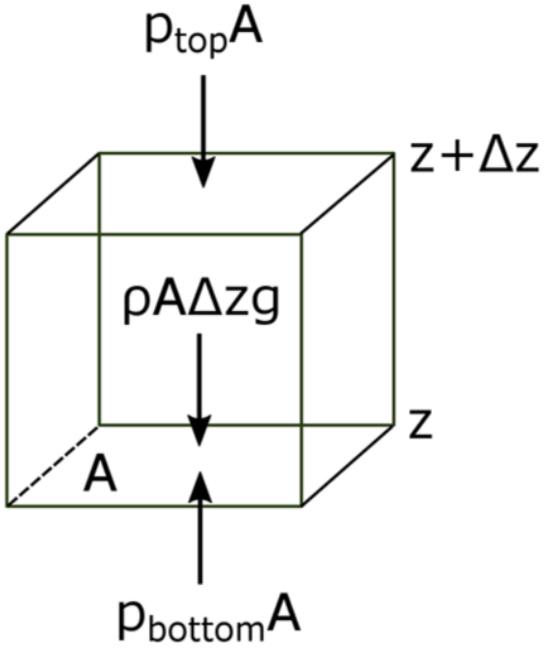



Abbildung 3.5: Schematische Skizze eines würfelförmigen Gaspäckchens mit Kantenlänge $z$ mit den von oben und unten einwirkenden Druckkräften; entnommen von [31].

Von oben wirkt auf das Gaspäckchen die Kraft $F_{top} = -p_{top}A$. Das Vorzeichen der Kraft ist negativ, weil die Kraft „nach unten“ drückt. Von unten ergibt die Kraft $F_{bottom} = +p_{bottom}A$. Als dritte Kraft muss noch die Gewichtskraft $F_G = -mg = -\rho g V$ des Gaspäckchens selbst berücksichtigt werden, wobei $m$ die Masse und $\rho$ die Dichte des Päckchens bezeichnet [31]. Das Minuszeichen resultiert wieder aus dem Grund, da die Gewichtskraft „nach unten“ wirkt. Das Volumen lässt sich über die Seitenfläche $A$ und der Höhe $\Delta z$ des Würfels zu $F = -\rho g \cdot A\Delta z$ umschreiben. Die Summe aller drei Kräfte $F_{tot}$ ergibt demnach [31]:

$$F_{tot} \stackrel{!}{=} 0 = -\rho g \cdot A\Delta z + p_{bottom}A - p_{top}A \tag{3.43}$$

Die Fläche $A$ kürzt sich heraus und vereinfachen des Ausdrucks liefert [31].

$$\underbrace{p_{top} - p_{bottom}}_{=\Delta p} = -\rho g \Delta z \tag{3.44}$$

$\Delta p$ ist hier die Differenz der Drücke von oben und unten. Umformen ergibt:

$$\frac{\Delta p}{\Delta z} = -\rho g \tag{3.45}$$

Den Ausdruck $\Delta p/\Delta z$ kann man auch mit dem Differentialoperator zu $dp/dz$ umschreiben oder allgemein durch Anwendung des Nabla-Operators zu $\vec{\nabla}p$.
$g = GM/r^2$ ist die aus der Mechanik bekannte Erdbeschleunigung (Ortsfaktor). Für ein allgemeines sphärisch-symmetrisches System mit einer $M$ und einem Radius $r$ verwendet man das Gravitationspotential $\Phi(r)$ und wendet den Nabla-Operator an. Allgemein lautet die Gleichung für das hydrostatisches Gleichgewicht [36]:

$$\vec{\nabla}p = -\rho\vec{\nabla}\Phi \tag{3.46}$$

Da sowohl das Gravitationspotential als auch der Druck für ein sphärisches System radial-symmetrisch sind, lässt sich Gl. (3.46) in Abhängigkeit vom Radius $r$ schreiben als [36]:

$$\frac{dp}{dr} = -\rho\frac{d\Phi}{dr} \tag{3.47}$$

Mit $\Phi(r) = GM/r$ ($\rightarrow$ Gl. (2.43)) ergibt sich daher $d\Phi/dr = GM/r^2$ [36]:

$$\frac{dp}{dr} = -\rho\frac{GM(r)}{r^2} \tag{3.48}$$

An dieser Stelle sei angemerkt, dass die eingeschlossene Masse $M(r)$ hier diejenige Masse darstellt, die sich über die Integration der NFW-Profildichte ergibt ($\rightarrow$ Gl. (3.27)) und berücksichtigt werden muss, während $\rho$ hier die Dichte des Gases meint und durch das $\beta$-Profil beschrieben wird. Abbildung 3.6 zeigt den qualitativen Verlauf der Druckkraft (blau gestrichelte Linie), wenn es sich um ein isothermes Gas handelt. Die Gravitationskraft überwiegt die Druckkraft nahe des Zentrums. Bei etwa $x \approx 12$ kommt es zum Schnittpunkt mit der Kurve für die Gravitationskraft und die Druckkraft überwiegt die Gravitation in den äußeren Bereichen des Gases.

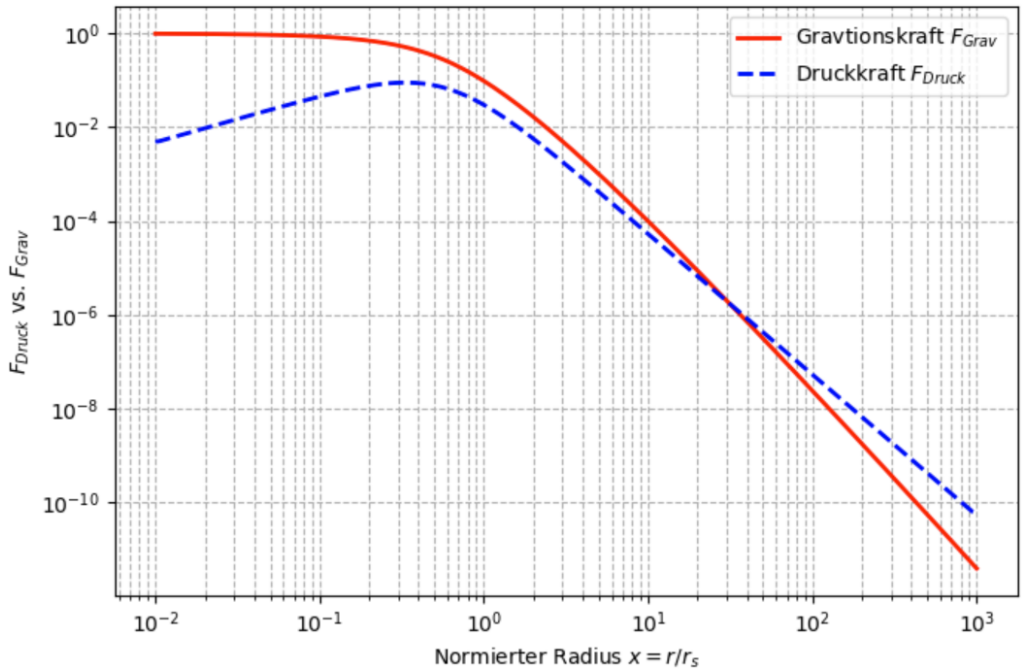



Abbildung 3.6: Veranschaulichung der Gravitationskraft gegen die Druckkraft unter Annahme eines isothermen Gases in doppelt-logarithmischer Darstellung.

Abbildung 3.6 zeigt also eindeutig, dass das Gas eben nicht isotherm sein darf, um ein hydrostatisches Gleichgewicht zu erreichen. Daher muss die Temperatur des Gases radial variieren.

## 3.4 Temperaturprofil

Nach Herleitung des hydrostatischen Gleichgewichts und konkreter Anwendung auf unseren Fall können wir mithilfe von Gl. (3.48) das notwendige Temperaturprofil $T(r)$ herleiten, um das geforderte Gleichgewicht zwischen dem Gas und der Gravitation zu erzielen. Den Ausdruck $dp/dr$ können wir über das ideale Gasgesetz umschreiben ($\rightarrow$ Gl. (3.19)) und erhalten:

$$\frac{d}{dr}\left(\frac{\rho(r')}{\mu m_p} k_B T(r')\right) = -\rho(r')\frac{GM(r')}{r'^2} \tag{3.49}$$

Integration der beiden Seiten von einem Radius $r$ bis ins Unendliche liefert:

$$\frac{k_B}{\mu m_p}\int_r^\infty \frac{d}{dr'}\left[\rho(r')T(r')\right]dr' = -\int_r^\infty \rho(r')\frac{GM(r')}{r'^2}dr' \tag{3.50}$$

Der Differentialoperator $d/dr$ und das Integral heben sich gemeinsam auf und ergeben letztendlich die Funktion selbst an den Integrationsgrenzen ausgewertet.
Unter Anwendung der Randbedingung und Annahme, dass die Gasdichte im Unendlichen verschwindet ($\lim_{r\to\infty} \rho(r) \approx 0$), vereinfacht sich die linke Seite zu:

$$-\frac{k_B}{\mu m_p}\rho(r)T(r) = -\int_r^\infty \rho(r')\frac{GM(r')}{r'^2}dr' \tag{3.51}$$

Natürlich wird die Gasdichte in Wirklichkeit für $r \to \infty$ nicht null werden und stellt hier eine Vereinfachung dar, weil die Gasdichte nicht die kritische Dichte des Universums unterschreiten wird, die ja selbst nicht bei null liegt, aber mit einem Wert von $\rho_c = 1,9 \cdot 10^{-26}\ \mathrm{kg\,m^{-3}\,h^2}$ äußerst klein ist.
Nach Umstellen von $k_B \cdot \rho(r)$ erhalten wir das Temperaturprofil $T(r)$:

$$T(r) = \frac{\mu m_p}{k_B\,\rho(r)}\int_r^\infty \rho(r')\frac{GM(r')}{r'^2}dr' \tag{3.52}$$

Wir überführen nun die Funktion für die Temperatur $T(r)$ in eine dimensionslose Form um, wie es schon in Kap. 2.3.5 mit den Bewegungsgleichungen gemacht wurde. Dafür schreiben wir die Masse zu $\tilde{M}(x) = M(r)/4\pi\rho_s r_s^3$ sowie $\tilde{\rho}(x) = \rho(r)/\rho_0$ in eine dimensionslose Gasdichte um. Außerdem substituieren wir wieder mit dem normierten Skalenradius $x_s = r/r_s$ und damit $dr = r_s dx_s$. Einsetzen in Gl. (3.52) resultiert zu:

$$T(x) = \frac{\mu m_p}{k_B}\frac{1}{\rho_0\tilde{\rho}(x)}\int_x^\infty \rho_0\tilde{\rho}(x')\frac{4\pi G\rho_s r_s^3\cdot\tilde{M}(x')}{(x'_s r_s)^2}r_s dx'_s \tag{3.53}$$

Der Integralausdruck lässt sich vereinfachen, indem die Konstanten vor das Integral gezogen werden.

$$T(x) = \left(\frac{\mu m_p G\rho_s r_s^2}{k_B}\right)\underbrace{\frac{4\pi}{\widetilde{\rho}(x)}\int_x^\infty \tilde{\rho}(x')\frac{\tilde{M}(x')}{x_s'^2}\,dx'_s}_{=\widetilde{T}(x)} \tag{3.54}$$

Für die verschiedenen Halomassen können wir dann die Temperatur im eingeklammerten Ausdruck jeweils berechnen, womit wir eine Art Referenztemperatur je nach Halomasse erhalten – oder anders ausgedrückt: wir berechnen eine Virialtemperatur, da wir die Werte $r_s$ und $\rho_s$ ($\rightarrow$ Tab. 3.2) aus dem Virialradius ermittelt haben.

Für ein vollständig ionisiertes Gas beträgt die mittlere Teilchenmasse $\mu \approx 0,6$, da im ionisierten Zustand nicht nur die Protonen der Wasserstoff- und Heliumatome, sondern auch ihre Elektronen berücksichtigt werden müssen [35]. Mit den Werten aus Tabelle 3.2 ergeben sich folgende Werte in Tabelle 3.4 zusammengefasst:

Tabelle 3.4: Übersicht die berechneten Temperaturwerte in Abhängigkeit der Halomasse.

| $M_{Halo}$ | $10^{10}\,M_\odot$ | $10^{11}\,M_\odot$ | $10^{12}\,M_\odot$ | $10^{13}\,M_\odot$ | $10^{14}\,M_\odot$ | $10^{15}\,M_\odot$ |
|---|---|---|---|---|---|---|
| $T$ [K] | $4,17 \cdot 10^4$ | $1,71 \cdot 10^5$ | $7,3 \cdot 10^5$ | $3,12 \cdot 10^6$ | $1,35 \cdot 10^7$ | $5,93 \cdot 10^7$ |

Abbildung 3.7 zeigt das erforderliche Temperaturprofil $\widetilde{T}(x)$, damit es zum hydrostatischen Gleichgewicht zwischen des Gam und der Gravitation kommt.

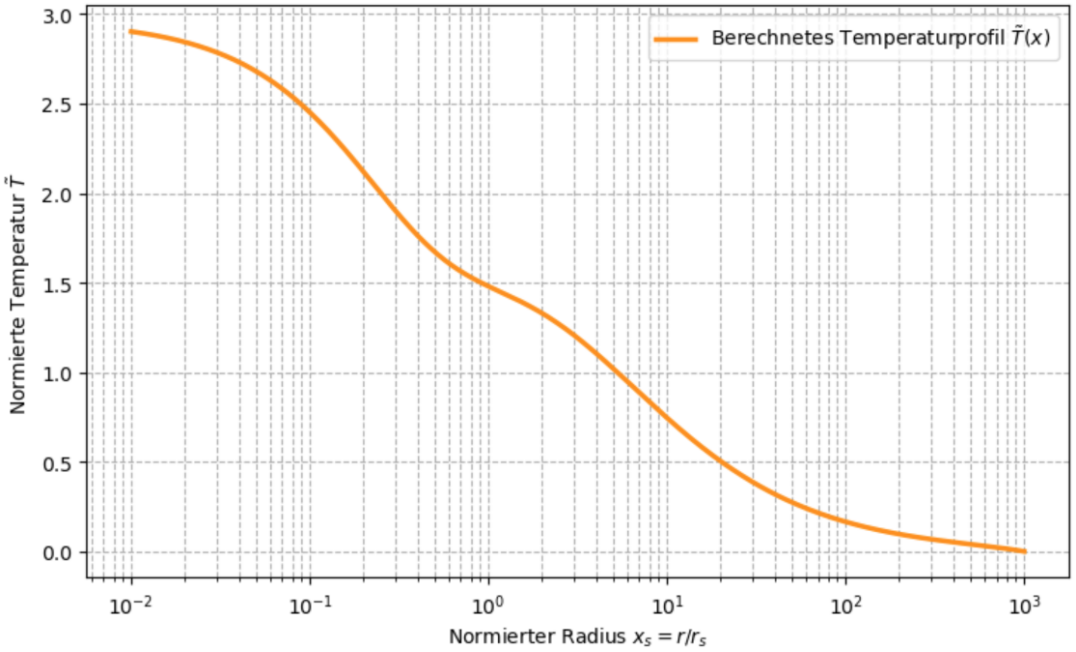



Abbildung 3.7: Dimensionsloses Temperaturprofil $\widetilde{T}(x)$ des Gases in doppeltlogarithmischer Darstellung, um hydrostatisches Gleichgewicht zu erreichen.

Abbildung 3.8 zeigt, dass die beiden Kurven $F_{Grav}$ und $F_{Druck}$ perfekt übereinander liegen.

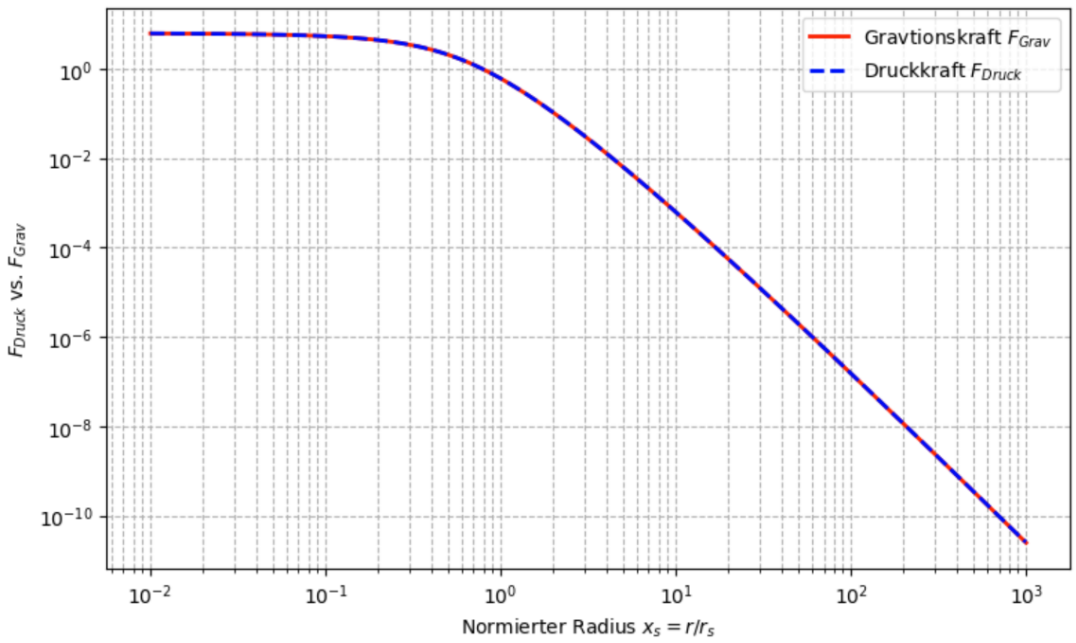



Abbildung 3.8: Veranschaulichung der Gravitationskraft gegen die Druckkraft mit dem in Abb. 3.7 ermittelten Temperaturprofil in doppelt-logarithmischer Darstellung.

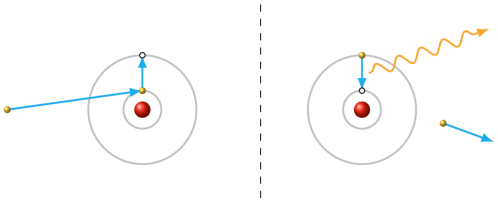
## 3.5 Kühlung von kosmischen Gas

Das Abkühlen von kosmischen Gas in Halos ist überhaupt der Grund, warum sich im Verlaufe der Zeit Sterne gebildet haben. Wenn eine Gaswolke abkühlt, heißt das, dass sie Energie in Form von Wärme abstrahlt. Infolgedessen kann sie keinen entsprechenden Druck gegenüber der Gravitation aufbauen, bis dann irgendwann die Kernfusion zündet. Anhand der Gesetze der Thermodynamik ist bekannt, dass die Erhöhung des Drucks auf ein Gas zur Erhöhung der Temperatur führt. Betrachtet man einen adiabatischen Prozess, bei dem das System keine Wärmeenergie mit der Umgebung austauscht, ausgedrückt durch $TV^{\gamma-1} = \text{konstant}$ (mit $\gamma$ =Adiabatenexponent), führt eine Verringerung des Volumens zur Erhöhung der Temperatur [35]. In der Astrophysik führt der Kollaps eines Gases aber zur Abstrahlung von Energie und dieser Energieverlust fördert die Abkühlung [35]. Die genauen atomar ablaufenden Mechanismen, die zur Abkühlung führen, sollen kurz thematisiert werden.

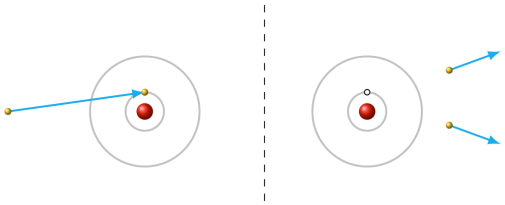
### 3.5.1 Kühlungsmechanismen

Es gibt vier verschiedene Möglichkeiten, wie es zur Abkühlung kommen kann, und alle Prozesse haben gemeinsam, dass es mindestens zwei Teilchen braucht, die mit-

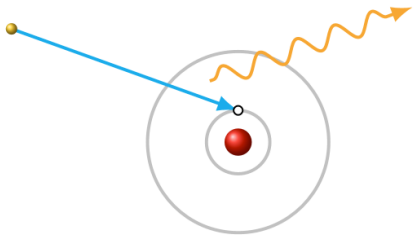
einander wechselwirken müssen [35]. Wie schon zuvor betrachten wir ein Gas in der Frühphase der Universums, das hauptsächlich nur aus Wasserstoff und Helium im 3:1-Verhältnis besteht. Abb. 3.9 fasst die Mechanismen schematisch zusammen.



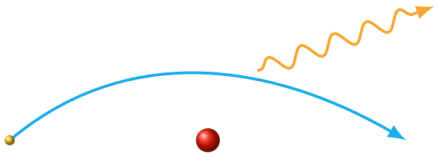
(a) Elektronische Anregung



(b) Ionisation



(c) Rekombination



(d) Bremsstrahlung

Abbildung 3.9: Schematische Darstellung aller vier Kühlungsmechanismen, die sich auf atomarer Ebene abspielen; entnommen und leicht adaptiert von [35].

Der erste Vorgang (vlg. Abb. 3.9a), der zur Abstrahlung führt, ist die elektronische Anregung eines Atoms. Ein freies Elektron kann durch Kollision seine kinetische Energie auf ein im Atom gebundenes Elektron übertragen, um dieses dann von einer energetisch tiefer liegenden Schale in eine energetisch höhere Schale anzuregen. Die für die Anregung notwendig Energie ergibt sich aus der Energiedifferenz $\Delta E$ der beiden Schalen. Weil das angeregte Elektron sich aber in einem instabilen Zustand befindet, fällt es unter Abstrahlung eines Photons der Wellenlänge $\lambda = hc/\Delta E$ in die energetisch tiefer liegende Schale zurück. Mit genügend hoher Energie, wie in Abb. 3.9b zu sehen ist, kann ein freies Elektron das gebundene Elektron aus dem Atom herausschlagen, sodass es nicht mehr gebunden ist und ein elektrisch geladenes Atom übrig bleibt. Diesen Prozess nennt man Ionisation. Beide Elektronen haben nach der Stoßionisation eine geringere Energie. Bei diesem Prozess wird zwar keine Strahlung emittiert, aber dadurch sinkt die mittlere kinetische Energie [35]. Des Weiteren wird auch der dritte Prozess gemäß Abb. 3.9c begünstigt, nämlich dass es zur Rekombination eines freien Elektrons mit einer Fehlstelle im Atom („Loch“) kommt.

