universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Modelling Primordial Black Holes as a Collisional Dark Matter Fluid

verfasst von / submitted by

Fridolin Glatter, BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2021 / Vienna, 2021

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 861
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Astronomie
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Phys. Dr. sc. ETH Oliver Hahn

Contents

1. Introduction

With this thesis, we want to contribute to the question of 'What if?'. In particular, we are asking what structure formation in the universe would look like if all of the dark matter is constituted by black holes formed within seconds after the Big Bang. Owing to their era of creation, these black holes are also called Primordial Black Holes (PBHs). Apart from assuming this form of dark matter, we are not changing anything about the standard ΛCDM model with its most current values as measured by Planck Collaboration et al. 2020, most notably the circa 26.4% contribution of Cold Dark Matter (CDM) to the overall energy budget of the universe. The reason for that is the overwhelming success with which this model describes current observations. Since this work focuses on dark matter, we briefly recount a non-exhaustive list of evidence here.

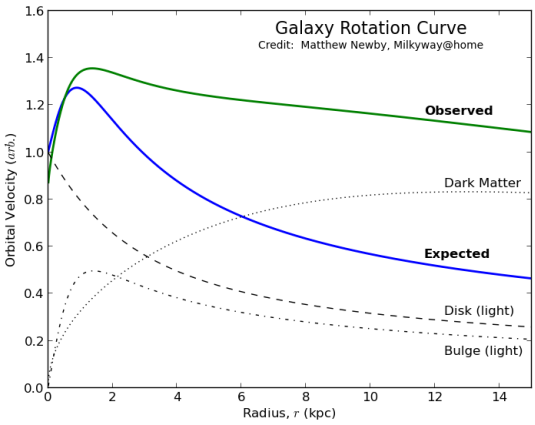
In addition to the Planck collaboration measurement from the Cosmic Microwave Background (CMB) power spectrum (ibid.), there is evidence for the existence of dark matter from the rotation curves of galaxies (Rubin and Ford 1970 any many others thereafter). Without dark matter, we would expect the rotational velocity of galaxies to decrease with radius more sharply than we observe (see figure 1.1a). This discrepancy requires some unknown ('dark') matter to be understood and was the reason that this dark matter was postulated in the first place. The term 'dark' also addresses another crucial aspect of this form of matter: it does not interact with radiation. Moreover, dark matter is usually assumed to interact with baryonic matter and itself only via gravity, which is why it is called 'collisionless'.

There is more evidence for the existence of dark matter based on large scale structure. If the only gravitationally attractive particles in the universe were baryons, structure formation would take a lot longer than it did. We can observe objects at such high redshift that they could only have formed fast enough with some additional gravity. As a final example, we can measure the mass of an object by the strength of the gravitational lensing effect it causes (for an observational example of this lensing, see figure 1.1b). By simultaneously estimating the mass of a galaxy in another way (e.g. from the rotation curve), we can conclude that dark matter is again necessary to explain the observed amount of lensing.

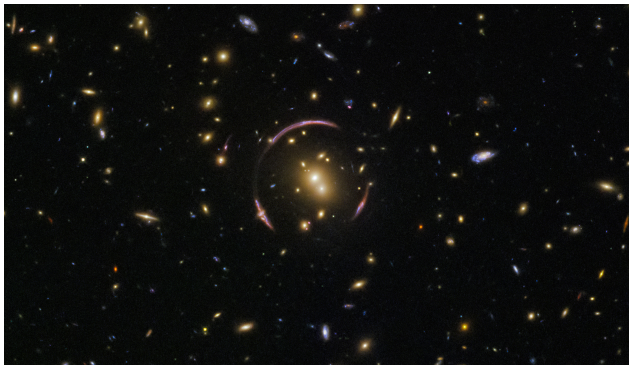
The microscopic nature of dark matter is still heavily debated and it is beyond the scope of this thesis to try and answer that question. Instead, we are simply going to assume a ΛPBH cosmology, i.e. with PBHs being the CDM, and want to know what structure formation looks like. Having an abundance of PBHs in the universe implies that there should be collisional interaction between dark matter in addition to collisionless gravitational attraction. Through passing very close by each other, the black holes should experience additional acceleration caused by elastic scattering. These scattering interactions effectively make PBHs a form of self-interacting dark matter (SIDM). To study their effect on structure formation, we are going to use GAlaxies with Dark matter and Gas intEracT, version four (`GADGET-4`), a cosmological code for N-body and Smoothed-Particle Hydrodynamics (SPH) simulations written by Springel, Pakmor et al. 2020 and freely available online[1]. Their SPH treatment resembles closely the way with which we want to describe the PBHs, namely as an effective fluid. This allows us to quickly start testing our implementation as we can begin to insert just the parts we are interested in into the already existing routines.

The outline of this thesis is as follows: chapter 2 begins by presenting the motivation for this project as well as some observational evidence for the existence of PBHs. It then introduces the mass function for the black holes and the theoretical description we use for the scattering interactions. In chapter 3, we then explain our changes to the source code of `GADGET-4`. Additionally, we showcase some plane wave tests performed with our algorithm and note several caveats. Next, chapter 4 describes our cosmological simulations and their results, before chapter 5 gives a summary of and outlook from this project.

[1] `https://wwwmpa.mpa-garching.mpg.de/gadget4/01_index/`



(a) Generic rotation curve of a galaxy. The expected curve is blue; the observed one green. The black curves show contributions from different parts of a galaxy.



(b) Visual example for strong gravitational lensing. Image credit: ESA/Hubble & NASA; acknowledgement: Judy Schmidt.

Figure 1.1.: Exemplary evidence for the existence of dark matter.

2. Primordial Black Holes

2.1. Motivation

The concept of PBHs was initially introduced by Stephen Hawking in 1971 (Hawking 1971). At first, his idea was only picked up by a few other authors, but due to the gravitational microlensing surveys starting in the 1990s (MACHO[2], EROS[3], OGLE[4]), the idea gained attention. Since these surveys seemed to mainly impose limits for the contribution of MAssive Compact Halo Objects (MACHOs) to the dark matter, interest was decreasing slightly, before it outright exploded after the detection of gravitational waves by LIGO[5] and Virgo[6] in 2015 (see figure 2.1).
The reason for this spiked interest is the mass range of the progenitor objects releasing gravitational waves as they slowly merge with each other, which is depicted in figure 2.2a. There is also an interactive version of this plot available online[7], which is being kept up to date. It shows data from all gravitational wave events detected by LIGO-Virgo, which are colour-coded to represent different types of objects. Depending on their detection method, which can either be through gravitational waves or electromagnetic signatures, there are neutron stars visualised in orange and yellow, and black holes depicted in blue and purple, respectively. The masses are indicated both by the y-axis and the size of the circles. All black holes detected by LIGO-Virgo fall into a mass range higher than that of the previously measured black holes. While this might just be due to the fact that more massive objects produce stronger gravitational waves and are therefore easier to detect, it might also indicate that there are much more of these massive black holes than previously thought. This is especially supported by the interactive version as there are more events in the lower mass range, yet even more in the higher one. A stellar origin of black holes in that mass range is not very likely since this would imply extremely massive progenitors. Similarly, there is the so-called mass gap between two and five solar masses, with the lower limit being the maximum mass of a neutron star (Özel and Freire 2016) and the upper limit being the minimum mass of black holes as derived from X-ray observations (Farr et al. 2011).

[2] http://wwwmacho.anu.edu.au/
[3] http://eros.in2p3.fr/
[4] http://ogle.astrouw.edu.pl/
[5] https://ligo.org/
[6] http://public.virgo-gw.eu/language/en/
[7] https://media.ligo.northwestern.edu/gallery/mass-plot

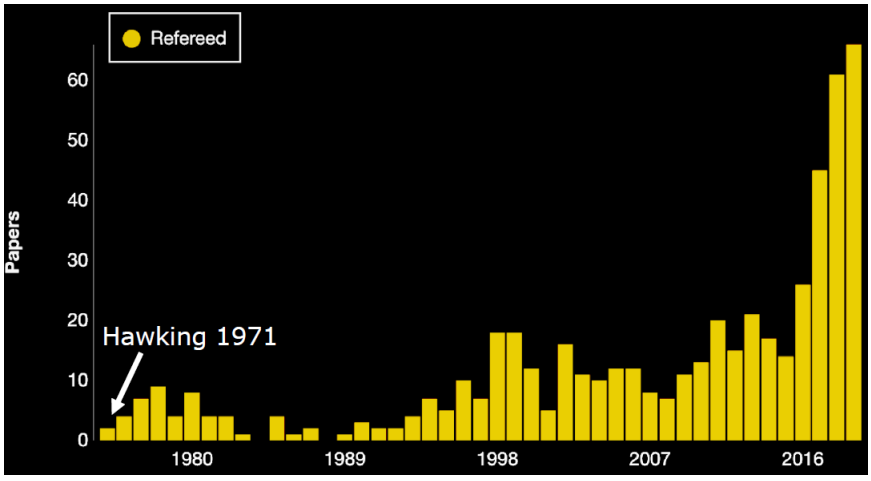
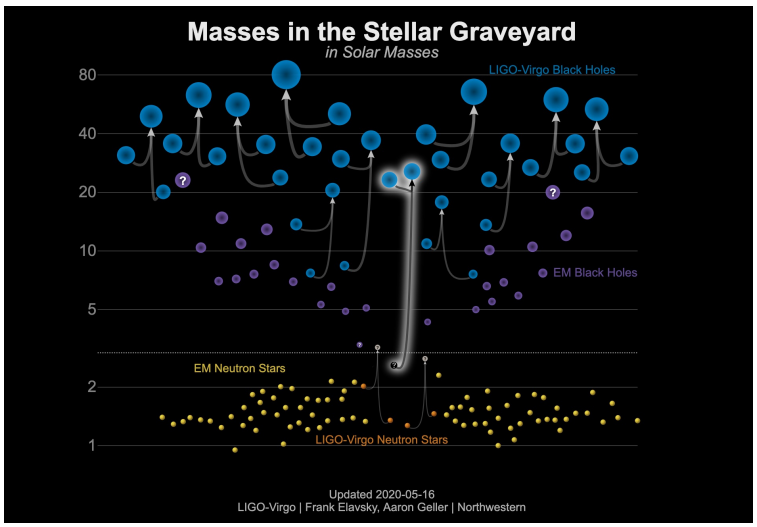
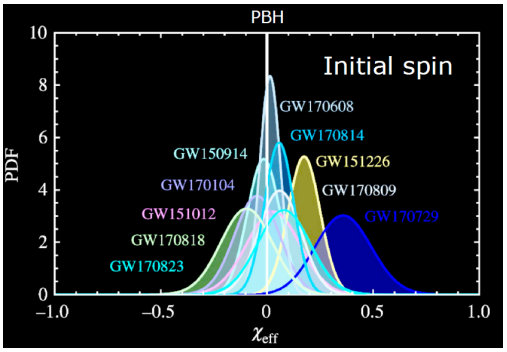


Figure 2.1.: Number of papers regarding PBHs throughout the last decades. Figure from Hasinger 2020.



(a) Neutron star and black hole masses detected by LIGO-Virgo compared to other detection methods. See text for more details. Image from LIGO-Virgo/Frank Elavsky & Aaron Geller (Northwestern).



(b) Spin distribution of black holes detected by LIGO-Virgo. Only a combined spin can be measured for now. See text for more details. Image adapted from García-Bellido 2019.

Figure 2.2.: Exemplary evidence for the existence of PBHs.

Thus, detecting black holes there would be further indication of another origin. However, this mass gap still seems mostly deserted for now, which might again be caused by missing sensitivity. In that case, though, future missions such as LISA[8] should be able to fill this gap.
In addition to the mass range being unusually high for black holes of stellar origin, LIGO-Virgo also found a spin distribution of these black holes which is difficult to explain for stellar origin black holes. Shown in this image (figure 2.2b) is the probability for each event to have a given spin $\chi_{\rm eff}$. Note that for now, it is not possible to resolve the spin of each object participating in an event individually. Nevertheless, all measurements are compatible with very low spin. This is difficult to explain for black holes of stellar origin because as the stellar remnant is smaller than its progenitor, it is expected to spin up rather than down. In contrast, PBHs are expected to have a very low or even zero spin since this is the most likely state for them to be in and it should thus be populated predominantly in the thermal equilibrium shortly after the Big Bang (Chiba and Yokoyama 2017).
Finally, the merger rate detected by LIGO-Virgo allows us to infer a fraction of CDM in PBHs ($f_{\rm PBH}$). The observations are plausible for $f_{\rm PBH}$ as low as 0.1 (Raidal et al. 2019), so even if PBHs do not constitute all or most of the dark matter, they might still be a non-negligible part of it, which warrants further inspection. And there are even more good reasons to study PBHs that we will present next.

2.2. Observational Evidence and Tests

There are numerous formation scenarios for the creation of PBHs (see B. Carr and Kühnel 2020 for a recent review; this whole section is based on their corresponding chapters in general, as is section 2.3). For now (see also section 2.3), it is sufficient to know that all these mechanisms yield a connection between the mass M of a PBH and its formation time, t, namely

$$M \sim \frac{c^3 t}{G} \sim 10^{15} \left(\frac{t}{10^{-23}\mathrm{s}} \right) \mathrm{g}, \tag{2.1}$$

with c being the speed of light in vacuum and G the gravitational constant. If we consider PBHs that formed at roughly the Planck time, i.e. 10^{-43} s after the Big Bang, we would find black holes of just 5 g, whereas black holes formed only 10 s after the Big Bang would already have $10^6\,\mathrm{M}_\odot$. This enormous mass range

[8] https://lisa.nasa.gov/

allows PBHs to be considered as possible explanations for a whole range of open questions while we can use observations at all scales to test our models of them.

2.2.1. Possible Applications

Starting with open questions, we first have a number of microlensing events that still require an explanation. There are observations of OGLE and MACHO within the Milky Way and towards the Large Magellanic Cloud (LMC) that show lensing caused by objects in the planetary mass range (e.g. Niikura et al. 2019). While these could also be free-floating planets, they would need to make up 1% of the CDM, which is some orders of magnitude above their estimated contribution. Elteren et al. 2019 estimate that there are around $5 \cdot 10^{10}$ free-floating planets in the Milky Way with an average mass of $10^{-4}\,M_\odot$, which results in a dark matter fraction of about $10^{-4}\%$ for a Milky Way halo mass of roughly $10^{12}\,M_\odot$. Further microlensing evidence comes from lensing observations of quasars. In some cases, the stellar regions of the galaxies that have been determined as lenses do not align with the background quasar (B. Carr and Kühnel 2020), which indicates the existence of some other type of objects in the planetary mass range within their halos.
Moving further along the mass range, we again find lensing events that provide information. The OGLE survey has detected an abundance of dark lenses in the Galactic bulge, some of which also have distances measured by GAIA[9] (Wyrzykowski and Mandel 2020). With this additional information, it is possible to infer the masses of these lenses. On the one hand, their distribution function is peaked precisely in the mass gap (see section 2.1). On the other hand, the masses tell us that these objects are probably black holes because objects with that mass should else be visible in the electromagnetic spectrum. Combining both facts implies again that these black holes are not remnants of stellar evolution.
PBHs of the same mass can be used to explain the spatial alignment of the infrared and X-ray backgrounds. Through the Poisson effect (see B. Carr and Silk 2018 for a detailed explanation), PBHs could already bind gas halos to them at high redshifts, which could then explain the observed overabundance of halos at high redshift. The accretion of material onto the black hole would power the X-ray source while star formation in the halo would provide infrared radiation.
On a slightly larger mass scale, namely $25\,M_\odot$–$100\,M_\odot$ (Boldrini et al. 2020), PBHs can explain the non-detection of ultra-faint dwarf galaxies even though the sensitivity of modern telescopes should be sufficiently high. Below a radius of a few ten parsecs, these galaxies would be dynamically unstable if their halos are dominated by PBHs.
Lastly, it is unclear where the seeds of the Super-Massive Black Holes (SMBHs) that sit in the centre of every galaxy come from. It is doubtful that stellar remnants of the first stars can grow fast enough to reach up to a few billion solar masses today. However, PBHs might already form with a mass of a million solar masses. These black holes would quickly start growing in mass through accretion and might reach several million solar masses. At that point, they could bind regions to them that are large enough to form entire galaxies. In that way, PBHs could provide the seeds of SMBHs and galaxies alike.

2.2.2. Observational Constraints

Since PBHs cover such an impressive mass range, this whole range can also be used to test the ΛPBH cosmology. All of these tests assume a quasi-monochromatic mass function, i.e. that all PBHs have about the same mass. See section 2.3 for a discussion of mass functions. With that assumption, we can predict that we should observe certain things, and the degree to which we observe them can constrain our model because it should match the observations. These constraints are shown in figure 2.3, in which f_{PBH} is plotted against the mass of the black holes. The different constraints are represented by different colours and different observations are indicated with some abbreviation, all of which we will explain now.
Starting at the lowest mass range, we know that black holes evolve via Hawking radiation. One model for this radiation states that particle-antiparticle pairs of photons are created close to the event horizon by vacuum fluctuations. Normally, these pairs quickly annihilate again, however, a black hole may absorb one partner while the other one is able to escape. Because we want total energy to be conserved, the black hole must absorb the partner with negative energy, corresponding to negative mass. Therefore, a black hole decreases its

[9] https://www.gaia-eso.eu/

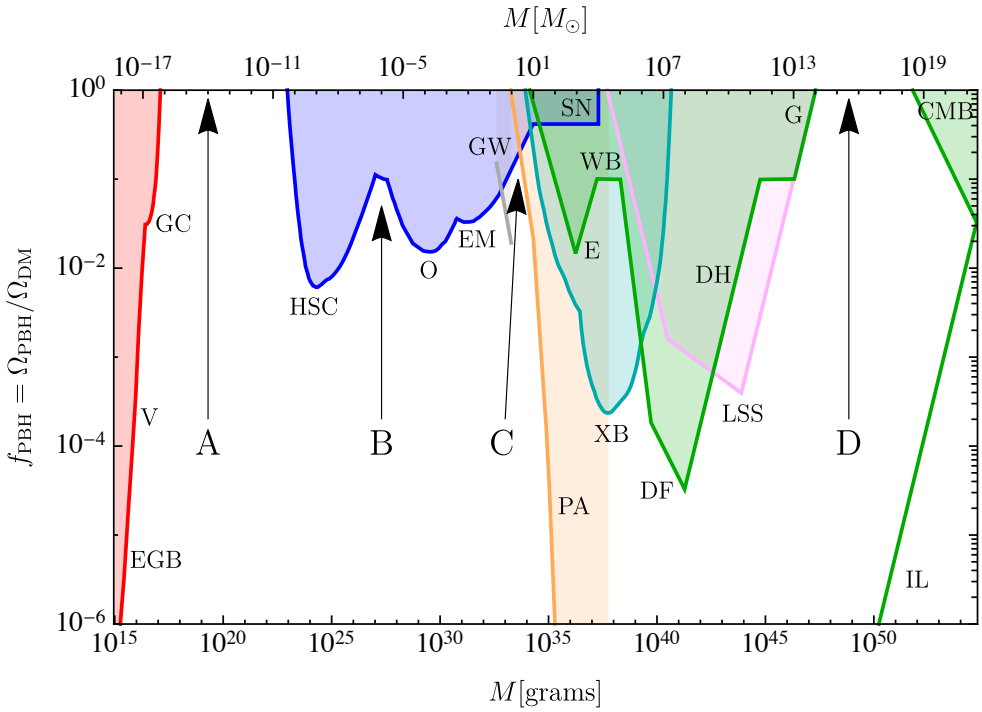


Figure 2.3.: Constraints on $f_{\rm PBH}$ are imposed from evaporation (red), lensing (blue), gravitational waves (grey), the μ-distortion (orange), accretion (turquoise), large-scale structure (purple), and dynamical considerations (green). The resulting mass windows are indicated by A, B, C, D. See section 2.2.2 for more details and B. Carr and Kühnel 2020 for the image source.

mass and finally evaporates due to this process on a time scale $\tau \sim M^3$ unless it has some means of accreting mass. Just before the evaporation is complete, the black holes emit strong jets. Thus, the evaporation can be constrained from measurements of the extragalactic γ-ray background (indicated by the acronym EGB). Additionally, positron measurements made by Voyager 1 have been used (V). Furthermore, the evaporation is visible from the annihilation line radiation at 511 keV, which can be observed coming from the centre of the Milky Way (GC) (Frampton and Kephart 2005). These constraints are shown as the red line in figure 2.3.
At the planetary to solar mass range, PBHs are limited by lensing observations. Under the assumption that PBHs are distributed uniformly in dark matter halos, we can expect a certain frequency of lensing observations if the majority of dark matter is inside black holes of that mass range. Dedicated observations of the Milky Way and M31 with the Subaru Hyper Suprime-Cam (HSC) have inferred an upper bound for $f_{\rm PBH}$, as have the observations of OGLE (O) and EROS and MACHO (EM). What is more, lensing of supernovae type Ia (SN) has also been used to probe PBHs. The supernovae should be demagnified or magnified by the lensing, which is only observed rarely, leading to this limit. All lensing limits are combined to the dark blue line.
Next up, we have the gravitational wave observations by LIGO-Virgo. While these support a primordial origin of the black holes (see section 2.1), they can also constrain the abundance of PBHs. This is because there should not only be singular gravitational wave events, but a whole gravitational wave background (GW) if PBHs constitute a large fraction of the dark matter. In figure 2.3, this limit is shown in grey and should cover the mass range from $0.5\,M_\odot$–$30\,M_\odot$, as indicated in the text of B. Carr and Kühnel 2020, but appears to be somewhat smaller in the image.
Moving further up the mass range, we find two strong constraints. The first one is due to the distortion of the chemical potential μ of the CMB. It is measuring the deviation of the CMB from a perfect black body spectrum and can only be caused during a certain era, namely roughly for redshifts $10^5 < z < 3 \cdot 10^6$. If PBHs originate from the densest parts of Gaussian density fluctuations, these density fluctuations would be dissipated via Silk damping in this so-called μ-distortion era (Chluba, Erickcek and Ben-Dayan 2012). The corresponding energy release would be then enough to cause some deviation from a black body profile. As usual for CMB measurements, our most accurate data stem from the Planck collaboration (PA). This

constraint is shown in orange. The turquoise line represents a more direct constraint coming directly from X-ray observations. PBHs should influence these through their accretion disks (Inoue and Kusenko 2017).
One of the reasons why we need dark matter to explain our observations of the universe comes from structure formation. In the scenario of hierarchical structure formation, small objects form first and then build up to more massive objects. Therefore, there are different redshifts at which we first observe dwarf galaxies, galaxies, and galaxy clusters. Without dark matter, gravity would not have been powerful enough to form these large-scale structures (LSS) in time to match our observations. If, on the other hand, the fraction of PBHs was too high in a certain mass range, they would have acted as very powerful seeds for these structures and we would observe more of it at earlier redshifts than we do. The corresponding constraint is given in purple in figure 2.3.
Finally, there are constraints from dynamical observations shown in green. As mentioned above, PBHs might explain the missing ultra-faint dwarf galaxies (G) due to the tidal disruptions they can cause. Similarly, large quantities of PBHs would affect the stability of other objects, too. Examples of this are globular clusters, one of which has been studied in detail near the centre of Eridanus II (E)(Brandt 2016), and wide binaries (WB) (Quinn et al. 2009), which can be studied in the Milky Way. Furthermore, gravitational interactions with stars would cause dynamical friction (DF), i.e. the drifting of more massive black holes towards the centre of galaxies. This process would cause the centres of galaxies to contain much more mass than is observed unless f_{PBH} is sufficiently small (B. J. Carr and Sakellariadou 1999). Gravitational interactions would also increase the velocity dispersion of the stars in galactic disks, which is called disk heating (DH). Through measurements of the velocity dispersion, we know that this effect has to be limited in strength (Lacey and Ostriker 1985).
On the largest mass scales, we have even more dynamical constraints. In the top right-hand corner, the constraint concerns black holes too large to be contained in a galactic halo. Due to their enormous size, these black holes would still affect all galaxies, such that each galaxy would have a peculiar velocity with respect to the black hole. However, we can use the dipole anisotropy of the (CMB) to conclude that the peculiar velocity of the Milky Way is not large enough to reflect many black holes of these masses. Lastly, the line in the bottom right-hand corner is called the 'incredulity limit' (IL). It is stating that there should be at least one PBH at a given mass scale to warrant the discussion in the first place. These mass scales are chosen as $10^{12}\,\mathrm{M}_\odot$ for galactic halos, $10^{14}\,\mathrm{M}_\odot$ for galaxy clusters, and $10^{22}\,\mathrm{M}_\odot$ for the universe.

With all of these constraints combined, there seem to be only four mass windows left where PBHs could constitute a significant amount of the dark matter, meaning $f_{\mathrm{PBH}} > 0.1$. First, we have window A in the asteroid and sublunar mass range ($10^{12}\,\mathrm{g}$–$10^{22}\,\mathrm{g}$). These objects would have about the size of an atom. Because of the lack of constraints, this window appears to be the cleanest one. Window B lies in the lunar to planet mass range ($\sim 10^{27}\,\mathrm{g}$), which corresponds to black holes of millimeter size. Mass window C in the solar mass range ($\approx 10^{33}\,\mathrm{g}$) was the one that piqued interest in PBHs again due to the detection of gravitational waves. Even these black holes would just have a radius in kilometer range, illustrating that direct observations of them are even harder than usual for objects not emitting any light. Finally, black holes of mass window D with their masses of $\sim 10^{48}\,\mathrm{g}$ and radii of about 50 pc have been termed 'stupendously large black holes (SLABs)' by B. Carr and Kühnel 2020. While these black holes are too massive to sit inside a galactic halo, they might contribute to intergalactic dark matter. The lack of constraints might simply show that this option has not been studied yet.

2.2.3. Caveats

As with all observations, the ones used to derive the constraints presented above come with measurement uncertainties and are sometimes based on assumptions that are not undisputed. Therefore, one finds a discussion about most of them in the literature and we are going to present a non-exhaustive selection of caveats here.
To begin with, the lensing constraints can be significantly weakened or even eliminated by considering that PBHs should form (in) clusters (Belotsky et al. 2019). Usually, constraints as presented in figure 2.3 assume a uniform distribution of black holes throughout a dark matter halo, from which it then follows that lensing events should occur with some frequency. If, however, the black holes are distributed in clusters, then the frequency would be much harder to predict and much lower because large amounts of time can pass before one such cluster crosses our line of sight to a given object. Simultaneously, the strength of the microlensing of

individual black holes in a cluster would be reduced to a level that the surveys used above were not looking for (B. Carr, Clesse et al. 2020). Note that PBHs would not be clustered everywhere, e.g. the stronger potential closer to the centre of a galaxy would disrupt clusters there and smooth the distribution.
It is also important to consider accretion rates in detail. On the one hand, we have to remember that the different constraints probe f_{PBH} at different redshifts: the CMB and structure formation constraints are testing the mass function several gigayears ago, while e.g. lensing observations are tests for the distribution today. Through significant accretion, the mass function of PBHs, i.e. f_{PBH} as a function of black hole mass, might look very different today than it did shortly after the Big Bang. Thus, a mass function that is excluded by constraints at high redshifts might be allowed today, and a detailed study of the accretion rate might warrant a reinterpretation of the constraints above. The accretion rate also affects the μ-distortion limit. Since this distortion can only be caused during a certain era and PBHs cannot cause it if their masses are small enough, these constraints can be avoided if the initial masses of PBHs are small enough and significant accretion starts only after the μ-distortion era (B. Carr and Kühnel 2020).
But the most important caveat is one that affects all constraints at the same time: the assumption of a quasi-monochromatic mass function. All of the limits above have been derived under the assumption that all PBHs have roughly the same mass and would change for an extended mass function. This is essentially because for an extended mass function, we do not constrain a specific mass, but rather require that the mass function integrated over the whole mass range should give f_{PBH}. Translating the constraints to an extended mass function, we then find that the integral over the mass function divided by the maximum value of f_{PBH} for each mass (so the monochromatic values can be plugged in here) should not be larger than 1. For any given distribution of maximum values of f_{PBH}, it is then possible to test any mass function and thus constrain it. This contributes to the discussion of why we should be interested in an extended mass function.