Infolgedessen wird hier Strahlung abgegeben. Der letzte Prozess ist die Bremsstrahlung (vgl. Abb. 3.9d), die auch in Röntgenröhren auftritt und Bestandteil von Röntgenstrahlung ist. Dabei wird ein Elektron in der Nähe einer positiven Ladung durch sein elektrisches Feld so stark „abgebremst“, dass es Strahlung emittiert. Aus der Elektrodynamik ist bekannt, dass eine Änderung des Geschwindigkeitsbetrag eines Elektrons durch das elektrische Feld des Atoms zu einer Beschleunigung führt. Und beschleunigte Ladungen emittieren elektromagnetische Strahlung [32].
Wenn also alle vorher beschriebenen Prozesse auf Kollisionen zwischen den Teilchen beruhen, lässt sich auch eine Kollisionsrate herleiten. Die Rate ist nach [35] proportional zur Anzahldichte $n_A$ eines Teilchens der Spezies $A$ und $n_B$ eines Teilchens der Spezies $B$. Da das Interesse auf primordialem Gas im vollständig ionisierten Zustand liegt, kann man nun die Anzahldichten der Heliumkerne $n_{He}$ sowie der Elektronen $n_{e^-}$ in Einheiten von der Anzahldichte des Wasserstoffkerns $n_H$ ausdrücken. Weil das Massenverhältnis von Wasserstoff- zu Heliumkernen bei 3 : 1 liegt, wobei für $m_H = 1m_p$ und $m_{He} = 4m_p$ in Einheiten der Protonenmasse gilt, kommen auf ein Wasserstoff- zwölf Heliumkerne. Die Verhältnisse der Anzahldichten ergeben dann [35]

$$\frac{n_{e^-}}{n_H} = \frac{14}{12} \quad \text{und} \quad \frac{n_{He}}{n_H} = \frac{1}{12} \tag{3.55}$$

Für ein ionisiertes Gas ergeben sich 14 Elektronen, zwölf vom Wasserstoff und zwei vom Helium. Die gesamte Anzahldichte $n$ ergibt aus der Gesamtzahl der Teilchen, nämlich

$$n = \frac{13 + 14}{12} = \frac{27}{12}\, n_H \tag{3.56}$$

Bis auf den konstanten Faktor ist die Kollisionsrate proportional zu $n_A \cdot n_B \approx n_H^2$ ist. Anschließend lässt sich damit eine sogenannte *Kühlrate* $\mathcal{C}$ definieren [35]:

$$\mathcal{C} = \Lambda(T) \cdot n_H^2 \tag{3.57}$$

Die Kühlrate gibt an, wie viel Energie pro Zeit und pro Volumen durch verloren geht. $\Lambda(T)$ beschreibt die Kühlfunktion in Abhängigkeit von der Temperatur, wobei $\Lambda(T)$ eine Vereinfachung ist, da die Funktion auch von der *Metallizität* $Z$ abhängt, also $\Lambda(T, Z)$. Die Metallizität gibt die Häufigkeit an, mit der Metalle bei Fusionsprozessen entstehen. Als Metalle werden die Elemente gesehen, deren Ordnungszahl höher als Helium ist. Abb. 3.10 zeigt den Kurvenverlauf von Kühlfunktionen qualitativ verschiedener Metallizitäten in Einheiten von $Z_\odot$.

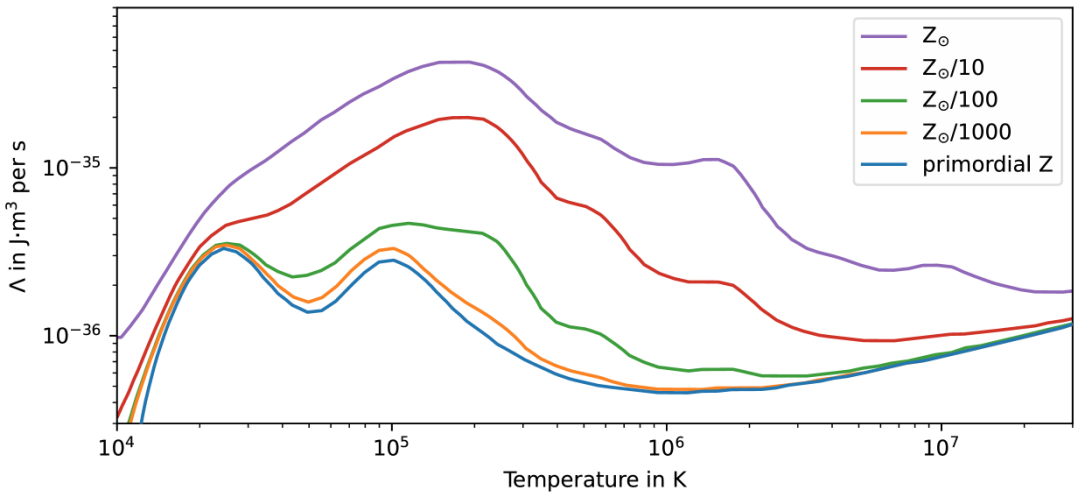



Abbildung 3.10: Kühlfunktionen im Temperaturbereich von $T = 10^4$ bis $10^8$ für verschiedene Metallizitäten $Z$; entnommen von [35]. Die Daten selbst zur Erstellung dieser Kühlfunktionen sind bei Sutherland et al. zu finden [39].

Auffällig ist, dass die Kühlkurven für zunehmende Metallizitäten höher liegen als für niedrig liegende. Das bedeutet, dass die Anwesenheit von schwereren Elementen die Kühlung eines Gases begünstigt. Das primordiale Gas kühlt damit am schwächsten. In der Kurve für das primordiale Gas (primordial $Z_\odot$) erkennt man zwischen $T = 10^4\,\mathrm{K}$ und $T = 10^5\,\mathrm{K}$ zwei Peaks. Der linke Peak ist auf den Prozess der elektronischen Anregung und Ionisation für Wasserstoff und der rechte Peak für Helium aufgrund derselben Prozesse zurückzuführen. Ab $T = 10^6\,\mathrm{K}$ ist der Anstieg der Kurve auf die Bremsstrahlung zurückzuführen [35]. Für unsere Zwecke betrachten wir die Kühlfunktion (blaue Kurve) des primordialen Gases genauer und wollen nun wissen, wie lange das Gas braucht, um abzukühlen. Denn die Gase dürfen ja nur eine bestimmte Kühlzeit brauchen, wenn man heute Sterne beobachten kann.

### 3.5.2 Kühlzeit

Wir betrachten das Gas während der Kühlzeit weiterhin im hydrostatischen Gleichgewicht. Außerdem vernachlässigen wir sogenannte *Rückkopplungen* (engl. *feedback*) im Galaxienkernen, wenn das Gas aufgrund von Super-Novae-Explosionen bei der Sternentwicklung von innen wieder aufgeheizt wird [35]. Für die Kühlzeit $t_{Khl}$ wird laut [35] die grobe Annahme gemacht, dass für das Verhältnis von der Energie $\mathcal{E}$ eines Teilchens zur Kühlrate $\mathcal{C}$ gilt:

$$t_{Khl} = \frac{\mathcal{E}}{\mathcal{C}} \tag{3.58}$$

Die Energie eines einzelnen Teilchen ist über den Gleichverteilungssatz definiert als $E = f/2 \cdot k_B T$, wobei $f$ die Anzahl der Freiheitsgrade beschreibt. Da (atomarer) Wasserstoff und Helium monoatomare Teilchen im Gasgemisch sind und damit keine Freiheitsgrade der Rotation und der Vibration aufweisen, bleiben nur noch drei translatorische Freiheitsgrade ($x$-, $y$- und $z$-Richtung) übrig, also $f = 3$ und damit folgt $E = 3/2 \cdot k_B T$ [42]. Multipliziert man die Energie mit der gesamten Anzahldichte $n$ ($\rightarrow$ Gl. (3.56)) ergibt sich für $t_{Khl}$ [35]

$$t_{Khl} = \frac{\frac{3}{2} n k_B T}{\Lambda(T) \cdot n_H^2} = \frac{3 n k_B T}{2 \Lambda(T) n_H^2} \tag{3.59}$$

Und mit $n/n_H = 27/12$ ($\rightarrow$ Gl. (3.56)) folgt:

$$t_{Khl} = \frac{6,75 k_B T}{2 \Lambda(T) n_H} = 3,375 \frac{k_B T}{\Lambda(T) n_H} \tag{3.60}$$

Die Temperatur $T$ ergibt sich aus den Referenzwerten, die in Tabelle 3.4 dargestellt wurden, multipliziert mit dem in Gl. (3.54) hergeleiteten Temperaturprofil. Damit erhält die Kühlzeit auch ein radiales Profil. Die Kühlfunktion $\Lambda(T)$ für ein primordiales Gas können wir anhand der Daten von [39] erstellen. Was noch fehlt, ist die Anzahldichte $n$, die wir über das $\beta$-Profil berechnen können. Denn gemäß Gl. (3.18) können wir Dichte $\rho$ in die Anzahldichte $n$ folgendermaßen umrechnen:

$$\frac{n}{n_0} = \frac{\frac{\rho}{\mu m_p}}{\frac{\rho_0}{\mu m_p}} = \frac{\rho}{\rho_0} = \left(1 + \left(\frac{r}{r_c}\right)^2\right)^{-\frac{3}{2}\beta} \tag{3.61}$$

Die Frage ist nun, wie wir auf die Zentraldichte $\rho_0$ kommen. Daher liegt es nahe, mithilfe von Gl. (3.27) die Masse $M_{Gas}$ zu berechnen, die das Gas einschließt, so wie auch die Masse durch das NFW-Profil hergeleitet wurde. Es gilt daher:

$$M_{Gas} = 4\pi \int_0^r r'^2 \rho_0 \left(1 + \left(\frac{r'}{r_c}\right)^2\right)^{-\frac{3}{2}\cdot\beta} dr' \tag{3.62}$$

Mit der Substitution $x_c := r/r_c$, woraus auch $dr = r_c\, dx$ folgt, und mit $\beta = 2/3$ ergibt sich:

$$M_{Gas} = 4\pi \int_0^x (x' r_c)^2 \rho_0 \left(1 + x'^2\right)^{-1} (r_c\, dx') \tag{3.63}$$

$$= 4\pi\rho_0 r_c^3 \int_0^x \frac{x'^2}{1+x'^2}\,dx' \tag{3.64}$$

Auch dieses Integral lässt sich elementar lösen, indem man den Zähler mit $+1-1$ erweitert:

$$\frac{x^2}{1+x^2} = \frac{x^2+1-1}{1+x^2} = 1 - \frac{1}{1+x^2} \tag{3.65}$$

Bildung der Stammfunktion und Integration von $r = 0 \to x = 0$ nach $r = r_{vir} \to x = r_{vir}/r_c$ ergibt:

$$M_{Gas} = 4\pi\rho_0 r_c^3 \left[x - \arctan(x)\right]_0^{\frac{r_{vir}}{r_c}} = 4\pi\rho_0 r_c^3 \left(\frac{r_{vir}}{r_c} - \arctan\left(\frac{r_{vir}}{r_c}\right)\right) \tag{3.66}$$

Kürzen liefert:

$$M_{Gas} = 4\pi\rho_0 r_c^2 \left(r_{vir} - r_c \arctan\left(\frac{r_{vir}}{r_c}\right)\right) \tag{3.67}$$

Mit der Beziehung $r_s = \sqrt{3}\,r_c$ ($\to$ Gl.(3.41)) und $c = r_{vir}/r_s$ lässt sich der Ausdruck elegant umschreiben zu:

$$M_{Gas} = 4\pi\rho_0 \left(\frac{r_s}{\sqrt{3}}\right)^2 \left(r_s c - \frac{r_s}{\sqrt{3}} \arctan\left(\sqrt{3}c\right)\right) \tag{3.68}$$

$$= \frac{4}{3}\pi\rho_0 r_s^3 \left(c - \frac{1}{\sqrt{3}}\arctan\left(\sqrt{3}c\right)\right) \tag{3.69}$$

Die Masse des Gases können wir als $M_{Gas} = f_{Gas} M_{Halo}$ schreiben, wobei $f_{Gas} \approx 0,16$ laut [4] ist und den Anteil der Masse des Gases an der Gesamtmasse meint. Damit lässt sich nun final die Dichte $\rho_0$ bestimmen:

$$\rho_0 = \frac{3 f_{Gas} M_{Halo}}{4\pi r_s^3 \left(c - \frac{1}{\sqrt{3}} \arctan\left(\sqrt{3}\,c\right)\right)} \tag{3.70}$$

Mit der Dichte $\rho_0$ lässt sich auch dann die zentrale Anzahldichte $n_0 = \rho_0/\mu m_p$ ermitteln. Die Tabelle 3.5 fasst die berechneten Werte zusammen:

Tabelle 3.5: Übersicht über die berechneten Werte der Dichte $\rho_0$ sowie der Anzahldichte $n_0$.

| $M_{Halo}$ | $10^{10}\,M_\odot$ | $10^{11}\,M_\odot$ | $10^{12}\,M_\odot$ | $10^{13}\,M_\odot$ | $10^{14}\,M_\odot$ | $10^{15}\,M_\odot$ |
|---|---|---|---|---|---|---|
| $\rho_0\ [10^{-23}\,\mathrm{m^3/kg}]$ | 4,95 | 2,93 | 2,01 | 1,31 | 0,87 | 0,57 |
| $n_0\,[10^3\,1/\mathrm{m^3}]$ | 0,49 | 0,29 | 0,20 | 0,13 | 8,67 | 5,66 |

Jetzt lässt sich die Kühlzeit, wie in Abb. 3.11 zu sehen ist, berechnen, für deren Darstellung die Kühlzeit farblich kodiert ist und $10^4$ bis $10^{14}$ Jahre als Zeitspanne genommen wurde. Als zweite Farbkodierung wurde außerdem das radiale Profil eines jeden Halos, das sich aus dem Temperaturprofil ergibt, abgebildet.

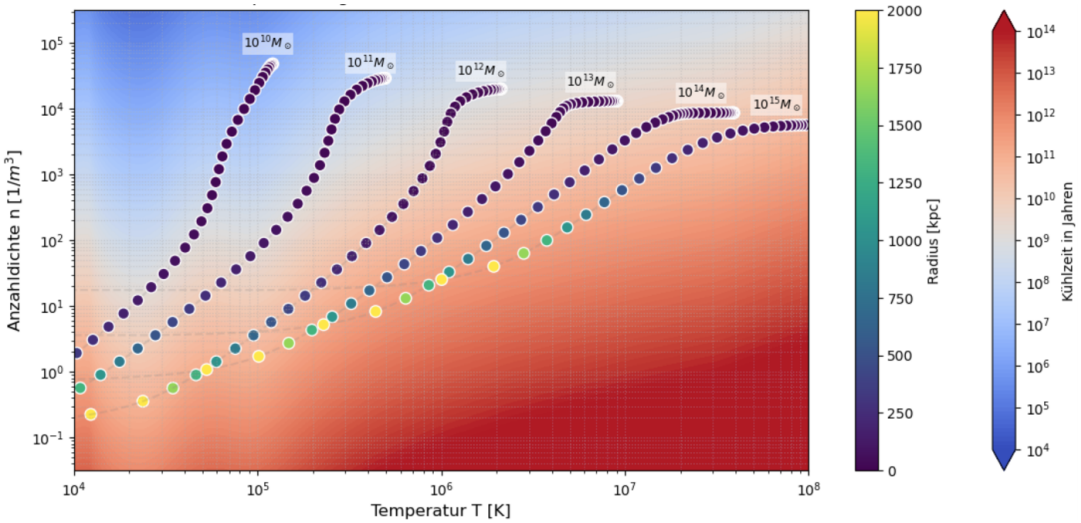



Abbildung 3.11: Kühlzeiten für Halos verschiedener Massen mit Farbkodierung sowohl für die Kühlzeit von $10^4$ bis $10^{14}$ Jahren als auch für das radiale Profil der Halos.

Interessant ist in der Abbildung zu sehen, dass die Kühlzeit der Halos nahe des Zentrums deutlich geringer ist als in den weiter äußeren Bereichen, was auch zu erwarten war. Das erklärt sich dadurch, dass die Dichte im Zentrum des Gases größer ist als weiter außen. Dadurch kühlt das Gas im Inneren schneller ab als in den äußeren Bereichen.

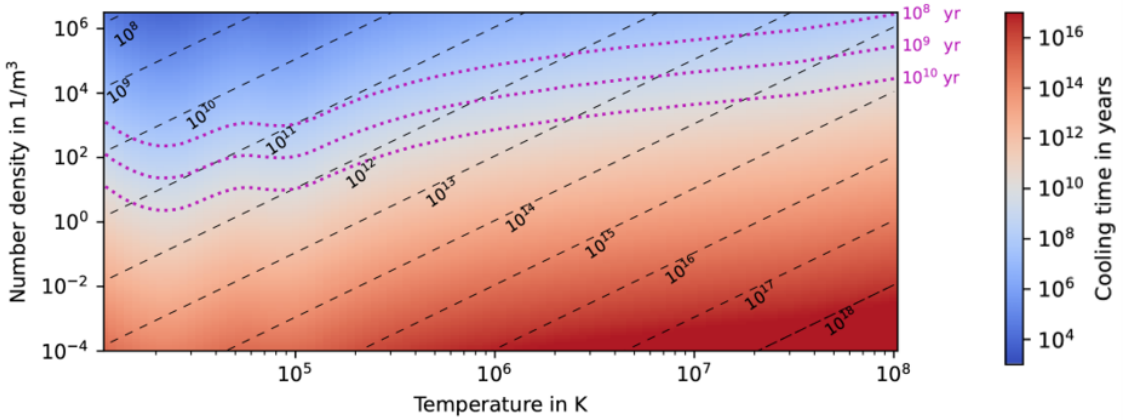



Abbildung 3.12: Kühlzeiten für Halos verschiedener Massen mit Farbkodierung für die Kühlzeit von $T = 10^4$ bis $10^{16}$ Jahre und unter Annahme einer konstanten, mittleren Dichte des Gases; entnommen und leicht adaptiert von [35].

Vergleicht man Abb. 3.11 mit der Abb. 3.12, ist zu erkennen, dass die grau gestrichelten Linien für die Halomassen alle diagonal durch das Diagramm verlaufen. Der Grund hierfür ist, dass bei der Herleitung der Virialtemperatur eine mittlere, konstante Dichte für das Gas angenommen wurde (vgl. Herleitung wieder über das Virialtheorem von [35]), wohingegen in unserem Fall die Dichte des Gases, durch das $\beta$-Profil beschrieben, eben nicht konstant ist und durch ein radiales Profil beschrieben wird. Auch die Temperatur ist nicht konstant, sondern weist ebenso auch radiales Profil auf, um das hydrostatische Gleichgewicht mit der Gravitation der Dunklen Materie zu wahren.

# Kapitel 4

# Zusammenfassung und Fazit

Die vorliegende Masterarbeit hat sich mit dem astronomischen Thema rund um „Dunkle-Materie-Halos“ mit vereinfachten Modellen, z.B. dem Modell des sphärischen Kollapses, und Simulationen vertieft auseinandergesetzt. Dafür wurde im ersten Teil der Arbeit untersucht, wie sich ein Halo, der mit einer unterschiedlichen Anzahl von $N$ Teilchen und unterschiedlichen softening-Werten in mehreren Durchläufen simuliert wurde, während der Zeit entwickelt hat. Damit die durchgeführten N-Körper-Simulationen zuverlässige Ergebnisse liefern konnten, wurde im Unterkapitel davor (s. Kap. 2.3) ausführlich erklärt, wie numerische Stabilität des verwendeten Algorithmus bewerkstelligt werden kann. Durch die Verwendung des Velocity-Verlet-Verfahrens gegenüber dem Euler-Verfahren, oder durch das Force-Softening, um unendliche Kräfte bei den Berechnungen zu vermeiden, sowie die Nutzung von adaptiven Zeitschritten, damit die Beschleunigung unter den Teilchen genauer berechnet werden, können die numerischen Fehler minimiert werden, sodass die wahren, physikalischen Vorgänge nicht verfälscht werden.
Die Ergebnisse zur zeitlichen Entwicklung des Halokollapses haben gezeigt, dass nach dem ersten Kollaps der Halo die sogenannte Virialisierung durchläuft, die durch das Virialtheorem beschrieben werden kann, und anschließend in einen quasistationären Gleichgewichtszustand übergeht. Laut dem Modell des sphärischen Kollapses erreicht ein Halo nach der durchlaufenen Virialisierung den Virialradius, der der Hälfte des ursprünglichen Radius vor dem Kollaps ($R_{vir} = R/2$) entspricht. Die Ergebnisse haben jedoch gezeigt, dass die Mittelwertskurve aller Simulationsdurchläufe sich fast wieder auf den ursprünglichen Ausgangsradius einpendelt, während der Teil der ursprünglich vorhandenen Teilchen aufgrund der *violent relaxation* aus gebundenen System entwichen ist. Um zu zeigen, dass das klassische Virialtheorem erfüllt ist, wurde außerdem die Virialsumme gegen die Simulationszeit aufgetragen. Des Weiteren wurde das Virialtheorem aufgrund des verwendeten softenings modifiziert und ebenfalls dargestellt. Es hat sich gezeigt, dass die simulierte Virialsumme

mit der theoretisch hergeleiteten annähernd deckungsgleich übereinstimmt.
Während des quasistationären Gleichgewichtszustandes bildet der Halo laut der Fachliteratur (vgl. [21]) eine Kern-Halo-Struktur (engl. *core-halo-structure*) aus, die sich durch einen kompakten Kern mit hoher Teilchendichte und einem diffusen Halo mit weit verstreuten Teilchen um den Kern herum auszeichnet. Weil Dunkle-Materie-Halos im thermodynamischen Sinne aufgrund ihrer negativen Wärmekapazität keinen Gleichgewichtszustand erreichen können und weiterhin das Virialtheorem erfüllen, kommt es letztendlich dazu, dass das System langfristig alle seine Teilchen verliert und „verdampft“. Die genauen Prozesse, wie Halos in diesen quasistationären Gleichgewichszustand übergehen, sind in der Kosmologie noch nicht vollständig verstanden und abschließend geklärt [5].
Im zweiten der Arbeit wurden das Modell um die Dunkle-Materie-Halos ein wenig realistischer gestaltet, indem nun sichtbare, baryonische Materie in Form von Gas in den Halo eingebettet wird, aus dem sich später weitere Strukturen, z. B. Sterne, gebildet haben. Nachdem zu Beginn des dritten Kapitels die theoretische Betrachtung rund um die Stabilität von Gaswolken mit dem Jeans-Kriterium erfolgte, wurden dann die unterschiedlichen Dichteprofile, nämlich das NFW-Profil für die dunkle Materie und das $\beta$-Profil für das Gas, thematisiert. Anschließend wurde mithilfe des hydrostatischen Gleichgewichts ein radiales Temperaturprofil hergeleitet, durch das das Gas mit seinem Gegendruck im Gleichgewicht zur Gravitation der dunklen Materie steht. Zum Abschluss wurde dann die Kühlzeit des Gases für verschiedene Halomassen mit dem radialen Profil der Gasdichte ermittelt. Es zeigt sich, dass die Kühlzeit für Halos kleinerer Massen geringer ist als für solche größerer Massen. Außerdem ist die Kühlzeit nahe des Zentrums im Gas aufgrund der höheren Dichte geringer als in weiter äußeren Bereichen.
Insgesamt hat die Beschäftigung mit diesem astronomischen Thema mir gezeigt, wie man mit einfachen Modellen, die Komplexität eines solchen Themas herunterbrechen kann, und dennoch einen Mehrwert daraus gewinnen kann. In den letzten Monaten habe ich viele, neue Dinge gelernt – sowohl fachlich als auch methodisch. Je mehr ich mich mit der Fachliteratur zur Forschung über Dunkle-Materie-Halos auseinander gesetzt habe, desto mehr habe ich erst festgestellt, wie umfangreich und komplex die Forschungsinhalte sind. Auch die Auseinandersetzung mit der Programmiersprache Python, der Implementierung und Verbesserung der durchgeführten Simulationen, dem Hochleistungsrechner *bwUniCluster* und dem vielseitigen Umgang mit der KI haben meine persönlichen Kompetenzen, die ich im Rahmen der Masterarbeit mitnehme, erprobt und vertieft. Die erworbenen Kompetenzen finden sicherlich einen Mehrwert später im Lehrberuf an der Schule.

Auch wenn die Arbeit auf vielen vereinfachten Annahmen basiert und vielleicht nicht unbedingt einen wichtigen Forschungsbeitrag für die Kosmologie liefert, zeigt diese Arbeit dennoch, dass mit gewissen Modellen abstrakte oder vom Verständnis schwer zugängliche Themen, mit denen Forscher eines Max-Planck-Instituts für Astrophysik sich tagtäglich beschäftigen, vereinfacht werden können. Dadurch kann die vereinfachte Behandlung dieses Thema einem breiteren Spektrum an Lesern zugänglich gemacht werden. Sicherlich bietet die Forschung zu Dunkle-Materie-Halos eine weiter- und tiefgehende Beschäftigung und damit Möglichkeiten für weitere Abschlussarbeiten (Bachelor-, Master- oder sogar Doktorarbeiten).

# Literaturverzeichnis


[1] S. J. Aarseth. Dynamical evolution of clusters of galaxies, I; Kapitel 1. , 126: 223, January 1963. doi: 10.1093/mnras/126.3.223.

[2] S. J. Aarseth. Dynamical evolution of simulated star clusters. I. Isolated models; Kapitel 1 (Introduction). *Astron. Astrophys., v. 35, no. 2, pp. 237-250*, 10 1974. ISSN AAEJA. URL `https://www.osti.gov/biblio/4229391`.

[3] S. J. Aarseth. Gravitational N-Body Simulations; Kapitel 7.4. 2003. URL `https://ui.adsabs.harvard.edu/abs/2003gnbs.book.....A`. Provided by the SAO/NASA Astrophysics Data System.

[4] S. W. Allen, R. W. Schmidt, and A. C. Fabian. Cosmological constraints from the X-ray gas mass fraction in relaxed lensing clusters observed with Chandra; Kapitel 3.1. , 334(2):L11–L15, August 2002. doi: 10.1046/j.1365-8711.2002.05601.x. URL `https://ui.adsabs.harvard.edu/abs/2002MNRAS.334L..11A`. Provided by the SAO/NASA Astrophysics Data System.

[5] Matthias Bartelmann. Das kosmologische Standardmodell: Grundlagen, Beobachtungen und Grenzen; Kapitel 7.1 und 7.4. Springer Spektrum, 2019. ISBN 978-3-662-59626-5. doi: 10.1007/978-3-662-59627-2. URL `https://doi.org/10.1007/978-3-662-59627-2`.

[6] Prof. Dr. Matthias Bartelmann. Persönliches Gespräch, Juni 2026.

[7] James Binney and Scott Tremaine. Galactic Dynamics: Second Edition, Kapitel 3.4, 4.7, 4.10, 7.1 und 9.2.1. Princeton University Press, 2008. URL `https://ui.adsabs.harvard.edu/abs/2008gady.book.....B`.

[8] T. Boekholt and S. Portegies Zwart. On the reliability of $N$-body simulations; Kapitel 1, 5.4.1 und 6.1-3. *Computational Astrophysics and Cosmology*, 2, 2015. doi: 10.1186/s40668-014-0005-3.

[9] J. Bovy. Dynamics and Astrophysics of Galaxies, Kapitel 17.3. Princeton University Press, Princeton, NJ, 2026. URL `https://galaxiesbook.org/`.

[10] Manuel Brea-Carreras, Michael Thiel, and Markus Pössel. Simulating Tidal Interactions between Galaxies: A Pre-University Student Project; Kapitel 3.1, 2018. URL `https://arxiv.org/abs/1812.00779`.

[11] Alfonso Cavaliere and Roberto Fusco-Femiano. X-rays from hot plasma in clusters of galaxies, Kapitel 2. *Astronomy and Astrophysics*, 49:137–144, 04 1976.

[12] Man Ho Chan. Two Analytic Relations Connecting the Hot Gas Astrophysics with the Cold Dark Matter Model for Galaxy Clusters, Kapitel 2. *The Astrophysical Journal*, 923(1):95, December 2021. ISSN 1538-4357. doi: 10.3847/1538-4357/ac32c4. URL `http://dx.doi.org/10.3847/1538-4357/ac32c4`.

[13] Y. Chen, T. H. Reiprich, H. Böhringer, Y. Ikebe, and Y.-Y. Zhang. Statistics of X-ray observables for the cooling-core and non-cooling core galaxy clusters; Table 1. *Astronomy amp; Astrophysics*, 466(3):805–812, April 2007. ISSN 1432-0746. doi: 10.1051/0004-6361:20066471. URL `http://dx.doi.org/10.1051/0004-6361:20066471`.

[14] Camila A. Correa, J. Stuart B. Wyithe, Joop Schaye, and Alan R. Duffy. The accretion history of dark matter haloes – III. A physical model for the concentration–mass relation; Kapitel 4.5, volume=452. *Monthly Notices of the Royal Astronomical Society*, (2):1217–1232, 2015. ISSN 1365-2966. doi: 10.1093/mnras/stv1363. URL `http://dx.doi.org/10.1093/mnras/stv1363`.

[15] A. J. Cuesta, F. Prada, A. Klypin, and M. Moles. The virialized mass of dark matter haloes, Kapitel 1. *Monthly Notices of the Royal Astronomical Society*, 389(1):385–397, 2008. ISSN 1365-2966. doi: 10.1111/j.1365-2966.2008.13590.x. URL `http://dx.doi.org/10.1111/j.1365-2966.2008.13590.x`.

[16] W. J. G. de Blok. The Core-Cusp Problem. *Advances in Astronomy*, 2010 (1), November 2009. ISSN 1687-7977. doi: 10.1155/2010/789293. URL `http://dx.doi.org/10.1155/2010/789293`.

[17] Farbod Hassani. From Spherical Collapse to Virialized Dark Matter Halos; Kapitel 1.2-3, 3.3-5, may 2026. URL `https://www.uio.no/studier/emner/matnat/astro/AST3220/v26/timeplan/course3.pdf`. Preprint.

[18] Arthur Hebecker. Vorlesungsskript – Theoretische Physik II: Analytische Mechanik und Statistik/Thermodynamik sowie mathematische Methoden der

theoretischen Physik, Kapitel 2.7. Universität Heidelberg, 2021. URL `https://www.thphys.uni-heidelberg.de/~hebecker/TP2/tp2.pdf`. Nachbearbeitete Version.

[19] M. Hoskin. Stukeley's Cosmology and the Newtonian Origins of Olbers's Paradox, S. 86-91. *Journal for the History of Astronomy*, 16:77, June 1985. doi: 10.1177/002182868501600201.

[20] Illustris Collaboration. The Illustris Simulation: Media and Data Release; Redshift evolution of a whole box slice from z=4 to z=0, showing four projections: dark matter density, gas density, gas temperature, and gas metallicity. `https://www.illustris-project.org/media/`. Letzter Zugriff am 11. August 2026.

[21] M. Joyce, B. Marcos, and F. Sylos Labini. Dynamics of finite and infinite self-gravitating systems with cold quasi-uniform initial conditions, Kapitel 2.1. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(04):P04019, apr 2009. doi: 10.1088/1742-5468/2009/04/P04019.

[22] Kyle Kabasares. Dynamical Friction in 1 Minute #VeritasiumContest. YouTube video, 2021. URL `https://www.youtube.com/watch?v=5fBvKb2JD9c`. Veröffentlicht am 20.07.2021; Letzter Zugriff am 27. Juli 2026.

[23] Rainer Kayser and Redaktion Welt der Physik. Hubblekonstante bleibt rätselhaft, Dezember 2024. URL `https://www.weltderphysik.de/gebiet/universum/nachrichten/2024/kosmologie-hubble-konstante-bleibt-raetselhaft/`. Letzter Zugriff am 26. Juli 2026.

[24] Cedric Lacey and Shaun Cole. Merger rates in hierarchical models of galaxy formation, Kapitel 1 (Introduction). *Monthly Notices of the Royal Astronomical Society*, 262(3):627–649, 1993. doi: 10.1093/mnras/262.3.627.

[25] Yan Levin, Renato Pakter, and Felipe B. Rizzato. Collisionless relaxation in gravitational systems: From violent relaxation to gravothermal collapse; Seite 1. *Physical Review E*, 78(2), August 2008. ISSN 1550-2376. doi: 10.1103/physreve.78.021130. URL `http://dx.doi.org/10.1103/PhysRevE.78.021130`.

[26] D. Lynden-Bell. Statistical mechanics of violent relaxation in stellar systems; Kapitel Introduction. , 136:101, Jan 1967. doi: 10.1093/mnras/136.1.101. URL `https://ui.adsabs.harvard.edu/abs/1967MNRAS.136..101L`.

[27] Michela Mapelli. N-body techniques for astrophysics: Lecture 5 – N-body methods for collisionless systems. PhD School in Astrophysics, University of Padova. URL `https://web.oapd.inaf.it/mapelli/2018_Nbody5.pdf`. Lecture notes from November 19–30, 2018.

[28] Julio F. Navarro, Carlos S. Frenk, and Simon D. M. White. The Structure of Cold Dark Matter Halos, Kapitel 3.1 und 4.2. *The Astrophysical Journal*, 462: 563, May 1996. ISSN 1538-4357. doi: 10.1086/177173. URL `http://dx.doi.org/10.1086/177173`.

[29] Julio F. Navarro, Carlos S. Frenk, and Simon D. M. White. A Universal Density Profile from Hierarchical Clustering, Kapitel 1 und 3.2. *The Astrophysical Journal*, 490(2):493–508, December 1997. ISSN 1538-4357. doi: 10.1086/304888. URL `http://dx.doi.org/10.1086/304888`.

[30] Werner Neundorf. Numerik gewöhnlicher Differentialgleichungen – Anfangswertprobleme, Kapitel 1.3. Preprint M 11/06, Technische Universität Ilmenau, Institut für Mathematik, Februar 2011. URL `https://www.db-thueringen.de/servlets/MCRFileNodeServlet/dbt_derivate_00022563/IfM_Preprint_M11_06.pdf`.

[31] Pennsylvania State University. The Atmosphere's Pressure Structure: Hydrostatic Equilibrium. METEO 300: Fundamentals of Atmospheric Science. URL `https://www.e-education.psu.edu/meteo300/node/7`. Letzter Zugriff am 6. September 2026.

[32] LEIFI Physik. Erzeugung von RÖNTGEN-Strahlung. URL `https://www.leifiphysik.de/atomphysik/roentgen-strahlung/grundwissen/bremsstrahlung`. Letzter Zugriff am 10. September 2026.

[33] H. C. Plummer. On the problem of distribution in globular star clusters; Gleichung 11. *Monthly Notices of the Royal Astronomical Society*, 71(5):460–470, 1911. doi: 10.1093/mnras/71.5.460.

[34] C. Power, J. F. Navarro, A. Jenkins, C. S. Frenk, S. D. M. White, V. Springel, J. Stadel, and T. Quinn. The Inner structure of ΛCDM haloes — I. A numerical convergence study; Kapitel 1 (Introduction) und 6. *Monthly Notices of the Royal Astronomical Society*, 338(1):14–34, Jan 2003. doi: 10.1046/j.1365-8711.2003.05925.x.

[35] Markus Pössel. Introduction to Astronomy Astrophysics II: Cosmology and Galaxies, Kapitel 5.4.1, 7.2 und 7.4. University of Heidelberg, 6th edition, September 2025.

[36] Peter Schneider. Einführung in die extragalaktische Astronomie und Kosmologie, Kapitel 6.2.4, 6.3.2 und 7.5.1. Springer, Berlin, Heidelberg, 1 edition, 2006. ISBN 978-3-540-25832-2.

[37] Karl Sims. How to generate random points in N-dimensional spheres, Abschnitt: Random in 3-sphere, 2024. URL `https://www.karlsims.com/random-in-sphere.html`. Letzter Zugriff am 1. September 2026.

[38] Roland Speith and Christoph Schäfer. Projekt 1: Das klassische N-Körper-Problem, Kapitel 1.3.3, 10 2022. URL `https://www.tat.physik.uni-tuebingen.de/~schaefer/teach/f/nbody.pdf`. Letzter Zugriff am 19. Juli 2026.

[39] Ralph S. Sutherland and M. A. Dopita. Cooling Functions for Low-Density Astrophysical Plasmas; S.307f. , 88:253, Sep. 1993. doi: 10.1086/191823. URL `https://ui.adsabs.harvard.edu/abs/1993ApJS...88..253S`. Provided by the SAO/NASA Astrophysics Data System; Daten für primordiales Gas sind auf S. 307f. zu finden.

[40] Francesco Sylos Labini. Violent and mild relaxation of an isolated self-gravitating uniform and spherical cloud of particles: Relaxation of a spherical cloud of particles; kapitel 1 (introduction). *Monthly Notices of the Royal Astronomical Society*, 423(2):1610–1622, May 2012. ISSN 0035-8711. doi: 10.1111/j.1365-2966.2012.21019.x. URL `http://dx.doi.org/10.1111/j.1365-2966.2012.21019.x`.

[41] Sylos Labini, Francesco and Capuzzo-Dolcetta, Roberto. Properties of self-gravitating quasi-stationary states; Kapitel 1. *AA*, 643:A118, 2020. doi: 10.1051/0004-6361/202039358. URL `https://doi.org/10.1051/0004-6361/202039358`.

[42] tec science. Gleichverteilungssatz (Äquipartitionstheorem), 21. März 2019. URL `https://www.tec-science.com/de/thermodynamik-waermelehre/kinetische-gastheorie/gleichverteilungssatz-aquipartitionstheorem/`. Letzter Zugriff am 10. September 2026.

# Anhang A

# Promptverzeichnis

Im Sinne der Dokumentationspflicht zur Nutzung von KI-Tools in wissenschaftlichen Abschlussarbeiten beinhaltet dieser Anhang eine Auflistung über die Prompts des verwendeten KI-Tools (Gemini von Google, Version: 3.6 Flash) zu den Python-Codes, mit denen die für die Abschlussarbeit durchgeführten Simulationen erstellt worden sind. Die folgende Beschreibung dient als kurzer Bericht dessen, was mir bei der Nutzung und dem Umgang mit der KI aufgefallen ist: Generell besteht die „Kunst“ eines gut geschriebenen Prompts darin, diesen so zu verfassen, dass er alle notwendigen Informationen für die KI enthält, um den Code so zu generieren, dass er die jeweiligen Zwecke erreicht, wobei die KI sicherlich auch ihrerseits Vorschläge zur Änderung oder Verbesserung der Codes vorschlagen wird. Daher sollte der KI auch der Kontext, für den der Code generiert wird – in unserem Fall: N-Körper-Simulationen zum Kollaps von Dunkle-Materie-Halos und der anschließende Miteinbezug von kosmischen Gas bei der Gaskühlung – erklärt werden. Weiterhin ist es bei der Verwendung der KI empfehlenswert, die Codes im selben Chatverlauf erstellen oder verändern zu lassen und verschiedene Inhalte/Themen nicht im selben, sondern in unterschiedlichen Chats zu behandeln. So wurden für die Masterarbeit mehrere Chats einmal für das Thema „N-Körper-Simulation zum Halokollaps“ und einmal für das Thema „Kosmisches Gas/Gaskühlung“ geführt. Außerdem ist es auch sinnvoll, die „Gespräche“ mit der KI auf mehrere Chats zu verteilen, da die KI bei sehr langen Chatverläufen fehleranfällig wird. Dies zeigt sich dadurch, dass die KI dann Sachen, die vorher im Chat schon einmal erwähnt worden sind, vergisst oder sonstige Fehler macht, die manche Leute umgangssprachlich als „Halluzinationen“ beschreiben.

Nichtsdestotrotz können die im Folgenden aufgelisteten Prompts nicht immer zu 100% den gewünschten Code generieren, sondern es bedarf immer einer selbstständigen, aufmerksamen Lektüre und ggf. Korrektur des Codes für die eigenen Zwecke. Die Entwicklung der Python-Codes erfolgte nicht über einen einzigen Prompt, son-

dern über mehrere Wochen/Monate hinweg in sehr langen Chatverläufen mit der KI. Die Codes mussten zudem immer wieder angepasst werden, unter anderem auch mithilfe von Rückmeldungen und Hilfen der Betreuer dieser Masterarbeit (Dr. Markus Pössel und Prof. Dr. Matthias Bartelmann). Die folgenden Prompts sind daher als Zusammenfassungen von der KI zu verstehen, um mit diesen dann die final verwendeten Codes zu generieren.