2.3. The Mass Function

Generally speaking, there are two families of mass functions: on the one hand, we have (quasi-)monochromatic mass functions, on the other hand, we have extended mass functions. Historically, monochromatic mass functions were assumed more often and that is not unreasonable. As was briefly mentioned in section 2.2, the mass of a PBH is connected to its time of formation. Thus, if a formation process implies (almost) simultaneity of formation, we can expect the resulting black holes to have (almost) the same mass. Furthermore, it is much simpler to deal with black holes of the same mass than with a composition of different masses. Note that we also use one mean mass of a PBH (M_{PBH}) in the end. Bordering on extended mass function, the idea of a composition of monochromatic mass functions has already been studied (Lehmann, Profumo and Yant 2018). An example of this is shown in figure 2.4. This figure was derived trying to maximise the total fraction of PBHs of dark matter using the given constraints. From left to right, these constraints come from evaporation (evap), femto- and microlensing (FL, HSC, K, EROS, M), and (CMB) measurements. The vertical purple lines represent masses of allowed PBHs, and the total f_{PBH} ($f_{\mathrm{PBH,tot}}$) from all of them is 1.502, so it could comfortably fit all dark matter.
Although this effect is less pronounced for a composite monochromatic mass function, by limiting PBHs to a monochromatic mass function, we would also limit their possible applications, making their discussion less appealing. Additionally, there is no good reason for PBHs to form at exactly these masses or alternatively, why they should stop forming when all dark matter is produced. The formation scenario that has been called 'most natural' is the collapse of primordial density fluctuations, that enter the horizon as overdense regions and collapse if their mass is larger than the Jeans mass. This criterion leads to a critical density threshold δ_{c}, which depends on the equation of state parameter w of the universe. Shortly after the Big Bang, the universe can be described as a perfect fluid, i.e. using the equation of state $p = w\rho c^2$, with p being the pressure and ρ being the density. During this process of overdense regions entering the horizon, the fraction of collapsing regions now declines exponentially with δ_{c} and therefore w (B. J. Carr 1975). In turn, this fraction directly corresponds to f_{PBH}. Looking at this dependence on w alone, it is not clear why there should only be black holes of one mass as there were several periods during which the pressure and consequently w dipped, e.g. at phase transitions, when non-relativistic particles were able to form.
In fact, the thermal history of the universe — clearly showing several phase transitions — has been used to infer an extended mass function by B. Carr, Clesse et al. 2020. Figure 2.5 shows the evolution of w with temperature T. Whenever T crosses a threshold at which particles become non-relativistic, radiation

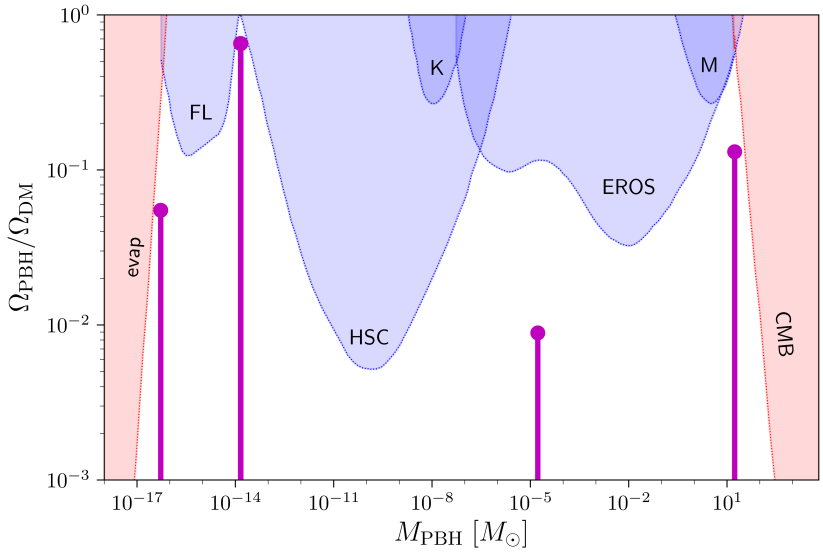


Figure 2.4.: An exemplary composite monochromatic mass function from Lehmann, Profumo and Yant 2018. Constraints are derived from evaporation (evap), femto- and microlensing (FL, HSC, K, EROS, M), and CMB measurements; vertical lines combine to the mass function.

pressure energy is converted into these particles, resulting in a drop of w. This starts at 172 GeV with the top quark and continues with the W and Z bosons, both roughly at 85 GeV. Afterwards, w can almost regain its relativistic value of $1/3$, before both protons and neutrons become non-relativistic during the quantum chromodynamics (QCD) transition at circa 200 MeV. Their combined impact creates the strongest drop in w. Both pions and electrons add further dips later on.
As mentioned above, the fraction of collapsing regions declines exponentially with w. Therefore, even small drops in w are expected to result in an increase of black hole formation. And since we know that the mass of PBHs depends on the formation time, knowing the times of increased formation even tells us the masses where we expect more PBHs. In mathematical terms, the mass function derived from the thermal history can be described as

$$f_{\mathrm{PBH}}(M) \approx 2.4\beta(M)\sqrt{\frac{M_{\mathrm{eq}}}{M}} \tag{2.2}$$

$$M_{\mathrm{eq}} \approx 2.8 \cdot 10^{17}\mathrm{M}_\odot \tag{2.3}$$

$$\beta(M) \approx \mathrm{erfc}\left(\frac{\delta_{\mathrm{c}}(w(T(M)))}{\sqrt{2}\delta_{\mathrm{rms}}(M)}\right) \tag{2.4}$$

$$T(M) \approx 200\sqrt{\frac{\mathrm{M}_\odot}{M}}\mathrm{MeV}, \tag{2.5}$$

where $\delta_{\mathrm{rms}}(M)$ is the root-mean-square amplitude of assumed Gaussian density fluctuations, M_{eq} is the mass of the horizon at the time of equality between matter and radiation energy density, and erfc is the complementary error function. The numerical factor 2.4 represents $2\left(1+\frac{\Omega_{\mathrm{B}}}{\Omega_{\mathrm{CDM}}}\right)$ with the energy density of CDM $\Omega_{\mathrm{CDM}} \approx 0.245$ and the energy density of baryons $\Omega_{\mathrm{B}} \approx 0.0456$, as measured by Planck (Planck Collaboration et al. 2020). Figure 2.6 shows this mass function as the dark blue line. The dashed red line and dotted green line also show this mass function, but they assume slightly different density fluctuations. To be specific, all three versions assume the same scalar power spectrum of masses (i.e. the same distribution of fluctuations with M), but they use different values of the so-called scalar spectral index n_{s} to reflect measurement uncertainties. This parameter can be measured from CMB observations, so the newest results can be found in ibid., where we see $n_{\mathrm{s}} \approx 0.967 \pm 0.004$ varying slightly for different combinations of measurements. In figure 2.6, the red curve corresponds to $n_{\mathrm{s}} = 0.965$, blue to $n_{\mathrm{s}} = 0.97$, and green to $n_{\mathrm{s}} = 0.975$. As we can

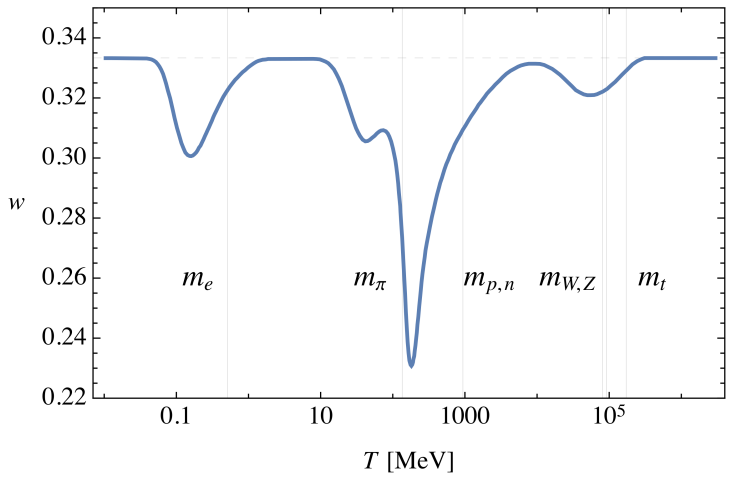


Figure 2.5.: Thermal history of the universe shown as the evolution of the equation of state parameter w with temperature T. The dashed horizontal line indicates the relativistic value of $1/3$ for w; the horizontal grey lines show temperatures where particles become non-relativistic. From right to left: top quarks, W and Z bosons, protons and neutrons, pions, and electrons. Source: B. Carr, Clesse et al. 2020.

see, a lower spectral index leads to a smaller fraction of low-mass black holes, but an increased fraction of massive black holes, while the fraction in the solar mass range remains unaffected. Each of the peaks of the mass function is highlighted by a faint vertical grey line. Some of the adapted constraints are also depicted, namely the ones from microlensing (M), from observations of Eridanus II and ultra-faint dwarf galaxies (E), from X-ray observations (X), from studies of wide binaries (WB), and from accretion (A). The accretion constraint is a dashed line to symbolise the uncertain physical assumptions that it relies on. Finally, there are several vertical lines, each coloured pair representing a different gravitational wave detection. Note that some of them fall into the mass gap, where formation of astrophysical black holes is difficult at least.
There are several remarkable features of this mass function. Most importantly, integrating over the whole mass range yields $f_{\mathrm{PBH,tot}} = 1$, so PBHs could constitute all of the dark matter. At the same time, this mass function does not violate any of the constraints except for the heavily debated accretion constraint. What is more, the peaks enable explanations of all the problems mentioned above: the first bump in the planetary mass range corresponds in position and amplitude precisely to what we need to understand the microlensing observations. The big peak in the solar mass range then explains the combined measurements of OGLE and Gaia in the Milky Way and it accounts for the existence of black holes in the mass gap. Next, the small bump attached to the big peak contains the black holes that can elucidate the correlation of X-ray and infrared background, disrupt dwarf galaxies, and act as the progenitors for some gravitational wave events. Lastly, the PBHs from the peak at $10^6\,\mathrm{M}_\odot$ would naturally become seeds for SMBHs and entire galaxies. So while dealing with an extended mass function is more difficult, it is also more realistic and provides a range of physical benefits. Therefore, we choose to work with this mass function.

2.3.1. Our Mass Function

Through private correspondence with Florian Kühnel, we acquired the data necessary to fit a mass function, which can then be used for further computations. Specifically, he kindly provided us with the data used in B. Carr, Clesse et al. 2020 in form of a Mathematica[10] notebook, but we preferred to use Python for this project, so we recreated the fitting procedure there.
For the fitting of the mass function, we need three tables: both the equation of state parameter w and the number of relativistic degrees of freedom g as functions of the temperature T, as well as the critical

[10] https://www.wolfram.com/mathematica/

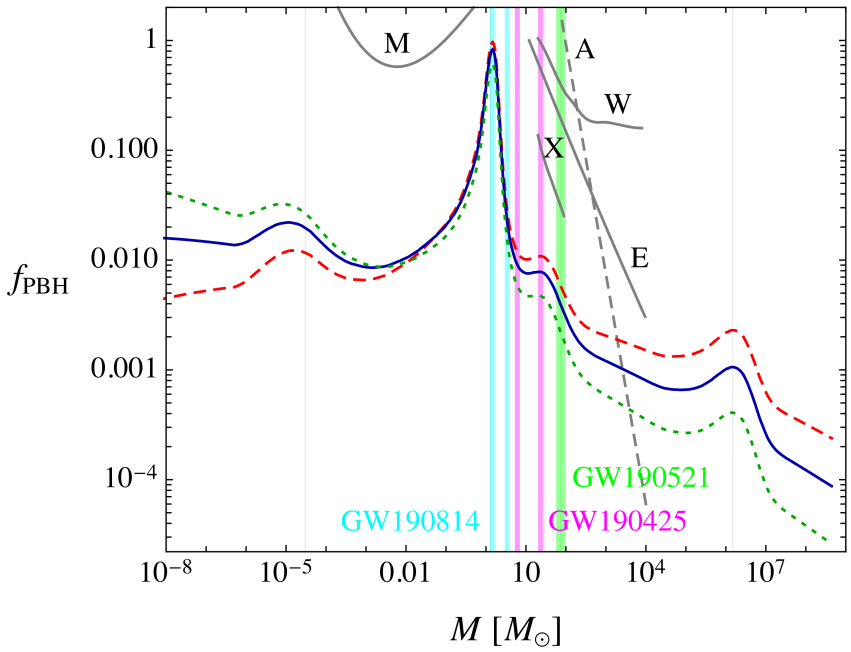


Figure 2.6.: The extended mass function derived from the thermal history of the universe. Red, blue, and green curves represent the mass function with $n_\mathrm{s} = 0.965, 0.97, 0.975$ respectively. Adapted constraints from microlensing, Eridanus II, X-ray observations, wide binaries and accretion are also shown. The vertical colourful lines indicate some progenitors of gravitational waves detected by LIGO/Virgo. Source: B. Carr, Clesse et al. 2020.

density threshold δ_c as a function of w. We copied these data from the notebook, where it is pointed out that the table for $\delta_\mathrm{c}(w)$ stems from Musco and Miller 2013. Then, we interpolate $w(T)$ and $g(T)$ and check via plotting the interpolated functions that they agree with the figures in B. Carr, Clesse et al. 2020. Note that the original data tables have limited ranges, e.g. the temperature range for w is $10 < T[\mathrm{MeV}] < 10^4$, and it is $10^{-2} < T[\mathrm{MeV}] < 3 \cdot 10^5$ for g. That is why we enable the option `extrapolate` at first, so that we do not run into problems while using the interpolated functions later on. Nevertheless, this limited range is also the reason why our mass function looks different in the end, in particular the outer parts.
Next, we compute masses M according to

$$M[\mathrm{M}_\odot] = \left(\frac{700}{g(T[\mathrm{MeV}])^{1/4} \cdot T[\mathrm{MeV}]} \right)^2 , \tag{2.6}$$

with a temperature range of $10^{-1} < T[\mathrm{MeV}] < 10^6$. With these values of M and T, we also interpolate a function $T(M)$. Until now, the interpolations have been linear, but for the last one, $\delta_\mathrm{c}(w)$, we have to use cubic interpolation to recreate the results in the notebook properly. Finally, we can combine all of these functions to one function for $\delta_\mathrm{c}(w(T(M))) = \delta_\mathrm{c}(M)$.
This function now enables us to calculate $f_\mathrm{PBH}(M)$ as shown in equations 2.2–2.5. The only missing ingredient is a description of the density fluctuations, for which we also adapt a power law:

$$\delta_\mathrm{rms}(M) = A \left(\frac{M}{\mathrm{M}_\odot} \right)^{(1-n_\mathrm{s})/4} , \tag{2.7}$$

where A is the amplitude of the fluctuations. It has been used as a free parameter to ensure $f_\mathrm{PBH,tot} = 1$, for which we need $A = 0.00416856$. More precisely, this value yields $f_\mathrm{PBH,tot} = 1.00002$. To complete this numerical integration, we have to split up the integration interval into smaller pieces because otherwise the integration does not converge. When splitting the interval into orders of magnitude, i.e. integrating between $10^x < M[\mathrm{M}_\odot] < 10^{x+1}$ for $x \in [-9, 6]$, the average integration error is of order 10^{-8}. However,

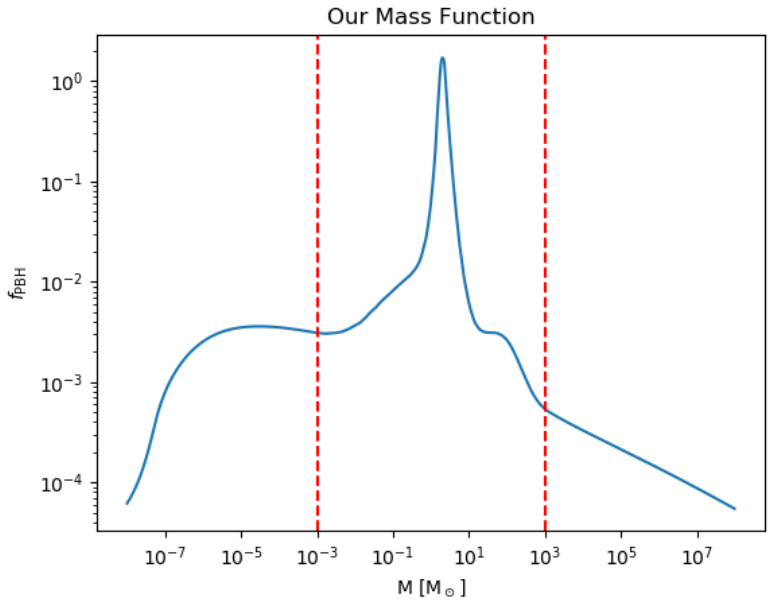


Figure 2.7.: The extended mass function resulting from our fitting procedure. Our limits of the mass range are indicated by the red dashed lines.

the corresponding mass function — shown in figure 2.7 — differs from the one in figure 2.6, especially in the outer parts. As mentioned above, this is due to the limited range of the tables on which we base this fitting process. Therefore, we limit the mass range under consideration to $10^{-3} < M[\mathrm{M}_\odot] < 10^3$, which is represented in figure 2.7 by the red dashed lines. Integrating the mass function between these limits still yields $f_{\mathrm{PBH,tot}} = 0.97$, so we are not missing much of the mass. Again via private correspondence, Florian Kühnel also estimates that the peak at $10^6\,\mathrm{M}_\odot$ would not make a significant difference for our following computations.

2.4. Theoretical Approach

If PBHs indeed constitute a significant fraction of the dark matter, as the thermal history mass function allows, we can expect there to be collisional interactions between these black holes in addition to the usual collisionless attraction. For example, these black holes might be merging with each other, resulting in a gravitational wave background. They might also pass by each other close enough to change their trajectory. This second effect is the one that we want to study and that makes PBHs a form of SIDM.
In his PhD thesis, Robertson 2017 lists two main methods of simulating SIDM: as part of N-body simulations (his section 3.1) or as an effectively collisional fluid (his section 3.3.1). His conclusion is that an effective fluid description is only reasonable for 'highly collisional' dark matter, by which he means that the scattering cross section (σ_{sca}) (see section 2.4.2) divided by the mass m should be larger than $1\,\mathrm{cm}^2\mathrm{g}^{-1}$. This is in contradiction with the findings of several studies for velocity-independent cross sections, which favour $0.1 < (\sigma_{\mathrm{sca}}/m)[\mathrm{cm}^2\mathrm{g}^{-1}] < 1$ (see the review of Tulin and Yu 2018, sections I.C and III, for an extensive summary). For example, Peter et al. 2013 use the shapes of dark matter halos to constrain $(\sigma_{\mathrm{sca}}/m) \leq 0.1\,\mathrm{cm}^2\mathrm{g}^{-1}$, whereas Kahlhoefer et al. 2015 find that $(\sigma_{\mathrm{sca}}/m) \sim 1.5\,\mathrm{cm}^2\mathrm{g}^{-1}$ is necessary to explain the observed offset between the stars and the dark matter halo of a galaxy moving towards the centre of the galaxy cluster Abell 3827. That is why Robertson 2017 treats SIDM as a collisionless fluid within an N-body simulation. However, as we will see in section 2.4.2, our scattering cross section *is* velocity-dependent, which is supported by numerous studies, e.g. Vogelsberger, Zavala and Loeb 2012, Tulin, Yu and Zurek 2013, and Robertson et al. 2019. This means that our cross section is expected to vary with environment, so that we do not need dark matter to be highly collisional everywhere to obey constraints. We therefore argue that we

can employ an effective fluid description for PBHs, which has been used for other types of dark matter in theoretical models (e.g. Pospelov, Ritz and Voloshin 2008, Arkani-Hamed et al. 2009, Boddy et al. 2014) and has been tested in simulations on cluster scale (e.g. Banerjee et al. 2020) as well as galaxy scale (e.g. Correa 2021, Nadler et al. 2020).

2.4.1. Effective Fluid Description

Tulin and Yu 2018 mention three examples of papers employing a fluid description of dark matter. However, Kaplinghat, Keeley et al. 2014 do not go into detail about their implementation (while still illuminating their model and the influence of baryons and feedback). Kaplinghat, Tulin and Yu 2016 illustrate their halo model more closely, but they split their halos in a collisional and a collisionless part, which is not what we are looking for. Finally, a more thorough explanation of implementing self-interactions is given in Rocha et al. 2013.
Following their SIDM model, every one of the N particles contains many PBHs (in our case). Each particle has a single velocity and fills some spatial volume. Therefore, considering the phase-space, every particle can be assigned a delta function for its velocity and a spatially extended function for configuration space. For this latter smoothing function, they use the same function $W(r,h)$ that was used per default in GADGET-2[11]:

$$W(r,h) = h^{-3} w(r/h), \tag{2.8}$$

where

$$w(x) = \frac{8}{\pi} \begin{cases} 1 - 6x^2 + 6x^3, & 0 \leq x \leq 1/2 \\ 2(1-x)^3, & 1/2 < x \leq 1 \\ 0, & x > 1 \end{cases} \tag{2.9}$$

Here, r is the distance of the particles and h is their smoothing length, so for $r > h$, the smoothing kernel drops to zero. In their simulations, ibid. fix h for all particles for the sake of simplicity.

The Boltzmann equation

As we are considering PBHs interacting with one another, there will be interactions between them that will lead to a flow of mass e.g. toward the center of a dark matter halo. Therefore, our system is not in thermodynamic equilibrium, which implies that we need to use the Boltzmann equation to describe it. For this part in particular, Piattella 2018 is a comprehensive collection of lecture notes on cosmology. Following the notation in there (cf. their section 3.7), the Boltzmann equation is

$$\frac{\mathrm{d}f}{\mathrm{d}t} = C[f], \tag{2.10}$$

where f is the distribution function for one particle and the interactions between the particles are described by the functional C, whose action on f is called the collisional term. f is a function of time t, position, and momentum of the particle. In the case without short-range interactions between individual microscopic particles, the system will instead be described by the Vlasov equation:

$$\frac{\mathrm{d}f}{\mathrm{d}t} = 0 \tag{2.11}$$

For our case, though, we need to determine the collisional term since scattering occurs only at short ranges between individual particles. Section 3.8 of ibid. states that this term has the dimension of inverse time and, thus, that it is a scattering rate. In their appendix A, Rocha et al. 2013 derive an expression of the scattering rate of interacting hard spheres from the Boltzmann equation, which we are adopting here.

[11] https://wwwmpa.mpa-garching.mpg.de/galform/gadget/html/index.html

The Scattering Rate

With their expression of a scattering rate, called R_{sca} or Γ interchangeably, scattering, i.e. an overlap of two of the simulation particles, can be resolved by computing the interaction rate between them. Here, the scattering rate of simulation particles i and j with mass m_{p}, positions $r_{\mathrm{i,j}}$, and velocities $v_{\mathrm{i,j}}$, is given by

$$R_{\mathrm{sca},1} = \rho_{\mathrm{ij}} |v_{\mathrm{i}} - v_{\mathrm{j}}| \frac{\sigma_{\mathrm{sca}}}{m}, \tag{2.12}$$

with ρ_{ij} being the density for scattering of j by i,

$$\rho_{\mathrm{ij}} = m_{\mathrm{p}} \int d^3 r W(|r - r_{\mathrm{i}}|, h) W(|r - r_{\mathrm{j}}|, h). \tag{2.13}$$

Note that the original definition of R_{sca} in the paper is slightly different; it already includes m_{p} whereas we have put it into equation 2.13. Additionally, the arguments of the kernels look different. This should not matter, though, as we can choose an arbitrary point of origin for our reference frame.
At first glance, it might look like we are mixing $(\sigma_{\mathrm{sca}}/m)$, which is a quality of the unresolved PBHs, with properties of the resolved simulation particles such as $|v_{\mathrm{i}} - v_{\mathrm{j}}|$. This should work, however, because we can assume that the velocity of the black holes contained in any simulation particle is on average the same as that of the particle, i.e. the particle corresponds to the mean of a local ensemble of black holes. The same holds for the distance of the particles and black holes.
From Tulin and Yu 2018, we can find two alternative explicit expression for the scattering rate. The first approach has mainly been used by earlier studies, such as Yoshida et al. 2000:

$$R_{\mathrm{sca},2} = \rho_{\mathrm{DM}} v_{\mathrm{rel}} \frac{\sigma_{\mathrm{sca}}}{m}, \tag{2.14}$$

where ρ_{DM} is the mass density of dark matter and $v_{\mathrm{rel}} = |v_{\mathrm{i}} - v_{\mathrm{j}}|$. When introduced in Tulin and Yu 2018, it seems to only depend on the radial distance, so it is probably either isotropic or averaged over θ and ϕ. In a later section, there is a discussion on scattering probabilities, of which only the first one follows this scattering rate. The corresponding scattering probability within a time step Δt is simply $R_{\mathrm{sca},2}\Delta t$. In this probabilistic description, ρ_{DM} is defined as the local background density found as a spatial average over the closest neighbours.
Lastly, Vogelsberger, Zavala and Loeb 2012 present a method combining the other two. The scattering rate looks like the first one, but here,

$$\rho_{\mathrm{ij}} = m_{\mathrm{p}} W(|r_{\mathrm{i}} - r_{\mathrm{j}}|, h). \tag{2.15}$$

This is the expression for ρ_{ij} that we will end up using because it is already built into the simulation code (see section 3.3.4). The smoothing length is adapted so that ∼40 particles are counted as neighbors of i, making it larger than in Rocha et al. 2013, i.e. the scattering processes are less localised. In this case, the total probability is given by $P_{\mathrm{i}} = \frac{1}{2}\sum_{\mathrm{j}} P_{\mathrm{ij}}$, so the scattering rate would be

$$R_{\mathrm{sca},3} = \frac{1}{2} \sum_{\mathrm{j}} \rho_{\mathrm{ij}} |v_{\mathrm{i}} - v_{\mathrm{j}}| \frac{\sigma_{\mathrm{sca}}}{m}, \tag{2.16}$$

with ρ_{ij} adjusted as explained above.
Interestingly, equation (7) in Mouri and Taniguchi 2002 is a rate coefficient $R_{\mathrm{i,j}}$ for merging black holes. It has the dimension of volume per time, so one could retrieve a merging rate by multiplying this equation with something like m/ρ_{DM}. Since $R_{\mathrm{i,j}} = \langle v_{\mathrm{rel}} \sigma_{\mathrm{mer}} \rangle$, this merging rate would be of the same form as $R_{\mathrm{sca},2}$ with the only differences being the cross section and the averaging with respect to v_{rel}, essentially.

2.4.2. Average Quantities

As mentioned in section 2.4.1, we will not resolve individual black holes in our simulation. Since we want to consider the cosmological problem of structure formation, this would require too much computational power. Therefore, we cannot assign masses to simulation particles according to the mass function presented in section 2.3.1. Instead, we are assuming that each simulation particle contains PBHs obeying that mass function. That allows us to derive average quantities from the function and use them for all particles alike. Here, we start with an average mass because it is more straightforward.

The Average Mass

Equation (3) in B. Carr, Clesse et al. 2020 defines

$$f_{\mathrm{PBH}}(M) = \frac{1}{\rho_{\mathrm{CDM}}} \frac{\mathrm{d}\rho_{\mathrm{PBH}}(M)}{\mathrm{d}\log M}, \tag{2.17}$$

which implies

$$f_{\mathrm{PBH,tot}} = \int f_{\mathrm{PBH}}(M)\mathrm{d}\log M = \frac{\rho_{\mathrm{PBH}}}{\rho_{\mathrm{CDM}}}. \tag{2.18}$$

Thus, if $\rho_{\mathrm{PBH}} = \rho_{\mathrm{CDM}}$, then $p(M) = f_{\mathrm{PBH}}/M$ is the probability distribution function for PBH mass M. The mean mass is then

$$\langle M \rangle = \int M\, p(M)\, \mathrm{d}M = \int f_{\mathrm{PBH}}(M)\mathrm{d}M. \tag{2.19}$$

Integrating f_{PBH} with respect to the mass using our mass limits of $10^{-3}\,\mathrm{M}_\odot$ and $10^{3}\,\mathrm{M}_\odot$ with the Python routine, we find an average mass of $3.09\,\mathrm{M}_\odot$. Using the interval from $10^{-2}\,\mathrm{M}_\odot$ to $10^{2}\,\mathrm{M}_\odot$ instead yields an average mass of $2.26\,\mathrm{M}_\odot$. This small deviation from our chosen interval again supports that the values we excluded from these computations would not change the result significantly.

The Average Scattering Cross Section

In Mouri and Taniguchi 2002, the scattering cross section σ_{sca} is defined to be

$$\sigma_{\mathrm{sca}} \simeq \pi \left(\frac{Gm}{v_{\mathrm{rel}}^2}\right)^2, \tag{2.20}$$

without further defining m and $v_{\mathrm{rel}} = |v_1 - v_2|$ being the initial relative velocity if the two bodies are denoted by 1 and 2. The expression in brackets should be equal to some kind of scattering radius r_{sca} or in other words,

$$r_{\mathrm{sca}} = \alpha \frac{Gm}{v_{\mathrm{rel}}^2}, \tag{2.21}$$

absorbing the proportionality factor in α. This can be rewritten as

$$\frac{m_1 m_2 v_{\mathrm{rel}}^2}{\alpha m} = \frac{G m_1 m_2}{r_{\mathrm{sca}}}, \tag{2.22}$$

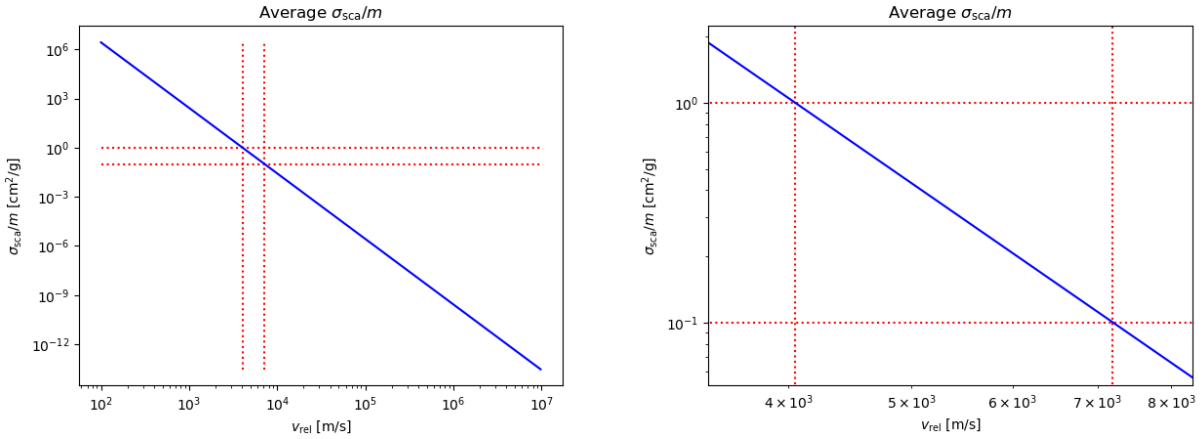
which looks a lot like a criterion relating energies. The right-hand side already gives the potential energy of the two bodies while the left-hand side represents the kinetic energy in the centre-of-mass frame for $\alpha = 2$ and $m = m_1 + m_2$. Thus, the scattering radius is the distance between two bodies inside of which the potential energy dominates over the kinetic energy. The corresponding scattering cross section is then

$$\sigma_{\mathrm{sca}} = 4\pi \left(\frac{Gm}{v_{\mathrm{rel}}^2}\right)^2. \tag{2.23}$$

Owing to the velocity dependence, this cross section differs depending on the environment. With the relative velocities declining from clusters over groups to the field, we can expect the cross section and, consequently, the scattering rate to increase at the same time.
Analogous to the average mass, the scattering cross section averaged over the masses is given by

$$\langle \sigma_{\mathrm{sca}} \rangle (v_{\mathrm{rel}}) = \int p(M_1, M_2) \sigma_{\mathrm{sca}}(M_1, M_2, v_{\mathrm{rel}}) \mathrm{d}M_1 \mathrm{d}M_2 \tag{2.24}$$

Initially, we tried to compute $p(M_1, M_2)$ as the convolution of $p(M_1)$ and $p(M_2)$, but this kept leading to complex numbers with which our Python routine was unable to find a solution. However, by assuming that the probability of finding a mass M_2 after finding M_1 is independent of the probability of finding M_1, we can to first order approximate $p(M_1, M_2) = p(M_1)p(M_2)$.