## A.1 Prompts für die Python-Codes zu den Abbildungen, Simulationen und Ergebnissen in Kapitel 2

- Abb. 2.1: Zeitlicher Verlauf in Einheit der Kollapszeit $t_c$ von der Expansion bis zum Kollaps einer überdichten Sphäre.

```
Erstelle einen Python-Code zum Modell des sph rischen
   Kollaps:
1. Verwendetes Modell: Trigonometrische Parametrisierung
   einer ueberdichten Sphaere r(\xi) und t(\xi) mit der
   Hilfsvariablen \xi \in [0, 2\pi].
2. Parameter:
   - R = 1.0 (Maximale Amplitude / Skalierungsfaktor)
   - tc = 1.0 (Kollapszeit)
   - xi = np.linspace(0, 2 * np.pi, 1000)
3. Parametrische Gleichungen:
   - r = (R / 2.0) * (1.0 - np.cos(xi))
   - t = (tc / (2.0 * np.pi)) * (xi - np.sin(xi))

Visualisierungs-Vorgaben (Matplotlib):
- Figure-Groesse: (9, 6)
- Plot-Linie: Farbe "b", Linienbreite 2.5, Label "Ueberdichte
   Sphaere"
- Scatter-Punkt & Annotation 1 (Turn-around):
  * Berechne t_turn und r_turn bei xi = np.pi
  * Rot markierter Punkt bei (t_turn, r_turn) mit zorder=5
  * Annotation mit Text: "Turn-around (Maximale
     Ausdehnung)\n$\\xi = \\pi$, $r = R_{max}$",
     Pfeilformatierung (facecolor="black", shrink=0.05,
```

```
    width=0.8, headwidth=4), Position xytext=(t_turn - 0.2,
    r_turn - 0.3)
- Annotation 2 (Kollaps-Endpunkt):
  * Berechne t_collaps = tc und r_collaps = 0.0
  * Annotation mit Text: "Kollaps bei \n$\\xi = 2\\pi$, $t =
    t_c$", Pfeilformatierung (facecolor="black",
    shrink=0.05, width=1, headwidth=6), Position
    xytext=(t_collaps - 0.3, r_collaps + 0.2)
- Achsenbeschriftungen & Grenzen:
  * xlabel: r"Zeit in Einheiten der Kollapszeit $t_c$"
    (fontsize=12)
  * ylabel: "Radius $r(t)$ in Einheiten von $R_{max}$"
    (fontsize=12)
  * xlim: (0, tc * 1.1)
  * ylim: (0, R * 1.2)
- Layout & Style: Grid aktiviert (linestyle=":", alpha=0.6),
   Legende oben rechts ("upper right"), tight_layout() und
   plt.show().
Code-Struktur:
- Nutze Kommentare zur Gliederung: # --- 1. PARAMETER
   FESTLEGEN ---, # --- 2. PARAMETRISCHE GLEICHUNGEN
   BERECHNEN ---, # --- 3. PLOTTEN DER ZYKLOIDE ---.
-Bibliotheken: 'numpy as np' und 'matplotlib.pyplot as plt'.
```

- Abb. 2.4: Qualitativer Vergleich zwischen dem Newtonschen Gravitationsgesetz mit und ohne dem softening-Parameter (hier exemplarisch mit $\epsilon = 0.25$).

```
Erstelle einen Python-Code zum Plummer-Softening vs.
   Newton-Gravitation:
1. Parameter:
   - Softening: epsilon = 0.25
   - Massen und Konstante: M = 1.0, m = 1.0, G = 1.0
2. Daten-Arrays:
   - r = np.linspace(0.0, max(1.5, epsilon * 10), 2000)
   - r_newton = np.linspace(0.01, max(1.5, epsilon * 5), 2000)
3. Kraft-Berechnungen:
   - F_softened = (G * M * m * r) / (r**2 + epsilon**2)**(1.5)
   - F_newton = (G * M * m) / (r_newton**2)
4. Mathematisches Maximum der Gesofteten Kraft:
```

```
   - r_max = epsilon / np.sqrt(2)
   - F_max = (G * M * m * r_max) / (r_max**2 +
      epsilon**2)**(1.5)

Visualisierungs-Vorgaben (Matplotlib):
- Figure-Groesse: (11, 6.5), dpi=100
- Plot 1 (Newton-Kraft):
  * Linie: color='firebrick', linestyle='-', linewidth=2.5
  * Label: 'Newton-Kraft ohne Softening ($F \\propto r^{-2}$)'
- Plot 2 (Plummer-Kraft):
  * Linie: color='dodgerblue', linewidth=2.5
  * Label: f'Newton-Kraft mit Softening ($F \\propto
     \\frac{{r}}{{(r^2 + \\epsilon^2)^{{3/2}}}}$)'
- Plot-Elemente & Markierungen:
  * Gruener Punkt bei (r_max, F_max): 'go', markersize=5,
     zorder=5
  * Textbeschriftung fuer Maximum: plt.text(r_max * 0.5,
     F_max * 1.05, f' ($F_{{max}} \\propto 1/\\epsilon^2$)',
     color='forestgreen', fontweight='bold', fontsize=15)
- Achsenbeschriftungen, Limits & Styling:
  * xlabel: 'Abstand $r$', fontsize=14
  * ylabel: 'Gravitationskraft $F(r)$', fontsize=14
  * xlim: (0, max(1.2, epsilon * 5))
  * ylim: (0, F_max * 2)
  * Grid: plt.grid(True, linestyle=':', alpha=0.6)
  * Legende: plt.legend(fontsize=16, loc='upper right')
  * Layout: plt.tight_layout() und plt.show()
- Verwendete Bibliotheken: 'numpy as np' und
   'matplotlib.pyplot as plt'.
```

- Abb. 2.6: Qualitativer Vergleich der Teilchenverteilung innerhalb einer Sphäre mit $R_{max} = 50$ für exemplarisch $N = 500$ Teilchen mit Farbcodierung: Im linken Bild ist die Teilchenanzahl in jeder Kugelschale konstant ($\rho_{Kugel}$ ist inhomogen) und im rechten Bild nicht konstant ($\rho_{Kugel}$ ist homogen).

```
Erstelle einen Python-Code zu einer homogenen
   Teilchenverteilung in einer Kugel:
1. Modell: Vergleich einer falschen (linear in r) vs.
   korrekten
```

```
   (Inverse Transform Sampling via u^(1/3)) homogenen
   Raumausfuellung einer Kugel.
2. Parameter:
   - np.random.seed(42)
   - N = 500
   - R_max = 50.0
3. Funktion ‘generate_sphere_particles(n, r_max,
   use_cube_root=True)‘:
   - Winkel: phi ~ Uniform(0, 2*pi), cos_theta ~ Uniform(-1,
      1),
     sin_theta = sin(arccos(cos_theta))
   - Radius: u ~ Uniform(0, 1)
   - Wenn use_cube_root=True: r = r_max * (u ** (1.0 / 3.0))
     (dV proportional zu r^2 dr)
   - Wenn use_cube_root=False: r = r_max * u
     (Dichte proportional zu 1/r^2)
   - Kartesische Koordinaten: x = r*sin_theta*cos(phi),
     y = r*sin_theta*sin(phi), z = r*cos_theta
   - Rueckgabe: x, y, z, r

Visualisierungs-Vorgaben (Matplotlib 3D):
- Figure-Groesse: (12, 6), Colormap: plt.cm.plasma
- Subplot 1 (Falsch: r = R * u):
  * 3D-Scatter (add_subplot(121, projection="3d"))
  * Parameter: c=r_wrong, cmap=cmap, vmin=0, vmax=R_max, s=50,
    edgecolor="k", linewidth=0.5, alpha=0.9
  * Title: r"Falsch: $r = R \cdot u$" "\n(Starke Verklumpung
     im Zentrum)"
  * Achsenlimits: [-R_max, R_max] fuer x, y, z
  * Gitter explizit einschalten mit ax1.grid(True)
- Subplot 2 (Korrekt: r = R * u^(1/3)):
  * 3D-Scatter (add_subplot(122, projection="3d"))
  * Parameter identisch zu Subplot 1 (c=r_correct)
  * Title: r"Korrekt: $r = R \cdot u^{1/3}$" "\n(Homogene
     Raumerfuellung)"
  * Achsenlimits: [-R_max, R_max] fuer x, y, z
  * Gitter explizit einschalten mit ax2.grid(True)
- Achsenanzeige: KEIN ax.axis("off") verwenden, damit das
   3D-Gitter
```

```
  und die Drahtmodell-Box sichtbar bleiben.
- Colorbar-Konfiguration (Links platziert):
  * Achsenposition: fig.add_axes([0.06, 0.2, 0.02, 0.6])
  * Label: r"Abstand $r$ zum Zentrum (von 0 bis $R_{max}$)",
     fontsize=10
  * Ticks und Label links von der Colorbar
     (yaxis.set_ticks_position("left"))
- Subplot-Anpassung: plt.subplots_adjust(left=0.09,
   right=0.98,
  wspace=-0.05, top=0.85, bottom=0.1)
- Abschluss: plt.show()
- Verwendete Bibliotheken: 'matplotlib.pyplot as plt' und
   'numpy as np'.
```

- N-Körper-Simulationen zu Abb. 2.7 bis 2.19, durchgeführt auf dem Hochleistungscomputersystem *bwUniCluster*. Hinweis: Der Code ist hier mit $N = 2000$ und dem entsprechenden softening aufgeführt. Die Simulationen mit anderen Teilchenzahlen $N$ basieren analog auf dem Code.

```
Erstelle einen Python-Code einer N-Koerper-Simulation zum
   Kollaps eines Halos, die dann auf einem
   Hochleistungsrechner gerechnet werden:
1. Physikalische Grundlagen & Kraftberechnung:
   - Gravitationskonstante: G_konst = 1.0
   - Implementiere eine Funktion
      `compute_accelerations_pot_and_csoft(pos, masses,
      softening)`:
     * Berechne Beschleunigungen via `np.einsum` unter
        Ber cksichtigung des Softening-Parameters
        (`softening**2`).
     * Berechne die potenzielle Energie: E_pot = -G * sum(m_i
        * m_j / sqrt(r^2 + eps^2)).
     * Berechne den Softening-Korrekturterm: C_soft = -G *
        sum(m_i * m_j * eps^2 / (r^2 + eps^2)^(3/2)).
2. Checkpoints setzen:
   - Implementiere `save_checkpoint(filename, state_dict)`
      (atomsicher via temporaere .tmp-Datei und os.replace)
      sowie `load_checkpoint(filename)`.
```

```
3. Hauptsimulationsfunktion
   (‘simulate_collapse_asym_3d_adaptive‘):
   - Parameter: n_particles=2000, t_end=13725.0, dt_max=0.01,
     run_index=1, ordner_name="Sim N2000 ep_max"
   - Geometrie & Massen: R_max = 50.0, Teilchenmassen m_i =
     1.0 / n_particles, Softening-Laenge eps = 3.2, Binning
     fuer Massenprofil (50 Bins von 0 bis R_max).
   - Checkpoint-Pruefung: Falls .pkl-Datei existiert, Zustand
     laden und Beschleunigung reinitialisieren; andernfalls
     homogene Kugel im Ortsraum (Zufallsradien via
     Kubikwurzel) und Geschwindigkeiten v=0 initialisieren
     (Schwerpunkt-Korrektur fuer Ort und Geschwindigkeit).
   * Radien via Kubikwurzel-Zufallsverteilung: r = R_max *
     (u^(1/3)) mit u ~ Uniform(0,1).
   * Raumwinkel isotrop verteilen: Azimutwinkel phi ~
     Uniform(0, 2*pi) und Polarwinkel theta via
     arccos(Uniform(-1, 1)).
   * Geschwindigkeiten v = 0 setzen und Schwerpunkt von Ort
     sowie Geschwindigkeit auf den Ursprung zentrieren.
   - Integrator (Velocity Verlet / KDK):
     * Adaptive Zeitschrittsteuerung: dt = min(dt_max,
       max(1e-5, eta * sqrt(eps / a_max))) mit eta = 0.2.
       Exakter Abgleich mit t_end.
     * Ausfuehrung des Kick-Drift-Kick-Schemas mit
       kontinuierlicher Schwerpunktsbereinigung.
   - Diagnostik & Datenerfassung (alle ‘save_every = 200‘
     Schritte):
     * Berechnung von E_kin, E_pot, E_ges und Impulsbetrag.
     * Ermittlung des Medianradius R_med ueber alle
       gebundenen Teilchen (E_ges,einzeln < 0) innerhalb von
       R_max, was als Energie- und Radiusfilter fungiert.
       Fallback auf den letzten gueltigen Wert.
     * Erfassung von Virialprodukt (pq_aktuell = sum(m * v *
       r)), Massenprofil-Histogramm (np.histogram) und
       C_soft.
   - Periodisches Speichern des Checkpoints alle
     ‘checkpoint_every = 10000‘ Schritte. Nach Abschluss der
     Simulation wird die Checkpoint-Datei entfernt.
   - Rueckgabewerte: (radii, zeit_hist, e_kin_hist,
```

```
      e_pot_hist, e_ges_hist, p_ges_hist, pq_hist,
      c_soft_hist, bins, massen_profil_hist).

Ausfuehrungsblock (‘__main__‘):
- CLI-Argumente: Ueberpruefe ‘sys.argv‘ auf Parameter
   ‘--run_id=X‘ (Standard: 1).
- Ordnererstellung via ‘os.makedirs‘.
- Existent-Pruefung der Ziel-NPZ-Datei
   ‘kollaps_{TEILCHENZAHL}_run_{run_index}.npz‘: Bei Existenz
   ueberspringen, andernfalls Simulation ausfuehren.
- Speichern aller Simulationsdaten in der .npz-Datei mittels
   ‘np.savez‘.
Bibliotheken: ’os’, ’sys’, ’pickle’, ’numpy as np’.
```

- Zeitliche Entwicklung des Halo-Kollapses (zu Abb. 2.7 bis 2.10.) und zur Virialsumme (zu Abb. 2.11 bis 2.14) – Hinweis: Der Code verwendet exemplarisch den Code zu den gespeicherten Daten für $N = 50$ Teilchen mit dem entsprechenden softening. Für die anderen Teilchenzahlen sind die Codes analog aufgebaut.

```
Erstelle einen Python-Code zur statistischen Auswertung der
   Daten von den N-Koerper-Simulationen:
1. Datei-Verarbeitung & Einlesen:
   - Parameter: R_max = 50.0, t_c_200 = 914
   - Erzeuge eine Liste ‘dateien‘ ueber os.path.join("Sim50",
      "N50_epmax_neu", f"kollaps_50_run_{i}.npz") fuer i in
      range(1, 101).
   - Pruefe mit FileNotFoundError, falls die Liste leer ist.
   - Ausgabe der Anzahl gefundener Laeufe per print().
2. Zeitachsen-Interpolation:
   - Gemeinsame Zeitachse: t_gemeinsam = np.linspace(0.5, 15,
      1000)
   - Lade pro Datei ’radii’, ’t_hist’, ’e_kin’, ’e_pot’,
      ’pq_hist’, ’c_soft_hist’.
   - Skaliere die Zeitachse des Laufs: t_skala_lauf = t_hist
      / t_c_200 + 0.5
   - Interpoliere alle Groessen (np.interp) auf t_gemeinsam:
     * Radii -> interpolierte_radien
```

```
     * Virialsumme (2.0 * e_kin + e_pot) ->
        interpolierte_virial_summe
     * Tafel-Endwert (pq_hist / t_hist, abfangen mit
        np.errstate, t_hist==0 auf 0.0) ->
        interpolierte_tafel_endwerte
     * Modifiziertes Gleichgewicht (tafel_endwert_lauf +
        c_soft_hist) ->
        interpolierte_modifiziertes_gleichgewicht
   - Wandle die Listen via np.vstack in Matrizen um.
3. Statistik-Berechnungen (axis=0):
   - Berechne Mittelwerte (np.mean) und Standardabweichungen
      (np.std) fuer Radien und Virialsumme.
   - Berechne Mittelwerte fuer Tafel-Endwert und
      Modifiziertes Gleichgewicht.

Visualisierungs-Vorgaben (Matplotlib):
- Plot 1 (Radius-Entwicklung, figsize=(12, 6)):
  * Analytische Zykloide plotten: theta = np.linspace(0,
     30*np.pi, 2000), radii =
     (R_max/2.0)*(1.0-np.cos(theta)), t =
     (1.0/(2.0*np.pi))*(theta-np.sin(theta)), color="blue",
     alpha=0.4, linewidth=2, Label="Symmetrischer Kollaps
     (Theorie)".
  * Einzellaeufe aus interpolierte_radien im Hintergrund
     plotten (color='gray', linestyle='--', alpha=0.08,
     linewidth=1).
  * Mittelwert-Linie: color="darkred", linewidth=2.5, Label
     r"Durchschnitt der Medianradien $(R_{\text{med}})$".
  * Streuung mit fill_between: color="red", alpha=0.15, Label
     r"Std. abw. $R_{\text{med}}$".
  * Virialradius-Linie: plt.axhline(y=R_max/2, color="green",
     linestyle="--", linewidth=2, Label r"Virialradius
     $R_{\text{vir}} = R_{\text{max}}/2$").
  * Achsen & Styling: xlim=(0.5, 15), ylim=(0, R_max*1.3),
     MajorLocator(1.0) auf X-Achse, Title="Statistische
     Auswertung des 3D-Kollapses: Medianradien", Legende oben
     links (ncol=2, fontsize=11), Grid aktiviert.

- Plot 2 (Virial-Summe vs. Theoretischer Endwert,
```

```
   figsize=(12, 6)):
  * Einzellaeufe aus interpolierte_virial_summe im
     Hintergrund plotten (color='gray', linestyle='--',
     alpha=0.08, linewidth=1).
  * Simulierte Virialsumme: color="darkgreen", linewidth=2.5,
     Label r"Simulierte Summe $\langle 2E_{\text{kin}} +
     E_{\text{pot}} \rangle$".
  * Streuung mit fill_between: color="green", alpha=0.15,
     Label r"Streuung ($\sigma$)".
  * Klassischer Endwert: color="red", linestyle="--",
     linewidth=2.0, Label r"Klassischer Endwert $\langle
     \frac{p \cdot x}{\tau} \rangle$".
  * Modifiziertes Gleichgewicht: color="fuchsia",
     linestyle="--", linewidth=2.5, Label r"Zusaetzlicher
     Endterm durch $\varepsilon$ $\langle \frac{p \cdot
     x}{\tau} + C_{\text{soft}} \rangle$".
  * Zero-Line: plt.axhline(y=0.0, color="black",
     linestyle=":", linewidth=1.2, Label r"Ideales
     Gleichgewicht ($\tau \to \infty$)").
  * Achsen & Styling: xlim=(0.5, 15.0), ylim=(-0.025, 0.025),
     Y-Achse MajorLocator(0.01), X-Achse MajorLocator(1.0),
     Title="Verlauf der Virial-Summe vs. theoretischer
     Endwert", Legende oben rechts (fontsize=13), Grid
     aktiviert.

Konsolen-Ausgabe:
- Drucke am Ende statistische Kennzahlen der Virial-Analyse
   aus (Mittelwerte des letzten Zeitschritts [-1] fuer
   simulierte Summe, Tafel-Endwert, Mod. Gleichgewicht sowie
   deren Differenzen).
Bibliotheken: 'os', 'matplotlib.pyplot as plt',
   'matplotlib.ticker as ticker', 'numpy as np'.
```

- Aufbauend auf dem vorherigen Code: Entfernungsverteilung der Teilchen (zu Abb. 2.16 bis 2.19)

```
Erstelle einen Python-Code zu einer kumulativen
   Entfernungsverteilung der Teilchen:
1. Datei-Verarbeitung & Parameter:
```

```
   - Parameter: R_max = 50.0, t_c_200 = 914, N_gesamtzahl = 50
   - Erzeuge eine Liste ‘dateien‘ ueber os.path.join("Sim50",
     "N50_epmax_neu", f"kollaps_50_run_{i}.npz") fuer i in
     range(1, 101).
   - Pruefe mit FileNotFoundError, falls die Liste leer ist.
   - Print-Ausgabe der Anzahl gefundener Laeufe.
2. Zeitachsen-Interpolation & Profil-Verarbeitung:
   - Gemeinsame Zeitachse: t_gemeinsam = np.linspace(0.5, 15,
     1000)
   - Lade pro Datei 't_hist', 'massen_profil_hist' (Form:
     [Zeitschritte, Bins]) und 'bins'.
   - Skaliere die Zeitachse: t_skala_lauf = t_hist / t_c_200
     + 0.5
   - Berechne die kumulative Summe ueber die Bins:
     kumulativ_lauf = np.cumsum(massen_profil_hist, axis=1)
   - Normalisiere auf Gesamtzahl: kumulativ_lauf_norm =
     kumulativ_lauf / N_gesamtzahl
   - Interpoliere fuer jeden Bin einzeln (range(n_bins)) auf
     t_gemeinsam (np.interp).
   - Wandle die Ergebnisse in eine Matrix um:
     matrix_kumulativ = np.array(interpolierte_kumulativ)
   - Berechne den Ensemble-Median ueber alle Laeufe:
     median_kumulativ = np.median(matrix_kumulativ, axis=0)
3. Ausgewaehlte Zeitschritte:
   - Zielzeiten: gewaehlte_zeiten = [0.5, 1.0, 2.0, 3.0, 5.0,
     7.0, 10.0]

Visualisierungs-Vorgaben (Matplotlib):
- Figure-Groesse: (10, 6)
- Plot-Schleife ueber ‘gewaehlte_zeiten‘:
  * Finde Naeherungsindex: idx = np.argmin(np.abs(t_gemeinsam
    - t_ziel))
  * Randpunkt-Fixing (Weg B): x_werte = bins, y_werte =
    np.pad(median_kumulativ[idx, :], (1, 0), mode="constant")
  * Linien-Plot: linewidth=2, Label rf"$t/t_c =
    {t_gemeinsam[idx]:.1f}$"
- Styling:
  * xlabel: r"Radius $R$ der Kugel", fontsize=14
  * ylabel: r"Kumulativer Anteil der Teilchen $N(<r) /
```

```
    N_{\text{gesamt}}$", fontsize=14
  * title: "Kumulative Entfernungsverteilung der Teilchen",
    fontsize=16
  * tick_params: labelsize=12
  * xlim: (0, R_max)
  * ylim: (0, 1.05)
  * Grid: plt.grid(True, linestyle="--", alpha=0.3)
  * Legende: plt.legend(loc="upper left", fontsize=12)
  * Layout: plt.tight_layout() und plt.show()
 Bibliotheken: 'os', 'matplotlib.pyplot as plt', 'numpy as
   np'.
```

## A.2 Prompts für die Python-Codes zu den Abbildungen und Ergebnissen in Kapitel 3

- Abb. 3.2: NFW-Profil

```
Erstelle einen Python-Code zum NFW-Dichteprofil:
1. Mathematisches Gitter & Profil:
   - Normierte Koordinate: x = r / r_s ueber einen
     logarithmischen Bereich von 10^-2 bis 10^2 mit 500
     Punkten (np.logspace(-2, 2, 500)).
   - Normiertes NFW-Profil: rho_normiert = 1.0 / (x * (1.0 +
     x)**2)

Visualisierungs-Vorgaben (Matplotlib):
- Figure-Groesse: figsize=(10, 6)
- Plot-Inhalt:
  * Doppel-logarithmische Achsen via plt.loglog(x,
    rho_normiert, color='black', lw=2, label=r'NFW-Profil').
  * Vertikale Hilfslinie beim Skalenradius: plt.axvline(1.0,
    color='red', linestyle='--', label=r'Skalenradius $x_s$
    = 1').
- Styling & Achsen:
  * xlabel: r' Radius $x_s$ = $r / r_s$'
  * ylabel: r' Dichte $\rho/\rho_s$'
  * title: 'Universelles NFW-Dichteprofil'
  * Grid: plt.grid(True, which='both', ls='--')
```

```
  * Legende: plt.legend()
  * Ausf hrung: plt.show()
Bibliotheken: 'numpy as np', 'matplotlib.pyplot as plt'.
```

- Abb. 3.3: $\beta$-Profil

```
Erstelle einen Python-Code zum Beta-Dichteprofil:
1. Parameter & mathematische Definition:
   - Beta-Parameter: beta = 2/3
   - Dimensionsloser Radius: x_c = r / r_c  ber  einen
      logarithmischen Bereich von 10^-2 bis 10^2 mit 500
      Punkten (np.logspace(-2, 2, 500)).
   - Normierte Gasdichte: rho_normalized = (1.0 +
      x_c**2)**(-1.5 * beta)

Visualisierungs-Vorgaben (Matplotlib):
- Figure-Groesse: figsize=(8, 6)
- Plot-Inhalt:
  * Doppel-logarithmische Achsen (plt.xscale('log') und
     plt.yscale('log')).
  * Linienplot: x_c vs. rho_normalized, color='dodgerblue',
     lw=2.5, Label r'$\beta$-Profil mit $\beta = 2/3$'.
- Styling & Achsen:
  * xlabel: r'Normierter Radius $x_c = r / r_c$', fontsize=12
  * ylabel: r'Normierte Dichte $\rho_{\text{gas}}(r) /
     \rho_0$', fontsize=12
  * title: r'Das $\beta$-Profil', fontsize=14
  * Grid: plt.grid(True, which="both", ls="--")
  * Legende: plt.legend(fontsize=11)
  * Layout: plt.tight_layout() und plt.show()
Bibliotheken: 'numpy as np', 'matplotlib.pyplot as plt'.
```

- Abb. 3.6: Gravitationskraft vs. Druckkraft für ein isotherm angenommenes Gas

```
Erstelle einen Python-Code zum hydrostatischen Gleichgewicht
   im einheitenlosen NFW- & Beta-Profil zum Vergleich
   zwischen Gravitations- und Druckkraft:
```

```
1. Parameter & mathematische Funktionen:
   - Beta-Parameter: beta = 2/3 #(Literatur: Chan 2021)
   - Kosmischer Gasanteil: f_gas = 0.16
   - Core-Radius (in Einheiten von r_s): x_c = 1.0 /
     np.sqrt(3.0)
   - Konsole: Print-Ausgaben fuer "--- REIN MATHEMATHISCHE
     PARAMETER ---", beta und x_c.
   - NFW DM-Dichte (einheitenlos): nfw_density_pure(x) = 1.0
     / (x * (1.0 + x)**2)
   - NFW Eingeschlossene Masse (einheitenlos):
     nfw_mass_pure(x) = np.log(1.0 + x) - (x / (1.0 + x))
   - Beta-Gasdichte (einheitenlos): gas_density_pure(x) =
     f_gas * (1.0 + (x / x_c)**2)**(-1.5 * beta)

2. Kraftberechnungen & Gitter:
   - Radiales Gitter x = r/r_s: x_array = np.logspace(-2, 3,
     200)
   - Berechne rho_dm_tilde, rho_gas_tilde und M_dm_tilde.
   - Schwerkraft-Term (pro Volumen): F_grav_tilde = (4 *
     np.pi * M_dm_tilde * rho_gas_tilde) / x_array**2
   - Gradient der Gasdichte: drho_gas_dx via
     np.gradient(rho_gas_tilde, x_array)
   - Dimensionslose Temperatur: T_dimensionless = 1.0 / (3.0
     * beta)
   - Druckkraft (negativer Gradient): F_press_tilde = -
     drho_gas_dx * T_dimensionless

Visualisierungs-Vorgaben (Matplotlib):
- Erstelle eine Figure mit figsize=(14, 5).
- Platziere den Plot als Subplot via plt.subplot(1, 2, 2).
- Plot-Inhalt (Einheitenloser Gleichgewichtstest):
  * Doppel-logarithmische Achsen via plt.loglog.
  * Gravitationskraft: F_grav_tilde, color='red', lw=2, Label
    r'Gravtionskraft $F_{Grav}$'.
  * Druckkraft: F_press_tilde, color='blue', lw=2,
    linestyle='--', Label r'Druckkraft $F_{Druck}$'.
  * Achsen & Styling:
    - xlabel: r'Normierter Radius $x = r / r_s$'
    - ylabel: '$F_{Druck}$ vs. $F_{Grav}$'
```

```
    - title: 'Einheitenloser Gleichgewichtstest'
    - Grid aktiviert (which="both", ls="--")
    - Legende, plt.tight_layout() und plt.show()
Bibliotheken: 'numpy as np', 'matplotlib.pyplot as plt'.
```

- Aufbauend auf dem vorherigen Code - zum Temperaturprofil (Abb 3.7) und Gravitationskraft vs. Druckkraft (Abb. 3.8.)

```
Erstelle einen Python-Code zum Temperaturprofil und
   anschlie end einen Gleichgewichtstest zwischen der
   Gravitations- und der Druckkraft:
1. Parameter & mathematische Funktionen:
   - Parameter: beta = 2/3, f_gas = 0.16, x_c = 1.0 /
      np.sqrt(3.0)
   - NFW Eingeschlossene Masse (einheitenlos):
      nfw_mass_pure(x) = np.log(1.0 + x) - (x / (1.0 + x))
   - Beta-Gasdichte (einheitenlos): gas_density_pure(x) =
      (1.0 + (x / x_c)**2)**(-1.5 * beta)
   - Radiales Gitter (feine Aufl sung): x_array =
      np.logspace(-2, 3, 400)
   - Berechne M_dm = nfw_mass_pure(x_array) und rho_gas =
      gas_density_pure(x_array).

2. Numerische Integration des Temperaturprofils T~(x):
   - Initialisiere `T_profile = np.zeros_like(x_array)`.
   - Iteriere ueber alle Radien i in range(len(x_array)):
     * Schneide die Vektoren ab i bis zum Ende ab
        (x_integration, M_integration, rho_integration).
     * Integrand: integrand = (M_integration *
        rho_integration) / x_integration**2
     * Numerische Integration nach der Simpson-Regel via
        `scipy.integrate.simpson` (falls len > 1, sonst 0.0).
     * Berechne T_profile[i] = (4.0 * np.pi / rho_gas[i]) *
        integral_value

3. Gleichgewichtstest mit variabler Temperatur:
   - Gravitationskraft (pro Volumen): F_grav_tilde = (4.0 *
      np.pi * M_dm * rho_gas) / x_array**2
   - Einheitenloser Gesamtdruck: P_tilde = rho_gas * T_profile
```

```
   - Druckkraft (negativer Gradient): F_press_tilde =
      -np.gradient(P_tilde, x_array)

Visualisierungs-Vorgaben (Matplotlib):
- Erstelle eine Figure mit figsize=(8, 10) und 2 vertikal
   angeordneten Subplots (plt.subplot(2, 1, 1) und
   plt.subplot(2, 1, 2)).
- Subplot 1 (Berechnetes Temperaturprofil):
  * Semilogarithmischer Plot (x-Achse logarithmisch via
     plt.xscale('log')).
  * Linienplot: x_array vs. T_profile, color='darkorange',
     lw=2.5, Label r'Berechnetes Temperaturprofil
     $\tilde{T}(x)$'.
  * Achsen & Styling:
    - xlabel: r'Normierter Radius $x_{s} = r / r_s$'
    - ylabel: r'Normierte Temperatur $\tilde{T}$'
    - title: 'Das notwendige Temperaturprofil f r
       hydrostatisches Gleichgewicht'
    - Grid aktiviert (which="both", ls="--")
    - Legende anzeigen.
- Subplot 2 (Gleichgewichtstest):
  * Doppel-logarithmische Achsen via plt.loglog.
  * Gravitationskraft: F_grav_tilde, color='red', lw=2, Label
     r'Gravtionskraft $F_{Grav}$'.
  * Druckkraft: F_press_tilde, color='blue', lw=2,
     linestyle='--', Label r'Druckkraft $F_{Druck}$'.
  * Achsen & Styling:
    - xlabel: r'Normierter Radius $x_{s} = r / r_s$'
    - ylabel: '$F_{Druck}$ vs. $F_{Grav}$'
    - title: 'Gleichgewichtstest mit variabler Temperatur'
    - Grid aktiviert (which="both", ls="--")
    - Legende anzeigen.
- Layout: plt.tight_layout() und plt.show()
 Bibliotheken: 'numpy as np', 'matplotlib.pyplot as plt',
    'from scipy.integrate import simpson'.
```

- Berechnung der Virialradien $r_{vir}$ (zu Tab. 3.1) und Skalenradien $r_s$ (zu Tab. 3.2):

```
Erstelle einrn Python-Code zur Berechnung von Virial- und
   Skalenradien:
1. Physikalische Konstanten:
   - Kritische Dichte: rho_crit = 9.2e-27 [kg/m^3]
   - Sonnenmasse: M_sun = 1.989e30 [kg]
   - Umrechnung kpc in Meter: kpc_in_m = 3.086e19 [m/kpc]

2. Funktion ‘berechne_radien(massen_liste, c_werte_liste)‘:
   - Eingaben: Listen von Halomassen M (in M_sun) und
      Konzentrationsparametern c.
   - Umrechnung Masse in kg: M_kg = M * M_sun
   - Virialradius in Metern: r_vir_m = ((3 * M_kg) / (800 *
      np.pi * rho_crit))**(1/3)
   - Umrechnung Virialradius in kpc: r_vir_kpc = r_vir_m /
      kpc_in_m
   - Skalenradius in Metern und kpc: rs_m = r_vir_m / c und
      rs_kpc = rs_m / kpc_in_m
   - Rueckgabe: Liste von Dictionaries mit den Schl sseln
      'masse', 'r_vir_m', 'r_vir_kpc', 'rs_m' und 'rs_kpc'.

3. Eingabedaten & Ausfuehrung:
   - Massenliste: massen = [1e10, 1e11, 1e12, 1e13, 1e14,
      1e15]
   - Konzentrationsparameter: c_werte = [10.5, 9.0, 7.8, 6.2,
      4.95, 3.9]
   - Aufruf der Funktion ‘berechne_radien(massen, c_werte)‘.
   - Ausgabe  ber  eine for-Schleife mit formatierter
      String-Ausgabe (Masse in e-Notation, R_vir und r_s in
      Metern (Scientific) und kpc mit 2 Nachkommastellen).
```

- Berechnung der Skalendichten $\rho_s$ (zu Tab. 3.2) und der Referenzwerte (zu Tab. 3.4) für die Temperaturen im eingeklammerten Ausdruck nach Gl. (3.54)

```
Erstelle einen Python-Code zur Berechnung von
   Virialtemperaturen & Skalendichte rho_s:
1. Physikalische Konstanten & Parameter:
   - Gravitationskonstante: G = 6.67430e-11 [m^3 kg^-1 s^-2]
   - Sonnenmasse: M_sun = 1.989e30 [kg]
```

```
   - Kritische Dichte: rho_crit = 9.2e-27 [kg/m^3]
   - Protonenmasse: m_p = 1.6726e-27 [kg]
   - Boltzmann-Konstante: k_B = 1.380649e-23 [J/K]
   - Mittleres Molekulargewicht: mu = 0.6
   - Gas-Massenbruch: f_gas = 0.16

2. Eingabedaten:
   - Massenliste: massen = [1e10, 1e11, 1e12, 1e13, 1e14,
     1e15] (in M_sun)
   - NFW-Konzentrationsparameter: c_werte = [10.5, 9.0, 7.8,
     6.2, 4.95, 3.9]

3. Berechnung & Schleifenlogik:
   - Formatiere einen Tabellenkopf in der Konsole:
     * Header: "Masse [M_sun] | rho_s [kg/m^3] |
       Temperatur-Skala [K]"
     * Trennlinie mit 80 Bindestrichen.
   - Iteriere ueber `zip(massen, c_werte)`:
     * Umrechnung Halomasse M_kg = M_m * M_sun und Gasmasse
       m_gas = f_gas * M_kg.
     * Virialradius: r_vir = ((3.0 * M_kg) / (800.0 * np.pi *
       rho_crit))**(1.0/3.0)
     * Skalenradius: rs = r_vir / c
     * Charakteristische NFW-Dichte rho_s:
       - Hilfsfunktion: g_c = np.log(1 + c) - (c / (1 + c))
       - rho_s = M_kg / (4.0 * np.pi * (rs**3) * g_c)
     * Gasprofil-Parameter:
       - Core-Radius: r_c = rs / np.sqrt(3.0)
       - Verh ltnis ratio = r_vir / r_c und klammer = r_vir
         - r_c * np.arctan(ratio)
       - Zentrale Dichte rho_0 = m_gas / (4.0 * np.pi *
         (r_c**2) * klammer)
     * Temperaturskala T_skala:
       - T_skala = (mu * m_p * G * rho_s * (rs**2)) / k_B
     * Formatierte Ausgabe pro Zeile (Masse in e-Notation,
       rho_s und T_skala in wissenschaftlicher Notation mit
       2 Nachkommastellen).
```

- Berechnung der Kernradien $r_c$ (Tab. 3.3) und der Zentraldichte $\rho_0$ sowie der

Anzahldichte $n_0$ (zu Tab. 3.5)

```
Erstelle ein Python-Code zur Berechnung der Zentraldichte
   rho_0 und Teilchendichte n_0 im Beta-Profil:
1. Physikalische Konstanten & Umrechnungsfaktoren:
   - Kritische Dichte: rho_crit = 9.2e-27 [kg/m^3]
   - Sonnenmasse: M_sun = 1.989e30 [kg]
   - Umrechnung kpc in Meter: 3.086e19 [m/kpc]

2. Hilfsfunktion ‘berechne_rs_in_metern(massen_liste,
   c_werte_liste)‘:
   - Eingabe: Listen von Halomassen (in M_sun) und
      Konzentrationsparametern c.
   - Berechnung pro Eintrag: M_kg = M * M_sun, r_vir = ((3 *
      M_kg) / (800 * np.pi * rho_crit))**(1/3), rs_meter =
      r_vir / c.
   - Rueckgabe: Liste der Skalenradien r_s in Metern.

3. Hauptfunktion
   ‘berechne_rho_0_rc_und_n0_fuer_alle_massen(massen_liste,
   rs_meter_liste, f_gas=0.16, beta=2/3, mu=0.6,
   m_p=1.67e-27)‘:
   - Kernradius: r_c = rs / np.sqrt(3.0)
   - Analytisches Integral (f r beta = 2/3): ratio = r_vir /
      r_c, klammer = r_vir - r_c * np.arctan(ratio)
   - Volumenintegrals-Nenner: nenner = 4.0 * np.pi * (r_c**2)
      * klammer
   - Zentrale Massendichte: rho_0 = (f_gas * M_kg) / nenner
   - Zentrale Teilchendichte: n_0 = rho_0 / (mu * m_p)
   - Rueckgabe: Tupel aus Listen (rc_werte, rho_0_werte,
      n_0_werte).

4. Ausfuehrung & Formatierung:
   - Eingabedaten: massen = [1e10, 1e11, 1e12, 1e13, 1e14,
      1e15], c_werte = [10.5, 9.0, 7.8, 6.2, 4.95, 3.9].
   - Aufruf von ‘berechne_rs_in_metern‘ und
      ‘berechne_rho_0_rc_und_n0_fuer_alle_massen‘.
   - Konsolenausgabe einer formatierten Tabelle:
     * Header: "Masse [M_sun] | r_s [kpc] | r_c [kpc] | rho_0
```

```
        [kg/m^3] | n_0 [1/m^3]"
     * Tabellenzeilen mit Konvertierung von r_s und r_c in
        kpc (r / 3.086e19), formatiert mit 2 Nachkommastellen.
```

- Abb. 3.10: Erstellung der Kühlfunktion nur für das primordiale Gas (Verwendung dann später für die Kühlzeit)

```
Erstelle einen Python-Code zur Kuehlfunktion fuer
   primordiales Gas:
1. Dateneingabe & Transformation:
   - Definiere die Arrays ‘log_T_paper‘ und
      ‘log_Lambda_paper‘ mit den exakten Werten aus der
      Literatur.
   - Umrechnung in lineare physikalische Einheiten:
     * Temperaturgitter: T_grid = 10**log_T_paper [K]
     * Kuehlrate in SI-Einheiten: Lambda_SI =
        10**log_Lambda_paper * 1e-13 [J m^3 s^-1]
   - Erstelle eine Liste fuer Halomassen: massen = [1e10,
      1e11, 1e12, 1e13, 1e14, 1e15]

Visualisierungs-Vorgaben (Matplotlib):
- Figure-Groesse: figsize=(10, 6)
- Plot-Inhalt:
  * Doppel-logarithmische Achsen via plt.loglog(T_grid,
     Lambda_SI, color='black', lw=2, linestyle='-',
     label='K hlfunktion $\Lambda(T)$').
- Styling & Achsen:
  * xlabel: 'Temperatur T [K]'
  * ylabel: r'K hlrate $\Lambda(T)$ [J m$^3$ s$^{-1}$]'
  * title: 'K hlraten verschiedener Halomassen'
  * Grid: plt.grid(True, which="both", ls=":", alpha=0.6)
  * Legende: plt.legend(loc='best')
  * Layout: plt.tight_layout() und plt.show()
Bibliotheken: 'numpy as np', 'matplotlib.pyplot as plt'.
```

- Abb. 3.11: Aufbauend auf dem vorherigen Code - Kühlzeiten für verschiedene Halomassen.

```
Erstelle einen Python-Code zum Kuehlphasendiagramm
   verschiedener Halomassen:
1. Physikalische Konstanten & Basis-Setups:
   - k_B = 1.38e-23, m_p = 1.67e-27, mu = 0.6, yr_to_s =
      3.15e7
   - Radiales Gitter (in Metern): r_array = np.logspace(19,
      23, 400)
   - Stelle sicher, dass die Datenpunkte log_T_paper,
      log_Lambda_paper und das berechnete T_profile im Skript
      definiert sind.

2. Funktion `berechne_halo_profil(M_halo, rho0, rs, T_vir)`:
   - Dichteprofil: rho_r = rho0 * (1 + (r_array / (rs *
      3**-0.5))**2)**(-1)
   - Teilchendichte: n_r = rho_r / (mu * m_p)
   - Temperaturprofil: T_r = T_vir * T_profile
   - Rueckgabe: (T_r, n_r)

3. Halo-Profile berechnen:
   - Berechne Profile fuer 6 Halomassen (1e12, 1e13, 1e14,
      1e15, 1e11, 1e10 M_sun) mit den jeweiligen Werten fuer
      rho0, rs und T_vir.

4. Hintergrundgitter & K hlzeit-Interpolation:
   - Gitter: T_grid = np.logspace(2, 8, 200), n_grid =
      np.logspace(5.5, -1.5, 200)
   - Gitter-Matrix via `np.meshgrid(T_grid, n_grid)` ->
      T_mesh, n_mesh.
   - Interpolation der K hlfunktion: `log_Lambda_mesh =
      np.interp(np.log10(T_mesh), log_T_paper,
      log_Lambda_paper)`
   - K hlrate in SI: Lambda_mesh = (10**log_Lambda_mesh) *
      1e-13
   - Logarithmische K hlzeit in Jahren: `log_t_cool_mesh =
      np.log10((3.375 * k_B * T_mesh) / ((n_mesh / 2.25) *
      Lambda_mesh) / yr_to_s)`

Visualisierungs-Vorgaben (Matplotlib):
```

```
- Figure-Groesse: figsize=(14, 6)
- Konturplot f r K hlzeiten:
  * ‘levels = np.linspace(4, 14, 100)‘
  * ‘plt.contourf(T_mesh, n_mesh, log_t_cool_mesh,
     levels=levels, cmap=’coolwarm’, extend=’both’, alpha=1,
     vmin=4, vmax=14)‘
  * Colorbar f r K hlzeit: ‘cbar = plt.colorbar(contour,
     pad=0.0001)‘ mit Ticks [4..14], Labels $10^{4}$ bis
     $10^{14}$ und Beschriftung ’K hlzeit in Jahren’.
- Trajektorien der Halos:
  * Plotte f r jeden Halo eine gestrichelte
     grau-transparente Linie (‘plt.plot(T_r, n_r,
     color=’grey’, linestyle=’--’, alpha=0.2)‘).
  * Plotte Scatter-Punkte ausged nnt ‘[::10]‘ mit
     ‘c=r_array[::10]/3.08e19‘, ‘cmap=’viridis’‘,
     ‘edgecolor=’white’‘, ‘vmin=0‘, ‘vmax=2000‘, ‘s=70‘.
  * Platziere Textetiketten (‘plt.text‘) f r jede Halomasse
     mit weissem, halbtransparenten Hintergrund (bbox).
- Zweite Colorbar f r den Radius:
  * Platziere ‘plt.colorbar(sc, label=’Radius [kpc]’)‘
     basierend auf dem letzten Scatter-Plot.
- Achsen & Styling:
  * Doppel-logarithmische Achsen (plt.xscale(’log’),
     plt.yscale(’log’)).
  * xlim: (1e4, 1e8)
  * xlabel: r’Temperatur T [K]’, fontsize=12
  * ylabel: r’Anzahldichte n $[1/m^3]$’, fontsize=12
  * title: ’K hlphasendiagramm mit verschiedenen Halomassen’
  * Grid: plt.grid(True, which="both", linestyle=’:’,
     alpha=0.5)
  * Ausf hrung: plt.show()
Bibliotheken: ’numpy as np’, ’matplotlib.pyplot as plt’,
   ’matplotlib.cm as cm’, ’matplotlib.colors as mcolors’.
```

# Anhang B

# Python-Codes

Dieser Anhang listet konkret die für die Abschlussarbeiten verwendeten Python-Codes auf.