(a) Showing the whole velocity range computed.

(b) Only for the part with desired values of σ_{sca}/m.

Figure 2.8.: Average gravitational scattering cross section divided by the average mass as a function of the relative velocity. Red dotted lines indicate preferable values of σ_{sca}/m and their velocity counterparts.

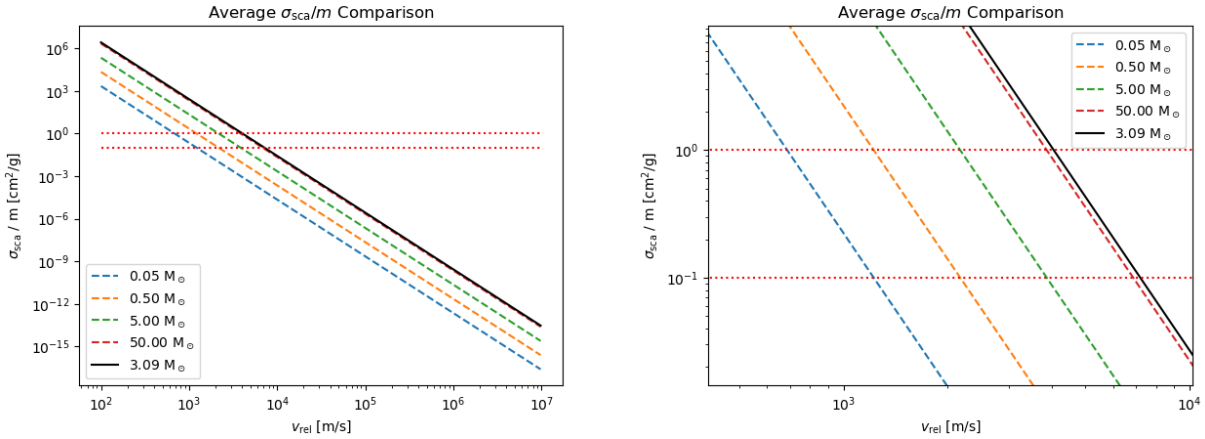
For some reason, the integration for the mean cross section over the same range with `scipy.integrate.dblquad`[12] is not converging. Unfortunately, we cannot pass options to `dblquad`, such that even splitting up the integration range in orders of magnitude manually does not work. Luckily though, we can pass options with `scipy.integrate.nquad`[13], which enables us to increase and limit the amount of subdivisions allowed for each integral and change the desired numerical error (both absolute and relative) from 10^{-8} to 10^{-3}. Even with all these measures, the integration for the mean scattering cross section still takes a few minutes. In fact, Lukas Winkler suggested changing the interpolation and extrapolation of the equation of state parameter w and the relativistic degrees of freedoms g. He changed the interpolation style to cubic and extrapolated manually, e.g. for w a constant extension for $T < 10\,\text{MeV}$ and a linear one for $10^4\,\text{MeV} < T$. However, he still could not integrate over the whole range, so that we decided to stick with our initial extrapolation. Nevertheless, his functions are significantly faster to integrate (less than a minute) and produce very similar results, e.g. the difference in the average mass is about $0.003\,\text{M}_\odot$. Therefore, if there are more integrations like this to be done, we might want to change to his version.
In this case, though, we only need one numerical integration to know the behaviour of σ_{sca} on the whole velocity range because the relative velocity does not depend on the mass and can therefore be pulled in front of the integral. So, we determine the average cross section for $2000\,\frac{\text{m}}{\text{s}}$, multiply the result with 2000^4 and use this value as γ in $\langle\sigma_{\text{sca}}\rangle = \gamma v_{\text{rel}}^{-4}$. Afterwards, we can plot that function divided by the average mass for different values of v_{rel} ranging from $10^2\,\frac{\text{m}}{\text{s}}$ to $10^7\,\frac{\text{m}}{\text{s}}$. The result is figure 2.8. On the left, we show the whole range of velocity, whereas the right shows a close-up of the region that is relevant for arriving at $0.1\,\text{cm}^2\text{g}^{-1}$ to $1\,\text{cm}^2\text{g}^{-1}$ for (σ_{sca}/m). A value of $1\,\text{cm}^2\text{g}^{-1}$ is achieved for $4048.3\,\frac{\text{m}}{\text{s}}$, whereas the $0.1\,\text{cm}^2\text{g}^{-1}$ is reached for a relative velocity of $7192.8\,\frac{\text{m}}{\text{s}}$.

Comparison with a Monochromatic Mass Function

As a quick sanity check, let us compare our cross section to the cross sections derived from monochromatic mass functions. For this purpose, we assume a probability distribution function of the form $p(M) = \delta(M - M_0)$ and check several values of M_0 representing different monochromatic mass functions.This eliminates the need for numerical integration as the delta distributions resolve any integral over them. In particular, we now have

$$\langle\sigma_{\text{sca}}\rangle(v_{\text{rel}}) = \sigma_{\text{sca}}(M_0, M_0, v_{\text{rel}}). \tag{2.25}$$

[12] https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.dblquad.html
[13] https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.nquad.html



(a) ...for the whole velocity range.

(b) ...focusing on the preferred cross section values.

Figure 2.9.: Scattering cross section from monochromatic mass functions compared to the averaged value of our mass function...

As can be seen in figure 2.9, the shape of the cross section does not change for monochromatic mass functions. This is because σ_{sca} is still proportional to v_{rel}^{-4}, so different masses will only influence the proportionality constant, which rises with the square of the mass. Hence, the monochromatic mass functions give increasing values of σ_{sca}/m as their mass is rising. Notably, though, the overall values are much lower than the one coming from the averaged broad mass spectrum. We need as much as $50\,\mathrm{M}_\odot$ to get to a similar range. So, generally, a broad mass function seems to increase σ_{sca}/m by almost two magnitudes for a fixed v_{rel}. This is probably due to the fact that the average cross section can be dominated by a few very large ones caused by the biggest PBHs, while the average mass stays comparatively low. In turn, the relative velocities that still lead to appreciable values of the cross section, which are again indicated by the red dotted lines, are roughly two times larger than those allowed for a monochromatic mass function peaking circa at our average mass. Since this change is quite significant, we are intrigued by how this change will affect structure formation. To study this question, we now turn to the cosmological simulation code that we use for this project.

3. Simulations

3.1. Code Selection

The first thing we need to decide is which simulation code we want to use. Over the last decade, there have been a multitude of simulations based on different codes, all published in this century. To name just a few, Benson 2012 presented the GALACTICUS model and code for galaxy formation, for which he assumed collisionless matter. Another model for galaxy formation and evolution, SAGE, was published by Croton et al. 2016. It was based on the result of various simulations that are largely derived from the GADGET-2 code by Springel, S. D. M. White et al. 2005. As the name suggests, this was the second version of the original GADGET code (Springel, Yoshida and S. D. White 2001), which is a code for collisionless cosmological simulations. Pillepich et al. 2017 introduced the IllustrisTNG simulations of cosmological galaxy formation. These were based on the AREPO code (Springel 2010), that also used a collisionless treatment of dark matter. Yet another set of cosmological galaxy formation simulations called SIMBA was released by Davé et al. 2019. With their use of the GIZMO code (Hopkins 2015, Hopkins 2017), they, too, rely on collisionless dark matter scenarios.
While this list is by no means exhaustive, it illustrates the main problem: none of these codes already contains a formulation of SIDM. This is not surprising since the standard cosmology contains only CDM that is not interacting with itself. However, this implies that there is no clear best choice because we will have to implement the scattering interactions on our own in each code. Moreover, all of these codes have advantages and disadvantages compared with each other so that none of them is the best generally speaking. It always depends on the setting one is interested in: resolution, computation speed and accuracy, different features like the inclusion of stellar feedback might all make one code desirable in one case, but another in the next one. For our purpose, we select GADGET-4, which is the newest version of the well-established GADGET code and was introduced recently by Springel, Pakmor et al. 2020. It is therefore also a combination of an N-body code with an SPH simulation, where the N bodies or particles represent a self-gravitating collisionless fluid and the SPH part describes collisional gas particles. The code is publicly available on its website[14], which contains some documentation, too.
Apart from this being a state-of-the-art cosmological simulation code, there are two main reasons for our choice: the first is familiarity. Despite GADGET not being known for its extensive documentation, my supervisor Oliver Hahn has already worked a lot with earlier versions of this code, so that we had a firm basis to start from. As for the specific version, we did not choose GADGET-2 because some required repositories are more than a decade old by now and it would be unnecessarily complicated to try and get them to work (GADGET-3 is not publicly available). Furthermore, the newer version provides a range of improvements regarding computational efficiency and accuracy, among others (see ibid., the documentation paper). Additionally, as we will see in section 5.2, this project may be used as a stepping stone to even better simulations of PBHs in the future, for which the most up-to-date version of the code will be even better suited. The second main reason, then, is that Rocha et al. 2013 implement their scattering interactions with GADGET-2. Since we are largely following their method, we also want to use GADGET for consistency. Besides, owing to its structure with a combination of N-body and SPH methods, we can rather easily implement our scattering interactions in the code.

3.2. Implementation Idea

In GADGETs basic SPH formulation, originally inspired by Gingold and Monaghan 1977, a Lagrangian description of the fluid is used, i.e. individual gas particles and their scatterings are considered. If two

[14] https://wwwmpa.mpa-garching.mpg.de/gadget4/

particles are determined to scatter, they receive velocity kicks according to the details of the scattering interaction. This procedure sounds very similar to our goal, and we can indeed hijack the hydrodynamic part of the code for a first implementation of the scattering.

3.2.1. Determining Interaction Occurrence

To decide whether there will be an interaction or not, we draw a random number between 0 and 1, and compare it to the scattering probability. The probability for an interaction of one particle around $(\vec{x}_\mathrm{i}, \vec{v}_\mathrm{i})$ and another particle at $(\vec{x}_\mathrm{j}, \vec{v}_\mathrm{j})$ during a time step δt is given by

$$P_\mathrm{ij} = \frac{P(i|j) + P(j|i)}{2}, P(i|j) = \Gamma(i|j)\delta t. \tag{3.1}$$

Here, we have ensured that the scattering probability is symmetric by taking the mean of the two individual probabilities of particle i scattering off of j and vice versa. Symmetry of the scattering is important because of energy conservation; if one particle scatters off of another, both particles should receive a corresponding kick. At the same time, this allows for computational efficiency since we only need to compute the scattering probability once for each pair and time step. We only need to guarantee that each pair is considered not more than once during every time step.
Determining a scattering event is quite simple, then: If the random number is bigger than the scattering probability, there will be no interaction. If the random number is smaller than or equal to the scattering probability, there will be an interaction.

3.2.2. Velocity Kicks

If there is an interaction going on between two particles, both will be given a kick corresponding to elastic scattering. For elastic collisions, the modulus of the relative velocity of the particles is conserved. Moreover, the mass of each particle should be conserved. The scattering is then considered in the centre of mass frame, where it is assumed to be isotropic.
From the definition of the centre of mass position,

$$\vec{x} = \frac{m_\mathrm{i}\vec{x}_\mathrm{i} + m_\mathrm{j}\vec{x}_\mathrm{j}}{m_\mathrm{i} + m_\mathrm{j}}, \tag{3.2}$$

applying a time derivative yields

$$\vec{v} = \frac{m_\mathrm{i}\vec{v}_\mathrm{i} + m_\mathrm{j}\vec{v}_\mathrm{j}}{m_\mathrm{i} + m_\mathrm{j}}. \tag{3.3}$$

The momentum should be conserved, such that

$$\vec{v} = \frac{m_\mathrm{i}\vec{v}_\mathrm{i} + m_\mathrm{j}\vec{v}_\mathrm{j}}{m_\mathrm{i} + m_\mathrm{j}} = \frac{m_\mathrm{i}\vec{v'}_\mathrm{i} + m_\mathrm{j}\vec{v'}_\mathrm{j}}{m_\mathrm{i} + m_\mathrm{j}}, \tag{3.4}$$

where a prime denotes quantities after scattering. Solving this e.g. for $\vec{v'}_\mathrm{i}$ gives

$$\vec{v'}_\mathrm{i} = \vec{v} + \frac{m_\mathrm{j}}{m_\mathrm{i}}\left(\vec{v} - \vec{v'}_\mathrm{j}\right) = \vec{v} + \frac{m_\mathrm{j}}{m_\mathrm{i} + m_\mathrm{j}}\left(\vec{v}_\mathrm{i} - \vec{v}_\mathrm{j}\right). \tag{3.5}$$

We can now write $\vec{v}_\mathrm{i} - \vec{v}_\mathrm{j} = |\vec{v}_\mathrm{i} - \vec{v}_\mathrm{j}|\vec{e} := V\vec{e}$, where $\vec{e}$ can be chosen randomly (as was done by Rocha et al. 2013) to mimic isotropic scattering. This results in the following velocities after scattering:

$$\vec{v'}_\mathrm{i} = \vec{v} + \frac{m_\mathrm{j}}{m_\mathrm{i} + m_\mathrm{j}} V\vec{e} \tag{3.6}$$

$$\vec{v'}_\mathrm{j} = \vec{v} - \frac{m_\mathrm{i}}{m_\mathrm{i} + m_\mathrm{j}} V\vec{e} \tag{3.7}$$

3.2.3. Picking a Random Direction

Picking points on a sphere randomly cannot be done by simply choosing spherical coordinates $\theta \in [0, 2\pi)$ and $\phi \in [0, \pi]$[15]. Instead, there are several ways to do this properly. The easiest one appears to be drawing two random variables u and v from $(0, 1)$ and converting them to spherical coordinates via

$$\theta = 2\pi u \tag{3.8}$$

$$\phi = \arccos(2v - 1) \tag{3.9}$$

This method seems slightly flawed, though, as it apparently excludes $\theta = 0$ and $\phi = \{0, \pi\}$. However, this should be fine because it only concerns single points. Thus, after drawing θ and ϕ, we can quickly transform the corresponding unit vector to the Cartesian coordinates used by GADGET-4.

3.3. Practical Realisation

As is often the case when going from theory to practice, we could not immediately implement everything just as we imagined. There were several (un-)expected problems, some of which we were able to solve. In order to not artificially inflate the volume of this work, we are not going to focus on these mistakes, but rather on how we did implement the scattering in the end. That way, this work might even serve as a documentation for interested readers in the future. Let us start with a schematic overview of our algorithm, which will be explained in detail in the following sections.

3.3.1. Schematic Summary

To summarise, algorithm 1 shows what we implemented in order to calculate and evaluate the scatter interactions of PBHs. Note that this overview contains only the main ideas while leaving out organisational details. For example, we have to set the variable holding the velocity change of every active particle, `HydroAccel`, to zero at the beginning of each run of the algorithm. There are also a number of diagnostic variables that allow us to gain further insight in how our algorithm is working. Furthermore, we will of course explain the data structures and lists below, although their names mostly indicate their functions already. All of these details are included below, but not in this schematic overview.

3.3.2. N-body Codes

To begin with, it is useful to introduce N-body codes in order to understand some terminology. A more detailed yet not too technical explanation can be found in chapter three of Robertson 2017, where this section is drawing from. The core of an N-body code is to describe the evolution of N particles that are interacting via gravity, where all other particles contribute to the gravitational force exerted on each particle. This evolution is usually governed by differential equations, the laws of motion. Since the gravitational force only depends on the distance between two particles, there are only two quantities that need to be tracked: the particle's position $\vec{x}$ and its velocity $\vec{v}$, which is the time derivative of its position. With gravity as the only force, all time changes to the velocity, i.e. accelerations $\vec{a}$, arise from the gravitational potential Φ, which can be retrieved from the density ρ according to the Poisson equation:

$$\nabla^2 \Phi = 4\pi G \rho. \tag{3.10}$$

An N-body code now performs very small time steps δt, during which the velocity can be assumed to stay constant. It can thus be used to update the particle's position, a so-called 'drift' by

$$\delta \vec{x} = \vec{v} \delta t. \tag{3.11}$$

[15] https://mathworld.wolfram.com/SpherePointPicking.html

Data: ρ_{ij}, $\vec{v}_{i,j}$, σ_{sca}/m, $\delta t_{i,j}$, $h_{i,j}$, a
Result: computes scattering probability and exchanges momentum based on that

Create `scatter_list` to contain all `scatter_events`;
for $i \leftarrow 1$ **to** N **do**
  Determine drift time step δt_{drift} at current scale factor a;
  Find desired number (K) of neighbours and store their data in `Ngbhydrodat`;
  Calculate physical time step δt_i of particle i;
  for $j \leftarrow 1$ **to** K **do**
    if *ID of* $j \geq$ *ID of* i **then**
      Skip this pair for now;
    end
    Compute distance d of the pair;
    if *distance of* i *and* $j \geq$ *both particles' smoothing lengths* $h_{i,j}$ **then**
      Skip this pair;
    else
      Calculate relative velocity $\vec{v}_{rel}$;
      if $\frac{\delta t_{drift} d}{\|\vec{v}_{rel}\|} > 1000a$ **then**
        Skip this pair;
      end
      Calculate physical time step δt_j of particle j;
      Compute $R_{sca,2}(i|j)$ and $R_{sca,2}(j|i)$;
      Define $P(i|j) \coloneqq R_{sca,2}(i|j)\delta t_i$ and $P(j|i) \coloneqq R_{sca,2}(j|i)\delta t_j$;
      if $m_i \neq m_j$ *or* $h_i \neq h_j$ *or* $\delta t_i \neq \delta t_j$ **then**
        $P_{ij} \coloneqq \frac{P(i|j)+P(j|i)}{2}$;
      else
        $P_{ij} \coloneqq P(i|j)$;
      end
      Draw a uniform random number $u \in [0, 1]$;
      if $u \leq P_{ij}$ **then**
        Add a `scatter_event` to the `scatter_list` for the pair and its `scatter_prob` P_{ij};
        Increase count of `nscatterevents`;
      end
    end
  end
end
Sort `scatter_list` by ascending `scatter_prob`;
Create `scatter_accel_update_list`;
for $s \leftarrow 1$ **to** *nscatterevents* **do**
  Calculate relative velocity $\vec{v}_{rel}$ of the pair;
  Compute velocity of the centre of mass;
  Draw a random direction $\vec{e}$ via sphere point picking;
  Determine new velocities $\vec{v}_i'$ and $\vec{v}_j'$ according to equation 3.6;
  Calculate difference between old and new velocities $\delta\vec{v}_i = \vec{v}_i' - \vec{v}_i$, $\delta\vec{v}_j = \vec{v}_j' - \vec{v}_j$;
  Save cumulative change of each particle as a `scatter_accel_update` in `scatter_accel_update_list`;
  For each distinct particle, increase `ndistinctparticles`;
end
Share `scatter_accel_update_lists` and `ndistinctparticles` between all processors;
for $l \leftarrow 1$ **to** *ndistinctparticles_total* **do**
  Update `HydroAccel` of particle l with cumulative velocity change from `scatter_accel_update_list_global`;
end

Algorithm 1: Schematic scattering algorithm of our fluid model

Afterwards, the velocity is updated, it receives a 'kick' according to

$$\delta\vec{v} = \vec{a}\delta t = -\nabla\Phi\delta t. \tag{3.12}$$

These operations are performed half a time step apart, which is called a 'leapfrog integration'.
Since the gravitational potential is inversely proportional to the distance of the particles, it diverges as particles reduce their distance to zero. This leads to enormous accelerations, which have to be treated carefully. A common solution for this is to reduce the time steps until such a close encounter is resolved. However, it would be highly inefficient to reduce the time steps of all particles because for most particles, integration with larger time steps would still be fine and small time steps would perform redundant work. It is therefore common practice to have particles on different time steps: particles that need to be treated carefully reside on smaller time steps than particles without strong interactions. The code then performs the smallest necessary time step only for those particles that need it. For this reason, particles can be divided into 'active' and 'passive' particles depending on whether their gravitational interactions are calculated during the current time step.
For cosmological simulations, the expansion of space has to be considered, too. This can be achieved by switching to comoving coordinates, i.e. switching to a frame of reference that is moving with the expansion of the universe also known as the Hubble flow. From physical coordinates $\vec{x}$, we can get to the comoving coordinates $\vec{r}$ employed by GADGET-4 via the scale factor a,

$$\vec{x}(t) = a(t)\vec{r}, \tag{3.13}$$

with the scale factor describing the expansion of the universe. It is related to physical time via the Hubble parameter $H(a)$,

$$\frac{da}{dt} = aH(a), \tag{3.14}$$

which can be used to describe the expansion of the universe, too. In GADGET-4, the time coordinate t is chosen to be

$$t = \log a \tag{3.15}$$

because a fixed step size in t then corresponds to time steps with the size of a fixed fraction of the Hubble time, which is the inverse of the Hubble constant (the value of $H(a)$ today, so $H(1) = H_0$) and hence roughly the age of the universe. Additionally, GADGET-4 uses the conjugate momentum $\vec{p}$ as its velocity variable with

$$\vec{p} = a^2\frac{\mathrm{d}\vec{r}}{\mathrm{d}t}. \tag{3.16}$$

With these choices made, it can be shown that the equations of motion look slightly different to before and imply different drifts and kicks:

$$\mathrm{d}\vec{x} = \vec{p}\int\frac{\mathrm{d}t}{a^2H(a)} \tag{3.17}$$

$$\mathrm{d}\vec{p} = \vec{a}\int\frac{\mathrm{d}t}{aH(a)} \tag{3.18}$$

In principle, these integrals representing the correct drift and kick time steps have to be solved for every particle, but in practice, the solution applies to all particles on the same synchronised time step. For us, it is therefore simply important to keep in mind that the quantities need not be physical quantities and that we have to consider differently sized time steps for the kick and drift operations.
Another thing that would be highly inefficient is to actually calculate the contributions of all other particles to the force acting on each particle. It is much faster to use a hierarchical tree algorithm like the one introduced by Barnes and Hut 1986 and shown in figure 3.1. For this algorithm, the simulation space is divided into eight cubic cells containing some particles. Each cell or `node` is allowed to have a maximum of one particle; if there are more particles present in a node, it will be subdivided into eight further nodes. This procedure is continued until only 'leaf nodes' are left, that is nodes containing only zero or one particles. This image of a root layer branching out until leaves are reached is the reason why the algorithm is likened to a tree. To calculate the gravitational force on each particle, then, a so-called 'tree walk' is performed. Starting with some cell on the root level, the distance to the other cells or nodes is determined. If the distance is small, the

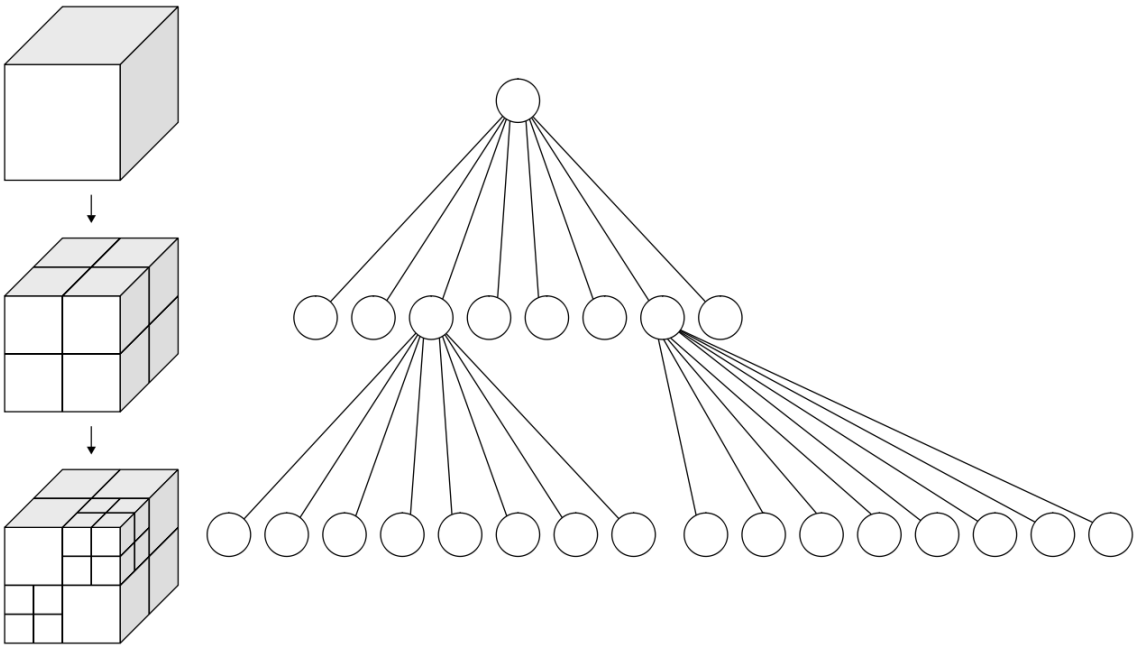


Figure 3.1.: Schematic idea of a hierarchical tree decomposition[16].

node is opened (and further subnodes, too, if necessary), and the force contributions from the individual leaf nodes are computed. If the distance is large enough, however, the exact distribution of the leaf nodes does not matter. Instead, the contribution of all leaf nodes can be approximated by a point mass equal to the sum of the particle masses of those leaves sitting at the centre of mass of the node.
GADGET-4 now combines these N-body methods with an SPH treatment of gas. The interactions between gas particles effectively lead to another force and some pressure acting on them. This can be readily accommodated, though, by considering that this force just causes additional acceleration. Similar to before, this can hence be described by another velocity kick performed at the same time as the gravitational one, so there will simply be another term in equation 3.12. This is also our approach: PBH particles will receive additional velocity kicks according to their scatter interactions. Because of the similarity with the SPH treatment, we use this already existing structure in the code to implement SIDM.