## B.1 Python-Codes zu den Abbildungen, Simulationen und Ergebnissen in Kapitel 2

- Abb. 2.1: Zeitlicher Verlauf in Einheit der Kollapszeit $t_c$ von der Expansion bis zum Kollaps einer überdichten Sphäre.

```
import numpy as np
import matplotlib.pyplot as plt

# --- 1. PARAMETER FESTLEGEN ---
# R ist die maximale Amplitude (Skalierungsfaktor f r den
   Radius), tc die Kollapszeit
R = 1.0
tc = 1.0

# Hilfsvariable xi l uft von 0 bis 2*pi (eine volle
   Kollaps-Periode)
# xi = 0 bis pi: Expansion bis zum Turnabout
# xi = pi bis 2*pi: Kollaps bis zum Zentrum (r = 0)
xi = np.linspace(0, 2 * np.pi, 1000)

# --- 2. PARAMETRISCHE GLEICHUNGEN BERECHNEN ---
r = (R / 2.0) * (1.0 - np.cos(xi))
t = (tc / (2.0 * np.pi)) * (xi - np.sin(xi))
```

```
# --- 3. PLOTTEN DER ZYKLODIE ---
plt.figure(figsize=(9, 6))
plt.plot(t, r, label="Ueberdichte Sph re", color="b",
   linewidth=2.5)

# Wichtige Punkte markieren
# Turnabout (xi = pi)
t_turn = (tc / (2.0 * np.pi)) * (np.pi - np.sin(np.pi))
r_turn = (R / 2.0) * (1.0 - np.cos(np.pi))
plt.scatter([t_turn], [r_turn], color="red", zorder=5)
plt.annotate(
    "Turn-around (Maximale Ausdehnung)\n$\\xi = \\pi$, $r =
       R_{max}$",
    xy=(t_turn, r_turn),
    xytext=(t_turn -0.2 , r_turn - 0.3),
    arrowprops=dict(facecolor="black", shrink=0.05,
       width=0.8, headwidth=4),
)

# Kollaps-Endpunkt (xi = 2*pi)
t_collaps = tc
r_collaps = 0.0
plt.annotate(
    "Kollaps bei \n$\\xi = 2\\pi$, $t = t_c$",
    xy=(t_collaps, r_collaps),
    xytext=(t_collaps - 0.3, r_collaps + 0.2),
    arrowprops=dict(facecolor="black", shrink=0.05, width=1,
       headwidth=6),
)

plt.xlabel(r"Zeit in Einheiten der Kollapszeit $t_c$",
   fontsize=12)
plt.ylabel("Radius $r(t)$ in Einheiten von $R_{max}$",
   fontsize=12)
plt.xlim(0, tc * 1.1)
plt.ylim(0, R * 1.2)
plt.grid(True, linestyle=":", alpha=0.6)
plt.legend(loc="upper right")
```

```
# Layout optimieren und anzeigen
plt.tight_layout()
plt.show()
```

- Abb. 2.4: Qualitativer Vergleich zwischen dem Newtonschen Gravitationsgesetz mit und ohne dem softening-Parameter (hier exemplarisch mit $\epsilon = 0.25$).

```
import numpy as np
import matplotlib.pyplot as plt

# ALLGEMEINE PARAMETER (Hier frei anpassen)
epsilon = 0.25
M = 1.0
m = 1.0
G = 1.0

# Radius-Array: Startet exakt bei 0, um den Mittelpunkt zu
   analysieren
r = np.linspace(0.0, max(1.5, epsilon * 10), 2000)

# KRAFT-BERECHNUNGEN (Plummer-Softening)
# Die physikalisch korrekte Formel fuer den Kraftbetrag:
F_softened = (G * M * m * r) / (r**2 + epsilon**2)**(1.5)

# Klassische Newton'sche Kraft zum Vergleich
# (Start bei 0.01, um Division durch Null im Plot zu
   verhindern)
r_newton = np.linspace(0.01, max(1.5, epsilon * 5), 2000)
F_newton = (G * M * m) / (r_newton**2)

# Das exakte Maximum der Kurve liegt bei r = epsilon / sqrt(2)
r_max = epsilon / np.sqrt(2)
# Der maximale Kraftwert (der Deckel) skaliert direkt mit
   1/epsilon^2
F_max = (G * M * m * r_max) / (r_max**2 + epsilon**2)**(1.5)

plt.figure(figsize=(11, 6.5), dpi=100)
```

```
# 1. Newton-Kurve (gestrichelt)
plt.plot(r_newton, F_newton, label='Newton-Kraft ohne
   Softening ($F \\propto r^{-2}$)',
         color='firebrick', linestyle='-', linewidth=2.5)

# 2. Plummer-Softening Kurve
plt.plot(r, F_softened,
         label=f'Newton-Kraft mit Softening ($F \\propto
            \\frac{{r}}{{(r^2 + \\epsilon^2)^{{3/2}}}}$)',
         color='dodgerblue', linewidth=2.5)

# 3. Markierung: Der Kraftdeckel (Das Maximum beim Ann hern)
plt.plot(r_max, F_max, 'go', markersize=5, zorder=5)

# Textbeschriftung fuer das Maximum (Skaliert mit 1/eps^2)
plt.text(r_max * 0.5, F_max * 1.05, f' ($F_{{max}} \\propto
   1/\\epsilon^2$)',
         color='forestgreen', fontweight='bold', fontsize=15)

# Diagramm
plt.xlabel('Abstand $r$', fontsize=14)
plt.ylabel('Gravitationskraft $F(r)$', fontsize=14)

# Achsenskalierung (dynamisch an das Maximum angepasst)
plt.ylim(0, F_max * 2)
plt.xlim(0, max(1.2, epsilon * 5))


plt.grid(True, linestyle=':', alpha=0.6)
plt.legend(fontsize=16, loc='upper right')

plt.tight_layout()
plt.show()
```

- Abb. 2.6: Qualitativer Vergleich der Teilchenverteilung innerhalb einer Sphäre mit $R_{max} = 50$ für exemplarisch $N = 500$ Teilchen mit Farbcodierung: Im linken Bild ist die Teilchenanzahl in jeder Kugelschale konstant ($\rho_{Kugel}$ ist inhomogen) und im rechten Bild nicht konstant ($\rho_{Kugel}$ ist homogen).

```
    import matplotlib.pyplot as plt
import numpy as np
# Reproduzierbarkeit
np.random.seed(42)

# Parameter
N = 500
R_max = 50.0

def generate_sphere_particles(n, r_max, use_cube_root=True):
    phi = np.random.uniform(0, 2 * np.pi, n)
    cos_theta = np.random.uniform(-1, 1, n)
    sin_theta = np.sin(np.arccos(cos_theta))

    u = np.random.uniform(0, 1, n)

    if use_cube_root:
        r = r_max * (u ** (1.0 / 3.0))  # Korrekt: dV
           proportional zu r^2 dr
    else:
        r = r_max * u  # Falsch: Dichte proportional zu 1/r^2

    x = r * sin_theta * np.cos(phi)
    y = r * sin_theta * np.sin(phi)
    z = r * cos_theta

    return x, y, z, r

# Erzeugung der Datens tze
x_wrong, y_wrong, z_wrong, r_wrong =
   generate_sphere_particles(
    N, R_max, use_cube_root=False
)
x_correct, y_correct, z_correct, r_correct =
   generate_sphere_particles(
    N, R_max, use_cube_root=True
)
```

```
# PLOTTING MIT COLORBAR LINKS UND ENGEM ABSTAND
fig = plt.figure(figsize=(12, 6))
# Colormap w h l e n
cmap = plt.cm.plasma

ax1.axis("off")

# --- Plot 1: Falsche Verteilung (R * u) ---
ax1 = fig.add_subplot(121, projection="3d")
sc1 = ax1.scatter(x_wrong, y_wrong, z_wrong,
    c=r_wrong, cmap=cmap, vmin=0, vmax=R_max,
    s=50, edgecolor="k", linewidth=0.5,
    alpha=0.9,
)
ax1.set_title(
    r"Falsch: $r = R \cdot u$" "\n(Starke Verklumpung im
       Zentrum)",
    fontsize=11,
    pad=10,
)
ax1.set_xlim([-R_max, R_max])
ax1.set_ylim([-R_max, R_max])
ax1.set_zlim([-R_max, R_max])

# --- Plot 2: Korrekte Verteilung (R * u^(1/3)) ---

ax2.axis("off")
ax2 = fig.add_subplot(122, projection="3d")
sc2 = ax2.scatter(x_correct, y_correct, z_correct,
    c=r_correct, cmap=cmap, vmin=0, vmax=R_max,
    s=50, edgecolor="k", linewidth=0.5, alpha=0.9,
)
ax2.set_title(
    r"Korrekt: $r = R \cdot u^{1/3}$" "\n(Homogene
       Raumerfuellung)",
    fontsize=11,
    pad=10,
)
ax2.set_xlim([-R_max, R_max])
```

```
ax2.set_ylim([-R_max, R_max])
ax2.set_zlim([-R_max, R_max])

# --- Colorbar ganz LINKS platzieren ---
# Position: [left, bottom, width, height]
cbar_ax = fig.add_axes([0.06, 0.2, 0.02, 0.6])
cbar = fig.colorbar(sc2, cax=cbar_ax)
cbar.set_label(
    r"Abstand $r$ zum Zentrum (von 0 bis $R_{max}$)",
    fontsize=10,
    labelpad=10,
)
cbar_ax.yaxis.set_ticks_position("left")  # Skalenwerte links
   von der Colorbar
cbar_ax.yaxis.set_label_position("left")

plt.subplots_adjust(left=0.09, right=0.98, wspace=-0.05,
   top=0.85, bottom=0.1)
plt.show()
```

- N-Körper-Simulationen zu Abb. 2.7 bis 2.19, durchgeführt auf dem Hochleistungscomputersystem *bwUniCluster*. Hinweis: Der Code ist hier mit $N = 2000$ und dem entsprechenden softening aufgeführt. Die Simulationen mit anderen Teilchenzahlen $N$ basieren analog auf dem Code.

```
import os
import pickle
import sys
import numpy as np

G_konst = 1.0

def compute_accelerations_pot_and_csoft(pos, masses,
   softening):
    """Berechnet die Beschleunigungen, die potenzielle
       Energie UND C_soft."""
    n = len(masses)
    diff = pos[np.newaxis, :, :] - pos[:, np.newaxis, :]
    dist_sq = np.sum(diff**2, axis=-1) + softening**2
```

```
    inv_dist = 1.0 / np.sqrt(dist_sq)
    inv_dist_cube = inv_dist**3

    # Beschleunigungen (einsum)
    acc = np.einsum("j,ijk,ij->ik", masses, diff,
       inv_dist_cube)

    triu_indices = np.triu_indices(n, k=1)
    m_i = masses[triu_indices[0]]
    m_j = masses[triu_indices[1]]
    inv_d_ij = inv_dist[triu_indices]
    inv_dist_cube_ij = inv_dist_cube[triu_indices]

    e_pot = -G_konst * np.sum(m_i * m_j * inv_d_ij)
    c_soft = -G_konst * np.sum(m_i * m_j * (softening**2) *
       inv_dist_cube_ij)

    return acc, e_pot, c_soft

def save_checkpoint(filename, state_dict):
    """Speichert den aktuellen Zustand ueber eine tempor re
       Datei."""
    temp_file = filename + ".tmp"
    with open(temp_file, "wb") as f:
        pickle.dump(state_dict, f)
    os.replace(temp_file, filename)

def load_checkpoint(filename):
    """L dt den gespeicherten Zustand aus der
       Checkpoint-Datei."""
    with open(filename, "rb") as f:
        return pickle.load(f)

def simulate_collapse_asym_3d_adaptive(
    n_particles=2000,
    t_end=13725.0,
    dt_max=0.01,
    run_index=1,
    ordner_name="Sim N2000 ep_max",
```

```
):
    R_max = 50.0
    masses = np.ones(n_particles) * (1.0 / n_particles)
    softening_val = 3.2

    num_bins = 50
    bins = np.linspace(0, R_max, num_bins + 1)

    checkpoint_file = os.path.join(
        ordner_name,
           f"checkpoint_{n_particles}_run_{run_index}.pkl"
    )

    # --- CHECKPOINT PRUEFEN UND LADEN ---
    if os.path.exists(checkpoint_file):
        print(f"--> Checkpoint gefunden! Lade Zustand aus
           '{checkpoint_file}'...")
        state = load_checkpoint(checkpoint_file)

        pos = state["pos"]
        vel = state["vel"]
        t_aktuell = state["t_aktuell"]
        schritt_zaehler = state["schritt_zaehler"]

        radii = state["radii"]
        e_kin_hist = state["e_kin_hist"]
        e_pot_hist = state["e_pot_hist"]
        e_ges_hist = state["e_ges_hist"]
        p_ges_hist = state["p_ges_hist"]
        zeit_hist = state["zeit_hist"]
        pq_hist = state["pq_hist"]
        c_soft_hist = state["c_soft_hist"]
        massen_profil_hist = state["massen_profil_hist"]

        last_valid_R = state["last_valid_R"]

        # Beschleunigung fuer geladene Positionen neu
           berechnen
        acc, e_pot, c_soft =
```

```
            compute_accelerations_pot_and_csoft(
                pos, masses, softening=softening_val
            )
            print(f"--> Fortsetzung bei t = {t_aktuell:.2f} /
               {t_end} (Schritt {schritt_zaehler})")

    else:
        print(f"--> Kein Checkpoint gefunden. Starte neue
           Simulation fuer Run {run_index}...")
        np.random.seed(run_index)

        # Initialisierung
        phi = np.random.uniform(0, 2 * np.pi, n_particles)
        cos_theta = np.random.uniform(-1, 1, n_particles)
        theta = np.arccos(cos_theta)
        r = R_max * (np.random.uniform(0, 1, n_particles) **
           (1.0 / 3.0))

        x = r * np.sin(theta) * np.cos(phi)
        y = r * np.sin(theta) * np.sin(phi)
        z = r * np.cos(theta)
        pos = np.column_stack([x, y, z])
        vel = np.zeros_like(pos)

        pos -= np.mean(pos, axis=0)
        vel -= np.mean(vel, axis=0)

        radii = []
        e_kin_hist, e_pot_hist, e_ges_hist = [], [], []
        p_ges_hist, zeit_hist, pq_hist, c_soft_hist = [], [],
           [], []
        massen_profil_hist = []

        last_valid_R = R_max

        acc, e_pot, c_soft =
           compute_accelerations_pot_and_csoft(
            pos, masses, softening=softening_val)
```

```
        t_aktuell = 0.0
        schritt_zaehler = 0

    save_every = 200
    checkpoint_every = 10000

    while t_aktuell < t_end:
        # --- 1. ADAPTIVES dt BERECHNEN ---
        acc_magnitudes = np.sqrt(np.sum(acc**2, axis=1))
        max_acc = np.max(acc_magnitudes)

        if max_acc < 1e-10:
            dt = dt_max
        else:
            eta = 0.2
            dt = eta * np.sqrt(softening_val / max_acc)

        dt = np.clip(dt, 1e-5, dt_max)

        if t_aktuell + dt > t_end:
            dt = t_end - t_aktuell

        # --- 2. VELOCITY VERLET (KDK) ---
        vel += acc * (dt / 2.0)
        pos += vel * dt

        acc, e_pot, c_soft =
           compute_accelerations_pot_and_csoft(
            pos, masses, softening=softening_val
        )

        vel += acc * (dt / 2.0)
        pos -= np.mean(pos, axis=0)
        vel -= np.mean(vel, axis=0)

        # --- 3. SPEICHERN & DIAGNOSTIK (ALLE save_every
           SCHRITTE) ---
        if schritt_zaehler % save_every == 0:
            v_sq = np.sum(vel**2, axis=1)
```

```
            e_kin = 0.5 * np.sum(masses * v_sq)
            e_ges = e_kin + e_pot

            p_betrag = np.linalg.norm(
                np.sum(masses[:, np.newaxis] * vel, axis=0)
            )

            all_radii = np.sqrt(np.sum(pos**2, axis=1))
            e_kin_einzeln = 0.5 * masses * v_sq

            diff = pos[np.newaxis, :, :] - pos[:, np.newaxis,
               :]
            dist_sq_soft = np.sum(diff**2, axis=-1) +
               softening_val**2
            inv_dist_soft = 1.0 / np.sqrt(dist_sq_soft)
            np.fill_diagonal(inv_dist_soft, 0.0)

            e_pot_einzeln = -G_konst * masses * np.sum(
                masses[:, np.newaxis] * inv_dist_soft, axis=0
            )
            e_ges_einzeln = e_kin_einzeln + e_pot_einzeln

            bound_mask = e_ges_einzeln < 0
            inside_mask = all_radii <= R_max
            valid_radii = all_radii[bound_mask & inside_mask]

            current_R = (
                np.median(valid_radii) if len(valid_radii) >
                   0 else last_valid_R
            )
            last_valid_R = current_R

            e_kin_hist.append(e_kin)
            e_pot_hist.append(e_pot)
            e_ges_hist.append(e_ges)
            p_ges_hist.append(p_betrag)
            radii.append(current_R)
            zeit_hist.append(t_aktuell)
```

```
            pq_aktuell = np.sum(masses[:, np.newaxis] * vel *
                pos)
            pq_hist.append(pq_aktuell)

            counts, _ = np.histogram(all_radii, bins=bins)
            massen_profil_hist.append(counts)
            c_soft_hist.append(c_soft)

        t_aktuell += dt
        schritt_zaehler += 1

        # --- 4. CHECKPOINT INTERVALLM SSIG SPEICHERN ---
        if schritt_zaehler % checkpoint_every == 0:
            state = {"pos": pos, "vel": vel,
                "t_aktuell": t_aktuell,
                "schritt_zaehler": schritt_zaehler,
                "radii": radii,
                "e_kin_hist": e_kin_hist,
                "e_pot_hist": e_pot_hist,
                "e_ges_hist": e_ges_hist,
                "p_ges_hist": p_ges_hist,
                "zeit_hist": zeit_hist,
                "pq_hist": pq_hist,
                "c_soft_hist": c_soft_hist,
                "massen_profil_hist": massen_profil_hist,
                "last_valid_R": last_valid_R,
            }
            save_checkpoint(checkpoint_file, state)
            print(f"[Checkpoint] Run {run_index}: Zustand bei
                t ={t_aktuell:.2f} gesichert.")

    # Wenn die Simulation fertig ist, l schen wir den
       Checkpoint wieder
    if os.path.exists(checkpoint_file):
        os.remove(checkpoint_file)

    return (np.array(radii), np.array(zeit_hist),
        np.array(e_kin_hist), np.array(e_pot_hist),
        np.array(e_ges_hist), np.array(p_ges_hist),
```

```
        np.array(pq_hist), np.array(c_soft_hist),
        bins, np.array(massen_profil_hist),)

if __name__ == "__main__":
    TEILCHENZAHL = 2000
    T_END = 13725

    run_index = 1
    for arg in sys.argv:
        if arg.startswith("--run_id="):
            run_index = int(arg.split("=")[1])

    ordner_name = "Sim N2000 ep_max"
    os.makedirs(ordner_name, exist_ok=True)

    dateiname = os.path.join(
        ordner_name,
           f"kollaps_{TEILCHENZAHL}_run_{run_index}.npz"
    )

    if os.path.exists(dateiname):
        print(f"Durchlauf {run_index} bereits vollst ndig
           vorhanden ({dateiname}). Ueberspringe.")
    else:
        print(f"Starte/Fortsetzung HPC-Durchlauf {run_index}
           fuer N={TEILCHENZAHL}...")

        r, t, e_kin, e_pot, e_ges, p_ges, pq, c_soft, bins,
           m_profil = (
            simulate_collapse_asym_3d_adaptive(
                n_particles=TEILCHENZAHL, t_end=T_END,
                   run_index=run_index,
                   ordner_name=ordner_name,
            )
        )

        np.savez(dateiname, radii=r, t_hist=t,
            e_kin=e_kin, e_pot=e_pot, e_ges=e_ges,
            p_ges=p_ges, pq_hist=pq, c_soft_hist=c_soft,
```

```
            bins=bins, massen_profil_hist=m_profil,)

        print(f"Durchlauf {run_index} erfolgreich
           abgeschlossen und gespeichert unter: {dateiname}")
```