3.3.3. Understanding the Code

Before we begin to explain how we implemented the scattering interactions, we need to understand some of the structure of GADGET-4. It would be beyond the scope of this work, though, to provide a comprehensive explanation, we will rather focus on the parts that are necessary to understand why we chose our implementation like that.
In the main directory of GADGET-4, there are four subfolders: `buildsystem`, `documentation`, `examples`, and `src`. The first one is necessary for the compilation of the code, but we do not need to change anything there. As the name suggests, the second one contains some documentation and we will need to make adaptations there once we introduce simulation parameters because in order to keep the code more understandable, the authors implemented the feature that every simulation parameter must be explained there. From `examples`, we mainly focus on one specific example, namely `DM-L50-N128`[17], to test our algorithm. Therefore, we have to configure only this example according to our ideas. Our main work, however, will go into the source code `src`, which is where the scattering interactions are implemented.
GADGET-4 supports up to six different particle types, of which only the first two are of interest to us: particles of type zero are gas particles (and hence collisional) and particles of type one are the usual collisionless dark

[16] https://commons.wikimedia.org/wiki/File:Octree2.svg
[17] https://wwwmpa.mpa-garching.mpg.de/gadget4/10_examples/#cosmological-dm-only-simulation-with-ic-creation

matter particles. Both of them share the same gravitational force computation, but for the computation of the hydrodynamic aspects of the gas particles, `src` includes the subdirectory `sph`. It consists of two so-called header files, indicated by the ending `.h`, and four code files ending on `.cc`. These code files contain the main functions of the computations, while the headers hold the definitions of some auxiliary functions, several individual variables, collections of variables called `struct`, and combinations of all of the above called `class`. In this case, the general structure for the hydrodynamic computations can be found in `sph.h`, whereas `kernel.h` comprises the functions to compute the hydrodynamic kernel (equation 2.8). From the code files, we are especially interested in `hydra.cc`, which contains the computation of the velocity changes caused by the collisions. That is where we implement our velocity changes, too.
Conveniently, GADGET-4's main function can be found in `src/main/main.cc`. Apart from beginning the simulation and initialising communication between processors, this function also calls the function `run()` in line 327 (note that all lines are given according to our version of the code[18] as of the writing of this thesis), which is defined in line 49 of `src/main/run.cc` and contains the main simulation loop, i.e. the repetition of calculations for progressing time steps. Line 146 there uses the function `do_hydro_step_second_half()` to compute the second half of the hydrodynamic interactions, whereas the first half is completed in line 197 with `do_hydro_step_first_half()`. This apparent confusion can be resolved by considering the exact time integration scheme that `GADGET` uses: the positions are drifted from one full time step to the next, whereas the velocity kicks are split in half for their application. First, a kick is performed to get the velocity to the value it reaches at half the time step, then the full drift is applied before another kick updates the velocity to the value it has at the end of the time step. This scheme is also called 'kick-drift-kick' or KDK integration. Changing perspective a bit, we can also think of the velocity kick as a complete operation, starting at half the time step and finishing at half of the next time step. From this perspective, the kick from the full time step to the half time step, with which the simulation starts, can be called the second half. And since the hydrodynamic interaction only leads to additional velocity kicks (as far as we are concerned), we can even say it is the second half of the hydrodynamic time step.
Both half step functions are defined in `src/time_integration/kicks.cc`, in lines 449 and 469. While the first half function also has an organisational task, the main job for both of them is to call the function `hydro_force()`, defined in the same file in line 480. This is the function that organises the calculation of the hydrodynamic force and applies the corresponding velocity kicks. It also oversees the change of entropy due to shock heating, but we are not interested in that. The only parameter going into this function is a `step_indicator` because it works slightly different during the first and second half time step. However, the first thing is always the same: it creates the `targetlist`, which will contain all active particles or 'targets' in the form of their index in the overall list of particles `P`. Enough memory is reserved so that this list could hold all gas particles, although this might not be needed in practice. Next, the `struct old_hydro_accel` and the variable `Nhydroforces` are defined and set to zero. The former is assigned the current value of `HydroAccel`, which is the variable holding the acceleration due to the hydrodynamic force, if the second half step is performed. For both steps, the `targetlist` now gets filled with active particles, while `Nhydroforces` is increased by one for each entry. It therefore counts the number of targets. Afterwards, the code usually starts the hydrodynamic force calculation, but we need to compute a variable in preparation for that.
Owing to the dependence of our scatter probability on the fourth power of the relative velocity, very small relative velocities could easily boost the probabilities above one, so that these particles would be guaranteed to scatter, independent of their actual position. Relative velocities close to or even at zero might arise from initial conditions such as for a plane wave (which we use for testing, see section 3.4.1) or for single stream particles (i.e. particles with the same velocity). To avoid all particles with vanishing relative velocity v_{rel} from always scattering, we employ the simple condition that they have to be able to cover the distance d to the nearest neighbour during the current time step $\mathrm{d}t_{\mathrm{drift}}$, keeping in mind that the time step requires an extra integration. In mathematical terms, this condition reads $\frac{d}{\mathrm{d}t_{\mathrm{drift}} v_{\mathrm{rel}}} > 1000a$ for a pair to be scattering, where a is the scale factor and

$$\mathrm{d}t_{\mathrm{drift}} = \int_{a_0}^{a_1} \frac{\mathrm{d}a}{a^3 H(a)}, \tag{3.19}$$

with a_0 and a_1 being the start and the end of the time step, respectively.
We have to consider this drift factor because in cosmological simulations, the particles drift for a time $\mathrm{d}t_{\mathrm{drift}}$ rather than $\mathrm{d}t$. The different time steps are computed in `src/time_integration/driftfac.cc` and are not exported to `src/sph/hydra.cc`, so we choose to compute $\mathrm{d}t_{\mathrm{drift}}$ in line 536 of

[18] https://github.com/glatterf42/PBH_EFD

src/time_integration/kicks.cc and only store it as dist_over_time, which is a property we added to the sph_particle_data_hydrocore structure (see line 50 of src/data/sph_particle_data.h). In line 1151 of src/sph/hydra.cc, we then take their distance, divide by their relative velocity, and multiply with the minimum of both particles' dist_over_time since it is only necessary for the particle with the higher time step to reach the other one for them to interact. After that, the result is compared to $1000a$ (line 1153) and the pair is skipped if appropriate. This value of $1000a$ has been determined through testing with a plane wave (section 3.4.1). At first, we let the simulation run without this criterion and tracked the usual amount of scatter events over time. Afterwards, we found a criterion value that would approximately reproduce these numbers while still prohibiting pairs of zero relative velocity to scatter. So note that this value is somewhat arbitrary and it might not be the best way to circumvent this problem.
Immediately after the calculation of the drift time step in src/time_integration/kicks.cc, line 552 sends us back to src/sph/hydra.cc because its core function hydro_forces_determine() is called with the arguments Nhydroforces and targetlist.

Understanding hydra.cc

hydro_forces_determine() is defined in line 284 of hydra.cc. It is already provided with the variables ntarget and targetlist, which represent the number of active particles or 'targets' and a list with their identifiers (see also section 3.3.4). At the beginning, several variables are initialised. e.g. the struct Ngbhydrodat holding all hydrodynamic data necessary to deal with the interaction partners or 'neighbours' of a target: the SphCore data, its position, mass, ID, and the hydrodynamic time bin. The SphCore, in turn, is another struct containing all hydrodynamic variables that are required to compute interactions with passive particles.
After the definitions of some of our variables (see section 3.3.4), the WorkStack is created in line 360, containing information on all possible targets. For each member of the target list, a new entry in the WorkStack is filled and NumOnWorkStack is increased by one, thus simply counting nodes and particles. Simultaneously, clear_hydro_result() (from line 1286) sets each target's HydroAccel to zero. Then, NumOnWorkStack is continuously reduced with a loop. For each node in the WorkStack, which could hold other nodes or zero or one particle, the pinfo structure (from line 45 of src/sph/sph.h) is invoked to create the variable pdat, which contains the target, the number of neighbours as well as the position of the particle (also called 'search center' because this is where the code will start looking for neighbours), a search range, and the smoothing length h. The smoothing length is determined by the criterion that the product of the local density, the cube of the smoothing length and the inverse of the particle mass should be constant. This is applied in the function density() from line 434 of src/sph/density.cc, which is always called directly before do_hydro_step_second_half() in src/main/run.cc.
Initially, the number of neighbours is set to zero during this step. Then, due to the way the WorkStack is created, each entry is considered as being processed for the first time, such that sph_hydro_interact() from line 235 is called with the option NODE_TYPE_LOCAL_NODE. Since the entry is viewed as a local node, the code first finds a node from which we can start the tree walk based on the processor or 'task' that is calling the function. Next, sph_hydro_evaluate_particle_node_opening_criterion() from line 43 will check if this starting node (also called nop) should be opened, for which it uses the search center and the maximum of the smoothing lengths of starting node and target node. If the starting node lies within a ball of radius corresponding to this maximum smoothing length, the node is flagged as NODE_OPEN. Otherwise, it is discarded. If the node needs to be opened, but cannot be opened locally (i.e. on this processor) because it is stored on another task (making it 'foreign' or 'fetched'), tree_add_to_fetch_stack() and tree_add_to_work_stack() are called. They are defined in lines 207 and 232 of src/tree/tree.h and increase the variables NumOnFetchStack and NewOnWorkStack, respectively. Both of these variables are added to the corresponding stack such that the node will be imported from another processor and another try to evaluate it can begin. tree_add_to_work_stack() is also called if the node could be opened locally, but the current buffer space is insufficient. If, however, the node can and should be opened locally, sph_hydro_open_node() from line 168 is called.
This function contains the loop that fills Ngbhydrodat, which is crucial for the hydrodynamic computations that follow. The condition that has to be met for the loop to stop running is rather complicated as it dives deep into how GADGET-4 stores particles across multiple processors. For our purpose, it is sufficient to know that the search for neighbours will be stopped when appropriate. The main task of the loop is to

NODE_TYPE_LOCAL_PARTICLE	0
NODE_TYPE_TREEPOINT_PARTICLE	1
NODE_TYPE_FETCHED_PARTICLE	2
NODE_TYPE_LOCAL_NODE	3
NODE_TYPE_FETCHED_NODE	4

Table 3.1.: Translation of node types to integers.

call `sph_hydro_interact()` again for every node encountered during the tree walk, but with the correct node type this time. There are five node types that are translated into integers according to table 3.1. The rest of the sorting in the loop is to determine if the loop should be continued. For the first and third case, `sph_hydro_interact()` will just call `sph_hydro_check_particle_particle_interaction()` from line 89, whereas for the last two cases, it will repeat what we have described above: redefine `nop`, check if this node should be opened, and call `sph_hydro_open_node()` if that is the case. Otherwise, it will add the node to either the `WorkStack` or the `FetchStack` just as before. The second case is a relic, those particles should not be encountered here, or else the whole code terminates.
Finally, `sph_hydro_check_particle_particle_interaction()` begins with drifting the particle to the current time (note: `sph_hydro_evaluate_particle_node_opening_criterion()` does the same for nodes). This time, the comparative distance measure is chosen as the maximum of the smoothing lengths of the particle from `pdat` and the one of the local or fetched particle. The actual distance is also computed (squared as `rad2`) and both numbers are compared. If the distance is bigger than the chosen distance measure or `rad2`$= 0$, there is no interaction. Furthermore, if the particle already has too many interacting neighbours (as defined in line 18 of `src/sph/sph.h`), this interaction is also discarded. If these checks are cleared, the interaction particle information is stored to one entry of `Ngbhydrodat`, and the number of neighbours is increased by one. Afterwards, this process is repeated for each node found by `sph_hydro_open_node()` until the desired number of neighbours, which is a simulation parameter, is reached.
Back in `hydro_forces_determine()`, we might also encounter the case that a node is not being processed for the first time. If that is the case and the node can be opened locally, `sph_hydro_open_node()` and its whole loop will be started directly and `Ngbhydrodat` will be filled like before.
Once we have `Ngbhydrodat`, the remaining part is pretty straightforward. Line 446 calls `hydro_evaluate_kernel()`, of which there are two versions. One is enabled if the configuration option `EXPLICIT_VECTORIZATION` is selected (line 629), but we do not choose it, so our version is defined in line 905. Both versions perform the same action, though, namely they compute the hydrodynamic interaction rate of each target particle with all its neighbours and the consequential velocity changes, i.e. accelerations. These are saved in each particle's `HydroAccel` variable. We skip the exact working procedure of this function here because we are going to replace it with our own scatter interaction calculation.
After `HydroAccel` is updated, `hydro_evaluate_kernel()` ends, so we can return to the loop that is working through the `WorkStack`. The rest of the loop starts by measuring the performance for diagnostic output via the time the hydro tree walk needed. It then gathers the foreign nodes that could not be opened before and were saved to the `FetchStack`, followed by shifting the `WorkStack` such that the new nodes are also included for the next turn of the loop. This is continued until there are no more entries (nodes) on the `WorkStack`, which ends the loop. The last part of `hydro_evaluate_kernel()` is mainly used for measuring performance metrics and organising the next run of the function by releasing memory space reserved for all the variables that are no longer needed. That is why we return to `src/time_integration/kicks.cc` to see how the velocity kicks are applied.

Applying the Hydrodynamic Kicks

Now that the hydrodynamic accelerations for each particle are computed, we only need the time steps to apply the velocity kicks. For both things, one loop over all active particles is started in line 562. As explained in section 3.3.2, the correct time steps require an additional integration in comoving coordinates. While equation 3.18 is accurate for kicks due to gravity, the time steps stemming from the hydrodynamic force follow a different law,

$$\mathrm{d}t_{\mathrm{hydrokick}} = \int_{a_0}^{a_1} \frac{\mathrm{d}a}{aH(a)a^{3(\Gamma-1)}}, \tag{3.20}$$

with the adiabatic index Γ, which can be set as a configuration option. However, since we did not set it specifically and did not choose an isothermal equation of state for the SPH treatment either, line 99 of `src/data/constants.h` defines it as $5/3$. This results in the hydrodynamic time step being equal to the drift time step. For $\Gamma = 4/3$, it would instead be equal to the time step for gravitational kicks. Once the time step is computed, the velocity update is calculated by adding `HydroAccel` times $\mathrm{d}t_{\mathrm{hydrokick}}$ to the velocity of each particle (lines 598-600). If we are on the second half time step, the code also updates the velocity predicted for the end of the full time step (meaning the end of the first hydrodynamic half time step), `VelPred`, which can be used to compute the hydrodynamic acceleration at that time (lines 624-626). The update is performed by adding the difference of new and old `HydroAccel` times $\mathrm{d}t_{\mathrm{hydrokick}}$. Lastly, the function releases the memory reserved for the old accelerations and the `targetlist`, which finishes the computation of the hydrodynamic kicks. However, there is one last thing we need to understand for our implementation of the scatter events: the units used by GADGET-4.

Concerning the Units

There is no mention in the documentation paper and on the website of how to include physical variables such as the scatter cross section in the code, but thankfully, we can learn from Robertson 2017 how GADGET handles units. It defines a set of internal unit values for the mass, velocity, and length such that the corresponding variables are dimensionless in the code. Further variables like the time are then derived from this set as needed. For the basis set, the relationships between physical values and the dimensionless program values (indicated by a hat) are:

$$m = U_{\mathrm{M}}\ h^{-1}\ \hat{m} \tag{3.21}$$

$$v = U_{\mathrm{V}}\ a^{-1}\ \hat{v} \tag{3.22}$$

$$l = U_{\mathrm{L}}\ a\ h^{-1}\ \hat{l}, \tag{3.23}$$

with U_{i} being the unit measure used by GADGET-4, a the scale factor, and h the dimensionless Hubble parameter from $H_0 = h100\,\mathrm{km\ s^{-1}Mpc^{-1}}$. The cgs-values for the unit measure can be set as simulation parameters, but we do not change them so they are equal to their default values:

$$U_{\mathrm{M}} = 10^{10}\,\mathrm{M}_\odot = 1.989 \cdot 10^{43}\,\mathrm{g} = \hat{U}_{\mathrm{M}}\ \mathrm{g} \tag{3.24}$$

$$U_{\mathrm{V}} = 1\,\mathrm{km\ s^{-1}} = 10^{5}\,\mathrm{cm\ s^{-1}} = \hat{U}_{\mathrm{V}}\ \mathrm{cm\ s^{-1}} \tag{3.25}$$

$$U_{\mathrm{L}} = 1\,\mathrm{Mpc} = 3.085\,678 \cdot 10^{24}\,\mathrm{cm} = \hat{U}_{\mathrm{L}}\ \mathrm{cm}. \tag{3.26}$$

For the unit time, we can then derive

$$U_{\mathrm{T}} = \frac{U_{\mathrm{V}}}{U_{\mathrm{L}}} \approx 3.086\,\mathrm{s} \approx 0.978\,\mathrm{Tyr}, \tag{3.27}$$

which is larger than the age of the universe. Note that ibid. use a unit length of 1 kpc, as is also suggested on the documentation website[19], but since the exact unit does not really matter, we keep the predefined value of 1 Mpc.
Again, we can find a description of how to convert one of GADGET's time steps to a physical one in ibid. This begins by noting that GADGET-4 saves time on an integer time line which is created from the start and end of the simulation, a_{start} and a_{end}, by splitting their logarithmic range into `TIMEBASE` intervals. The logarithm is used because $\log(a)$ is the time variable rather than the scale factor a itself. For our simulations, `TIMEBASE` is defined in line 325 of `src/data/constants.h` as 2^{29}. Consequently, the interval size on the integer time line `Timebase_interval` is given by

$$\mathrm{Timebase\_interval} = \frac{\log(a_{\mathrm{end}}/a_{\mathrm{start}})}{\mathrm{TIMEBASE}}, \tag{3.28}$$

as computed in line 271 of `src/main/init.cc`. An integer time step `ti_step` between e.g. a_0 and a_1 is then given by

$$\mathrm{ti\_step} = \frac{\log(a_1/a_0)}{\mathrm{Timebase\_interval}} = \frac{\mathrm{d}\log(a)}{\mathrm{Timebase\_interval}}. \tag{3.29}$$

[19] https://wwwmpa.mpa-garching.mpg.de/gadget4/05_parameterfile/#system-of-units

From equation 3.14, we see that

$$\mathrm{d}t = \frac{\mathrm{d}\log(a)}{H(a)}, \tag{3.30}$$

and the Hubble parameter is calculated in line 39 of `src/time_integration/driftfac.h` by

$$\text{hubble\_function}(a) = h \cdot \hat{H}_0 \sqrt{\frac{\Omega_0}{a^3} + \Omega_\Lambda + (1 - \Omega_0 - \Omega_\Lambda) a^{-2}}\; U_\mathrm{T}^{-1} = h \cdot \hat{H}_0 \; E(a) \; U_\mathrm{T}^{-1} \tag{3.31}$$

where Ω_0 is the matter density relative to the value today, Ω_Λ is the vacuum (or dark) energy density relative to the value today, and U_T enters the equation to adapt the physical value of H_0 to the internal units. Combining all of the above, we find

$$\mathrm{d}t = \frac{\mathrm{d}\log(a)}{h \; U_\mathrm{T}^{-1} \; \hat{H}_0 \; E(a)} = \frac{\text{Timebase\_interval} \cdot \text{ti\_step}}{h \; U_\mathrm{T}^{-1} \; \hat{H}_0 \; E(a)}. \tag{3.32}$$

Finally, we consider what these units mean for our equations. Since the velocity change caused by a scatter event only requires knowledge of the old velocities and the particle masses, we do not have to adapt anything there; we can simply use the internal values to compute a velocity kick in internal units. However, we do introduce physical values for the scattering cross section and the mean PBH mass in the computation of the scattering probability. As mentioned in section 2.4.2, the scattering cross section also depends on the relative velocity, which we want to evaluate for every pair in our algorithm. Therefore, we can only introduce a value of the form $v_\mathrm{rel}^4 \; (\sigma_\mathrm{sca}/m)$ into the code, so we also have to correct the units accordingly. Writing the scatter density as $m_\mathrm{p} \; W$ (equation 2.15) and the cross section term as $v_\mathrm{rel}^4 \; (\sigma_\mathrm{sca}/m)_0$, we can transform every term in the scatter probability except for the cross section term to see which transformation arises:

$$P = v_\mathrm{rel} \; m_\mathrm{p} \; W \; (\sigma_\mathrm{sca}/m)_0 \; v_\mathrm{rel}^{-4} \; \mathrm{d}t \tag{3.33}$$

$$= U_\mathrm{M} \; h^{-1} \; \hat{m}_\mathrm{p} \; U_\mathrm{L}^{-3} \; a^{-3} \; h^3 \; \hat{W} \; (\sigma_\mathrm{sca}/m)_0 \; U_\mathrm{V}^{-3} \; a^3 \; \hat{v}_\mathrm{rel}^{-3} \; U_\mathrm{T} \underbrace{\frac{\text{Timebase\_interval} \cdot \text{ti\_step}}{h \; \hat{H}_0 \; E(a)}}_{:= \text{dt\_phys}} \tag{3.34}$$

$$= \frac{U_\mathrm{M}}{U_\mathrm{L}^2 \; U_\mathrm{V}^4} \; h^2 \; \hat{m}_\mathrm{p} \; \hat{W} \; \hat{v}_\mathrm{rel}^{-3} \; (\sigma_\mathrm{sca}/m)_0 \; \text{dt\_phys} \tag{3.35}$$

By giving v_rel^4 in kilometers per second, we can drop U_V because its numerical value is 1. We then see that it is convenient to also give $(\sigma_\mathrm{sca}/m)_0$ in `GADGET`'s internal units because this lets us avoid U_M and U_L. However, these units still include factors of h as detailed in equations 3.21 and 3.23, which we must not forget. The only factor that is left to be considered in our algorithm is the square of the dimensionless Hubble parameter, which is easily accessible via `All.HubbleParam`. Now, we are ready to turn to our implementation of the scatter interactions.

3.3.4. Including Our Algorithm

Our additions to the code are quite unique in that not a lot of research is performed with PBHs being treated as an effective fluid. Therefore, we want our algorithm to only be working when desired, meaning we want to have some switch to enable it at any time. This option is provided by the configuration parameters that can be selected during the preparation of each simulation run. A comprehensive list of these parameters is given in `Template-Config.sh` and each one is explained in `documentation/04_config-options.md`. In order to add another configuration option, we thus have to add it to both files[20]. Because of our model description of PBHs, we call the configuration option `PBH_EFD` and add it in the section for extra physics in line 93 of `Template-Config.sh`. Consequently, we also add a description of `PBH_EFD` in the extra physics section of the documentation file following the syntax of the other explanations there. Note that for the code to accept the new option, it also has to be used somewhere in the source code. In our case, we include all changes to the code in a condition of the form `#ifdef` or `#ifndef PBH_EFD ... #endif` such that they are only working if `PBH_EFD` is (or is not) selected in the `Config.sh` of the simulation.

[20] See https://wwwmpa.mpa-garching.mpg.de/gadget4/11_migration/#adding-a-config-option for the official guidelines.

On a similar level, we want to define $(\sigma_{\text{sca}}/m)_0$ as a simulation parameter that can be set via the parameter file (per default called `param.txt`) because it is constant for all times and particles. In addition, we also introduce the mean basis value of the cross section $\sigma_{\text{sca},0}$ and the average PBH mass m separately. These three parameters are called `SigmaOverM`, `MeanPBHScatteringCrossSection`, and `AveragePBHMass`, respectively. To define our new constant parameters, we first head to `src/data/allvars.h` and look for a suitable spot in the structure `global_data_all_processes`. There, we define the new parameters of type `double` following the other examples, including a brief description (lines 147-149). Next, we need `allvars.cc` from the same subdirectory as it includes the function `register_parameters()` which needs to register our parameters. Therefore, we add a function call of `add_param()` in the same way as for already existing parameters (lines 72-74). By including this call within `#ifdef PBH_EFD ... #endif`, we ensure that these parameters are only available with our algorithm enabled. Finally, we have to document these parameters in `gadget4/documentation/05_parameterfile.md` (lines 383-406). For this, we can again follow the already existing examples. As exemplary values, we provide the values we derived from the extended mass function from B. Carr, Clesse et al. 2020 in `GADGET-4`'s internal units, namely a mean scattering cross section of $1.73189 \cdot 10^{-13}\ U_{\text{L}}^2\ U_{\text{V}}^4$, an average mass of $3.086978 \cdot 10^{-10}\ U_{\text{M}}$, and their quotient as $5.6103 \cdot 10^{-4}\ U_{\text{L}}^2\ U_{\text{V}}^4\ U_{\text{M}}^{-1}$. Note that for both values including the scattering cross section, we provide the basis value that still needs to factor in the relative velocity. Furthermore, the first value still lacks a factor of h^2, and the second and third a factor h to be entirely in internal units.

In section 3.3.3, we have already explained our first addition to the source code in `src/time_integration/kicks.cc`. Of course, this change is surrounded by `#ifdef PBH_EFD ... #endif` as well. Following the progress of the code as outlined in the same section, we now focus on our changes to `src/sph/hydra.cc`.

The very first addition in the beginning of `hydro_forces_determine()` is some diagnostic output that helps us to better understand how our code is working in practice (lines 293-295). This includes e.g. the number of particles stored on the current task, the total amount of particles on all tasks, and the number of active hydrodynamic particles. Adopting the idea of defining variables for different functions in the beginning of the main function, we also add several variables only available for out algorithm there (lines 336-351). The most important one is the `scatter_list`, which we will need to evaluate the scatterings (see section 3.3.4). For this list, we reserve enough memory to store two scatter events for each particle (see section 3.5 for why this should be enough). Most of the other variables are used to understand the progressing of the code and if our algorithm works as intended. These are `nscatterevents`, `pairsconsidered`, `numberofparticles`, `numberoflocalparticles`, `numberofforeignparticles`, `novrelbefore`, `novrelafter`, `nsimilarpairs`, and `ndistinctparticles`. All of them are set to zero and count the things suggested in their names, so `nscatterevents` counts the number of scatter events, whereas `pairsconsidered` tells us for how many pairs random numbers have been drawn to see if they scatter. The next three are almost self-explanatory, they count the local numbers they are named after, i.e. the numbers on the current task. `novrelbefore` and `novrelafter` are used to assess how well our criterion for avoiding pairs with zero relative velocity works. Next, it is clear from equation 3.1 that we usually have to compute the mean of the scatter probabilities of i scattering with j and j scattering with i. If, however, these scatter probabilities are the same, we can be more efficient by not computing the mean. This is the case if the time step, mass, and smoothing length are the same for both particles. Therefore, we implement a test to check this condition and count the number of times we encounter such similar pairs. `ndistinctparticles` will help us to apply the acceleration updates efficiently and properly. The two other active variables are `ti_step_to_phys` and `scatter_prob_to_phys`, with which we convert the integer time step `ti_step` and the scatter probability `scatter_prob` to internal units. As described above, we calculate `ti_step_to_phys` as $1/(h\ H_0\ E(a)\ U_{\text{T}}^{-1})$, followed by `scatter_prob_to_phys` as h^2. For velocity-independent scatter cross sections, the conversion factor needs to be h^2/a^4 instead, which we also compute here. Furthermore, there are the inactive `max_density` and `ndensitylimitapplied`. They can be used to impose a maximum scattering density (ρ_{ij} from equation 2.15) for each event and count how many times this limit is applied. The idea for a density limit arose from a misguided solution attempt to a problem with the code, and we keep it because it might be interesting for some diagnostic output. Finally, we introduced a test to check that our conversion of internal to physical time steps works, for which we added the simulation parameter `Physical_Time` analogous to the other ones to give the starting time. Line 351 adds the current time step to the starting time and line 352 then prints the current time of the simulation. In order to be able to use all these variables throughout `src/sph/hydra.cc`, we also have to add them to the `sph` class in `src/sph/sph.h`. Lines 129 to 141 there hold the single digit variables. For the `scatter_list`, however, we define the `struct scatter_event` (line 114). All such events consist of the

scattering probability and the two partners, for which we define the `struct scatter_partner` in line 105. Each `scatter_partner` holds the ID, mass, time step, (predicted) velocity, and the scatter-induced velocity change `scatter_delta_vel` of a particle. In line 12, we then define each entry of the `scatter_list` to be a `scatter_event`.
Within the tree walk, we change almost nothing. The only additions are made to the function `sph_hydro_check_particle_particle_interaction()`, where `Ngbhydrodat` is filled. Since we want each `scatter_partner` to have an ID (so that we can later assign the correct velocity update to each particle), we need to store it in this list. For that purpose, we first have to change the underlying structure of that list, namely `ngbdata_hydro` from line 90 of `src/sph/sph.h` such that an ID can be stored in every entry. This is done by simply adding the `unsigned int ID` to the structure in line 98. Back in `src/sph/hydra.cc`, it is straightforward to access the ID of a local particle and add it to `Ngbhydrodat` (line 129). For foreign particles, though, the ID is not usually stored in their data structure `foreign_sphpoint_data`. It is defined in line 103 of `src/ngbtree/ngbtree.h` and we simply add the ID in line 110. We still have to fill this variable with data, which is done in line 347 of `src/mpi_utils/shared_mem_handler.cc`. These additions allow us to access and save the ID even for foreign particles, which we do in line 159 of `src/sph/hydra.cc`. Finally, we add some particle counters to the function. For each local particle, we increase `numberoflocalparticles` by one in line 130, while we raise `numberofforeignparticles` for each foreign particle in line 161. Both of these counters are preceded by an addition of one to `numberofparticles` because this should count all the particles. With that, we return to the main loop, where the main inclusion of our algorithm begins.
Rather than leaving the call of `hydro_evaluate_kernel()`, we include it in our `#ifdef PBH_EFD ... #endif` condition such that it is only called if `PBH_EFD` is *not* enabled. If `PBH_EFD` is active, the new function `scatter_evaluate_kernel()` will be called instead. Similar to its hydrodynamic equivalent, this function only takes `pdat` as its argument and is declared in line 158 of `src/sph/sph.h`. It begins its work in line 1091 of `src/sph/hydra.cc`.
Especially in the beginning, this function is quite similar to `hydro_evaluate_kernel()`. The first action in `scatter_evaluate_kernel()` is to retrieve the target particle and all its information necessary for the determination of a scatter event from `pdat`. Next, an instance of the `struct kernel_hydra` (from `src/sph/kernel.h`, line 25) is created, which will hold all data required for and resulting from the evaluation of the smoothing function (equation 2.8). In line 1104, the smoothing length of the target particle is immediately saved as `kernel.h_i`. Afterwards, we compute the time step of the target particle and directly convert it to internal units using `ti_step_to_phys`, which we then store as `dt_i_phys`. Furthermore, we set the variable `scatter_occurrence` to zero for the target particle. This variable was added to the `sph_particle_data` structure in line 79 of `src/data/sph_particle_data.h` so that we can trace which particles do scatter and thus receive a velocity kick.
In line 1113, we then begin a loop over all neighbours. For each neighbour, its information is saved just like for the target particle, the only difference being that the information is reduced because it could be a foreign particle and only a minimum of information is imported for those. Before we start the calculation of the scatter probability, we impose the condition that the ID of the target must be smaller than that of the neighbour. Since we want scattering to be symmetric, we are only interested in unique particle pairs for the evaluation and hence need to avoid double counting, which this condition achieves precisely. With double counting being excluded, we first increase `pairsconsidered` by one in line 1122. Next, the squared distance between the two particles is computed, this time called `r2`, and the smoothing length of the neighbour is saved as `kernel.h_j`. At this point, we check again that the particles are close enough to scatter, i.e. either the square of `kernel.h_i` or `kernel.h_j` has to be bigger than `r2`. If that is the case, the distance (the square root of `r2`) is saved as `kernel.r` and it is examined if that distance is bigger than zero to avoid scattering of a particle with itself.
Then, for different particles and unique pairs, the relative velocity of the particles is computed from the predicted velocities `VelPred` (line 1141-1145). These velocities are used because they are already stored in the `SphCore`, which is also imported for foreign particles. Moreover, these velocities are used in the same way in the original `hydro_evaluate_kernel()`. The relative velocity is saved as `kernel.dv` and we also save $1/v_{\mathrm{rel}}^3$ as `kernel.dvinv3`. Both properties are newly added to the kernel in line 33 of `src/sph/kernel.h`.
Now, we employ the criterion described in section 3.3.3 to avoid particles with zero velocity. First, we raise `n0vrelbefore` by one if the relative velocity is zero. After applying the criterion, we again check if there are pairs with zero velocity, for which we would increase `n0vrelafter` by one each. However, the control output indicates that no pair with zero relative velocity makes it past our criterion.
While we initially only required that one of both particles' smoothing lengths needs to be bigger than the

distance, this condition is now tested in more detail. Only if a particle's smoothing length is larger than the distance of the pair will the smoothing function be evaluated. For the evaluation, the auxiliary function `kernel_hinv()` (defined in line 115 of `src/sph/kernel.h`) first computes different powers of the smoothing length, e.g. h^{-3}. Next, `kernel_main()` (from line 139 of `src/sph/kernel.h`) is called to evaluate the kernel. There are different modes of evaluation defined from line 111 to 113 of `src/sph/kernel.h`, of which we choose `COMPUTE_WK` because we are only interested in the value of the function and not its derivative. Ensuing this kernel evaluation, we compute the scattering densities in lines 1184 and 1185. These are followed by some commented out lines which can be used to measure or set a density limit. The final step before calculating the `scatter_prob` is to determine the time step of the neighbour particle, which we do just like above for the target.
As explained above, we want to increase efficiency by not calculating the mean of the scatter probabilities for similar pairs, i.e. particles with the same mass, time step, and smoothing length. Therefore, if we have a similar pair, we immediately calculate the scatter probability according to equation 3.33. We use the density and time step of the target particle here, but since the pair is similar, we could use the values for the neighbour just as well. Additionally, we increase `nsimilarpairs` by one. If the pair is not similar, however, we calculate the probabilities separately. For i scattering on j, we use the density and time step of j and vice-versa. Afterwards, we save the mean of both probabilities as `scatter_prob`. To ensure that nothing went wrong, we check in line 1229 that the scatter probability is positive. If it is not, the code terminates.
In order to determine the occurrence of a scatter event, we now need to draw a random number. Luckily, GADGET-4 provides a random number generator exactly for that purpose in line 49 of `src/system/system.h`. It uses `gsl_rng`[21], which, according to its documentation, provides truly decorrelated numbers. The employed distribution function `gsl_rng_uniform()` has a default range of $[0, 1)$ and `get_random_number()` thus returns a double precision number from that range. Every element of that range has the same probability of being picked, so we can use this function to draw our random numbers. To enable this usage, though, we have to declare in the preamble of `src/sph/hydra.cc` that we want to use the function. `#include <gsl/gsl_rng.h>` in line 13 and `#include "../system/system.h"` in line 35 achieve just that, so we can draw our random number u in line 1232.
If u is smaller than or equal to `scatter_prob`, we fill an entry in the `scatter_list` by adding both particles as `partner_one` and `partner_two` as well as their `scatter_prob` combined to a `scatter_event`. Finally, we add one to the count of scatter events in line 1250. The evaluation of the `scatter_list` then begins in line 480 once the whole `WorkStack` has been dealt with.

Evaluating the `scatter_list`

Because of the storing of particles on different processors, the evaluation of the `scatter_list` is separated from the application of the velocity updates and the whole procedure is included in `#ifdef PBH_EFD ... #endif` again. We begin by reserving memory for a list holding the acceleration updates, `scatter_accel_update_list`. Each entry of this list has the form of the new `struct scatter_accel_update` defined in line 122 of `src/sph/sph.h`: it contains the ID of a particle and its acceleration update `scatter_delta_a`. For `scatter_accel_update_list`, we reserve memory equivalent to two times the number of scatter events. After that, we evoke the function `scatter_list_evaluate()`, which we declare in line 159 of `src/sph/sph.h` and define in line 1296 here. It receives the `scatter_list` and the number of scatter events as its parameters.
The function immediately checks if there are scatter events at all because if there are no scatter events, there is no list to evaluate and it can already return. If there are scatter events, however, we begin by sorting the scatter list by ascending scatter probability. This can be done via the built-in function `mycxxsort()`, which is defined in line 39 of `src/sort/cxxsort.h`. We can use it just like that because there already is `#include "../sort/cxxsort.h"` in line 32. This function needs the list, the end of the list, and a sorting criterion as its parameters. For the latter, we define two functions in `src/sph/sph.h`: `by_scatter_prob_descending()` (line 161) and `by_scatter_prob_ascending()` (line 165). They take in two values and return `TRUE` if the first value is bigger than the second or the second is bigger than the first, respectively. We decide to use the ascending sorting because of the way in which we evaluate the `scatter_list` now. However, this might not be the ideal way (see section 3.5), so we also keep the descending option available.

[21] http://www.gnu.org/software/gsl/doc/html/rng.html

To evaluate the `scatter_list`, we start a loop over all scatter events. For each event, we load the data of both scatter partners in the lines 1311 and 1312. Next, we compute the relative velocity, the sum of the masses and the velocity of the centre of mass as given by equation 3.3. Then, we choose a random direction via sphere point picking (section 3.2.3) and using the random number generator. Here, we have to `#include "../data/constants.h"` in order to use `M_PI` as π (defined in line 56 there). The additional default inclusion of `<math.h>` allows us to utilize `acos()`, `cos()`, and `sin()` directly. From line 1335 to line 1341, we then calculate the velocities after the scattering event as described in equation 3.6.
At this point, we could calculate the difference between new and old velocity and save that as our velocity kicks. However, as Robertson 2017 reminds us, we cannot allow a particle to scatter more than once with the same initial velocity during any one time step because this would break energy conservation. Therefore, we need to apply the velocity kick immediately after each scatter interaction. If the particle already scattered before during the same time step, we have to consider this change. For this purpose, we begin a loop over all scatter events that came before the current one and compare the IDs of all former scatter partners to the two current ones. If a particle is found in the list, we save its index and the partner type, which can be either one or two, and overwrite these findings should the particle be found again. This will leave us with the data of the last scatter event the particle took part in. If a particle has not scattered before, we simply set `scatter_delta_vel` -a property of the `scatter_partner`- as the difference between the new and the old velocity. But if it has scattered before, we use our knowledge of the partner type to set `scatter_delta_vel` to the difference between the new velocity and the one resulting from the last scatter event. After the completion of this loop, we start another loop over all scatter events in line 1418 to fill the `scatter_accel_update_list`. As it is empty at first, the loop will first evaluate its second part, which fills entries of the list if the particle is not already in the list (lines 1442-1459). To do this properly, we employ the variable `ndistinctparticles`. If a particle is not already in the list (which is indicated by the flag values `found_part_one` and `found_part_two`), the list spot with index `ndistinctparticles` will be filled with its ID and `scatter_delta_vel`, after which `ndistinctparticles` is increased by one (lines 1449 and 1458). As the list begins to fill, it might happen that a particle has already received an entry in the list. This is checked in the first part of the loop every time (lines 1425-1441). If the particle is already in the list, we do not need to change its ID, but we have to update its velocity change according to the new scatter event. Note that we truly update the value using `+=` instead of simply setting it to the new value. We also set the flag values to one such that the second part of this loop will be ignored.
Since the velocity update evaluation is quite technical, it might be beneficial to also note it in more mathematical terms. For the first scatter event of a particle occurring in the `scatter_list`, we set the change of velocity $\Delta\vec{v}$ equal to the difference of the velocity after scattering $\vec{v}'$ and before $\vec{v}$,

$$\Delta\vec{v} \;=\; \vec{v}' \;-\; \vec{v}. \tag{3.36}$$

This value is not saved for the particle, but rather for the `scatter_partner` object in the `scatter_list`. For subsequent scatter events of that particle, we instead set the corresponding `scatter_partner`'s `scatter_delta_vel` to the difference between $\vec{v}'$ and the new velocity $\vec{v}''$,

$$\Delta\vec{v}' \;=\; \vec{v}'' \;-\; \vec{v}', \tag{3.37}$$

and so on for further scatter events. After all events have been evaluated that way, we go through them in the same order again to fill the `scatter_accel_update_list`. If a particle is not already in that list, we create a new entry (so here, each entry corresponds to a distinct particle) and set its variable `scatter_delta_a` $\Delta\vec{a}$ to the computed velocity change,

$$\Delta\vec{a} \;=\; \Delta\vec{v}. \tag{3.38}$$

If, however, the particle already has an entry in this list, we simply update $\Delta\vec{a}$ with the new change:

$$\Delta\vec{a} \;+=\; \Delta\vec{v}' \tag{3.39}$$

and so on for further events. This concludes the evaluation of the `scatter_list`.
What is left to do now is to apply the computed velocity changes to each particle. Unfortunately, we cannot do this directly because the corresponding variable `HydroAccel` is only stored on each particle's local processor, whereas scatter partners might come from different cores. That is why we have to exchange data between all tasks using `MPI`[22] now.

[22] https://www.open-mpi.org/

Communication Between Processors

Luckily for us, GADGET-4 uses MPI per default to communicate between processors. There is even some of this communication going on in src/sph/hydra.cc already, so we do not have to change the preamble and can use all functions declared in src/mpi_utils/mpi_utils.h. In our case, we want to exchange the scatter_accel_update_lists between all tasks. For this purpose, we first need to count the total amount of acceleration updates to be applied or, equivalently, the total number of ndistinctparticles. In line 489, we create the variable ndistinctparticles_total and set it to zero. Then, we call the function myMPI_Allreduce(), which is defined in line 24 of src/mpi_utils/allreduce_debugcheck.cc. It is essentially the same as the regular MPI function MPI_Allreduce()[23] but it performs an additional check that each task is indeed dealing with the same number of values. The function generally requires the address of the object it should share, the address of the result (which is why we defined ndistinctparticles_total beforehand) , the number of values it has to consider, the data type of those values, the operation it should perform, and the communicator on which it should work. A communicator is basically a collection of processes that can communicate with each other. For the data and operation types, there are specific names that can be used listed in the official documentation[24]. For example, MPI_MAX returns the maximum of all processes, whereas MPI_MIN returns the minimum. To get the sum of some variable stored on all processors, we have to use MPI_SUM. Our data type here is an integer, so we employ MPI_INT, and we use the communicator holding all simulation tasks, MPI_COMM_WORLD. In exactly the same way, though without further significance for the application of velocity updates, we calculate the total of nscatterevents and the sum of all similar pairs and save them as nscatterevents_total and nsimilarpairs_total, respectively. Both will be used for diagnostic output. There is also the option of doing this for the number of times the density limit was applied, but it is currently inactive.
In line 521, we want to perform the actual collection of all tasks' scatter_accel_update_lists to one scatter_accel_update_list_global using MPI_Allgatherv()[25]. This function collects data from all processes and then sends the gathered data back to all tasks. It requires a number of parameters, namely the starting address of the data we want to sent, the size of that data and its type, the starting address of where we want to save the collected data, how much data we expect from each task, where we want to place them in our collection, their type, and lastly the communicator. Since we are sending scatter_accel_updates, we cannot choose MPI_INT again, so we select MPI_BYTE for both parameters instead. As the communicator, we again use MPI_COMM_WORLD. Finally, we already know that the size of the data we want to send from each task is ndistinctparticles times sizeof(scatter_accel_update). All other parameters require a bit more work to be done beforehand.
First, we save the number of tasks reserved for communication (Shmem.Sim_NTask) as NTask_here in line 496. Then, we create three lists of integer values, each with a length corresponding to the number of communication tasks: ndistinctparticles_per_task, ndistinctparticles_sizes, and ndistinctparticles_offsets. To measure the number of distinct particles per task, we utilize MPI_Allgather()[26], which works very similar to MPI_Allgatherv() except that it does not have a displacement parameter, i.e. we cannot tell the function where to store the data in our collection variable. This means that we the address we provide for the collection will be filled entry by entry with the data from different tasks. For this collection, we want to send ndistinctparticles, which is one value of type MPI_INT. These values should be collected in ndistinctparticles_per_task and remain one per task and of type MPI_INT, and the communicator is again set to MPI_COMM_WORLD. We then start a loop over all communication tasks to fill ndistinctparticles_sizes (line 503). Each entry should simply be the product of the corresponding entry in ndistinctparticles_per_task (i.e. the ndistinctparticles of that task) times sizeof(scatter_accel_update). Next, we set the first entry of ndistinctparticles_offsets to zero because the data from the first task should start at the beginning of scatter_accel_update_list_global without any offset. All remaining entries are then filled via another loop over all communication tasks, each offset being equal to the offset of the task before plus the size of the data of the task before. In order to test that we have made no mistake here, we run a quick condition that the ndistinctparticles_total times sizeof(scatter_accel_update) should be equal to the sum of the last entries of ndistinctparticles_offsets and ndistinctparticles_sizes (line 512). Finally, we reserve memory

[23] https://docs.microsoft.com/en-us/message-passing-interface/mpi-allreduce-function
[24] https://www.mpich.org/static/docs/latest/www3/Constants.html
[25] https://www.mpich.org/static/docs/latest/www3/MPI_Allgatherv.html
[26] https://www.mpich.org/static/docs/latest/www3/MPI_Allgather.html

space for our global list according to `ndistinctparticles_total` times `sizeof(scatter_accel_update)`. Now, we are ready to call `MPI_Allgatherv()`. In addition to the parameters mentioned above, we provide the `scatter_accel_update_list` as the data we want to send, the `scatter_accel_update_list_global` as the collection variable, `ndistinctparticles_sizes` as how much data we expect from each task, and `ndistinctparticles_offsets` as the displacement parameters. After this function call, each processor now holds a list of all `scatter_accel_update`s everywhere, so we can now update `HydroAccel` with each processor just dealing with the particles that are stored on it.

Applying the Acceleration Updates

The acceleration updates are applied by the function `scatter_accel_update_apply()`, which is declared in line 160 of `src/sph/sph.h` and defined in line 1463 of `src/sph/hydra.cc`. It takes the global list of acceleration updates and the number of acceleration updates as its arguments and checks immediately if there are any acceleration updates at all. If there are no updates to be applied, it returns, only printing the information that it was skipped this time due to a lack of scatter events. Before we can update `HydroAccel` as desired, there is one final problem: the particles, and their `HydroAccel` attribute with them, are saved in lists on each processor, and are usually accessed by their index in those lists. Since we only have the particle IDs, we create the auxiliary function `get_index_from_ID()` in line 1262 (declared in line 169 of `src/sph/sph.h`). This function was designed to take the ID of a particle and another variable to indicate the partner type, i.e. one or two. With this distinction, one could print information about which particle of a pair could not be found in a task's list of particles (lines 1273-1276); however, this functionality was disabled because when each task is looking for all the particles, the resulting output would be overwhelming. Instead, this variable is now simply set to zero when the function is called.
To identify the particle indices, we loop over all active particles stored on the current processor. For each of them, we retrieve the ID and compare it to the one we are looking for. If they match, we save the index as `particle_index`. If the particle cannot be found in the whole list on this task, we instead set `particle_index` to -1. The function then ends by returning `particle_index`. As an aside, you might remember that we chose to save the particle IDs before. At least for the active particles, we could have saved their list indices directly instead. However, we wanted `scatter_accel_update_apply()` to work uniformly for all particles, so we opted for saving the IDs of all particles.
Back in `scatter_accel_update_apply()`, we start a loop over all updates that need to be applied, which corresponds to all particles that require an update. For each particle, we retrieve the list index via `get_index_from_ID()`. If that index is negative, meaning the particle is not stored on this task, we skip to the next update. But if the index is not negative, we use it to access this task's list of particles at the correct spot and set `HydroAccel` to the velocity change we computed earlier. Simply setting the variable here is not overwriting any relevant information because it was set to zero for all active particles and passive particles should not just increase their former value, anyway. Note that our velocity change is still a velocity. While it is technically incorrect to save this as a variable that is used in other places as an acceleration (e.g. lines 279-281 of `src/time_integration/timestep.cc`), it is okay here because our first action after completing this function will be to convert it to an acceleration. Lastly, we set the attribute `scatter_occurrence` of the particle to one.
After `HydroAccel` has been updated according to the scatter interactions, there is not much left to do in `src/sph/hydra.cc`. From line 527 to 531, we release the memory reserved for several lists. Lines 533 to 536 then print information about all the diagnostic variables we introduced and used. Finally, we release the memory reserved for the `scatter_list` in line 544, which concludes our changes to `src/sph/hydra.cc`. For the application of the velocity kicks, we return to `src/time_integration/kicks.cc`, line 555.

Applying the Velocity Kicks

In principle, all velocity kicks are applied through a loop over all active particles. However, we want some additional information about the workings of our algorithm, so we introduce two variables before the loop begins and set them to zero. With `n_updates`, we count the number of velocity kicks that are applied, while we use `mean_vel_update` to calculate the intensity of the average kick. Both variables are of course only active if `PBH_EFD` is enabled. The next part is not changed at all by us, the code is simply calculating the time

step for the kicks, i.e. evaluating equation 3.20. Directly afterwards, we enter the conversion of the velocity updates stored in `HydroAccel` to accelerations, by dividing them by the time step if `scatter_occurrence` of the particle is one. This conversion allows us to essentially keep the rest of the velocity kicks as it is for the usual hydrodynamic kicks. From line 598 to 600, the particle velocities are updated with `HydroAccel` times the time step, and if the second half of a time step is performed, a particle's `VelPred` is updated by the difference of new and old `HydroAccel` times the time step. In between these usual updates, we calculate our diagnostic variables.
From line 603 to 606, we compute the absolute value of the velocity kick. We then check that this value exists by terminating the code if the value is not a number, for which we use the standard `math.h` function `isnan()`[27]. If this check is cleared and the velocity kick is furthermore not zero, we update `mean_vel_update` with it and increase `n_updates` by one. Here, we exclude all particles that did not receive a velocity kick because their `HydroAccel` will still be zero. Theoretically, this also excludes particles whose velocity after scattering was determined to be exactly the same as before. However, since we pick the direction of the velocity at random, the fraction of particles for which this is the case should be negligible. Finally, we print the mean velocity kick as the quotient of `mean_vel_update` and `n_updates` if `n_updates` is not zero (and only if `PBH_EFD` is active). These are all the changes to the code we introduced in order to incorporate our scattering algorithm.

3.4. Testing

3.4.1. Plane Wave Tests

Rather than assuming that the implementation of our algorithm immediately works, we perform a number of tests to check that everything works as intended. Because it is very complex to predict the outcome of a cosmological simulation and there are no previous results quite like the ones we are expecting so that we can not directly compare our results with the literature, we choose the simplest scenario to study our algorithm: a plane wave.
To create the initial conditions of the test simulations, we are using Python[28]. We take a box of length `Lbox`$= 1$, where we already employ `GADGET` units, so $1\,\mathrm{Mpc}/h$. Within this box, we arrange a number of particles on a mesh, with `Nres` $\in \{32, 64, 128\}$ particles in each direction. This means that the particles have a distance of `Lbox`/`Nres` in every direction. The idea of a wave then arises from the velocity distribution of these particles. Every particle begins the simulation with no velocity in either x- or y-direction, but a velocity in z-direction v_z that might be different from zero. v_z only depends on the z coordinate of the particle, so all planes of particles sharing the same z coordinate start with the same velocity. The velocity distribution now follows a sine wave with a wavelength corresponding to `Lbox`, such that particles receive a positive velocity below $z = 0.5$ and a negative velocity above it. That is why the setup is called 'plane wave'. In practice, however, we do not leave our particles on this exact grid with distances of `Lbox`/`Nres` in every direction. Instead, we change the equal spacing by adding 0.1% of each particle's velocity in the corresponding direction to the particle's coordinate value. This effectively means that the particles around $z = 0.5$ are more clustered than particles on the outer edges of the box. On average, though, their distances remain `Lbox`/`Nres`. Figure 3.2 depicts the main point of our starting configuration by plotting v_z against the z coordinates and indicating the absolute value of each particle's velocity through the colour code. A brighter colour represents a higher absolute velocity. The particles in the upper part are PBH particles, whereas the lower half contains standard CDM particles.
In addition to position and velocity, each particle requires an ID and a mass. For the IDs, we simply enumerate all particles from zero to `Nres`$^3 - 1$ and use this number as the particle ID. To compute the mass of a particle m_p, we have to consider that the total mass of the box `Mbox` consists of `Nres`^{3} particles. At the same time, we can write the mass as the product of the matter density ρ_M and the volume of the box `Lbox`^{3}. In turn, we can write $\rho_\mathrm{M} = \Omega_\mathrm{M}\rho_\mathrm{crit}$, where Ω_M is the matter density in terms of the critical density today and ρ_crit is precisely that critical density. It can be derived from the first Friedmann equation assuming that the spatial curvature is zero as $27.75 \cdot 10^{10}\,\mathrm{M}_\odot\mathrm{Mpc}^{-3}h^2$, using `GADGET` units again. For our plane wave scenario, we are

[27] https://www.cplusplus.com/reference/cmath/isnan/
[28] https://www.python.org/

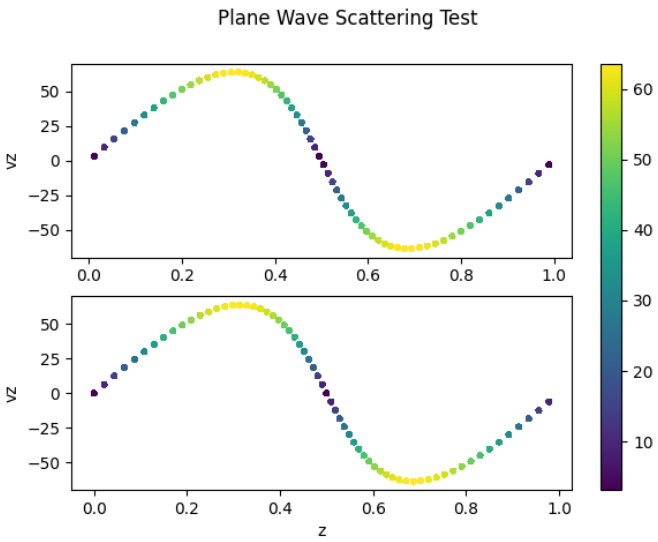


Figure 3.2.: Initial conditions of our plane wave tests in phase space. Plotting v_z versus z coordinates reveals the wave structure. Brighter colours indicate higher absolute velocities (given in internal units). This test was for `Nres` $= 64$ particles in each component, with PBH particles on top and CDM particles below.

assuming for simplicity that all the energy content of the universe is contained in matter, i.e. $\Omega_\mathrm{M} = 1$, but we are creating CDM and PBH particles at the same time so that we can compare both cases from the same simulation. This means that we create two sets of particles as described before, with the positions (and, consequently, the velocities) shifted ever so slightly such that the particles are not beginning at exactly the same points in phase space. Regarding the particle masses, each particle type gets assigned only one half of the total box mass, so we arrive at

$$m_\mathrm{p} \;=\; \frac{1}{2}\,\Omega_\mathrm{M}\,\rho_\mathrm{crit}\,\frac{\mathrm{Lbox}^3}{\mathrm{Nres}^3}. \tag{3.40}$$

After saving the particle masses and IDs, we also save the cosmological parameters used for the compilation in the initial condition file. However, since these parameters are set again via the parameter file, we skip the description here.

A setup like ours leads to so-called shell crossing at a given time. In fact, this time already enters the calculation of the initial conditions because the wave amplitude of the velocity distribution (and, consequently, the particle spacing) are chosen in such a way that shell crossing occurs precisely at the prescribed time a_cross. At a_cross, a big group of particles with positive velocity and another such group with negative velocity reach the coordinate $z = 0.5$, making the simulation 'multi-stream' rather than 'single-stream' because there now exist particles with different velocities at the same z coordinates. This vicinity of particles is why we expect there to be some scatter events at or even slightly before a_cross. See figure 3.3 below.

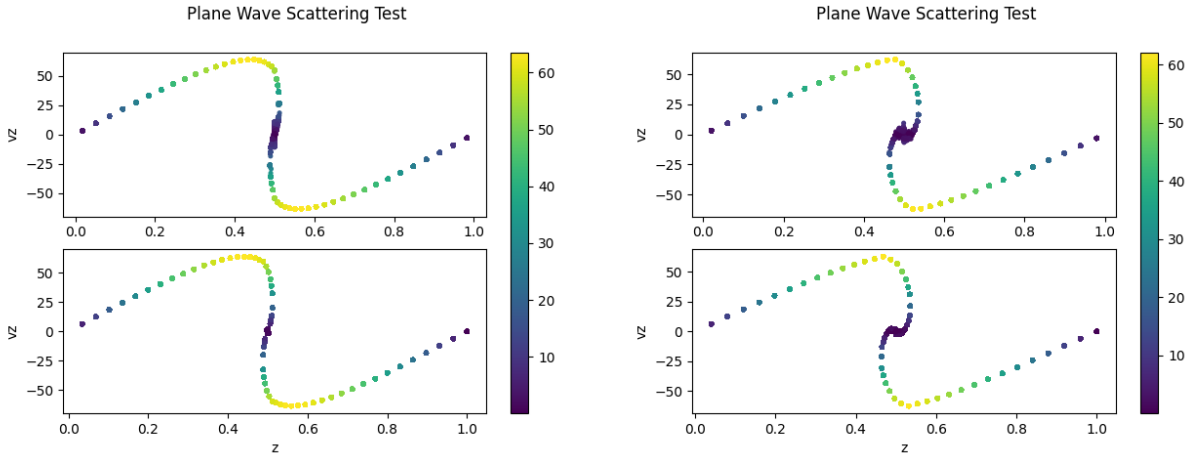
Before we present the results, though, we have to illustrate our exact simulation setup aside from this particle positions and velocities, which we provide as an `hdf5` file. For the most part, we keep the setup as it is presented in the default example `DM-L50-N128` but we change some details, of course. By far the most important one is contained in the configuration file of the simulation, `Config.sh`: `PBH_EFD`. We add it in the section for extra physics, which we also added to this example. In contrast, we delete the option `LEAN` which would require us to only have a single particle type but reduces memory usage significantly, as well as all options from the section for the generation of initial conditions. While `GADGET-4` allows creating initial conditions on the start of a simulation, we have already created the initial conditions for this test as described above. From the remaining options, there is only `PMGRID` that we need to change, which details the size of the mesh that the code should use to compute the gravitational forces between particles. For maximum efficiency, it should be set to two times `Nres`. Finally, due to reasons discussed in section 3.5, we want to push all particles down to the lowest time step such that all particles are active during every time step. This is achieved by the option `FORCE_EQUAL_TIMESTEPS`, for which we include a section for time integration options. Almost all other setup options are given by the parameter file, which we rename to `working-param.txt`.

Similar to the configuration file, we do not make drastic changes here. The biggest change again is that we delete the section for creation of initial conditions at startup and instead provide our initial conditions as the `InitCondFile`. In the same category, we provide the desired output times of the results via `OutputListFilename`, which is simply `outputs.txt`. That file contains values of a where the code will save all kinds of information as so-called snapshot files. The exact composition of the snapshot data can be controlled via configuration options; for our test, we simply need the IDs, positions, and velocities of the particles. We save these snapshots after every 0.05 interval in a. In order to give the output times with this file, the parameter `OutputListOn` in the output section must be set to its default of one. Next, we have to change the parameter `ICFormat` to three because our initial conditions are given as an `hdf5` file. Moving on, we increase the CPU time limit by a factor of ten to ten days because some of our simulations needed longer than a day and we wanted to avoid manual restarts of the code. However, this is only a choice of convenience because restarts from designated restart files or snapshots are extremely uncomplicated[29]. Due to limited resources, we have to reduce the maximum memory allowed for one task, `MaxMemSize` to 1200MB.
From the cosmological parameters, we adapt everything so that it fits our plane wave model. First, we set `TimeBegin` to 0.1. Then, we prescribe `Omega0` $= \Omega_\mathrm{M} = 1$, `OmegaLambda` $= \Omega_\Lambda = 0$, and `OmegaBaryon` $= 0.5$, because the PBHs are treated as baryonic gas particles by our algorithm. We define the Hubble parameter via `HubbleParam` $= h = 0.703$ and use `BoxSize` $=$ `Lbox`. At this point, we introduce all values for our algorithm using internal units (so including h), too, namely `MeanPBHScatteringCrossSection` as $8.559 \cdot 10^{-14}$, `AveragePBHMass` as $2.1701 \cdot 10^{-10}$, and `SigmaOverM` as $3.944 \cdot 10^{-4}$. What is more, we also give the inactive `Physical_time` for the beginning of our simulation. Lastly, we change the initial gas temperature of the SPH to 10, though this does not enter our calculations.
Our other changes are all made in order to increase the computation speed, so we increase the error tolerance in the time integration and the maximum allowed as well as the minimum required time step. Analogously, we slightly raise the error tolerance in the force computation and enhance the hydrodynamic time steps. The rest of the options remains untouched, though they still contain important information. They define the internal system of units from section 3.3.3 and control the gravitational softening.