- Zeitliche Entwicklung des Halo-Kollapses (zu Abb. 2.7 bis 2.10.) und zur Virialsumme (zu Abb. 2.11 bis 2.14) – Hinweis: Der Code verwendet exemplarisch den Code zu den gespeicherten Daten für $N = 50$ Teilchen mit dem entsprechenden softening. Für die anderen Teilchenzahlen sind die Codes analog aufgebaut.

```
import os
import matplotlib.pyplot as plt
import matplotlib.ticker as ticker
import numpy as np

# --- 1. PARAMETER UND DATEI-SUCHE ---
R_max = 50.0
t_c_200 = 915  # Passende Kollapszeit

dateien = [
    os.path.join("Sim50", "N50_epmax_neu",
       f"kollaps_50_run_{i}.npz")
    for i in range(1, 101)
]

if len(dateien) == 0:
    raise FileNotFoundError(
        "Keine abgespeicherten Simulationsdaten (.npz)
           gefunden!"
    )

print(f"Gefundene L ufe fuer die Statistik: {len(dateien)}")

# --- 2. GEMEINSAME ZEITACHSE FUER DIE INTERPOLATION ---
t_gemeinsam = np.linspace(0.5, 15, 1000)

interpolierte_radien = []
interpolierte_virial_summe = []
```

```
interpolierte_tafel_endwerte = []
interpolierte_modifiziertes_gleichgewicht = []

# --- 3. DATEN LADEN UND INTERPOLIEREN ---
for datei in dateien:
    data = np.load(datei)
    radii = data["radii"]
    t_hist = data["t_hist"]
    e_kin = data["e_kin"]
    e_pot = data["e_pot"]
    pq_hist = data["pq_hist"]
    c_soft_hist = data["c_soft_hist"]

    # Zeitachse des aktuellen Laufs in Einheiten von t_c
       umrechnen
    t_skala_lauf = t_hist / t_c_200 + 0.5

    # 1. Radius interpolieren
    _interp = np.interp(t_gemeinsam, t_skala_lauf, radii)
    interpolierte_radien.append(_interp)

    # 2. Direkte Summe des Virialtheorems berechnen: 2 *
       E_kin + E_pot
    virial_summe_lauf = (2.0 * e_kin) + e_pot
    virial_summe_interp = np.interp(
        t_gemeinsam, t_skala_lauf, virial_summe_lauf
    )
    interpolierte_virial_summe.append(virial_summe_interp)

    # 3. Dynamischen Tafel-Endwert fuer jeden Zeitschritt
       berechnen
    with np.errstate(divide="ignore", invalid="ignore"):
        tafel_endwert_lauf = pq_hist / t_hist
        tafel_endwert_lauf[t_hist == 0] = 0.0

    tafel_endwert_interp = np.interp(
        t_gemeinsam, t_skala_lauf, tafel_endwert_lauf
    )
    interpolierte_tafel_endwerte.append(tafel_endwert_interp)
```

```
    # 4. Modifiziertes Gleichgewicht berechnen (Tafel-Endwert
       + C_soft)
    mod_gleichgewicht_lauf = tafel_endwert_lauf + c_soft_hist
    mod_gleichgewicht_interp = np.interp(
        t_gemeinsam, t_skala_lauf, mod_gleichgewicht_lauf
    )
    interpolierte_modifiziertes_gleichgewicht.append(
    mod_gleichgewicht_interp)

# Arrays in Matrizen umwandeln (Form: [Anzahl_Laeufe, 1000])
matrix_radien = np.vstack(interpolierte_radien)
matrix_virial_summe = np.vstack(interpolierte_virial_summe)
matrix_tafel = np.vstack(interpolierte_tafel_endwerte)
matrix_mod_gleichgewicht =
   np.vstack(interpolierte_modifiziertes_gleichgewicht)

# --- 4. STATISTIKEN BERECHNEN ---
mittelwert_R = np.mean(matrix_radien, axis=0)
std_abweichung_R = np.std(matrix_radien, axis=0)

mittelwert_VS = np.mean(matrix_virial_summe, axis=0)
std_abweichung_VS = np.std(matrix_virial_summe, axis=0)

mittelwert_Tafel = np.mean(matrix_tafel, axis=0)
mittelwert_ModGleichgewicht =
   np.mean(matrix_mod_gleichgewicht, axis=0)

# --- 5. GRAPHISCHE DARSTELLUNG (PLOT 1: Zeitliche
   Entwicklung Halo-Kollaps) ---
fig1, ax1 = plt.subplots(figsize=(12, 6))

theta_analytisch = np.linspace(0, 30 * np.pi, 2000)
radii_analytisch = (R_max / 2.0) * (1.0 -
   np.cos(theta_analytisch))
t_analytisch = (1.0 / (2.0 * np.pi)) * (
    theta_analytisch - np.sin(theta_analytisch)
)
ax1.plot(t_analytisch, radii_analytisch,
```

```
         label="Symmetrischer Kollaps (Theorie)",
         color="blue", alpha=0.4, linewidth=2,)

for radii_interp in interpolierte_radien:
    ax1.plot(t_gemeinsam, radii_interp, alpha=0.08,
             linestyle="--", color="gray", linewidth=1,)

ax1.plot(t_gemeinsam, mittelwert_R,
         color="darkred", linewidth=2.5,
    label=r"Durchschnitt der Medianradien
       $(R_{\text{med}})$",)

ax1.fill_between(t_gemeinsam, mittelwert_R - std_abweichung_R,
    mittelwert_R + std_abweichung_R, color="red", alpha=0.15,
    label=r"Std. abw. $R_{\text{med}}$",)

ax1.axhline(y=R_max / 2, color="green", linestyle="--",
    linewidth=2, label=r"Virialradius $R_{\text{vir}} =
       R_{\text{max}}/2$",)

ax1.xaxis.set_major_locator(ticker.MultipleLocator(1.0))
ax1.set_xlabel(
    r"Zeit in Einheiten der Kollapszeit ($t \, / \, t_c$)",
       fontsize=12
)
ax1.set_ylabel(r"Radius der Kugel R", fontsize=12)
ax1.set_title("Statistische Auswertung des 3D-Kollapses:
   Medianradien")
ax1.set_xlim(0.5, 15)
ax1.set_ylim(0, R_max * 1.3)
ax1.legend(loc="upper left", ncol=2, fontsize=11)
ax1.grid(True, which="both", linestyle="--", alpha=0.3)
plt.show()

# --- 5b. GRAPHISCHE DARSTELLUNG (PLOT 2: VIRIAL-SUMME) ---
fig2, ax2 = plt.subplots(figsize=(12, 6))

for virial_summe_interp in interpolierte_virial_summe:
    ax2.plot(t_gemeinsam, virial_summe_interp, alpha=0.08,
```

```
            linestyle="--", color="gray", linewidth=1,)

ax2.plot(t_gemeinsam, mittelwert_VS, color="darkgreen",
   linewidth=2.5,
label=r"Simulierte Summe $\langle 2E_{\text{kin}} +
   E_{\text{pot}} \rangle$",)

ax2.fill_between(t_gemeinsam, mittelwert_VS -
   std_abweichung_VS,
                mittelwert_VS + std_abweichung_VS,
                   color="green",
                alpha=0.15, label=r"Streuung ($\sigma$)",)

ax2.plot(t_gemeinsam, mittelwert_Tafel, color="red",
   linestyle="--",
         linewidth=2.0,
         label=r"Klassischer Endwert $\langle \frac{p \cdot
            x}{\tau} \rangle$",)

ax2.plot(t_gemeinsam, mittelwert_ModGleichgewicht,
   color="fuchsia",
         linestyle="--", linewidth=2.5,
label=r"Zus tzlicher Endterm durch $\varepsilon$ $\langle
   \frac{p \cdot x}{\tau} + C_{\text{soft}} \rangle$",)

ax2.axhline(y=0.0, color="black", linestyle=":",
   linewidth=1.2,
            label=r"Ideales Gleichgewicht ($\tau \to
               \infty$)",)

ax2.set_ylim(-0.025, 0.025)
ax2.yaxis.set_major_locator(ticker.MultipleLocator(0.01))

ax2.xaxis.set_major_locator(ticker.MultipleLocator(1.0))
ax2.set_xlabel(
    r"Zeit in Einheiten der Kollapszeit ($t \, / \, t_c$)",
       fontsize=12
)
ax2.set_ylabel(r"Virialsumme", fontsize=12)
```

```
ax2.set_title("Verlauf der Virial-Summe vs. theoretischer
   Endwert")
ax2.set_xlim(0.5, 15.0)
ax2.legend(loc="upper right", fontsize=13)
ax2.grid(True, which="both", linestyle="--", alpha=0.3)
plt.show()

# --- 6. KONSOLEN-AUSGABE ---
print("\n" + "=" * 60)
print("STATISTISCHE KENNDATEN DER VIRIAL-ANALYSE:")
print(f"-> Mittlere simulierte Virial-Summe am Ende:
   {mittelwert_VS[-1]:.4f}")
print(f"-> Theoretischer Tafel-Endwert am Ende:
   {mittelwert_Tafel[-1]:.4f}")
print(f"-> Modifiziertes theoretisches Gleichgewicht:
   {mittelwert_ModGleichgewicht[-1]:.4f}")
print(f"-> Verbleibende numerische Abweichung (Softening):
   {mittelwert_VS[-1] - mittelwert_Tafel[-1]:.4f}")
print(f"-> Code-Genauigkeit (Simulierte Summe - Mod.
   Gleichgewicht): {mittelwert_VS[-1] -
   mittelwert_ModGleichgewicht[-1]:.4f}")
print("=" * 60)
```

- Aufbauend auf dem vorherigen Code: Entfernungsverteilung der Teilchen (zu Abb. 2.16 bis 2.19) – hier wieder mit $N = 50$ als Beispiel

```
import os
import matplotlib.pyplot as plt
import numpy as np

# --- 1. PARAMETER UND DATEI-SUCHE ---
R_max = 50.0
t_c_200 = 915

dateien = [
    os.path.join("Sim50", "N50_epmax_neu",
       f"kollaps_50_run_{i}.npz")
    for i in range(1, 101)
]
```

```
if len(dateien) == 0:
    raise FileNotFoundError(
        "Keine abgespeicherten Simulationsdaten (.npz)
           gefunden!"
    )

print(f"Gefundene L ufe f r die kumulative Auswertung:
   {len(dateien)}")

# --- 2. GEMEINSAME ZEITACHSE & INTERPOLATION ---
t_gemeinsam = np.linspace(0.5, 15, 1000)
interpolierte_kumulativ = []
N_gesamtzahl = 50

for datei in dateien:
    data = np.load(datei)
    t_hist = data["t_hist"]
    massen_profil_hist = data[
        "massen_profil_hist"
    ]  # Form: [Zeitschritte, Bins]
    bins = data["bins"]

    # Zeitachse des aktuellen Laufs in Einheiten von t_c
       umrechnen
    t_skala_lauf = t_hist / t_c_200 + 0.5

    # Kumulative Summe ueber die Bins berechnen
    kumulativ_lauf = np.cumsum(massen_profil_hist, axis=1)

    # Normalisierung auf die feste theoretische
       Start-Gesamtzahl N = 50
    kumulativ_lauf_norm = kumulativ_lauf / N_gesamtzahl

    # Auf die gemeinsame Zeitachse interpolieren (fuer jeden
       Bin einzeln)
    n_bins = kumulativ_lauf_norm.shape[1]
    kumulativ_interp_lauf = np.zeros((len(t_gemeinsam),
       n_bins))
```

```
    for b in range(n_bins):
        kumulativ_interp_lauf[:, b] = np.interp(
            t_gemeinsam, t_skala_lauf, kumulativ_lauf_norm[:,
               b]
        )

    interpolierte_kumulativ.append(kumulativ_interp_lauf)

# Ueber alle L ufe den Ensemble-Median bilden (robust gegen
   Ausrei er)
matrix_kumulativ = np.array(
    interpolierte_kumulativ
)  # Form: [Laeufe, Zeit, Bins]
median_kumulativ = np.median(matrix_kumulativ, axis=0)  #
   Form: [Zeit, Bins]


# --- 3. GRAPHISCHE DARSTELLUNG (WEG B IMPLEMENTIERT) ---
fig, ax = plt.subplots(figsize=(10, 6))

gew hlte_zeiten = [0.5, 1.0, 2.0, 3.0, 5.0, 7.0, 10.0]

for t_ziel in gew hlte_zeiten:
    # Index des passenden Zeitschritts auf der Zeitachse
       finden
    idx = np.argmin(np.abs(t_gemeinsam - t_ziel))

    # --- WEG B: Randpunkte exakt von R = 0 bis R = R_max
       setzen ---
    # 1. X-Werte: Nutzen direkt alle Bin-Grenzen (inklusive
       0.0 und 50.0)
    x_werte = bins

    # 2. Y-Werte: Eine 0.0 ganz am Anfang anfuegen (bei R=0
       sind 0% der Teilchen)
    y_werte = np.pad(median_kumulativ[idx, :], (1, 0),
       mode="constant")
```

```
    # 3. Plotten
    ax.plot(
        x_werte,
        y_werte,
        linewidth=2,
        label=rf"$t/t_c = {t_gemeinsam[idx]:.1f}$",
    )

# Achsenbeschriftungen & Styling
ax.set_xlabel(r"Radius $R$ der Kugel", fontsize=14)
ax.set_ylabel(
    r"Kumulativer Anteil der Teilchen $N(<r) /
       N_{\text{gesamt}}$", fontsize=14
)
ax.set_title("Kumulative Entfernungsverteilung der Teilchen",
   fontsize=16)

# Achsenticks gut lesbar gestalten
ax.tick_params(axis="both", which="major", labelsize=12)

ax.set_xlim(0, R_max)
ax.set_ylim(0, 1.05)
ax.grid(True, linestyle="--", alpha=0.3)
ax.legend(loc="upper left", fontsize=12)  # Position unten
   rechts angepasst

plt.tight_layout()
plt.show()
```

## B.2 Python-Codes zu den Abbildungen und Ergebnissen in Kapitel 3

- Abb. 3.2: NFW-Profil

```
import matplotlib.pyplot as plt
import numpy as np
```

```
# Normierte Koordinate x = r / r_s ueber einen
   logarithmischen Bereich
x = np.logspace(-2, 2, 500)

# Normiertes NFW-Profil: rho / rho_s
rho_normiert = 1.0 / (x * (1.0 + x) ** 2)

# Plot erstellen
plt.figure(figsize=(10, 6))
plt.loglog(x, rho_normiert, color='black', lw=2,
   label=r'NFW-Profil')
plt.axvline(1.0, color='red', linestyle='--',
   label=r'Skalenradius $x_s$ = 1')
plt.xlabel(r' Radius $x_s$ = $r / r_s$')
plt.ylabel(r' Dichte $\rho/\rho_s$')
plt.title('Universelles NFW-Dichteprofil')
plt.legend()
plt.grid(True, which='both', ls='--')
plt.show()
```

- Abb. 3.3: $\beta$-Profil

```
import numpy as np
import matplotlib.pyplot as plt

# Parameter fuer das beta-Profil
beta = 2/3
# Dimensionsloser Radius x_c = r / r_c (logarithmisch von
   10^-2 bis 10^2)
x_c = np.logspace(-2, 2, 500)

# Normiertes Dichteprofil: \rho_gas(x_c) / \rho_0
rho_normalized = (1.0 + x_c**2)**(-1.5 * beta)

# Plot erstellen
plt.figure(figsize=(8, 6))
plt.plot(x_c, rho_normalized, color='dodgerblue', lw=2.5,
   label=r'$\beta$-Profil mit $\beta = 2/3$')
```

```
plt.xscale('log')
plt.yscale('log')
plt.xlabel(r'Normierter Radius $x_c = r / r_c$', fontsize=12)
plt.ylabel(r'Normierte Dichte $\rho_{\text{gas}}(r) /
   \rho_0$', fontsize=12)
plt.title(r'Das $\beta$-Profil', fontsize=14)
plt.grid(True, which="both", ls="--")
plt.legend(fontsize=11)

plt.tight_layout()
plt.show()
```

- Abb. 3.6: Gravitationskraft vs. Druckkraft für ein isotherm angenommenes Gas

```
import numpy as np
import matplotlib.pyplot as plt

beta = 2/3
f_gas = 0.16

# Aus dem Paper von Man Ho Chan (2021): r_s \approx \sqrt{3}
   * r_c
# Da wir Radien in Einheiten von r_s messen (x = r/r_s), ist
   x_s = 1.
# Der einheitenlose Core-Radius x_c ist demnach:
x_c = 1.0 / np.sqrt(3.0)

print("--- REIN MATHEMATHISCHE PARAMETER ---")
print(f"Beta-Parameter:  {beta:.3f}")
print(f"Core-Radius x_c: {x_c:.3f} (in Einheiten von r_s)\n")

def nfw_density_pure(x):
    """ Gl. fuer normierte DM-Dichte: \rho_DM(r) / \rho_s """
    return 1.0 / (x * (1.0 + x)**2)

def nfw_mass_pure(x):
    """ Gl. fuer eingeschlossene Masse normiert auf (4 * pi *
       \rho_s * r_s^3) """
```

```
    return np.log(1.0 + x) - (x / (1.0 + x))

def gas_density_pure(x):
    """ Gl. fuer das Beta-Profil des Gases """
    return f_gas * (1.0 + (x / x_c)**2)**(-1.5 * beta)

# TEST auf hydrostatisches Gleichgewicht
# Radiales Gitter x = r / r_s
x_array = np.logspace(-2, 3, 200)

# Berechne die Kurvenverl ufe
rho_dm_tilde = nfw_density_pure(x_array)
rho_gas_tilde = gas_density_pure(x_array)
M_dm_tilde = nfw_mass_pure(x_array)

# Schwerkraft-Term (pro Volumen): M(x) * \rho_gas(x) / x^2
F_grav_tilde = (4* np.pi * M_dm_tilde * rho_gas_tilde) /
   x_array**2

# Druckkraft-Term (negativer Dichtegradient) bei konstanter
   Temperatur
# np.gradient berechnet d(\rho_gas)/dx auf dem Gitter x_array
drho_gas_dx = np.gradient(rho_gas_tilde, x_array)

# Dimensionslose Temperatur
T_dimensionless = 1.0 / (3.0 * beta)
F_press_tilde = - drho_gas_dx * T_dimensionless

# Plot: Gleichgewichtstest
plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 2)
plt.loglog(x_array, F_grav_tilde, label=r'Gravtionskraft
   $F_{Grav}$', color='red', lw=2)
plt.loglog(x_array, F_press_tilde, label=r'Druckkraft
   $F_{Druck}$', color='blue', lw=2, linestyle='--')
plt.xlabel(r'Normierter Radius $x = r / r_s$')
plt.ylabel('$F_{Druck}$ vs. $F_{Grav}$')
plt.title('Einheitenloser Gleichgewichtstest')
plt.legend()
```

```
plt.grid(True, which="both", ls="--")
plt.tight_layout()
plt.show()
```

- Aufbauend auf dem vorherigen Code - zum Temperaturprofil (Abb 3.7) und Gravitationskraft vs. Druckkraft (Abb. 3.8.)

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import simpson

# 1. PARAMETER UND PROFILE (WIE GEHABT)
beta = 2/3
f_gas = 0.16
x_c = 1.0 / np.sqrt(3.0)

def nfw_mass_pure(x):
    return np.log(1.0 + x) - (x / (1.0 + x))

def gas_density_pure(x):
    return (1.0 + (x / x_c)**2)**(-1.5 * beta)

# Radiales Gitter
x_array = np.logspace(-2, 3, 400)

M_dm = nfw_mass_pure(x_array)
rho_gas = gas_density_pure(x_array)

# 2. Berechnung des Temperaturprofils T~(x)
T_profile = np.zeros_like(x_array)

# Schleife ueber jeden Radius, um das Integral von x bis
   unendlich zu berechnen
for i in range(len(x_array)):
    # Integriere nur vom aktuellen Punkt i bis zum ganz
         ueren   Rand [-1]
    x_integration = x_array[i:]
    M_integration = M_dm[i:]
    rho_integration = rho_gas[i:]
```

```
    # Der Integrand: (M * \rho) / x^2
    integrand = (M_integration * rho_integration) /
       x_integration**2

    # Numerische Integration nach der Simpson-Regel
    if len(x_integration) > 1:
        integral_value = simpson(integrand, x_integration)
    else:
        integral_value = 0.0

    # T~(x) = (4 * pi / \rho_gas(x)) * Integral
    T_profile[i] = (4.0 * np.pi / rho_gas[i]) * integral_value

# 3. Gleichgewichtstest zwischen Druckkraft und
   Gravitationskraft
# Schwerkraft-Term
F_grav_tilde = (4.0 * np.pi * M_dm * rho_gas) / x_array**2

# NEU: Der einheitenlose Gesamtdruck ist jetzt P~ = \rho_gas
   * T~
P_tilde = rho_gas * T_profile

# Die Druckkraft ist der negative Gradient des Drucks nach x:
   -dP~/dx
F_press_tilde = -np.gradient(P_tilde, x_array)

# 4. Ergebnisse
plt.figure(figsize=(8, 10))
# Plot 1: Das neu berechnete einheitenlose Temperaturprofil
plt.subplot(2, 1, 1)
plt.plot(x_array, T_profile, color='darkorange', lw=2.5,
   label=r'Berechnetes Temperaturprofil $\tilde{T}(x)$')
#plt.axvline(1.0, color='red', linestyle='--',
   label=r'Skalenradius $x_{s} = 1$')
#plt.axvline(x_c, color='darkgreen', linestyle=':',
   label=r'Gas Kernradius $x_{c}$')
plt.xscale('log')
plt.xlabel(r'Normierter Radius $x_{s} = r / r_s$')
```

```
plt.ylabel(r'Normierte Temperatur $\tilde{T}$')
plt.title('Das notwendige Temperaturprofil fuer
   hydrostatisches Gleichgewicht')
plt.legend()
plt.grid(True, which="both", ls="--")

# Plot 2: Der neue Gleichgewichtstest (Kurven muessen
   aufeinanderliegen!)
plt.subplot(2, 1, 2)
plt.loglog(x_array, F_grav_tilde, label=r'Gravtionskraft
   $F_{Grav}$', color='red', lw=2)
plt.loglog(x_array, F_press_tilde, label=r'Druckkraft
   $F_{Druck}$', color='blue', lw=2, linestyle='--')
plt.xlabel(r'Normierter Radius $x_{s} = r / r_s$')
plt.ylabel('$F_{Druck}$ vs. $F_{Grav}$')
plt.title('Gleichgewichtstest mit variabler Temperatur')
plt.legend()
plt.grid(True, which="both", ls="--")

plt.tight_layout()
plt.show()
```

- Berechnung der Virialradien $r_{vir}$ (zu Tab. 3.1) und Skalenradien $r_s$ (zu Tab. 3.2):

```
import numpy as np

def berechne_radien(massen_liste, c_werte_liste):
    rho_crit = 9.2e-27    # [kg/m^3]
    M_sun = 1.989e30      # [kg]
    kpc_in_m = 3.086e19   # [m pro kpc]

    ergebnisse = []

    for M, c in zip(massen_liste, c_werte_liste):
        # 1. Masse in kg umrechnen
        M_kg = M * M_sun

        # 2. Virialradius in Metern berechnen
```

```
        r_vir_m = ((3 * M_kg) / (800 * np.pi *
           rho_crit))**(1/3)

        # Virialradius in kpc umrechnen
        r_vir_kpc = r_vir_m / kpc_in_m

        # 3. Skalenradius in Metern und kpc berechnen
        rs_m = r_vir_m / c
        rs_kpc = rs_m / kpc_in_m

        ergebnisse.append({
            'masse': M,
            'r_vir_m': r_vir_m,
            'r_vir_kpc': r_vir_kpc,
            'rs_m': rs_m,
            'rs_kpc': rs_kpc
        })

    return ergebnisse

massen = [1e10, 1e11, 1e12, 1e13, 1e14, 1e15]
c_werte = [10.5, 9.0, 7.8, 6.2, 4.95, 3.9]