Test Results

Since we are getting snapshots at regular intervals throughout the simulation, we can keep track of the simulation progress by visualising these data. For this purpose, we once more utilize Python. Plotting again both particle components separated and colour coded as before, we can clearly see the point when shell crossing occurs for the wave from the initial configuration (figure 3.3a). As we expected, this process leads to increased scatter activity, which smears up the wave structure visible for CDM particles. However, this can even start slightly before the exact time of shell crossing because particles are already close enough for scattering to occur if their scattering cross sections are large enough. All images from these results use a cross section four times bigger than the value we determined in section 2.4.2, purely for illustrative purposes.
After shell crossing, the particles from below coordinate $z = 0.5$ keep moving right, while the particles from above it keep moving left. During that evolution, the particles around $v_\mathrm{z} = 0$ have lower absolute velocities than particles with higher or lower v_z, which leads to the creation of a spiral in these plots (see figure 3.4a). Through the gravitational influence of the particles remaining around $z = 0.5$, some part of the particles with otherwise positive v_z changes to negative values, thus moving towards the center again. The same happens vice-versa for the other part of the wave. This process keeps forming a spiral until the simulation stops. Our final test results from plotting v_z versus z coordinates can be seen in figure 3.4b. In both particle components, a spiral structure has formed. For PBH particles, though, this structure is blurred due to the scattering. Note that a little bit of blurring is also visible for the CDM particles; it is induced by gravitational interaction.
While these images show that scatter interactions can have a distinct impact on the formed structures, there is even more we can learn from these tests. For example, we can investigate the impact of the size of the scattering cross section. In order to better quantify this question, we study it by considering quantitative parameters, namely the velocity dispersion σ in each direction and the anisotropy, which we define as the quotient of the difference of maximum and minimum of σ in all directions and the sum over all directions. If σ is isotropic, it is the same in each direction and our anisotropy parameter would thus be zero. If, on the other hand, the particles are maximally anisotropic, i.e. if there only is velocity dispersion in one direction,

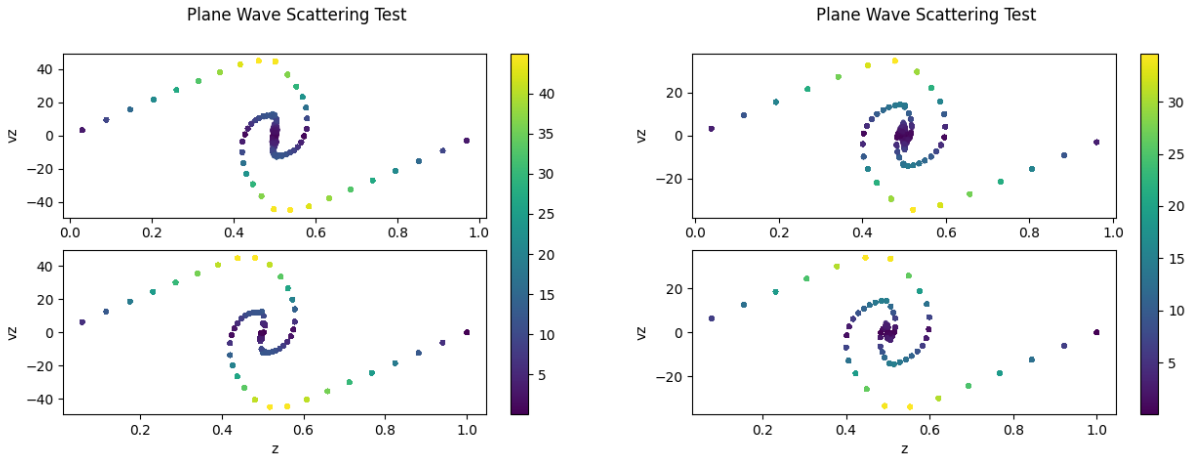
[29] https://wwwmpa.mpa-garching.mpg.de/gadget4/02_running/#restarting-a-run



(a) At a_{cross}, scattering intensifies and CDM particles enter a multi-stream regime.

(b) After a_{cross}, the PBH particles begin to form more extended structures.

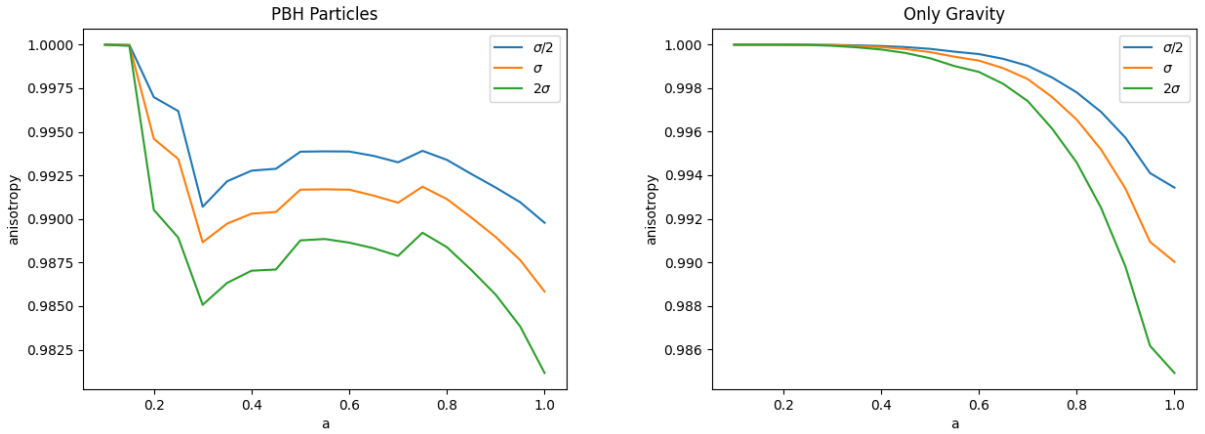
Figure 3.3.: Phase space state of our simulation. Colour coding and particle separation as described above.



(a) The continuous plane wave movement keeps creating a spiral.

(b) Finally, a spiral is visible in both components, though the PBH spiral is clearly smudged.

Figure 3.4.: Phase space state of our simulation. Colour coding and particle separation as described above.



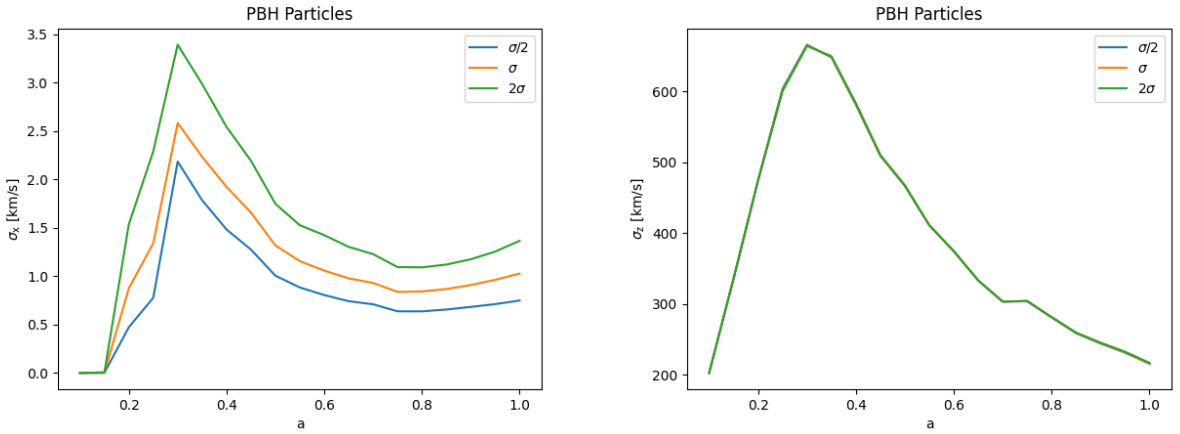
(a) Considering PBH particles.

(b) Considering CDM particles.

Figure 3.5.: Varying the scattering cross section and studying the effect on the anisotropy. The blue line represents only half of, green twice, and orange exactly the usual value.

then the anisotropy parameter would yield one.
With this definition, we can now compare tests with each other. One of our comparisons comes from varying the cross section parameter `SigmaOverM` by a factor two or one half and examining the difference this makes for velocity dispersions and anisotropy. When converting the velocities stored in the snapshots to physical values, we learn from Robertson 2017 that we still need to multiply the snapshot velocities with a factor $\sqrt{a}$ to achieve physical values. Plotting the anisotropy parameter versus the time a, we then find figure 3.5a for PBH particles and figure 3.5b for CDM particles[30]. In both cases, the orange line shows the usual value of the scattering cross section, whereas it is lower by a factor of two for the blue line and higher by a factor two for the green line. As we can see, the PBH particles may start with an anisotropic configuration, but as soon as scatter interactions begin to happen, this anisotropy is reduced. It is bouncing back a little after shell crossing because most particles retain some of their original movement direction and hence prevent further scattering by increasing particle distances. However, scatter events become more important again toward later times, causing the anisotropy to decrease again. For CDM particles, we see that they are able to hold on to their initial anisotropy for longer since they experience no scatter events. Nevertheless, the gravitational interaction with the PBH particles scattering in different directions also leads to CDM particles straying from pure movement in the z direction, thus decreasing the anisotropy. In both cases, we can clearly recognise the influence of differently sized cross sections: the larger the cross section, the more scattering events occur. Therefore, the importance of scattering is boosted, resulting in a steeper and more significant decline of the anisotropy. Note, however, that the anisotropy remains relatively high in all cases, indicating that the scatter interactions are not as influential as the plane wave movement.
We can understand this continued high level of anisotropy by considering the effect of the different cross sections on the velocity dispersion in e.g. x and z direction. σ behaves the same in x and y direction due to the symmetry of the velocities after scattering; by picking part of the new direction at random, we ensure precisely that. Figure 3.6a depicts the evolution of σ_x for PBH particles. Starting from zero because all particles begin with $v_x = 0$, we see the same effect as described above: the scatter interactions before and during shell crossing first lead to particle movement in x direction, before the wave movement dampens this trend again until scattering becomes important once more. The impact of different cross section sizes is clearly visible here. However, there is no difference to be seen for σ_z in figure 3.6b. Due to the other directions and the anisotropy showing a clear distinction, it is safe to assume that changing the cross section *does* have an effect and the code conveys this effect as intended. Nevertheless, 3.6b also teaches us that the importance of the scatter interactions can be suppressed by the environmental circumstances. Despite numerous scatter events, the initial plane wave configuration simply dominates these events in the z direction, causing the

[30] These results originate from an outdated version of the code that artificially increased the scatter probability. With the current version and our usual plane wave setup, scatter events are negligible compared to the plane wave movement, so we show these old plots instead.



(a) Considering σ_x.

(b) Considering σ_z.

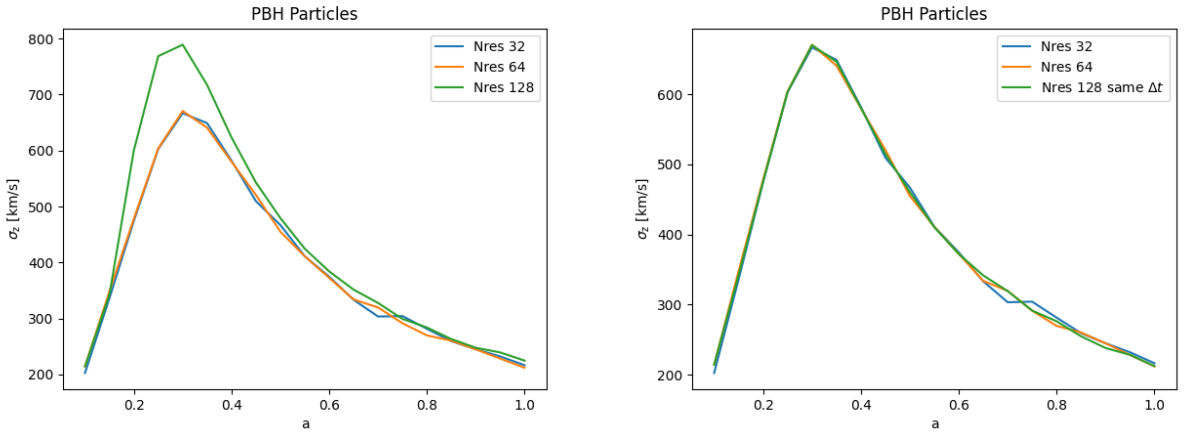
Figure 3.6.: Varying the scattering cross section and studying the effect on σ for PBH particles. The blue line represents only half of, green twice, and orange exactly the usual value.

overall anisotropy to remain high.
Another simulation parameter whose influence we can study is `Nres`, or in other words: we can check that the code converges for different particle numbers. To study this aspect, we employ the same quantitative parameters as above[31]. Plotting the evolution of σ_z for different particle numbers, we notice something odd at first (see figure 3.7a): while the lines for `Nres` $= 32$ (blue) and `Nres` $= 64$ (orange) are closely aligned to each other, the line for `Nres` $= 128$ (green) exhibits a significant bump at the time of shell crossing. This convergence problem is extensively discussed in section 8.1 of Springel, Pakmor et al. 2020, where we can find an explanation for this behaviour. For a box length of $1\,\mathrm{Mpc}/h$ and `Nres` $= 128$, the mean particle distance is $0.007\,812\,5\,\mathrm{Mpc}/h$, which is smaller than the default softening length of $0.01\,\mathrm{Mpc}/h$. In order to correctly simulate such small scales, sufficiently small time steps are necessary. As we can see from figure 3.7b, the bump can be avoided, i.e. the simulations converge, if we employ the configuration option `FORCE_EQUAL_TIMESTEPS`. However, this reduces the efficiency of the code by reducing all particles to the smallest time step determined for any particle, so we note this as our first caveat.

3.5. Caveats

The original goal of this project was to present a working algorithm within the GADGET-4 code that could simulate structure formation for PBH dark matter. During the inclusion of this algorithm in the code and while testing the code afterwards, we have found several problems that we could not solve due to time restrictions of this project. Instead, we use this section to list all known problems and some possible explanations as well as treatments.
Regarding the time step mentioned above, we note that allowing the code to choose individual time steps for all particles does not work well with our extension. However, this problem can be resolved by using the smallest time steps determined for any particle for all particles. This solution requires additional calculations for particles for which fewer calculations would already suffice to achieve the desired accuracy, though, so it is far from ideal. Since this convergence problem originates from the consideration of scales similar to the default softening length, it can also be resolved by adapting the softening length so that it is small compared to the mean particle distance again.
Nevertheless, enforcing the same time step for all particles helps us in another regard, which can be understood by considering the effect of using the same time step for all particles: it implies that all particles are active during every time step. This distinction between active and passive particles enters our algorithm in several

[31] Again, these results stem from an outdated version of the code which artificially increased the scatter probability and did not adapt the softening length. Still, they illustrate that time steps can resolve convergence problems.



(a) Allowing variation of time steps between particles. (b) Considering the smallest time step for all particles.

Figure 3.7.: Varying `Nres` and studying the effect on σ_z for PBH particles. The blue line represents 32, the orange 64, and the green 128 particles per dimension.

places. First of all, only active particles are added to the `targetlist`. Consequently, only active particles have their `HydroAccel` set to zero in `hydro_forces_determine()`. Crucially, active particles are allowed to scatter off of passive particles in the usual hydrodynamic interactions, while only active particles end up receiving velocity kicks. In our algorithm, we create symmetry in the interaction by assigning each scatter partner a new velocity caused by the scattering. Thus, if these new velocities are not applied to passive interaction partners, our interactions are no longer symmetric if different particle time steps are allowed, meaning that the interactions do not conserve energy or momentum.
Next to symmetry, there are two other important aspects of evaluating the scatter events: the order and how to deal with particles scattering on different processors. Concerning the order, we note that the way in which we evaluate the scatter list is far from perfect. Expanding equation 3.39, we see that after k scatter events,

$$\Delta\vec{a} = \Delta\vec{v}^{(k)} + \Delta\vec{v}^{(k-1)} + ... + \Delta\vec{v}^{(0)} = \vec{v}^{(k)} - \vec{v}^{(k-1)} + \vec{v}^{(k-1)} - \vec{v}^{(k-2)} + ... - \vec{v}^{(0)} = \vec{v}^{(k)} - \vec{v}^{(0)}, \quad (3.41)$$

meaning that the only scatter event that affects the particle is the last one. This is the reason why we have sorted the `scatter_list` by ascending probability, to have the last and only important scatter event be that with the highest scatter probability, which would arguably be the first one to happen in time. Effectively, this means that we only allow one scatter event per particle and time step, which seems to be a viable approximation judging by the scarcity of scatter events present in the diagnostic output. It is this lack of scatter activity that allows us to only reserve enough memory for the `scatter_list` to hold two events per particle. If, however, the scatter activity is significantly increased to an average of about two events per particle and time step, caution is advised for the memory allocated for the `scatter_list` as well as for the evaluation of that list.
In the case of many scatter events for any particle, we ideally want to consider each of those events for applying velocity kicks. For that purpose, it would be beneficial to sort the `scatter_list` by descending probability because of the argument that pairs with a higher probability are likely to scatter before pairs with lower probability. Since we cannot allow a particle to scatter twice during the same time step with the same initial velocity, each particle would have to receive a velocity kick immediately following a scatter event. Based on this new velocity, all pairs including this particle with probabilities below the event that just occurred should be evaluated again to see if their probabilities change as a result of the new relative velocity. They should then be ordered by descending scatter probability and the process continued until the particle has no further possibilities to scatter during this time step. For an approach like this, our method of evaluating the scatter interactions using multiple lists might not be the best. It would require tremendous computational effort to go through all scatter partners of a particle repeatedly. Because scatter events seem to remain scarce, we keep with our incorrect version of evaluating the scatter events and just note that a better way without considerably longer computation time would be preferable.
Concerning scattering across multiple processors, Robertson 2017 remarks that this is possible to happen

because particles are stored on different processors. If a particle required as a neighbour of a target particle is stored on a different processor, its essential data is imported from that processor to determine the interaction outcome. It might happen, though, that this particle also scatters on its native processor during the same step. In that case, the evaluation of the scatter events would proceed with the same initial velocity, not preserving energy and momentum just like before. To prevent this behaviour, Robertson 2017 implements an intricate scheme of rules about which particles are allowed to be fetched by which processor (see his section 3.4.3). He also states that a commonly used method is to reduce the size of the time steps, making multiple scatter events highly unlikely and thereby preventing these unwanted effects. Despite this benefit, having very small time steps is generally undesirable because it increases the computation time needed for a simulation. However, since enforcing the smallest time step for all particles solves the convergence and symmetry problems for us, we choose this method anyway, so we can apply it for this problem as well. We stress, though, that moving towards more efficiency and larger time steps also requires a reconsideration of this problem.

As mentioned in section 3.3.4, we measure the frequency of particles being imported to foreign processors with the variables `numberoflocalparticles` and `numberofforeignparticles`. This diagnostic output reveals that there were no foreign particles in any of our tests and simulations, which most likely is a result of our simulation setup. Until now, our simulations generally featured many more halos than processors, making it unlikely for any halo to be split up. Since we want to consider the formation of structure, i.e. of multiple halos, this behaviour remains the same throughout our simulations. We still caution in general that our algorithm has not been tested with the occurrence of foreign particles, so there might be further problems hiding there.

Lastly, just to reiterate from section 3.3.3, we remark that the criterion used to avoid pairs with zero relative velocity from scattering has a rather arbitrary form. By printing how many particles were prevented from scattering, we can see that in our plane wave tests, there is a considerable number of particles prevented during the very first step, when initial conditions assign exactly the same velocities to a range of particles. Afterwards, however, this condition affects no other particles, so it seems to work just fine. Still, it has been found through trial and error and a better understanding would be desirable, as would be a better criterion, potentially.

4. Results

Keeping the numerous caveats in mind, we now present some cosmological simulations. These simulations are the first we ran and consequently feature only low resolutions. Nevertheless, they can still improve our understanding of scatter interactions. Broadly speaking, we perform two kinds of simulations. For the first type, we prescribe the size of the simulation box via `Lbox` and the number of particles `Nres` and find the particle mass from equation 3.40. With a usual box length of $30\,\mathrm{Mpc}/h$ and our maximum particle resolution of 128 particles per dimension, though, the resulting particle mass m_P is rather high ($\sim 10^9\,\mathrm{M}_\odot$). Since we want each of our halos to consist of at least 100 particles, we can only resolve halos above $10^{11}\,\mathrm{M}_\odot$. This halo mass is not where we expect the scatter interactions to have an appreciable effect, which we can see from the following approximation:
In a simulation with `Lbox` $= 30\,\mathrm{Mpc}/h$ and `Nres` $= 128$, we find the total mass `Mbox` to be $232\,209.9 \cdot 10^{10}\,\mathrm{M}_\odot/h$, resulting in a mean density $\bar{\rho} = \mathrm{Mbox}/\mathrm{Lbox}^3 \approx 8.600 \cdot 10^{10}\,\mathrm{M}_\odot/\mathrm{Mpc}^3 h^2$. A typical definition defines a halo as a region of space with an average density equal to 200 times this mean density, so we write $\rho_\mathrm{halo} = 200\ \bar{\rho}$. We already noted above that the minimum resolvable halo mass for this setup is $M_\mathrm{halo} = 0.067\,742 \cdot 10^{10}\,\mathrm{M}_\odot/h$ (for $h = 0.67742$). Assuming a spherical halo, we can thus estimate the halo radius to be

$$r_\mathrm{halo} = \left(\frac{3\ M_\mathrm{halo}}{4\pi\ \rho_\mathrm{halo}} \right)^{1/3} \approx 0.021\ \mathrm{Mpc}/h. \tag{4.1}$$

From the virial theorem, we can find the virial velocity of a halo, which we assume to be the mean relative particle velocity inside the halo, as

$$v_\mathrm{halo} = v_\mathrm{vir} = \sqrt{\frac{G\ M_\mathrm{halo}}{r_\mathrm{halo}}}, \tag{4.2}$$

where G is the gravitational constant in `GADGET` units, namely 42.981 for our choice of h. Using this equation, we find that halos of $10^9\,\mathrm{M}_\odot$ correspond to velocities inside the halo of roughly 11.75 km/s, which is significantly higher than the 4-7 km/s that we determined to yield appreciable values of σ_sca/m in section 2.4.2, especially considering the strong dependence of σ_sca on the relative velocity. Instead, a halo mass of $10^8\,\mathrm{M}_\odot$ leads to relative velocities of roughly 5.54 km/s, so we also want to consider halos of this mass or somewhat lower with the second type of simulations.
For this purpose, we need particle masses of 10^5–$10^6\,\mathrm{M}_\odot$ to fit 100 particles inside a single desired halo. At the same time, our computation resources are still limited to `Nres` $= 128$ so that we have to adapt our box length accordingly. These smaller boxes would become too small to consider structure formation during the simulations, which is why we stop the simulations earlier than at $z = 0$.

4.1. Large Cosmological Boxes

As mentioned above, we use `Lbox` $= 30\,\mathrm{Mpc}/h$ for the simulations of our large cosmological boxes. To create the initial conditions for the simulations, we use `monofonIC`[32], which was introduced by Hahn, Rampf and Uhlemann 2020. This program requires a few parameters in addition to the box size to be given in the configuration file `example.conf`, the first one being `GridRes`, which is essentially the number of particles in each dimension, i.e. `Nres`. For our simulations, we use values of 64 and 128 here. The setup options also include the redshift of the start of the simulation `zstart`, which we leave at its default value of 24 and keep in mind that `GADGET-4`'s parameter file has to match this choice. Since our simulations only feature a low resolution for now, we decrease the order of accuracy of the Lagrangian perturbation theory calculations `LPTorder` to two. Apart from that, we do not change any further default options except for the output

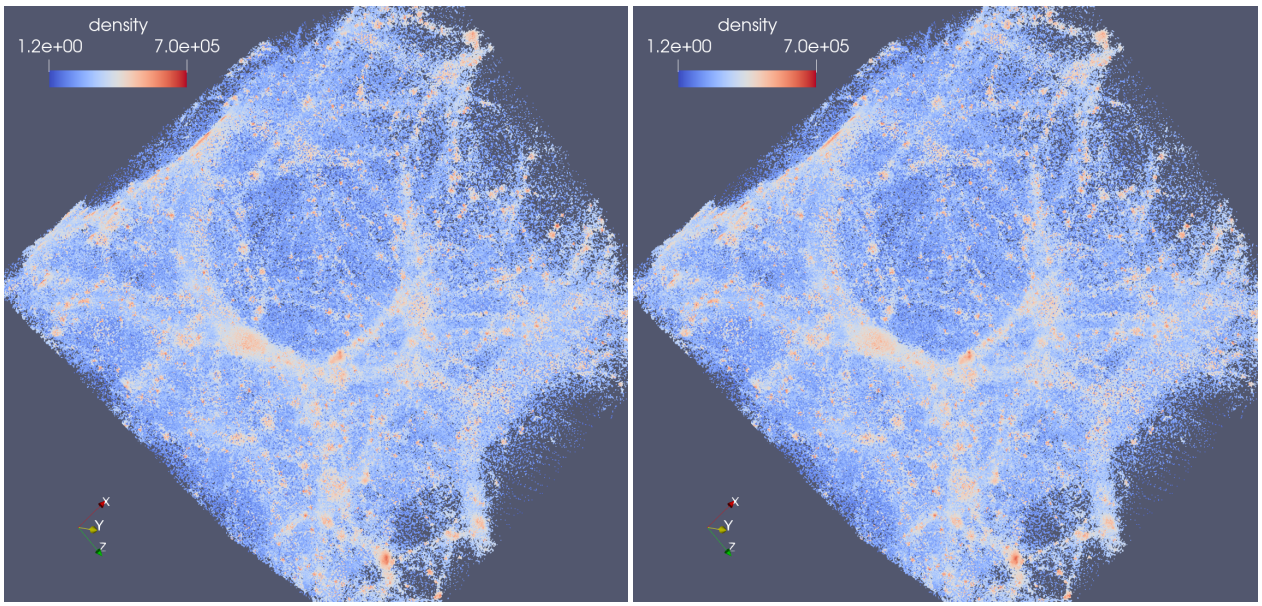
[32] https://bitbucket.org/ohahn/monofonic/src/master/

section, where we enable the `Gadget-2/3 HDF5 format` for the initial conditions to be stored in and provide a file name. This choice uses cosmological parameters according to the `Planck2018EE+BAO+SN` measurements (Planck Collaboration et al. 2020), i.e. $\Omega_\mathrm{M} = 0.3099$, $\Omega_\Lambda = 0.690021$, $\Omega_\mathrm{B} = 0.0488911$, and $h = 0.67742$. Of course, we have to provide GADGET-4 with these values in the parameter file used for the simulations. Lastly, this setup lets `monofonIC` create initial conditions for one of `GADGET`'s particle types, which is the CDM type 1 per default. In order to retrieve initial conditions for type 0 (gas or PBH particles), we have to navigate to `src/plugins/output_gadget_hdf5.cc`, lines 177 and 179. By exchanging the 1 and 0 of both lines, we instead create initial conditions for PBH particles, though we have to recompile the whole code after this change.

In the new parameter file `cosmo-param.txt`, created as a copy of `working-param.txt`, we begin by including the correct initial condition file in line 3. Since we are more interested in the exact evolution of these simulations compared to the plane wave tests, we double the frequency with which snapshots are created by decreasing the time interval between them to 0.025 in the new file `cosmo-outputs.txt`. Because the computer on which we run these simulations is still the same, we do not change the `TimeLimitCPU` and `MaxMemSize`. While we let these simulations run until $a = 1$ ($z = 0$), our choice above implies that we have to set `TimeBegin` to $a = 0.04$ ($z = 24$). Next, we include the cosmological parameters as detailed above, which means that we also have to change the newly introduced scattering parameters as their units depend on h. Considering the different value here, we set `MeanPBHScatteringCrossSection` to $7.9476 \cdot 10^{-14}$, `AveragePBHMass` to $2.091\,18 \cdot 10^{-10}$, and `SigmaOverM` to $3.8005 \cdot 10^{-4}$ (all in `GADGET` units). Despite the fact that it is not currently used, we also adapt `Physical_time` to our starting time, namely to $3.644 \cdot 10^{15}$ s. We do not change anything about the accuracy of the time integration, force calculation, system of units, etc, but we have to adapt the gravitational softening length section. The main reason for that is that `monofonIC` still saves the information that there are six particle types, even though in our case five of them always contain zero particles. Nevertheless, GADGET-4 requires that a softening class be provided for each particle type, which is why we include `SofteningClassOfPartType<particle type>` for particle types one to five and set them to zero. This way, we only have to specify `SofteningComovingClass0`, which is already given as a default, i.e. we only need to adapt the numbers. Following the idea that the softening length should amount to 1/20–1/30 of the mean particle distance, we can keep the default value of $0.01\,\mathrm{Mpc}/h$ for these simulations. Similarly, we do not change much in the configuration file compared to the plane wave configuration. According to the inclusion of new particle types, we have to change `NTYPES` to six, but since all share the same softening class, we can keep `NSOFTCLASSES` at one. Of course, we include our algorithm with `PBH_EFD`, use the smallest time steps, and adapt `PMGRID` as indicated by the particle number. Finally, we make use of GADGET-4's built-in ability to identify halos via a friends-of-friends algorithm[33] by using `FOF`. This algorithm identifies gravitationally bound structures within a certain particle type, which has to be specified via `FOF_PRIMARY_LINK_TYPE`. Note that the particle type has to be given as two to the power of the particle type, i.e. the gas particles (type 0) are $2^0 = 1$ and the CDM particles (type 1) are $2^1 = 2$. This option therefore needs to be changed when switching from CDM to PBH simulations, otherwise the group finding will not work. However, this would not be a big deal so long as we keep the snapshot we are interested in because GADGET-4 can also perform group finding in postprocessing. Additionally, we specify that each identified group should contain at least 32 members via `FOF_GROUP_MIN_LEN`. To even identify substructures within the halos, we further include `SUBFIND`. Lastly, we also use `SUBFIND_STORE_LOCAL_DENSITY` in order to retrieve local density values for all particles rather than just for the ones in `FOF` halos. Before running the simulation, we now only need to recompile the example `DM-L50-N128` and ensure that the script `job.sh` inside it uses the correct parameter file.

Once again, we use Python to evaluate the snapshots. Typically, 3D-images of cosmological simulations are created by assigning a colour to each particle depending on its density. The particle positions required for that are readily available from the snapshots. While `SUBFIND_STORE_LOCAL_DENSITY` allows the computation of the local density for all particles, we did not always have this option enabled. For these cases, we have to compute the densities ourselves because we could not find an existing module for Python to compute particle distances considering our periodic boundary conditions. With periodic boundaries, a particle should search for neighbours across the boundary of the simulation box by entering the box again from the opposite side. Our solution for this problem uses brute force and, consequently, does not scale well with much higher particle numbers. We simply copy the simulation box 26 times and add it to each side of the original cube as well as diagonally across borders and corners. Then, we construct a tree similar to the one used by `GADGET`

[33] https://wwwmpa.mpa-garching.mpg.de/gadget4/08_groupfiles/



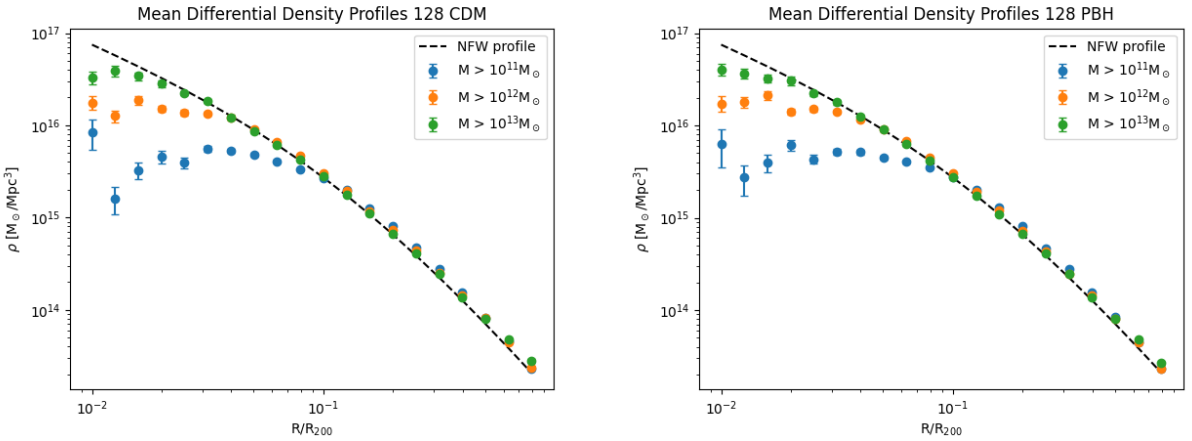
(a) Containing only CDM particles.

(b) Containing only PBH particles.

Figure 4.1.: Typical visualisation of cosmological snapshot data at $z = 0$. Particles are colour coded according to their density; red particles have high density, while blue particles have low density (given in internal units). `Nres` $= 128$ was used here for `Lbox` $= 30\,\mathrm{Mpc}/h$.

using `scipy.spatial.KDTree`[34], which was designed to allow quick searches for neighbours. In our case, we want to use the function `query()` for that, to which we can pass the coordinates of the points in the original box, the number of neighbours we want to find, and the number of processors available for the task. For the list of neighbours, it is important to note that the first neighbour of a particle will always be the particle itself, whereas we want to find the 40 closest neighbours that are not the particle itself. The choice of the value 40 is rather arbitrary, but the exact value should not matter as long as there are enough particles to avoid coincidental cluster detection. Vogelsberger, Zavala and Loeb 2012 adjusted the smoothing lengths h of their particles such that roughly 40 particles were included in a ball with radius h around each particle. They argue that this is enough to prevent measuring very small or large values of h at random, and we adopt their reasoning. Here, we finish the preparation of the evaluation by saving the positions, densities, and IDs of the particles in separate text files for each snapshot. We also save the velocities and the computed absolute velocities of all particles, though they are not needed for visualisation based on densities.
These new files enable us to use existing programs to visualise the snapshot data. For this project, we utilise ParaView[35] (a very useful tutorial[36] for this program was created by Lukas Winkler). With ParaView, we can easily open all evaluation files created from the snapshots as a single object and work on them simultaneously. To visualise the data, we simply tell the program which columns correspond to the coordinates and which should be used for the colour, and we already achieve the images 4.1 from the last snapshots of our large cosmological box with `Nres` $= 128$. Whereas the left image shows the box containing only CDM particles, the right box features PBH particles exclusively. The density scale is still given in internal units.
The figures are very similar to each other. In both cases, we can see that a network of structures emerges, the so-called 'cosmic web'. It is made up of high density filaments in red and low density areas in blue in between. So close is the similarity of the images, that it requires very thorough inspection to find even minute differences. They can be found as tiny changes in the shape of some structures. However, these differences could also be random fluctuations, so we try to quantify them.
For the quantitative analysis, we want to plot mean radial density profiles for the halos that are found by the `FOF` algorithm. These halos are stored in separate files from the snapshots that also have the `hdf5` format, so there is no problem accessing them with Python. Each group has numerous data stored for itself, however,

[34] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html
[35] https://www.paraview.org/
[36] https://video.lw1.at/w/470a188e-e6d0-4531-9ba3-43361564df27

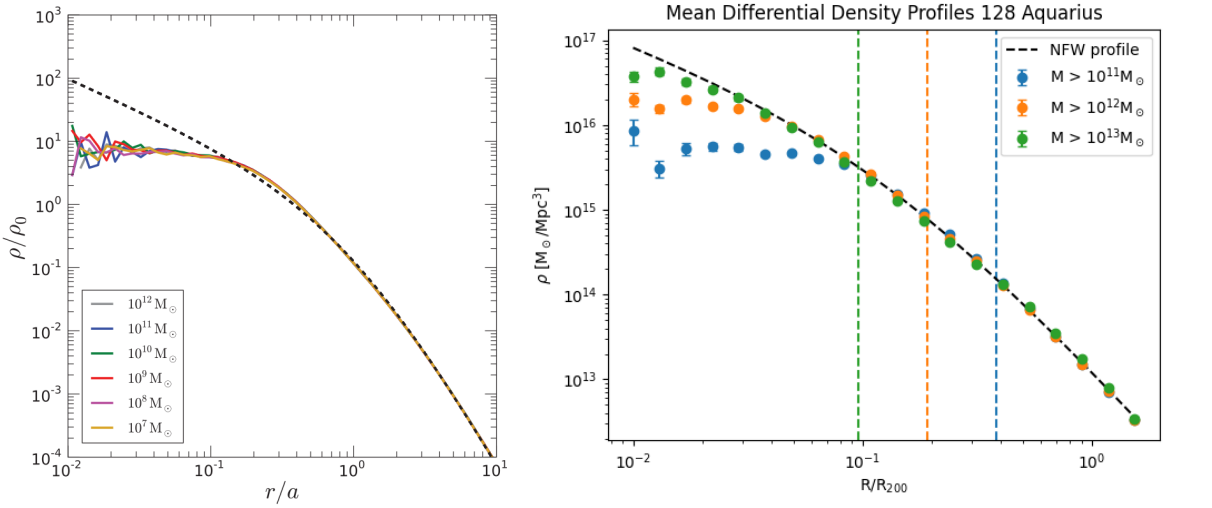


(a) Containing only CDM particles.

(b) Containing only PBH particles.

Figure 4.2.: Mean differential density profiles for our large cosmological boxes and `Nres` = 128 at $z = 0$. Density is given in $M_\odot/Mpc^3$, whereas the radius is normalised to the group radius. Different colours indicate different group mass bins, the legend gives the lower border of these bins. The black line shows an NFW profile for comparison.

there is no list of particles belonging to that group. Therefore, we employ our method of copying the original box 26 times again to construct another auxiliary tree. From the group data, we save the positions of the group centres as well as the radii and masses of the groups. Note that there are several options for the radii and masses, we choose the radius within which the mean density is 200 times the average density, following the definition presented above. Consequently, we take the group mass to be the mass within a ball of that radius. Within the tree, we then search for all particles in a sphere around each group centre, with the radius of the sphere corresponding to the group radius. This task can be performed with the function `query_ball_point()`. Afterwards, we sort through all groups, first eliminating all groups with zero radius or mass, which occur occasionally. Simultaneously, we compute the distance of each group member to the group centre and store all distances as well as a group ID (for which we simply enumerate the groups), its radius, and its mass in new files. In doing so, we avoid repeating the tree construction and member identification every time we want to make minor changes to our plots.
After loading the group mass, radius r_{group}, and member distances in a new script, we begin the plotting by renormalising the member distances according to the group radius. We immediately check if there are at least 100 particles in each group, otherwise we skip to the next group. Next, we create 20 radial bins for a distance $0.01 \leq d/r_{\text{group}} \leq 1$ in which we sort all members. As the volume of each radial bin, we calculate the difference between the volumes of two balls with radii corresponding to the upper and lower limit of the bin. Using this volume and the particle mass, we calculate the mean density of each radial bin. Rather than storing it directly, we now sort the groups in mass bins, with a new bin for each order of magnitude in solar masses. Finally, we can then compute the average density in each radial bin for all groups with similar masses. As error indications, we calculate the standard deviation of this average and divide by $\sqrt{n-1}$, where n is the number of groups under consideration. Plotting the mass-sorted density values against the normalised radii, we find figure 4.2. Different colours represent different mass bins as indicated by the legend. Note that we indicate the lower limit of each mass bin there. For comparison, we also show a Navarro-Frenk-White (NFW) profile as black dashed lines (Navarro, Frenk and S. D. M. White 1997). These typical dark matter halo profiles require two parameters to be specified, a reference density ρ_s and a scale radius r_s. As the scale radius, we use the average value of r_{group}, and we fit ρ_s to our halo density data.
Comparing both forms of dark matter, there are no big differences to be found. The only slight changes between these simulations are visible towards the center, where the PBH particles show a slightly smoother course. All mass bins deviate from the NFW profile, though at different centre distances. But in general, the density values of CDM and PBH particles hardly ever vary more than the error bars suggest as plausible random variance. While this was to be expected using large boxes and low particle numbers, the effect is still so small that we decide to run another test to see if our code works as intended.



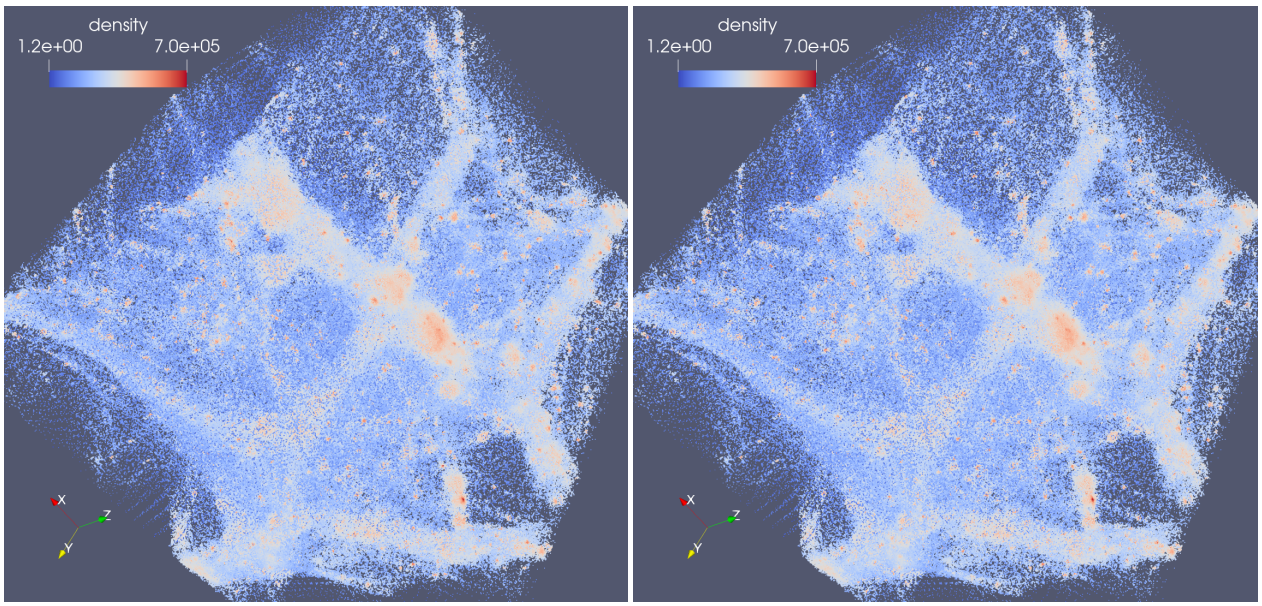
(a) From Vogelsberger, Zavala and Loeb 2012 with a Hernquist profile for comparison.

(b) Using our algorithm and large box setup and an NFW profile for comparison. Dashed vertical lines indicate softening length divided by mean group radius of each mass bin.

Figure 4.3.: Mean differential density profiles for fixed scattering cross sections sorted into mass bins as explained above.

4.1.1. Comparative Test

For this final test, we try to reproduce results presented by Vogelsberger, Zavala and Loeb 2012. In particular, we enforce a fixed value of $\sigma_{\text{sca}}/m = 10\,\text{cm}^2/\text{g}$ and study whether the density profiles will develop a core as in their figure 2 (figure 4.3a) or whether halos become as round as in their figure 3. The setup for this simulation is almost identical to the one before, the only differences being the smoothing length, which we set to $1/25 \cdot 30/128 = 0.009375$, and the value for `SigmaOverM`, for which we use $10\,\text{cm}^2/\text{g}$ in `GADGET` units, i.e. $1.523\,51 \cdot 10^{-6}$. However, we do need to change the source code slightly to incorporate velocity-independent cross sections. In `src/sph/hydra.cc`, we need to select line 346 rather than 345, which computes the conversion factor to physical units as h^2/a^4. This change can be derived from considering units or by simply following Robertson 2017 since he derived his equation 3.75 for velocity-independent SIDM. Moreover, we need to change the formula for the calculation of the scatter probability since we do not include v_{rel}^{-4} now. Enabling lines 1217, 1226, and 1227 while disabling lines 1216, 1224, and 1225 achieves just that. After running this simulation until $a = 1$ to allow the evolution of cores, we compare the results in figure 4.3.
The first thing we see when comparing these simulations is that Vogelsberger, Zavala and Loeb 2012 span a much larger range of halo masses, implying that they used higher resolutions. All of their halos converge to roughly the same normalised density value in the centre, which deviates from the theoretical prediction. Their comparison model, also represented by a black dashed line, is a Hernquist profile (Hernquist 1990). Strictly speaking, this model is different from the NFW profile we use, but their functional form is very similar and the NFW profile fits our data better, so we again utilise it for comparison here. In order to visualise the resolution, we also plot a vertical line for each mass bin at four times the softening length divided by the mean halo radius of that bin. Since more massive halos have larger radii, the position of this line moves towards the centre with increasing mass. Even for the largest halos, though, this line is placed at $0.1\,\text{R}/\text{R}_{200}$, so everything at smaller radii has to be treated with care.
Without the normalisation, we can see that our profiles do not converge to the same central value, but they all exhibit the same behaviour towards the core. Just as in the comparison case, the density profiles deviate from the theoretical model and form a cored centre. This behaviour reassures us that our algorithm works as intended; if the scatter cross section is large enough, we can recover the results from other SIDM simulations. Therefore, we are now interested if smaller cosmological scales are indeed subject to stronger scatter interactions without fixing the cross section.



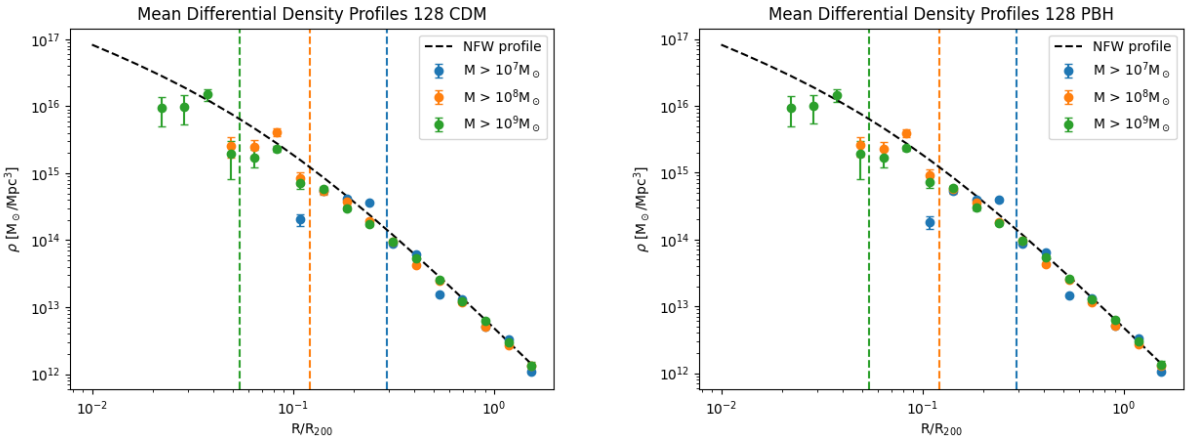
(a) Containing only CDM particles.

(b) Containing only PBH particles.

Figure 4.4.: Typical visualisation of cosmological snapshot data at $z = 1$. Particles are colour-coded according to their density; red particles have high density, while blue particles have low density (given in internal units). `Nres` $= 128$ was used here for `Lbox` $= 1.182\,19\,\mathrm{Mpc}/h$.

4.2. Small Cosmological Boxes

To simulate the small cosmological boxes, we revert the changes in the source code that accommodate a velocity-independent scattering cross section. In contrast to the large boxes, we now fix the particle mass and number and use equation 3.40 to find the corresponding box size. `Lbox` $= 0.591\,093\,\mathrm{Mpc}/h$ achieves a particle mass of $10^5\,\mathrm{M}_\odot$ with `Nres` $= 64$ and doubling `Nres` simply adds a factor two to `Lbox`. After changing the length of the box according to the particle number and adapting the file name the initial conditions should have in `example.conf`, we create new initial conditions for these boxes with `monofonIC`. Of course, these initial conditions need to be specified to `GADGET-4` explicitly through the parameter file. Since the size of the box stays fixed during the simulation, the structures forming inside will eventually outgrow the box, making it uninteresting for us to continue simulating so small a volume to redshift $z = 0$. Therefore, we also specify in the parameter file that the simulations should stop at $z = 1$ or scale factor $a = 0.5$, equivalently. Furthermore, we set `BoxSize` to `Lbox`, and adjust the smoothing length to $3.694\,343\,75 \cdot 10^{-4}\,\mathrm{Mpc}/h$. Apart from that, we keep the same parameter file as for the large boxes. The configuration file remains completely unchanged, we only adjust `PMGRID` for the particle number and `FOF_PRIMARY_LINK_TYPE` according to the particle type.
For the evaluation of the simulations, we use the same combination of Python and ParaView as detailed above. Figure 4.4 depicts the same visualisation as before, using the density for colour coding. While the structures forming in these boxes appear to be larger than in the larger boxes, the reduced `Lbox` implies that they are actually smaller. Similar to before, we can see some structures of the cosmic web emerging, but we also still see large parts of the box without these structures. Even for these particle masses, the density maps do not show clear distinctions between the dark matter types under consideration. Again, zooming into these images reveals tiny changes in the shapes of the structures, though they might be random fluctuations.
Trying to quantify these differences using mean density profiles sorted by group mass (shown in figure 4.5), we again find no significant distinctions between CDM and PBH particles. In fact, the differences are even smaller now. However, we also see that the density profiles are not as smooth as before, hinting at the fact that not the same amount of time has passed. The jumps in the profiles originate from the fact that these halos did not have enough time to reach virial equilibrium. This, in turn, can explain the lack of difference we observe. From the diagnostic output of the plane wave tests, we know that scatter probabilities increase over time. To understand this, we simply have to keep in mind that we are using roughly equally sized time



(a) Containing only CDM particles.

(b) Containing only PBH particles.

Figure 4.5.: Mean differential density profiles for our small cosmological boxes and `Nres` $= 128$ at $z = 1$. Density is given in $M_\odot/Mpc^3$, whereas the radius is normalised to the group radius. Different colours indicate different group mass bins, the legend gives the lower border of these bins. For comparison, the black line shows an NFW profile, while the vertical lines indicate the force resolution as above.

steps $\mathrm{d}\log(a)$, which correspond to increasingly larger time steps $\mathrm{d}t$, boosting the probabilities. Exactly the same is expected to happen in the cosmological simulations, but we stop the simulations of the small boxes at $a = 0.5$, avoiding the half with increased scatter probabilities. Consequently, we would need to simulate boxes large enough so that we can go to $a = 1$ in order to study the low particle masses in more detail and draw more observation-oriented conclusions.

5. Conclusion

5.1. Summary

In this project, we assume that dark matter consists completely of primordial black holes. These black holes are expected to form from primordial density fluctuations, but further alternative models have been proposed as well. This assumption is motivated by e.g. the LIGO observations featuring black holes that are likely of non-stellar origin. Moreover, these black holes could explain several observational conundra at once, among others the missing of ultra-faint dwarf galaxies and the origin of the super-massive black holes residing in the centres of galaxies. Through additional scatter interactions between them, PBHs would effectively be a form of SIDM and could, consequently, alleviate the problems standard CDM faces on small scales. While PBHs have already been dismissed as a probable candidate for dark matter, these early studies were based on monochromatic mass functions, which are not realistic. In fact, recent progress suggests that PBHs should follow an extended mass function corresponding to the thermal history of the universe. Such a mass function naturally provides peaks in the PBH-distribution at precisely the scales needed to explain the conundra mentioned above. That is why we assume this mass function for our project and study its effect on structure formation.
For this purpose, we use the simulation code `GADGET-4`. Its structure for hydrodynamic calculations closely resembles the effective fluid approach we take towards simulating the PBHs. We derive a mean scattering cross section and mass for primordial black holes from their extended mass function and use them to calculate a scatter probability for particle pairs in the N-body code. Notably, our cross sections (and, subsequently, our scatter probabilities) are very sensitive to the relative velocity of the particles. If a pair scatters, we determine new velocities by picking random directions while taking care of energy and momentum conservation. The application of these additional velocity kicks to each particle is the whole mechanical effect the scatter interactions have.
Testing our algorithm in a plane wave scenario, we find that there can be a small yet significant difference in the structures forming from PBHs, if the scattering cross section is sufficiently large. At the same time, scatter interactions can be suppressed by the environment. This is apparent again in our cosmological simulations, where a test with a fixed large cross section shows the formation of cored density profiles as expected. For our velocity-dependent cross section, though, neither large nor small scales reveal any difference between CDM-only and PBH simulations. However, this conclusion is subject to caveats. On the one hand, our current resolution might simply be too low in order to detect an effect even by simulating large scales all the way to $z = 0$. On the other hand, the resolution limit implies that we cannot reasonably simulate small scales further than $z = 1$, excluding the part of evolution where most scatter activity is expected to take place. Despite the lack of differences between CDM and PBH particles in our current simulations, it would therefore still be interesting for future simulations to consider sufficiently small scales over the whole evolution of the universe.

5.2. Outlook

Since this project is supposed to be completed in the usual curriculum for master students, it is exposed to time limitations. These limitations imply that we cannot solve all the problems we are aware of or address all aspects of interest. To not lose sight of various unfinished aspects, we collect them in this section.

Integration with GADGET-4

First, we note that there are official guidelines for changing the source code of GADGET-4[37]. For the most part, we followed these instructions, e.g. by starting global variables with upper case letters. However, our algorithm currently still hijacks the hydrodynamic calculations. While it can be disabled easily, we should ideally create our own subdirectory in the `src` folder containing our algorithm to compute scatter interactions completely. In `src/time_integration/kicks.cc`, we should then add an update to the particle velocities other than from hydrodynamic kicks. At the same time, it is not very likely that we will need the simulation parameter `Physical_time` again, so we should make it non-essential at least or remove it altogether. Similarly, the parameters `MeanPBHScatteringCrossSection` and `AveragePBHMass` are not utilised by the code right now. We could remove them, too, or introduce functionality so that σ_{sca}/m can be given to the code by these individual values. In the latter case, we should ensure that the value inferred from the separate parameters equals the one from the combined parameter `SigmaOverM`.

Optimisation

Second, our algorithm has not been optimised for maximum efficiency yet. For example, it is not the most efficient way to communicate between processors by sending the acceleration updates from all processors to all other processors. Instead, it would be better to send only those updates to a processor that are required for the application to particles residing there. The most important issue with time efficiency, though, remains the time step. Enforcing the smallest time step for all particles causes the code to slow down significantly because a lot of computations are carried out that are not necessary to reach the desired accuracy. In our algorithm, we use the smallest time steps to ensure convergence for different particle numbers `Nres`, but we also lean on it to prevent scattering of one particle across different processors during the same time step. Therefore, releasing the time step again to normal values would face two challenges: on the one hand, we would need to find a different solution for the convergence problem, which might be as simple as using sufficiently small softening lengths. On the other hand, we would have to employ another solution to avoid scattering across processors, e.g. the communication scheme from Robertson 2017. Nevertheless, if we allow all particles to be on different time steps, we still might want to reign in the size of these time steps. Rocha et al. 2013 halve the time step of each particle whose maximum scatter probability in the current time step was larger than 0.2 in order to achieve higher accuracy for particles experiencing multiple scatter events in a short amount of time. Once the maximum scatter probability of these particles is below 0.1 again, they raise the time step back to its normal value.
Regarding efficient use of memory, we should find a better way to reserve memory for the `scatter_list` as our current way scales linearly with particle number and thus faces challenges at higher resolutions. One possibility is to evaluate all pairs once while simply counting the number of scatter events. Afterwards, we could reserve memory accordingly before evaluating all pairs again. However, this solution would increase computation time, so it might not be ideal, either.

Physical Accuracy

Third, it would be very useful to come up with a better method to evaluate the `scatter_list` or the scatter interactions in general. At the moment, we only allow the most likely scatter event to affect a particle, which is physically inaccurate. Likewise, we could increase the physical accuracy of our simulations by considering accretion. As of now, we assume the mass function of the PBHs to stay constant over time, but this is not necessarily the case. In fact, the black holes are expected to grow via accretion, which might change the average mass and scattering cross section and thus influence our calculations. Considering only mean values, it might be possible to derive an equation governing the evolution of these properties over time, which would affect our scatter probabilities if mass and cross section are growing at different speeds. Analogously, we do not consider mergers of black holes yet, and they might be too infrequent to play an important role anyway. Still, if PBHs make up all or most of the dark matter, we expect there to be a whole background of gravitational waves, which could be predicted from our simulations. This would be a wonderful way to

[37] https://wwwmpa.mpa-garching.mpg.de/gadget4/11_migration/

test our simulations and the scenario, in general, because LISA should be sensitive enough to detect such a background once it launches (Bartolo et al. 2019).

Recalculations

Fourth, we can increase accuracy and understanding by recomputing two things in particular. For now, we have restricted the mass function from B. Carr, Clesse et al. 2020 to masses $10^{-3}\,\mathrm{M}_\odot < M_{\mathrm{PBH}} < 10^{3}\,\mathrm{M}_\odot$. We exclude less massive black holes because their low masses would most likely not influence average measurements much, while more massive black holes would simply be too scarce to have an impact. Moreover, this range already includes roughly 97% of the dark matter, so we are missing at most 3% of the matter when taking the average. Still, since the cross section is proportional to the sum of the masses squared, the high-mass end of the mass function might wield some influence on the average cross section. It would therefore be interesting to see how much the average values change when considering the whole mass range, as well as the change considering possible updates to the mass function from theory. Similarly, it would improve our understanding to derive a proper criterion to keep particles with zero relative velocity from scattering. The current criterion has been found through trial and error and requires constant diagnostic output to check that no pairs are artificially kept from scattering. Though this seems to never be happening, a more rigorous derivation of a criterion would be desirable.

Increasing Resolution

Finally, we could learn a lot from increasing the resolution, i.e. particle number, of our simulations. While we have seen in section 2.4.2 that there already is a difference between using a monochromatic and the average of an extended mass function, the latter still means that we are essentially prescribing one value for all simulation particles, each holding multiple black holes. Instead, one might implement a true extended mass function by significantly increasing the particle number and assigning different masses to particles based on the mass function. Alternatively, a first step in that direction would be to employ the different particle types of GADGET-4 and assign different masses to them, splitting the mass function in mass bins. As different masses imply different cross sections, there would then be classes of interaction types depending on the mass bins of the particles under consideration. This would allow studying additional effects like mass segregation. A setup like this might also make the inclusion of accretion and merger effects easier. Due to the high particle number required for that, these simulations would only be viable on small scales. Additionally, we would need to find other methods to calculate density profiles as we would no longer be able to simply copy the whole simulation box 26 times. Despite these problems, it would also be interesting to study larger scales with increased resolution because this would allow predictions and comparisons to observations to be more detailed. Nevertheless, halos of around $10^{8}\,\mathrm{M}_\odot$ are still the most promising environment for PBHs to have an appreciable influence, which is why simulating dwarf galaxies such as the Milky Way satellites at higher resolution is of particular interest. Through better comparison to observations, higher resolution would therefore help us to better answer the question: 'What if primordial black holes are the dark matter?'

5.3. Acknowledgements

I want to thank my supervisor, Oliver Hahn, Florian Kühnel, and Lukas Winkler for all the support and guidance which enabled me to complete this project. Thanks also to Volker Springel and his team for making the GADGET-4 code publicly available.

Appendix A.

Abbreviations

PBH Primordial Black Hole
MACHO MAssive Compact Halo Object
CDM Cold Dark Matter
CMB Cosmic Microwave Background
GADGET-4 GAlaxies with Dark matter and Gas intEracT, version four
SPH Smoothed-Particle Hydrodynamics
f_{PBH} fraction of CDM in PBHs
LMC Large Magellanic Cloud
SMBH Super-Massive Black Hole
M_{PBH} mass of a PBH
QCD quantum chromodynamics
SIDM self-interacting dark matter
σ_{sca} scattering cross section
NFW Navarro-Frenk-White

Appendix B.

Project Summary

In this project, we study the difference in structure formation between standard collisionless cold dark matter and primordial black holes, a form of self-interacting dark matter. While primordial black holes with a monochromatic mass function have already been ruled out as a viable source of dark matter, the thermal history of the universe naturally provides an extended mass function for these black holes obeying all current constraints. At the same time, this mass function allows primordial black holes to explain a range of cosmological conundra, among others microlensing events, gravitational waves, missing satellites, and the origin of the super-massive black holes in the centres of galaxies. However, the viability of this explanation strongly depends on the existence of some kind of proof that primordial black holes indeed constitute the dark matter.

Through their sheer abundance, these black holes can effectively be described as a collisional fluid, where individual black holes interact not only via gravity, but also through scatter processes. These additional interactions can then resolve small-scale problems of the traditional collisionless cold dark matter model, enabling a test for our model. Using the cosmological simulation code GADGET-4, we can compare pure cold dark matter simulations with simulations using primordial black holes as the dark matter.

With these simulations, we are unable to resolve individual black holes. Instead, each simulation particle represents an ensemble of black holes in phase space, which we assume to follow the thermal history mass function. Therefore, we derive average values for the black hole masses and scattering cross sections from this extended mass function. To implement the scattering interactions, we then use the already existing structure of hydrodynamic calculations in GADGET-4 because it is very similar to our algorithm: We check for all particle pairs if they interact during a given time step and compute a change of velocity if that is the case. These additional velocity kicks are the only effect the scatter interactions have on the rest of the code.

In order to test our algorithm, we perform a series of plane wave simulations. From these tests, we can learn that our algorithm works as intended as it produces a distinct difference between the two types of dark matter. However, we also learn that this difference is very small and can easily be dominated by the surrounding circumstances. If e.g. the relative velocities between the particles are too high, the scattering probabilities and the resulting effect will be negligible. Nevertheless, another test using cosmological initial conditions reveals that our algorithm can reproduce literature results for self-interacting dark matter, so we move on to cosmological simulations.

We consider two setups for these simulations, both using 2097152 particles. For the first setup, we set the box length to $30\,\mathrm{Mpc}/h$ and derive the particle mass to be roughly $10^9\,\mathrm{M}_\odot$. Since we want each dark matter halo to consist of at least 100 particles for statistical significance, this allows us to resolve halos above $10^{11}\,\mathrm{M}_\odot$. These halo masses, however, are not the ones for which we expect desirable relative velocities; those halos have a mass of about $10^8\,\mathrm{M}_\odot$. For the second setup, we therefore prescribe a particle mass of $10^5\,\mathrm{M}_\odot$ and derive a box length of roughly $1.2\,\mathrm{Mpc}/h$. This volume, though, is not large enough to study structure formation until redshift $z = 0$, because it would only contain a single structure at that point. Thus, we finalise the second type of simulations at $z = 1$.

The differential density profiles resulting from both simulation types show no difference between cold dark matter and primordial black hole simulations. However, they also show that our resolution has not been high enough to study the inner parts of the halos, where the differences are expected to be most noticeable. Similarly, the scattering interactions are becoming more important over time and our resolution does not allow to study desirable halo sizes below redshift $z = 1$. Still, our initial tests show that there *is* a difference between collisionless cold dark matter and collisional black holes. That is why our main conclusion is that it is highly desirable to improve our simulations using higher resolution. For this purpose, the implementation of our algorithm needs to be adapted, which we cannot do immediately due to time restrictions of this project.

Appendix C.

Projektzusammenfassung

In diesem Projekt untersuchen wir die Unterschiede in der Strukturbildung zwischen der klassischen kalten dunklen Materie und primordialen Schwarzen Löchern, einer Form selbstwechselwirkender dunkler Materie. Primordiale Schwarze Löcher mit einer monochromatischen Massenfunktion wurden zwar bereits als mögliche Quelle der dunklen Materie eliminiert, doch aus der thermischen Geschichte des Universums folgt direkt eine ausgedehnte Massenfunktion, die alle aktuellen Anforderungen erfüllt. Gleichzeitig erlaubt diese Massenfunktion eine Erklärung für diverse ungelöste kosmologische Fragestellungen, unter anderem zu Microlensing Events, Gravitationswellen, fehlenden Begleitgalaxien und dem Ursprung der supermassiven Schwarzen Löcher in den Zentren der Galaxien. Die Gültigkeit dieser Erklärung hängt jedoch sehr von einem Beweis ab, dass tatsächlich primordiale Schwarze Löcher die dunkle Materie darstellen.
Durch ihre bloße Menge können diese Schwarzen Löcher effektiv wie ein Fluid mit Stößen beschrieben werden, in dem die einzelnen Schwarzen Löcher sich nicht nur gravitativ anziehen, sondern auch aneinander streuen. Diese zusätzlichen Interaktionen können Probleme des traditionellen, stoßfreien Modells kalter dunkler Materie auf kleinen Skalen beheben und ermöglichen einen Test unseres Modells. Mithilfe des kosmologischen Simulationscodes GADGET-4 können wir Simulationen mit reiner kalter dunkler Materie mit solchen vergleichen, in denen die primordialen Schwarzen Löcher die dunkle Materie sind.
Mit diesen Simulationen können wir keine einzelnen Schwarzen Löcher auflösen. Stattdessen repräsentiert jedes Simulationsteilchen eine Ansammlung Schwarzer Löcher im Phasenraum, von denen wir annehmen, dass sie der thermischen Massenfunktion folgen. Daher berechnen wir die durchschnittlichen Massen und Streuquerschnitte der Schwarzen Löcher aus dieser Massenfunktion. Um die Streuwechselwirkungen in den Code einzubauen, nutzen wir dann die Struktur für hydrodynamische Wechselwirkungen aus GADGET-4, da sie unserem Algorithmus sehr ähnlich ist: Wir prüfen für alle Teilchenpaare, ob sie während eines Zeitschritts streuen, und berechnen eine Geschwindigkeitsänderung, wenn dem so ist. Diese zusätzlichen Geschwindigkeitskicks sind der einzige Effekt, den die Streuungen auf den Rest des Codes haben.
Als Test für unseren Algorithmus betrachten wir Simulationen von ebenen Wellen. Von diesen Tests können wir erkennen, dass unser Code wie geplant funktioniert, weil ein merklicher Unterschied zwischen den zwei Arten dunkler Materie sichtbar wird. Dieser Unterschied ist jedoch auch klein und kann leicht durch äußere Umstände dominiert werden. Wenn zum Beispiel die Relativgeschwindigkeiten zwischen den Teilchen zu hoch sind, werden Streuwahrscheinlichkeit und resultierender Effekt verschwindend gering. Nichtsdestotrotz zeigt ein weiterer Test mit kosmologischen Anfangsbedingungen, dass unser Algorithmus Literaturergebnisse reproduzieren kann, sodass wir uns kosmologischen Simulationen widmen.
Wir betrachten zwei Arten von Simulationen, beide mit 2097152 Teilchen. Für die erste Art setzen wir die Länge der Simulationsbox auf $30\,\mathrm{Mpc}/h$, was zu einer Teilchenmasse von etwa $10^9\,\mathrm{M}_\odot$ führt. Da jeder Halo dunkler Materie aus statistischen Gründen aus mindestens 100 Teilchen bestehen soll, ermöglicht uns dies, Halos mit mehr als $10^{11}\,\mathrm{M}_\odot$ aufzulösen. Diese Halomassen sind jedoch nicht jene, für die wir geeignete Relativgeschwindigkeiten erwarten; die bekommen wir bei Massen von $10^8\,\mathrm{M}_\odot$. Für die zweite Art fixieren wir daher die Teilchenmasse auf $10^5\,\mathrm{M}_\odot$ und erhalten eine Boxlänge von circa $1.2\,\mathrm{Mpc}/h$. Allerdings ist dieses Volumen nicht groß genug, um Strukturbildung bis zu einer Rotverschiebung $z = 0$ zu betrachten, da es dann nur noch eine einzige Struktur enthalten würde. Deswegen beenden wir diese Simulationen bei $z = 1$.
Die differentiellen Dichteprofile, die sich aus beiden Simulationen ergeben, zeigen keinen Unterschied zwischen den Arten dunkler Materie. Sie zeigen aber auch, dass unsere Auflösung nicht hoch genug war, um die inneren Teile der Halos zu untersuchen, wo die deutlichsten Unterschiede erwartet werden. Ein ähnliches Problem ist, dass die Streuinteraktionen mit der Zeit wichtiger werden, unsere Auflösung jedoch nicht reicht, um geeignete Halomassen weiter als Rotverschiebung $z = 1$ zu simulieren. Trotzdem zeigen unsere anfänglichen Tests, dass es einen Unterschied zwischen stoßfreier kalter dunkler Materie und streuenden Schwarzen Löchern *gibt*.

Deswegen ist unsere wichtigste Schlussfolgerung, dass es sehr wünschenswert wäre, unsere Simulationen mit höherer Auflösung zu wiederholen. Zu diesem Zweck muss unser Algorithmus angepasst werden, was wir aufgrund zeitlicher Beschränkungen des Projekts nicht gleich selber tun können.

Bibliography

Arkani-Hamed, Nima et al. (Jan. 2009). 'A theory of dark matter'. In: *Physical Review D* 79.1. ISSN: 1550-2368. DOI: 10.1103/physrevd.79.015014. URL: http://dx.doi.org/10.1103/PhysRevD.79.015014 (cit. on p. 14).

Banerjee, Arka et al. (Feb. 2020). 'Signatures of self-interacting dark matter on cluster density profile and subhalo distributions'. In: *Journal of Cosmology and Astroparticle Physics* 2020.02, pp. 024–024. ISSN: 1475-7516. DOI: 10.1088/1475-7516/2020/02/024. URL: http://dx.doi.org/10.1088/1475-7516/2020/02/024 (cit. on p. 14).

Barnes, Josh and Piet Hut (Dec. 1986). 'A hierarchical O(N log N) force-calculation algorithm'. In: *Nature* 324.6096, pp. 446–449. DOI: 10.1038/324446a0 (cit. on p. 23).

Bartolo, N. et al. (May 2019). 'Testing primordial black holes as dark matter with LISA'. In: *Physical Review D* 99.10. ISSN: 2470-0029. DOI: 10.1103/physrevd.99.103521. URL: http://dx.doi.org/10.1103/PhysRevD.99.103521 (cit. on p. 53).

Belotsky, Konstantin M. et al. (Mar. 2019). 'Clusters of Primordial Black Holes'. In: *The European Physical Journal C* 79.3. ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-019-6741-4. URL: http://dx.doi.org/10.1140/epjc/s10052-019-6741-4 (cit. on p. 8).

Benson, Andrew J. (Feb. 2012). 'Galacticus: A semi-analytic model of galaxy formation'. In: *New Astronomy* 17.2, pp. 175–197. ISSN: 1384-1076. DOI: 10.1016/j.newast.2011.07.004. URL: http://dx.doi.org/10.1016/j.newast.2011.07.004 (cit. on p. 19).

Boddy, Kimberly K. et al. (Nov. 2014). 'Strongly interacting dark matter: Self-interactions and keV lines'. In: *Physical Review D* 90.9. ISSN: 1550-2368. DOI: 10.1103/physrevd.90.095016. URL: http://dx.doi.org/10.1103/PhysRevD.90.095016 (cit. on p. 14).

Boldrini, Pierre et al. (Jan. 2020). 'Cusp-to-core transition in low-mass dwarf galaxies induced by dynamical heating of cold dark matter by primordial black holes'. In: *Monthly Notices of the Royal Astronomical Society* 492.4, pp. 5218–5225. ISSN: 1365-2966. DOI: 10.1093/mnras/staa150. URL: http://dx.doi.org/10.1093/mnras/staa150 (cit. on p. 6).

Brandt, Timothy D. (June 2016). 'CONSTRAINTS ON MACHO DARK MATTER FROM COMPACT STELLAR SYSTEMS IN ULTRA-FAINT DWARF GALAXIES'. In: *The Astrophysical Journal* 824.2, p. L31. ISSN: 2041-8213. DOI: 10.3847/2041-8205/824/2/l31. URL: http://dx.doi.org/10.3847/2041-8205/824/2/L31 (cit. on p. 8).

Carr, B. J. (Oct. 1975). 'The primordial black hole mass spectrum.' In: *ApJ* 201, pp. 1–19. DOI: 10.1086/153853 (cit. on p. 9).

Carr, B. J. and M. Sakellariadou (May 1999). 'Dynamical Constraints on Dark Matter in Compact Objects'. In: *ApJ* 516.1, pp. 195–220. DOI: 10.1086/307071 (cit. on p. 8).

Carr, Bernard, Sebastien Clesse et al. (2020). *Cosmic Conundra Explained by Thermal History and Primordial Black Holes*. arXiv: 1906.08217 [astro-ph.CO] (cit. on pp. 9, 11, 12, 16, 30, 53).

Carr, Bernard and Florian Kühnel (Oct. 2020). 'Primordial Black Holes as Dark Matter: Recent Developments'. In: *Annual Review of Nuclear and Particle Science* 70.1, pp. 355–394. ISSN: 1545-4134. DOI: 10.1146/annurev-nucl-050520-125911. URL: http://dx.doi.org/10.1146/annurev-nucl-050520-125911 (cit. on pp. 5–9).

Carr, Bernard and Joseph Silk (May 2018). 'Primordial black holes as generators of cosmic structures'. In: *Monthly Notices of the Royal Astronomical Society* 478.3, pp. 3756–3775. ISSN: 1365-2966. DOI: 10.1093/mnras/sty1204. URL: http://dx.doi.org/10.1093/mnras/sty1204 (cit. on p. 6).

Chiba, Takeshi and Shuichiro Yokoyama (Aug. 2017). 'Spin distribution of primordial black holes'. In: *Progress of Theoretical and Experimental Physics* 2017.8. 083E01. ISSN: 2050-3911. DOI: 10.1093/ptep/ptx087. eprint: https://academic.oup.com/ptep/article-pdf/2017/8/083E01/19488901/ptx087.pdf. URL: https://doi.org/10.1093/ptep/ptx087 (cit. on p. 5).

Chluba, Jens, Adrienne L. Erickcek and Ido Ben-Dayan (Sept. 2012). 'PROBING THE INFLATON: SMALL-SCALE POWER SPECTRUM CONSTRAINTS FROM MEASUREMENTS OF THE COSMIC MICROWAVE BACKGROUND ENERGY SPECTRUM'. In: *The Astrophysical Journal* 758.2, p. 76. ISSN: 1538-4357. DOI: 10.1088/0004-637x/758/2/76. URL: http://dx.doi.org/10.1088/0004-637X/758/2/76 (cit. on p. 7).

Correa, Camila A (Feb. 2021). 'Constraining velocity-dependent self-interacting dark matter with the Milky Way's dwarf spheroidal galaxies'. In: *Monthly Notices of the Royal Astronomical Society* 503.1, pp. 920–937. ISSN: 1365-2966. DOI: 10.1093/mnras/stab506. URL: http://dx.doi.org/10.1093/mnras/stab506 (cit. on p. 14).

Croton, Darren J. et al. (Feb. 2016). 'SEMI-ANALYTIC GALAXY EVOLUTION (SAGE): MODEL CALIBRATION AND BASIC RESULTS'. In: *The Astrophysical Journal Supplement Series* 222.2, p. 22. ISSN: 1538-4365. DOI: 10.3847/0067-0049/222/2/22. URL: http://dx.doi.org/10.3847/0067-0049/222/2/22 (cit. on p. 19).

Davé, Romeel et al. (Apr. 2019). 'simba: Cosmological simulations with black hole growth and feedback'. In: *Monthly Notices of the Royal Astronomical Society* 486.2, pp. 2827–2849. ISSN: 1365-2966. DOI: 10.1093/mnras/stz937. URL: http://dx.doi.org/10.1093/mnras/stz937 (cit. on p. 19).

Elteren, A. van et al. (Apr. 2019). 'Survivability of planetary systems in young and dense star clusters'. In: *Astronomy & Astrophysics* 624, A120. ISSN: 1432-0746. DOI: 10.1051/0004-6361/201834641. URL: http://dx.doi.org/10.1051/0004-6361/201834641 (cit. on p. 6).

Farr, Will M. et al. (Oct. 2011). 'THE MASS DISTRIBUTION OF STELLAR-MASS BLACK HOLES'. In: *The Astrophysical Journal* 741.2, p. 103. ISSN: 1538-4357. DOI: 10.1088/0004-637x/741/2/103. URL: http://dx.doi.org/10.1088/0004-637X/741/2/103 (cit. on p. 4).

Frampton, Paul H. and Thomas W. Kephart (July 2005). 'PRIMORDIAL BLACK HOLES, HAWKING RADIATION AND THE EARLY UNIVERSE'. In: *Modern Physics Letters A* 20.21, pp. 1573–1576. ISSN: 1793-6632. DOI: 10.1142/s0217732305017688. URL: http://dx.doi.org/10.1142/S0217732305017688 (cit. on p. 7).

García-Bellido, Juan (2019). 'Primordial black holes and the origin of the matter–antimatter asymmetry'. In: *Phil. Trans. Roy. Soc. Lond. A* 377.2161. Ed. by John Dainton, p. 20190091. DOI: 10.1098/rsta.2019.0091 (cit. on p. 5).

Gingold, R. A. and J. J. Monaghan (Nov. 1977). 'Smoothed particle hydrodynamics: theory and application to non-spherical stars.' In: *MNRAS* 181, pp. 375–389. DOI: 10.1093/mnras/181.3.375 (cit. on p. 19).

Hahn, Oliver, Cornelius Rampf and Cora Uhlemann (Dec. 2020). 'Higher order initial conditions for mixed baryon–CDM simulations'. In: *Monthly Notices of the Royal Astronomical Society* 503.1, pp. 426–445. ISSN: 1365-2966. DOI: 10.1093/mnras/staa3773. URL: http://dx.doi.org/10.1093/mnras/staa3773 (cit. on p. 44).

Hasinger, Günther (2020). *Is the Dark Matter made of Primordial Black Holes?* URL: https://www.cosmos.esa.int/documents/13611/3707333/20200528_Hasinger.pdf (cit. on p. 4).

Hawking, Stephen (Apr. 1971). 'Gravitationally Collapsed Objects of Very Low Mass'. In: *Monthly Notices of the Royal Astronomical Society* 152.1, pp. 75–78. ISSN: 0035-8711. DOI: 10.1093/mnras/152.1.75. eprint: https://academic.oup.com/mnras/article-pdf/152/1/75/9360899/mnras152-0075.pdf. URL: https://doi.org/10.1093/mnras/152.1.75 (cit. on p. 4).

Hernquist, Lars (June 1990). 'An Analytical Model for Spherical Galaxies and Bulges'. In: *ApJ* 356, p. 359. DOI: 10.1086/168845 (cit. on p. 48).

Hopkins, Philip F. (Apr. 2015). 'A new class of accurate, mesh-free hydrodynamic simulation methods'. In: *Monthly Notices of the Royal Astronomical Society* 450.1, pp. 53–110. ISSN: 1365-2966. DOI: 10.1093/mnras/stv195. URL: http://dx.doi.org/10.1093/mnras/stv195 (cit. on p. 19).

– (2017). *A New Public Release of the GIZMO Code*. arXiv: 1712.01294 [astro-ph.IM] (cit. on p. 19).

Inoue, Yoshiyuki and Alexander Kusenko (Oct. 2017). 'New X-ray bound on density of primordial black holes'. In: *Journal of Cosmology and Astroparticle Physics* 2017.10, pp. 034–034. ISSN: 1475-7516. DOI: 10.1088/1475-7516/2017/10/034. URL: http://dx.doi.org/10.1088/1475-7516/2017/10/034 (cit. on p. 8).

Kahlhoefer, Felix et al. (July 2015). 'On the interpretation of dark matter self-interactions in Abell 3827'. In: *Monthly Notices of the Royal Astronomical Society: Letters* 452.1, pp. L54–L58. ISSN: 1745-3933. DOI: 10.1093/mnrasl/slv088. URL: http://dx.doi.org/10.1093/mnrasl/slv088 (cit. on p. 13).

Kaplinghat, Manoj, Ryan E. Keeley et al. (July 2014). 'Tying Dark Matter to Baryons with Self-Interactions'. In: *Physical Review Letters* 113.2. ISSN: 1079-7114. DOI: 10.1103/physrevlett.113.021302. URL: http://dx.doi.org/10.1103/PhysRevLett.113.021302 (cit. on p. 14).

Kaplinghat, Manoj, Sean Tulin and Hai-Bo Yu (Jan. 2016). 'Dark Matter Halos as Particle Colliders: Unified Solution to Small-Scale Structure Puzzles from Dwarfs to Clusters'. In: *Physical Review Letters* 116.4. ISSN: 1079-7114. DOI: 10.1103/physrevlett.116.041302. URL: http://dx.doi.org/10.1103/PhysRevLett.116.041302 (cit. on p. 14).

Lacey, C. G. and J. P. Ostriker (Dec. 1985). 'Massive black holes in galactic halos ?' In: *ApJ* 299, pp. 633–652. DOI: 10.1086/163729 (cit. on p. 8).

Lehmann, Benjamin V., Stefano Profumo and Jackson Yant (Apr. 2018). 'The maximal-density mass function for primordial black hole dark matter'. In: *Journal of Cosmology and Astroparticle Physics* 2018.04, pp. 007–007. ISSN: 1475-7516. DOI: 10.1088/1475-7516/2018/04/007. URL: http://dx.doi.org/10.1088/1475-7516/2018/04/007 (cit. on pp. 9, 10).

Mouri, Hideaki and Yoshiaki Taniguchi (Feb. 2002). 'Runaway Merging of Black Holes: Analytical Constraint on the Timescale'. In: *The Astrophysical Journal* 566.1, pp. L17–L20. ISSN: 0004-637X. DOI: 10.1086/339472. URL: http://dx.doi.org/10.1086/339472 (cit. on pp. 15, 16).

Musco, Ilia and John C Miller (June 2013). 'Primordial black hole formation in the early universe: critical behaviour and self-similarity'. In: *Classical and Quantum Gravity* 30.14, p. 145009. ISSN: 1361-6382. DOI: 10.1088/0264-9381/30/14/145009. URL: http://dx.doi.org/10.1088/0264-9381/30/14/145009 (cit. on p. 12).

Nadler, Ethan O. et al. (June 2020). 'Signatures of Velocity-dependent Dark Matter Self-interactions in Milky Way-mass Halos'. In: *The Astrophysical Journal* 896.2, p. 112. ISSN: 1538-4357. DOI: 10.3847/1538-4357/ab94b0. URL: http://dx.doi.org/10.3847/1538-4357/ab94b0 (cit. on p. 14).

Navarro, Julio F., Carlos S. Frenk and Simon D. M. White (Dec. 1997). 'A Universal Density Profile from Hierarchical Clustering'. In: *The Astrophysical Journal* 490.2, pp. 493–508. ISSN: 1538-4357. DOI: 10.1086/304888. URL: http://dx.doi.org/10.1086/304888 (cit. on p. 47).

Niikura, Hiroko et al. (Apr. 2019). 'Constraints on Earth-mass primordial black holes from OGLE 5-year microlensing events'. In: *Physical Review D* 99.8. ISSN: 2470-0029. DOI: 10.1103/physrevd.99.083503. URL: http://dx.doi.org/10.1103/PhysRevD.99.083503 (cit. on p. 6).

Özel, Feryal and Paulo Freire (2016). 'Masses, Radii, and the Equation of State of Neutron Stars'. In: *Annual Review of Astronomy and Astrophysics* 54.1, pp. 401–440. DOI: 10.1146/annurev-astro-081915-023322. eprint: https://doi.org/10.1146/annurev-astro-081915-023322. URL: https://doi.org/10.1146/annurev-astro-081915-023322 (cit. on p. 4).

Peter, Annika H. G. et al. (Jan. 2013). 'Cosmological simulations with self-interacting dark matter – II. Halo shapes versus observations'. In: *Monthly Notices of the Royal Astronomical Society* 430.1, pp. 105–120. ISSN: 0035-8711. DOI: 10.1093/mnras/sts535. URL: http://dx.doi.org/10.1093/mnras/sts535 (cit. on p. 13).

Piattella, Oliver (2018). 'Lecture Notes in Cosmology'. In: *UNITEXT for Physics*. ISSN: 2198-7890. DOI: 10.1007/978-3-319-95570-4. URL: http://dx.doi.org/10.1007/978-3-319-95570-4 (cit. on p. 14).

Pillepich, Annalisa et al. (Dec. 2017). 'First results from the IllustrisTNG simulations: the stellar mass content of groups and clusters of galaxies'. In: *Monthly Notices of the Royal Astronomical Society* 475.1, pp. 648–675. ISSN: 1365-2966. DOI: 10.1093/mnras/stx3112. URL: http://dx.doi.org/10.1093/mnras/stx3112 (cit. on p. 19).

Planck Collaboration et al. (Sept. 2020). 'Planck 2018 results. VI. Cosmological parameters'. In: *A&A* 641, A6, A6. DOI: 10.1051/0004-6361/201833910. arXiv: 1807.06209 [astro-ph.CO] (cit. on pp. 2, 10, 45).

Pospelov, Maxim, Adam Ritz and Mikhail Voloshin (Dec. 2008). 'Bosonic super-WIMPs as keV-scale dark matter'. In: *Physical Review D* 78.11. ISSN: 1550-2368. DOI: 10.1103/physrevd.78.115012. URL: http://dx.doi.org/10.1103/PhysRevD.78.115012 (cit. on p. 14).

Quinn, D. P. et al. (June 2009). 'On the reported death of the MACHO era'. In: *Monthly Notices of the Royal Astronomical Society: Letters* 396.1, pp. L11–L15. ISSN: 1745-3933. DOI: 10.1111/j.1745-3933.2009.00652.x. URL: http://dx.doi.org/10.1111/j.1745-3933.2009.00652.x (cit. on p. 8).

Raidal, Martti et al. (Feb. 2019). 'Formation and evolution of primordial black hole binaries in the early universe'. In: *Journal of Cosmology and Astroparticle Physics* 2019.02, pp. 018–018. ISSN: 1475-7516. DOI:

10.1088/1475-7516/2019/02/018. URL: http://dx.doi.org/10.1088/1475-7516/2019/02/018 (cit. on p. 5).

Robertson, Andrew (2017). 'The Cosmological Implications of Self-Interacting Dark Matter'. Durham University. URL: http://etheses.dur.ac.uk/12305/ (cit. on pp. 13, 21, 28, 33, 40, 42, 43, 48, 52).

Robertson, Andrew et al. (July 2019). 'Observable tests of self-interacting dark matter in galaxy clusters: cosmological simulations with SIDM and baryons'. In: *Monthly Notices of the Royal Astronomical Society* 488.3, pp. 3646–3662. ISSN: 1365-2966. DOI: 10.1093/mnras/stz1815. URL: http://dx.doi.org/10.1093/mnras/stz1815 (cit. on p. 13).

Rocha, Miguel et al. (Jan. 2013). 'Cosmological simulations with self-interacting dark matter – I. Constant-density cores and substructure'. In: *Monthly Notices of the Royal Astronomical Society* 430.1, pp. 81–104. ISSN: 0035-8711. DOI: 10.1093/mnras/sts514. URL: http://dx.doi.org/10.1093/mnras/sts514 (cit. on pp. 14, 15, 19, 20, 52).

Rubin, Vera C. and Jr. Ford W. Kent (Feb. 1970). 'Rotation of the Andromeda Nebula from a Spectroscopic Survey of Emission Regions'. In: *ApJ* 159, p. 379. DOI: 10.1086/150317 (cit. on p. 2).

Springel, Volker (Jan. 2010). 'E pur si muove:Galilean-invariant cosmological hydrodynamical simulations on a moving mesh'. In: *Monthly Notices of the Royal Astronomical Society* 401.2, pp. 791–851. ISSN: 1365-2966. DOI: 10.1111/j.1365-2966.2009.15715.x. URL: http://dx.doi.org/10.1111/j.1365-2966.2009.15715.x (cit. on p. 19).

Springel, Volker, Rüdiger Pakmor et al. (2020). *Simulating cosmic structure formation with the GADGET-4 code*. arXiv: 2010.03567 [astro-ph.IM] (cit. on pp. 2, 19, 41).

Springel, Volker, Simon D. M. White et al. (June 2005). 'Simulations of the formation, evolution and clustering of galaxies and quasars'. In: *Nature* 435.7042, pp. 629–636. ISSN: 1476-4687. DOI: 10.1038/nature03597. URL: http://dx.doi.org/10.1038/nature03597 (cit. on p. 19).

Springel, Volker, Naoki Yoshida and Simon D.M. White (Apr. 2001). 'GADGET: a code for collisionless and gasdynamical cosmological simulations'. In: *New Astronomy* 6.2, pp. 79–117. ISSN: 1384-1076. DOI: 10.1016/s1384-1076(01)00042-2. URL: http://dx.doi.org/10.1016/S1384-1076(01)00042-2 (cit. on p. 19).

Tulin, Sean and Hai-Bo Yu (Feb. 2018). 'Dark matter self-interactions and small scale structure'. In: *Physics Reports* 730, pp. 1–57. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2017.11.004. URL: http://dx.doi.org/10.1016/j.physrep.2017.11.004 (cit. on pp. 13–15).

Tulin, Sean, Hai-Bo Yu and Kathryn M. Zurek (June 2013). 'Beyond collisionless dark matter: Particle physics dynamics for dark matter halo structure'. In: *Phys. Rev. D* 87 (11), p. 115007. DOI: 10.1103/PhysRevD.87.115007. URL: https://link.aps.org/doi/10.1103/PhysRevD.87.115007 (cit. on p. 13).

Vogelsberger, Mark, Jesus Zavala and Abraham Loeb (May 2012). 'Subhaloes in self-interacting galactic dark matter haloes'. In: *Monthly Notices of the Royal Astronomical Society* 423.4, pp. 3740–3752. ISSN: 0035-8711. DOI: 10.1111/j.1365-2966.2012.21182.x. URL: http://dx.doi.org/10.1111/j.1365-2966.2012.21182.x (cit. on pp. 13, 15, 46, 48).

Wyrzykowski, Łukasz and Ilya Mandel (Apr. 2020). 'Constraining the masses of microlensing black holes and the mass gap with Gaia DR2'. In: *Astronomy & Astrophysics* 636, A20. ISSN: 1432-0746. DOI: 10.1051/0004-6361/201935842. URL: http://dx.doi.org/10.1051/0004-6361/201935842 (cit. on p. 6).

Yoshida, Naoki et al. (Dec. 2000). 'Weakly Self-interacting Dark Matter and the Structure of Dark Halos'. In: *The Astrophysical Journal* 544.2, pp. L87–L90. ISSN: 0004-637X. DOI: 10.1086/317306. URL: http://dx.doi.org/10.1086/317306 (cit. on p. 15).