# Funktion ausf hren
radien_ergebnisse = berechne_radien(massen, c_werte)

# Formatierte Ausgabe f r R_vir und r_s in m und kpc
for res in radien_ergebnisse:
    print(f"Masse: {res['masse']:.0e} M_sun ---> "
          f"R_vir = {res['r_vir_m']:.2e} m
             ({res['r_vir_kpc']:.2f} kpc) | "
          f"r_s = {res['rs_m']:.2e} m ({res['rs_kpc']:.2f}
             kpc)")
```

- Berechnung der Skalendichten $\rho_s$ (zu Tab. 3.2) und der Referenzwerte (zu Tab. 3.4) für die Temperaturen im eingeklammerten Ausdruck nach Gl. (3.54)

```
import numpy as np
```

```
G = 6.67430e-11          # Gravitationskonstante [m^3 kg^-1
   s^-2]
M_sun = 1.989e30         # Sonnenmasse [kg]
rho_crit = 9.2e-27       # Kritische Dichte [kg/m^3]
m_p = 1.6726e-27         # Protonenmasse [kg]
k_B = 1.380649e-23       # Boltzmann-Konstante [J/K]
mu = 0.6                 # Mittleres Molekulargewicht
f_gas = 0.16             # Gas-Massenbruch

#Halomassen und zugehoerigen NFW-Konzentrationen
massen = [1e10, 1e11, 1e12, 1e13, 1e14, 1e15]
c_werte = [10.5, 9.0, 7.8, 6.2, 4.95, 3.9]

print(f"{'Masse [M_sun]':<12} | {'rho_s [kg/m^3]':<16} |
   {'Temperatur-Skala [K]'} ")
print("-" * 80)

for M_m, c in zip(massen, c_werte):
    M_kg = M_m * M_sun
    m_gas = f_gas * M_kg

    # 1. Virialradius und Skalenradius (in Metern)
    r_vir = ((3.0 * M_kg) / (800.0 * np.pi *
       rho_crit))**(1.0/3.0)
    rs = r_vir / c

    # 2. Charakteristische Dichte des NFW-Halos rho_s
    g_c = np.log(1 + c) - (c / (1 + c))
    rho_s = M_kg / (4.0 * np.pi * (rs**3) * g_c)

    # Optional: r_c und rho_0 f r das Gasprofil bleiben
       erhalten
    r_c = rs / np.sqrt(3.0)
    ratio = r_vir / r_c
    klammer = r_vir - r_c * np.arctan(ratio)
    rho_0 = m_gas / (4.0 * np.pi * (r_c**2) * klammer)

    # 3. Berechnung des Vorfaktor-Klammerausdrucks mit r_s
       und rho_s
```

```
    T_skala = (mu * m_p * G * rho_s * (rs**2)) / k_B

    print(f"{M_m:<12.0e}  | {rho_s:<16.2e} | {T_skala:.2e} K")
```

- Berechnung der Kernradien $r_c$ (zu Tab. 3.3) und der Zentraldichte $\rho_0$ sowie der Anzahldichte $n_0$ (zu Tab. 3.5)

```
import numpy as np

def berechne_rs_in_metern(massen_liste, c_werte_liste):
    rho_crit = 9.2e-27  # [kg/m^3]
    M_sun = 1.989e30    # [kg]

    ergebnisse_meter = []
    for M, c in zip(massen_liste, c_werte_liste):
        M_kg = M * M_sun
        r_vir = ((3 * M_kg) / (800 * np.pi * rho_crit))**(1/3)
        rs_meter = r_vir / c
        ergebnisse_meter.append(rs_meter)

    return ergebnisse_meter

def berechne_rho_0_rc_und_n0_fuer_alle_massen(massen_liste,
   rs_meter_liste, f_gas=0.16, beta=2/3, mu=0.6,
   m_p=1.67e-27):
    """
    Berechnet den Kernradius r_c, die zentrale Gasdichte rho_0
    und die zentrale Teilchendichte n_0.
    """
    rho_crit = 9.2e-27  # [kg/m^3]
    M_sun = 1.989e30    # [kg]

    rc_werte = []
    rho_0_werte = []
    n_0_werte = []

    for M, rs in zip(massen_liste, rs_meter_liste):
        M_kg = M * M_sun
        m_gas = f_gas * M_kg
```

```
        # 1. Virialradius f r die Integrationsgrenze
        r_vir = ((3 * M_kg) / (800 * np.pi * rho_crit))**(1/3)

        # 2. Kernradius nach Vorgabe: r_c = r_s / sqrt(3)
        r_c = rs / np.sqrt(3.0)
        rc_werte.append(r_c)

        # 3. Analytischer Klammerausdruck f r beta = 2/3 im
           Integral
        ratio = r_vir / r_c
        klammer = r_vir - r_c * np.arctan(ratio)

        # 4. Nenner des Volumenintegrals
        nenner = 4.0 * np.pi * (r_c**2) * klammer

        # 5. Zentrale Massendichte rho_0
        rho_0 = m_gas / nenner
        rho_0_werte.append(rho_0)

        # 6. Zentrale Teilchendichte n_0 = rho_0 / (mu * m_p)
        n_0 = rho_0 / (mu * m_p)
        n_0_werte.append(n_0)

    return rc_werte, rho_0_werte, n_0_werte

# --- Anwendung ---
massen = [1e10, 1e11, 1e12, 1e13, 1e14, 1e15]
c_werte = [10.5, 9.0, 7.8, 6.2, 4.95, 3.9]

# r_s Werte in Metern berechnen
rs_werte_meter = berechne_rs_in_metern(massen, c_werte)

# r_c, rho_0 und n_0 berechnen
rc_werte_meter, rho_0_liste, n_0_liste =
   berechne_rho_0_rc_und_n0_fuer_alle_massen(massen,
   rs_werte_meter)

# Formatierte Ausgabe (in kpc, kg/m^3 und 1/m^3)
```

```
print(f"{'Masse [M_sun]':<12} | {'r_s [kpc]':<10} | {'r_c
   [kpc]':<10} | {'rho_0 [kg/m^3]':<15} | {'n_0 [1/m^3]'}")
print("-" * 75)

for i in range(len(massen)):
    rs_kpc = rs_werte_meter[i] / 3.086e19
    rc_kpc = rc_werte_meter[i] / 3.086e19
    print(f"{massen[i]:<12.0e} | {rs_kpc:<10.2f} |
       {rc_kpc:<10.2f} | {rho_0_liste[i]:<15.2e} |
       {n_0_liste[i]:.2e}")
```

- Abb. 3.10: Erstellung der Kühlfunktion nur für das primordiale Gas (Verwendung dann später für die Kühlzeit)

```
import numpy as np
import matplotlib.pyplot as plt

# 0. Daten
log_T_paper = np.array([4.075, 4.122, 4.165, 4.209, 4.257,
   4.299, 4.347, 4.387, 4.434, 4.490, 4.535, 4.586, 4.640,
   4.693, 4.747, 4.801, 4.852, 4.903, 4.954, 5.007, 5.061,
   5.114, 5.168, 5.222, 5.276, 5.329, 5.383, 5.437, 5.490,
   5.544, 5.598, 5.652, 5.705, 5.759, 5.813, 5.866, 5.920,
   5.974, 6.027, 6.081, 6.134, 6.188, 6.242, 6.295, 6.349,
   6.402, 6.456, 6.509, 6.562, 6.616, 6.669, 6.722, 6.775,
   6.828, 6.881, 6.933, 6.986, 7.038, 7.091, 7.142, 7.194,
   7.245, 7.296, 7.346, 7.395, 7.444, 7.500])

log_Lambda_paper = np.array([-23.67, -23.27, -23.03, -22.84,
   -22.68, -22.58, -22.51, -22.48, -22.50, -22.57, -22.66,
   -22.75, -22.82, -22.86, -22.85, -22.79, -22.70, -22.62,
   -22.56, -22.55, -22.59, -22.66, -22.75, -22.84, -22.92,
   -22.98, -23.05, -23.10, -23.15, -23.19, -23.23, -23.26,
   -23.28, -23.30, -23.31, -23.32, -23.33, -23.34, -23.34,
   -23.34, -23.33, -23.33, -23.32, -23.32, -23.32, -23.31,
   -23.29, -23.28, -23.27, -23.25, -23.24, -23.22, -23.20,
   -23.18, -23.16, -23.15, -23.13, -23.11, -23.09, -23.07,
   -23.05, -23.03, -23.01, -22.99, -22.97, -22.95, -22.92])
```

```
# 1. Vorbereitung der Daten
massen = [1e10, 1e11, 1e12, 1e13, 1e14, 1e15]

# 2. Plot erstellen
plt.figure(figsize=(10, 6))

# Die durchgezogene Kuehlkurve
T_grid = 10**log_T_paper
Lambda_SI = 10**log_Lambda_paper * 1e-13
plt.loglog(T_grid, Lambda_SI, color='black', lw=2,
   linestyle='-', label='Kuehlfunktion $\Lambda(T)$')

# Plot
plt.xlabel('Temperatur T [K]')
plt.ylabel(r'Kuehlrate $\Lambda(T)$ [J m$^3$ s$^{-1}$]')
plt.title('Kuehlraten verschiedener Halomassen')
plt.grid(True, which="both", ls=":", alpha=0.6)
plt.legend(loc='best') # Legende zeigt jetzt alle Massen
   farbig an
plt.tight_layout()
plt.show()
```

- Abb. 3.11: Aufbauend auf dem vorherigen Code - Kühlzeiten für verschiedene Halomassen.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import matplotlib.colors as mcolors

# --- 1. Konstanten & Vorbereitung ---
k_B = 1.38e-23
m_p = 1.67e-27
mu = 0.6
yr_to_s = 3.15e7
r_array = np.logspace(19, 23, 400) # Bereich in Metern

def berechne_halo_profil(M_halo, rho0, rs, T_vir):
    rho_r = rho0 * (1 + (r_array / (rs * 3**-0.5))**2)**(-1)
```

```
    n_r = rho_r / (mu * m_p)
# T_r berechnen:
    T_r = T_vir * T_profile
    return T_r, n_r

# Halo 1 (z.B. 10^12 M_sun)
T_r1, n_r1 = berechne_halo_profil(1e12, 2.01e-23, 4.71e+20,
   7.30e5)

# Halo 2 (z.B. 10^13 M_sun)
T_r2, n_r2 = berechne_halo_profil(1e13, 1.31e-23, 1.28e+21,
   3.12e6)

# Halo 3 (z.B. 10^14 M_sun)
T_r3, n_r3 = berechne_halo_profil(1e14, 8.69e-24, 3.45e+21,
   1.35e7)

# Halo 4 (z.B. 10^15 M_sun)
T_r4, n_r4 = berechne_halo_profil(1e15, 5.67e-24, 9.43e+21,
   5.93e7)

# Halo 5 (z.B. 10^11 M_sun)
T_r5, n_r5 = berechne_halo_profil(1e11, 2.64e-23, 1.90e+20,
   1.67e5)

# Halo 6 (z.B. 10^10 M_sun)
T_r6, n_r6 = berechne_halo_profil(1e10, 3.54e-23, 7.54e+19,
   3.84e4)

# 2. Hintergrund-Gitter (Temperatur vs Dichte)
T_grid = np.logspace(2, 8, 200)   # 10^4 bis 10^8 K
n_grid = np.logspace(5.5, -1.5, 200) # 10^-4 bis 10^6 1/m^3
T_mesh, n_mesh = np.meshgrid(T_grid, n_grid)

# Kuehlzeit fuer das Hintergrundgitter berechnen (fuer den
   Kontur-Plot)
# Lambda-Interpolation
log_Lambda_mesh = np.interp(np.log10(T_mesh), log_T_paper,
   log_Lambda_paper)
```

```
Lambda_mesh = (10**log_Lambda_mesh) * 1e-13
n_H_mesh = n_grid / 2.25
#t_cool_mesh = (3.375 * k_B * T_mesh) / (n_H_mesh *
   Lambda_mesh)
# Stelle sicher, dass wir log10 verwenden, um die Skalierung
   zu beherrschen
log_t_cool_mesh = np.log10((3.375 * k_B * T_mesh) / ((n_mesh
   / 2.25) * Lambda_mesh) / yr_to_s)

#levels = np.linspace(log_t_cool_mesh.min(),
   log_t_cool_mesh.max(), 100)
levels = np.linspace(4, 14, 100)

plt.figure(figsize=(14, 6))
contour = plt.contourf(T_mesh, n_mesh, log_t_cool_mesh,
   levels=levels, cmap='coolwarm', extend='both', alpha=1,
   vmin=4, vmax=14)
cbar = plt.colorbar(contour, pad=0.0001)

# 2. Definiere die Werte (Exponenten), an denen die Striche
   sein sollen
ticks = [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]

# 3. Weise die Ticks zu
cbar.set_ticks(ticks)

# 4. Weise die Beschriftungen zu
cbar.set_ticklabels([r'$10^{4}$', r'$10^{5}$', r'$10^{6}$',
   r'$10^{7}$', r'$10^{8}$', r'$10^{9}$',
   r'$10^{10}$',r'$10^{11}$',r'$10^{12}$',
r'$10^{13}$',r'$10^{14}$'])

# 5. Label setzen
cbar.set_label('Kuehlzeit in Jahren')

# Halo 1 plotten
plt.plot(T_r1, n_r1, color='grey', linestyle='--', alpha=0.2,
   label='10^12 M_sun')
plt.scatter(T_r1[::10], n_r1[::10], c=r_array[::10]/3.08e19,
```

```
    cmap='viridis', edgecolor='white', vmin=0, vmax=2000,
    s=70, zorder=1)
plt.text(1.4*T_r1[200], 20*n_r1[200], r'$10^{12} M_\odot$',
    fontsize=9, color='black', bbox=dict(facecolor='white',
    alpha=0.5, edgecolor='none', pad=1), fontweight='bold',
    rotation=0, ha='left', va='bottom')

# Halo 2 plotten
plt.plot(T_r2, n_r2, color='grey', linestyle='--', alpha=0.2,
    label='10^13 M_sun')
plt.scatter(T_r2[::10], n_r2[::10], c=r_array[::10]/3.08e19,
    cmap='viridis', edgecolor='white', vmin=0, vmax=2000,
    s=70, zorder=2)
plt.text(1.2*T_r2[200], 5*n_r2[200], r'$10^{13} M_\odot$',
    fontsize=9, color='black', bbox=dict(facecolor='white',
    alpha=0.5, edgecolor='none', pad=1), fontweight='bold',
    rotation=0, ha='left', va='bottom')

# Halo 3 plotten
plt.plot(T_r3, n_r3, color='grey', linestyle='--', alpha=0.2,
    label='10^14 M_sun')
plt.scatter(T_r3[::10], n_r3[::10], c=r_array[::10]/3.08e19,
    cmap='viridis', edgecolor='white', vmin=0, vmax=2000,
    s=70, zorder=2)
plt.text(1.2*T_r3[200], 1.8*n_r3[200], r'$10^{14} M_\odot$',
    fontsize=9, color='black', bbox=dict(facecolor='white',
    alpha=0.5, edgecolor='none', pad=1), fontweight='bold',
    rotation=0, ha='left', va='bottom')

# Halo 4 plotten
plt.plot(T_r4, n_r4, color='grey', linestyle='--', alpha=0.2,
    label='10^15 M_sun')
plt.scatter(T_r4[::10], n_r4[::10], c=r_array[::10]/3.08e19,
    cmap='viridis', edgecolor='white', vmin=0, vmax=2000,
    s=70, zorder=2)
plt.text(0.7*T_r4[200], 1.5*n_r4[200], r'$10^{15} M_\odot$',
    fontsize=9, color='black', bbox=dict(facecolor='white',
    alpha=0.5, edgecolor='none', pad=1), fontweight='bold',
    rotation=0, ha='left', va='bottom')
```

```
# Halo 5 plotten
plt.plot(T_r5, n_r5, color='grey', linestyle='--', alpha=0.2,
   label='10^11 M_sun')
plt.scatter(T_r5[::10], n_r5[::10], c=r_array[::10]/3.08e19,
   cmap='viridis', edgecolor='white', vmin=0, vmax=2000,
   s=70, zorder=2)
plt.text(1.5*T_r5[200], 120*n_r5[200], r'$10^{11} M_\odot$',
   fontsize=9, color='black', bbox=dict(facecolor='white',
   alpha=0.5, edgecolor='none', pad=1), fontweight='bold',
   rotation=0, ha='left', va='bottom')

# Halo 6 plotten
plt.plot(T_r6, n_r6, color='grey', linestyle='--', alpha=0.2,
   label='10^10 M_sun')
sc=plt.scatter(T_r6[::10], n_r6[::10],
   c=r_array[::10]/3.08e19, cmap='viridis',
   edgecolor='white', vmin=0, vmax=2000, s=70, zorder=2)
plt.text(1.7*T_r6[200], 10*n_r6[100], r'$10^{10} M_\odot$',
   fontsize=9, color='black', bbox=dict(facecolor='white',
   alpha=0.5, edgecolor='none', pad=1), fontweight='bold',
   rotation=0, ha='left', va='bottom')

# Radius-Punkte auf der Linie (um den Radius sichtbar zu
   machen)
#sc = plt.scatter(T_r[::10], n_r[::10],
   c=r_array[::10]/3.08e19, vmin=0, vmax=1800,
   cmap='viridis', s=70, edgecolor='white', zorder=2)
plt.colorbar(sc, label='Radius [kpc]')

plt.xscale('log')
plt.yscale('log')
plt.xlabel(r'Temperatur T [K]', fontsize=12)
plt.ylabel(r'Anzahldichte n $[1/m^3]$', fontsize=12)
plt.title('Kuehlphasendiagramm mit verschiedenen Halomassen')
plt.grid(True, which="both", linestyle=':', alpha=0.5)
plt.xlim(1e4, 1e8)
plt.show()
```

# Anhang C

# Eigenständigkeitserklärung

## Erklärung über eigenständige Leistungen und Nutzung KI-basierter Hilfsmittel bei der Masterarbeit im Master of Education, Profillinie Lehramt Gymnasium

Hiermit versichere ich, dass ich die vorliegende Arbeit mit dem Titel

Einfache Modelle zur Strukturbildung im Universum

_____

_____

_____

1. selbständig angefertigt habe,
2. keine anderen als die angegebenen Quellen und Hilfsmittel (einschließlich KI-basierter) benutzt habe,
3. sämtliche wörtlichen oder sinngemäß übernommenen Textstellen als solche kenntlich gemacht habe.

Ich versichere, dass die ggf. eingereichte Papierversion mit dem Dokument im elektronischen Format inhaltlich identisch ist.

**Angaben zu verwendeten KI-basierten Hilfsmitteln (bitte Entsprechendes ankreuzen)**

☐ Ich habe keine KI-basierten Hilfsmittel verwendet

**oder**

☑ Die von mir verwendeten KI-basierten Hilfsmittel sind in der **Tabelle auf Seite 2** dieser Erklärung dokumentiert. Der Gebrauch dieser Hilfsmittel inklusive Art, Ziel und Umfang des Gebrauchs habe ich **vorab mit meiner:m Erstbetreuer:in abgesprochen**. Die besprochenen Regelungen wurden eingehalten. Mir ist bewusst, dass insbesondere der Versuch einer nicht dokumentierten Nutzung KI-basierter Hilfsmittel als **Täuschungsversuch** entsprechend § 8 im Allgemeinen Teil der Prüfungsordnung für den Master of Education, Profillinie Lehramt Gymnasium zu werten ist: *„Versucht die zu prüfende Person, das Ergebnis der Prüfungsleistung durch Täuschung oder Benutzung nicht zugelassener Hilfsmittel zu beeinflussen, wird die betreffende Prüfungsleistung mit „nicht ausreichend“ (5,0) bzw. mit „nicht bestanden“ bewertet.“*

Heidelberg, den 01.10.2026 _____ Martin Reitenbach _____

Unterschrift des:r Studierenden

## Übersicht verwendeter KI-Hilfsmittel

Die Tabelle dient dazu, den Einsatz von KI-Tools in Ihrer Abschlussarbeit transparent zu machen. Bitte tragen Sie alle KI-Tools ein, die Sie im Rahmen Ihrer Arbeit verwendet haben und füllen Sie die Tabelle wie folgt aus:

1. **KI-Werkzeug**: Geben Sie den Namen des verwendeten KI-Tools an (z. B. ChatGPT, DeepL).
2. **Verwendungszwecke**: Beschreiben Sie, wie Sie das Tool eingesetzt haben (z. B. Erstellung eines Gliederungsentwurfs, Übersetzung ins Englische).
3. **Kommentare (optional)**: Nennen Sie die Kapitel und Abschnitte, in denen Sie KI-Tools verwendet haben. Wenn nötig, ergänzen Sie zusätzliche Informationen (z.B. „Korrektur durch Muttersprachler:in“, „als Inspiration verwendet“, „erste Strukturierung der Inhalte“).

**Wichtig**: Es müssen **alle verwendeten KI-Tools** angegeben werden, unabhängig davon, wie umfangreich oder gering ihr Einsatz war. Dies stellt sicher, dass die Nutzung von KI-Werkzeugen vollständig und im Sinne der guten wissenschaftlichen Praxis dokumentiert wird.

**Beispieltabelle:**

| KI-Werkzeug | Verwendungszwecke | Kommentare (optional) |
|---|---|---|
| ChatGPT | Erstellung eines Gliederungsentwurfs | Erste Strukturierung der Inhalte |
| DeepL | Übersetzung ins Englische | Kontrolle durch Muttersprachler:in |

**Ihre Tabelle:**

| KI-Werkzeug | Verwendungszwecke | Kommentare (optional) |
|---|---|---|
| Google's Gemini | Erstellung von Python-Codes zu den für die MA durchgeführten Simulationen, Ergebnissen und Abbildungen | Die genauen Prompts für die Codes befinden sich im Anhang A der Masterarbeit |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |